JETS

The USENIX Journal of Election Technology and Systems

Volume 1, Number 1 • August 2013

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

JETS

The USENIX Journal of Election Technology and Systems

Volume 1, Number 1 • August 2013

JETS articles are presented at the Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE).
www.usenix.org/conferences/evtwote

ISBN 978-1-931971-04-1 ISSN 2328-2797

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

From Helios to Zeus

GEORGIOS TSOUKALAS, Greek Research and Education Network (GRNET)
KOSTAS PAPADIMITRIOU, Greek Research and Education Network (GRNET)
PANOS LOURIDAS*, Greek Research and Education Network (GRNET)
PANAYIOTIS TSANAKAS, National Technical University of Athens

We present Zeus, a verifiable internet ballot casting and counting system based on Helios, in which encrypted votes are posted eponymously to a server, then are anonymized via cryptographic mixing, and finally are decrypted using multiple trustee keys. Zeus refines the original Helios workflow to address a variety of practical issues, such as usability, parallelization, varying election types, and tallying through a separate computing system. In rough numbers, in the first nine months of deployment, Zeus has been used in 60 elections, tallying a total of more than 12000 votes.

1. INTRODUCTION

Helios [Adida 2008] (http://heliosvoting.org) is a well known system for internet voting, in which the entire process is carried out through digital means—there is no paper trace, nor ballots in physical form. Instead, there is a trail of mathematical proofs linked together that can verify the correctness of every step in the voting process. Helios has been used in several real world elections and its basic architectural design has proven robust.

In the summer of 2012 our organization was asked to provide a system for electronic voting to be used in elections in universities in Greece. Based on its good track record and the body of work that had been published on it, we decided to see whether Helios could fit our needs. Unfortunately, we found that it could not, because of the voting system that would be used in the elections we were called to run.

We therefore had the problem of creating a system that could be used for the required elections, scheduled to take place in approximately three months time. Having no time to start from scratch, and having no intention to re-invent the wheel or to embark on cryptography research, we set out to develop a system based on Helios, as much as possible, with the necessary modifications to suit our needs. The system came to be known as Zeus (thus remaining in the pantheon of Greek gods) and has been used with unequivocal success for many elections over several months now. That was especially satisfying since the use of Zeus was not without critics, for reasons that we will explain below.

The contribution of this paper is not any new electronic voting protocols or an entirely new voting workflow; we have adopted and used proven systems and ideas that have already been described in the literature. Our contribution is:

(1) An extension of an existing e-voting system, Helios, so as to be able to handle any kind of voting systems, such as Single Transferable Vote in a completely open source implementation.
(2) The organization of the Helios cryptographic workflow as a process of populating a single document with all relevant cryptographic content in a well-defined portable format, independent of any other details of the voting system.
(3) A new use of audit votes, both as a means of checking the integrity of voting from the user's side, and as a channel of authenticated communication with the user.
(4) A description of a working e-voting system that has been used in many real-world elections, by not especially technically-savvy users.
(5) An account of the experience of using an e-voting system in a particularly adversarial context, in over 60 real-world elections involving more than 12000 voters in total.
(6) A description of the usability, accessibility, and logistical issues we had to tackle.
(7) An outline of the internal design and implementation choices.

*Corresponding author.

Items 1–3 relate to technical differences with previous implementations of Helios and related systems. The other items relate to lessons from real world usage of the system in an especially challenging setting.

The rest of this paper proceeds as follows. In Section 2, we discuss related work. In Section 3 we discuss the birth of Zeus. In Section 4 we offer an overview of the voting process from the user's point of view. In Section 5 we go through the complete cryptographic workflow elaborating on technical details and the population of the *poll document*, which contains all data and proofs to verify the election. In Section 6 we relate our experience with the deployment and use of Zeus as a service for elections over the internet. We conclude with some general discussion points in Section 7.

2. RELATED WORK

Zeus is based on Helios [Adida 2008], an electronic voting system developed by Ben Adida. The original Helios publication [Adida 2008] proposed cryptographic mixing via Sako-Kilian [Sako and Kilian 1995] / Benaloh [Benaloh 2006] mixnets [Sako and Kilian 1995] of ballots for their anonymization. Mixnets were dropped in version 2 of Helios, which was used for elections at the Université Catholique de Louvain in 2008, because in the then-existing implementation they could not accommodate the need for votes having different weights according to the voter's category [Adida et al. 2009]. In place of mixnets homomorphic tallying [Cohen and Fischer 1985] was adopted, as it was argued that it was "easier to implement efficiently, and thus easier to verify" [Adida et al. 2009].

Homomorphic tallying operates directly on *encrypted* votes, tallying them all up in a single encrypted result, which is the only piece of information that is decrypted. This is much easier computationally but sacrifices generality because the tallying algorithm must be encoded in the cryptographic processing itself—one cannot get results in the form of individual ballots as they were submitted.

On the other hand, mixnets shuffle the set of votes in a provably correct but practically untraceable way, therefore anonymizing it. The output is the same set of unencrypted ballots as they were before encryption and submission, only in random, unknowable order. Then any method of tallying can be applied to compute results. Although Helios itself is not using mixnets any more, a variant of Helios with mixnets has been developed that does [Bulens et al. 2011]; however the code is not publicly available.

Due to the circumstances around the birth of Zeus (see Section 3) and the controversy around it (see Section 6) there was no way we could have used any implementation whose source was not completely open source and freely available to the community. Hence, even if after contacting the developers of Helios with mixnets we were given access to the code, we could still not have been able to use it unless it were completely open sourced. Judging this unlikely, we proceeded on our own, without contacting them. As a result, Zeus and Helios with mixnets share many conceptual similarities, although we did not try to avoid the submission of related ballots using the Cramer-Shoup cryptosystem, as such behaviour is not prohibited by law. Moreover, due to time constraints and ample hardware at our disposal, we had the luxury of not having to use more efficient mixnets than the Sako-Kilian model.

From the voter's point of view, there has been a published usability analysis of Helios [Karayumak et al. 2011]. From the elections we held we have gathered our own experience on usability, and in fact the user interface of Zeus is completely re-worked.

Over time researchers have explored possible vulnerabilities in different versions of Helios [Estehghari and Desmedt 2010; Heiderich et al. 2012]. To the best of our knowledge these attacks are not applicable to Zeus.

More recently, the question of endowing Helios with everlasting privacy has been explored [Demirel et al. 2012]. Helios offers computationally secure privacy, i.e., it is currently prohibitively expensive, from a computational point of view, to break voter anonymity. This does not preclude, however, that it will be practical in several years' time. Protocols for everlasting privacy can be used for both homomorphic and mixnet-based elections. Zeus, like Helios, relies on computational privacy and it would be possible to explore the adoption of an everlasting privacy approach.

Due to the switch from homomorphic tallying to mixnets and other requirements and refinements presented below, the Zeus software only retains 50% of the original Helios source code, including the entire homomorphic tallying support, which is unused.

3. THE BIRTH OF ZEUS

Zeus was initiated after the use of electronic voting was permitted by decree for the election of the Governing Councils of Higher Education Institutions in Greece. In each institution the Governing Council is directly elected by its faculty and is its main governing body. The election uses the Single Transferable Vote (STV) system, in which voters do not simply indicate the candidates of their preference, but also rank them in order of preference.

When we were charged with providing an implementation of a system implementing electronic voting we decided to investigate Helios's suitability, as we needed a mature system with a proven record in real world elections and published open source code. The current version of Helios (version 3) allows internet elections from end-to-end: from the moment the voter casts a ballot through a web browser to the publication of election results. It does that by never actually decrypting the ballots but performing a series of homomorphic calculations on them. In the end, the results of the calculations are decrypted and published.

Although using a single system for the whole process is appealing, the use of homomorphic tallying in Helios cannot accommodate voting systems in which not just the individual choices on the ballot matter, but the whole ballot itself. In STV, homomorphic tallying as currently implemented in Helios could pass to the STV algorithm the information that a certain candidate has been selected in rank r by n voters, but this is not enough, as the whole ballot and not just each rank separately is passed around during STV's counting rounds.

That does not mean that homomorphic tallying cannot be used with STV. It is indeed possible to do homomorphic STV using the Shuffle-Sum approach [Benaloh et al. 2009][1], although Helios does not provide this capability now.

At that point we realized that it is not necessary to use Helios's homomorphic tallying capabilities. We decided to use Helios for *counting the ballots*, not for producing the election results. Once we do have a verifiable ballots count, this can be fed to an STV calculator, or indeed to a calculator of any voting system. Since the ballots are published, and the algorithm is also published, a third party can always verify that the results are correct.

4. OVERVIEW OF THE VOTING PROCESS

Each poll in Zeus comprises the following phases: poll preparation, voting, processing, and tallying.

4.1. Poll Preparation

A Zeus account is created for the organizers of the poll. Logging into the system with this account enables creation of new polls through a submission form. The account holder, that is the *poll administrator*, sets the title, description, dates, and other information for the new poll, and chooses who the trustees are going to be. Next, they proceed to select the type of the election and define the content of the poll accordingly. Depending on the election type, the content can typically be either a single list of candidates to rank, or multiple party lists with candidates to select candidates from one, or multiple questions for each to be answered by choosing one or more of the provided answers; see Section 5.3 and Section 5.4 for details on the types of elections supported. Then, the administrator uploads the list of voters that may participate in the poll. A simple CSV format is used, easily exported from spreadsheets.

[1] We are grateful to one of the reviewers for pointing out this work to us. We have not worked out how to use Shuffle-Sum in the particular STV variant used in the elections we had to support, since in order to get elected, a candidate must pass the preferences quota (similar to the Droop quota) and not violate a quota of two elected candidates per school of the university. A candidate that would be elected but violates the per school quota is eliminated. At the end, if the STV iterations result in empty seats, eliminated candidates are brought back and are elected, provided the per school quota is not violated.

Before voting starts each of the trustees must visit Zeus's website and generate and register their encryption keys for the poll. The trustees do not need an account to enter the system. Instead, upon their designation as trustees, Zeus sends them a confidential invitation which they can use to log in to the system and perform their duties.

Thus far, trustees are designated by their e-mail address, and the invitation is sent via e-mail as a secret link to Zeus's website. However, it is easy for Zeus to be adapted to use any system for designation and notification of trustees. Indeed, an adaptation for notifications via mobile phones was actually implemented but has not been used or streamlined within the software yet.

4.1.1. Trustee Key Generation. Trustees must follow their invitation and log in to generate their key pair. The generation takes place entirely in the web browser, assisted with randomness downloaded from the server itself, due to poor randomness in browsers, a practice inherited from Helios. After key creation, the trustee is prompted to save their private key. Once this is done, the trustee is automatically logged out and the browser session ended. Zeus then sends to the trustee a new invitation to verify that they do indeed have their private key. The trustee, following the link, starts a new browser session, in which they are shown the hash value of their public key, and are asked to locate the private key that they have saved. The browser verifies the key locally by comparing its hash value to the one registered by the server. The whole process is roughly equivalent to having to type a password twice, since it is important that the trustees have indeed saved their keys, or decryption will not be possible.

4.1.2. Finalizing the Poll, and Inviting Voters. The administrators may edit all aspects of the poll, up until they decide to *finalize* (or *freeze*) the poll. After that, only limited editing is possible, and most importantly, all registered voters are sent their confidential invitations to vote.

As with trustees, voters do not need an account, but use the secret information in their invitation. Visiting Zeus through the invitation link will initially display information about the poll along with contact details and voting dates. Once the appointed voting time has come, the voting booth is automatically enabled and visiting voters are transferred there directly.

We decided to embed the login credentials in the confidentatial invitation URL for usability and logistical reasons: the users would not have to keep track of a personal account in Zeus, and we would not have to deal with users forgetting their passwords etc. This of course requires that users have access to their e-mails; if they do not have access to the link, they cannot vote. As we report in Section 6.1 this was used as an attack vector, but was dealt with by extending the duration of the attacked polls and adding the capability to send the links by SMS messages. A remaining worry is that a malicious Zeus server, knowing each voter when they come to vote before they cast their ballot, could target them with malicious code. This could be mitigated by requiring the user to submit with their vote audit codes received by a separate channel, like SMS, although this is not immune to attacks (see Section 5.5).

4.2. Voting

The voting booth, once opened, operates entirely locally without further interactions with the Zeus server or any other site. The voter may even disconnect from the internet while inside the booth. The booth appearance depends on the election type, and after vote selection users are asked to review and confirm their choices. After vote submission, a receipt (see Section 5.2) is generated which can immediately be downloaded, but which is also sent to them via e-mail.

During voting administrators have access to information about the ongoing election. The information includes the number of voters that have voted, and for each voter their e-mail, name, whether they have voted, when was the latest e-mail sent to them and the last time the visited the voting booth.

Section 5.1.2 elaborates on the technical issues.

4.3. Processing and Tallying

After the appointed end time, voting is automatically disabled. However, administrators may choose to extend the voting to a new end time and date, even after expiration. For the poll to be finally closed,

the administrator has to explicitly select so. Then, the first mixing by Zeus proceeds automatically. Once the first mix is complete, additional mixes by external agents may be added one by one using generated confidential links. Zeus provides a command line toolkit that, among other things, can perform mixing given these confidential links.

Finally, the administrator ends the mixing process, and the trustees are notified again for the decryption process. Each one visits the Zeus website through the invitation link they have just received. The mixed encrypted ballots are downloaded to their client browser, and they are prompted to present their secret poll key. When this is provided, the ballots are partially decrypted and the output is sent back to Zeus.

When all partial decryptions are available, they are combined to produce the final anonymous and unencrypted ballots. Depending on the poll type, the ballots are either sent into another system for tallying and reporting the results (if the tallying and results presentation algorithm is implemented externally, as was the case in the majority of elections we have run), or the ballots are tallied and reported on by Zeus itself. After the results are available, the administrator may download an archive with all generated proofs.

5. CRYPTOGRAPHIC WORKFLOW

Zeus uses essentially the same cryptographic workflow as in Helios [Adida 2008]. In this section we contribute the organization of the Helios workflow around a logical document that contains every cryptographically significant piece of data for a poll, which can be stored in a portable textual canonical form.

The handling of this *poll document* has been implemented in a generic and standalone software module, which is then extended by the web application and the user interface to create a usable voting system.

This core module implements all the required cryptographic functions and validations. The web application extends the abstract workflow module and builds a usable voting system upon a database that handles authentication, election formulation, vote submission, and trustee interactions for key setup and decryption. The user interface offers access to the web application through a web browser, and locally performs vote preparation and encryption, as well as trustee key generation and (partial) ballot decryption.

The core module handling the poll document is completely independent of the other aspects of the system, and therefore can provide much more accessible and reliable verification by third parties, than if they had to dig through irrelevant software layers to validate results. Ideally, users of Zeus should have an API which they could use to query the poll document at will. Zeus does not provide such an API right now, but it is clearly something worth implementing.

From the Zeus point of view, it does not matter where or how the poll document is created, one can inspect and verify the actual process that took place.

5.1. Poll Stages

Zeus represents each poll as a structured document and defines five consecutive stages for it, from creation to completion. At each stage more data are recorded into the document, either coming from the outside, such as votes, or as results of processing existing data, such as vote mixing, or both, such as final ballots decryption.

Once recorded in the poll document, data are never modified, and each stage is validated upon transition into the next one.

At the final stage, the document represents a full account of a completed poll, containing everything needed to verify the process, from votes, to results, to proofs.

5.1.1. Creating. At this stage, candidates, voters, and trustees are recorded into the document. The candidate list is just an ordered list of unique candidate names, which are otherwise not important and are not interpreted. However, different types of elections put special structure within the candidate list, which is used at tallying time to verify and count the ballot, as discussed in Section 5.4.

Voters are likewise represented by uninterpreted but unique names. For each voter, Zeus generates several random numbers, the *voter codes*, that can independently be used both for audit votes and voter authentication, as discussed in Section 5.5.

For trustees, only their public key is recorded. For each poll, and additionally to any external trustees, Zeus always creates and registers a new trustee key for each poll. As per the Helios workflow, this allows the service operator to provide anonymity guarantees in addition to those provided by the appointed committee of trustees.

5.1.2. Voting. When all intended trustees have been recorded, the poll may transition to the *voting* stage by combining (by a multiplication) the trustees' public keys into a single *poll public key* that the voters must use to encrypt their votes. After the formulation of the poll public key, no trustees, voters, or candidates can be modified.

The voter submits to the server the encrypted ballot, (discrete log knowledge) proof that they possess the randomness the ballot was encrypted with and optionally a voter-provided audit code, to verify that their local system really does submit the vote they choose, and that it does not alter it in any way. The process is detailed in Section 5.5.

For each submitted vote, Zeus generates a cryptographically signed receipt for the voter. With the receipt the voter can later prove that his vote submission was acknowledged as successful, perhaps in the context of an investigation for a complaint. Receipts are detailed in Section 5.2.

Each voter may submit a vote multiple times. Each time, the new vote replaces the old one, and the new receipt explicitly states which vote it replaces. No vote can be replaced more than once.

5.1.3. Mixing. After voting has ended, the poll is put into the *mixing* stage, where the encrypted ballots are anonymized. Only the ballots eligible for counting are selected, by excluding all audit votes, all replaced votes, and all votes by excluded voters.

Voter exclusion is a digital convenience owning to the voter's identity still being known after voting has ended. Anonymity is achieved at the end of the mixing stage, and not at cast time as in traditional ballot box polls. The poll's officials may choose to disqualify voters if it is discovered that they were wrongly allowed to vote, or if their conduct during the election was found in breach of rules. Neither the disqualified voter nor their votes are deleted from the poll document. Instead, the voter is added into an exclusion list along with a statement justifying the decision to exclude them.

The eligible encrypted votes are first mixed by Zeus itself by a Sako-Kilian mixnet. The bulk of the computational work in mixing is to produce enough rounds of the probabilistic proof. The work is distributed to a group of parallel workers, one round at a time. Our deployment used 128 rounds and 16 workers. Verification of mixes is likewise parallel.

After the first mix is complete, additional mixes by external agents may be verified and added in a mix list using Zeus's command line toolkit.

5.1.4. Decrypting. After mixing is complete, the final mixed ballots are exported to the trustees to be partially decrypted. The resulting decryption factors are then verified and imported into the poll document. Zeus's own decryption factors are computed last.

5.1.5. Finished. In the transition to *finished*, all decryption factors are distributed in parallel workers to be verified and combined together to yield the final decrypted ballots. The decrypted ballots are recorded into the poll document, and then the document is cast into a canonical representation in text. This textual representation is hashed to obtain a cryptographically secure identifier for it, which can be published and recorded in the proceedings.

5.2. Vote Submission Receipts

For each vote cast a receipt is generated and returned to the voter, as well as registered in the poll document. The receipt is a text file with simple structure listing cryptographic information about the vote and poll. The receipt text is signed with the Zeus's own trustee key for the poll, which is possible because Zeus is a trustee for every poll as described in Section 5.1.1. The ElGamal signature is according to [Schneier 1995].

The receipt contains a unique identification of the cast vote, the vote (if any) which it replaces, the cryptosystem used, the poll's public key along with the public keys of all trustees, the list of candidates, the cryptographic proof that it was a valid encryption, and the signature itself. The validity of the encryption is established by checking the discrete log proof submitted along with the encrypted ballot.

Note that in the current version of Zeus we do not exclude the submission of related ciphertexts; however this was not a major problem for us, as in real elections it is not forbidden for candidates in a party list to give their voters prepared ballots which they can use to vote.

A related problem, indicated by one of the anonymous reviewers, is that "a voter who would like to compromise the privacy of another voter can simply copy the ciphertext of that voter, maybe modify it a bit in a controlled way using the homomorphic property of the encryption scheme, and then look for duplicate ballots (or for a pair of ballot with a difference matching the modification made on the ciphertext) after decryption". It was easy to guard against that, and it was an oversight that we did not do it from the outset. To calm our worries, we checked all past elections and verified that no such chicanery occurred. We subsequently addressed this for any submited ciphertext (g^r, c_2) by including c_2 in the hash that produces the challenge for the proof of knowledge of r. This way, the proof of knowlegde is bound to the specific ciphertext and therefore is not reusable.

Although private information such as user name, IPs, and times are logged in the Zeus servers, they do not appear in the receipt, because we need the entire poll document to be publishable if the election authorities decides so.

The vote submission receipts are not enough to guarantee that a malicious Zeus server cannot submit a vote on behalf of the user, and then argue that the user has simply lost their receipt; or that a user, taking care to delete their receipt, does not come forward and argues that their last vote on the server is a bogus one. The problem could be solved by posting encrypted ballots in a bulletin board after the polls close and before tallying starts. However this was not possible in the elections we had to support, since it would add a step to the users' sequence of actions, and a delay in the production of results; the speedy announcement of results was of major importance, especially as there were days where more than one election was taking place in different institutions.

5.3. Ranked Ballot Encoding

In Helios, ballots consist of answers to binary "yes" or "no" questions, framed in the appropriate way. For instance, a voter indicates k out of n candidates on a ballot by selecting them, in which case the answer "Would you like X as a Y?" gets a yes (1), otherwise a no (0). In STV, as in any system in which we would need the whole ballot to be decrypted in the end, the ballot must be encoded in some way. We encode each ballot as a integer by assigning a unique number to each possible candidate selection and ranking. When enumerating, we first count the smaller selections (i.e. take *zero* candidates, then *one*, then *two*, ...) so that for small numbers of choices the encoded ballot will have a small value, thus saving valuable bit-space.[2]

5.4. Tallying Multi-Question Approval Polls

In Section 5.3 we have established the encoding used for ballots that select a subset of the candidates and ranks each one in it, in a strict order.

Multi-question approval ballots, however, are different. There is a number of questions separated into distinct groups, and each question has a number of possible answers. Each question can demand a minimum and a maximum number of answers to be selected for a ballot to be valid. The minimum can be zero, which means a ballot may select a question without selecting any of its answers. This is useful for party list elections, where you can vote for a party list without having to vote for any of the candidates in it. Only questions from the same group can be selected in the same ballot. It is not allowed to select an answer without selecting the question it belongs to. Selecting no answer at all creates an empty ballot and a blank vote.

[2]For example, selecting up to all of 300 candidates needs 2043 bits, while selecting 10 out of 1000 candidates needs only 100 bits.

In order to encode this into an ElGamal plaintext we had two options; either create a new encoding for multi-question approval ballots, or translate the approval ballots into ranked ballots and use the existing encoding. Due to time constraints we used approval-to-ranking translation for its simplicity and malleability, a solution that proved to be very practical.

The translation works mainly by structuring the candidate names. Each candidate name has two parts, the question and the answer. All answers to the same question are required to occur together on the candidate list. The first candidate of a question is special, and instead of an answer, the second part of its name contains configuration information such as the minimum and maximum answers, and the group it belongs to. Selecting a questions is signified by selecting this candidate entry. Question groups are identified by consecutive integers starting with 0.

Given this structure, each approval ballot can be represented by correct choices among those entries, i.e. those that respect grouping, minimum-maximum answers, etc. The selections are also required to appear in a strictly ascending order within the ranked ballots.

Note that this embedding of approval ballots within ranked ones is only relevant at ballot preparation and ballot counting. Between those two stages they are treated as ranked ballots. The user interface will never produce an invalid ballot, but Zeus will always validate them at counting time. An invalid ballot is an alert, because it means there was either an error in encoding, or someone has voted incorrectly without using the official web interface. Voting non-interactively via Zeus's programmatic web interface should not pose any problem in itself, but because the voter might get it wrong, or might be executing an attack, it is prudent to look into such incidents.

5.5. Audit codes

Audit codes are random numbers generated by Zeus and sent to each voter. The purpose of these codes is dual. First, they can be used to submit audit votes to test that the local system, mainly the web browser, faithfully submits the votes it is being instructed to. Secondly, and optionally, they can be used to authenticate voters in close integration with audit votes.

Audit votes work by ruining the plans of an attacker who might otherwise compromise the voter's computer and alter the voter's selections. To ruin those plans one introduces uncertainty; either the vote will be submitted normally and be counted, or the vote will be revealed and published for all to see what it was. If the attacker does not know what will be the fate of the vote, they cannot decide whether to cheat or not. If they cheat and the vote is published, they will be exposed. If they don't and the vote is counted, they will fail.

Audit votes in Helios create this uncertainty by having the voter decide on the fate of the vote, only *after* it has been uploaded to the server and therefore is made immune to the local attacker. After submission of each vote, the voter is prompted to choose between a normal vote and an audit one.

This is explicit and safe, but for Zeus it was deemed too dangerous to *require* all voters to make such a decision, because it would require of all voters to understand what an audit vote does, while a wrong button press, even accidentally, would cause the true intended vote to be published.

In Zeus, a vote can be submitted either with an accompanying code, or with no code at all. Submissions without an accompanying code are accepted as normal votes to be counted and no auditing is performed. If a voter distrusts their local system and wishes to verify it and submit their vote safely, then they should never submit a vote without a code.

If a vote is submitted with an accompanying code, two things may happen. If the code is *not* among the ones the voter has received, then the server considers it to be an audit. If the code *is* among those supplied by the server, then it is cast as a normal vote. Two different votes must never be submitted with the same code, because the second time the attacker will know what to do. The process in detail is as follows:

(1) The voter enters one of the audit codes they have received via e-mail.
(2) The voter proceeds to submit the vote to the server.
(3) The server receives the encrypted vote. If the encrypted vote is accompanied by a correct audit code or is not accompanied by any audit code, the vote is treated as a normal vote. Otherwise,

that is, if the vote is accompanied by an audit code but the audit code is not correct, we continue with the following steps.

(4) The system stores the vote ciphertext as an audit vote request.
(5) The system asks the voter to confirm that they have cast an audit vote.
(6) If the voter responds negatively, the audit process is canceled and the voter is redirected to the initial ballot casting view. The voting request remains in the system but is not displayed to the users.
(7) If the voter responds in the affirmative, the browser reposts the vote ciphertext along with the ElGamal randomness with which the vote has been encrypted.
(8) The system tries to match the fingerprint of the re-posted vote with that of an existing vote audit request. Failure to do so implies that the ciphertext initially posted is mangled and the system informs the user that the audit request cannot be accepted. This guards against malware that would submit an altered vote and then after realising that the vote is an audit vote would send the actual ciphertext. If the match succeeds the system stores the audit vote and returns to the voter a unique identifier of the audit vote.
(9) The decrypted contents of each audit vote, based on the randomness sent previously by the client, are posted on an audit vote bulletin board. The voter can go to the audit bulletin board and look for the vote using the identifying number they have received from the server.

The user interface of Zeus makes this process intentionally somewhat obscure to the layman, so that its use without being certain what to do is discouraged, but it is intended to be easy enough for those who do understand: vote with a random "wrong" code many times and check the audits, then vote with a "correct" and be done.

Therefore, Zeus makes auditing optional, by creating the required uncertainty for auditing using the audit codes. However, Zeus can be configured to *require* a code with each vote submission in which case the audit codes provide a secondary function as an additional authentication factor. Making the audit code compulsory would also strengthen the system against an attack in which a compromised browser notices when a ballot is submitted without an audit code and so will cheat only then.

This function is very useful because it helps combine a very convenient but potentially insecure means, for example the e-mail, with a quite secure and personalized one, for example the mobile phone. Invitations, notifications, and descriptions relating to voting, which can be voluminous texts with graphics, can be sent by e-mail over the internet, while the audit codes, which are conveniently small as a text, can be sent to a mobile phone. The voter would need both the e-mail and the mobile phone message in order to vote successfully.

Although the use of audit codes improves confidence in the system, it does not make it invulnerable to attacks. Such an attack would be by a malware that scans the voter's email for emails coming from Zeus and getting the codes from there. Then the malware could know when the user is submitting a normal vs an audit code, and behave accordingly. The risk could be reduced by having the audit codes received from a second channel, e.g., SMS-messages, although the SMS channel could also be attacked, as shown in possible attacks described against the Norwegian system [Koenig et al. 2013].

5.6. Performance

Mixing the votes dominates the total processing time and exponentiation dominates the mixing calculations. Zeus is written in Python. Python as a programming language has native support for arbitrarily sized integers and exponentiation. It is quite fast but we have used a high performing implementation from `libgmp` via the `gmpy` python module. This sped our exponentiations by about 7 times.

Still, the computational cost of mixing is so high that it only became practical when we parallelized it, which was an algorithmically straightforward undertaking. Hardware was not a problem for us (we are a cloud services provider, among other things), so we could deal with the cost in the old-fashioned way of throwing hardware at the problem. Had we more time in our disposal, we

could have explored more efficient mixnet implementations [Bayer and Groth 2012; Terelius and Wikström 2010; Wikström 2009].

Zeus is by default a 2048-bit cryptosystem with a 2047-bit order. Using 16 cores at 2.26 GHz it can mix 10000 votes for 128 rounds, a total of 12.8 million ciphers, in about 65 minutes, producing about 2 billion bytes of on-disk proof text data, compressible to about half of it, since numbers are in a hexadecimal canonical form and thus 2 bytes of text correspond to 1 byte of raw number data. Although not negligible, this load has allowed us not to allocate more hardware so far, our biggest election tallying about 1600 votes.

6. EXPERIENCE

At the point of this writing (June 2013), Zeus has been used in 60 elections around Greece. Figure 1 shows the number of registered voters and the turnout in each election. The registered voters in all elections totaled 15599, while the voters that actually took part in the elections totaled 12171; Elections were carried out successfully in all planned institutions. This was a significant success, as the stakes were particularly high and the voting process charged with emotions. To understand that it is necessary that we provide some surrounding context.

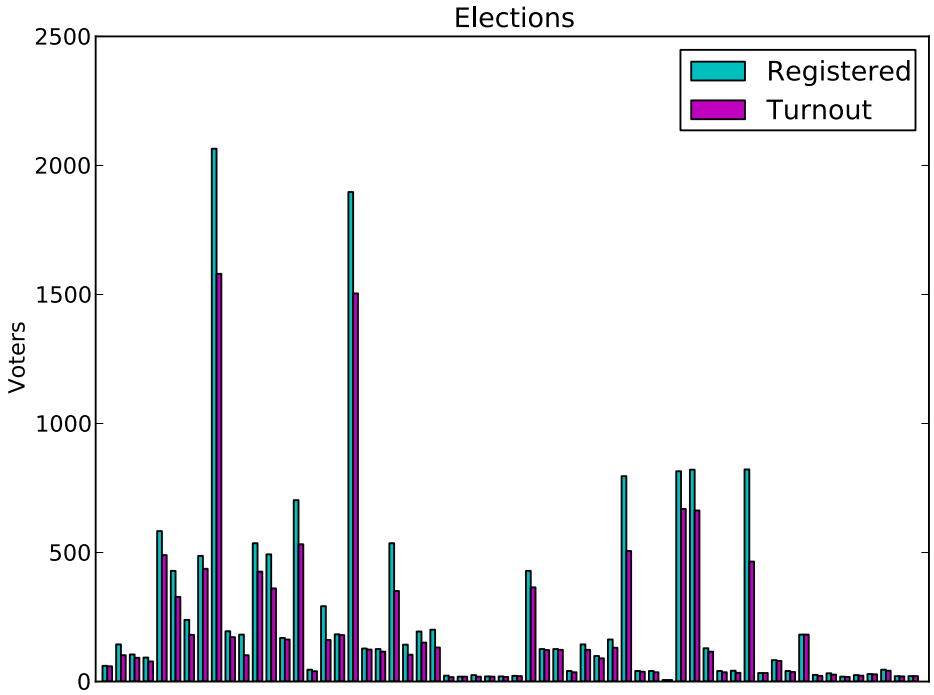


Fig. 1. Zeus held elections to June 2013.

The law that instituted the Governing Councils in Greek educational institutions met resistance from some political parties, student and academics unions. A widespread means of protest was to disrupt the election of Governing Councils, so that they could not be selected. Successive aborted

elections pushed forward the adoption of electronic voting, which then in brought Zeus into the controversy.

In particular, the controversy meant that some political parties were against the use of Zeus, or any form of electronic voting. As a result, Zeus was the subject of a number of public denouncements and letters, detailing weaknesses of electronic voting systems in general and Helios in particular. These did not challenge Zeus, as the weaknesses applied to older versions of Helios. Although Zeus was open sourced from the start and we welcomed constructive criticism or the exposure of unknown vulnerabilities, we did not receive any. The atmosphere was at times vitriolic, as when a mainstream political party called the people behind Zeus "techno-fascists", thus introducing a new term in the greek lexicon and bemusing the people involved in the whole effort.

This led to attacks on Zeus, and the voting process supported by Zeus. None of them was eventually successful. We describe them briefly in Section 6.1. A discussion on other security-related aspects follows in Section 6.2. We round up with a presentation of voting logistics in Section 6.3.

6.1. Attacks

6.1.1. University of Thrace. The elections at the University of Thrace had been planned for October 24, 2012. Due to a coding bug, the polls opened the at the time the election was frozen, on October 22. This was discovered by some voters, who went on to vote before the official poll opening. The issue was publicized and the election was annulled and repeated at a later date. Technically, the election could have proceeded without a problem, since there was no way anybody could have tampered with the election results, but the issue was sensitive and politicized, so that the election committee took the most cautious route.

Elections were called again for October 29, 2012, with polls opening at 9:00. At dawn our Networks Operations Center noticed a large number of connections to Zeus. Upon investigation, it was found that Zeus was the subject of a slowloris attack [ha.ckers.org 2013]. The attach was swiftly dealt with, and apart from some initial inconvenience did not cause any real problems. Moreover, the attackers used static IP addresses and could easily be traced and blocked.

6.1.2. University of the Aegean. The University of the Aegean had planned its elections for November 2, 2012. After freezing the election, and when the notifications to the voters had been sent, voters started receiving additional notifications containing bogus URLs from a cracked university server. Although the bogus URLs were not functional, a number of users were confused. The electoral committee decided to cancel the elections and call new elections at a later date. To avoid similar situations in coming elections we recommended the use of Sender Policy Framework (SPF) [Wong and Schlitt 2006] to institutions that had not been using it.

The repeat elections were held on November 9, 2012. The authorities at the university had authorized the set up of a new mail server, to be used solely for the elections, in which all voters were opened accounts. The mail server was hosted outside the university's premises. This avoided a problem that would have risen from a sit-in at the building housing the main mail server, which had been switched off by protesters at election day.

6.1.3. Agricultural University of Athens. On election day, November 5, 2012, protesters staged a sit-in at the mail server building of the Agricultural University of Athens. The mail server was switched off, cutting off access to e-mail, and depriving voters that had not downloaded the access link from voting. The university authorities responded by asking voters to come physically to a location outside the campus, where they could be issued with a new URL.

6.1.4. University of Patras. The University of Patras held its elections on November 7, 2012. Just prior to the opening of the polls a slowloris attack was noticed and dealt with.

6.1.5. University of Athens. A little while after polls opened at the University of Athens, on November 12, 2012, protesters staged a sit-in at the university's Network Operations Server and cut off internet access. This took offline mail access to members of the university and, like in the Agricultural University of Athens, deprived voters that had not downloaded the access link from

voting. More dramatically, the internet shutdown affected all sorts of university services, including internet connectivity to university hospitals.

The university responded by extending polling by three days. At the same time, an alternative voter's notification scheme was implemented, via which the voters would receive instructions for voting via SMS. The sit-in was lifted on the last election day, as it was realized that the elections would go on regardless.

6.1.6. National Technical University of Athens. A sit-in similar to the one at the University of Athens was planned at the National Technical University of Athens. However, it did not last long, as the SMS-based infrastructure was already in place, so that it would be irrelevant, and the university authorities made it clear that they would not tolerate the disruption. The election took place without problems on December 10, 2012.

A different, more insidious kind of attack was revealed later on. A few days before the election, one voter had contacted us complaining that the link he had received did not seem to be functioning at all (it should link to a page notifying the user about the coming elections; instead, the user was redirected to the main Zeus page). Two days before the election it was discovered that the user's e-mail stored in Zeus had been changed, and a new e-mail had been sent to the newly entered mail address. The user complained that he never used that address, so another URL was issued and sent to his institutional mail address. The voter voted without a problem on election day.

A few days afterwards a protesters' site announced they had circumvented Zeus security by taking hold of a voter's URL after contacting the election committee and asking them to update the voter's contact details. That explained the e-mail change, and it meant that it was the result of a social engineering attack. Luckily, it had been caught in time—something the protesters did not let on.

6.1.7. Recreate Greece. Encouraged by the success of Zeus in previous elections, a new political party, "Recreate Greece" ("dimiourgia, xana!" in Greek) approached us and enquired whether they could use Zeus for the elections leading to, and during, their party congress. Although not among our target constituency, we decided to allow the use of Zeus in a "best-effort" basis from our part, since that entailed the development of new, interesting functionality—specifically, party list elections, elections with multiple questions and answers, and elections where blank votes are not allowed.

It turned out that in one of their elections a voter wanted to show that e-voting cannot be trusted since anybody can vote with a voter's access credentials, if these are leaked. On April 6, 2013, the voter posted his voter URL on a Facebook page. The electoral committee noted the incident, and therefore moved to disqualify the ballots cast under that voter's name.

6.2. General Security Issues

As explained above, Zeus was born in controversy, and its use was accompanied by much nay-saying. An important argument against the system was that we were not to be trusted. After all, to the voters, as well as to each election committee, Zeus is a black box running in some remote virtual machine. It could be that we were really running a show: instead of Zeus, voters were interacting with something else, and we were putting on a pretense of anonymity and security.

As we were not aware of a practical method to verify the integrity of a running virtual machine, doubters were presented with two choices: either they would have to take us on our word that we were indeed honest, or they could download the code and run it themselves. Nobody took the second option.

Still, at several times throughout different elections we became aware that people did not really believe us in our assertions that we were honest and that the system behaved as we had argued it would. In two distinct occasions members of the election committee discovered that their election key could not be read by Zeus—a bad file. We had instructed them to keep two copies on two different USB sticks. Luckily, in both instances the back-ups were readable. People, though, were to various degrees sure that somehow we would be able to "do something" about it, disregarding our assurances that the only thing we could really do would be to re-run the elections from the beginning.

A similar veil of doubt appeared when people called us confidentially during elections, and shortly after the ballot box had been closed, to ask us "how things were going", and whether we had an indication of the results. We were at pains to explain that there is no way we could have such knowledge.

Doubt in whether Zeus does indeed what is purported to do manifested itself also in a request we received twice to provide to the election committee a log of the exact times each voter had cast a ballot. We explained to the election committee that if the issue at hand was to verify that all votes had been counted correctly, the correct procedure was other, and that the requested information could in fact be damaging in enabling fine-tuned attacks on privacy (an attacher would check that a voter who had been coerced to vote did not vote again afterwards).

Zeus gives to the election committee a full set of cryptographic proofs that can be used to verify the results. We had stressed that these proofs would be the sole arbitrer of any dispute, and the only material that could be really trusted. At some point in the elections, though, we decided to provide, just for convenience, a PDF summary output of the results. We found out to our chagrin that a plain, unsigned PDF document carries more weight than any of our pronouncements.

In particular, we used ReportLab's open source libraries for PDF creation (http://www.reportlab.com/software/opensource/). In one election (elections for president of the Piraeus Technical Educational Institute, held on March 26, 2013), a disgruntled candidate complained to the election committee that the PDF results' creation date was before the ballot closing time. It turned out that he was right, and we found a bug with the PDF libraries (which we reported to ReportLab). The fact that we never meant the PDF document to be the definitive results record of an election seemed to matter little. To resolve the issue we issued an official explanation, complete with details of the bug and possible fixes. We had thought the matter settled, but the candidate came back to us about 20 days later saying that when he had communicated directly with ReportLab he had been told that the problem we had identified could not explain the actual time difference he saw in the PDF. It turned out that he was right again, and the actual source of the error was that the creation timestamp was memoized; since Zeus uses long running processes, the timestamp was stored with its initial value and re-used by all documents that were created by the same process. Again we had to issue an official explanation with reference to the problematic lines of code. The authority of the PDF document is something we came up against repeatedly: people did not seem to care for any other kind of authority apart from that given by the printed matter. Perhaps we should have followed the example of [Adida et al. 2009] where *signed* PDF documents were an integral part of the process: since they somehow assumed that role anyhow, we had better invest them with more substantial authority.

Regarding usability, we did not receive any particular complains from the voters, except for one: that the system would not work with Internet Explorer. The users were informed of that in the notifications they received, but apparently not all of them noticed it, or knew exactly how to download and install one of the supported browsers (recent editions of Firefox, Chrome). The problem was pointed out to them when they contacted the respective election committee about their difficulties with and was thereby resolved; we did receive an angry e-mail, though, from a voter who felt snubbed that she had to go through such an ordeal. Be it as it may, we could extend the system to support recent versions of Internet Explorer, perhaps for the voters if not for the election committee. At the other end of the spectrum we saw in an interview a mention of a voter who had voted in an airplane and was very happy for it.

Voters were not expected to be knowledgeable about cryptography, and we took care to smooth the use of Zeus for them. A lack of understanding led to a few interesting interactions, with some voters trying to read and understand their voting receipts, being simple text files—although the e-mail attachments that contained them did not have a text suffix in order to discourage careless handling of them. They then inquired why they could not understand the receipt contents, and how they could be sure that it was a valid one. This is something that could be solved in time with better informational material and voter education on the basic premises of the system.

Taking into account the feedback we had from our users, it came as no surprise that nobody bothered to use the audit vote feature, whereby a voter tests that their browser is not compromised.

A few asked about it, and when they understood what it was about decided it was not worth the trouble. In truth, the voters that asked about it found it confusing, and we had to visually downplay the related user interface elements in order not to confuse the electorate. In more than one election the election committee asked us if it would be possible to remove any reference to audit votes from the user interface, as well as the audit codes from the notifications sent to voters. We declined; we should probably have put more effort into educating voters and recruiting champions among the voting institutions.

In the same vein, nobody bothered to use our support for external mixnets, although we did explain to those who asked that this was the only way to guarantee anonymity if they could not trust us, otherwise a malicious Zeus server could break the anonymity of all the users. We understood there were two reasons for their reluctance: they were wary of adding any complexity to the voting process, even though this complexity would not be reflected to the voters but only to the election administrator and the election committee. In addition to that, their typical view was that they trusted us that we would do our job properly and take every step possible to safeguard the elections, and that we would not want to shoot ourselves in the foot and compromise the elections ourselves. An important factor may have been that some people in the various election committees knew us personally. Since a flattering view does not really solve the underlying risks, we have been investigating whether it would be possible to have an independent third party that would run an additional mixnet: in this way users would not be inconvenienced and they would no longer need to put their trust in us. For example, Bulens et al. [Bulens et al. 2011] have suggested having a set of servers offering mixing services around the world that could be used by election organizers depending on availability.

Finally, Zeus, when used to run STV elections, suffers from the Italian attack: it is easy to "mark" ballots, by asking voters to put unlikely candidates low down in their preferences, and to check that such ballots appear in the results. An incisive academic that voted in an election in a relatively small institution (so the risks of identification were higher) noted the danger and asked us about it, even though he had had no previous experience with STV, that particular attack, or e-voting in general. He suggested that we could devise a mechanism by which only the part of each ballot that had actually been used in the STV rounds would be displayed in the final results.

6.3. Logistics

Zeus is offered as a hosted service; hosting here entails more than just a well-endowed virtual machine. Elections are a sensitive matter, since people exercise their basic democratic right. They must feel secure, and they must feel they are not slighted by their lack of technical knowledge.

In every Zeus election we have a helpdesk of three persons involved part time (and not all of them concurrently) before the election starts, during the election itself, and after the ballot box closes.

Before the election started the helpdesk is tasked with helping the election committee setting up the election. Although we have produced manuals, for both the election committee and voters, people frequently requested more engagement from our part. As an aside, we should point out that RTFM (Read the F* Manual) rules, as it usually does, and the more so the more technical a user is. We found that users with no particular computing experience were more careful in their use of the system and more conscientious in their use of the manuals provided, than users who, emboldened by their prowess in computing, went on without bothering to consult the documentation.

Our helpdesk colleagues help the election committee to generate their keys and to set up the voters list, ensuring that the input file, a simple CSV document, was well formed—not a trivial matter, as it turned out, as despite our detailed instructions college registrars often failed to produce an up to date list of eligible voters with their current e-mails, or to create a spreadsheet with the required columns. Once the e-mails containing the voters links are sent the helpdesk handles bounces. The bounces are forwarded to the election committee with the voter links removed, so that the election committee can contact the voters and correct their e-mail addresses but cannot abuse the system and start voting in their stead.

It could be possible, in this setting, for the administrators of Zeus not to forward the bounces and to therefore compromise the system by doing the voting themselves. This, though, would require of

us to know in advance which voters would not bother to contact the election committee inquiring why they have not received their voter links, at which point the election committee would discover that somebody else has been voting for them. Even if we were crooks we would not run a risk as high as this one.

During the election the helpdesk guides the election committee when they are in doubt about how to correct a voter's details. This happens when a voter contacts the election committee asking them to change their e-mail address to another one. Then, at the end of each election process, the helpdesk is sometimes asked to show the steps for the mixing and decryption process, as well as the presentation of the actual results.

In general, even though the helpdesk workload is not particularly high, it has special time requirements, such as working out of normal business hours to ensure that the communication with the election committee and the voters is as prompt as possible. Both election committees and voters can get very anxious when things are not working in the way they may have anticipated.

7. DISCUSSION

Zeus was a success beyond our expectations. It carried out all the elections we were called to run by law. People must have been reasonably happy with it, since following our official mandate we were asked, and we continue to be asked, to help organize other elections around the country: appointment of representatives for university teachers' union congress, various elections for the selection of rectors, deans, and department presidents, congress election for a political party. We understand that the practical advantages of e-voting outweighs any reservations the corresponding constituencies may have. In a telling interchange, a member of an election committee expressed his gratitude in high spirits, mentioning that in previous elections they would have to spend the night counting, re-counting, and dealing with possible complaints—while with Zeus they packed up early in the evening.

We are continuing to maintain and improve Zeus in various ways, mainly in allowing more kinds of elections with different ballot configurations and improving usability based on the feedback we receive. Zeus mixing is embarrassingly parallel, so we can easily scale it up to bigger elections; decryption from the part of the election committee however takes place on a local computer and may be limited by the resources there. We are exploring way in which we could improve performance, for instance, by developing and distributing a native application.

As explained above, Zeus was controversial, its use was actively opposed, and in the initial set of elections that we were mandated by law to support there were calls to boycott the whole process by abstaining. We want to stress that we had no part in the political debate. We dealt with Zeus as a purely technical problem of implementing a system that would allow elections to take place even when voting in physical ballot boxes was unfeasible. In the end, it came out that people wanted to vote. In these first election batch, which comprised 23 elections, the average turnout percentage was 80.76%, widely accepted as being very high. It was no small relief to know that we did not seem to be doing something against the will of the people.

That points also to a wider application area of e-voting. E-voting is often presented as a adjunct to traditional voting, and no substitute of it. We saw that it can be used as an effective substitute when the conditions on the ground prohibit normal elections. It is our belief, though, that for this to happen any e-voting system must be based in compete openness, the opaqueness of many existing solutions being one of the main arguments against wider adoption of electronic elections [Simons and Jones 2012; Jones and Simons 2012]. If Zeus had not been open source, we could not have been able to defend its use against protesters and detractors. We tried to be as open as possible. The complete Zeus code was available from the beginning of the project, not just released versions, but the whole repository, complete with all our commits and activity. So, in the issue we described with the University of Thrace (recall Section 6.1.1) we were able to issue an explanation with an exact diagnosis to the point of highlighting the commit and the problematic lines of code. As we were not only the developers of the system, but also its administrators, we tried hard to ensure that nobody could doubt our professional integrity.

In hindsight, there are two things that we wished we had done at the appropriate time. First, better educate the voters about e-voting, the concepts of Zeus itself, and vote auditing. Secondly, engage third parties in the process to run mixnets and audit the whole election process independently from us. The use of Scantegrity II at Takoma Park provides a good example both of working together with the local community and employing two independent auditors [Carback et al. 2010].

At the time of this writing Zeus is actively used, and new elections are announced that are using it in the coming days.

8. AVAILABILITY

Zeus is open software, available at https://github.com/grnet/zeus.

9. ACKNOWLEDGMENTS

We would like to thank Professor Aggelos Kiayias for the interesting discussions we had in the start of the project.

REFERENCES

Ben Adida. 2008. Helios: web-based open-audit voting. In *Proceedings of the 17th conference on Security symposium (SS'08)*. USENIX Association, Berkeley, CA, USA, 335–348.

Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. 2009. Electing a university president using open-audit voting: analysis of real-world use of Helios. In *Proceedings of the 2009 conference on Electronic voting technology/workshop on trustworthy elections (EVT/WOTE'09)*. USENIX Association, Berkeley, CA, USA.

Stephanie Bayer and Jens Groth. 2012. Efficient zero-knowledge argument for correctness of a shuffle. In *Proceedings of the 31st Annual international conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'12)*. Springer-Verlag, Berlin, Heidelberg, 263–280. DOI: http://dx.doi.org/10.1007/978-3-642-29011-4_17

Josh Benaloh. 2006. Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop (EVT'06)*. USENIX Association, Berkeley, CA, USA.

Josh Benaloh, Tal Moran, Lee Naish, Kim Ramchen, and Vanessa Teague. 2009. Shuffle-Sum: Coercion-Resistant Verifiable Tallying for STV Voting. *IEEE Transactions on Information Forensics and Security* 4, 4 (December 2009).

Philippe Bulens, Damien Giry, and Olivier Pereira. 2011. Running mixnet-based elections with Helios. In *Proceedings of the 2011 conference on Electronic voting technology/workshop on trustworthy elections (EVT/WOTE'11)*. USENIX Association, Berkeley, CA, USA.

Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. 2010. Scantegrity II municipal election at Takoma Park: the first E2E binding governmental election with ballot privacy. In *Proceedings of the 19th USENIX conference on Security (USENIX Security'10)*. USENIX Association, Berkeley, CA, USA.

Josh D. Cohen and Michael J. Fischer. 1985. A robust and verifiable cryptographically secure election scheme. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (SFCS '85)*. IEEE Computer Society, Washington, DC, USA, 372–382. DOI: http://dx.doi.org/10.1109/SFCS.1985.2

Denise Demirel, Jeroen Van De Graaf, and Roberto Araújo. 2012. Improving Helios with everlasting privacy towards the public. In *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE'12)*. USENIX Association, Berkeley, CA, USA.

Saghar Estehghari and Yvo Desmedt. 2010. Exploiting the client vulnerabilities in internet E-voting systems: hacking Helios 2.0 as an example. In *Proceedings of the 2010 international conference on Electronic voting technology/workshop on trustworthy elections (EVT/WOTE'10)*. USENIX Association, Berkeley, CA, USA, 1–9.

ha.ckers.org. Accessed in 2013. Slowloris HTTP DoS. http://ha.ckers.org/slowloris/. (Accessed in 2013).

Mario Heiderich, Tilman Frosch, Marcus Niemietz, and Jörg Schwenk. 2012. The bug that made me president a browser- and web-security case study on Helios voting. In *Proceedings of the Third international conference on E-Voting and Identity (VoteID'11)*. Springer-Verlag, Berlin, Heidelberg, 89–103. DOI: http://dx.doi.org/10.1007/978-3-642-32747-6_6

Douglas W. Jones and Barbara Simons. 2012. *Broken Ballots: Will Your Vote Count?* CSLI Publications, Stanford, California.

Fatih Karayumak, Maina M. Olembo, Michaela Kauer, and Melanie Volkamer. 2011. Usability Analysis of Helios—An Open Source Verifiable Remote Electronic Voting System. In *Proceedings of the Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)*. USENIX Association, Berkeley, CA, USA.

Reto E Koenig, Philipp Locher, and Rolf Haenni. 2013. Attacking the Verification Code Mechanism in the Norwegian Internet Voting System. In *Proceedings of the 4th International Conference on e-Voting and Identity (VoteID 2013)*. Springer-Verlag, Berlin, Heidelberg.

Kazue Sako and Joe Kilian. 1995. Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth. In *Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques (EUROCRYPT'95)*. Springer-Verlag, Berlin, Heidelberg, 393–403.

Bruce Schneier. 1995. *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.). John Wiley & Sons, Inc., New York, NY, USA.

Barbara Simons and Douglas W. Jones. 2012. Internet voting in the U.S. *Commun. ACM* 55, 10 (Oct. 2012), 68–77. DOI: http://dx.doi.org/10.1145/2347736.2347754

Björn Terelius and Douglas Wikström. 2010. Proofs of restricted shuffles. In *Proceedings of the Third international conference on Cryptology in Africa (AFRICACRYPT'10)*. Springer-Verlag, Berlin, Heidelberg, 100–113. DOI: http://dx.doi.org/10.1007/978-3-642-12678-9_7

Douglas Wikström. 2009. A Commitment-Consistent Proof of a Shuffle. In *Proceedings of the 14th Australasian Conference on Information Security and Privacy (ACISP '09)*. Springer-Verlag, Berlin, Heidelberg, 407–421. DOI: http://dx.doi.org/10.1007/978-3-642-02620-1_28

M. Wong and W. Schlitt. 2006. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408 (Experimental). (April 2006). http://www.ietf.org/rfc/rfc4408.txt

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System

Susan Bell, Office of the Travis County Clerk
Josh Benaloh, Microsoft Research
Michael D. Byrne, Rice University
Dana DeBeauvoir, Office of the Travis County Clerk
Bryce Eakin, independent researcher
Gail Fisher, Office of the Travis County Clerk
Philip Kortum, Rice University
Neal McBurnett, ElectionAudits
Julian Montoya, Office of the Travis County Clerk
Michelle Parker, Office of the Travis County Clerk
Olivier Pereira, Université catholique de Louvain
Philip B. Stark, University of California, Berkeley
Dan S. Wallach, Rice University
Michael Winn, Office of the Travis County Clerk

STAR-Vote is a collaboration between a number of academics and the Travis County (Austin), Texas elections office, which currently uses a DRE voting system and previously used an optical scan voting system. STAR-Vote represents a rare opportunity for a variety of sophisticated technologies, such as end-to-end cryptography and risk limiting audits, to be designed into a new voting system, from scratch, with a variety of real world constraints, such as election-day vote centers that must support thousands of ballot styles and run all day in the event of a power failure. This paper describes the current design of STAR-Vote which is now largely settled and whose development will soon begin.

1. INTRODUCTION

A decade ago, DRE voting systems promised to improve many aspects of voting. By having a computer mediating the user's voting experience, they could ostensibly improve usability through summary screens and a variety of accessibility features including enlarged text, audio output, and specialized input devices. They also promised to improve the life of the election administrator, yielding quick, accurate tallies without any of the ambiguities that come along with hand-marked paper ballots. And, of course, they were promised to be secure and reliable, tested and certified. In practice, DRE systems had problems in all these areas.

Many current DRE voting systems experienced their biggest sales volume after the failures of punch card voting systems in Florida in the 2000 presidential election. The subsequent Help America Vote Act provided a one-time injection of funds that made these purchases possible. Now, a decade later, these machines are near the end of their service lifetimes.

Last year, the Travis County election administration, having used Hart InterCivic's eSlate DRE system for over a decade, concluded that no system on the market—DRE or optical scan—met their future needs. They prefer to avoid hand-marked paper ballots because they open the door to ambiguous voter intent, a source of frustration in their previous centrally-tabulated optical scan system. They didn't want to go back.

Travis County's needs and preferences impose several significant constraints on the design of STAR-Vote:

DRE-style UI. Hand-marked ballots are not to be used, for the reason above. DRE-style systems were also preferred for their ability to offer facilities for voters with disabilities.

Printed paper ballot summaries. While the DRE-style UI was desired for entering selections, printed ballots were desired for their security benefits, verifiability by voters, and redundancy against failures in the electronic system. In order to save on paper and paper management, the county wished to only print a list of each voter's selections, analogous to the summary screens on many current-generation DRE systems.

All-day battery life. Power failures happen. Current-generation DRE systems have batteries that can last for hours. The new system must also be able to operate for hours without external power.

Early voting and election-day vote centers. Travis County supports two weeks of early voting, where any voter may vote in any of more than 20 locations. Also, on Election Day, any voter may go to any local polling place. Our county's voters informally report their appreciation of these benefits.

COTS hardware. Current DRE systems are surprisingly expensive. Travis County wants to use commercially available, off-the-shelf equipment, whenever possible, to reduce costs and shorten upgrade cycles. That is, "office equipment" rather than "election equipment" should be used where possible.

Long ballots. While voters in many countries only select a candidate for member of parliament, in the U.S., voters regularly face 100 or more contests for federal, state, and regional offices; judges; propositions; and constitutional amendments. STAR-Vote must support very long ballots as well as long list of contestants in each race.

These constraints interact in surprising ways. Even if the county did not have a strong preference for a DRE-like UI, pre-printed paper ballots are inefficient for vote centers, which may need to support hundreds or thousands of distinct ballot styles. Likewise, the requirement to run all-day on battery backup eliminates the possibility of using laser printers for ballot-on-demand printing, which consume far too much power.[1] We note that counties that face fewer constraints may choose to adopt quite different architectures. For example, a county without election-day vote centers might instead use pre-printed ballots and electronic ballot marking devices.

These constraints likewise eliminate prior-fielded e2e systems like Scantegrity [Carback et al. 2010; Chaum et al. 2008], and Prêt à Voter [Ryan and Peacock 2006; Burton et al. 2012], which rely on hand-marked paper, and Helios [Adida et al. 2009; Adida 2008], which is meant for use in web browsers, not traditional polling locations. Wombat [Ben-Nun et al. 2012] has a paper trail, but it's only designed for single-issue ballots. VoteBox [Sandler et al. 2008] has a DRE-like interface, but it's an entirely paperless system. Instead, to satisfy our constraints, we must build something new, or at least extend an existing system to satisfy our constraints.

We were charged with using the latest advances in human factors, end-to-end cryptography, and statistical auditing techniques, while keeping costs down and satisfying many challenging constraints. We want to generate quick, verifiable tallies when the election is over, yet incorporate a variety of audit mechanisms (some voter-verifiable, others facilitated by auditors with additional privileges).

We notably have chosen to design STAR-Vote without explicitly worrying about the constraints of State or Federal certification. Of course, for STAR-Vote to go into production, these challenges need to be addressed, but at least for now, our focus has been on designing the best possible voting system given our constraints.

2. VOTER FLOW

Figure 1 shows how STAR-Vote works from the perspective of a voter. The STAR-Vote voting system bears a resemblance to the Hart InterCivic eSlate system and to VoteBox [Sandler et al. 2008], in that the voting machines are networked together, simplifying the movement of data. Like eSlate, our design contains a networked group of voting machines that share a common judge's station with a computer like Hart InterCivic's "Judge Booth Controller" (JBC) that manages everything.

(1) *Registration (pollbook).* The first step for the voter is to check-in with a poll worker. This is where voter registration is verified and the voter's precinct and ballot style are determined. The

[1] A laser printer might consume 1000 watts or more while printing. A reasonably good UPS, weighing 26 kg, can provide that much power for only ten minutes. Since a printer must warm up for each page when printed one-off (perhaps 10 seconds per page), the battery might be exhausted by printing as few as 60 ballots.

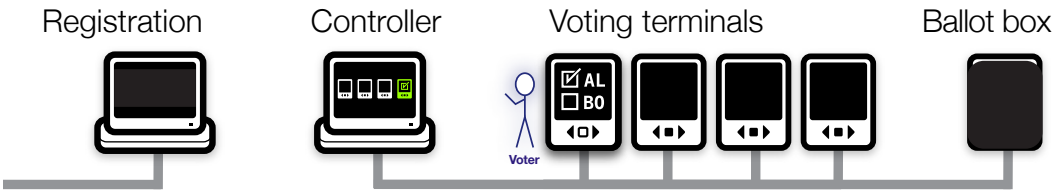


Fig. 1. The design of the STAR-Vote system. The voter registration system (left) is connected to the Internet but not to the internal LAN. Voters move left to right. First, the voter's registration is validated, and a thermal printout indicates the proper ballot style. This moves to the controller, which scans it and issues the voter a PIN, again printed on thermal paper. The voter proceeds to any open voting terminal, enters the PIN, and is given the proper ballot style. The ballot summary is printed, and deposited into the ballot box (right).

registration system, via cellular modem, notifies a centralized database of the voter's change in status, to eliminate any risk of double-voting.

The registration system will use a thermal label printer to generate a sticker with the voter's name, precinct and ballot style indicated. The precinct and ballot style are also indicated with a 1-D barcode. This sticker goes into a poll book which the voter signs, providing a backup to the online database. The barcode can also be read by an off-the-shelf scanner connected to the controller. This represents the only data flow from the outside world into the internal voting network, and helps avoid data entry errors that might come from human transcription. Nothing in the barcode is secret nor is it unique to the voter. Consequently, the flow of this information does not compromise the voter's privacy, so long as the voter is not the only voter with the same precinct and ballot style to vote at that polling location.

Provisional voters will be indicated with a suitable prefix to their precinct code, allowing the voting system to suitably distinguish their ballots from regular ones. (Provisional votes are cast by voters who, for whatever reason, do not appear in the voter registration database, and believe this to be in error. They are only tabulated after the voter's registration status is verified, typically not until at least a few days after the end of voting.)

(2) *Controller.* The controller scans the barcode on the sticker to identify the voter's precinct and ballot style. The controller then prints a 5-digit code, unique for the remainder of the election in this polling place. Holding this printout, the voter can then approach any open voting terminal, enter the code, and be presented with the correct ballot style. (There will probably need to be a special alternative for ADA compliance as not all voters can see or handle paper. One possible solution is that a poll worker could enter the relevant code, then depart before the voter begins voting.)

There are only ever a small number of 5-digit codes active at any one time, reducing the odds of a voter successfully guessing an active code and casting multiple ballots. We note that there will be no record binding the 5-digit code to the voter, helping ensure voter anonymity. We also note that these codes reduce the attack surface, relative to other voting systems that use smartcards or other active electronic devices to initialize a voting machine for each voter.

(3) *Voting terminals.* The voter makes selections with the GUI (for sighted voters) or auditory UI (for non-sighted voters). There is a review screen (or the auditory equivalent) so that the voter can confirm all selections before producing a paper record.

(4) *Print.* When the voter has finished making selections, the voting terminal prints two (possibly joined) items: (1) a paper ballot which includes a human-readable summary of the voter's selections and a random (non-sequential) serial number, and (2) a take-home receipt that identifies the voting terminal used, the time of the vote, and a short (16-20 character) hash code that serves as a commitment to the vote but does not reveal its contents.[2] The voting terminal also sends

[2] A secondary hash code with as many as 16-20 additional characters may be included for additional assurance.

data about the vote and receipt to the judge's station. (See Section 6 for the exact cryptographic design.)

(5) *Review printed record.* The voter may then review the printed record to confirm the indicated selections. There will be at least one offline station available that can scan the paper record and read it back to the voter for those who cannot visually read the paper record.

(6) *Option: Cast or challenge/spoil.* After reviewing the ballot, the voter has a choice: Cast the ballot or spoil it. A voter might spoil the ballot because of an error (or change of heart) or because the voter wishes to challenge the voting terminal, demanding it to show that the voter's selections were correctly recorded and committed to. This process represents a novel variant on Benaloh challenges [Benaloh 2006; 2007]; rather than asking the voter a "cast or challenge" question, the voter either deposits the ballot in the box or not. This represents a potentially significant usability gain over prior variants of the Benaloh challenge.

The two procedures are described below. Note also that there is a special procedure for provisional ballots.

Regardless, the voter may keep the take-home paper receipt. We note that most thermal printers include a cutting device that leaves a small paper connection between the two sides of the cut. It is therefore a simple matter for the voting terminal to print a single sheet that the voter can easily separate into the ballot summary and the take-home receipt. We also note that "privacy sleeves" (i.e., simple paper folders) can protect the privacy of these printed ballots as voters carry them from the voting machine either to the ballot box to be cast, or to the judge's station to be spoiled.

   (a) *Ballot box: cast ballot.* A voter who wishes to cast the ballot takes the paper ballot summary to the ballot box. The ballot box has a simple scanner that can read the serial number from the ballot (the serial number might also be represented as a one-dimensional barcode for reliability) and communicate this to the controller, allowing the controller to keep a record of which ballots have found their way to the ballot box, and thus, which ballots should be tabulated. *An electronic ballot record is not considered complete and should not be included in the tally unless and until its corresponding paper ballot summary has been deposited in the ballot box.*

   (b) *Spoil ballot.* If the paper record is to be spoiled, the voter returns to a poll worker. The ballot serial number is scanned so that the controller can record that the ballot is to be spoiled. This informs the controller that the corresponding encrypted ballot record should not be included in contest results. Instead, it should be decrypted and published as such after the election is over. The original printed paper ballot thus corresponds to a *commitment* by the voting machine, before it ever "knew" it might be challenged. If the voting machine cannot produce a suitable proof that the ballot encryption matches the plaintext, then it has been caught cheating. Voters who don't care about verification can simply restart the process. For voters who may feel uncomfortable with this process, as it might reveal their intent to a poll worker, we note that voters could deliberately spoil ballots that misstate their true intent. We note that dedicated election monitors could be allowed to use voting machines, producing printed ballots that they would be forbidden from placing in the ballot box, but which would be spoiled and then the corresponding ciphertext would be decrypted. In effect, election monitors can conduct *parallel testing in the field* on any voting machine at any time during the live election.

   (c) *Provisional ballot.* In the case of a provisional ballot, the voter must return the printed ballot to a poll worker. The voter can choose to spoil the ballot and re-vote or to cast the ballot provisionally by having it placed—under an identifying seal—into a distinct provisional ballot box. The voter may retain the receipt to see if the ballot ends up being counted. Because the ballot box is connected to the controller over the LAN, it can also query the controller as to whether the ballot is provisional. In the event that a voter accidentally puts a provisional ballot into the ballot box, the scanner can detect this and reject the printed ballot. (Provisional ballots need to go into dedicated envelopes that are processed after the voting has ended.)

(7) *At home (optional): Voter checks crypto.* The encrypted votes will be posted on a public "bulletin board" (i.e., a web site maintained by the county). The voter receipt corresponds to a cryptographic hash of the encrypted vote. A voter should be able to easily verify that this vote is present on the bulletin board. If a voter spoiled a ballot, that should also be visible on the bulletin board together with its decrypted selections. This allows independent observers to know which ballots to include in the tally and allows independent verifiers to check that all spoiled ballots are correctly decrypted. Individual voters can check, without any mathematics, that the decryptions of their own spoiled ballots match their expectations.

While this process is more cumbersome than a traditional DRE voting system, it has several advantages. By having the paper elements, this system not only benefits from sophisticated end-to-end cryptographic techniques (described in Section 6), it also can be audited post-election, by hand, using a risk-limiting audit (see Section 5). Voters also have the confidence that comes from holding, verifying, and casting a tangible record of their votes, whether or not they trust the computers.

3. DESIGN

From the perspective of voters, the process of registration and poll-station sign-in is unchanged from current practice. Once authorized, voters proceed to a voting terminal where they use a rich interface that prevents overvotes, warns of undervotes, and supports alternative input/output media for disabled and impaired voters. The printed ballot summary and the corresponding electronic ballot record both include a variety of cryptographic features, which we now describe.

3.1. Crypto Overview

From the perspective of election officials, the first new element in the election regimen is to generate the cryptographic keys. A set of election trustees is designated as key holders and a threshold number is fixed. The functional effect is that if there are n election trustees and the threshold value is k, then any k of the n trustees can complete the election, even if the remaining $n-k$ are unavailable. This threshold mechanism provides robustness while preventing any fewer than k of the trustees from performing election functions that might compromise voter privacy. Threshold cryptosystems are straightforward extensions of traditional public-key cryptosystems [Desmedt and Frankel 1989].

The trustees each generate a key pair consisting of a private key and a public key; they publish their public keys. A standard public procedure is then used to compute a single public key from the n trustee public keys such that decryptions can be performed by any k of the trustees. This single election public key K is published and provided to all voting terminals together with all necessary ballot style information to be used in the election. A start value z_0, which is unpredictable and unique to the election, is also chosen and distributed to each voting terminal for reasons discussed below.

During the election, voters use voting terminals to make their selections. Once selections are completed, the voting terminal produces paper printouts of two items. The first is the paper ballot summary which consists of the selections made by the voter and also includes a random (non-sequential) serial number. The second is a receipt that consists of an identification number for the voting terminal, the date and time of the vote, and a short hash of the encryption of the voter's selections together with the previous hash value. Specifically, if the voter's selections are denoted by v, the i^{th} hash value produced by a particular voting terminal m in an election is computed as

$$z_i = H(E_K(v), m, z_{i-1})$$

where H denotes the hash function and E denotes encryption. This separation of the ballots into two parts makes sure that the ballot summary does not contain any voter-related information, while the take-home receipt does not leak any information about the voter choices. Furthermore, since we only store votes in an encrypted form, and since the decryption keys are kept out of the system, there is no problem with storing the votes with timestamps: they could only allow linking a voter to a ciphertext that will never be decrypted, which is harmless.

The voting terminal should retain both the encrypted ballots and the current hash value. At the conclusion of the election (if not sooner), the encrypted ballots should be posted on a publicly-accessible web page and digitally signed by the election office using a simple signature key (not the key generated by the trustees). The posting of each encrypted ballot should also include a non-interactive zero-knowledge (NIZK) proof that the ballot is well-formed. Once they receive their ballot summaries and take-home receipts, voters may either deposit their ballot summaries into a ballot box or take them to a poll-worker and have them spoiled. Ballot summaries deposited in a ballot box have their serial numbers scanned and recorded. The electronically stored encrypted vote is not considered complete (and not included in the tally) unless and until its corresponding serial number has been recorded in the ballot box.

Any electronically stored encrypted ballots for which no corresponding serial number has been scanned and recorded are deemed spoiled. The published election record should include all spoiled ballots as well as all cast ballots, but for each spoiled ballot the published record should also include a verifiable decryption of the ballot's contents. Voters should be able to easily look up digitally-signed records for any receipts they hold and verify their presence and, for spoiled receipts, the ballot contents.

A voter who takes a completed paper ballot summary to a poll worker can request that the poll worker spoil the ballot and give the voter an opportunity to re-vote. The poll worker marks both the take-home receipt and the paper ballot summary as spoiled (including removing or marking the serial number so that it will not be recorded if subsequently placed in the ballot box) and returns the spoiled ballot summary to the voter.

Upon completion of the election, the election office homomorphically combines the cast ballots into an aggregate encryption of the election tally (this can be as simple as a multiplication of the public encrypted ballots). At least k of the election trustees then each perform their share of the decryption of the aggregate as well as individual decryptions of each of the spoiled ballots. The trustees also post data necessary to allow observers to verify the accuracy of the decryptions.

A privacy-preserving risk-limiting audit is then performed by randomly selecting paper ballot summaries and matching each selected ballot with a corresponding encrypted ballot to demonstrate the correct matching and provide software-independent evidence of the outcome [Rivest and Wack 2006; Lindeman and Stark 2012; Stark and Wagner 2012].

3.2. Triple Assurance

Three lines of evidence are produced to support each election outcome [Stark and Wagner 2012]. The homomorphic tallying process proves that the announced tally corresponds to the posted encrypted ballot records. The ballot challenge and receipt checking processes allow voters to check that these encrypted ballot records correctly reflect their selections. The risk-limiting audit process serves to verify the correspondence between the paper records and the electronic records. In addition, the paper records remain available in case of systemic failure of the electronic records or if a manual count is ever desired. The paper and electronic records are conveyed to the local election office separately, providing additional physical security of the redundant audit trail.

The design of the election system ensures that all three of these checks should be perfectly consistent. There is sufficient information in the records so that if any discrepancies arise (for instance because of loss of some of the electronic or paper records), the discrepancies can be isolated to individual ballots that are mismatched or counted differently.

Why combine e2e with risk-limiting auditing? Each provides different guarantees and they support each other's strengths. E2e techniques, for example, provide cryptographically strong evidence that a voter's receipt corresponds to a ballot, on the bulletin board, which has been included correctly in the final tally—a guarantee that risk-limiting audits alone cannot accomplish. However, if there's a discrepancy, e2e techniques cannot necessarily identify where things went wrong. Risk-limiting audits provide a backstop to prevent cryptographic failures from ruining the election outcome. They also provide a secondary check against machines that might be producing paper and electronic records that disagree, even if voters aren't bothering to conduct e2e challenge audits.

3.3. Software and Hardware Engineering

An important criteria for STAR-Vote is that it should leverage commodity components whenever feasible. This reduces cost and simplifies the ability for an election administrator to replace aging hardware by sourcing it from multiple vendors. While this paper isn't intended to cover certification issues, the separation of hardware and software allows for the possibility of *commercial off-the-shelf* (COTS) hardware, which *could* be subject to a lower bar for certification than the software.

Ideally, the voting terminals and the judge station could use identical hardware. In particular, we believe that a reasonable target might be "point of sale" terminals. These are used in restaurants worldwide. They are used in relatively demanding environments and, on the inside, are ordinary PCs, sometimes built from low-power laptop-class parts. The only missing hardware from a COTS point of sale terminal, relative to our needs for STAR-Vote, are a printer and a battery.

If you want a reliable, low-power printer, without having to worry about consumable ink or toner, there's only one choice: thermal printers. They come in a variety of widths, up to US Letter size. Thermal paper, particularly higher cost thermal paper, can last for years in an air-conditioned warehouse, although some experimentation would be required to see whether it can survive an un-air-conditioned trip in a hot car in the summer. Every shipping label from major online vendors like Amazon is printed thermally, lending some credence to its survivability in tough conditions.

Specifying a battery is more complicated. We could require that the voting terminal have an internal (and removable) battery, but this eliminates COTS point of sale terminals. Tablet computers come with built-in batteries that, at least in some cases, can last all day. Tablet computers have smaller screens than we might prefer, but they don't have hardware Ethernet ports or enough USB ports to support accessibility devices and printers[3]. Also, we would prefer to use wired networks, rather than the wireless networks built into most tablets. We note that a number of vendors are now releasing touchscreen-enabled laptops and larger touchscreen desktop models to support Windows 8. This new hardware is likely to provide good options for running STAR.

For the software layer, we see no need for anything other than a commodity operating system, like Linux, which can be stripped of unessential features to reduce the attack surface. For example, we don't need a full-blown window system or 3D graphics pipeline. All we need are basic pre-rendered ballots, as in pVote [Yee et al. 2006; Yee 2007] or VoteBox [Sandler et al. 2008]. We would specify that the voting system software be engineered in a type-safe language like Java or C# (eliminating buffer overflow vulnerabilities, among other problems) and we would also specify that the software be engineered with *privilege separation* [Provos et al. 2003], running separate parts of the voting software as distinct applications, with distinct Unix user-ids, and with suitably reduced privileges. For example, the storage subsystem can maintain append-only storage for ballots. The voter-facing UI would then have no direct access to ballot storage, or the network, and could be "rebooted" for every voter. Consequently, a software compromise that impacts the UI application could impact at most one voter. A tablet that includes a Trusted Platform Module (TPM) can provide additional assurance that the correct software — and only the correct software — is running on the device.

A separation architecture like this also provides some degree of protection over sensitive cryptographic key materials, e.g., if we want every voting terminal to have a unique private key to compute digital signatures over ballots, then we must restrict the ability for compromised software to extract the private keys. DStar [Zeldovich et al. 2008], for example, used this technique to protect the key material in an SSL/TLS web server.

4. USABILITY

4.1. Design Considerations

In designing this reference voting system it was important to maximize the usability of the system within the framework of enhanced security and administrative expediency. The overall design of the

[3]While a single USB port can connect to a USB hub, which would then have more expandability, a *powered* USB hub might be necessary to drive some devices like a USB Ethernet adapter, complicating our requirement to keep STAR running even when on battery power.

system was strongly influenced by usability concerns. For example, a proposal was put forth to have all voters electronically review the paper record on a second station; this was rejected on usability grounds. ISO 9241 Part 11 [ISO] specifies the three metrics of usability as effectiveness, efficiency, and satisfaction, and these are the parameters we attempt to maximize in this design. Effectiveness of the system means that users should be able to reliably accomplish their task, as they see it. In voting, this means completing a ballot that correctly records the candidate selections of their choice, whether that be though individual candidate selection by race, straight party voting, or candidate write-ins. Efficiency measures the ability of a voter to complete the task with a minimum of effort, as measured through time on task or number of discrete operations required to complete a task. Efficiency is important because users want to complete the voting task quickly and voting officials are concerned about voter throughput. Reduced efficiency means longer lines for waiting voters, more time in the polling booth, and higher equipment costs for election officials. Satisfaction describes a user's subjective assessment of the overall experience. While satisfaction does not directly impact a voter's ability to cast a vote in the current election, it can have direct impact on their willingness to engage in the process of voting at all, so low satisfaction might disenfranchise voters even if they can cast their ballots effectively and efficiently. How does this design seek to maximize these usability metrics? For voting systems, the system must generally be assumed to be walk-up-and-use. Voting is an infrequent activity for most, so the system must be intuitive enough that little to no instruction is required to use. The system should minimize the cognitive load on voters, so that they can focus on making candidate selections and not on system navigation or operation. The system should also mitigate the kinds of error that humans are known to make, and support the easy identification and simple correction of those errors before the ballot is cast.

Why not paper?. Paper ballots (bubble ballots in particular) have many characteristics that make them highly usable [Everett et al. 2006; Byrne et al. 2007]. Users are familiar with paper, and most have had some experience with bubble-type item selection schemes. Voting for write-in candidates can be relatively simple and intuitive. Unlike electric voting machines, paper is nearly 100% reliable and is immune from issues of power interruption. Further, paper leaves an auditable trail, and wholesale tampering is extremely difficult. However, paper is not a perfect solution. Voters actually show higher satisfaction with electronic voting methods than they do with paper [Everett et al. 2008] and paper has significant weaknesses that computers can overcome more easily. First, the ambiguity that can be caused by partial marks leads to substantial problems in counting, recounting, and re-interpreting paper ballots. Second, voting by individuals with disabilities can be more easily accommodated using electronic voting methods (e.g., screen readers, jelly switches). Third, electronic voting can significantly aid in the reduction of error (e.g. undervotes, overvotes, stray marks) by the user in the voting process. Fourth, electronic voting can more easily support users whose first language is not English, since additional ballots for every possible language request do not have to be printed, distributed and maintained at every polling location. This advantage is also evident in early voting and vote center administration; rather than having to print, transport, secure, and administer every possible ballot for every precinct, the correct ballot can simply be displayed for each voter. Computers also facilitate sophisticated security and cryptography measures that are more difficult to implement in a pure paper format. Finally, administration of the ballots can be easier with electronic formats, since vote counting and transportation of the results are more efficient. We have taken a hybrid approach in this design, by using both paper and electronic voting methods in order to create a voting system that retains the benefits of each medium while minimizing their weaknesses.

Usability vs Security. Usability and security are often at odds with each other. Password design is a perfect example of this tension. A system that requires a user have a 32-character password with upper and lower case letters, digits, and symbols with no identifiable words embedded might be highly secure, but it would have significant usability issues. Further, security might actually be *compromised* since users are likely to write such a difficult password down and leave it in an insecure location (e.g., stuck to the computer monitor). For voting systems, we must strive for maximum usability while not sacrificing the security of the system (our security colleagues might argue that

we need to maximize security while not sacrificing usability). In our implementation, many of the security mechanisms are invisible to the user. Those that are not invisible are designed in such a way that only those users who choose to exercise the enhanced security/verifiability of the voting process are required to navigate additional tasks (e.g., ballot challenge, post-voting verification).

Accessibility vs Security. STAR-Vote makes strategic use of paper to enhance the overall security and auditability of the voting process. From an auditability standpoint, the presence of the paper ballot allows matching of the paper and electronic records and preserves a separate physical copy apart from the electronic tally. From a security standpoint, it allows a voter to verify that the choices selected on the electronic voting terminal (DRE) have been faithfully recorded on the paper ballot (although this voter verification is not a robust as one might hope [Everett 2007]), and challenge their vote if they choose to do so. However, the added benefits provided by the inclusion of paper come at a cost to the accessibility of the system. Visually impaired voters must now be given a way to verify the contents of printed material and be guided in the handling of that paper into the scanners and ballot boxes. Voters with mobility impairments must now handle these paper ballots with moderate dexterity in order to feed them into the scanning ballot boxes as well. Solutions to this tradeoff are still under evaluation. Many obvious solutions, such as giving voters with disabilities the option to simply cast an electronic ballot without a paper record, seriously compromise the overall security and auditability of the voting system, and also present significant privacy concerns, since voters who opt out the main flow might be easily identified. Simple but non-optimal solutions are being considered (test-to-speech scanning stations, ballot privacy sleeves and increased poll worker involvement), but we continue to investigate more elegant solutions that involve automatic paper handling mechanisms. A final design has still not been identified.

Error reduction. The use of computers in combination with paper is anticipated to reduce errors committed by voters. Because voters will fill out the ballot on electronic voting terminals, certain classes of errors are completely eliminated. For example, it will be impossible to over vote or make stray ballot marks, as the interface will preclude the selection of more than a single candidate per race. Under voting will be minimized by employing sequential race presentation, forcing the voter to make a conscious choice to skip a race [Greene 2008]. Undervotes will also be highlighted in color on the review screen, providing further opportunity for users to correct accidental undervotes. This review screen will also employ a novel party identification marker (see below) that will allow a voter to easily discern the party for which they cast a vote in each race. The use of the paper ballot (printed when the voter signals completion) provides the voter with a final chance to review all choices before casting the final ballot.

4.2. User Interface Design Specification

The basic design for the UI is a standard touchscreen DRE with auditory interface for visually impaired voters and support for voter-supplied hardware controls for physical impairments (e.g., jelly switches).

The VVSG. The starting point for UI specifications is the 2012 draft version 1.1 of the Voluntary Voting System Guidelines (VVSG). These guidelines specify many of the critical properties required for a high-quality voting system user interface, from simple visual properties such as font size and display contrast to more subtle properties such as ballot layout. They also require that interfaces meet certain usability benchmarks in terms of error rates and ballot completion time. We believe that no extant commercial voting UI meets these requirements, and that any new system that could meet them would be a marked improvement in terms of usability. That said, there are some additional requirements that we believe should be met.

Accessibility. While the VVSG includes many guidelines regarding accessibility, more recent research aimed at meeting the needs of visually-impaired voters [Piner and Byrne 2011] has produced some additional recommendations that should be followed. These include:

— In order to capitalize on user preference, a synthesized male voice should be used.
— Navigation should allow users to skip through sections of speech that are not important to them as well as allowing them to replay any parts they may have missed or not comprehended the first time.
— At the end of the voting process, a review of the ballot must be included, but should not be required for the voter.

Review Screens. Another area where the VVSG can be augmented concerns review screens. Voter detection of errors (or possible malfeasance) on review screens is poor [Everett 2007], but there is some evidence that UI manipulations can improve detection in some cases [Campbell and Byrne 2009a]. Thus, STAR-Vote requires the following in addition to the requirements listed in the VVSG:

— Full names of contests and candidates should be displayed on the review screen; that is, names should be text-wrapped rather than truncated. Party affiliation should also be displayed.
— Undervotes should be highlighted using an orange-colored background.
— Activating (that is, touching on the visual screen or selecting the relevant option in the auditory interface) should return the voter to the full UI for the selected contest.
— In addition to party affiliation in text form, graphic markings should be used to indicate the state of each race: voted Republican, voted Democratic, voted Green, etc.—with a distinctive graphic for "not voted" as well. These graphic markings should be highly distinguishable from each other so that a rapid visual scan quickly reveals the state of each race, while taking note of potential usability issues with graphics symbols [Smith et al. 2009]. Exact graphic symbols for STAR-Vote have not yet been determined.

Paper Record. The VVSG has few recommendations for the paper record. For usability, the paper record should meet VVSG guidelines for font size and should contain full names for office and candidate. To facilitate scanner-based retabulations, the font should be OCR-friendly. Contest names should be left-justified while candidate names should be right-justified to a margin that allows for printing of the same graphic symbols used in the review screen to facilitate rapid scanning of ballots for anomalies. Candidate names should not be placed on the same line of text as the contest name and a thin horizontal dividing line should appear between each office and the next in order to minimize possible visual confusion.

4.3. Issues that still need to be addressed

There are still several issues that need to be addressed in order to make the system have the highest usability. The first of these is straight party voting (SPV). SPV can be quite difficult for a voter to understand and accomplish without error, particularly if voters intend to cross-vote in one or more races [Campbell and Byrne 2009b]. Both paper and electronic methods suffer from these difficulties, and the optimum method of implementation will require additional research. Races in which voters are required to select more than one candidate (k of n races) also create some unique user difficulties, and solutions to those problems are not yet well understood.

5. AUDIT

The E2E feature of STAR-Vote enables individual voters to confirm that their votes were included in the tabulation, and that the encrypted votes were added correctly. The challenge feature, if used by enough voters, assures that the encryption was honest and that substantially all the votes are included in the tabulation. But there might not be many voters who challenge the system; the voters who do are hardly representative of the voting public; and some problems may go unnoticed. Moreover, the anonymized form of E2E used here does not allow a voter to confirm that *others'* ballots were included in the tabulation, only that those ballots that were included were included correctly.

The paper audit trail enables an entirely independent check that the votes were included and tabulated accurately, that the visible trace of voter intent as reflected in the ballot agrees with the encryption, and, importantly, that the winners reported by the voting system are the winners that a

full hand count of the audit trail would reveal. The key is to perform a compliance audit to ensure that the audit trail of paper ballots is adequately intact to determine the outcomes, and then to perform a risk-limiting audit of the machine interpretation against a manual interpretation of the paper ballots. For the risk-limiting audit, STAR-Vote uses SOBA [Benaloh et al. 2011] with improvements given by [Lindeman and Stark 2012].

A risk-limiting audit guarantees a large minimum chance of a full hand count of the audit trail if the reported outcome (i.e., the set of winners) disagrees with the outcome that the full hand count would reveal. The full hand count then sets the record straight, correcting the outcome before it becomes official. Risk-limiting audits are widely considered best practice for election audits [Lindeman et al. 2008; Bretschneider et al. 2012].

The most efficient risk-limiting audits, ballot-level comparison audits, rely on comparing the machine interpretation of individual ballots (cast vote records or CVRs) against a hand interpretation of the same ballots [Stark 2010; Benaloh et al. 2011; Lindeman and Stark 2012]. Current federally certified voting systems do not report cast vote records, so they cannot be audited using the most efficient techniques [Lindeman and Stark 2012; Stark and Wagner 2012]. This necessitates expensive work-arounds.[4] The preamble to conducting a ballot-level comparison audit using currently deployed voting systems can annihilate the efficiency advantage of ballot-level comparison audits [Stark and Wagner 2012].

A big advantage of STAR-Vote is that it records and stores individual cast vote records in a way that *can* be linked to the paper ballot each purports to represent, through encrypted identifiers of the ballot corresponding to each voter's selections, separately for each contest. This makes ballot-level comparison audits extremely simple and efficient. It also reduces the vulnerability of the audit to human error, such as accidental changes to the order of the physical ballots.[5]

A comparison audit can be thought of as consisting of two parts: Checking the addition of the data,[6] and randomly spot-checking the accuracy of the data added, to confirm that they are accurate enough for their tabulation to give the correct electoral outcome. The data are the votes as reported by the voting system. For the audit to be meaningful, the election official must commit to the vote data before the spot-checking begins. Moreover, for the public to verify readily that the reported votes sum to the reported contest totals, it helps to publish the individual reported votes. However, if these votes were published ballot by ballot, pattern voting could be used to signal voter identity, opening a communication channel that might enable widespread wholesale coercion [Rescorla 2009; Benaloh et al. 2011].

The SOBA risk-limiting protocol [Benaloh et al. 2011] solves both of these problems: It allows the election official to commit cryptographically and publicly to the vote data; it publishes the vote data in plain text but "unbundled" into separate contests so that pattern voting cannot be used to signal. Moreover, the computations that SOBA requires are extremely simple (they are simplified even further by [Lindeman and Stark 2012]). The simplicity increases transparency, because observers can confirm that the calculations were done correctly with a pencil and paper or a hand calculator.

The encrypted ballot/contest identifiers on the ballot that STAR-Vote produces allow the electronic cast vote records for each contest to be linked to the paper they purport to represent. This simplifies SOBA procedures because it eliminates the need to store ballots in a rigid order. Moreover, because the voting terminal generates both the electronic vote data and the paper ballot, the audit should find very few if any discrepancies between them.

---

[4]For instance, a *transitive audit* might require marking the ballots with unique identifiers or keeping them in a prescribed order, re-scanning all the ballots to make digital images, and processing those images with software that can construct CVRs from the images and associate the CVRs with the ballots. That software in turn needs to be programmed with the all the ballot definitions in the contest, which itself entails a great deal of error-prone handwork.

[5]For instance, we have seen groups of ballots dropped on the floor accidentally; even though none was lost, restoring them to their original order was impossible.

[6]This presupposes that the contest under audit is a plurality, majority, super-majority, or vote-for-*k* contest. The operation that must be checked to audit an instant-runoff contest is not addition, but the same principle applies.

But since voters and election workers will handle the ballots in transit from the voting terminal to the scanner to the audit, voters might make marks on their ballots. Depending on the rules in place for ascertaining voter intent from the ballot, those marks might be interpreted as expressing voter intent different from the machine-printed selections, in which case the SOBA audit might find discrepancies.

It could also happen that a ballot enters the ballot box but its serial number is not picked up, so the electronic vote data ends up in the "untallied but unspoiled" group. This should be detectable by a compliance audit [Benaloh et al. 2011; Lindeman and Stark 2012; Stark and Wagner 2012] as a mismatch between the number of recorded votes and the number of pieces of paper, providing an opportunity to resolve the problem before the audit begins.

If such cases remain and turn up in the audit sample, SOBA would count them as discrepancies and the sample might need to expand, either until there is strong evidence that the electoral outcomes are correct despite any errors the audit uncovers, or until there has been a complete hand count.

The random selection of ballots for the SOBA audit should involve public participation in generating many bits of entropy to seed a high-quality, public, pseudo-random number generator (PRNG), which is then used to select a sequence of ballots to inspect manually [Lindeman and Stark 2012]. (For instance, audit observers might roll 10-sided dice repeatedly to generate a 20-digit number.) Publishing the PRNG algorithm adds transparency by allowing observers to verify that the selection of ballots was fair.

6. THE CRYPTOGRAPHIC WORKFLOW

The core elements. At its core, the cryptographic workflow of STAR-Vote follows the approach of Cramer, Gennaro and Schoenmakers [Cramer et al. 1997], also used in Helios [Adida et al. 2009] and VoteBox[Sandler et al. 2008], among others. Cryptographic analyzes of this approach can be found in [Bernhard et al. 2012; Cortier et al. 2013]. We then augment this approach in various ways in order to ease the detection of and recovery from potential problem.

STAR-Vote keeps an electronic record of all the votes encrypted with a threshold cryptosystem (so that decryption capabilities are distributed to protect voter privacy) that has an additive homomorphic property (to allow individual encrypted ballots to be combined into an aggregate encryption of the tally). The common exponential version of the Elgamal cryptosystem [ElGamal 1985] satisfies the required properties, and stronger security is obtained by using PPATS encryption [Cuvelier et al. 2013], in particular against key manipulation errors by the trustees and long-term security. The encryption scheme comes with an extraction function Ext that, from a ciphertext, extracts a commitment on the encrypted value. In the case of Elgamal, this commitment is be the ciphertext itself, while in the case of PPATS, it is a perfectly hiding homomorphic commitment.

Cryptographic key generation can be accomplished in one of two ways, depending on the availability of the election trustees and the desired amount of robustness. The preferred process offers general threshold key generation requires multiple rounds (see [Gennaro et al. 2007] for Elgamal and PPATS), but can be simplified into a single-round solution if redundancy is eliminated (as in Helios for instance [Adida et al. 2009]). At the end of the key generation procedure, the trustees each hold a private key share that does not contain any information on the full private key, and the unique public key K corresponding to those shares is published.

During the polling phase, the ballot marking devices encrypt the votes of each voter using the public key K. This encryption procedure is randomized in order to make sure that two votes for the same candidates result in ciphertexts that look independent to any observer.

Following Benaloh [Benaloh 2006], a cryptographic hash value of the commitment extracted from each ciphertext (and of a few more data, as discussed below) is also computed, fingerprinting the ballot to a 256 bit string. An abridged form of which is provided to the voter in a human readable form as part of the take-home receipt. All the hashes and commitments are computed and posted on a publicly accessible web page, as soon as the polls are closed. This web page is digitally signed by the election office using a traditional signature key (as performed by [Adida et al. 2009]). This signature operation makes it infeasible to consistently modify the content of the web page without

the help of the signer, and provides evidence against a malicious signer who would try to sign various versions of the bulletin board.

The posting of all the hashes gives all voters the ability to verify that their ballots have been recorded properly. The commitments can also be checked for consistency with the hashes and used to confirm the homomorphic aggregation of the individual ballots into a single encryption of the sum of the ballots, which constitutes an encryption of the election tallies.

At the end of the election, any set of trustees that achieve the pre-set quorum threshold use their respective private keys to decrypt the derived aggregate tally encryption. This procedure is simple and efficient and can be completed locally without interaction between the trustees. We note that the individual encrypted ballots, from which the aggregate encryption of the tallies is formed, are never individually decrypted. However, each spoiled ballot *is* individually decrypted using exactly the same process that is used to decrypt the aggregate tally encryption.

The elements we just described make the core of the workflow and are sufficient to compute an election tally while preserving the privacy of the votes. We now explain various ways in which this simple workflow is hardened in order to make sure that the tally is also correct. All the techniques that follow enable the verification of different aspects of the ballot preparation and casting.

Hardening encryption. Since the tally does not involve the decryption of any individual ballot, and since the audit procedure relies on the fact that all counted ballots are properly formed, it is crucial to make sure that all the encrypted ballots that are added correspond to valid votes [Benaloh and Fischer 1985]. This is achieved by requiring the ballot marking devices to compute, together with the encryption of the votes, a non-interactive zero-knowledge (NIZK) proof that each ballot is well-formed. Such a proof guarantees that each ciphertext encrypts a valid vote and does not leak any other information about the content of the vote. As a side benefit, this proof can be designed to make the ballots non-malleable, which provides an easy technique to prevent the replay of old ballots (i.e., reject duplicates). Traditional sigma proofs provide the required security properties and are described and analyzed in [Bernhard et al. 2012].

We note that, if malicious software were to get into the voting system, it could use the randomness inherent in the encryption process to encode a subliminal message to an external observer. This sort of threat, along with the threat of a malicious voting machine that simply records every vote cast, in plaintext, in internal memory, is something that cryptography cannot address. (More discussion on this appears in Section 3.3.)

Hardening decryption. Making sure that the encrypted ballots are valid is not enough: we also need to make sure that the tally is correctly decrypted as a function of those encrypted ballots: otherwise, malicious trustees (or trustees using corrupted devices) could publish an outcome that does not correspond to these ballots. As a result, we require the trustees to provide evidence of the correctness of the decryption operations that they perform. This can also be accomplished with sigma proofs in the case of Elgamal or more simply by publishing commitment openings in the case of PPATS.

Hardening the timeline. The procedures described above prevent malfunctioning or corrupted voting terminals or trustees to falsify individual ballots or decryption operations.

The detection of manipulation of encrypted ballots can be more effective by linking ballots with each other, using hash chaining [Sandler and Wallach 2007; Benaloh and Lazarus 2011]. For this purpose, each ballot marking device is seeded, at the beginning of the election, with a public start value z_0 that includes a unique identifier for the election. This unique identifier is chosen at random shortly before the election, either in a central way or by the poll workers themselves at the beginning of election day.

From this seed, all election events are chain hashed, with z_{i+1} being computed as a hash of z_i concatenated to the id of the machine on which the event happens and to the event content. Two such chains are maintained and properly separated. One is internal and contains the full election data, including the encryption of the votes, the casting time of each paper ballot, and information

on machines being added or removed. The second is public and chains the commitment extracted from all encrypted votes, together with time and identifiers for the election and voting machine. The public hash is the one actually printed on the take-home receipt. When the polls close, the final value of the hash chains are digitally signed, and the public chain is made public together with all the information needed for its reconstruction.

As a result of this procedure, any removed ballot will invalidate the hash chain which is committed to at the close of the election and whose constituents appear on the voter take-home receipts.

Hardening the link between the paper and electronic election outcome. As described in Section 5, STAR-Vote includes a risk limiting audit (RLA) based on the human-readable versions of each ballot summary printed by the voting terminals and inspected for correctness by voters. This RLA comes in addition to the cast or challenge procedure discussed above, and the production of the inputs for the RLA is an original contribution of STAR-Vote.

The requirement for running the RLA is to commit on a full electronic record including a 1-to-1 mapping and evidence that this electronic record leads to the announced outcome. This is achieved as follows.

(1) For each ballot, the ballot marking device selects a random ballot id sequence number *bid*. This *bid* is printed on the ballots as a barcode. Furthermore, for each race r to which the voter participates, an encryption of $H(bid\|r)$ is also computed and appended to the encryption of the choices.
(2) At the end of the day, and before decryption of the tallies, the trustees (or their delegates) shuffle and rerandomize all encrypted votes, race by race. This shuffle does not need to be verifiable, even though a verifiable shuffle would improve accountability by making it possible to verify that the shufflers did not cheat if it happens that a discrepancy is detected during the RLA. However, in the case of a non verifiable shuffle, the shufflers must save their permutation and randomness until the end of the election audit. The non-verifiable solution is preferred for its simplicity (verifiable shuffles are particularly challenging to implement properly) and for its efficiency (permutations and reencryption factors can be precomputed, leaving only one multiplication to perform per ciphertext in the online phase, which is convenient when millions of ciphertexts have to be shuffled).
(3) When the trustees decrypt the homomorphically added votes, they also decrypt the output of this shuffle. For each race, this provides a list of elements of the form $H(bid\|r)$ and the corresponding cleartext choices.
(4) Now, auditors can sample the paper ballots, read the *bid* printed on them, recompute the value of $H(bid\|r)$ for all races present on the paper ballot, and compare to the electronic record (as well as check many other things, as prescribed for the risk-limiting audit).

The use of hashed *bid*'s has the important benefit of making sure that someone who does not know a *bid* value cannot, by looking at the electronic record, link the selections made for the different races on a single ballot, which protects from pattern voting attacks. There is no need for such a protection from someone who can access the paper ballots, since that person can already link all races just by looking at the paper.

The full cryptographic protocol. The resulting cryptographic workflow is as follows.

(1) The trustees jointly generate a threshold public key/private key encryption pair. The encryption key K is published.
(2) Each voting terminal is initialized with the ballot and election parameters, the public key K and seeds z_0^p and z_0^i that are computed by hashing all election parameters and a public random salt z_0.
(3) When a voter completes the ballot marking process selection to produce a ballot v, the voting terminal performs the following operations:
    (a) It selects a unique and unpredictable ballot identifier *bid*, as well as a unique (but possibly predictable) ballot casting identifier *bcid*.

(b) It computes an encryption $c_v = E_K(v)$ of the vote, as well as a NIZK proof p_v that c_v is an encryption of a valid ballot. This proof is written in such a way that it can be verified from $Ext(c_v)$ only.
(c) For each race $r_1, \ldots, r_n$ to which the voter takes part, it computes an encryption $c_{bid} = E_K(bid\|r_1)\|\cdots\|E_K(bid\|r_n)$.
(d) It computes a public hash code $z_i^p = H(bcid\|Ext(c_v)\|p_v\|m\|z_{i-1}^p)$, where m is the voting terminal unique identifier, as well as an internal hash $z_i^i = H(bcid\|c_v\|p_v\|c_{bid}\|\|m\|z_{i-1}^i)$
(e) It prints a paper ballot in two parts. The first contains v in a human readable format as well as c_{bid} and *bcid* in a robust machine readable format (e.g., as barcodes). The second is a voter take-home receipt that includes, the voting terminal identifier m, the date and time, and the hash code z_i^p (or a truncation thereof), all in a human-readable format.
(f) It transmits $(bcid, c_v, p_v, c_{bid}, m, z_i^p, z_i^i)$ to the judge's station.

(4) When a ballot is cast, the ballot casting id *bcid* is scanned and sent to the judge's station. The judge's station then marks the associated ballot as cast and ready to be included in the tally. This information is also broadcast and added in the two hash chains.
(5) When the polls are closed, the tally is computed: the product of all cast encrypted votes is computed and verifiably decrypted, providing an election result.
(6) The data needed for the risk limiting audit is computed, as described above.

All the data included in the public hash chain are eventually digitally signed and published by the local authority. Those audit data are considered to be valid if the hash chain checks, if all cryptographic proofs check, that is, if the ballot validity proofs check, if the homomorphic aggregation of the committed votes is computed and opened correctly, and if all spoiled ballots are decrypted correctly.

Write-in votes. So far, we have not described how our cryptographic construction can support write-in voting. Support for write-in votes is required in many states. To be general-purpose, STAR-Vote adopts the vector-ballot approach [Kiayias and Yung 2004], wherein there is a separate homomorphic counter for the write-in slot plus an encryption of the string in the write-in. If there are enough write-in votes to influence the election outcome, then the write-in slots, across the whole election, will be mixed and tallied (together with the corresponding counters).

We note that, at least for elections in our state, write-in candidates must be registered in advance. It's conceivable that we could simply allocate a separate homomorphic counter for each registered candidate and have the STAR-Vote terminal help the voter select the desired "write-in" candidate. Such an approach could have significant usability benefits but is expected to require some update of regulations.

7. THREATS

To evaluate the design and engineering of STAR-Vote, it's helpful to have a threat model in mind. The obvious place to start would be VoteBox [Sandler et al. 2008], which is closely related to STAR-Vote. The original VoteBox authors did not state a concise threat model, but considered several kinds of threats and security design goals:

Software independence. STAR-Vote, like VoteBox, should be able to produce a proof of the correctness of an election that does not require any statement about the correctness of the software used in STAR-Vote. VoteBox achieved this through end-to-end cryptographic means. STAR-Vote uses similar cryptography and adds a risk-limiting audit that can verify the correspondence between STAR-Vote printed ballot records and their electronic counterparts, adding a degree of flexibility if the cryptography cannot prove an exact correspondence to determine exactly what went wrong.

Reduced trusted computing base. STAR-Vote, like VoteBox or any other software artifact, would benefit from having simpler code and less of it. This makes it easier to verify and less likely to have bugs. Software independence means that STAR-Vote's software is not required for *cor-*

rectness of the election outcome, but it does help defeat attacks which could disable the system, destroy records, or otherwise cause grief to election officials running STAR-Vote. VoteBox specifies that it uses pre-rendered user interfaces [Yee 2007; Yee et al. 2006]. STAR-Vote should probably use this technique as well.

Robustness against data loss. STAR-Vote, like VoteBox, specifies that vote records be stored on every voting terminal in the local polling place, using tamper-evident logging techniques. STAR-Vote adds a printed ballot record, stored in a ballot box. VoteBox went a step further by considering the real-time one-way transfer of vote records out of the polling place, across the Internet, to a central election headquarters. While STAR-Vote could add this in the future, it's not part of the initial design.

Mega attacks. In the VoteBox paper, the authors considered a variety of attacks with highly capable attackers. Such attackers might run a concurrent election on parallel equipment, in an attempt to substitute the results for genuine votes. Other attackers might mount a "booth capture" attack, wherein armed gunmen take over a polling place and cast votes as fast as possible until the police arrive. These attacks, needless to say, are well within the ability of STAR-Vote's cryptographic and risk-limiting infrastructure to detect. The best such attackers can hope to do is, in effect, mount a denial of service attack against the election. Attackers with that as their goal can arrive at much simpler approaches and STAR-Vote has relatively little it can offer beyond any other election system in this regard.

A full consideration of threats to STAR-Vote and their corresponding countermeasures or mitigations would be far too long to fit in this paper. Instead, we focus on several areas where STAR-Vote differs from other e2e voting systems in the literature.

7.1. Coercion

In designing STAR-Vote, we made several explicit decisions regarding how much to complicate the protocol and impede the voter experience in order to mitigate known coercion threats. Specifically, one known threat is that a voter is instructed to create a ballot in a particular way but to then execute a decision to cast or spoil the ballot according to some stimulus received after the ballot has been completed and the receipt has been generated. The stimulus could come, for example, from subtle motions by a coercer in the poll site, the vibration of a cell phone in silent mode, or some of the (unpredictable) data that is printed on the voter's receipt. Some prior protocols have required that the receipt, although committed to by the voting device, not be visible to the voter until after a cast or spoil decision has been made (perhaps by printing the receipt face down behind a glass barrier) and configuring poll sites so that voters cannot see or be seen by members of the public until after they have completed all steps. We could insist on similar measures here, but in an era where cell phones with video recording capabilities are ubiquitous and eyeglasses with embedded video cameras can easily be purchased, it seems unwise to *require* elaborate measures which mitigate some coercion threats but leave others unaddressed.

7.1.1. Chain voting. A similar threat of "chain voting" is possible with this system wherein a voter early in the day is instructed to neither cast nor spoil a ballot but to instead leave the poll site with a printed ballot completed in a specified way. This completed ballot is delivered to a coercer who will then give this ballot to the next voter with instructions to cast the ballot and return with a new printed ballot—again completed as specified. Chain voting can be mitigated by instituting timeouts which automatically spoil ballots that have not been cast within a fixed period after having been printed. We also expect to have procedures in place to prevent voters from accidentally leaving poll sites with printed ballots. We note that the timeout period need only cover the time we expect will be required for a voter to cross the room with a printed ballot and place it in the box, allowing for a relatively tight time bound, probably less than 5 minutes, although we'd need to run this in practice to understand the distribution of times that might happen in the real world.

(We note that traditional paper ballots sometimes include a perforated header section which includes a serial number. A poll worker keeps one copy of this number and verifies that the ballot a

voter wishes to cast matches the expected serial number. If so, the serial number is then detached from the ballot and deposited in the box. STAR-Vote could support this, but we believe it would damage STAR-Vote's usability. The timeout mechanism seems like an adequate mitigation.)

We do, however, take measures to prevent wholesale coercion attacks such as those that may be enabled by pattern voting. For instance, The SOBA audit process is explicitly designed to prevent pattern-voting attacks; and the high assurances in the accuracy of the tally are acheived without ever publishing the full set of raw ballots.

An interesting concern is that our paper ballots have data on them to connect them to electronic ballot records from the voting terminals and judge's console. The very data that links a paper ballot to an electronic, encrypted ballot creates a potential vulnerability. Since some individual paper ballot summaries will be selected for post-election audit and made public at that time, we are careful to not include any data on the voter's take-home receipt which can be associated with the corresonding paper ballot summary.

7.1.2. Absentee and provisional ballots. There are several methods available for incorporating ballots which are not cast within the STAR-Vote system, such as absentee and provisional ballots. The simplest approach is to completely segregate votes and tallies, but this has several disadvantages, including a reduction in voter privacy and much lower assurance of the accuracy of the combined tally.

It may be possible to eliminate all "external" votes by providing electronic means for capturing provisional and remote ballots. However, for the initial design of the STAR-Vote system, we have chosen to avoid this complexity. Instead, we ask that voting officials receive external votes and enter them into the STAR-Vote system as a proxy for voters. While this still does not allow remote voters to audit their own ballots, the privacy-preserving risk-limiting audit step is still able to detect any substantive deviations between the paper records of external voters and their electronically recorded ballots. This provides more supporting evidence of the veracity of the outcome without reducing voter privacy.

7.2. Further analysis

If we wished to conduct a more in-depth threat modeling exercise, one place to begin would be the threat model developed by the California Top To Bottom Review's source code audit teams (see, e.g., [Inguva et al. 2007]). They considered different levels of attacker access, ranging from voters to election officials. They also considered different attacker motives (disrupt elections, steal votes, coerce voters) and different attack outcomes (detectable vs. undetectable, recoverable vs. unrecoverable, prevention vs. detection, wholesale vs. retail, and casual vs. sophisticated). A complete consideration of STAR-Vote against all these criteria would take far more space than is available in this venue. Instead, we now focus on where STAR-Vote advances the state of the art in these areas.

Most notably, STAR-Vote's combination of end-to-end cryptography with risk-limiting audits of paper ballots is a game changer, in terms of thwarting attackers who might want to disrupt elections. Unlike paperless systems, STAR-Vote has the ability to fall back to the paper records, with efficient processes to detect when inconsistencies exist that would require this. This radically improves STAR-Vote's recoverability from extreme failures.

Similarly, while STAR-Vote is "software independent," we must concern ourselves with software tampering that does not change any of the cryptographic computations, but instead causes the STAR-Vote machines to silently record everything the voter does. This threat cannot be mitigated by better cryptography or ballot auditing. The only likely solution is some sort of trusted platform management (TPM), where the hardware will refused to run third-party code (more discussion on this appears in Section 3.3).

Lastly, we consider a threat that only arises in e2e systems: presentation of a fraudulent voting receipt. Consider the case where a voter may spoil her ballot and take it home to verify against the public bulletin board. A malicious voter with access to similar printers could produce a seemingly legitimate ballot for which there is no correspondence on the public bulletin board, thus "proving"

that the election system lost a record. Similar defaming attacks could be made by forging the receipt that a voter can take home after casting a ballot. For STAR-Vote, we have considered a number of mitigations against these attacks, ranging from cryptographic (having the voting terminals compute a digital signature, with protected key material) to procedural (e.g., watermarking the paper or having poll workers physically sign spoiled ballots). Real STAR-Vote deployments will inevitably use one or more of these mitigations.

8. CONCLUSIONS AND FUTURE WORK

In many ways, STAR-Vote is a straightforward evolution from existing commercial voting systems, like the Hart InterCivic eSlate, mixing in advanced cryptography, software engineering, usability, and auditing techniques from the research literature in a way that will go largely unnoticed by most voters, but that have huge impact on the reliability, accuracy, fraud-resistance, and transparency of elections. Of course, we can also take this opportunity to improve more pragmatic features, such as offering better support for the election administration's desired workflow. Clearly, we're long overdue for election systems engineered with all the knowledge we now have available.

STAR-Vote also opens the door to a variety of interesting future directions. For example, while STAR-Vote is intended to service any given county as an island unto itself, there's no reason why it cannot also support *remote voting*, where ballot definitions could be transmitted to a remote supervised kiosk, which securely returns the electronic and paper records. By virtue of STAR-Vote's cryptographic mechanisms, such a remote vote is really no different than a local provisional vote and can be resolved in a similar fashion, preserving the anonymity of the voter. (A variation on this idea was earlier proposed as the RemoteBox extension [Sandler and Wallach 2008] to VoteBox [Sandler et al. 2008].) This could have important ramifications for overseas and military voters with access to a suitable impromptu polling place, e.g., on a military base or in a consular office.

(We do not want to suggest that STAR-Vote would be suitable for *Internet* voting. Using computers of unknown provenance, with inevitable malware infections, and without any systematic way to prevent voter bribery or coercion, would be a foolhardy way to cast ballots. A STAR-Vote variant, running in a web browser and printing a paper ballot returned through the postal mail, might well be feasible as a replacement for current vote-by-mail practices. A full consideration of this is left for future work.)

STAR-Vote anticipates the possibility that voting machine hardware might be nothing more than commodity computers running custom software. It remains unclear whether off-the-shelf computers can be procured to satisfy all the requirements of voting systems (e.g., long-term storage without necessarily having any climate control, or having enough battery life to last for a full day of usage), but perhaps such configurations might be possible, saving money and improving the voting experience.

ACKNOWLEDGMENTS

Olivier Pereira's work was supported by the Belgian French Community through the SCOOP ARC project and by the European Commission through the HOME/2010/ISEC/AG/INT-011 B-CCENTRE project.

REFERENCES

Ben Adida. 2008. Helios: Web-based Open-Audit Voting. In *17th USENIX Security Symposium*. San Jose, CA.

Ben Adida, Oliver de Marneffe, Oliver Pereira, and Jean-Jaques Quisquater. 2009. Electing a University President using Open-Audit Voting: Analysis of real-world use of Helios. In *Proceedings of the Electronic Voting Technology Workshop / Workshop On Trustworthy Elections (EVT/WOTE 2009)*. Montreal.

Jonathan Ben-Nun, Niko Farhi, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. 2012. A New Implementation of a Dual (Paper and Cryptographic) Voting System. In *5th International Conference on Electronic Voting (EVOTE 2012)*. Lochau/Bregenz, Austria.

Josh Benaloh. 2006. Simple Verifiable Elections. In *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '06)*. Vancouver, B.C., Canada. http://www.usenix.org/events/evt06/tech/full_papers/benaloh/benaloh.pdf

Josh Benaloh. 2007. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*. Boston, MA.

J. Benaloh and M. Fischer. 1985. A robust and verifiable cryptographically secure election scheme. In *28th IEEE Symposium on Foundations of Computer Science (FOCS)*. Portland, OR, 372–382.

J. Benaloh, D. Jones, E. Lazarus, M. Lindeman, and P.B. Stark. 2011. SOBA: Secrecy-preserving Observable Ballot-level Audits. In *Proceedings of the 2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '11)*. USENIX. http://statistics.berkeley.edu/~stark/Preprints/soba11.pdf

Josh Benaloh and Eric Lazarus. 2011. *The Trash Attack: An Attack on Verifiable Voting Systems and a Simple Mitigation*. Technical Report MSR-TR-2011-115. Microsoft Technical Report.

David Bernhard, Olivier Pereira, and Bogdan Warinschi. 2012. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT 2012 (Lecture Notes in Computer Science)*, X. Wang and K. Sako (Eds.), Vol. 7658. Springer, 626–643.

J. Bretschneider, S. Flaherty, S. Goodman, M. Halvorson, R. Johnston, M. Lindeman, R.L. Rivest, P. Smith, and P.B. Stark. 2012. Risk-Limiting Post-Election Audits: Why and How. http://statistics.berkeley.edu/~stark/Preprints/RLAwhitepaper12.pdf. (2012).

Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. 2012. Using Prêt à Voter in Victorian State elections. In *2012 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '12)*. Bellevue, WA.

M. D. Byrne, K. K. Greene, and S. P. Everett. 2007. Usability of voting systems: Baseline data for paper, punch cards, and lever machines. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '07)*. ACM, San Jose, CA, 171–180.

Bryan A. Campbell and Michael D. Byrne. 2009a. Now do voters notice review screen anomalies? A look at voting system usability. In *2009 USENIX/ACCURATE Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)*. Montreal, Canada.

Bryan A. Campbell and Michael D. Byrne. 2009b. Straight party voting: What do voters think? *IEEE Transactions on Information Forensics and Security* 4, 4 (2009), 718–728.

Richard Carback, David Chaum, Jeremy Clark, Aleksander Essex, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. 2010. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *USENIX Security 2010*. Washington, DC.

David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y.A. Ryan, Emily Shen, and Alan T. Sherman. 2008. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *Electronic Voting Technology Workshop 2008*. San Jose, CA.

Veronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachene. 2013. A generic construction for voting correctness at minimum cost - Application to Helios. Cryptology ePrint Archive, Report 2013/177. (2013). http://eprint.iacr.org/.

R. Cramer, R. Gennaro, and B. Schoenmakers. 1997. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Advances in Cryptology - EUROCRYPT '97 (LNCS)*, Walter Fumy (Ed.), Vol. 1233. Springer, 103–118.

Edouard Cuvelier, Olivier Pereira, and Thomas Peters. 2013. Election Verifiability or Ballot Privacy: Do We Need to Choose? Cryptology ePrint Archive, Report 2013/216. (2013). http://eprint.iacr.org/.

Yvo Desmedt and Yair Frankel. 1989. Threshold Cryptosystems. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*. Santa Barbara, CA, 307–315. http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C89/307.PDF

T. ElGamal. 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 4 (1985), 469–472.

Sarah Everett, Kristen Greene, Michael Byrne, Dan Wallach, Kyle Derr, Daniel Sandler, and Ted Torous. 2008. Is Newer Always Better? The Usability of Electronic Voting Machines versus Traditional Methods. In *Proceedings of CHI 2008*. Florence, Italy. http://www.chi2008.org/ap/112.html

Sarah P. Everett. 2007. *The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection*. Ph.D. Dissertation. Rice University, Houston, TX.

Sarah P. Everett, Michael D. Byrne, and Kristen K. Greene. 2006. Measuring the usability of paper ballots: Efficiency, effectiveness, and satisfaction. In *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting*. Human Factors and Ergonomics Society, Santa Monica, CA, 2547–2551.

Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. 2007. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *J. Cryptology* 20, 1 (2007), 51–83.

K. K. Greene. 2008. *Usability of New Electronic Voting Systems and Traditional Methods: Comparisons Between Sequential and Direct Access Electronic Voting Interfaces, Paper Ballots, Punch Cards, and Lever Machines*. Master's thesis. Rice University, Houston, TX.

Srinivas Inguva, Eric Rescorla, Hovav Shacham, and Dan S. Wallach. 2007. *Source Code Review of the Hart InterCivic Voting System*. California Sec. of State's "Top to Bottom" Review. http://www.sos.ca.gov/elections/voting_systems/ttbr/Hart-source-public.pdf http://www.sos.ca.gov/elections/voting_systems/ttbr/Hart-source-public.pdf.

ISO. *Ergonomic requirements for office work with visual display terminal (VDT's)–Part 11: Guidance on usability*. ISO, Geneva, Switzerland. ISO 9241-11(E).

Aggelos Kiayias and Moti Yung. 2004. The Vector-Ballot E-Voting Approach. In *8th International Conference on Fianncial Cryptography (FC '04)*. Key West, Florida.

M. Lindeman, M. Halvorson, P. Smith, L. Garland, V. Addona, and D. McCrea. 2008. Principles and Best Practices for Post-Election Audits. www.electionaudits.org/files/best%20practices%20final_0.pdf. (2008). Retrieved April 20, 2011.

Mark Lindeman and Philip B. Stark. 2012. A Gentle Introduction to Risk-Limiting Audits. *IEEE Security and Privacy* 10 (2012), 42–49.

Gillian E. Piner and Michael D. Byrne. 2011. The experience of accessible voting: Results of a survey among legally blind users. In *Proceedings of the Human Factors and Ergonomics Society 54th Annual Meeting*. Human Factors and Ergonomics Society, Santa Monica, CA.

Niels Provos, Markus Friedl, and Peter Honeyman. 2003. Preventing Privilege Escalation. In *Proceedings of the 12th USENIX Security Symposium*. Washington, DC.

E. Rescorla. 2009. Understanding the Security Properties of Ballot-Based Verification Techniques. In *Proceedings of the 2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '10)*. http://www.usenix.org/event/evtwote09/tech/full_papers/rescorla-ballot.pdf Retrieved March 6, 2011.

Ronald L. Rivest and John P. Wack. 2006. On the Notion of "Software Independence" in Voting Systems. (2006). http://vote.nist.gov/SI-in-voting.pdf.

Peter Y. A. Ryan and T. Peacock. 2006. A Threat Analysis of Prêt à Voter. In *Workshop On Trustworthy Elections (WOTE 2006)*. Cambridge, U.K.

Daniel R. Sandler, Kyle Derr, and Dan S. Wallach. 2008. VoteBox: A tamper-evident, verifiable electronic voting system. In *Proceedings of the 17th USENIX Security Symposium*. San Jose, CA.

Daniel R. Sandler and Dan S. Wallach. 2007. Casting Votes in the Auditorium. In *Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*. Boston, MA. http://accurate-voting.org/wp-content/uploads/2007/08/evt07-sandler.pdf

Daniel R. Sandler and Dan S. Wallach. 2008. The case for networked remote voting precincts. In *Proceedings of the 3rd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'08)*. San Jose, CA.

B. Smith, S. J. Laskowski, and S. Lowry. 2009. Implications of Graphics on Usability and Accessibility for the Voter. In *Proceedings of the Second International Conference on E-Voting and Identity, VOTE-ID 2009*. Luxembourg, 54–74.

P.B. Stark. 2010. Super-Simple Simultaneous Single-Ballot Risk-Limiting Audits. In *Proceedings of the 2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '10)*. USENIX. http://www.usenix.org/events/evtwote10/tech/full_papers/Stark.pdf. Retrieved April 20, 2011.

Philip B. Stark and David A. Wagner. 2012. Evidence-Based Elections. *IEEE Security and Privacy* 10 (2012), 33–41.

Ka-Ping Yee. 2007. Extending prerendered-interface voting software to support accessibility and other ballot features. In *USENIX/ACCURATE Electronic Voting Technology Workshop 2007*. Boston, MA. http://pvote.org/docs/evt2007/paper.pdf

Ka-Ping Yee, David Wagner, Marti Hearst, and Steven M. Bellovin. 2006. Prerendered User Interfaces for Higher-Assurance Electronic Voting. In *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '06)*. Vancouver, B.C., Canada. http://zesty.ca/voting/prui/prui.pdf

Nickolai Zeldovich, Silas Boyd-Wickizer, and David Mazières. 2008. Securing Distributed Systems with Information Flow Control. In *Proceedings of the 5th Symposium on Networked Systems Design and Implementation (NSDI '08)*. San Francisco, CA.

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

How To Build an Undervoting Machine: Lessons from an Alternative Ballot Design

KRISTEN K. GREENE, RICE UNIVERSITY*
MICHAEL D. BYRNE, RICE UNIVERSITY
STEPHEN N. GOGGIN, UNIVERSITY OF CALIFORNIA

Despite the importance of usability in ensuring election integrity, it remains an under-studied aspect of voting systems. Voting computers (a.k.a. DREs) offer the opportunity to present ballots to voters in novel ways, yet this space has not been systematically explored. We constructed a DRE that, unlike most commercial DREs, does not require voters to view every race, but instead starts at the "review screen" and lets voters directly navigate to races. This was compared with a more traditional, sequentially-navigated, DRE. The direct access navigation model had two effects, both of which were quite large. First, voters made omission (undervote) errors markedly more often. Second, voters who were free to choose who to vote for chose to vote in substantially fewer races. We also examined the relationship between the true error rate—which is not observable in real elections—and the residual vote rate, a measure of effectiveness commonly used for real elections. Replicating the findings of [Campbell and Byrne 2009a], the mean residual vote rate was close to the mean true error rate, but the correlation between these measures was low, suggesting a loose coupling between these two measures.

1. INTRODUCTION

One of the most obviously desirable qualities of a voting system is that it accurately measures the intent of the voters. With the advent of direct recording electronic (DRE) voting machines, the concern for accuracy has been raised in the form of discussions about computer security, since most commercial DREs are susceptible to various form of attack, including computer viruses, e.g., [Ohio Secretary of State 2007]. While security concerns have led many states to move away from DREs towards optical scan paper-ballots, there are several compelling reasons why DREs remain an important research area. For example, with electronic technology, there exists significant potential for accessibility accommodations that traditional paper-based systems cannot offer; there will always be a need for systems that support advances in accessible technology. From a much broader research perspective, DREs offer the opportunity to present ballots to voters in novel ways, yet this space has not been systematically explored. Without understanding how specific design features—such as navigation style—impact usability of known voting systems, we will have no baseline data against which to evaluate emerging and future voting methods. For obvious reasons, experimental manipulations of potentially significant ballot design features are neither feasible nor desirable in real-world elections, lest the will of some voters be more accurately reflected than others. By conducting a mock election in a controlled setting, we were able to gather both objective and subjective usability data across multiple voting systems (two DREs with different navigation styles, paper ballots, punch cards, and lever machines) at a much finer level of granularity than is possible in a real election. The laboratory environment also allowed a comparison of true voter error with the indirect measure commonly used in political science, i.e., the residual vote.

Studies outside the laboratory, such as archival studies of voting records from enormous samples of counties across multiple elections utilize "residual votes" as the measure for error. [Stewart 2006] regards it as "a measure of voting machine accuracy that was first championed by the Caltech/MIT Voting Technology Project in 2001 and has been regularly used ever since..." These residual vote rate studies, while important, often neglect to examine the different types of voter errors, and utilize different methodologies for counting errors.

For instance, [Ansolabehere and Stewart 2005] found that the types of voting technologies in use have an impact on under- or overvote occurrences, or the residual vote rate. Ansolabehere and Stewart found clear differences between technologies in terms of the residual vote rate, with the results differing based on the location of the race on the ballot. Specifically, they found that punch

This research was supported by the National Science Foundation under grant #CNS-0524211 (the ACCURATE center).
*Now at the National Institute of Standards and Technology (NIST).

cards created the most residual votes for presidential races, yet lever machines produced the highest rates for down-ballot races. Optical scan ballots perform best at presidential, or top-of-the-ballot races, while there is a three-way tie between optical scan, DRE, and traditional paper ballots for lowest residual vote rate among down-ballot races. Importantly, Ansolabehere and Stewart (p. 378), unlike most other studies in Table 1, note the importance of a race's place on the ballot, stating that "which machine is 'best' appears to depend, in part, on which race you are studying." While [Ansolabehere and Stewart 2005], as well as [Traugott et al. 2005] note the differences in residual vote rate among different races on the ballot, neither study focuses on these differences.

These results highlight an important problem with using the measure of residual votes for the metric of effectiveness in studying voting machine usability. Because residual vote rates include both undervotes and invalid overvotes (or spoiled ballots), they are sensitive to the effects of ballot roll-off, and do not provide as precise a measure as possible in the laboratory. While ballot roll-off rates, estimated for DREs and voters in such articles as [Nichols and Strizek 1995] or [Kimball et al. 2001] could be controlled for in the residual vote rate, the residual vote rate is often not completely separated into its constituent parts. Any "switched," or "wrong choice" ballots, in which a vote was cast for any candidate(s) other than the candidate(s) the voter intended, are not captured as error at all. As [Kimball and Kropf 2008] noted, residual vote rates on propositional measures are much higher than races at the top of the ballot, and while this may be driven by rational undervoting, aggregate data does not allow the separation of the residual vote into intentional and unintentional undervotes, let alone wrong choice votes and overvotes.

Furthermore, in massive archival studies of voting records using residual vote rates, reporting methods differ based on county and state, and often different rules must be applied to calculate the residual vote rate based on the information given by the election officials. In nearly all studies, the measure of an undervote, is merely whether a voter left the first race on the ballot blank. Often, this is the Presidential race in a Presidential election year. While it is likely that many who leave this race blank likely do so unintentionally, it is still feasible voters abstain intentionally, which would then be measured as error.

Even if the residual vote rate is an accurate representation of the errors produced by voters in the election process, traditional studies of election technology in political science typically neglect the other two metrics of usability, efficiency and satisfaction. Usability is a multifaceted problem, as described clearly in a 2005 report from the National Institute of Standards and Technology (NIST) [Laskowski et al. 2005], which recommends voting systems be evaluated by the ISO standard metrics of usability: effectiveness (for voting systems, accuracy), efficiency, and satisfaction. While some studies, such as [Stein et al. 2008] have examined the time it takes a voter to cast a ballot in the field, no comprehensive field studies have been able to record measures for all three metrics for the same voters. While this difficulty is due to election law and the privacy of the voter, alternative methodologies are available for studying voting machine usability, such as in the laboratory.

In 2006, Byrne, Everett, Greene and colleagues began collecting baseline usability data for traditional voting methods. Across the series of three usability studies [Everett et al. 2006; Greene et al. 2006; Byrne et al. 2007], they gathered baseline usability data for multiple forms of paper ballots, punch cards, and lever voting machines. Their research showed that on all technologies, error rates were high, across these three studies and multiple voting methods, between 11 and 26% of the ballots contained at least one error and error rates per-race were generally between 1 and 4%. Furthermore, the differences between these technologies on objective measures of usability are not large, though they found some evidence that punch cards and lever machines are more error-prone for some populations, e.g., older voters tend to have more trouble with lever machines [Byrne et al. 2007]. However, these studies showed a small but consistent advantage for paper ballots in terms of subjective usability.

In 2008, Everett and colleagues [Everett et al. 2008] report on laboratory studies that compared the usability of a new prototype DRE versus traditional voting methods (paper ballots, punch cards, and lever machines). While there were few differences between the DRE and the older voting methods on efficiency or effectiveness, participants were substantially more satisfied when using the DRE. The bubble ballot received high SUS scores as it had in prior research, but

with the addition of a DRE, the bubble ballot was no longer participants' favorite voting method. Despite the large subjective advantage for DREs, participants' objective performance was not any better with the DRE. This disassociation between subjective and objective usability has been seen in a number of domains [Bailey 1995, Neilsen 1995], and may have ramifications for election officials who decide to return to traditional methods after having previously adopted DREs.

[Herrnson et al. 2008] reports another substantial usability evaluation; they evaluated six electronic voting systems and four verification systems using expert reviews, a laboratory experiment, and a field study. The six commercial DREs included an ES&S Model 100, a Diebold Accuvote-TS, an Avante Vote-trakker, a Hart InterCivic eSlate, a Nedap Liberty Vote, and a Zoomable prototype system developed specifically for this research. Participants were asked to read a voter guide and circle their choices, then use the marked-up guide while voting. However, even though they were asked to make their own choices, for some races they were instructed on who to vote for, and were also asked to intentionally omit one down-ballot race, change a vote, and do a write-in.

When using a standard office-bloc ballot design and performing no special tasks, voters cast their ballots accurately over 97% of the time. In this scenario, there was little difference between voting systems in terms of accuracy. However, attempting to select multiple candidates, change a selection, and vote straight-party ticket caused accuracy to drop sharply, down to the range of 80-90%. Most importantly, differences in accuracy between voting systems were then seen. For example, accuracy for the Hart InterCivic system was only 72% and accuracy for the Avante was much worse, at 50%. Soberingly, roughly 20% of voters cast ballots that were not completely accurate. Despite this, all six DREs were viewed favorably by voters, who had relatively high confidence with the commercial DREs. In fact, voters were more confident with touch screens than with paper, and also judged touch screens as more trustworthy than paper. [Herrnson et al. 2008]

Unfortunately, the 2008 study by Herrnson et al. did not evaluate the efficiency of the various systems; ballot completion times were not recorded. An additional study limitation was the use of a non-standardized questionnaire to assess voter satisfaction. Use of a standardized instrument, such as the System Usability Scale [Brooke 1996], would have facilitated comparison of results across studies and technologies. Finally, the experimental procedures used were somewhat artificial. Having participants make their own selections, yet instructing them to change those selections in specific ways, is not a task scenario that is representative of what most voters experience in a real election. Despite these methodological limitations, the 2008 study by Herrnson et al. is an important landmark it the study of voting system usability and provides an important reference point for future research, including its focus on the separation of types of voter errors to include undervotes, overvotes, and "wrong choice" votes.

So, while there has been some recent research which has advanced our understanding of voting system usability, a great deal more research is needed. Our current research is aimed at two issues: first, we want to widen the space of inquiry into DRE design. The [Herrnson et al. 2008] research demonstrates that the differences between various DREs, while meaningful, are mostly not large (with some exceptions), and some of these differences are almost certainly exacerbated by the requirement that voters make a selection and then change it. While looking at commercial DREs is certainly valuable, it is also limited because most commercial DREs use similar designs for many functions. For instance, most DREs use touchscreen-activated textual buttons rather than alternatives like handwriting recognition, presentation of images rather than text, or other non-traditional designs. These design features are likely motivated by a desire (sometimes driven by state laws regarding ballot design) to keep the voting experience isomorphic to more traditional presentation on a paper ballot, but there are aspects of the paper ballot experience that are not captured in commercial DREs. In particular, most major commercial DREs present the ballot sequentially: that is, the race for Representative follows the race for Senator which follows the race for President. While a paper ballot typically also presents races in this order, voters can typically see all (or most) of the races simultaneously and can make their selections in whatever

order they wish.[1] Does giving voters the same flexibility with a DRE alter usability? It is clear that navigability is a critical factor in the usability of other systems such as Web sites—perhaps this applies to ballots as well.

Second, we want to further clarify the relationship between residual votes and true error rates through experimental work. Residual vote rate is an indirect measure of error; in an experiment, error can be more directly assessed. How well does residual vote rate compare to true error rate? Some work on this has already been published. e.g., [Campbell and Byrne 2009] showing that while the overall average residual vote and true error rates may be similar, the variability in one is not mirrored by the other. That is, when ballots are aggregated, the total residual vote proportion and the true error proportion may be similar, but this is not true at the individual ballot level; the correlation between the residual vote rate and the true error rate was found to be low. We wanted to further explore this relationship.

2. EXPERIMENT

2.1 Method

Participants. Sixty-four participants (30 male, 34 female) from the greater Houston, TX area were recruited through newspaper and online ads for participation in a mock election study. Ages ranged from 18 to 77 years, with a mean age of 50.3 (SD = 14.8). There was reasonable variability in terms of participant ethnicity, annual income, and education.

Design. The primary independent variable of interest was navigation style for the DRE, which was manipulated between subjects with random assignment. In addition to the sequential navigation style used in most commercial DREs and other studies—e.g. [Everett et al. 2008]—a "direct access" navigation style was also used. Half of participants voted with a sequential DRE; the other half voted with a direct access DRE. Each participant voted twice with either the sequential or the direct access DRE system. Both DREs had the same 27 contests, comprised of 21 candidate races and six propositions. The direct access DRE was similar to a webpage, in that all race titles appeared on its Main Page and acted as hyperlinks; see Figure 1. From the Main Page, a voter could click on the race titles to pick and choose exactly the races s/he wanted to see, with the option to skip races and go straight to the Record Vote screen at any time; see Figures 2 and 3 for a candidate race screen and the Record Vote screen, respectively. Note that merely moving from the Main Page to the Record Vote screen did not cast the ballot. The Record Vote screen presented the voter with buttons allowing the voter to either return to the Main Page or to cast his/her vote by selecting the "Record Vote" button (Figure 3).

In sharp contrast to the "pick and choose" navigation model of the direct access DRE, the sequential DRE forced voters to page sequentially through every race on the ballot—followed by a Review Choices screen much like the direct access DRE's Main Page—before they could get to the Record Vote screen. While the overall navigation schemes differed substantially between DREs, their Record Vote screens were nearly identical.

In addition to voting twice on a DRE, each subject voted on one of three other non-DRE methods: paper ballots, punch cards, or lever machines. Participants voted on the non-DRE system in between the two votes with the DRE. The assignment of participants to each level of this between-subjects variable was random.

A third independent variable was termed "information condition" and had four levels; participants were randomly assigned to one of these four. One level was termed "undirected;" in this condition participants were given a fictional voter guide modeled after the guides provided by the League of Women Voters, identical to the one first used in [Greene et al. 2006]. This guide

[1] It is, in principle, possible to present all of the races simultaneously on a DRE, but doing so for many U.S. elections would require extremely large, and therefore prohibitively expensive, computer displays. Machines such as the AVC Advantage and the Nedap LibertyVote do in fact do this, using a display of lights paired with paper-printed text on a large face, controlled by a computer. These machines, while DREs, largely resemble other styles of traditional voting machines, such as lever-based machines, as the user interface is not a traditional computer display.

provided specific information about each candidate for office, in addition to arguments for and against the propositions on the ballot. Participants read this guide and then chose for themselves the candidates and propositions to vote for. The remaining three conditions were "directed" in that participants were given a list of candidates to vote for. In the "directed with no roll-off" condition, this list was complete, instructing voters to cast a vote in every race on the ballot. Other levels instructed voters to abstain in some races. "Directed with moderate roll-off" was based on the [Nichols and Strizek 1995] data on roll-off rates. We used the average roll-off rates for jurisdictions with electronic voting machines for the various offices: for federal offices (9.75%), for state offices (12.00%) and for county offices (16.37%). This was designed to more closely mimic real-world voting patterns in which people do not vote for every race on the ballot. The "directed with additional roll-off" condition used lists where each race had a 52% chance of being omitted in order to simulate voters who only vote in a few races in each election.

A more complete listing of the materials including screen shots for most of the DRE screens, the full ballot, the full voter's guide, and example slates can be found at <http://chil.rice.edu/research/jets13/>.

The three dependent variables measured were ballot completion time, errors, and satisfaction. Ballot completion time was measured in seconds, and indicated how long it took participants to complete each of their three ballots. In the directed information conditions, errors were recorded if a participant chose a candidate other than the one they were instructed to select, if they failed to make a choice for a race in which they were supposed to vote, or if they made a choice in a race they were supposed to skip. In the undirected information condition, errors were recorded if there was a discrepancy between responses on a subject's three measures. Finally, satisfaction was measured using the System Usability Scale (SUS) [Brooke 2006], a ten-item battery of questions regarding subjective usability. Participants filled out three separate SUS questionnaires, one for each ballot they completed.

Errors were further broken down by error type: an error was a *wrong choice error* if the option selected was the incorrect one (e.g., a vote for Alice when a vote for Bob was intended). If a voter chose a candidate for a race s/he had planned to (or was instructed to) omit, this was considered an *extra vote error*. If a voter omitted a race that required a vote, this was considered an *omission error*. Finally, on the paper or punchcard systems, voters could vote for more choices than allowed, this was an *overvote error*.

In contrast with an undervote error, an *intentional abstention* occurred when a voter correctly abstained from voting in a race s/he had intended to skip. This is not an error, but is an important measure of voter behavior, as these would be counted as errors in the residual vote rate.

Procedures. Participants were first given a set of written instructions explaining the purpose of the study, accompanied by either a slate of candidates they were directed to vote for or a fictional voter guide. If given the guide, the participants were given as much time as they wanted to familiarize themselves with the political candidates and propositions. Once the participant indicated he or she was ready to begin, several key instructions were reiterated verbally. Participants were instructed to vote for the same candidates on all ballots.

Participants voted first on a DRE, then on one of the three non-DREs, then again on the DRE. This did away with the need for an exit interview in the undirected information condition. Finally, the participants filled out a longer questionnaire about their voting history and demographic information.

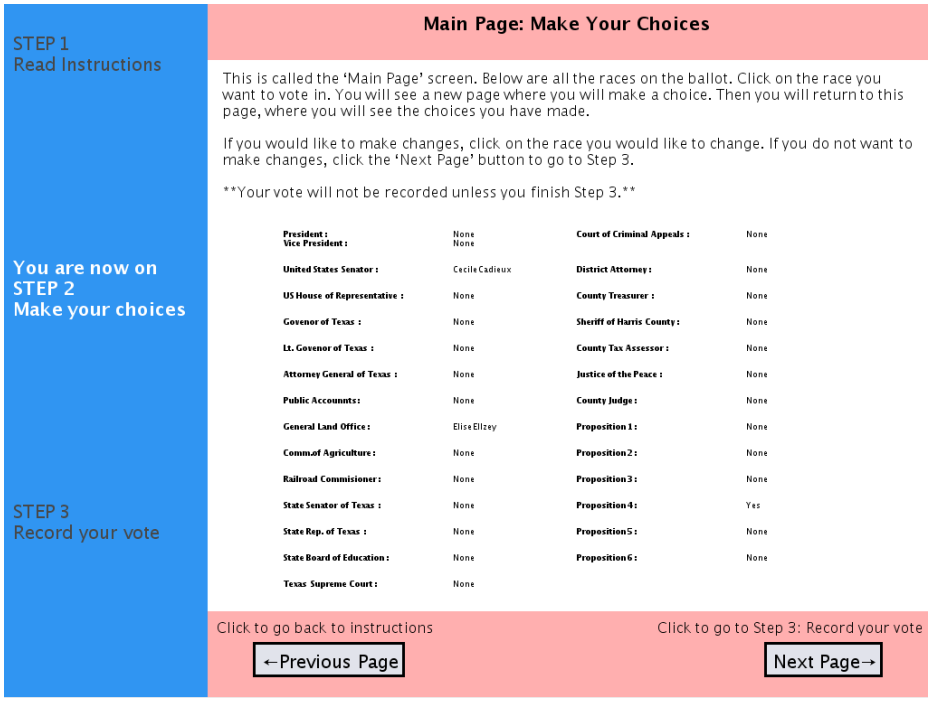


Fig. 1. "Main Page" for the Direct Access DRE.

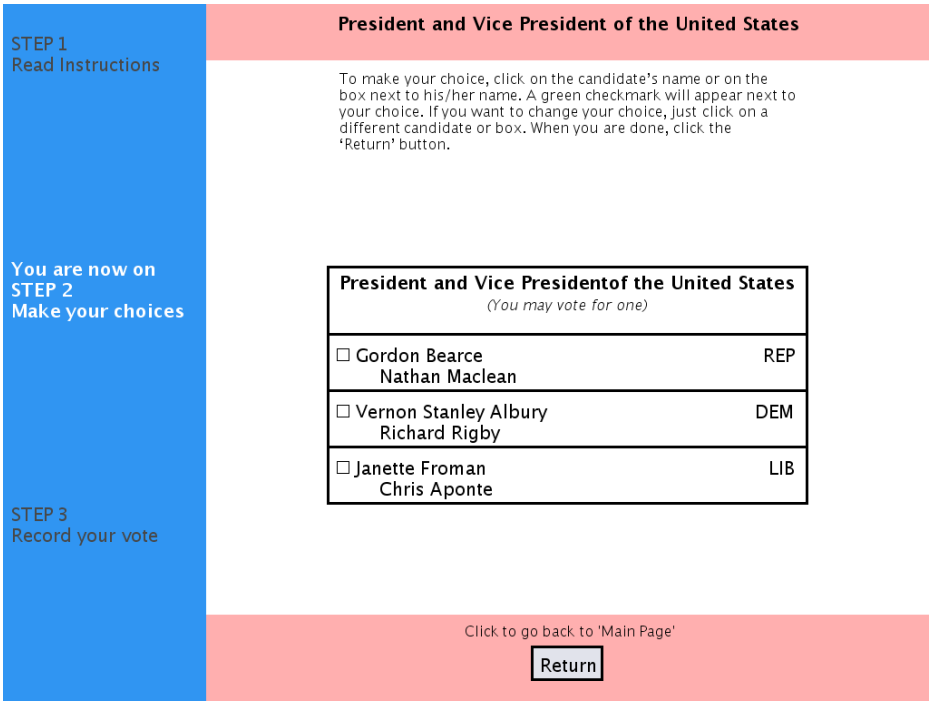


Fig. 2. Candidate Race Screen for the Direct Access DRE.

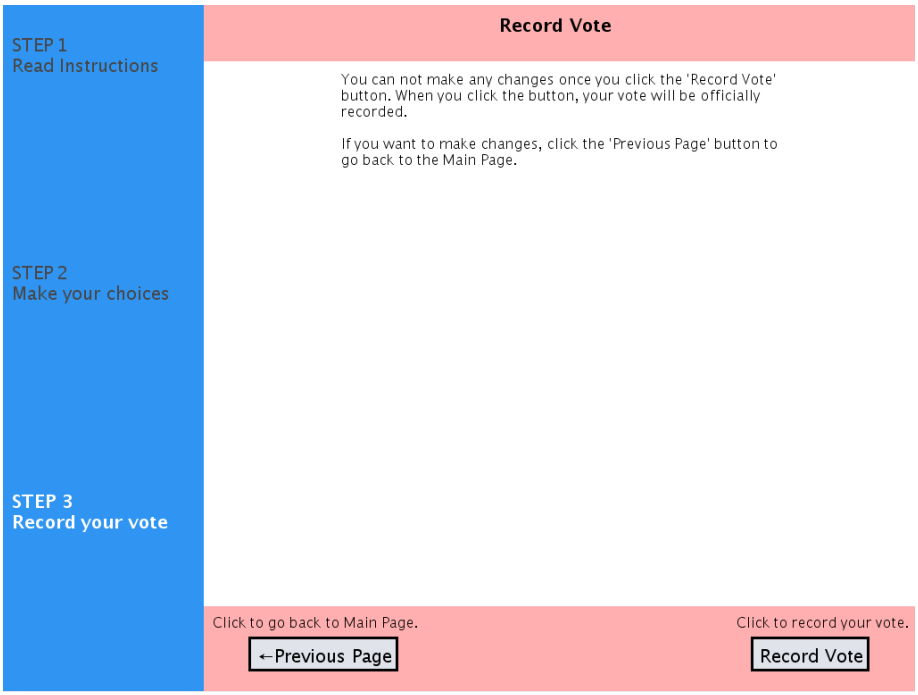


Fig. 3. Record Vote Screen for the Direct Access DRE.

2.2 Results

Efficiency. Data from six participants were not included in the efficiency analyses due to being outliers, defined as observations falling outside three interquartile ranges (IQRs) from the 25th or 75th percentiles of the distribution of ballot completion times. The most striking effects of navigation type on efficiency were seen in the undirected information condition, where ballot completion times for the sequential DRE were over twice as long as times for the direct access DRE: 453 seconds (SD=123) versus only 205 seconds (SD=119) . The interaction between navigation type and information condition, depicted in Figure 4, was statistically reliable, $F(3, 34) = 5.26$, $p = .004$, as was the main effect of navigation type, $F(1, 34) = 11.4$, $p = .002$. This result is inconsistent with previous results in that effect of voting system have generally been absent (or small); however, this result should not be considered in isolation from results on efficiency and intentional undervotes.

While navigation type had a significant impact on efficiency, type of non-DRE voting method used did not.

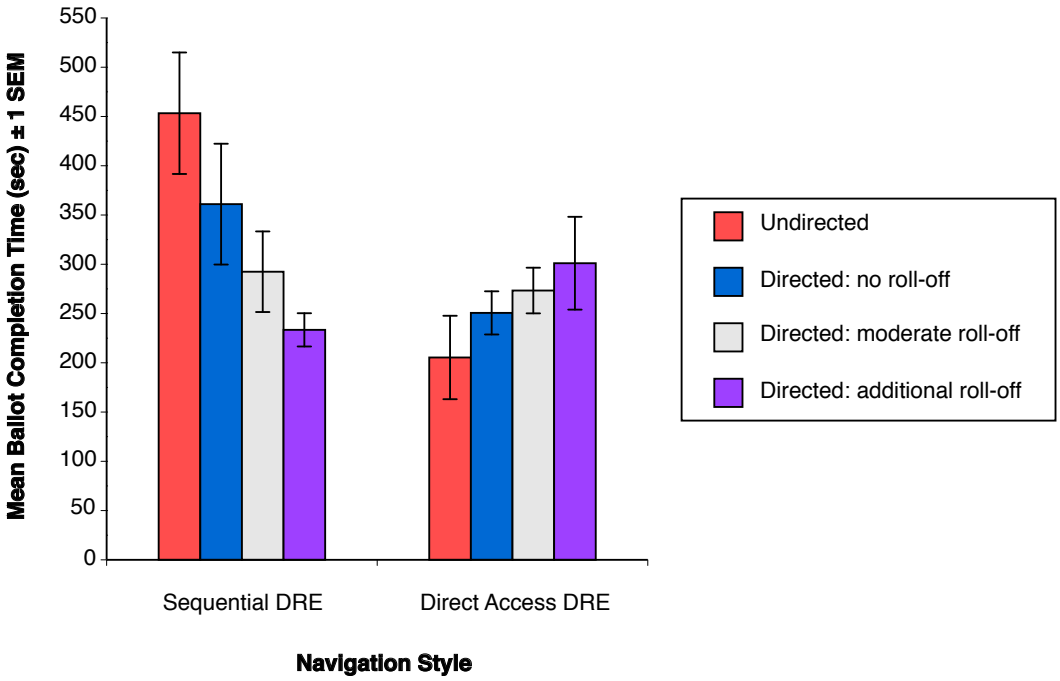


Fig. 4. Effects of DRE navigation style and information condition on ballot completion times

Effectiveness. Data from six participants were not included in the effectiveness analyses due to being outliers, defined as having greater than 15% errors on all three ballots they completed. This definition is in keeping with previous work [Byrne et al. 2007]. No overvotes were observed. Error rates for each voting technology are presented in Table 1.

Table 1. Mean Error Rates by Voting Technology and Type of Error

Technology	Extra vote	Omission	Wrong choice	Total error
Sequential DRE	0.0%	0.2%	1.1%	1.3%
Direct Access DRE	0.2%	13.1%	1.2%	14.5%
Bubble ballot	0.0%	0.2%	0.2%	0.4%
Lever machine	0.0%	0.6%	1.1%	1.7%
Punch card	0.0%	0.0%	0.2%	0.2%

Most interesting was the extremely high undervote error rate for the direct access DRE, which was about 13%, as opposed to only 0.2% for the sequential DRE. The main effect of navigation type on error rates was reliable, $F(1, 17) = 7.39$, $p = .02$. The abnormally high undervote error rate for the direct access DRE can be explained by the number of people who cast their ballots prematurely with the direct access DRE. In comparison with more traditional voting methods, DREs offer opportunities for voters to commit two particularly severe errors. A voter can fail to cast their vote entirely by not pressing the "Record Vote" button at all. This happens in real elections, and has been termed the "fleeing voter" problem [Felten 2009]. In a real election, when a voter fails to cast their vote, there is still a chance that the next voter or a poll worker will cast it for them. Of course, this is a function of the behavior of the next voter, the poll worker, and voting laws in the jurisdiction. A malicious next voter could see that the machine was left in an un-cast

state, change all the selections, and effectively vote twice, though this is illegal in most jurisdictions. Alternatively, the voter could get a poll worker, who in some jurisdictions is allowed to cast the vote, and in others is required to cancel the vote, therefore disenfranchising the voter who fled.

In this study, if a participant failed to cast their ballot, we cast it for them and counted their choices as intended. In contrast with failing to cast a vote at all, a voter can cast their vote prematurely, by pressing the "Record Vote" button too soon. This is also quite problematic, for when a vote is cast prematurely, voters irreversibly rob themselves of the opportunity to vote in some or all of the races on a ballot.

Of the 32 people who used the sequential DRE, only two people failed to cast their vote (6.3%), and not a single person cast their vote too soon. However, of the 32 people who used the direct access DRE, four people failed to cast their vote (12.5%), and eight people cast their vote too soon (25%). The number of races erroneously omitted due to premature ballot casting varied, and in several cases, no choices at all had been made when the ballot was cast. The direct access DRE had a significantly greater "cast too soon" error rate than did the sequential, $F(1, 62) = 4.33$, $p = .002$.

Intentional Abstentions. In a real election, it is impossible to discern whether an omission was an error on the part of the voter, or whether it was intentional. The controlled nature of this mock election allowed a distinction to be made between undervote errors and intentional undervotes. The intentional abstention rates reported here are for the undirected information condition only. Such rates are not of interest for the directed information conditions, in which participants were not allowed to make their own decisions regarding abstentions.

There was quite a large disparity in intentional undervote rates between the sequential and direct access DREs. Participants who used the sequential DRE almost never abstained from a race, resulting in an intentional abstention rate of 0.7% for those voters. In sharp contrast, those who used the direct access DRE abstained from nearly half of all races, with an intentional undervote rate of 45.4%, a dramatic difference, $F(1, 14) = 6.94$, $p = .02$.

Residual Vote versus True Error Rate. Of great interest was the relationship between the study's true error rate (a direct measure of effectiveness) with what would have been reported as the residual vote (an indirect measure) in a real-world election. The residual vote in our analysis is comprised of overvote errors, omission errors, and intentional abstentions. The residual vote does not include any information about wrong choice errors because it is impossible to identify such errors without knowing voter intent, something which is impractical to do in real elections due to privacy concerns. For the same reason, the residual vote cannot differentiate between omission errors and intentional abstentions. The residual vote rate was then compared to our measure of the true overall error rate, i.e. the "total" error rate, which was comprised of extra vote errors, overvotes, omission errors, and wrong choice errors combined. Comparisons between the residual vote rate versus the true error rate were done for the undirected information condition only, as voters the directed conditions did not have the option to choose their own abstention opportunities. It is common to compare accuracy between voting systems by focusing on the top-ballot residual vote rate, since intentional abstention rates are known to increase for down-ticket races. Therefore, comparisons between what would have been reported as the residual vote rate versus the true error rate are shown separately for the Presidential race (Figure 5) and all other down-ballot races (Figure 6). Given the extremely large intentional abstention rate for those voters using the direct access DRE, residual vote rates and error rates are also broken down by navigation type.

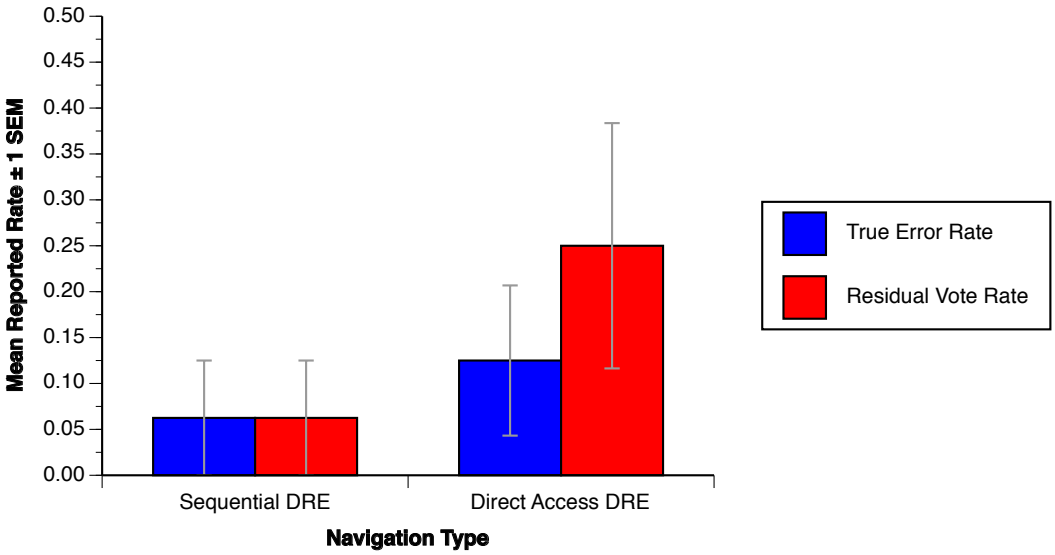


Fig. 5. Top-ballot (e.g., Presidential) residual votes and true error rates for the sequential navigation DRE and the direct access navigation DRE

The correlation between the residual vote and true error rate for the Presidential race was not significant for sequential navigation, $r(6) = -.14$, $p = .74$, nor was the difference between means reliable, $t(7) = 0.00$, $p = 1.00$. A similar pattern of results for the presidential race was observed for direct access navigation: the correlation between the residual vote and true error rate was not significant, $r(6) = .41$, $p = .32$, nor was the difference in means, $t(7) = 1.00$, $p = .35$.

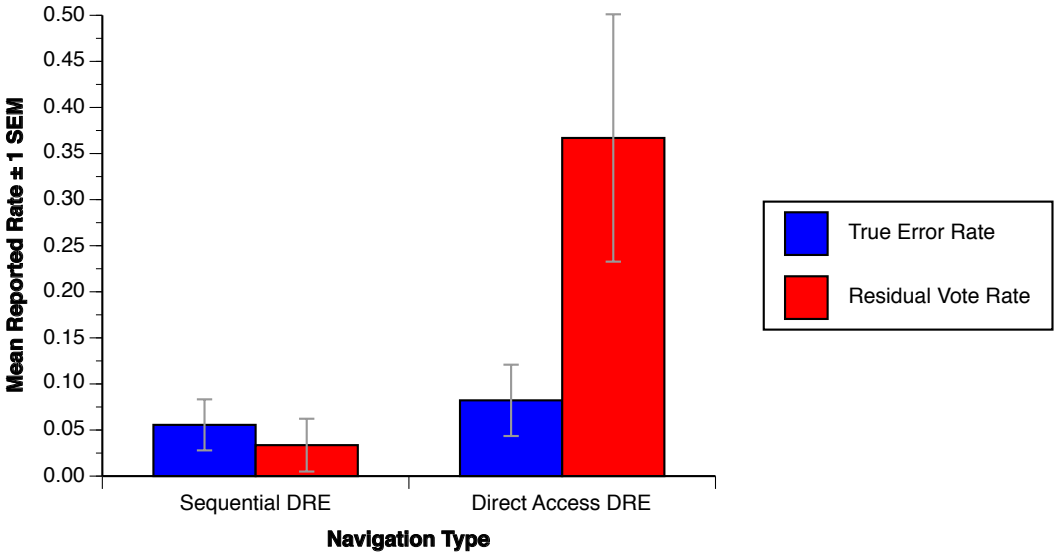


Fig. 6. Down-ballot residual votes and true error rates for the sequential navigation DRE and the direct access navigation DRE

Similarly, the correlation between the residual vote and true error rate for the remaining down-ballot races was not significant when navigation type was sequential, $r(6) = .22$, $p = .60$. Again, the difference in means was not reliable, $t(7) = .43$, $p = .68$. When navigation type was direct access, the correlation between the down-ballot residual vote and true error rates was not significant, $r(6) = .16$, $p = .70$, nor was there a significant difference in means, $t(7) = 1.64$, $p = .15$.

These non-significant correlations suggest the residual vote may not be as tightly coupled to the true error rate as has often been assumed.

Satisfaction. Data from two participants were excluded due to being outliers, defined as any point falling outside three IQRs from the 25th or 75th percentiles of the SUS scores distribution.

The sequential DRE consistently received higher SUS scores than did the direct access DRE, see Figure 7; the effect of navigation type on SUS scores was reliable, $F(1, 38) = 9.53$, $p = .004$. Although differences in SUS scores were seen as a result of changing DRE navigation style, information condition did not significantly impact subjective usability.

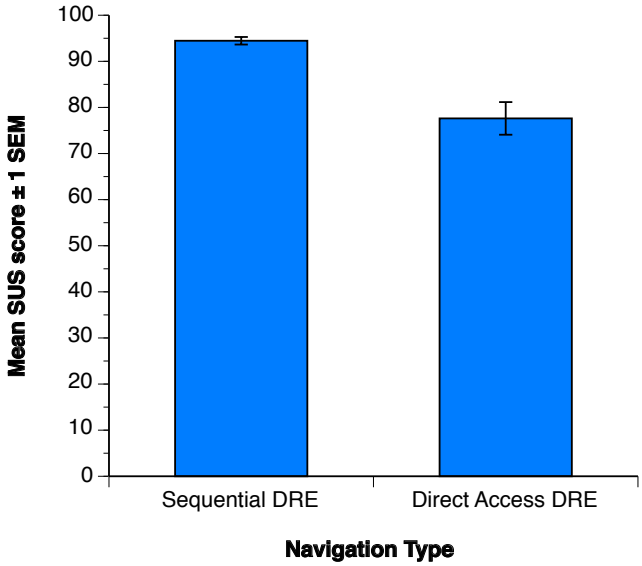


Fig. 7. Mean subjective usability (SUS) scores for sequential vs. direct access DREs

Consistent with multiple other studies, the sequential DRE received the highest mean SUS score of any voting method; ratings for the non-DRE voting methods overall were lower than ratings for the DREs. This is depicted in Figure 8. Unsurprisingly, all three pairwise differences between the DRE and the other methods were statistically reliable, $p < .05$ in each case.

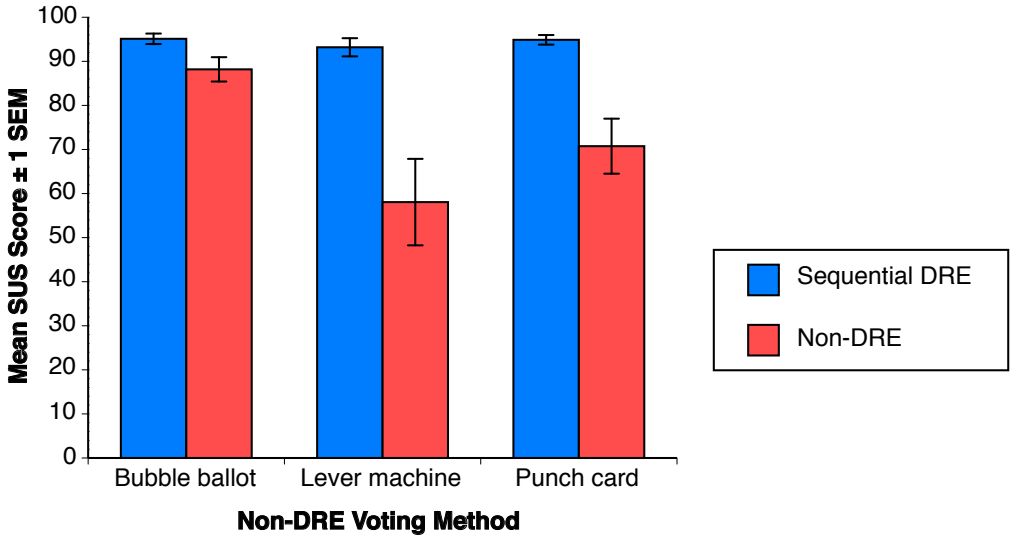


Fig. 8. Mean SUS scores for sequential DRE versus non-DRE voting methods

3. DISCUSSION

Exploration of alternatives in terms of ballot presentation has the potential to be valuable. From a user interface perspective, many DRE systems are essentially electronic implementations of paper ballots and do not take advantage of alternatives made possible by the electronic medium, such as alternate navigation models. We attempted to make a foray into this space, and examined a navigation style more like the Web. However, manipulation of a that single design feature—how the ballot was navigated—resulted in substantial effects on both subjective and objective usability. The lesson here is that changes in one aspect of the voting interface can have far-reaching consequences in terms of overall system usability. This should raise serious concerns about other purportedly small changes in voting procedures—such changes can have substantial impacts and no such changes should be deployed without being thoroughly tested for their effects on usability. For instance, there are a variety of proposals in the literature for voting systems with "end to end" verifiability (e.g., Prêt à Voter [Ryan, et al. 2009]) that introduce changes to the voting procedure. The usability ramifications of these changes have not been thoroughly tested, and it is not unreasonable to wonder if they have similar impacts.

In our case, intentional abstention rates, efficiency, effectiveness, and satisfaction were all greatly impacted simply by changing the DRE navigation style from sequential to direct access. When allowed to choose candidates for themselves in the undirected information condition, participants with the sequential DRE voted in almost every race, whereas participants using the direct access DRE abstained from nearly half of all races. Voting in so few races made the direct access DRE significantly faster than the sequential, but this increase in speed came at a tremendously high cost in accuracy. The direct access DRE had a much greater omission error rate than any of the other voting methods, and was plagued by a new type of error: voters frequently cast their ballots prematurely with the direct access DRE. While some voters cast their ballot too early, others failed to cast their ballot at all, replicating the real-world "fleeing voter" phenomenon in the laboratory.

That the results were so large was certainly unexpected. The change in navigation model generated substantial amounts of undervotes, both erroneous omissions and intentional abstentions. It is possible that some kind of visual cue to highlight races that had not yet been voted, such as the bright orange coloration employed by [Campbell and Byrne 2009a] to improve anomaly detection might also better alert voters to those races that they had not yet voted in. However, this seems somewhat unlikely, as when the voter first starts using the interface, *all* races would be flagged this way, so nothing would stand out. This is most likely to help only when there are a few isolated incomplete races. This does not imply that all visual UI manipulations would in principle be ineffective, but it seems unlikely that there is a simple visual fix on the main screen.

We strongly suspect that it is the sequential presentation that forces voters to see every single race that encourages voters to consider each race, making them much more likely to choose to cast a vote. However, this raises issues of scale. Our ballot was designed to represent an average length ballot in a U.S. election, but there is considerable variance in this measure and in some jurisdictions, ballots can be more than 100 contests long. How such a long ballot would be handled in a direct access DRE is not entirely clear. In order to maintain a readable font size, races would either have to be broken across multiple pages or the interface would have to support scrolling. Recent research [Campbell, 2013] suggests that a paginated display is likely to be better, but neither of these circumstances would be ideal.

One of the methods deployed in some jurisdictions to help alleviate the time taken for long ballots is the straight-party voting (SPV) option. This is currently available in roughly 16 U.S. states. This allows voters to make a single selection of a political party that then applies to all contests in which a member of that party is a valid option. There is evidence that the SPV option contributes to higher error rates on ballots [Campbell and Byrne 2009b, Herrnson et al. 2012] and it is not at all clear how SPV would interact with the direct access navigation model, as we did not provide an SPV option to voters in this experiment.

Given our result that the direct access DRE results in significantly more intentional abstention, one might argue this navigation style is not ideal, and that encouraging abstention is

not a desirable quality for a voting machine. The argument presented here is that we should ideally separate intentional abstention from unintentional undervoting, as including the former in residual error rates categorizes an intentional behavior with an error. What is an error is a normative argument contingent on the goals for our democratic instruments, and a highly debatable one. While some empirical work has examined hypothetical scenarios of election outcomes under full (hypothetically forced) turnout [e.g., Citrin et al. 2003], particularly Senate outcomes and not further down-ballot races, whether encouraging rational abstention should be a goal of an election system is a normative question we leave aside.

It is not immediately clear how this situation could be improved for any future direct access DRE. More visually salient highlighting of undervotes, as in [Campbell and Byrne, 2009a], might serve to better alert voters that some races have not been filled in. However, since many voters do abstain, particularly in down-race ballots, the issue is not simply whether there are any undervoted races, but whether only the intended abstentions are missing. Additional prompting of the voter when they attempt to cast the ballot (e.g., a warning message such as "There are contests for which you have made no selection. Are you sure you want to submit your ballot?") seems similarly ineffective exactly because some level of intentional abstention is not unusual. Sequential ballot presentation is likely a better solution.

This raises other interesting questions about what can or should be done to reduce unintentional abstention in other ballot formats. There is little that could be done with paper ballots (or punch cards), but sequential DREs could be programmed to not accept a ballot that did not have all races marked. In order to allow intentional abstentions, an explicit "abstain" option would have to be added to each contest. This style of ballot would likely run into legal problems in many U.S. jurisdictions. While as far as we know, no data on usability for such a system has ever been published, such a configuration could be quite onerous for voters, particularly those voters faced with many races who intend to abstain in most of the contests.

As noted in other research, participants were highly satisfied with their experience using the sequential DRE. Despite receiving significantly higher subjective usability ratings than any other method, there were no corresponding objective benefits to using the sequential DRE. Although voters preferred it, they were neither faster nor more accurate with it than with any of the older technologies.

Another issue that warrants closer scrutiny is the utility of the residual vote as a measure of accuracy. Consistent with [Campbell and Byrne 2009a], our data indicate that the residual vote and the true error rate may not be as tightly coupled as has traditionally been assumed. In both this experiment and Campbell and Byrne, the mean error rate generated by both measures was approximately the same. So, for use as a tool to detect large-scale problems across many voter, the residual error rate is indeed probably a good tool. It appears that, on average, the overall rate of wrong choice errors (included in the true error rate but not the residual vote rate) somewhat mirrors the intentional abstention rate (included in the residual vote rate but not the true error rate). There are probably circumstances where this breaks down, but as a low-precision tool, the use of residual vote rate is probably not a serious problem.

However, the near-zero correlation observed between residual vote rate and true error rate at the ballot level is cause for some concern. This suggests that the residual vote is incapable of identifying individual ballots, or probably even subsets of ballots, that contain errors. While wrong choice errors and intentional abstentions balance out in the aggregate, they do not do so at the level of individual ballots. Further research on the exact nature of this relationship is clearly warranted. Of course, this requires further laboratory research since the true error rate cannot be known in a real election for privacy reasons. Voting studies run in the laboratory environment, such as those reported here, complement field research and can shed light on issues that are impossible to address during real elections, such as the accuracy of the residual vote and evaluation of experimental voting system designs. The laboratory setting allows for more granular measurement of the objective and subjective facets of usability, both of which are important for meaningful comparisons of voting methods over time. Regardless of whether voting is electronic or paper-based, we must first understand the design space of our current systems before we can predict and evaluate the usability of emerging and future voting methods.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant #CNS-0524211 (the ACCURATE center). The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the NSF, the U.S. Government, or any other organization.

REFERENCES

Ansolabehere, S. and Stewart, C. 2005. Residual Votes Attributable to Technology. *The Journal of Politics*. 67, 2, 365–389.

Bailey, R. W. 1993 Performance vs. preference. I*n Proceedings of the Human Factors and Ergonomics Society 37th Annual Meet*ing, Human Factors and Ergonomics Society.

Brooke, J. 1996 SUS: A "quick and dirty" usability scale. I*n Usability Evaluation in Indust*ry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and A. L. McClelland, (Eds.). New York: Taylor and Francis.

Byrne, M. D., Greene, K. K., and Everett, S. P. 2007 Usability of voting systems: Baseline data for paper, punch cards, and lever machines. I*n Human Factors in Computing Systems: Proceedings of CHI 20*07, ACM.

Campbell, B. A. (2013). The Usability Implications of Long Ballot Content for Paper, Electronic, and Mobile Voting Systems. Doctoral dissertation, Rice University, Houston, TX.

Campbell, B. A. and Byrne, M. D. 2009a Now do voters notice review screen anomalies? A look at voting system usability. I*n 2009 USENIX/ACCURATE Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOT*E), Montreal, Canada.

Campbell, B. A. and Byrne, M. D. 2009b. Straight party voting: What do voters think*? IEEE Transactions on Information Forensics and Securi*ty. 4, 4, 718-728.

Citrin, J., Schickler, E., and Sides, J. 2003. What if everyone voted? Simulating the impact of increased turnout in Senate elections. *American Journal of Political Scien*ce. 47, 1, 75-90.

Everett, S. P., Byrne, M. D., and Greene, K. K. 2006 Measuring the usability of paper ballots: Efficiency, effectiveness, and satisfaction. I*n Proceedings of the Human Factors and Ergonomics Society 50th Annual Meet*ing, Human Factors and Ergonomics Society.

Everett, S. P., Greene, K. K., Byrne, M. D., Wallach, D. S., Derr, K., Sandler, D., and Torous, T. 2008 Electronic voting machines versus traditional methods: Improved preference, similar performance. I*n Human Factors in Computing Systems: Proceedings of CHI 20*08, ACM.

Felten, E. 2009. Finnish Court Orders Re-Vote After E-Voting Snafu. https://freedom-to-tinker.com/blog/felten/finnish-court-orders-re-vote-after-e-voting-snafu/

Greene, K. K., Byrne, M. D., and Everett, S. P. 2006. A comparison of usability between voting methods. *2006 USENIX/ACCURATE Electronic Voting Technology Worksh*op.

Herrnson, P. S., Hanmer, M. J., and Niemi, R. G. 2012. The impact of ballot type on voter errors. *American Journal of Political Scien*ce. 56, 3, 716–730.

Herrnson, P. S., Niemi, R. G., Hanmer, M. J., Bederson, B. B., Conrad, F. C., and Traugott, M. W. 200*8 Voting technology: The not-so-simple act of casting a ballo*t. Brookings Institution Press.

Kimball, D. C. and Kropf, M. 2008. Voting Technology, Ballot Measures, and Residual Votes. *American Politics Researc*h. 36, 479–509.

Kimball, D. C., Owens, C., and McAndrew, K. 2001. Who's Afraid of an Undervote*? Annual meeting of the Southern Political Science Associatio*n, Atlanta, GA.

Laskowski, S. J., Autry, M., Cugini, J., Killam, W., and Yen, J. 2004. *Improving the Usability and Accessibility of Voting Systems and Produc*ts. NIST Special Publication.

Mebane, W. R. 2004. The wrong man is president! Overvotes in the 2000 presidential election in Florida. *Perspectives on Politics*. 2, 3, 525–535.

Nichols, S. M. and Strizek, G. A. 1995. Electronic Voting Machines and Ballot Roll-off. *American Politics Quarterl*y. 23, 300–318.

Nielsen, J. and Levy, J. 1995. Measuring usability: Preference vs. performance. *Communications of the A*CM. 37, 4, 66–75.

Norden, L., Kimball, D., Quesenbery, W., and Chen, M. 200*8 Better ballo*ts. Brennan Center for Justice at NYU School of Law.

Ohio Secretary of State. 2007. Project EVEREST: Risk Assessment Study of Ohio Voting Systems Executive Report. http://www.sos.state.oh.us/SOS/upload/everest/00-SecretarysEVERESTExecutiveReport.pdf

Ryan, P. Y. A., Bismark, D., Heather, J., Schneider, S., Xia, Z. 2009. The Prêt à Voter verifiable election system. *IEEE Transactions on Information Forensics and Security, 4,* 662–673.

Stein, R. M., Vonnahme, G., Byrne, M., and Wallach, D. 2008. Voting technology, election administration, and voter performance. *Election Law Journ*al. 7, 123–135.

Stewart, C. 2006. Residual Vote in the 2004 Election. *Election Law Journ*al. 5, 158–169.

Traugott, M. W., Hanmer, M. J., Park, W.-h., Herrnson, P. S., Niemi, R. G., Bederson, B. B., and Conrad, F. G. 2005. The impact of voting systems on residual votes, incomplete ballots, and other measures of voting behavior. *Annual conference of the Midwest Political Science Associatio*n, Chicago, IL,. 7–10.

Wand, J. N., Shotts, K. W., Sekhon, J. S., Mebane, W. R., Herron, M. C., and Brady, H. E. 2001. The butterfly did it: The aberrant vote for Buchanan in Palm Beach County, Florida. *American Political Science Review*. 95, 4, 793–810.

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

Testing Voters' Understanding of a Security Mechanism used in Verifiable Voting

MORGAN LLEWELLYN, IMT Lucca, Italy
STEVE SCHNEIDER, University of Surrey, UK
ZHE XIA, Wuhan University of Technology, China
CHRIS CULNANE, University of Surrey, UK
JAMES HEATHER, University of Surrey, UK
PETER Y. A. RYAN, University of Luxembourg, Luxembourg
SHRIRAMKRISHNAN SRINIVASAN, University of Surrey, UK

Proposals for a secure voting technology can involve new mechanisms or procedures designed to provide greater ballot secrecy or verifiability. These mechanisms may be justified on the technical level, but researchers and voting officials must also consider how voters will understand these technical details, and how understanding may affect interaction with the voting systems. In the context of verifiable voting, there is an additional impetus for this consideration as voters are provided with an additional choice; whether or not to verify their ballot. It is possible that differences in voter understanding of the voting technology or verification mechanism may drive differences in voter behaviour; particularly at the point of verification. In the event that voter understanding partially explains voter decisions to verify their ballot, then variance in voter understanding will lead to predictable differences in the way voters interact with the voting technology.

This paper describes an experiment designed to test voters' understanding of the 'split ballot', a particular mechanism at the heart of the secure voting system Prêt à Voter, used to provide both vote secrecy and voter verifiability. We used a controlled laboratory experiment in which voter behaviour in the experiment is dependent on their understanding of the secrecy mechanism for ballots. We found that a two-thirds majority of the participants expressed a confident comprehension of the secrecy of their ballot; indicating an appropriate level of understanding. Among the remaining third of participants, most exhibited a behaviour indicating a comprehension of the security mechanism, but were less confident in their understanding. A small number did not comprehend the system. We discuss the implications of this finding for the deployment of such voting systems.

1. MOTIVATION

In response to problems with the counting and certifying of ballots in countries such as the United States of America and Iran, academics and politicians have worked together to create and implement new voting technologies. Many of these technologies provide new mechanisms or procedures that are designed to provide greater ballot security and opportunities for voters to verify their ballot. While technically sophisticated, there can arise a disconnect between technical understandings of the threats and an understanding of how voters will accept and interact with the technology. This disconnect is in some sense surprising as one motivation for technical work is the lack of voter confidence in some current voting technologies.

Typical concerns of the technical variety include distinguishing between trusted and untrusted agents [Rivest 2001], security and access to publicly posted ballot data, threats stemming from the inclusion of proprietary software, and the lifetime of the encryption. While valid, these concerns primarily consider threats in the context of the voting technology and only tangentially consider the impact of voter understanding of the selected voting technology. It is likely that most voters will avoid the technical deliberation and alternatively focus on the evaluation of non-technical questions such as, "Can I use it?", "Is my vote choice secret?" and "Will my vote count?" [Oostveen and den Besselaar 2004; Alvarez et al. 2009; Oostveen and den Besselaar 2009]. If voters believe a new security or verification mechanism endangers ballot secrecy or accuracy, these beliefs may influence voter interaction with the technology; such as not voting or choosing not to verify their vote.

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/G025797 (Trustworthy Voting Systems) and the FNR (National Research Fund) Luxembourg under project SeRTVS–C09/IS/06.

For instance, a loss of confidence in the butterfly ballot following the 2000 U.S. Presidential election, is partially responsible for its being driven into oblivion in U.S. elections [Alvarez et al. 2008]. In India, election officials have sought to increase public confidence in the election process by adopting electronic voting machines as a method to overcome "booth capturing" which is often associated with paper ballots. If voters believe, rightly or wrongly, that a voting technology leaks ballot information or that the technology provides an advantage to one political party, these voters may move to block the use of the technology or under some regimes alter their voting behaviour. In fact, research on American voters suggests individuals without voting experience may alter their turnout decision based upon their perceptions of ballot secrecy [Gerber et al. 2011]. In light of the role voter confidence can have on the voting process as a whole, we wish to evaluate voter understanding of ballot secrecy and the subsequent impact on verifiable voting technologies.

2. VERIFIABLE VOTING

One basic requirement for a verifiable voting technology is to provide a voter with the ability to check that his/her vote has been counted and verify that the election tally is correct. While straightforward in the absence of a secret ballot, verifiable voting is more difficult to achieve under a secret ballot. Despite the creation of several voting schemes which allow for verifiable voting with a secret ballot, it is unclear how voters will response to these "sophisticated" voting technologies. In particular, it is an open question how skeptical voters, those who do not understand the underlying security and secrecy mechanism(s), will react when given the choice to verify a ballot. One possibility is skeptical voters may accept election official guarantees or verify out of interest. Another possibility is those skeptical voters may simply refrain from verifying their ballot, or worse exit the electoral process. This interaction between a voter's understanding of the voting technology's security mechanisms and their decision to verify their ballot are at the heart of any serious debate seeking to implement a verifiable voting technology.

In order to evaluate how voters may behave during a verifiable election, we evaluate the rationality of individual decisions to verify their ballot through the lens of their understanding of ballot secrecy. The verifiable voting technology used in this context is a basic implementation of Prêt à Voter [Ryan et al. 2009]. In the version considered here, Prêt à Voter represents a paper-based voting system that allows voters to verify their ballot was properly recorded in the vote tally. Prêt à Voter was chosen due to the simplicity of the mechanism that ensures the "receipt" or posted information contains no information. The Prêt à Voter scheme requires that individuals who wish to verify their ballot retain a portion of their ballot.

In our design, voters possessing strong beliefs over ballot secrecy during the election and verification procedures will have an incentive to post their receipt to a public bulletin board.[1] Whereas, voters that possess significant doubts over either the ballot or verification system's ability to maintain ballot secrecy will possess an incentive to refrain from posting their receipt. We suspect that those participants with significant uncertainty over ballot secrecy may fluctuate between posting and not posting their ballot.

3. PRÊT À VOTER

We now give a high level overview of the classic Prêt à Voter system [Ryan et al. 2009; Chaum et al. 2005; Ryan and Schneider 2006]. The ballot form consists of two columns with a perforation vertically down the middle. The left hand side (LHS) lists the candidate names in a random order and the candidate ordering varies from ballot to ballot. The voter can use the right hand side (RHS) to mark her choice, and at the bottom of the RHS, there is an encrypted value. When it is decrypted, the corresponding candidate ordering on the LHS can be retrieved. An example of the ballot form is shown in Figure 1.

On the election day, each authenticated voter will be provided with such a ballot form, in secret, for example in a sealed envelope. She takes it into a voting booth, and marks her choice

[1] The posting of the ballot to a public bulletin board is a step found in many variants of verifiable voting.

Bob	
Echo	
Crystal	
David	
Alice	
	7q3Kyr

Fig. 1. A Prêt à Voter ballot form example

on the RHS against her preferred candidate. After that, she separates the ballot form into two halves along the perforation and shreds the LHS which contains the candidate names. Then the voter takes the remaining RHS to the election officials who will scan it into the election system. Finally, the RHS will be returned to the voter, and she can take it home as her receipt.

After the election day, the election system will publish all the received votes onto a public bulletin board, which may be understood as like a newspaper: once information is published, it cannot be removed and it will be available to the public. The voter can now check whether her receipt is correctly displayed on the bulletin board. If not, she can use her receipt as a proof of her vote to challenge the election.

The key innovation of the Prêt à Voter system is that each voter is provided with a receipt. The receipt contains the voter's vote, but only in encrypted form. Hence it does not reveal how the voter has voted. Meanwhile, thanks to the receipt, the voter does not need to trust the election system to correctly include her vote, since the receipt can be used to enforce this through verification.

4. METHODOLOGY

Whether voters understand the security of Prêt à Voter and how differences in understanding affect voter interaction and behaviour is an open question. We hypothesize that voters will understand the security mechanism of Prêt à Voter. We developed a simple game to test our hypothesis where the actions of the participants will vary depending upon their understanding. In particular, in the context of Prêt à Voter we are interested in evaluating whether voters understand the choice of making public their receipt will not reveal their vote choice.

To test our hypothesis we slightly modified the Prêt à Voter protocol and designed a game theoretic experiment where participant decisions to verify their ballot truthfully reveal their understanding of the security mechanism. Monetary rewards are used to motivate the subjects to behave truthfully and to take actions that are in the subject's perceived best interest. The game works as follows: within a group of 12 subjects each subject casts a vote in a fictitious election. When marking their choices, each subject will also need to select whether they wish to "post" their receipt (anonymously) so that all subjects can see it. In this case, a receipt will be similar to that shown in Figure 2. The incentives are structured such that if a subject chooses to post her receipt, she will receive a reward of £1 (the amount is denoted as $\mathcal{A}$). Otherwise, she will receive nothing. After all subjects have cast their votes, all those who selected to post receipts will have their receipts made public. Once the receipt is made public, each subject makes a guess of every participant's vote choice, whether or not that participant published her receipt. This subject will receive a reward of £.50 for each correct guess and zero otherwise. Moreover, she will lose an amount of £.50 for every participant who correctly guesses her vote choice. Note that the game design and reward values are chosen so that the experiment can separate those who believe that the receipt divulges no information from those who believe that the receipt divulges some information.

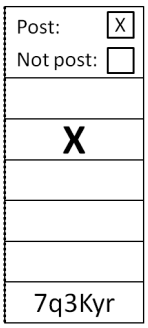


Fig. 2. A receipt example

The games economic incentives will induce subjects to self-reveal their understanding of the security mechanism. As the receipt contains no information, the dominant strategy is for subjects to post their ballot in each and every round. Given the receipt provides no information over vote choice and there is a reward for posting ones ballots, voters who possess a full or high level of understanding about the security mechanism will choose to post their ballot. However, a voter who does not fully understand the security mechanism may be afraid that the publication of her receipt will allow the other voters to guess her correct vote choice. Hence she may choose not to publish her receipt.

5. SUBJECTS & LOCATION

A total of five experiments were run at the University of Surrey, UK. The dates of the experiments were the 28^{th} and 30^{th} of June and the 1^{st} of July, 2011; with two experiments on each of the 30^{th} and 1^{st}. Three experiments were conducted during lunchtime, with an additional afternoon experiment on each of the 30^{th} and 1^{st}. Each experiment contained twelve subjects and lasted about one and a half hours.

Subjects were recruited during the week prior to the experiment on the University of Surrey campus via fliers and email. Prospective subjects were informed they would be paid between £10 and £20. While some university employees did express interest in participating, the majority of responses to the call for subjects came from the student body.

The experiment location was the seminar room within the Department of Computing at the University of Surrey. This location was chosen as it was both large enough to allow the installation of the voting equipment and was equipped with the proper technology necessary for the experiment. A designated voting area was built within the seminar room containing a voter registration table, ballot box, and three voting booths. The voting booths were equipped with walls to ensure privacy, a table for marking the ballot and a shredder for destroying the left-hand portion of the ballot.

In addition to the designated voting area, a staging area was constructed where subjects read the experiment instructions (reproduced in Appendix A), waited for their turn in the voting booth, and participated in the game following voting. The staging area consisted of 12 rectangular tables facing a whiteboard. These twelve tables were spaced such that no individual seated at one desk could read a piece of paper on any other desk.

As subjects arrived they were assigned an individual table in the staging area. The subject was handed a copy of the instructions and told that when all the subjects arrived the instructions would be read aloud to the entire group.

Before the start of each experiment, the instructions were read aloud to the entire group. Following the instructions, participants answered two questions to verify they understood the purpose of the experiment, to assess understanding of the voting technology, and to verify each

participant understood how they would be compensated. Next the randomized ballot ordering was explained and displayed to each participant. This demonstration consisted of displaying multiple ballots simultaneously on an overhead projector and discussing how the placement of a candidate's name on any one ballot varied randomly between ballots. After all subjects made a vote choice over all the ballots, individual payoffs for the round were averaged for those who chose to post their ballot and those subjects who did not. The averages of these two groups were then publicly posted for the subjects to observe. After the posting of the average payoffs, the next round of voting started.

Principal steps in the experiment:

(1) Each subject receives a ballot and takes it into the polling booth;
(2) The subject fills in her vote and chooses whether to post her receipt;
(3) The subject separates the ballot into two halves, shreds its LHS and drops its RHS into a ballot box;
(4) The receipts are made public for those who choose to post their receipts;
(5) Each subject attempts to identify the vote choice of every participant, including herself;
(6) Average payoffs for the posting group and the non-posting group are announced;
(7) Start a new round from Step 1 if the experiment is not finished.

6. EXPERIMENT RESULTS

In total there were 29 rounds of voting, with 5 rounds in the June 28^{th} experiment and 6 rounds in each subsequent experiment. Participant earnings ranged between £10 - 20. On average the experiments took approximately 1.5 hours from start to finish.

Given the game design and the careful choice of payoffs, participants who fully understood the security mechanism possessed an incentive to always post their ballot—as participants were paid £1 for posting their ballot. However, participants possessing beliefs that viewing the right-hand side of the ballot helps others guess their vote choice should be less likely or unwilling to post their ballot. This behaviour was expected as a result of the game's payoffs; participants were penalized for each participant who correctly guessed their vote choice. Thus, we expected that individuals who possessed significant doubts or a high degree of uncertainty over the secrecy of the election technology should either refrain from posting or engage in a mixing strategy; switch between posting and not posting.

Of the 348 votes cast, in 87% of the votes cast the voter chose to publicly post the right-hand side of their ballot. The breakdown of posting by round is displayed in Table *I*. In each of the six rounds, the majority of subjects chose to post their ballot. On average, there were 1 to 2 people who did not to post their ballot in any one round, out of a possible 12 people. We anticipated that by the later rounds all participants would post their ballot. While Table *I* does indicate that the proportion of participants who post their ballot is increasing over time, the size of this statistic is insignificant.[2] We conclude that even in the later rounds, round five or six, there is one individual, on average, who does not post their ballot. This implies that even after observing multiple occurrences of the voting process, posting of results, and ballot guessing game, there remains some doubt over the amount of information contained in the right-hand side of the ballot.

Analyzing the posting behaviour between the five experiments reveals similar rates of non-posting activity. In each of the five experiments, there was little variance in the total number of non-posted ballots; between 7 and 10 ballots. While posting behaviour between rounds varied slightly across experiments, there was remarkably little variance in the aggregate non-posting behaviour at the experiment level. The consistency of the posting behaviour across the five exper-

[2]A simple logistic regression of round on subject posting behaviour reveals a positive but insignificant coefficient for the variable round.

Table I. Ballot Posting by Round: All Experiments

	Did Not Post	Posted
Round 1	13%	87%
Round 2	18%	82%
Round 3	15%	85%
Round 4	15%	85%
Round 5	7%	93%
Round 6	8%	92%

Table II. Posting behaviour by Participant Type

	High Certainty	Low Certainty
High comprehension	Always Post	Usually Post
Low comprehension	Never Post	Usually not Post

iments lends credibility to the robustness of the result; i.e. more experiments are unlikely to yield different results.

While in the aggregate the security mechanism appears fairly well understood, as 87% of ballots were posted, we analyze individual decisions to evaluate two dimensions of participant perceptions over the voting system. Table *II* separates participant understanding of the security mechanism along two dimensions; comprehension and certainty. Participants may either comprehend the security mechanism or not, and these participants may possess either a high or low degree of certainty over this assessment. The comprehension dimension approximates a participants posting behaviour within any one round, while the certainty dimension helps explain participant behaviour over multiple rounds. Participants with a high degree of comprehension and high level of certainty will exhibit a behaviour consistent with always posting; this person will post in each and every round. On the other-hand, participants with a low degree of comprehension and a high level of certainty will never post their ballot. While individuals with a high degree of certainty will play the same strategy across rounds, we expect that individuals with a low level of certainty will mix between posting and not posting in an effort to reduce their exposure to the wrong strategy. Thus, individuals with a low degree of certainty over their comprehension will be observed mixing between the two posting behaviours; where high certainty types will always, or never, post.

Analyzing the individual posting behaviour, the majority of subjects posted their ballot in every single period. Out of a total 60 subjects, 62% (37 participants) posted their ballot at every opportunity. These results indicate that a majority of participants possessed a high comprehension of the security mechanism and possessed a high degree of certainty over this assessment. This data suggests that a majority of individuals took actions consistent with those of someone who understands that the right-hand side ballot contained little or no information.

Of the 23 participants who did not post a ballot in at least one round, we find evidence of only two subjects who made decisions consistent with an individual possessing a low level of comprehension and a high level of certainty over this assessment. That is out of a total of 60 participants only two subjects never posted a ballot. This behaviour is consistent with a profile that does not understand the security mechanisms provided by candidate order randomization.

The remaining 21 participants are classified as low certainty individuals, due to their posting behaviour deviating in at least one round. Of the 21 participants classified within the low certainty group, no individual posted in less than 50% of possible opportunities. These results indicate that while roughly one-third of participants possessed some uncertainty over their comprehension of the security mechanism, in no case was this assessment so low as to warrant that individual posting

in fewer than 50% of the rounds. These results indicate that even among uncertain participants the overall level of understanding was fairly high.

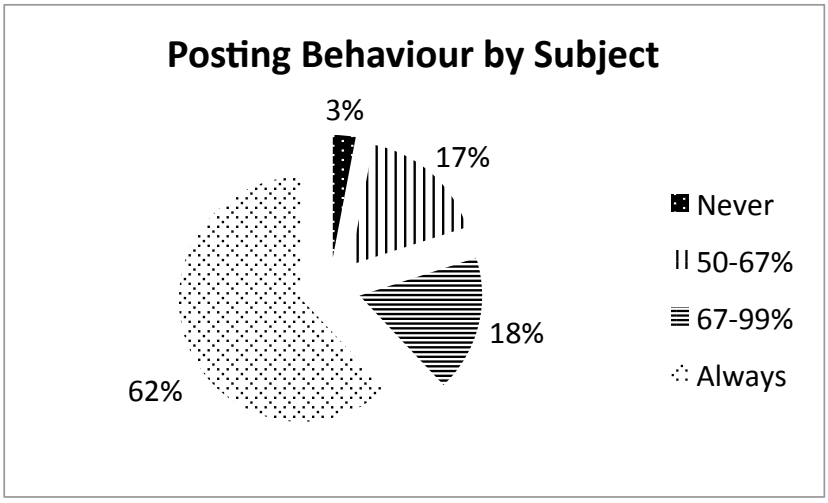


Fig. 3. Posting Behaviour by Subject

Despite the posting of average payoffs following each round, we did not observe wild swings in behaviour between rounds as a result of the previous round's outcome. Due to the low number of participants who did not post their ballot, there were several incidents where the individual who did not post their ballot received a higher average payoff than those who posted their ballot. Despite reporting this result, we did not observe a large change in the posting behaviour during subsequent round. Additionally, despite showing that on average individuals who posted their ballots tended to receive higher payoffs, even in the sixth round we observed two participants choosing not to post their ballot. We conclude that aggregate and outside information may have an affect on individuals with higher levels of uncertainty, but these information flows may not affect those who are either rightly or wrongly are certain of their understanding.

7. CONCLUSION

The majority of participants took actions consistent with those of a voter fully understanding the security mechanism of Prêt à Voter. However, we did show that differences in understanding can directly affect voter interaction with the voting technology. Approximately one-third of participants took actions consistent with a voter expressing a high level of comprehension over the security mechanism, but a corresponding low level of certainty over this assessment. Our findings present initial evidence that voter interaction with a verifiable voting technology may significantly vary by the individual voter's understanding of the technology and their confidence in this understanding. This finding raises the additional concern that, if implemented, a Prêt à Voter style voting system may be vulnerable through indirect attacks via voter beliefs over the secrecy of the voting process. While any such attack may be reduced via public information campaigns and party support, additional research is needed to determine the size and significance of these strategies.

Given the high educational level of the participants, detailed instructions, repeated nature of the experiment, and financial incentives, we hypothesize that in a general election setting the level of understanding will likely be lower than that which we observed. However, in a general election it is also uncertain to what degree understanding of the security mechanism and will simply be

replaced with a voter's notion of trust. Additional research is needed to better understand the interaction of voter understanding of the election technology and voter trust in that technology. While results indicate the transfer of information has a limited affect on individuals "certain" in their understanding of the security mechanism, it is necessary to further study this behaviour.

ACKNOWLEDGMENTS

We are grateful to Vincent Koenig, and to the anonymous referees, for comments on an earlier draft of this paper.

REFERENCES

R. Michael Alvarez, Thad E. Hall, and Morgan H. Llewellyn. 2008. Are Americans confidence their ballots are counted? *The Journal of Politics* (2008), 754–766.

R. Michael Alvarez, Gabriel Katz, Ricardo Llamosa, and Hugo E. Martinez. 2009. Assessing Voters' Attitudes towards Electronic Voting in Latin America: Evidence from Colombia's 2007 E-Voting Pilot. *E-Voting and Identity* (2009), 75–91.

David Chaum, Peter Y. A. Ryan, and Steve Schneider. 2005. A practical voter-verifiable election scheme. *Proceedings of the 10th European Symposium on Research in Computer Science (ESORICS'05)* (2005), 118–139. LNCS 3679.

Alan S. Gerber, Gregory A. Huber, David Doherty, and Conor M. Dowling. 2011. Is There a Secret Ballot? Ballot Secrecy Perceptions and Their Implications for Voting Behaviour. *British Journal of Political Science* 43, 1 (2011), 77–102.

Anne-Marie Oostveen and Peter Van den Besselaar. 2004. Internet voting technologies and civic participation: the users' perspective. *The Public* (2004), 61–78.

Anne-Marie Oostveen and Peter Van den Besselaar. 2009. Users' experiences with e-voting: a comparative case study. *International Journal of Electronic Governance* (2009), 357–377.

Ronald L. Rivest. 2001. Electronic voting. *Proceedings of Financial Cryptography (FC'01)* (2001), 243–268. LNCS 2339.

Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. 2009. Prêt à Voter: a Voter-Verifiable Voting System. *In IEEE Transactions on Information Forensics and Security (Special Issue on Electronic Voting)* 4, 4 (2009), 662–673.

Peter Y. A. Ryan and Steve Schneider. 2006. Prêt à Voter with re-encryption mixes. *Proceedings of the 11th European Symposium on Research in Computer Science (ESORICS'06)* (2006), 313–326. LNCS 4189.

A. VOTING INSTRUCTIONS

The following instructions were provided to the participants to read when they arrived at the experiment, and were read out when they were all present, before the start of the experiment. The questions at the end were used as a self-test for the participants to check that they understood the instructions.

University of Surrey Voting Experiment Instructions

General. You are about to participate in a voting process experiment in which you will cast a ballot for one of numerous alternatives. The purpose of the experiment is to gain insight into your understanding of the voting technology and features of the ballot form. The instructions are simple. You will be paid at the conclusion of the experiment. You have the right to withdraw from the study at any time and all identifiable data and information will be confidential. This study has received a favourable ethical opinion from the University of Surrey Ethics Committee. If you have any complaints or concerns about any aspect of this experiment please contact Professor Steve Schneider, s.schneider@surrey.ac.uk, 01483 689637.

Overview. You are about to participate in a voting experiment. Your compensation for this experiment will depend upon the decisions you make. While some aspects of the voting process may resemble those you have encountered in the past, there are some differences. We therefore ask you to follow these instructions and ask any questions that may arise during the course of these instructions. We kindly ask you to refrain from conversation during the experiment.

The voting procedure you will take part in tests a ballot form where the left-hand side of the ballot contains a random ordering of candidate names. The experiment consists of several rounds and the random candidate ordering is independent within and across rounds. That is in each round, the ordering of the candidates on your ballot is unrelated to the ordering of candidates on the other participants' ballots both within and across all previous rounds.

Instructions to Participants.. At the beginning of the experiment you will be allocated £10. This experiment will consistent of a set number of rounds. Each round will consist of two PHASES:

PHASE I

(1) First you will vote for a candidate. To vote for a candidate you will enter the voting booth and place a mark in the appropriate box to the right of the candidates name.
(2) Second, you must choose whether to publicly post your ballot. To publicly post your ballot mark the box that says "Post". To decline to post your ballot mark the box that says "Don't Post". In each round, if you post publicly post your ballot you will receive £1.00. If you do not publicly post your ballot in a round, you will not receive the £1.00.
(3) Next, you will separate the ballot along the perforated edge. You will then shred the left-hand side of the ballot; the portion containing the candidates' names.
(4) Deposit the right-hand side of your ballot in the ballot box provided.

PHASE II

(1) Next you will select the vote choices of the other participants using the electronic handset provided. If an individual chose to publicly post the right-hand side of their ballot, you will view the right-hand portion of their ballot prior to selecting their vote choice. For individuals who chose not to post the right-hand side of their ballot, you will be asked to select their vote choice but will not view the right-hand side of the ballot.
(2) You will win £0.50 for each vote choice you correctly identify. You will lose £0.50 for each individual who correctly identifies your vote choice.

The experimenters will keep track of payments and obtain the totals to pay participants. You will learn your total earnings only after the completion of the experiment.
After each round some information will be publicly posted. The first piece of information is the average payoff for the group that publicly posted their ballot. The second piece of information is the average payoff for the group that did not publicly post the ballot.
Are there any questions? We kindly ask you to complete the following questions as these should help you understand the instructions.

Questions

1). The purpose of this experiment is to study which item?
a). Who you will vote for in the next election.
b). Your social interaction in an election setting.
c). Your understanding of the voting technology and features of the ballot form.

2). What three activities comprise your compensation?

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

Proving Prêt à Voter Receipt Free using Computational Security Models*

Dalia Khader, Peter Y. A. Ryan, Qiang Tang
Interdisciplinary Centre for Security Reliability and Trust, SnT
and Université du Luxembourg

Abstract

Prêt à Voter is a supervised, end-to-end verifiable voting scheme. Informal analyses indicate that, subject to certain assumptions, Prêt à Voter is receipt free, i.e. a voter has no way to construct a proof to a coercer of how she voted. In this paper we propose a variant of Prêt à Voter and prove receipt freeness of this scheme using computational methods. Our proof shows that if there exists an adversary that breaks receipt freeness of the scheme then there exists an adversary that breaks the IND-CCA2 security of the Naor-Yung encryption scheme.

We propose a security model that defines receipt freeness based on the indistinguishability of receipts. We show that in order to simulate the game we require an IND-CCA2 encryption scheme to create the ballots and receipts. We show that, within our model, a non-malleable onion is sufficient to guarantee receipt freeness. Most of the existing Prêt à Voter schemes do not employ IND-CCA2 encryption in the construction of the ballots, but they avoid such attacks by various additional mechanisms such as pre-commitment of ballot material to the bulletin board, digitally signed ballots etc. Our use of the Naor-Yung transformation provides the IND-CCA2 security required.

1 Introduction

The role of voting is to ascertain the collective intent of the electorate. Voting systems are required to guarantee that the announced result is a true reflection of the intent of the electorate, given appropriate rules for the expression of preferences and computing the outcome.

In order to ensure that voters can express their intent freely, voting systems are usually required to guarantee *ballot privacy*, i.e. that no-one other than the voter herself will know how she cast her vote. Failure to guarantee this will open voters up to being deflected from expressing their true intent via either vote buying or coercion attacks. Note that it is not enough for the voting system to provide this property, it is essential that it is perceived by (vast majority of) the electorate to do so.

Security experts and cryptographers took an interest in the challenge of minimizing the level of trust in officials or in the technology used in the process. This gave rise to the notion of *voter-verifiability* or *full auditiability*, i.e. to provide every voter with the means to confirm that her vote is accurately included in the tabulation, while at

*Funded by the FNR (National Research Fund) Luxembourg under project SeRVTSC09/IS/06

the same time avoiding any violation of ballot secrecy. Typically voter-verifiability is achieved by providing each voter with an encrypted version of her vote. She can later use this to confirm that her (encrypted) vote is entered into the tabulation process. Various implementations of this idea have been proposed, [12, 25, 31], and in section 3 we will outline a particular approach: Prêt à Voter .

These two requirements, one a form of verifiability and the other of a form of secrecy, are clearly in conflict and achieving them simultaneously with minimal trust assumptions in a hostile environment has proved immensely challenging. In particular, the mechanisms that provide voter-verifiability (the encrypted ballots, the bulletin board etc.) can, if not carefully designed, introduce threats to ballot privacy.

This conflict has given rise to the introduction of more elaborate notions of ballot secrecy, namely *receipt-freeness* and *coercion resistance*. We will go into more precise definitions of these notions later in the paper, but these new properties can be thought of informally as guaranteeing ballot secrecy against increasingly powerful attackers. Ballot secrecy ensures that a passive attacker, one who simply observes the unfolding of the voting protocol, cannot determine how any individual voter cast her vote. Receipt-freeness (first defined by Benaloh and Tuinstra [11]) takes account of the fact that a voter might cooperate with an attacker to construct a proof of how she voted, using the data generated in the execution of the voting protocol. In particular the voter may be able to reveal certain data that ordinarily is assumed secret, for example passwords, credentials or randomization factors using in encryption. For coercion resistance we assume further that the attacker may interact with the voter at various stages of the voting protocol: before, during and after.

Prêt à Voter is an end–to–end verifiable scheme that gives the voters a receipt when they submit their vote to verify that their votes were never modified. Informal analysis of Prêt à Voter indicates that, subject to various assumptions, it is receipt free. however, certain subtle attacks on Prêt à Voter original version [27], made it not coercion resistance (§4.1). Variants of Prêt à Voter were proposed after to address these attacks [26]. In this paper we focus on analyzing the receipt freeness property using computational models. Strengthening our models and proofs to include different levels of coercion resistance is kept for future studies.

1.1 Contribution

Our work is the first game based analysis of the receipt freeness of (a variant of) Prêt à Voter based on indistinguishablity of receipts. Formal methods (symbolic) techniques have been used earlier on to analyze the same notion in Prêt à Voter [24, 33], but these methods do not capture non-malleability of the ballot.

It should be noted that the standard versions of Prêt à Voter assume that blank receipts are posted to the Bulletin Board and signed ahead of the election. Hence, casting vote corresponds in effect to filling in the vote information in an existing blank ballot. The approach counters related plaintext style attacks on the tabulation that involve the attacker posting receipts that are constructed in some way from real receipts. Our model does not assume such a mechanism, hence the need to ensure non-malleability of receipts. In this model we use the Noar-Yung construction as this meshes nicely with some existing constructions to enable print-on-demand of Prêt à Voter ballots. These constructions involved encrypting the candidate order under two, distinct public keys,

one for the tabulation tellers and one for the printer, along with Zero-Knowledge proofs of the correspondence of the orders.

We will see that the model presented in this paper captures many of the notions mentioned earlier, essentially those that can be classed as receipt-freeness, but fails to capture certain attacks that fall into the category of coercion-resistance, e.g randomisation attacks. Capturing notions such as receipt-freeness and coercion resistance has proved very challenging in any formalism, but poses particular challenges in the context of computational models. We note also that modelling Prêt à Voter poses new challenges due to the special way that encrypted ballots are created: in contrast to most voter-verifiable schemes, Prêt à Voter does not directly encrypt the voter's choice but rather pre-prepares an encryption of a randomized frame of reference in which the vote is encoded.

Finally, we should mention that we provide an abstract description of Prêt à Voter in §3. A voting scheme that fits that description can be proven secure in our security model in §5. If we make changes on the system level a new model needs to be provided to analyze the system, however if we make changes on the construction level only, a new proof of security needs to be provided but the same security model can be used[1].

1.2 Related Work

Although e-voting schemes use cryptography to ensure various levels of privacy, little work has been done on proving them secure using computational models. The complexity of e–voting schemes and the different security requirements needed, makes it extremely challenging to find one computational model suitable for all voting schemes, therefore researchers proposed several models each suitable for specific voting scheme, we list some below.

Okamoto provided the first provable secure receipt free voting scheme [20]. Moran and Naor [17, 18] aimed for the everlasting–privacy property in their voting schemes. They use Universal Composability framework in their proofs and their constructions are based on commitment schemes. Küsters et al. analyzes four e-voting schemes using simulation based models: Bingo, TheeBallot, VAV, and Scantegrity II [14–16]. The main idea in all papers is that a coercer can distinguish whether a vote has followed his orders or have voted as to his intended goal and that is done by running a counter-strategy. Bernhard et al. [2] proposed a voting-friendly encryption scheme that can be used in the Helios voting system and proved that such a scheme is needed to prove ballot secrecy using game based models. Helios is a remote voting scheme and unlike Prêt à Voter it does not require going to a voting booth.

Work has been done on proving receipt freeness in voting schemes using symbolic analysis [1, 9, 13], §2.1 explains the difference between the two models briefly.

2 Preliminary Concepts

2.1 Provable Security

In formal method analysis the cryptographic primitives are represented by function symbols considered as black–boxes, and the analysis assumes perfect cryptography

[1] Preliminary version of this paper is published on ePrint Archive: Report 2011/594.

(e.g. the algorithms "Encrypt" and "Decrypt" are treated as symbolic operations and that means agents can decrypt only if they know the right keys, but cannot otherwise "crack" encrypted messages). In computational models the cryptographic primitives are functions from bit–strings to bit–strings, and the adversary is any probabilistic Turing machine. Even though the computational models seem more realistic, but they complicate the proofs, and it is hard to study a large system using them.

In this paper we study receipt freeness of Prêt à Voter using a computational model referred to as *provable security* paradigm. Evaluating the security of a protocol in the *provable security* paradigm proceeds in two steps.

1. The first step is to design a security model for a category of cryptographic protocols. The security model usually takes the form of an adversary vs challenger game, where the adversary represents the potential attacker which may break the protocol in practice and the challenger represents the honest parties in the protocol execution. In the game, the adversary interacts with the challenger through some oracles, which reflects the privileges that he can have in practice. Moreover, an advantage will be defined to indicate the condition that the adversary has broken the security of the protocol and won the game. Section 4 provides a security model for Prêt à Voter receipt freeness.
2. Given a protocol and the corresponding security model, the proof is essentially to show that if any adversary can win the game with a non-negligible advantage then it can actually be used as a subroutine to solve a hard problem, e.g. discrete logarithm problem. If we can make the assumption that the hard problem is intractable then we can draw the conclusion that the protocol is secure in the security model based on the assumption (See §5).

As an example, security models have been defined for public-key encryption schemes in the next subsection and the same example will be used in our security proof of receipt freeness (See §5). With respect to provable security, one important notion is negligibility, formally defined as follows.

Definition 2.1 *Function $P(k) : \mathbb{Z} \to \mathbb{R}$ is said to be negligible with respect to k if, for every polynomial $f(k)$, there exists an integer N_f such that $P(k) < \frac{1}{f(k)}$ for all $k \geq N_f$.*

2.2 Security Notions for Public-key Encryption

A public-key encryption scheme consists of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, defined as follows. $\mathsf{KeyGen}(\lambda)$ takes a security parameter λ as input and outputs a public/private key pair (pk, sk). $\mathsf{Enc}(M, pk)$ takes a message M and the public key pk as input, and outputs a ciphertext C. $\mathsf{Dec}(C, sk)$ takes a ciphertext C and the private key sk as input, and outputs a plaintext M or an error symbol $\perp$.

For public-key encryption schemes, the strongest security notion is IND-CCA2 (indistinguishability under adaptive chosen ciphertext attack). Informally, this property guarantees that an (adaptive and active) attacker will not be able to learn any information about the plaintext in a ciphertext c, even if it can decrypt any ciphertext except for c. This property is essential when applying public-key encryption schemes to build interactive security protocols, where an attacker may try to learn private information through interactions with honest parties.

Definition 2.2 *A public-key encryption scheme achieves IND-CCA2 security if any*

polynomial time attacker only has negligible advantage in the attack game, shown in Figure 1. Note that the advantage is defined to be $|\Pr[b' = b] - \frac{1}{2}|$.

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker is given pk and can issue a polynomial number of decryption queries with any input: Given C, the challenger returns $\mathsf{Dec}(C, sk)$. At some point, the attacker chooses M_0, M_1 of equal length and sends them to the challenger for a challenge.
3. *Challenge*. The challenger selects $b \in_R \{0, 1\}$ and returns $C_b = \mathsf{Enc}(M_b, pk)$ as the challenge.
4. *Phase 2*. The attacker can issue a polynomial number of decryption oracle queries with any input except for C_b.
5. *Guess*: At some point the attacker terminates Phase 2 by outputting a guess b' for b.

Figure 1: IND-CCA2 Game

In Definition2.2, if we remove *Phase 2* in the attack game then it becomes the definition for IND-CCA1 (indistinguishability under chosen ciphertext attack). Furthermore, if we completely disallow the attacker to access the decryption oracle then it becomes the standard IND-CPA (indistinguishability under chosen plaintext attack).

The above definition for IND-CCA2 is the standard one, but it has other equivalent forms. Below, we propose a different security model (i.e. IND-CCA2† security) for public key encryption schemes and show that this new security model is equivalent to the standard IND-CCA2 (Appendix A). The information format in this definition matches with the data format in our voting scheme, hence, this new model facilitates our security analysis in Section 5.

Definition 2.3 *A public-key encryption scheme achieves IND-CCA2† security if any polynomial time attacker only has negligible advantage in the attack game, shown in Figure 2. Note that the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker is given pk and can issue a polynomial number of decryption queries with any input: Given C, the challenger returns $\mathsf{Dec}(C, sk)$. At some point, the attacker chooses M_0, M_1 of equal length and sends them to the challenger for a challenge.
3. *Challenge*. The challenger selects $d \in_R \{0, 1\}$. If $d = 0$, the challenger returns $E_0 = (\mathsf{Enc}(M_0, pk), \mathsf{Enc}(M_1, pk))$ as the challenge, otherwise returns $E_1 = (\mathsf{Enc}(M_1, pk), \mathsf{Enc}(M_0, pk))$ as the challenge.
4. *Phase 2*. The attacker can issue a polynomial number of decryption queries with any input except for the two ciphertexts in E_d.
5. *Guess*: At some point the attacker terminates Phase 2 by outputting a guess d' for d.

Figure 2: IND-CCA2† Game

It is known that if a public-key encryption scheme is IND-CCA2 secure, then it must be IND-CPA secure. In appendix A, we prove the following theorem on the equivalence between IND-CCA2 and IND-CCA2†. This theorem allows us to employ an IND-CCA2 secure scheme and prove the security in an IND-CCA2† security model.

Theorem 2.4 *A public-key encryption scheme* (KeyGen, Enc, Dec) *is IND-CCA2 secure if and only if it is IND-CCA2† secure.*

2.3 Zero Knowledge Proofs

Loosely speaking, a zero knowledge proof is a cryptographic primitive that remarkably provides a convincing proof that an assertion holds without yielding any further information. In this paper we use non–interactive Zero Knowledge proofs (NIZK). Given language L_R defined by NP–relation R. The pair $(x, w) \in R$ if w is a witness that $x \in L_R$. A proof of system L_R is given by a pair of algorithms $(Prove, Verify)$. The prover and verifier both have an element $x \in L_R$ as input. In addition the prover has witness w such that $R(x, w) = 1$. The prover computes and sends a proof π to the verifier. The verifier outputs a decision to accept or reject the proof π. The requirements of the proofs is to be sound (if $x \notin L_R$ the verifier should reject π with high probability), correct (if $x \in L_R$ then verifier should accept proof) and zero knowledge (Nothing other than the validity of the assertion is revealed[2]).

2.4 Naor–Yung Transformation

The Naor-Yung IND-CCA2 construction [19] needs two ingredients: an IND-CPA encryption scheme (KeyGen, Enc, Dec) and a sound zero-knowledge proof system (Prove, Verify) that can prove that two ciphertexts encrypt the same message (i.e. A plaintext equivalence proof of different public keys, See Appendix B.3). In more detail, the Naor-Yung Transformation produces a scheme $NY.E = (NY.\mathsf{KeyGen}, NY.\mathsf{Enc}, NY.\mathsf{Dec})$ as follows.

- $NY.\mathsf{KeyGen}(\lambda)$: Take security parameter λ as input and run KeyGen twice to produce two key pairs $(sk_1, pk_1) = \mathsf{KeyGen}(\lambda)$ and $(sk_2, pk_2) = \mathsf{KeyGen}(\lambda)$. The secret key $NY.sk = (sk_1, sk_2)$ and public key $NY.pk = (pk_1, pk_2)$.
- $NY.\mathsf{Enc}(m, NY.pk)$: Compute $c_1 = \mathsf{Enc}(m, pk_1)$, $c_2 = \mathsf{Enc}(m, pk_2)$, and $\pi = \mathsf{Prove}(m, pk_1, pk_2, r_1, r_2, c_1, c_2)$, where r_1, r_2 represents the randomness used in constructing c_1 and c_2 respectively. The final ciphertext is (c_1, c_2, π).
- $NY.\mathsf{Dec}(c_1, c_2, \pi)$: If $\mathsf{Verify}(c_1, c_2, \pi) = 1$ then output $m = \mathsf{Dec}(c_1, sk_1)$; otherwise output an error symbol $\perp$.

Theorem 2.5 ***(Sahai [28]):*** *If the zero knowledge proof system* (Prove, Verify) *is a proof of knowledge and has uniquely applicable proofs (essentially each proof can only be used to prove one statement) and if the encryption scheme* (KeyGen, Enc, Dec) *is IND-CPA then applying Naor-Yung transformation gives an IND-CCA2 secure encryption scheme.*

In Appendix B we provide examples of the zero knowledge proofs that can be used with exponential ElGamal to obtain a Naor–Yung encryption.

[2] A proof is zero knowledge if there exist a simulator capable to produce transcripts indistinguishable from the real transcripts.

3 Prêt à Voter

The Prêt à Voter approach to verifiable voting, randomizing candidate order on ballot to encode votes, was first proposed by Ryan in [27] and elaborated in, for example, [23]. "Classical" Prêt à Voter is a polling station electronic voting scheme shown to be receipt-free. In this section we describe the key idea behind the design.

To facilitate our discussions, instead of denoting an encryption algorithm as $\mathsf{Enc}(m, pk)$, we use a slightly different notation $\mathsf{Enc}(pk, m, r)$, where pk is the public key, m is the message, and r is the randomness used in the encryption.

3.1 Prêt à Voter Overview

Here we explain the the key elements of Prêt à Voter , abstracting from details that are not relevant to the present anaysis.

Ballot Structure: We first explain the structure of the "Prêt à Voter " ballots. The ballot has a left hand side (LHS) with a randomly permuted list of candidates, and a right hand side (RHS) which carries an encryption of the order of the candidates in the LHS, usually referred to as *the onion* for historical reasons. Each ballot has a unique serial number (which could be a hash of the onion), (SN), for administrative purposes such as searching for the ballot on the bulletin board, etc. (See Fig. 3).

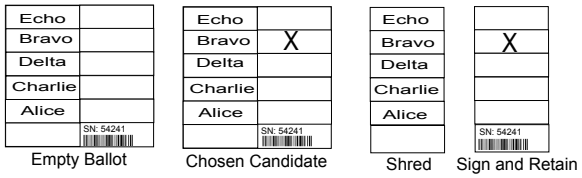


Figure 3: Prêt à Voter : The Ceremony

Overall System: Once registered in the polling station, the voter receives, via a confidential channel (e.g. Print–on–demand, sealed envelopes, etc), the ballot. The voter is offered the choice to cast the vote with this ballot or to audit it. If the chooses the former, she marks here preferences in the privacy of a voting booth. If the latter, officials or helpers are available to assist her in the ballot audit, after which she can obtain a fresh ballot.

In the booth the voter places a mark next to the name of the candidate she wants to vote for. She separates the RHS from LHS, shreds the LHS and takes the RHS to an official who scans and sends it to the tallying authority. A signed, franked copy of the RHS is given to the voter to keep. All receipts and audited ballots are published on a bulletin board. Voters can verify that their votes has been accurately entered in the tally by checking the onion, serial number and choice of index, against the published results on the bulletin board.

Tallying authorities compute the result of the elections and the result is published on the board. Figure 4 shows the ballots' cycle from the point its created to the point where a receipt or audited ballot is published on the bulletin board.

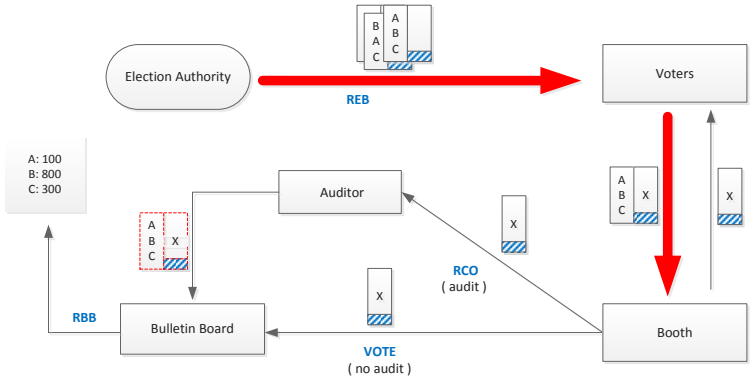


Figure 4: Prêt à Voter : The System

Random auditing the ballots is used in all versions of Prêt à Voter to ensure the well-formedness of ballot forms but differs slightly in style. The auditing procedure involves decrypting onions on selected ballot forms and checking that they correspond to the LHS order. Given that the authorities responsible of creating the ballots can not predict which ballots will be chosen for auditing, it is hard to cheat without a high possibility of getting caught. An audited ballot can not be used for voting.

Tallying the result: The onions are used in the tabulation to interpret the voters mark on the scanned RHS, enabling the tallying authorities to count the votes. The details of the procedure varies in the different versions of Prêt à Voter [23,24,32]. Loosely speaking, the vote processing part of Prêt à Voter takes a set of encrypted votes (receipts) and outputs a set of unencrypted votes, but without allowing anyone (including those involved in the decryption) to perform end–to–end matching. This is done by mixing, decrypting, and tallying; where some variants of Prêt à Voter combine the mixing and decrypting procedure (using mix–nets), and other variants combine the decrypting and tallying phases (homomorphic tallying). Regardless of what vote processing method is used in Prêt à Voter the bulletin board will always have the serial number, onion and the index published.

To be able to provide a proof for the general case of Prêt à Voter we assume:

- Every ballot is identified by a unique serial number and two ballots should not have the same RHS.
- The voting takes place in a private booth in the supervised polling station.
- The ballots are distributed in a confidential way (e.g.post or print–on–demand).
- Counter-measures are in place to prevent any subliminal channels leaking information about the LHS via data on the RHS (e.g. distributed creation of encryptions to suppress kleptographic channels.)
- The attacker does not get access to ballots forms that are used to cast votes.
- The LHS of the ballot is shredded and destroyed in the booth.

- Audited ballots are not used for voting.
- The tabulation process reveals only the final tally information. Strictly speaking it will typically also provide proofs of correctness of these results, but these we assume to be zero–knowledge and hence revealing nothing further that could undermine ballot secrecy. We might use for example, a tabulation scheme proven secure in the UC paradigm, such as Wilkstroms, [30].

Different methodologies exist in literature to enforce these properties and counter measures can be found in the Prêt à Voter literature [23, 24, 32].

3.2 Construction of Ballots in the Versatile Prêt à Voter

We revisit the work [3–5, 32], the latest versions of Prêt à Voter , and adopt certain ideas used in their work to create the ballots. The ballots in these papers encoded the candidates as $\{M_0, \ldots, M_k\}$. The ballot generator creates a list of ciphertexts $\{\mathsf{Enc}(pk, M_0, 1), \ldots, \mathsf{Enc}(pk, M_k, 1)\}$. Anyone can verify that the ciphertexts are unique encryptions of the candidates since the randomization value is 1. Then the ciphertexts are re–encrypted and shuffled in a verifiable manner. The output is a permutation of $\{\mathsf{Enc}(pk, M_0, r_0), \mathsf{Enc}(pk, M_1, r_1), \mathsf{Enc}(pk, M_2, r_2), \ldots, \mathsf{Enc}(pk, M_k, r_k)\}$ that corresponds to LHS of the ballot. The proof of shuffle and the outputted list of ciphertext presents the onions. The papers used either exponential ElGamal [10] or Paillier [21].

3.3 The New Ballot Construction

In this section we show how one can have an IND-CCA2 ballot with minimum changes to the construct above. We assume the ballot generator has two sets of keys (pk_1, sk_1) and (pk_2, sk_2). The ballot generator creates a "shuffled" list of $\{(c_1, \grave{c}_1, \pi_1), \ldots, (c_k, \grave{c}_k, \pi_k)\}$, where $c_i = \mathsf{Enc}(pk_1, M_i, r_i)$, $\grave{c}_i = \mathsf{Enc}(pk_2, M_i, \grave{r}_i)$, π_i is a zero knowledge proof as described in §2.4 and M_i is one of the candidates codes. Furthermore, the ballot generator provides a verifiable proof of shuffle Υ on the input list of ciphertexts $\{\mathsf{Enc}(pk_1, M_0, 1), \ldots, \mathsf{Enc}(pk_1, M_k, 1)\}$ and the output of the shuffle are ciphertexts $\{c_1, \ldots, c_k\}$. Note that everyone can verify the inputted ciphertexts since the randomizations are 1, and no need to provide a proof of shuffle of $\{\grave{c}_1, \ldots, \grave{c}_k\}$ since $\pi_1, \ldots, \pi_k$ are provided. The onion of the ballot this time contains of the following: $\{(c_1, \grave{c}_1, \pi_1), \ldots, (c_k, \grave{c}_k, \pi_k)\}$ and Υ.

A ballot is well formed if an onion includes all ciphers of the different candidates (done by checking the verifiable shuffle proofs) and if the permutation in the onion corresponds to the LHS of the ballot (done by auditing). This relies on the soundness property of the employed zero knowledge proofs for Naor–Yung encryption, shown in Appendix B.3.

The voting procedure and processing remain the same as explained in §3.1. The fact that each candidate is encrypted with two different public keys is not entirely new for Prêt à Voter like schemes. Similar suggestions were proposed in literature [26] to provide Print–On–Demand for ballots and relax the trust assumptions based on the chain of custody[3].

[3] In Prêt à Voter the LHS is a plaintext which means that the chain of custody between the ballot generation and until the ballot reaches the voter should be trusted.

4 The Receipt Freeness Security model

The main advantage of using a cryptographic security model (in §4.3) over existing symbolic analysis of Prêt à Voter is to base the security of the scheme on exact computational hardness assumptions. A cryptographic proof should answer the question "How secure should the encryption scheme and proofs (used in creating the ballots) be, in order for Prêt à Voter to be receipt free?". Before introducing the security model of receipt freeness, we first give an overview in §4.1 on existing definitions and attacks related to confidentiality of the votes and coercion resistance. Our security model will capture some of them, while leaves the study of others as a future work.

4.1 Discussion on Confidentiality of Votes

The easiest way to preserve confidentiality of the vote is to have the voters encrypt the votes and then break the link between the voter and the votes by using mix–nets. Observing however that a coercer could demand that the voter reveal her private key or the randomness used for the encryption motivates stronger definitions that hold against such a coercer. In the context of Prêt à Voter , the encryptions are formed in advance and so the voter does not have access to such information as the randomization.

In literature certain subtle attacks were discovered and studied that relate to the confidentiality of a vote in an election scheme. A lot were addressed for Prêt à Voter by changing certain implementation details [26]. The following is a list:

- Auditing voted ballots: if an attacker is able to audit a ballot that is used to cast a vote he can violate its secrecy. Our model captures such attacks.
- Ballot dependence: if an attacker can construct a ballot from one or more cast ballots and inject it into the tabulation he may be able to obtain information about the cast ballots. Our model captures such attacks.
- Chain-voting. This is an attack to which a coercer obtains a blank ballot, marks it with his candidate and hands it to a voter with instructions to cast it and then pass him a fresh ballot. The coercer can continue the process indefinitely, marking each fresh ballot and obtaining another from the next voter. Our model does not capture such attacks.
- Randomization: if an attacker can instruct a voter before voting on some aspect of the receipt he may be able to prevent the voter from voting freely, even though he might not be able to determine the vote. In Prêt à Voter for example the coercer can instruct the voter to place her X in the first position. In the absence of counter-measures, this effectively randomizes the vote. Our model does not capture such attacks.
- Ballot signature ("Italian Attack"): If the options available to the voter as large, as is the case for example with Single Transferable Vote (STV) systems with a large number of candidates, it is possible for the coercer to require the voter to provide a unique "signature" on her ballot, e.g. a particular ranking of low ranked candidates. Our model does not capture such attacks.

In general, our model does not capture attacks, in which the attacker not only gets access to receipts but also interacts with the voters. This can be justified in terms of assumptions we make about the implementation of the scheme (See §3.1). These assumptions are reflected in the model in terms of restrictions that the adversary has

when interacting with the challenger. We argue that, for the purposes of defining receipt freeness for Prêt à Voter , our model embodies the most powerful attacker consistent with such reasonable assumptions. On the other hand it is not powerful enough to capture coercion resistance including randomization attacks, Italian attacks and chain voting. Elaborating the model for this purpose will be the subject of future work.

4.2 Oracles for the Adversary

The security model is defined using an attack game between a hypothetical adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{A}$ tries to win the game by issuing certain oracles to $\mathcal{C}$. Referring to Figure 4 which shows these oracles and what they represent in the real system, our security model has the following oracles.

- **Retrieve Empty Ballot (REB):** $\mathcal{A}$ sends a request to $\mathcal{C}$ for an empty ballot that he would like to get. $\mathcal{C}$ generates and returns an empty ballot (i.e. a ballot with both RHS and LHS).
- **Reveal Candidate Order (RCO):** $\mathcal{A}$ sends the RHS of the ballot used or not, and $\mathcal{C}$ responds with the candidate order on that ballot. This oracle reflects the audit capability that an adversary may have.
- **Vote:** $\mathcal{A}$ sends a vote (i.e a RHS of the ballot marked on the position of $\mathcal{A}$'s choice) to $\mathcal{C}$. Upon receiving a vote, $\mathcal{C}$ first validates it according to the procedure specified in the underlying voting scheme, and then puts it on the bulletin board if it is valid. Otherwise, $\mathcal{C}$ rejects the vote. It is worth noting that $\mathcal{A}$ can either use an empty ballot from $\mathcal{C}$ to construct a vote or forge a vote at its own interest.
- **Reveal Status of Bulletin Board (RBB):** $\mathcal{A}$ sends a request to reveal status of the bulletin board. $\mathcal{C}$ returns the tallying result of the board until the moment of the query. Votes can be added subsequently and $\mathcal{A}$ can query RBB again.

4.3 Receipt Freeness Attack Game

In the game, we assume that there are two candidates (i.e. Alice and Bob) for simplicity of explanation. Nevertheless, it can be easily extended to include more. Given the above oracles and the public information in the system, the goal of a receipt freeness adversary is to distinguish between two honestly generated receipts by telling which one is for Alice. Formally, the game is shown below.

In the attack game, trusted and compromised parties are each considered to be a single entity, i.e. the challenger $\mathcal{C}$ and adversary $\mathcal{A}$ respectively. As we focus here on proving receipt-freeness only, we assume that $\mathcal{A}$ is not interested in any other attacks. As long as the ballots used to query the oracles are not compromised, $\mathcal{A}$ should not be able to distinguish between the value of the votes of receipts i and j. Note that, if all voters have voted the same way, the confidentiality of the votes is by nature broken. Therefore, we require that the two receipts represent votes for different candidates.

We refer to the model above as EXP_{RF}, the probability of guessing the receipt in case of two candidates should be $\frac{1}{2}$. Receipt freeness for a two-candidate election is defined in Definition 4.1.

Definition 4.1 *A "Prêt à Voter " voting scheme is receipt free if for all polynomial time adversaries $\mathcal{A}$, $Adv.RF(k) = |Pr(EXP_{RF} = 1) - 1/2| \leq \epsilon$ where k is the security parameter and ϵ is negligible.*

- RF.Setup: $\mathcal{C}$ sets up the system by creating the private and public parameters, then sets up an empty bulletin board. The bulletin board is accessible to $\mathcal{A}$ as "read only". $\mathcal{C}$ sends the public parameters to $\mathcal{A}$.
- RF.Phase[I]: $\mathcal{A}$ queries the REB, RCO, RBB, and Vote oracles.
- RF.Challenge: $\mathcal{A}$ asks for a challenge challenge and $\mathcal{C}$ returns the two receipts, one as a vote to Alice and the other as vote to Bob, randomly assigned. The ballots corresponding to the receipts should be generated by $\mathcal{C}$ and have not been queried in the REB, Vote, and RCO oracles. Let's assume the serial numbers of the two receipts are i and j
- RF.Phase[II]: $\mathcal{A}$ queries the REB, RCO, RBB, and Vote oracles. The challenged receipts i and j can only be queried in RBB simultaneously.
- RF.Guess: $\mathcal{A}$ returns a serial number $b \in \{i, j\}$ as its guess to which ballot was used in voting for Alice. If the guess is correct then $\mathcal{A}$ wins the game and the output is 1, if the adversary does not guess it right then the output is 0.

Figure 5: Receipt Freness Attack Game

For Prêt à Voter , the receipt freeness is about the ability to construct a proof to another party via using the receipts and the bulletin board information. The physical constraints (private booth and shredding LHS) in the real system are presented in the model by allowing the challenger to submit the challenged votes without the influence of the adversary while it gives $\mathcal{A}$ the receipts and read only access to BB. Relating the model to §4.1 auditing voted ballots is captured by restricting the RCO oracle where the receipts i and j are not allowed to be queried in RCO. Ballot dependence is captured because the votes submitted to RCO or Vote oracles can be constructed based on any existing ones (including the challenged ones).

5 The Proof of Receipt Freeness

In this section, we prove the following theorem for the new construction in section 3.3.

Theorem 5.1 *Given the proofs used in constructing ballots are zero knowledge, if there exists an Adversary $\mathcal{A}$ that can break the receipt freeness of Prêt à Voter then there exists a simulator $\mathcal{S}$ that breaks the IND-CCA2† security of Naor–Yung encryption.*

The intuition behind our proof is fairly straightforward. Assuming the existence of an adversary $\mathcal{A}$ who can win the receipt freeness game described in § 4 with a non-negligible advantage, then we can construct a simulator $\mathcal{S}$ which can win the IND–CCA2† security game of Naor–Yung scheme. Basically, the simulator $\mathcal{S}$ takes input from Naor–Yung challenger $\mathcal{C}$, who sets up the Naor–Yung Encryption scheme, and uses that input in playing the role of the receipt freeness challenger with $\mathcal{A}$ in the receipt freeness game.

Proof. Let's assume that $\mathcal{A}$'s advantage in attacking the receipt freeness game is ϵ^*, where the game is defined in Figure 5. The proof can be conducted with the standard game-hopping technique.

Consider a new game, denoted as Game1. In Game1, $\mathcal{S}$ plays the role of receipt freeness challenger and performs identically to what is required in Figure 5, except for the following: in the RF.Challenge phase, the shuffle proofs of the two receipts are removed. Note that the shuffle proof on a receipt proves that the two Naor–Yung ciphertexts are encryptions for Alice and Bob respectively. Let's assume that $\mathcal{A}$'s advantage in this game is ϵ. Given the fact that the shuffle proofs are zero knowledge, $\mathcal{A}$'s's advantage should be the same with/without the presence of the proofs, namely $|\epsilon - \epsilon^*|$ is negligible.

Consider a new game, denoted as Game2. In this game, $\mathcal{S}$ tries to win IND–CCA2† security game of Naor–Yung scheme (shown in Figure 2), by leveraging on $\mathcal{A}$'s responses. In more detail, $\mathcal{S}$ interacts with the Naor–Yung challenger $\mathcal{C}$ and $\mathcal{A}$ as follows.

- **Simulating the RF.Setup**
  $\mathcal{C}$ sets up $NY.E$ by running $NY.KeyGen(k)$. He gives $\mathcal{S}$ the public parameters $NY.pk = (pk_1, pk_2)$. The $\mathcal{S}$ initializes the bulletin board BB which is accessible to $\mathcal{A}$ as read only and is initially empty. He also creates a private database of ballots DB that will contain $(SN, onion, candidates.order, vote)$. Initially that database is empty (it gets filled in RF.Phase I, II). He passes to $\mathcal{A}$ the values in $NY.pk$. See Figure 6.

$\mathcal{A}$		$\mathcal{S}$		$\mathcal{C}$
	$\xleftarrow{NY.pk}$	Initialize BB and DB	$\xleftarrow{NY.pk}$	Setup $NY.E$

Figure 6: Setup Phase

- **Simulating RF.Phase[I] and RF.Phase[II]**
  In these two phases, $\mathcal{A}$ queries the REB, RCO, RBB, and Vote oracles, and $\mathcal{S}$ answers as in Figure 7.

$\mathcal{A}$		$\mathcal{S}$		$\mathcal{C}$
Oracle REB:				
	$\xrightarrow{REB}$	Create $Ballot$;		
		Update DB with vote=null;		
	$\xleftarrow{response}$	response = empty ballot;		
Oracle Vote:				
	$\xrightarrow{Vote:Onion,index}$	If $Ballot \in DB$, vote$\neq null$, $Ballot \in BB$;		
	$\xleftarrow{response}$	response = Ballot used.		
		If $Ballot \in DB$, vote$= null$, $Ballot \notin BB$;		
		Update BB, Update DB vote$= c.name$		
		If $Ballot \notin DB$	$\xrightarrow{Dec:Ballot}$	candidates.order =
		Update DB, Update BB	$\xleftarrow{candidate.order}$	NY.Dec(ballot,..)
	$\xleftarrow{response}$	response = receipt,vote accepted		
Oracle RCO:				
	$\xrightarrow{RCO:RHS}$	If $Ballot \in DB$;		
	$\xleftarrow{response}$	response =candidate.order		
		If $Ballot \notin DB$	$\xrightarrow{Dec:Ballot}$	candidates.order =
		Update DB	$\xleftarrow{candidate.order}$	NY.Dec(ballot,..)
	$\xleftarrow{response}$	response =candidate.order		
Oracle RBB:				
	$\xrightarrow{RBB}$	Result = Tally BB		
	$\xleftarrow{Results}$			

Figure 7: Phase I and Phase II

The following explains the details of the simulation:

- REB: $\mathcal{S}$ receives a request for an empty ballot. He creates a new ballot by creating the proper encryptions given he has the public parameters required (i.e. $NY.pk$). He adds the onion in the database of (SN, $onion$, $candidates.order$, $vote$), with $vote$ being $null$. Note that SN is the unique serial number for the empty ballot.
- Vote: After receiving the query (.ie. an onion and index), $\mathcal{S}$ checks the ballot on the bulletin board if it has been used before. If so, he returns the used ballot to the adversary. Otherwise, he checks the $vote$ element in the database, if the ballot exists but the vote does not, he updates bulletin board and updates database to have $vote = c.name$. If the ballot does not exist in database then it must be created by the adversary, $\mathcal{S}$ queries the decryption oracle from $\mathcal{C}$ to get the information of the ballot and updates the database and the bulletin board as required. At the end of this oracle, the adversary should get one of two responses: either the "ballot has been used" or "the vote accepted, and a receipt". If it is the latter response then the adversary should be able to see an updated bulletin board.
- RCO: $\mathcal{S}$ checks if the ballot is in the database to get the candidate's order otherwise query the decryption oracle, update the database then return the candidate order.
- RBB: $\mathcal{S}$ tallies the bulletin board and returns the result.

- **Simulating RF.Challenge and RF.Guess**
  In the Challenge and Guess phases, the interactions among different entities are summarised in Figure 8. Without loss of generality, we assume that the first element of each ballot is ticked, namely the candidate corresponding to the first ciphertext is voted and the receipt is a pair of Naor–Yung ciphertexts.

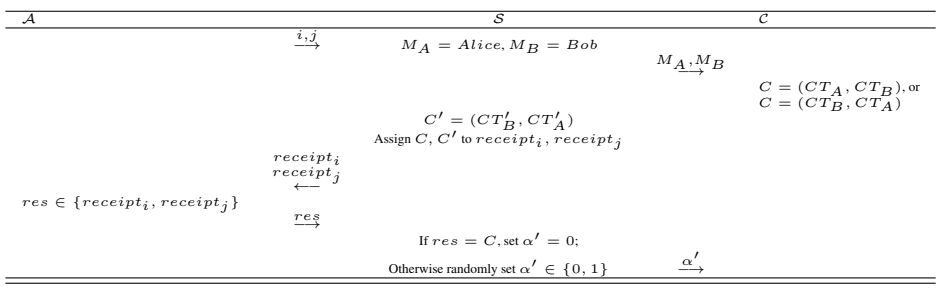


Figure 8: Challenge and Guess Phases

As shown in the figure, the essential strategy of $\mathcal{S}$ is to simulate the challenge (i.e. two receipts $\{receipt_i, receipt_j\}$) for $\mathcal{A}$ with two pieces of information: one is the Naor–Yung challenge C from $\mathcal{C}$ and the other is a Naor–Yung ciphertext pair C' generated by itself. With the response from $\mathcal{A}$, $\mathcal{S}$ can form a guess for $\mathcal{C}$'s random coin α. The details of RF.Challenge and RF.Guess are elaborated below.

1. $\mathcal{A}$ picks two serial numbers i and j, where the two ballots have not been queried before in REB, Vote and RCO and sends them to $\mathcal{S}$ [4].

[4]Recall from §3 the game restricts the two ballots i and j from being queried in oracles above because

2. $\mathcal{S}$ assigns $M_A = Alice$ and $M_B = Bob$ and sends them to $\mathcal{C}$.
3. $\mathcal{C}$ first computes Naor–Yung ciphertexts CT_A, CT_B for M_A, M_B respectively. $\mathcal{C}$ then tosses a coin $\alpha \in \{0,1\}$, and returns $C = (CT_A, CT_B)$ if $\alpha = 0$ and $C = (CT_B, CT_A)$ otherwise.
4. After receiving C, $\mathcal{S}$ generates $C' = (CT'_B, CT'_A)$, where CT'_A, CT'_B are two new Naor–Yung ciphertexts for M_A, M_B respectively. Based on C and C', $\mathcal{S}$ generates two receipts $\{receipt_i, receipt_j\}$ by flipping a coin $\beta \in \{0,1\}$. If $\beta = 0$, set $receipt_i = C$ and $receipt_j = C'$; otherwise, set $receipt_i = C'$ and $receipt_j = C$. Finally, $\mathcal{S}$ sends $\{receipt_i, receipt_j\}$ to $\mathcal{A}$ and adds them to the bulletin board.
5. $\mathcal{A}$ sends a receipt $res \in \{receipt_i, receipt_j\}$ back to indicate a vote for Alice.
   - When $\mathcal{C}$'s random coin is $\alpha = 0$, $\{receipt_i, receipt_j\}$ represent two receipts for Alice and Bob respectively and $\mathcal{S}$ is playing a faithful receipt freeness game with $\mathcal{A}$. Therefore, the probability that $\mathcal{A}$ can identify Alice's receipt is $\frac{1}{2} + \epsilon$.
   - When $\mathcal{C}$'s random coin is $\alpha = 1$, $\{receipt_i, receipt_j\}$ represent two receipts for Bob and $\mathcal{S}$ is not playing a faithful receipt freeness game with $\mathcal{A}$. Nevertheless, we can still ask $\mathcal{A}$ to pick one receipt res and send it back. Due to the fact that $receipt_i$ and $receipt_j$ have the identical distribution in this case, the probability that res equals to either of $\{C, C'\}$ is exactly $\frac{1}{2}$.
6. After receiving res from $\mathcal{A}$, $\mathcal{S}$ answers the challenge from the $\mathcal{C}$ as follows.
   - If the res is the Naor–Yung ciphertext pair C generated by $\mathcal{C}$, $\mathcal{S}$ outputs a guess $\alpha' = 0$.
   - If res is the Naor–Yung ciphertext pair C' generated by $\mathcal{S}$, $\mathcal{S}$ randomly selects α' from $\{0,1\}$ as a guess.

Depending on the value of α, various probabilities are summarised in Table 1.

$\alpha = \mathbf{0}$	$\alpha = \mathbf{1}$
$C = (CT_A, CT_B), C' = (CT'_B, CT'_A)$	$C = (CT_B, CT_A), C' = (CT'_B, CT'_A)\}$
$\Pr[res = (CT_A, CT_B)] = \frac{1}{2} + \epsilon$ $\Pr[res = (CT'_B, CT'_A)] = \frac{1}{2} - \epsilon$	$\Pr[res = (CT'_B, CT'_A)] = \frac{1}{2}$ $\Pr[res = (CT_B, CT_A)] = \frac{1}{2}$
$\Pr[\alpha' = \alpha \mid res = (CT_A, CT_B)] = 1$ $\Pr[\alpha' = \alpha \mid res = (CT'_B, CT'_A)] = \frac{1}{2}$	$\Pr[\alpha' = \alpha \mid res = (CT_B, CT_A)] = 0$ $\Pr[\alpha' = \alpha \mid res = (CT'_B, CT'_A)] = \frac{1}{2}$

Table 1: Probability Summary

In summary, $\mathcal{S}$'s advantage in winning the IND-CCA2† game against the Naor–Yung scheme is:

$$\varepsilon' = |\tfrac{1}{2}(\tfrac{1}{2} + \epsilon + \tfrac{1}{2}(\tfrac{1}{2} - \epsilon)) + \tfrac{1}{2}(\tfrac{1}{2} \cdot \tfrac{1}{2}) - \tfrac{1}{2}| = \tfrac{\varepsilon}{4}.$$

Based on Theorem 2.4 and Theorem 2.5, ε' is negligible so that ε is also negligible. Recall that $|\epsilon - \epsilon^*|$ is negligible. As a result, ϵ^* is negligible and the theorem follows. ■

they reveal the candidate order which contradicts with the challenge. In REB or RCO the order of the candidates get known directly. On the other hand, in Vote, the order can get to be known by querying these oracles as votes then tallying the bulletin board using RBB.

6 Conclusion

Prêt à Voter has been introduced in [23] as an end-to-end verifiable, receipt free voting scheme. Several formal method analysis exist in the literature to prove the properties of the scheme [24, 33]. In this paper we provide a proof of receipt freeness of Prêt à Voter using computational models. In order to do so, we defined a new security model that presents receipt freeness for Prêt à Voter . We show that creating the onions on the ballots with an IND–CCA2† secure encryption scheme is sufficient to achieve receipt freeness. We propose using Naor–Yung encryption for that purpose and that helps in maintaining the properties of one of the latest versions of Prêt à Voter [32]. We prove that the existence of an adversary that wins the receipt freeness security model implies the existence of a simulator that can break the IND-CCA2† security of Naor–Yung transformation. Extending this work to cover a larger family of voting schemes and to include stronger security notions such as receipt freeness is left for future work.

References

[1] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 544–553, New York, NY, USA, 1994. ACM.

[2] David Bernhard, Vèronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In *(ESORICS'11)*, 2011.

[3] Richard Buckland and Roland Wen. The future of e-voting in australia. *IEEE Security and Privacy*, 10(5):25–32, 2012.

[4] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. A supervised verifiable voting protocol for the victorian electoral commission. In *Electronic Voting*, pages 81–94, 2012.

[5] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. Using prêt à voter in victorian state elections. In *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'12, pages 1–1, Berkeley, CA, USA, 2012. USENIX Association.

[6] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *EUROCRYPT'87*, LNCS, pages 127–141. Springer.

[7] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing It. In *CRYPTO'86*, volume 263 of *LNCS*, pages 200–212. Springer.

[8] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[9] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, jul 2009.

[10] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18. Springer, 1985.

[11] Benaloh Josh and Tuinstra Dwight. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, 1994.

[12] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05*, pages 61–70. ACM Press, 2005. See also `http://www.rsa.com/rsalabs/node.asp?id=2860`.

[13] Ralf Küsters and Tomasz Truderung. An epistemic approach to coercion-resistance for electronic voting protocols. *CoRR*, abs/0903.0802, 2009.

[14] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *CSF*, pages 122–136, 2010.

[15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Proving coercion-resistance of scantegrity ii. In *ICICS*, pages 281–295, 2010.

[16] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *IEEE Symposium on Security and Privacy*, pages 538–553, 2011.

[17] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy . In *Crypto'06*, volume 4117 of *LNCS*, pages 373–392. Springer-Verlag, 2006.

[18] Tal Moran and Moni Naor. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Secur.*, 13:16:1–16:43, March 2010.

[19] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *In Proc. of the 22nd STOC*, pages 427–437. ACM Press, 1995.

[20] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols Workshop*, pages 25–35, 1997.

[21] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[22] Torben P. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *EUROCRYPT'91*, number 547 in LNCS, pages 522–526. Springer, 1991.

[23] Peter Y. A. Ryan. A variant of the chaum voter-verifiable scheme. In *Issues in the theory of security*, WITS, pages 81–88. ACM, 2005.

[24] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: a voter-verifiable voting system. *Trans. Info. For. Sec.*, 4:662–673, December 2009.

[25] Peter Y. A. Ryan and Vanessa Teague. Pretty Good Democracy. In *Proc. of the 17th Security Protocols Workshop*, LNCS. Springer, 2009.

[26] Peter Y.A. RYAN, David BISMARK, James HEATHER, Steve SCHNEIDER, and ZHE XIA. The Prêt à Voter Verifiable Election System. In *Transactions in Information Security and Forensics*. IEEE, 2009.

[27] P.Y.A. Ryan. A variant of the chaum voting scheme. Technical Report CS-TR-864, UNT, 2004.

[28] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:543, 1999.

[29] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer.

[30] Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *ASIACRYPT 05*, volume 3788 of *LNCS*, page 273292. Springer-Verlag.

[31] Z. Xia, S. Schneider, J. Heather, P. Ryan, D.Lundin, R. Peel, and P. Howard. Prêt à Voter: All-In-One. In *IAVoSS (WOTE'07)*, 2007.

[32] Zhe Xia, Chris Culnane, James Heather, Hugo Jonker, Peter Y. A. Ryan, Steve Schneider, and Sriramkrishnan Srinivasan. Versatile Prêt à Voter : Handling Multiple Election Methods with a Unified Interface. In *INDOCRYPT*, volume 6498 of *LNCS*, pages 98–114. Springer-Verlag, 2010.

[33] Zhe Xia, Steve A. Schneider, James Heather, and Jacques Traoré. Analysis, improvement and simplification of prêt à voter; voter with paillier encryption. In *Proceedings of the conference on Electronic voting technology*, pages 13:1–13:15, Berkeley, CA, USA, 2008. USENIX Association.

A Analysis of IND-CCA2†

Proof of Theorem 2.4. We first show that an IND-CCA2† secure scheme is also IND-CCA2 secure (namely the "if" part). Suppose that the scheme is not IND-CCA2 secure, then some attacker $\mathcal{A}$ has a non-negligible advantage in the game shown in Fig. 1. Then we construct an attacker $\mathcal{A}^\dagger$, which runs $\mathcal{A}$ as a subroutine to attack the encryption scheme in the game shown in Fig. 2. The attack is described in Fig. 9.

In this attack, $\mathcal{A}^\dagger$ can faithfully answer $\mathcal{A}$'s decryption queries as long as the two ciphertexts in E_d are not queried by $\mathcal{A}$. We take $d = 0$ as an example, where $E_0 = (\mathsf{Enc}(M_0, pk), \mathsf{Enc}(M_1, pk))$. In this case, $\mathsf{Enc}(M_0, pk)$ is not allowed to be the input of the decryption oracle in the game shown in Fig. 1 because it is the challenge, so that it will not be queried. We only need to show that $\mathcal{A}$ can only generate $\mathsf{Enc}(M_1, pk)$ with a negligible probability. This is a straightforward fact from the IND-CPA security

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker $\mathcal{A}^\dagger$ is given pk, and it sends pk to $\mathcal{A}$. If $\mathcal{A}$ issues a decryption query with C, $\mathcal{A}^\dagger$ submits C to the challenger. Once $\mathcal{A}^\dagger$ receives $\mathsf{Dec}(C, sk)$ from the challenger, it forwards this value to $\mathcal{A}$. When $\mathcal{A}$ submits M_0, M_1 for a challenge, $\mathcal{A}^\dagger$ send them to the challenger.
3. *Challenge*. Once receiving E_d from the challenger, $\mathcal{A}^\dagger$ sends the first ciphertext of E_d to $\mathcal{A}$. In more detail, if $d = 0$, then $\mathsf{Enc}(M_0, pk)$ is sent; otherwise $\mathsf{Enc}(M_1, pk)$ is sent.
4. *Phase 2*. The attacker $\mathcal{A}^\dagger$ deals with $\mathcal{A}$'s decryption queries in the same way as in *Phase 1*. If $\mathcal{A}$ terminates by outputting a guess b', then $\mathcal{A}^\dagger$ terminates by outputting a guess $d' = b'$.

Figure 9: IND-CCA2† Attack

property: if $\mathcal{A}$ can predicate the output of Enc for the input M_1, then it can trivially tell the ciphertext of M_1 from anything else and break the IND-CPA security. As a result, $\mathcal{A}^\dagger$ faithfully answer $\mathcal{A}$'s decryption queries! It is clear that $\mathcal{A}$ wins if and only if $\mathcal{A}^\dagger$ wins, so that $\mathcal{A}^\dagger$ has a non-negligible advantage. This conflicts with the assumption that the scheme is IND-CCA2† secure, hence, the scheme should be IND-CCA2 secure.

We next show that an IND-CCA2 secure scheme is also IND-CCA2† secure (namely the "only if" part). Suppose that the scheme is not IND-CCA2† secure, then some attacker $\mathcal{A}^\dagger$ has a non-negligible advantage ϵ in the game shown in Fig. 2. Then we construct an attacker $\mathcal{A}$, which runs $\mathcal{A}^\dagger$ as a subroutine to attack the encryption scheme in the game shown in Fig. 1. The attack is described in Fig. 10.

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker $\mathcal{A}$ is given pk, and it sends pk to $\mathcal{A}^\dagger$. If $\mathcal{A}^\dagger$ issues a decryption query with C, $\mathcal{A}$ submits C to the challenger. Once $\mathcal{A}$ receives $\mathsf{Dec}(C, sk)$ from the challenger, it forwards this value to $\mathcal{A}^\dagger$. When $\mathcal{A}^\dagger$ submits M_0, M_1 for a challenge, $\mathcal{A}$ sends them to the challenger.
3. *Challenge*. Once receiving C_b from the challenger, $\mathcal{A}$ selects $t \in_R \{0, 1\}$ and sends $E_d = (C_b, \mathsf{Enc}(M_t, pk))$ to $\mathcal{A}^\dagger$. Note that, if $t \neq b$ then E_d is a valid challenge, otherwise it is not valid.
4. *Phase 2*. The attacker $\mathcal{A}$ deals with $\mathcal{A}^\dagger$'s decryption queries in the same way as in *Phase 1*. If $\mathcal{A}^\dagger$ terminates by outputting a guess d', then $\mathcal{A}^\dagger$ terminates by outputting a guess $b' = d'$.

Figure 10: IND-CCA2 Attack

In this attack, $\mathcal{A}$ can faithfully answer $\mathcal{A}^\dagger$ decryption queries but the challenge is faithfully generated only when $t \neq b$. Moreover, when $t \neq b$, $\mathcal{A}$ wins if and only if $\mathcal{A}^\dagger$ wins. Therefore, if $t \neq b$, $\mathcal{A}$'s advantage is ϵ. When $t = b$, suppose $\mathcal{A}$'s advantage is ϵ', which is a positive number and can be negligible with respect to the security parameter. As a result, $\mathcal{A}$'s overall advantage is $\frac{\epsilon+\epsilon'}{2}$, which is non-negligible given that ϵ is non-negligible. This conflicts with the assumption that the scheme is IND-CCA2 secure,

hence, the scheme should also be IND-CCA2† secure.

B Non-interactive Zero knowledge proofs

We start with some notations and conventions. Let $\mathcal{H}$ denote a hash function and (p, q, g) be cryptographic parameters, where p and q are large primes such that $q \mid p-1$ and g is a generator of the multiplicative subgroup $\mathbb{Z}_p^*$ of order q.

B.1 Knowledge of discrete logs:

Proving knowledge of x, given h where $h \equiv g^x \bmod p$ [6, 7, 29].

- Sign. Given x, select a random nonce $w \in_R \mathbb{Z}_q^*$ and compute, Witness $g' = g^w \bmod p$, Challenge $c = \mathcal{H}(g') \bmod q$ and Response $s = w + c \cdot x \bmod q$. Output Signature (g', s).
- Verify. Given h and signature (g', s), check $g^s \equiv g' \cdot h^c \pmod p$, where $c = \mathcal{H}(g') \bmod q$.

A valid proof asserts knowledge of x such that $x = \log_g h$; that is, $h \equiv g^x \bmod p$.

B.2 Equality between discrete logs:

Proving knowledge of the discrete logarithm x to bases $f, g \in \mathbb{Z}_p^*$, given h, k where $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$ [8, 22].

- Sign. Given f, g, x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute Witnesses $f' = f^w \bmod p$ and $g' = g^w \bmod p$, Challenge $c = \mathcal{H}(f', g') \bmod q$ and Response $s = w + c \cdot x \bmod q$. Output signature as (f', g', s)
- Verify. Given f, g, h, k and signature (f', g', s, c), check $f^s \equiv f' \cdot h^c \pmod p$ and $g^s \equiv g' \cdot k^c \pmod p$, where $c = \mathcal{H}(f', g') \bmod q$.

A valid proof asserts $\log_f h = \log_g k$; that is, there exists x, such that $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$. Note that this proof of knowledge can be used to prove equality of plaintexts for two ElGamal ciphertext encryptions such a proof is referred to as plaintext equivalence proof (PEP).

B.3 Plaintext Equivalence Proof

This proof relies on Equality between discrete logs (see appendix B.2). Let $CT_1 = Enc(pk_1, m, \zeta_1) = (u_1, v_1) = (g^{\zeta_1}, {h_1}^{\zeta_1} g^m)$, $CT_2 = Enc(pk_2, m, \zeta_2) = (u_2, v_2) = (g^{\zeta_2}, {h_2}^{\zeta_2} g^m)$, where $h_1 = g^{sk_1}$ and $h_2 = g^{sk_2}$. The PEP is as follows:

- Compute $e_1 = {h_1}^{\zeta_2}$, and $e_2 = {h_2}^{\zeta_1}$
- Compute two Zero knowledge proofs of Equality between discrete logs. One between (u_1, e_2) and one between (u_2, e_1). This is to prove that e_1, and e_2 are well formed.
- Compute $e_3 = \frac{u_1}{u_2} = g^{\zeta_1 - \zeta_2}$
- Compute $e_4 = \frac{v_1}{v_2} \cdot {e_1}^{-1} \cdot e_2 = (h_1 \cdot h_2)^{\zeta_1 - \zeta_2}$

The PEP is a proof of knowledge of the equality of the exponent $\zeta_1 - \zeta_2$ in (e_3, e_4) to the bases $(g, h_1 \cdot h_2)$. Given we have a Fiat–Shamir proof one can add to the hash function used in appendix B.2 the order of the ciphertexts that we are testing. In other words, commit to the order of either (CT_1, CT_2) or (CT_2, CT_1) such that if an adversary manipulates the order the proof fails to verify.

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

Rethinking Voter Coercion: The Realities Imposed by Technology

JOSH BENALOH, Microsoft Research

When the Australian secret ballot was introduced in the 1850s, it not only provided privacy for those voters who wanted it, but it also effectively eliminated coercion by allowing no viable means for voters to prove their votes to third parties. In an environment where the privacy of voters is enforced by independent observers, coerced voters could freely express their true preferences while making their selections.

In contrast, modern technologies render the traditional poll-site protections largely ineffective, and the limited remaining options for preserving these protections will almost certainly disappear in the not-too-distant future. Today, in-person voters routinely carry video recording equipment and other technologies that facilitate coercion into polls, and although not yet ubiquitous, inexpensive and unobtrusive wearable video recording devices are readily available. In view of these realities, it is appropriate to re-examine the efforts and countermeasures currently employed and explore what defenses are possible and reasonable against various forms of voter coercion.

1. INTRODUCTION

The Australian ballot is regarded as the gold standard of voting. For over a century, coercion has been effectively thwarted by the process of compelling voters to mark and cast their ballots in an environment where their privacy is enforced by independent monitors. Although they are common in many jurisdictions, remote voting systems such as vote-by-mail are generally regarded as providing inferior protection from coercion, and as such, their use is often discouraged by experts.

The design of new voting systems is heavily influenced by measures intended to thwart coercion. If they can't match the gold standard of the Australian ballot, these systems risk out of hand rejection. To meet "best practices", new systems often include elaborate and complex procedures such as printing behind a screen so that a commitment made by a voting device cannot be seen by a voter until a subsequent step in the process ([Neff 2004], [Chaum et al. 2005], [Moran and Naor 1997], [Benaloh 2007], [Benaloh 2008], [Chaum et al. 2009], [Küsters and Truderung 2009], [Küsters et al. 2010], [Juels et al. 2010], [Araujo et al. 2010]). However, the ubiquity of cell phones gives many voters an easy means to record their entire voting sessions and show them to third parties. Even more alarming, video cameras are now available surreptitiously embedded in clothing and eyeglasses,[1] and no effective means to eliminate this threat is evident. As miniaturization continues, costs decrease, and new applications and innovations are promulgated, attempts to prevent voters from recording their voting sessions will become increasingly futile. With current technologies, a coercer can demand that a victimized voter wear a video recorder, establish identity by standing before a mirror (perhaps in a bathroom near a poll site), and then move into a poll site to vote. The entire unbroken process can be recorded as a continuous video stream. Anyone wishing to voluntarily sell a vote can produce a similar video stream.

Because of these emerging threats, the gold standard of the Australian ballot is losing its luster, and the hope of preventing all forms of voter coercion is waning. In this light, it is appropriate to categorize the different forms of coercion and analyze what forms of coercion can reasonably be controlled and to also address the impacts of these realities on election system design.

2. VARIETIES OF COERCION AND VOTE-SELLING

There are many different types of coercion and vote-selling, but they can be partitioned and classified across a small number of axes. It will be seen that, perhaps surprisingly, some axes which seem significant are not. For instance, it turns out that from the perspective of mitigation, vote-selling (voluntary) and coercion (involuntary) are surprisingly similar. For this reason, the general term *co-*

[1] For example, Spy Net Stealth Video Glasses are available for under 30 US dollars and KJB Security Camcorder Glasses are available for under 20 US dollars. Both have ample memory to record an entire visit to a poll.

ercion will often be used to describe both cases. A more thorough treatment of the distinctions will be given below.

2.1. Time of Coercion

It is very difficult to imagine strong defenses against a concerted pre-planned attack by a determined adversary. The example where a voter wears a hidden camera to first capture the voter's identity (either directly or with a mirror) and then record the voter's complete interactions within a poll site is compelling. However, such an effort must be planned. As wearable cameras become more common, it might be possible for voters to inadvertently record themselves voting and then subsequently decide to offer this recording as proof of a particular selection, but it is not yet likely that a voter can be coerced into revealing a vote after the fact. People who are not being coerced and have no intention to subsequently sell their votes can generally be expected to heed requests to turn off recording equipment prior to voting. Thus, under "normal" circumstances, a voter who has not been coerced and follows well-constructed instructions through the voting process should not be coercible afterwards.

It would therefore seem that an important design principle of voting systems would be to preclude "after the fact" coercion. Examples of systems that fail to satisfy this principle would include systems that do not instruct voters to disable recording devices and systems that provide voters with (authenticated) take-home receipts that reveal their selections.

There is a close analogy here with *deniable encryption*. The original protocols for deniable encryption ([Canetti et al. 1997]) put substantial effort into allowing the sender of an encrypted message to avoid divulging its contents to a coercer. However, the protocol required randomness, and a "before the fact" attacker could simply provide the target with a source of randomness and require its use — thereby defeating the protocol. By contrast, any randomized encryption protocol that calls for the sender to delete the relevant random values before sending provides a strong defense against an "after the fact" request to disclose the encrypted value. The conclusion there was that any common randomized encryption scheme can provide good protection against a post-protocol disclosure request, but no system can provide full protection against a coercive attack that is launched prior to the execution of the encryption protocol unless a secondary "untappable" channel is available to which the attacker has no access.

Analysis of an assumption of a physically untappalbe channel for elections goes back to at least [Benaloh and Tuinstra 1994] and has been leveraged heavily in well-known protocols such as [Hirt and Sako 2000] and [Juels et al. 2010]. [Hirt 2001] explores the limits of voting without this assumption and makes a strong case that coercion cannot be prevented without an untappable channel. While a properly constructed and managed physical poll-site could serve as an embodiment of an untappable channel a decade ago, surreptitious wearable video cameras now render poll-sites as an ineffective means of achieving an untappable channel, and no alternative is evident. Without the ability to construct an untappable channel, pre-election coercion is virtually impossible to thwart in practice.

A middle ground may be available if coercion takes place after registration but before voting. Some remote voting protocols such as [Juels et al. 2010] assume that voters are able to obtain information during registration that protects them from coercion during subsequent voting. If voters are coerced prior to registration, the same surreptitious video recording technique can be used by a coercer to obtain credentials or other data from prospective voters. However, voters who are not coerced until after registration may be protected so long as the protocol is engineered to not allow the voter to offer "proof" of the secret registration data. In [Juels et al. 2010] there is an additional assumption that voters have some private time to use their credentials to cast their true votes — although one can imagine protocols where this additional assumption is not necessary.[2]

[2]For example, a voter may be able to deposit a "flip my vote" chit during registration. The voter may then be able to subsequently vote in view of a coercer who doesn't know that this chit has been deposited. More sophisticated versions of

Some remote protocols (e.g. Helios - [Adida 2008]) attempt to use a "last vote counts" strategy to allow voters who have been coerced to return later to cast their true votes. However, this class of strategies is easily thwarted by a coercer who requires voters to vote at the end of the voting period or to continue uninterrupted video recording from the time of the vote through the end of the voting period.

While it may initially appear impractical for a coercer to monitor the video recordings of a large number of voters, two observations show that wholesale coercion is both possible and practical. First, video recordings can be viewed at a much faster speed than that at which they were made. This, for example, makes it practical for a coercer to observe that a voter who cast a vote early in the day did not return to cast a subsequent overriding vote. Second, and even more devastating, is the opportunity for coercers to use simple random sampling. An employee who risks the loss of a job for not voting as instructed may be unlikely to attempt to skirt the coercion – even if the chance of the employer viewing the video and discovering the employee's "indiscretion" is only 1 in 100.

2.2. Voluntary Vote-Selling and Involuntary Coercion

While it may initially seem as though there might be a substantial difference between involuntary coercion and voluntary vote-selling, closer examination shows that the differences are small. In the pre-election case, a voter who can be involuntarily coerced can voluntarily ask to be coerced by a potential buyer of the vote, and a voter who can convince a purchaser of the contents of a vote can offer the same proof to a coercer. Although this is by no means a formal reduction, it demonstrates a rough equivalence between the two cases.

There are, of course, some differences between the two cases. It is easier, for instance, to purchase 100 votes from willing sellers than to exhort 100 otherwise unwilling individuals with threats and intimidation, and large-scale coercion is likely to be even more difficult if the protocol requires it to be initiated well before the actual voting takes place. However, if we focus attention on a single voter and a well-resourced attacker who can make good on viable threats, then the unwillingly-coerced voter has little more protection than the voter willing to sell a vote.

One might look for a distinction by exploring a case where a voter takes pre-election steps in order to sell a vote to an (as yet unknown) post-election purchaser. But there are problems with this scenario. First, once a voter has completed a vote, there is no clear advantage to a potential vote-buyer to pay for a vote that has already been cast and can no longer be changed. Second, even if we imagine that a voter is simply aware of a vague rumor that a particular vote might be rewarded after the election and that rumor turns out to be true and is backed up by post-election specifics, a voter can — as with the involuntary case — preserve a full record (video recording, derive "randomness" deterministically from a preserved seed, etc.)

In the post-election case, it's clear that good secret-ballot election protocols can and should allow neither voluntary nor involuntary disclosure of a voter's selections. Once the voting process is complete, a voter who has not taken any prior actions to preserve the record of a vote should be unable to prove details about a vote to third-parties, and virtually every extent election protocol satisfies this property.

2.3. Local and Remote Coercion

Traditionally, local coercion has been seen as easier to implement than remote coercion. A coercer who can be located within a polling site and who can watch and perhaps even signal a voter can have more coercion options that one who is not present. Some voting protocols (e.g. STAR-vote [Benaloh et al. 2012] and VeriScan [Benaloh 2008]) allow a voter to make a choice — after committing to the contents of a ballot — as to whether to cast or spoil that ballot, and if the spoil option is exercised, the contents of the ballot are publically revealed. In this scenario, a voter who is being coerced locally would not dare violate the wishes of the coercer because the coercer might signal

---

this approach include depositing a chit to indicate the number of positions to shift a ballot or simply pre-voting with the instruction to quietly ignore a subsequent vote.

that the ballot should be spoiled and made public. It might seem as though a voter whose coercer is not present would not face these concerns and would be free to vote without coercion. However, the distinction disappears when a complete video recording of the voting session is possible. A continuous recording that shows the selections made by the voter and the voter then casting the ballot with these selections makes it unnecessary to have a coercer be present in a poll site.

2.4. Retail and Wholesale Coercion

A coercer who can expend the resources to watch a voter through the entire period during which polling is open may have more abilities than one who can only spend a few minutes with each voter. If so, this suggests that there is substantial benefit to election protocols which allow voters to override prior votes with new votes.

Once again, however, a video recording can act as a surrogate for an active human enforcer. In the extreme, a coerced voter can make a video recording that includes a full voting session and all subsequent actions taken until the polls are closed and the results are announced. A continuous and uninterrupted recording of this process can serve as compelling evidence of how someone voted and that there was no opportunity to make subsequent changes. A coercer with limited resources need not even expend the resources to view the recordings of all coerced voters; instead, the reviewing of a small random sampling of the recordings can be a sufficient deterrent to prevent large numbers of voters from voting their true preferences, and any selected recordings can be viewed at high speed to reduce the resources expended in coercion.

3. OTHER VOTING METHODS

If the gold standard of in-person voting with the Australian ballot has lost much of its luster, how does it compare with the alternatives?

Vote-by-mail is the most prevalent alternative to poll-site voting, and this form of voting is already regarded as highly susceptible to coercion. As with poll-site voting, coercion should only be effective if initiated "before the fact", so there is little difference in that respect. No special technology is required to coerce a postal voter, so in that respect poll-site voting is still somewhat stronger. However, an argument can be made that a wholesale attack is now harder to mount on a postal voting system than on a poll-site voting system. This is because the most effective postal voting attack is probably to have voters physically transfer their ballots and (signed) envelopes to the coercer, while video recordings of poll-site voting sessions can be transferred electronically and spot-checked randomly. Although postal voters could also be coerced to provide video recordings, the process is harder to enforce because postal voting systems often allow voters to obtain multiple ballots and generally use a "*first* valid vote counts" strategy for dealing with multiple votes.

Although it is a far less common medium, several Internet voting pilots have been conducted in recent years. Most Internet voting systems make little or no effort to prevent coercion and are quite vulnerable. The simplest coercive attack on these systems is usually for a coercer to collect credentials from voters. A few Internet voting systems make valiant efforts to subvert coercion (e.g. [Juels et al. 2010] allows voters to produce "fake" credentials that coercers cannot distinguish from real credentials), but these protocols are still susceptible to coercion begins before registration rather than before voting. Another attack vector for Internet voting systems is malware which an attacker can even *require* a voter to install to monitor the voter's actions. A large wholesale coercive attack could therefore be employed by simply instructing targeted voters to install specialized malware, and the verification process could be completely automated. This is a vulnerability that goes well beyond those of poll-site and postal voting.

It's somewhat ironic that voter choice is *not* beneficial in deterring coercion. More options in the hand of a voter become more options in the hand of a coercer. If a voter today is given a choice between voting by mail and voting in-person at a poll site that rigidly enforces rules against video recording, then a coercer can simply insist that the vote-by-mail option be exercised. In a real sense, a multi-modal voting system is no stronger than its weakest mode.

4. PRACTICAL IMPLICATIONS AND OPPORTUNITIES

Given the realities described above, what are the practical implications on election protocol design?

There seems to be little point to adding significant complexity to election protocols in an increasingly futile attempt to defeat pre-election coercion. While protocols should not facilitate post-election coercion by giving voters the means to disclose their votes after the fact, the ease with which ever more ubiquitous video recording technologies can defeat measures designed to prevent pre-election coercion is alarming.

The metaphor of a chain being only as strong as its weakest link is apt. While measures can be taken that effectively deter some forms of pre-election coercion, there is no justification for employing countermeasures against coercion methods which are more difficult to execute than simply pushing a "start recording" button. Voters should not be encumbered with burdensome extra processes whose only effects are to prevent sub-optimal coercion vectors, and electoral systems which remove these barriers should not be considered inferior.

Although it is not the focus of this work, it may even be worth considering whether the coercion potential of unmonitored voting systems such as vote-by-mail remains a valid argument against its use.

It is more than a little surprising to think that the best defense against coercion today may *not* be found in an in-person voting system. Instead, the approach that offers the most resistance to coercion may be a remote voting system using the paradigm of [Juels et al. 2010] in which voters who have successfully registered without coercion can convincingly pretend to vote according to a subsequent coercers wishes while secretly voting their own true preferences. It may be possible to leverage this approach by allowing voters who have been coerced during registration to quietly invalidate their coerced credentials at some later opportunity and to receive new valid credentials. This approach would, however, have similar weaknesses to the defense of allowing voters to vote multiple times with only the last vote counting — specifically, a coercer could force voters to register under coercion near the deadline and prevent any further changes until the deadline has passed.[3] Another concern is that a system which enables voters to silently replace their credentials would likely allow election officials to silently replace voters' credentials without alerting the affected voters. Nevertheless, this direction may offer some opportunities for research into protocols which can allow voters to appear to be asceding to the demands of a coercer while taking advantage of priavte moments to express their true preferences.

5. CONCLUSIONS

Defenses against coercion and vote-selling have historically been a major component of voting system design. While these threats remain legitimate, technologies which easily defeat pre-election coercion countermeasures are now readily available. As such, it is now appropriate to consider the value and effectiveness of continuing to incorporate such countermeasures and to begin exploring the options that may be available for voting systems that are free from the burdens imposed by these now ineffective countermeasures. While steps can and should be taken to prevent post-election coercion, we should also be realistic and admit to ourselves that we have no effective technical means to prevent simple, economically-scalable, remotely-enforced, pre-election coercion.

Surrender is not an attractive option. But squandering scarce resources in support of a hopeless cause is even worse. Election systems should be designed to have the best achievable combination of properties — including integrity, privacy, reliability, and usability. Protection from prior coercion is a crucial element of privacy, but we should be realistic about what we can and cannot achieve and not weaken other elements in an illusory attempt to protect a lost element.

[3]It may, however, be plausible for registration centers to enforce higher standards of scrutiny to deter coercion during, say, the final week of registration.

REFERENCES

B. Adida. 2008. Helios: Web-Based Open-Audit Voting. In *Proceedings of the Usenix Security Conference*. USENIX Association, 14.

R. Araujo, N. Ben Rajeb, R. Robbana, and J. Traoré. 2010. Towards Practical and Secure Coercion-Resistant Electronic Elections. In *Proc. CANS 2010,Cryptology and Network Security*. 278–297.

J. Benaloh. 2007. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *Proc. 2007 Electronic Voting Technology Workshop*.

J. Benaloh. 2008. Administrative and Public Verifiability: Can We Have Both?. In *Proc. 2008 Electronic Voting Technology Workshop*.

J. Benaloh, M. Byrne, P. Kortum, N. McBurnett, O. Pereira, P. Stark, and D. Wallach. 2012. *STAR-Vote: Secure, Transparent, Auditable, and Reliable Voting*. Technical Report. http://arxiv.org/abs/1211.1904.

J. Benaloh and D. Tuinstra. 1994. *Uncoerciible Communication*. Technical Report TR-MCS-94-1. Clarkson University.

R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. 1997. Deniable Encryption. In *Proc. Crypto 1997*.

D. Chaum, R. Carback, J. Clark, A. Essex, S. Popovenuic, R. Rivest, P. Ryan, E. Shen, A. Sherman, and P. Vora. 2009. Scantegrity II: End-to-End Verifiability by Voters of Optican Scan Elections Through Confirmation Codes. *IEEE Transactions on Information Forensics and Security* 4, 4 (2009), 611–627.

D. Chaum, P. Ryan, and S. Schneider. 2005. A ractical Voter-Verifiable Election Scheme. In *Proc. ESORICS 2006, European Symposiums on Research in Computer Security*. 313–326.

M. Hirt. 2001. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. Ph.D. Dissertation. ETH Zurich.

M. Hirt and K. Sako. 2000. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *Proc. Eurocrypt 2000*.

A. Juels, D. Catalano, and M. Jakobsson. 2010. Coercion-Resistant Electronic Elections. *Towards Trustworthy Elections* (2010), 37–63.

R. Küsters and T. Truderung. 2009. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *Proc. IEEE Symposium on Security and Privacy*. 251–266.

R. Küsters, T. Truderung, and Andreas Vogt. 2010. Proving Coercion-Resistance of Scantegrity II. In *Proc. ICICS 2010, International Conference on Information and Communications Security*. 281–295.

T. Moran and M. Naor. 1997. Receipt-free universally-veriable voting with everlasting privacy. In *Proc. Crypto 2006*.

C. A. Neff. 2004. Practical High Certainty Intent Verication for Encrypted Votes. (2004). (unpublished).

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

Improved Support for Machine-Assisted Ballot-Level Audits

Eric Kim, University of California, Berkeley
Nicholas Carlini, University of California, Berkeley
Andrew Chang, University of California, Berkeley
George Yiu, University of California, Berkeley
Kai Wang, University of California, San Diego
David Wagner, University of California, Berkeley

This paper studies how to provide support for ballot-level post-election audits. Informed by our work supporting pilots of these audits in several California counties, we identify gaps in current technology in tools for this task: we need better ways to count voted ballots (from scanned images) without access to scans of blank, unmarked ballots; and we need improvements to existing techniques that help them scale better to large, complex elections. We show how to meet these needs and use our system to successfully process ballots from 11 California counties, in support of the pilot audit program. Our new techniques yield order-of-magnitude speedups compared to the previous system, and enable us to successfully process some elections that would not have reasonably feasible without these techniques.

1. Introduction

Post-election audits form one of the most compelling tools for providing transparency and securing elections that use electronic technology. Recent research has shown that ballot-level machine-assisted audits [Calandrino et al. 2007; Benaloh et al. 2011] can offer significant improvements over current practice: significantly better assurance, at lower cost. Unfortunately, the voting systems currently deployed in the US do not support ballot-level post-election audits.

In this paper, we develop tools to facilitate ballot-level machine-assisted audits of elections conducted using current voting systems. We have been working with the State of California and various California counties to pilot new methods for ballot-level machine-assisted election audits. The approach involves re-tabulating the election using a second system that *was* designed from the start with support for ballot-level audits, checking that the second system selects the same winners as the official results, and then conducting an efficient ballot-level audit of the results from the second system [Lindeman et al. 2013]. This work focuses on the design of such a second system, called OpenCount, intended for this task.

One might wonder, why build a new system to re-tabulate the election from scratch? An alternative approach would be to extend deployed voting systems with the support needed for ballot-level machine-assisted audits. However, many of the major commercial vendors are focusing their development efforts primarily on their next-generation systems rather than on upgrading deployed systems; the deployed systems are proprietary, so it is not easy for third parties to develop extensions without vendor assistance; and updates to legacy systems may require that the entire system be first certified under new EAC standards, which may be impossible, as those systems were not designed to meet the new EAC standards. More fundamentally, many existing systems cannot be retrofitted in this way due to hardware limitations: in many deployed systems, the precinct-count optical scanners do not have the hardware capacity to record scanned images of all ballots, and there is no way to link each individual ballot to its scanned record. Therefore, pragmatically it may be easier to deploy ballot-level audits by re-tabulating the ballots using a second system that was designed with machine-assisted auditing in mind. That is the path we explore in this work. Of course, our work may be of direct relevance to future generations of voting systems that wish to provide support for efficient audits.

This work extends OpenCount [Wang et al. 2012] to provide better support for ballot-level audits, based upon our experience using it to support pilots in several California counties. This experience has helped us gain a better understanding of what is needed to allow ballot-level audits to be used in

This work was supported by National Science Foundation under grants CNS-0524745 and CCF-0424422.

practice. In particular, we identified two major shortcomings of the previous version of OpenCount. First, the previous version required election officials to compile a collection of all blank ballots (one of each possible ballot style). Due to the number of ballot types, this process proved to be labor-intensive for election officials and was a hurdle to deployment. Second, we learned that the previous version of OpenCount did not scale to large, complex elections. When there are many ballot styles, operator data entry of contest titles, candidate names, and ballot attributes (e.g., language, precinct, tally group) became extremely tedious and time-consuming. Each of these two is an independent barrier to being able to use OpenCount in large elections; either alone would be a showstopper, so we must solve both problems.

In this paper, we show how to solve both of these two problems. First, we develop new techniques to eliminate the need for scanned blank ballots. This allows us to re-tabulate an election given only scans of the voted ballots (and without access to the election database from the official voting system). Second, we develop new methods to reduce the human effort so that OpenCount will scale to large elections with many ballot types. We implement these improvements in OpenCount.

We found that these improvements are enough that we can now successfully handle large, complex elections and meet the needs of the California pilots. We evaluate the improved OpenCount on over 560,000 ballots from 12 different elections in 11 California counties and 1 Florida county. Our experiments show that these new methods enable order-of-magnitude speedups on medium-sized elections ($\sim$ 30k ballots) with many ballot styles, and enable us to process large elections ($\sim$ 120k double-sided ballots) that could not reasonably have been processed without them.

This paper makes the following contributions:

— We develop new methods to analyze and decode ballot styles, from scans of only the voted ballots (without needing scans of blank ballots or other external information). We show how to rapidly identify the ballot style of each voted ballot, how to reverse-engineer the structure of contests on the ballot, and how to recognize precinct numbers and decode barcodes that identify the ballot style of each voted ballot.

— We develop new methods to reduce the amount of human effort needed to re-tabulate an election. We show how to robustly identify multiple instances of the same contest on different ballot styles (so that the operator only needs to enter in candidate names once), how to associate voting targets with contests, how to infer the bounding box of each contest robustly, and how a human operator can verify that the results of these automated methods are accurate.

— We build a tabulation system that election officials can use to conduct ballot-level audits of their elections, and that researchers can use as a basis for future research. The system is open source and is publicly available at `https://code.google.com/p/opencount/`.

2. Related Work

OpenCount was inspired by the pioneering work of TEVS [Trachtenberg 2008] and Votoscope [Hursti 2005], which aim to solve the same problem. BallotTool [Lopresti et al. 2008] is a related system which assists an operator in retabulating a set of scanned ballots. Our work distinguishes itself in our tight integration of computer vision techniques with focused operator interaction. In [Smith et al. 2008], the authors develop a system that segments voter marks from the ballot form. However, no attempt is made to classify the marks as filled or unfilled. [Xiu et al. 2009] introduces specialized classifiers that fully automate the voter intent classification task. As future work, we intend to explore applying classifiers to further improve the ballot interpretation stages.

Document analysis is a field that considers many of the same challenges as ballot analysis. In document analysis, researchers develop systems that automatically infer the structure of documents such as forms or articles. The X-Y Cut algorithm [Nagy et al. 1992], shape-directed cover algorithm [Baird et al. 1990], and whitespace cover algorithm [Breuel 2002] are methods which excel at segmenting documents with white backgrounds and rectangular layouts. See [Mao et al. 2003; Namboodiri and Jain 2007] for a survey of the document analysis field.

Ballot analysis distinguishes itself in that it requires near-perfect accuracy in its interpretation of voted ballots. Thus, any approach must be designed with this requirement in mind - towards that

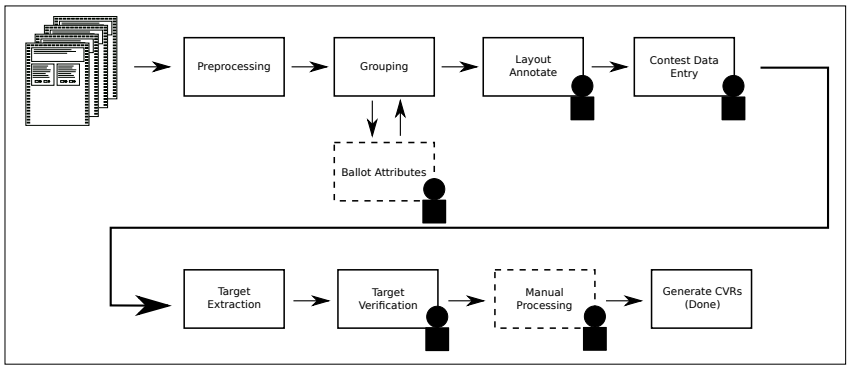


Fig. 1. Overview of the OpenCount architecture. The human outline indicates each steps that involve the operator. Steps with a dotted-line border may or may not be required.

end, OpenCount combines automated analysis with operator interaction to accurately and efficiently process ballots.

3. Overview of the OpenCount Architecture

Audit Process. OpenCount is designed to support a *transitive audit* [Lindeman and Stark 2012], where one re-tabulates the election using a second system (in this case, OpenCount), checks that the second system declares the same winners as the official results, and then audits the second system. We focus on elections conducted using paper ballots.

The audit process works as follows. After an election, election workers collect all the cast ballots and scan them all using an ordinary document scanner. Then, OpenCount processes those scanned images to extract cast vote records (CVRs) that describe the votes present on each ballot. Election officials tally the cast vote records, check that they declare the same winners as the official results, and then commit to the OpenCount CVRs. Finally, election officials conduct a public, risk-limiting audit of the ballots. During this audit process, officials repeatedly select a random ballot, pull the corresponding paper ballot, visually compare the marks on the paper ballot to the electronic CVR produced by OpenCount, and confirm that they match. Because the paper ballots are retained in the same order they were scanned, it is possible to uniquely associate each paper ballot to its corresponding CVR, which enables the ballot-level audit. Standard methods can be used to calculate how many ballots to examine and compute the level of confidence thus attained.

For instance, this process was successfully used in Napa County to audit their June 2012 primary election [Farivar 2012]. Election officials scanned 6,809 ballots, and OpenCount was used to process those scanned ballot images. Then, election officials used the CVRs produced by OpenCount to audit the election in a very close contest. Napa County officials audited 559 ballots, and found that in every case the votes on the ballots exactly matched the CVRs produced by OpenCount.

In this paper, we focus on the design of OpenCount and the algorithmic techniques needed to successfully analyze scanned images of ballots and produce CVRs. OpenCount is designed to avoid relying upon election definition files from the official voting system or other external information; we would like OpenCount to rely only on scanned images, with no further external information. The need to re-derive this information, the scale of elections, the diversity of ballot formats and voter marks, and the imperfections in scanned images make this a challenging image-processing task.

Architecture. There are four main stages to OpenCount's processing (Figure 1). First, *grouping* divides the ballots into groups, where all ballots within a group share the same ballot style (that is, the same layout and same contests). We present improvements to the grouping state that allows OpenCount to scale to larger elections. Second, in *layout annotation*, a human operator assists OpenCount in identifying the structure of the ballot (location of contests, candidates, and voting targets) and enters the titles of contests and the names of candidates. The operator only must annotate one ballot

from each group. We develop novel methods to reduce the workload of layout annotation on the operator; this is crucial to enabling us to scale. One challenge here is to analyze the structure of the ballot robustly despite the presence of voter marks. Third, *ballot interpretation* involves automated processing to identify voter marks and associate them with the corresponding contest and candidate. Fourth, and finally, in *target verification*, a human operator checks OpenCount's interpretation of voter marks and inspects ambiguous or marginal marks. The output is a cast vote record for each ballot that identifies all votes found on that ballot. Ballot interpretation and target verification remain mostly unchanged, compared to the previous version of OpenCount [Wang et al. 2012]; our new work primarily affects the first two stages.

4. Design

4.1. Terminology

A *voted ballot* is the ballot after a voter has marked his/her ballot and cast it. Each ballot contains a set of *contests*. A *contest* includes a list of *candidates*, with one voting target per candidate. A *voting target* is an empty oval, broken arrow, or other location on the ballot where the voter should mark her ballot, if she wants to indicate a vote for the associated candidate. A *cast vote record* (CVR) is a record of all selections made by the voter on a single voted ballot.

The *ballot style* is the set of contests found on the ballot as well as the visual organization and location of these contests on the ballot.[1] For example, an English-language ballot may have the same set of contests as a Spanish-language ballot, but because their text is different, we consider them as two different ballot styles. Similarly, two ballots may contain identical contests, but the order of candidates in the contests may not be the same; in this case, we consider them as distinct ballot styles. Ballots may also contain a precinct number or a tally group (e.g., absentee vs. polling-place) for accumulation.

Min/Max Overlay Verification. OpenCount uses *overlays* [Cordero et al. 2010] to help the human operator verify the correctness of automated computations. Overlays help the operator quickly verify that a set of images is identical, or carries the same meaning. For instance, in Figure 3(b) the operator can immediately verify that all the images contain the word "Democratic".

Let S be a set of grayscale images of uniform dimension. The *min*-overlay of S is the image S_{min} where the intensity value at (x,y) is the *lowest* intensity value at (x,y) out of every image in S. The *max*-overlay of S is the image S_{max} where the intensity value at (x,y) is the *highest* intensity value at (x,y) out of every image in S. Intuitively, if any image in S has a black pixel at (x,y), then so does S_{min}. Similarly, if S_{max} has a white pixel at (x,y), then at least one image in S does.

If the min and max overlays of the images in S suggest to the operator that not all the images in S match, then the operator can choose to *split* S into two smaller sets. The split operation uses the k-means algorithm [MacQueen 1967] (with $k = 2$) on the images in S. The feature representation of each image is simply the pixel intensities arranged row-by-row into a vector, with the L2 norm as the distance metric. This method works well, provided there are two image classes present that exhibit different spatial-visual distributions. If there are more than two image classes present in the set S, the operator may need to perform several consecutive splits to further refine the results until each cluster is homogeneous. See Figure 2(b,c) for an illustration.

4.2. Grouping

In the first step of the pipeline, OpenCount separates the input set of voted ballots into groups, so that all ballots within a given group have the same ballot style. This stage reduces the effort required in later stages: rather than asking a human operator to annotate each individual ballot, we ask the operator to annotate only one ballot from each group.

Table I summarizes the number of groups (the number of distinct styles) for a range of elections that we processed. Notice that the number of groups is orders of magnitude smaller than the number

[1] Previous versions of OpenCount required one (unmarked) *blank ballot* per ballot style.

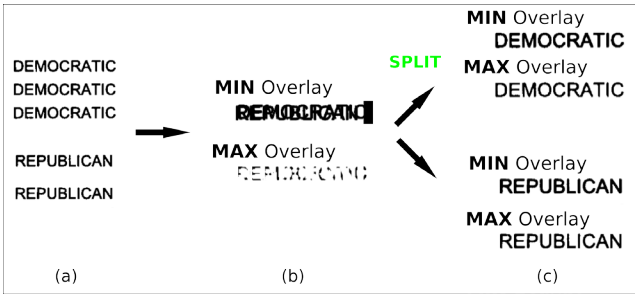


Fig. 2. The min/max overlay-based verification process. (a) The set of images. (b) The min and max overlays. (c) A single "split" operation, and the resulting overlays.

of voted ballots. Also, the number of groups does not necessarily scale linearly with the size of an election. Instead, the number of groups scales with the number of possible precincts, political party affiliations, languages, and tally groups. For instance, although Leon County has four times more voted ballots than Marin County, Leon actually has *fewer* groups than Marin.

In the previous version of OpenCount [Wang et al. 2012], we grouped ballots by matching each voted ballot to a corresponding blank ballot. However, this required election officials to gather and scan one of every possible kind of blank ballot, which we discovered is very burdensome. Therefore, we cannot rely on knowing all ballot styles a priori. Instead, our approach is to decode the barcodes present on optical scan ballots, supplemented by additional information when necessary.

4.2.1. Vendor-Specific Barcodes In deployed optical scan systems, the scanners rely on the presence of specialized, vendor-specific barcodes to determine ballot metadata such as the ballot style. These barcodes can range from simple binary bit-strings to more complex encodings (Figure 7). In most cases, the barcodes present on each ballot fully determine the ballot style, so we can group ballots by decoding the barcode and then put ballots with the same barcode into the same group. (Section 4.2.3 describes how to handle exceptional cases where the barcode alone is not enough.)

OpenCount does not require knowledge of the semantic meaning of the barcodes; we merely need to be able to decode them to bit-strings. That said, we have reverse-engineered the semantic meaning of most of the barcodes[2]. For instance, several vendors incorporate a checksum in the barcode; when possible, we verify that the checksum is valid during the decoding process.

OpenCount currently has built-in support for paper ballots from four major election vendors: Hart InterCivic, Premier Election Solutions (formerly Diebold Election Systems), Sequoia Voting Systems, and Election System & Software (ES&S). It would not be difficult to add more vendors to OpenCount in the future.

4.2.2. Ballot Attributes To facilitate comparison of OpenCount's results to the official results, OpenCount includes support to identify the precinct number and tally group of each ballot. This information is not always encoded in the barcode, but it is always printed in human-readable form on the ballot. Therefore, we provide support to enable a human operator to identify these features on the ballot; OpenCount then uses this information to automatically decode these ballot attributes. The previous system [Wang et al. 2012] did include a mechanism for this purpose. However, there were two major shortcomings present in the previous design that we address here. First, we extend the automated annotation process to make it more robust to variations in visual appearance, such as varying background colors and patterns. Second, we introduce a novel approach to decoding precinct number stamps on ballots. Decoding ballot attributes is a two-step process: attribute definition, and then (if necessary) exemplar detection.

[2]For details, see: https://code.google.com/p/opencount/wiki/Barcodes

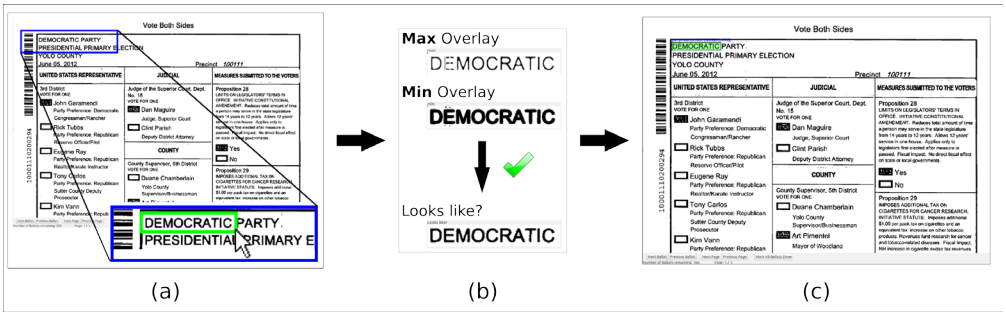


Fig. 3. Defining the "party" ballot attribute. (a) The operator draws a bounding box around the region where the party affiliation is printed on the ballot. (b) OpenCount finds other ballots that contain the same image patch and shows the min and max overlays of all matches to the operator. The operator uses the overlays to verify the matches. (c) The operator continues this labeling process until all ballots have been labeled.

Attribute Definition. Within OpenCount, the operator declares a number of "ballot attributes" for each desired property (precinct number, tally group, etc.), and then defines the possible attribute values. Additionally, the operator associates each attribute value with an image patch where this value is present in human-readable form on the ballot.

Workflow. The operator workflow is as follows. At any given moment, there is a set of ballots that have not been fully labeled. First, an unlabeled ballot is displayed. The operator draws a bounding box around the location of the desired attribute (Figure 3(a)), and specifies an attribute type and value (e.g., "party: independent"). OpenCount then searches the remaining unlabeled ballots to find ballots containing a matching image patch. Matches are identified using normalized cross-correlation (NCC) and a fixed threshold.

OpenCount then displays an overlay for all matches, so the operator can confirm that all detected matches are indeed a true match (Figure 3(b)). This process is repeated until all ballots have a label for every defined attribute type. Precinct-number attributes are handled separately (see Section 4.2.4 for details).

When defining an attribute, the operator indicates whether the value of the attribute is known to be consistent within each group. If the attribute is known to be *group-consistent*, then OpenCount will select one ballot from each group, perform the above process, and apply the discovered labels to all ballots automatically; no further processing is needed. Otherwise, to allow for the possibility that an attribute value may vary within a single group, OpenCount randomly selects 20% of the ballots from each group, and applies the above process to each of them; then, OpenCount applies exemplar detection and automatic annotation, detailed next.

Robust Exemplar Detection. At this point, only one image patch is defined for each attribute value. Depending on the election, a single image patch may not be sufficient to annotate the ballots. In Figure 4, the difference in background color (white vs. grey) overwhelms the difference in printed number ("005" vs. "006"). The image patch in Figure 4(c) is misclassified as "006" because it shares the same background as the "006" representative.

More generally, we would like visual distractions such as the background style to not affect the matching process. The previous system [Wang et al. 2012] did not have a sufficient mechanism in place to handle such variations, leading to failure cases such as Figure 4. Thus, we introduce the following extension. Rather than use only one image patch for each attribute value, OpenCount instead selects multiple representative image patches for each attribute value. These representative image patches are chosen such that they capture the possible diversity present in the dataset. The algorithm is a fixed-point iteration that takes advantage of the fact that, at this point, the software already has a set of image patches labeled with their correct attribute values (done by the operator during attribute definition).

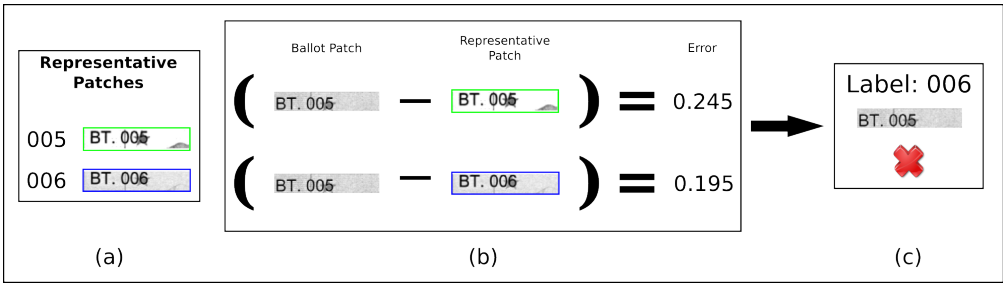


Fig. 4. (a) The representative image patches for this attribute. (b) Matching an image patch to one of the representative patches. Error is computed with the SSD metric. (c) Due to background variation, the patch was misclassified as "006".

For each attribute value, we initially choose a single image patch as the first representative patch. The algorithm then classifies all image patches with the current set of representative image patches. If an image patch is misclassified (i.e., a "democratic" patch was mistaken for a "republican" patch), then it is likely that this image patch contains some significant variation. Thus, we add this image patch to the set of representatives for that attribute value, and repeat the process.

To classify a new patch, we compare it to each representative and choose the attribute value whose sum-of-squared-difference (SSD) between the patch and the representative patch is minimized. To allow for translation and rotation invariance, we first align the image patch to the representative patch prior to computing each error score. The algorithm terminates when all image patches are classified correctly by the current set of representative image patches. OpenCount chooses the initial representative values by selecting the image patch with the highest average pixel intensity.

Automatic Attribute Annotation. Once the operator has defined the desired attributes and representative patches, OpenCount determines the attribute values of each voted ballot. For brevity, we refer the reader to Section 4.5 of [Wang et al. 2012], as the core approach remains unchanged.

Once the automated annotation is complete, OpenCount asks the user to verify the resulting labels using overlays (see Figure 2(b–c)). The operator is able to correct any errors by manually relabeling any misclassified image patches.

4.2.3. Additional Grouping In some cases, the ballot style may not be completely encoded within the barcodes. For instance, the choice of language on Sequoia-style ballots affects the location of voting targets. However, the language is not encoded in the barcodes—thus, in this case grouping by the barcodes is insufficient. In this scenario, one may use attributes as an additional grouping criterion after the barcode-based grouping has been performed. For instance, in the Sequoia example the operator should define a ballot attribute for the language[3].

4.2.4. Precinct Number Recognition We added support for OpenCount to determine the precinct of each ballot, based upon the precinct number printed on the ballot. The previous version of OpenCount used the attribute decoding process described above for this purpose. However, while processing ballots from the June 2012 primary, we found that this approach was inadequate: the number of different precincts is large enough that this imposes an overwhelming burden on the operator. Additionally, without precinct number recognition we would require one attribute per precinct; since grouping runs linear in the number of different groups, this becomes prohibitively slow.

Therefore, we designed an alternative method for decoding precinct numbers. After experimenting with off-the-shelf OCR software, we found it was not accurate enough for our purposes. Instead,

[3]We discovered that Sequoia election machines interpret voting targets via a vertical scan line through the middle of the complete-the-arrow targets, allowing invariance to vertical translation. This is in contrast to other optical scan systems, where the location of voting targets within a ballot style is fixed—hence why this additional grouping step is required. For instance, in the Alameda election, the Chinese-language ballots sometimes have voting targets at a different location than the corresponding Spanish-language ballot, because the Chinese-language candidate names used more vertical space.

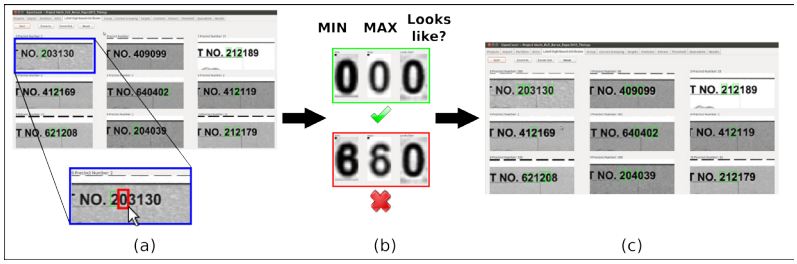


Fig. 5. The user interface for selecting digit templates. (a) The user selects one example of a "0" digit. (b) The overlay of all matches is presented to the user. After splitting once, we obtain two groups, shown stacked vertically. The top overlay shows that all digits in that group are clearly a "0". In contrast, the bottom overlay reveals that the group contains some mismatches, e.g., "6" or "8". (c) The accepted matches are labeled with their value ("0").

we designed a novel solution to this problem, which works by recognizing each individual digit and taking advantage of the fact that all digits within the same election are printed in the same font. The algorithm works in two steps: digit exemplar selection, and digit parsing.

Digit Exemplar Selection. In this step, we obtain an example of each digit (0–9) directly from the ballot images themselves. Once OpenCount knows what each digit looks like for this particular election, we can decode a precinct number by visually comparing the digits in the precinct region to its set of example digits.

To obtain examples of each digit, OpenCount displays all the precinct numbers in a grid[4], one patch from each group. See Figure 5(a). The operator selects an unlabeled digit and draws a bounding box around that digit, and enters the correct digit value (e.g., "0"). OpenCount then uses template matching across all precinct patches to find all matches for the selected digit. Any matches whose NCC (normalized cross-correlation) score exceeds a fixed threshold is retained as a potential candidate, and the operator confirms all candidate matches using an overlay verification step (Figure 5(b)). After the verification is complete, the operator-accepted matches can be labeled with their value (as shown in Figure 5(c)). The task is finished once all digits in all precinct patches have been labeled. To account for possible background pattern variations, OpenCount employs the same robust exemplar patch detection technique from Section 4.2.2 for each digit.

Digit Parsing. Once OpenCount has example image patches for each digit, it can begin to interpret the precinct patches (i.e., given an image containing a precinct number, recover the decimal string). Intuitively, each unknown digit in the precinct patch could be decoded by comparing it against all exemplars to find its closest match. However, we would like our approach to not require an initial segmentation of the unknown precinct patch into its individual digits. Ballots may not be scanned at very high resolution, and coupled with scanner noise, segmentation algorithms are likely to output incorrect results. This is exacerbated by the fact that precinct stamps are often printed in a small font. Instead, we implemented a dynamic programming algorithm to simultaneously identify the location of all digits in the unknown patch and find each digit's best match among the exemplars.

The core of our approach is a scoring algorithm that compares a representative digit against a particular precinct number through a combination of the individual digit match confidence scores and spatial relationships between adjacent pairs of digits. More formally, let $L = (l_1, ..., l_n)$ represent a configuration of an n-digit precinct number on a ballot where l_i is the (x, y) coordinate of the i^{th} digit. Let $m(l_i^{(c)})$ be the match cost of a digit, c at location l_i. We compute this as the NCC score between the template image of that digit and the ballot at location l_i. Now let $M(l_i) = \max_c l_i^{(c)}$ store the best match score of all digits at a given location and $Q(l_i) = \text{argmax}_c l_i^{(c)}$ store the identities of the digits at those locations. Finally, let $d(l_i, l_j)$ represent the pairwise penalty of two adjacent digits

[4]The operator identifies the location where the precinct number is printed on the ballot during attribute definition.

placed at l_i and l_j. In our case, the ideal location for l_j given l_i is to be one character's width away on the x-axis while being on the same y-axis. We set $d(l_i, l_j)$ to be a quadratic cost for deviations from the ideal location. Our algorithm solves for the optimal configuration L^*, using the function:

$$L^* = \underset{L}{\operatorname{argmin}} \left(\sum_{i=1}^{n} M_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right) \quad (1)$$

The cost of any configuration of digits L is the sum of the cost of their individual match scores and sum of pairwise spatial costs. Our formulation draws from the Pictorial Structures work of [Fischler and Elschlager 1973; Felzenszwalb and Huttenlocher 2005] and a solution can be found efficiently using dynamic programming. This algorithm allows us to efficiently and accurately decode precinct numbers on all ballots, using the exemplars selected by the operator in the previous stage. Once the precinct decoding is complete, OpenCount asks the user to verify the labeling results via overlay verification. The user can correct any misclassified digits here, if necessary.

4.3. Layout Annotation

After OpenCount has grouped the ballots, the operator must then annotate the layout of one ballot from each group. In the previous version of OpenCount [Wang et al. 2012], annotation of contests and candidates in very large elections required a lot of operator effort: directly proportional to the number of contests per ballot, times the number of different ballot styles. We designed several new methods to greatly reduce the workload on the operator in complex elections.

Detecting Voting Targets. OpenCount assists the operator in identifying the location of all voting targets in every group. When blank ballots are available, this is easy: the operator can identify one example of an empty voting target, and the system can find all matches on all blank ballots. Since the blank ballots do not have any voter marks, all matches should be clean. However, when we do not have blank ballots, more sophisticated methods are necessary.

Our improved procedure works as follows. For each group, one ballot is arbitrarily selected and displayed to the operator. The operator then draws a bounding box around one of the voting targets. Once a voting target is selected, OpenCount automatically tries to detect as many matching voting targets as possible using template matching on all remaining unannotated groups. Any match whose NCC score is above a fixed threshold is accepted as a candidate voting target. To prevent multiple overlapping matches, once a match is processed, all other NCC scores within a fixed region around the match are suppressed. We apply smoothing with a Gaussian kernel to the template and ballot images prior to template matching to improve detection rates.

OpenCount applies an additional technique to further reduce operator effort. We select N representative ballots from each group, template match against all representatives, and union the results together[5]. This is intended to solve the problem that voter marks often interfere with the template matching search. For instance, if the operator draws a bounding box around an empty voting target, filled targets will not be identified as matches during the template matching. However, the same voting target will be present on the other $N-1$ representatives in that group, and it is likely that it will be empty in at least one of these cases; thus the union trick allows us to detect the location of that voting target. We set $N = 5$, which offers a good balance between convenience and performance.

Detecting Contest Bounding Boxes. We implement a method to identify the "bounding boxes" of contests on a ballot—a rectangular region that surrounds all of the voting targets in a contest. These bounding boxes help recognize which voting targets belong to the same contest, which is used for tabulation purposes. The contest bounding boxes also enable us to identify redundant contests by performing pair-wise comparisons between the set of detected contests; this enables a major reduction in the amount of data entry required.

[5]Note that this requires all N representatives within a single group to be aligned to each other.

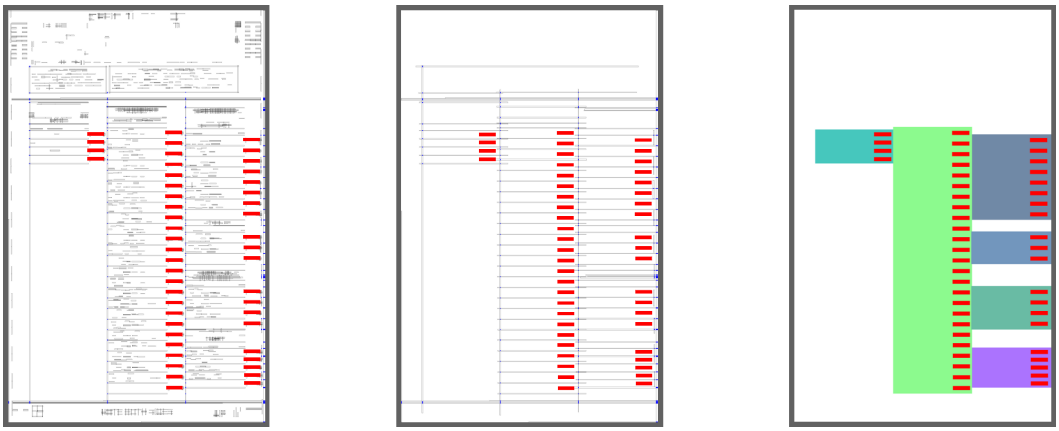


Fig. 6. The three stages of bounding box inference. First, all the lines on the ballot are identified. Second, lines are extended and extraneous lines are removed. Third, boxes are formed.

The previous version of OpenCount implemented a more naive approach. Instead of attempting to identify the actual contest bounding box, voting targets were instead grouped together by distance. However, as we experimented with large elections with many ballot styles, we discovered that this naive heuristic was both incorrect and could not find a box surrounding all candidate names.

We developed a new algorithm which is much more accurate, and also provides the full bounding boxes. We find it only makes errors the ballot has significant stray voter marks, and in those cases we provide a mechanism to select an alternate ballot in the same group without such marks. Our algorithm takes advantage of the fact that, on all ballot styles we have encountered, each contest is at least partially surrounded by horizontal and/or vertical lines that demarcate it from other contests.

First, OpenCount identifies all vertical and horizontal lines on the ballot. This is done by scanning over the ballot for pixels darker than some threshold, and attempting to identify if this is part of a long vertical or horizontal line segment. Only segments of a large enough size are retained. Line segments are identified by continuous segments of pixels darker than some threshold.

We then reduce all the scattered line segments through several intermediate steps. First, line segments are joined together if they extend in the same direction and are overlapping. Second, line segments are removed if they do not intersect with any perpendicular segments. Third, segments are removed if no voting target is located in the area it could possibly be extended to.

Next, we process the ballot to determine if there is some constant C where, if all lines were extended by a factor of C, significantly more intersections are found. This allows for cases where vertical and/or horizontal lines are present on the ballot, but do not fully intersect each other, as happens on several styles.

Finally, OpenCount searches over all candidate rectangles that can be formed by these lines (from smallest rectangle to largest), looking for rectangles that contain voting targets. When a rectangle does contain at least one voting target, all of the targets in its interior are removed and the process is repeated. For some vendors, it is necessary to combine multiple bounding boxes together to form the final set of bounding boxes; in these ballot styles, contests are finally merged. For example, Figure 6 shows that each contest is contained within its own bounding box in the second image, but in the third image many are merged together to form the correct contest bounding boxes.

Once OpenCount infers all bounding boxes, it presents them to the operator, who can manually correct any errors. OpenCount can optionally detect bounding boxes on the other four representative ballots and warn the operator if the bounding boxes are not identical on all five.

Contest Data Entry. The final step in ballot annotation is to label all detected contests with the contest title and the candidate names. This is crucial information to allow OpenCount to output human-readable election results. In principle, this is not a difficult task—one can simply ask the

operator to enter the relevant text on a representative ballot from each group. In fact, this is what was done in the previous version of OpenCount.

However, in many elections there are several hundred different groups, and manually annotating every contest on every group would take many hours of manual effort. To address this challenge, we developed new methods to detect contests that are duplicates of each other. While there may be hundreds of groups, each having several contests, there are typically only a few distinct contests in the entire election. If duplicates can be detected, the operator only needs to label one instance of each contest. In practice, we find that this speeds up the contest data entry process from between a factor of ten to a hundred for large elections, not counting computation time.

Unfortunately, detecting duplicate contests is a difficult task. We experimented with many strategies for recognizing when two contests are the same by comparing them as images, but all failed. There are three key challenges. First, contests that are *semantically* equivalent may not necessarily be *visually* identical. Examples of such visual discrepancies include varying image dimensions, text lines wrapped at different locations, and inconsistent spacing between the words in candidate names. Second, the order of candidates on a given contest may be different on two different instances, due to ballot rotation. And third, some contests do not fit on a single column of a ballot, but continue onto the next column.

We resolve the first challenge by using an off-the-shelf Optical Character Recognition (OCR) software as a preprocessor to extract the text from the contest images[6]. We extract the title and the candidate names independently. To compare the similarity of two contests, we then use the Levenshtein (edit) distance [Levenshtein 1966] between the extracted text of the two contests. By comparing the OCR outputs between two contests, rather than the contest images themselves, we remain invariant to the visual confounding factors mentioned previously. Crucially, this approach allows us to tolerate inaccuracies in the OCR output. For example, if the OCR were only accurate on 50% of the characters, we would expect the Levenshtein distance to be 25% smaller for two identical contests than for two different contests (of similar length). In practice, we found that Tesseract makes many errors, but it is still over 90% accurate on most ballot styles and languages.

Solving the second challenge requires more work. At a high level, whenever we test a pair of contests for equivalence, we search over all possible rotations of the candidate names and find the best match. For all elections we are aware of, candidate names are not permuted arbitrarily. Instead, they are simply rotated some number of positions, with one modification: the write-in candidate(s) always appear at the end. Therefore, our algorithm takes advantage of this fact. Once we have extracted both the contest title and the candidate names for each voting target, we attempt to determine (a) the number of write-in candidates and (b) the rotation amount. In particular, we search over all possible number of write-ins, and all possible rotations. We then treat two contests as duplicates if they have both the same number of write-in contests, and if they have a small Levenshtein distance between the names of corresponding candidates.

We solve the third challenge by letting the operator mark contests that span multiple columns while labeling the voted ballots: if the operator finds such a contest while entering text on the ballots, she clicks a button marking it as such, and continues on.[7] OpenCount then performs another search over all consecutive pair of contests to attempt to identify other contests that also are split across two bounding boxes by comparing their similarity against the pair the operator marked.

We implement several optimizations to improve performance. First, contests are only compared if they have the same language and same number of targets; and second, we sort candidates by length and perform an initial linear scan where we compare consecutive contests, to detect some duplicates quickly. These two optimizations typically reduce computation time by a factor of a hundred on large elections.

---

[6] We use Tesseract [Smith 2007] for this task. OpenCount does support multiple languages for this task. As Tesseract is able to interpret a range of languages, if the operator defines a "language" ballot attribute (Section 4.2.2), then OpenCount will interpret each ballot with the correct language.

[7] The user interface displays the ballot image during the data entry process, making it easy for the user to notice such contests.

Table I. General information on elections processed with OpenCount.

County	Number of Ballots	Number of Styles	Number of Sides	Image Resolution
Alameda	1,374	8	1	1460x2100
Merced	7,120	1	1	1272x2100
San Luis Obispo	10,689	27	1	1700x2200
Stanislaus	3,151	1	1	1700x2800
Ventura	17,301	1	1	1403x3000
Madera	3,757	1	1	652x1480
Marin	29,121	398	1	1280x2104
Napa	6,809	11	2	1968x3530
Santa Cruz	34,004	136	2	2400x3840
Yolo	35,532	623	1	1744x2878
Leon	124,200	216	2	1400x2328
Orange	294,402	1,839	1-3	1715x2847

Finally, the we present operator with an overlay of the contests. We align candidates independently, and merge them together to form a single overlay. The user then verifies that the resulting overlays indeed correspond to the same contest. See Figure 8 (Right) for an example overlay.

4.4. Ballot Interpretation and Result Generation

Once the operator has annotated a representative of each group, OpenCount uses this information to interpret the remaining ballots. We did not need to make major improvements or changes to this stage from the previous version of OpenCount [Wang et al. 2012], though we did take the opportunity to improve performance in several places. For convenience, we summarize this stage:

— **Extract voting targets.** In each group, OpenCount takes a representative and globally aligns all other ballot to it. OpenCount then locally aligns each voting target to the corresponding voting target on the representative, and extracts it. The current version runs on average three times faster.

— **Classify voting targets.** OpenCount then presents the operator with a grid of targets sorted by average intensity, who then identifies which targets are filled. We added the capability for the operator to set filters to only show overvotes and undervotes or only targets from a given contest, or to run a second classifier over the voting targets and display only disagreements.

— **Handle quarantined ballots.** Finally, the operator handles all ballots that were quarantined. The operator manually indicates the votes for each contest, as well as the ballot attribute values. Partial information about each ballot is automatically populated when possible. For instance, if a ballot was quarantined after contests were inferred, this information is automatically filled. The previous version of OpenCount did not have this feature, making it much more labor-intensive.

— **Generate results.** OpenCount generates CVRs and cumulative vote tallies. Each CVR specifies the candidates that the voter voted for. Overvotes and undervotes are identified as such. The cumulative vote tallies show the vote totals for each candidate in each contest. These totals are further broken down by precinct and tally group if those attributes are available, which assists in comparing with the official results at a finer granularity.

5. Evaluation

5.1. Methodology

We ran OpenCount on the 12 elections listed in Table I, and also ran the previous version of OpenCount [Wang et al. 2012] on a subset of the elections. We timed how long each took, on a six-core machine Intel i7-3930K processor with 16GB of RAM. Table II records the results.

Different operators ran different elections, so comparisons on timing statistics from election to election may not be meaningful. However, whenever we ran both versions of OpenCount on an election, the same operator ran both versions. On elections where both versions of OpenCount were run, the results between election runs match on $\geq 99.9\%$ of ballots.[8] The only differences were

[8]Exception: The previous version of OpenCount produced inaccurate results in one election, Napa, as described below.

Table II. Election timing data with OpenCount. Steps that require operator interaction are labeled with (H), and steps that only perform computation are labeled with (C). The second column identifies the version of OpenCount used; the 2013 version contains the improvements described in this paper, whereas the 2012 version does not. Times are rounded to the nearest minute, except for the "Total" column. Entries tagged with a * are extrapolated (see text for details).

County	Version	Decode (C)	Ballot Attrs. (H)	Group (C)	Group Check (H)	Layout Annotate (H)	Label Contests (H)	Target Extract (C)	Target Check (C)	Avg. per ballot	Total
Stanislaus	2013	0m	-	-	-	1m	1m	5m	1m	0.14s	7m 18s
	2012	-	-	-	-	1m	2m	11m	4m	0.33s	17m 30s
Merced	2013	0m	-	-	-	0m	1m	8m	3m	0.11s	12m 31s
	2012	-	-	-	-	1m	1m	22m	2m	0.22s	25m 32s
Ventura	2013	0m	-	-	-	2m	1m	17m	3m	0.08s	23m 6s
	2012	-	-	-	-	1m	1m	39m	3m	0.15s	43m 8s
Alameda	2013	1m	6m	2m	1m	3m	3m	1m	1m	0.75s	17m 6s
	2012	-	3m	4m	0m	2m	3m	6m	3m	0.96s	22m 1s
SLO	2013	10m	3m	0m	0m	4m	1m	11m	1m	0.17s	30m 35s
	2012	-	3m	32m	9m	2m	2m	37m	1m	0.48s	1h 25m
Madera	2013	0m	-	-	-	1m	0m	3m	2m	0.11s	6m 38s
	2012	-	-	-	-	1m	1m	5m	2m	0.14s	8m 30s
Napa	2013	20m	11m	26m	1m	3m	7m	33m	22m	1.02s	1h 56 9s
	2012	-	10m	1h 2m	42m	4m	13m	2h 30m	42m	2.85s	5h 23m
Marin	2013	52m	6m	0m	0m	2h 47m	2h 20m	5h 17m	32m	1.47s	11h 53m
Santa Cruz	2013	3h 38m	-	-	-	40m	1h 2m	9h 46m	3h 45m	2.00s	18h 50m
Yolo	2013	25m	7m	0m	0m	52m	26m	1h 40m	6m	0.37s	3h 36m
	2012	-	16m	10h 5m	40h*	25m	52m*	6h 42m	9m	5.92s	58h 27m
Leon	2013	3h 40m	4m	0m	0m	29m	44m	8h 27m	36m	0.41s	14h 2s
Orange	2013	4h 9m	2h 19m	0m	0m	8h 48m	8h 13m*	2d 9h 27m	13h 5m*	1.149s	3d 22h 39s

Table III. Quarantined ballots requiring manual processing. For entries marked with a (*), we processed 10% and extrapolated the required time.

County	Version	Number of Quarantined Ballots	Fraction of Election Size	Total Time
Stanislaus	2013	1	0.03%	29s
Merced	2013	1	0.01%	48s
Ventura	2013	1	0.01%	8s
Alameda	2013	191	13.90%	1h 4m 22s*
SLO	2013	14	0.13%	5m 46s
Madera	2013	0	0.00%	0s
Napa	2013	16	0.23%	7m 23s
Marin	2013	129	0.44%	1h 46m 10s*
Santa Cruz	2013	163	0.48%	2h 16m 30s*
Yolo	2013	60	0.17%	1h 35m 0s*
Leon	2013	602	0.48%	8h 11m 34s*
Orange	2013	192	0.00065%	4h 14m*

almost always the result of the operator interpreting ambiguous marks differently (i.e., a mark that may either be caused by an erasure, or may be a lightly-filled in mark). OpenCount developers processed each elections.

We also invited Tom Stanionis, an election official from Yolo county, to evaluate the OpenCount software. Mr. Stanionis was able to successfully process an election subset without any significant intervention from the developer conducting the evaluation. We gathered valuable feedback and constructive criticism about the software from a usability standpoint.

5.2. 2011 Elections

We analyzed five special elections from 2011 (the same ones evaluated in [Wang et al. 2012]): Stanislaus, Merced, Ventura, Alameda, and San Luis Obispo (SLO). All of these have only a few contests. Three have only a single ballot style. Alameda has 8 ballot styles, but is the smallest

election we analyzed, with only 1,374 ballots. SLO has 27 ballot styles, but all styles contain the same two contests, differing only in precinct and tally group.

The results are given in Table I. Because these elections are so small and contain so few contests and so few ballot styles, the 2011 elections do not stress-test the ability of OpenCount to handle complex elections with many ballot styles. For instance, Alameda shows the least benefit from our improvements: with so few ballots, very few of the additions have an impact on total running time. Instead, the total time spent is dominated by initial setup. As it happens, many ballots in the Alameda election were quarantined because the scanned ballot was missing the top portion of the ballot, which included the barcode, see Table III for full quarantine statistics.

5.3. 2012 Elections

We also analyzed ballots from the June 2012 primary in five counties. These elections were more complex. As a primary election, they contained many more ballot styles. Madera was by far the smallest, with only one ballot style, and we encountered no difficulties.

Napa. Napa had 28 ballot styles and we were able to process all 6,809 ballots using our improved version of OpenCount without incident. The previous version of OpenCount made serious errors during target extraction and extracted the wrong regions on nearly 50% of ballots. On ballots where targets were extracted correctly, the two versions matched with over 99.9% accuracy. On the remaining 50%, we randomly sampled 20 ballots and, in every case, the current version of the software was completely correct, and the previous version was wrong. Our improvements led to a modest ($2.8\times$) speedup; however, with so few ballot styles, many of our improvements do not take effect.

Marin. The Marin election had 398 ballot styles, so it was a good stress test of our improvements to reduce operator effort. The improvements were crucial to our ability to successfully analyze this election. However, target extraction failed for 115 ballots, due to poor image alignment, and these ballots had to be flagged for manual processing. This dominated the time it took for us to process the Marin ballots.

Santa Cruz. Santa Cruz had fewer ballot styles, which made processing go quickly. However, verifying the interpretation of marks was made more challenging because Santa Cruz uses a Sequoia complete-the-arrow style ballot, and OpenCount's visualization tool is not as effective for these ballots. This is a good opportunity for further improvements. Also, target extraction failed for 84 ballots, which had to be flagged for manually processing. In each case, the initial image alignment between the ballot and the reference image was incorrect, causing a wildly inaccurate alignment to occur. See Section 6.2 for additional discussion about these alignment issues.

Yolo. Yolo was our second most complex and third-largest election. The time to label contests includes the 13 minutes of computation OpenCount spent detecting equivalent contests, which massively reduced the amount of data-entry required, from 4,603 contests to fewer than 50.

The absence of blank ballots made it significantly more challenging to identify all voting targets: this task took two times longer than on the previous version, when blank ballots were available. This is due both to the increase in the number of groups—there are 117 different blank ballots, but this increases to 623 groups when we do not have blank ballots—and to the increase in operator workload due to the presence of voter marks—the operator must repeatedly identify empty voting targets, to ensure all are found. As an experiment, we ran the ballot annotation process of the current version of OpenCount on the 117 blank ballots, and found that it ran six times faster.

We did not complete grouping verification in its entirety on the previous version of OpenCount. After finishing 5% of grouping verification (in two hours), we extrapolated that grouping verification would take about 40 hours. We used data from the current version of OpenCount to fill in the remaining data.

Leon. With 124,200 ballots, the November 2008 general election in Leon County, Florida was our second largest election processed. We only ran Leon on the most recent version of OpenCount;

processing it with the previous version would have taken an unreasonable amount of time. The contest labeling step of Leon took much longer than that of Yolo, because Leon had several contests of the form "District court of appeals. Shall Justice __ be retained in office" for each of five different justices. This caused the contest equivalence class computation to erroneously infer that all of these contests were identical; we had to reject and manually label these contests, which took extra time.

Orange. Finally, we processed the Orange county June 2012 Presidential Primary Election, which consisted of 294,402 voted ballots with a variable number of sides per ballot, ranging from one to three. This election has as many ballots as all other elections combined. Similar to Leon County, we only processed the ballots with the most recent version of OpenCount.

This election dataset posed several significant challenges. First, the scan quality of these images is noticeably poorer than that of any of the other elections. Rather than rescanning the paper ballots with commercial scanners to obtain high-quality image scans, these images were output directly from the Hart voting systems. This is both good and bad; while no additional scanning effort was necessary, we had to modify OpenCount to handle much more challenging images due to scanning conditions outside of our control. Additionally, the images are binary images, thresholded by the Hart system, which further degraded the image quality through the loss of information. Second, the complexity of the election is by far the greatest: there are 1,839 distinct ballot styles [9].

While completing the "Target Check" stage, we discovered that the target sorting metric currently used is inadequate for elections of this size, and requires the operator to inspect an overwhelming number of voting targets. The average-intensity heuristic used for sorting targets breaks down when voter marks are small in comparison to the voting target itself. Image noise then dominates the sorting metric, resulting in empty voting targets being mixed with filled-in voting targets, requiring many hours of manual corrections. It is worth noting that these sparsely filled-in voting targets are typically cases where the voter drew in an "X" or a line to indicate his or her vote. Improving this step is a focus for future work, and may draw upon ideas from the style-based mark classifier of [Xiu et al. 2009]. Thus, we completed a small portion of the "Target Check" stage and extrapolated the total time required. The timing data for the "Label Contests" stage was similarly extrapolated.

Interestingly, Orange County had the lowest percentage of quarantined ballots (Table III).

6. Discussion

One question we must consider is how well OpenCount will generalize. For this system to be useful, it must minimize the restrictive assumptions it makes on specific ballot details. We consider the major assumptions that OpenCount makes in this section.

6.1. Ballot Assumptions

We assume each ballot image contains a barcode-like structure that can be used to group ballots by ballot style. We do not assume that the barcode uniquely determines ballot style.

Also, contests must be roughly box-shaped and be at least partially surrounded by lines on at least three sides. Furthermore, each contest must contain the contest title followed by a list of candidate names, with one voting target corresponding to each candidate (including write-in candidates[10]). OpenCount does support ballot rotation: the ordering of the candidate names displayed within a contest is allowed to vary from ballot to ballot.[11]

Voting targets may be any consistent structure that the voter is asked to mark. This includes the common "fill-in-the-bubble" and "fill-in-the-arrow" styles.

[9] We used a spreadsheet file output by the Hart voting system that mapped precinct number to "ballot type" to do the grouping - otherwise there would be 22,025 ballot styles. This was a pragmatic decision on our part, as we would prefer not to rely on files generated by the voting system.

[10] OpenCount does not attempt to interpret handwriting for write-in votes—it only records the presence of write-in votes.

[11] Our current implementation assumes that only rotations of the candidates are possible, not arbitrary permutations. So far, we have not observed an election that violates this assumption. However, if necessary, it would be trivial to extend OpenCount to support arbitrary permutations using maximum-weighted matching.

Consequences and Relaxation. OpenCount is currently able to support optical scan ballots from Diebold (Premier), ES&S, Hart, and Sequoia systems, which accounts for the overwhelming majority of opscan ballots in the US. Of the requirements listed above, the most stringent is the requirement that the ballot contain a barcode that can be used for grouping. Fortunately, all ballots that we have encountered to date meet this requirement. To relax this requirement, one could imagine applying algorithms that group ballots on visual appearance. However, we have not investigated the challenges associated with doing so.

Finally, while OpenCount currently does not support voting schemes such as straight-party voting or fusion voting, it would be simple to modify the logic to accommodate such voting schemes.

6.2. Scanner Assumptions

In several stages of the pipeline, ballots are aligned to a representative ballot. The image alignment method used in OpenCount assumes a rigid transformation model, which allows translation and rotation variation. The simpler rigid model performed better than more complex models such as the affine model, as the rigid model has fewer parameters to solve for, typically leading to higher-quality solutions. Additionally, in practice, almost all images can be aligned with only translations and rotations, and do not require other affine warps (scaling, shearing, etc.).

Our alignment scheme is able to tolerate a moderate amount of non-rigid warping, due to the fact that it performs two alignment steps: a coarse "global" alignment of the entire image, followed by a finer "local" alignment applied to small regions of the image. Both alignments still use rigid transformations. If an image contains non-uniform warping (e.g., the bottom-half of the image is stretched/skewed more than the top), then no precise global alignment will exist between this warped image and its reference image. However, a good local alignment can often still be recovered, allowing our scheme to tolerate a moderate amount of non-uniform warping. If ballots contain too much non-rigid warping, then mis-alignment errors will cause issues during target extraction. Such issues affected 0.2% of Santa Cruz ballots, as mentioned in Section 5.3.

6.3. Scalability

The improvements made to the OpenCount system in this paper have enabled the processing of significantly larger and more complex elections. The previous system was not able to reasonably process the Yolo county election, let alone the Leon or Orange datasets. The current proposed system, on the other hand, is able to process all three without trouble.

However, it remains to be seen how the system will scale to even larger elections. For instance, about 3.2 million ballots were cast in November 2012 in Los Angeles county [LA-Registrar 2012]. This is roughly $11\times$ larger than the Orange county election. Future work is needed to investigate whether OpenCount is able to efficiently handle elections of that magnitude.

7. Conclusion

We have introduced several significant improvements to the original OpenCount system that enables OpenCount to scale to significantly larger elections, and process them an order of magnitude faster. The new system does not require the collection of blank ballots, which reduces the barrier on election officials; we also greatly reduce the amount of data entry required, which speeds the re-tabulation process. These improvements enabled OpenCount to successfully support audit pilots in 11 California counties and advance the state of the art in ballot image analysis.

ACKNOWLEDGMENTS

We thank the California Secretary of State's office and election officials at Stanislaus, Merced, Ventura, Alameda, San Luis Obispo, Madera, Marin, Napa, Santa Cruz, Yolo, Orange, and Leon counties for sharing scanned ballots, and Clear Ballot for their assistance with obtaining ballot scans. This research would not have been possible without their generous assistance. We thank Philip Stark for helpful guidance and Tom Stanionis for providing feedback on OpenCount.

REFERENCES

H.S. Baird, S.E. Jones, and S.J. Fortune. 1990. Image segmentation by shape-directed covers. In *Proceedings of ICPR*, Vol. i. 820–825 vol.1.

Josh Benaloh, Douglas Jones, Eric L. Lazarus, Mark Lindeman, and Philip B. Stark. 2011. SOBA: Secrecy-preserving Observable Ballot-level Audits. In *Proceedings of EVT/WOTE*.

Thomas M. Breuel. 2002. Two Geometric Algorithms for Layout Analysis. In *Proceedings of DAS*. 188–199.

Joseph A. Calandrino, J. Alex Halderman, and Edward W. Felten. 2007. Machine-Assisted Election Auditing. In *Proceedings of EVT*.

Arel Cordero, Theron Ji, Alan Tsai, Keaton Mowery, and David Wagner. 2010. Efficient user-guided ballot image verification. In *Proceedings of EVT/WOTE*.

Cyrus Farivar. 2012. Saving throw: securing democracy with stats, spreadsheets, and 10-sided dice. http://arstechnica.com/tech-policy/2012/07/saving-american-elections-with-10-sided-dice-one-stats-profs-quest/. (24 July 2012).

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 2005. Pictorial Structures for Object Recognition. *International Journal of Computer Vision* 61, 1 (Jan. 2005), 55–79.

Martin A. Fischler and R.A. Elschlager. 1973. The Representation and Matching of Pictorial Structures. *Computers, IEEE Transactions on* C-22, 1 (1973), 67–92.

Harri Hursti. 2005. Votoscope Software. (2 October 2005). http://vote.nist.gov/comment_harri_hursti.pdf.

LA-Registrar. 2012. Los Angeles County: Statement of Votes Cast Election Results (Nov. 2012 General Election). http://www.lavote.net/VOTER/PDFS/STATEMENT_VOTES_CAST/11062012_SVC.pdf. (2012).

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10 (1966).

Mark Lindeman, Ronald L. Rivest, and Philip B. Stark. 2013. Retabulations, Machine-Assisted Audits, and Election Verification. (2013). http://www.stat.berkeley.edu/~stark/Preprints/retabulation13.htm.

Mark Lindeman and Philip B. Stark. 2012. A Gentle Introduction to Risk-limiting Audits. *IEEE Security and Privacy* 10, 5 (2012). Special Issue on Electronic Voting.

Daniel Lopresti, George Nagy, and Elisa Barney Smith. 2008. A Document Analysis System for Supporting Electronic Voting Research. In *Proceedings of DAS*. 167–174.

J. B. MacQueen. 1967. Some Methods for Classification and Analysis of MultiVariate Observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. 281–297.

Song Mao, Azriel Rosenfeld, and Tapas Kanungo. 2003. Document structure analysis algorithms: a literature survey. In *Proceedings of SPIE*, Vol. 5010. 197–207.

George Nagy, Sharad Seth, and Mahesh Viswanathan. 1992. A Prototype Document Image Analysis System for Technical Journals. *Computer* 25, 7 (July 1992), 10–22.

Anoop M. Namboodiri and Anil K. Jain. 2007. Document Structure and Layout Analysis. (2007).

Elisa H. Barney Smith, Daniel Lopresti, and George Nagy. 2008. Ballot Mark Detection. In *Proceedings of ICPR*.

R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Proceedings of ICDAR*, Vol. 02. 629–633.

Mitch Trachtenberg. 2008. Trachtenberg Election Verification System (TEVS). (2008). https://code.google.com/p/tevs/.

Kai Wang, Eric Kim, Nicholas Carlini, Ivan Motyashov, Daniel Nguyen, and David Wagner. 2012. Operator-assisted tabulation of optical scan ballots. In *Proceedings of EVT/WOTE*.

Pingping Xiu, Daniel Lopresti, Henry Baird, George Nagy, and Elisa Barney Smith. 2009. Style-Based Ballot Mark Recognition. In *Proceedings of ICDAR*.

A. Appendix

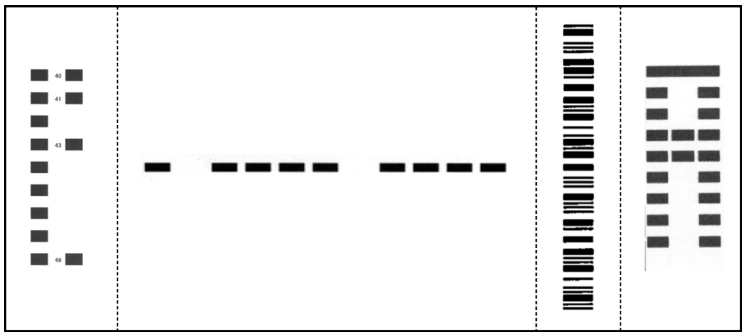


Fig. 7. Examples of barcodes from major vendors. From left to right: ES&S, Diebold, Hart, Sequoia.

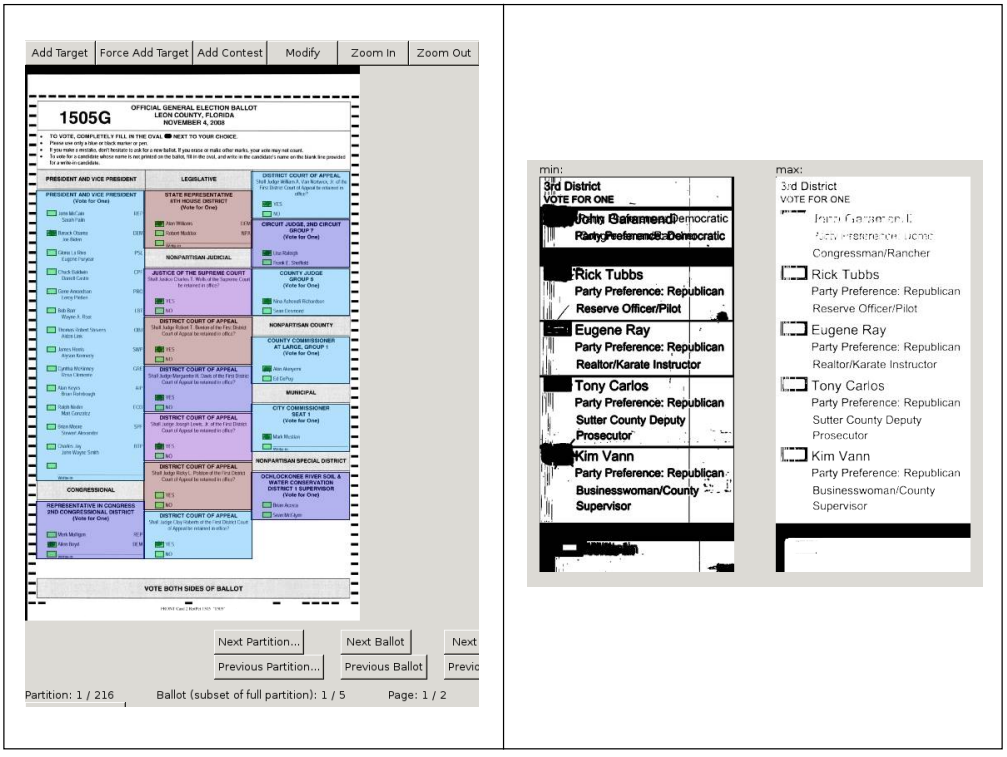


Fig. 8. (Left) The interface where the operator annotates the ballot style for each group. Here, both the voting targets and contest bounding boxes have already been found. (Right) An example of overlay-based verification. Note that there is an error on the first contest.

USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

An Analysis of Long Lines in Richland County, South Carolina

Duncan A. Buell, University of South Carolina

In his State of the Union address, President Obama referred to long lines faced at the polls on November 6, 2012, and said, "we have to fix that." Although it seems to have received relatively little national attention, Richland County, South Carolina, with more than 12% of its votes cast after polls were officially closed, was probably among the very worst counties in the nation for lines and wait times.

In this paper, we analyze the data from the DREs used for voting in South Carolina, and we compare the voting process in Richland County with that in Greenville County, where there were more total votes and more votes per DRE voting terminal, but where there were fewer than one-half of one percent of the votes cast after closing time.

1. INTRODUCTION

In his State of the Union address, President Obama referred to long lines faced at the polls on November 6, 2012, and said, "we have to fix that."[Obama 2013] Although it seems to have received relatively little national attention, Richland County, South Carolina[1], with more than 12% of its votes cast after polls were officially closed, was probably among the very worst counties in the nation for lines and wait times. Although South Carolina has been observed to have had the fourth-worst wait times of any state [Stewart III 2013], we have not seen an analysis that examines the South Carolina experience at the county level. We would expect a significant difference across the 46 counties in the state, however, if only because Richland County cast about 12% of the total votes in the state for the November 2012 general election but had 64% of the votes cast after polls had closed [South Carolina State Election Commission]. Richland had more than 14,000 votes cast after closing time. In contrast, the county with the next largest number of votes cast after closing was Charleston County, almost identical in population and number of voters and votes cast, but with only 1158 late votes. Chester County had the next largest percentage of late votes, at about 3% cast after closing.

Long lines were experienced throughout Richland County [Fretwell and LeBlanc 2012; LeBlanc 2012b; LeBlanc and Lucas 2012; LeBlanc 2012c; Monk 2012]. The presence of a controversial sales tax on the ballot prompted conspiracy theories about anti-tax precincts being shortchanged by a pro-tax elections office. Counting in Richland was not completed until November 15, after two trips to the South Carolina Supreme Court and the sequestration, then return, of the election data by the South Carolina Law Enforcement Division.

South Carolina votes exclusively (except for paper absentee ballots) on ES&S iVotronic direct recording electronic (DRE) terminals with no paper trail, and it declares the entire output of the ES&S system to be public record. We have extensive experience with software for processing the system output and performed an analysis of the Richland County data in an attempt to determine the "what" of what happened on Election Day. We received the data on November 21 and were able to prepare a preliminary analysis of the data by November 23. That analysis was distributed to the county legislative delegation and the media in time for a legislative hearing on the matter on November 26.

This paper is the result of a more extensive analysis of the data from the DREs. In this paper, we analyze the data and we compare the voting process in Richland County with that in Greenville County, where there were more total votes and more votes per DRE voting terminal, but where fewer than one-half of one percent of the votes were cast after closing time.

Our primary goal in this analysis is to understand, as best as possible from the data available to us, what the impact was on voters from the insufficient allocation of terminals. More specifically: How many voters had to stand in line and wait a long time to vote? How long did they have to wait?

[1]Home of Columbia, SC, the state capital, and the University of South Carolina

Download supplemental material

How many voters had to wait in excess of one hour? Did all voters have to wait about the same amount of time, or were some voters affected significantly more than others? News reports gave us some indication that a significant number of voters were affected, but those reports are inherently incomplete, qualitative, and anecdotal and thus cannot give us a full picture. In this paper our goal is to develop methods to answer these questions in a more principled, systematic, and quantitative way and apply them to understand what happened in Richland County.

South Carolina is unusual in that it declares all the output from the ES&S election system to be public record, thus making this kind of analysis possible. However, as will be shown below, our analysis comes with a number of caveats because the data is not as complete as we would like it to be. If electronic terminals are to be used for voting, thus making possible the automatic production of detailed log records, then we would hope that future electronic systems produce better logs. Improved log files, if they were to be available in states other than South Carolina, would permit analysis such as this to be done more widely and thus contribute to a better understanding of the election process in future elections.

2. COMMENTS

There are a number of issues that are not really relevant to a mathematical analysis of the election data; we will mention some but not describe them in detail. One conspiracy theory surrounding the apparent shortfall in allocation of voting terminals was that precincts that had voted against the tax proposal in 2010 (when it failed) had been allocated fewer terminals in 2012 as a way of supppressing the anti-tax vote by members of a presumed pro-tax establishment. We performed an analysis at the request of a newspaper reporter and found that 40 precincts that had voted against the tax in 2010 had fewer terminals in 2012 than in 2010, 49 precincts that had voted for the tax in 2010 had fewer terminals in 2012, and about an equal number on both sides had the same number or slightly more [LeBlanc 2012a]. It thus did not seem apparent from the data that an obvious correlation could be made between the number of voting terminals and past or predicted precinct votes on the tax.

The South Carolina Code of Law Section 7-13-1680 states that polling places should have one voting machine for every 250 registered voters [Hamm 2013]. For this to have been followed, Richland County would have had to deploy nearly 1000 terminals, more than it actually owns. A lawsuit seeking to void the entire election on the basis of the insufficiency of the terminal allocation was filed by a losing candidate for county council. The suit was rejected at all levels, eventually reaching the state Supreme Court, which ended the matter by not taking up the case on appeal.

We have not included in our analysis the data from in-person absentee voting or from the paper absentee ballots cast prior to Election Day or from the in-person voting at county headquarters on Election Day. The actual data from the first two cannot contribute to an analysis of long lines on Election Day (except in the general notion that a smaller number of voters on Election Day would lead to fewer problems with resource allocation and long lines). And the in-person voting at headquarters on Election Day should have been for provisional, fail-safe, etc., voting. By definition, these are not voters arriving at their customary polling place with proper registration and credentials, and we would expect the increased complexity attendant to signing in these voters to vote would skew the wait time statistics.

Further, we have focused here on issues that can be dealt with from the data of the election process. Some other papers on long lines deal largely with the policy issues of how to reduce the lines [American Bar Association Standing Committee on Election Law 2013; Norden 2013; Stewart III 2013], such as whether expansion of early voting could decrease the number of voters on Election Day itself and thus reduce pressure on the terminals at the polling places. Other papers have dealt with the effects on the election outcome of voters who cannot vote due to long lines. In the work of Highton [Highton 2006], for example, the purpose was to show how many votes for Kerry in 2008 in Ohio were lost due to long lines, and that paper did not analyze DRE data. While much work addresses long lines, no other scholarship of which we are aware analyzes DRE data in order to understand whether we had the proper allocation of equipment or to quantify the effect of under-

allocation. These previous analyses do not help us understand how we might determine wait times from the election data the way our data do, and the present study thus provides a unique picture of long lines as well as more definitive evidence of the inadequate allocation of equipment.

We remark finally that other authors have looked at log file analysis [Baxter et al. 2012].

3. DATA

Richland County on Election Day had 244,721 registered voters, having seen an increase of approximately 17,000 registered voters in 2012 prior to the general election. Turnout was 65.43% [South Carolina State Election Commission] with 121,206 (75.7% of the total) ballots cast in person in precincts on election day, 24,118 (15.1%) ballots cast in person at county headquarters during the absentee voting period, and 14,055 (8.8%) ballots cast on paper absentee ballots. All our analysis is based on the data for the 121,206 votes cast[2] "in the precincts", because the votes cast at county headquarters would have very different characteristics as an "election process".

South Carolina votes entirely (except for paper absentee ballots) on ES&S iVotronics terminals. Each terminal produces an EL152 file that is the "event log" record for that terminal, and an EL155 file that is the cast vote record (in randomized order). These files are public records in South Carolina and are posted on the State Election Commission website [South Carolina State Election Commission]. Most of the analysis of this paper comes from what is contained in the EL152 files, which includes the terminal opening time; the terminal closing time; a record of the existence of a cast vote, with a timestamp; and exceptions and error messages.

There are four primary problems with the EL152 files that make the data less useful than it might be and that affect the kind of analysis we can perform. These problems thus contribute to questions about the validity of any analysis such as we have done.

(1) We have an event recorded, with a time, only when a voter actually casts a vote; there is no event recorded when the terminal was opened by a poll worker to allow that person to vote. During the regular Election Day hours, the time between cast votes could thus include idle time when no voters existed as well as the time taken for a voter to vote. After closing, however, we assume that there is a steady stream of voters (or else the precinct would be closed) and thus that the time between votes cast after closing represents the actual time taken for voting.
(2) The internal time in the voting terminal actually has no effect on the terminal's ability to collect votes. Because the software was written prior to the 2007 changeover in Daylight Savings Time, and not updated, this means that some terminals started the day with the internal clock off by one hour. Some of these were corrected during the day; some were corrected after closing; some were not corrected. Among other things, however, this means that there are a few votes (a very few) whose timestamp in the EL152 file is earlier than the timestamp recorded for previous votes. In addition to the one-hour error, there were six terminals set to November 6 of 2014 or 2015, not 2012. As described below, we have chosen to avoid questions of whether we have improperly massaged the data by relying as much as possible only on data we have no reason to believe is incorrect and that needs no interpretation on our part.
(3) We do not in fact get in the EL152 file positive information that a terminal was "not in use" due to error conditions. The EL152 file includes entries for vote events over the course of Election Day, so we can tell when a terminal is in fact recording votes. But we cannot determine for sure whether a terminal that is not recording votes is not recording votes because there are no voters (idle time) or whether it has been effectively taken out of service by the poll workers. Only on rare occasions do we see a terminal formally shut down early. Most of the time, however, a terminal that seems to be performing badly is simply folded up (physically) and not used, but not actually "closed" until it is closed along with the other terminals at the end of Election Day.

[2]There were in fact 121,335 votes cast in precincts on Election Day, but the smaller number is the official count. It was in the process of our analysis of the data that we discovered two terminals in two different precincts, with 27 and 102 votes, respectively, whose votes had not been included in the official count due to errors in the election process.

Table I. Late Votes by Hour, Richland County SC

			Nov 2010	Nov 2012
7pm	-	8pm	1888	6297
8pm	-	9pm	43	4544
9pm	-	10pm	0	2520
10pm	-	11pm	0	1145
11pm	-	12pm	0	255
12pm	-	1am	0	29

(4) We apparently do not always get good information in the EL152 file about some of the malfunctions, or else the poll workers or technicians are unable properly to diagnose malfunctions. Testimony was given that there were abnormally many memory card errors, and there was testimony and reporting of battery problems [Crum 2012; Fretwell and LeBlanc 2012; LeBlanc 2012b; 2012c; McBride 2012]. However, a scan for such problems by the State Election Commission staff did not show Richland to be abnormal relative to the rest of South Carolina [Whitmire 2012]. Subsequent investigation suggested that the problems were not with batteries but with the power supplies, including frayed electrical cords [Goodwin]. As extensive as the reporting of problems was, it would be hard to believe that problems did not exist, but we have, for example, not a single example in the EL152 file for Richland county of any of the event messages related to low battery power.

4. METHODOLOGY

Our analytic approach is to find parameters for a queueing model which, if run on precinct-by-precinct data from Richland and Greenville counties, will accurately model the observed data in that the number of late votes in simulation will be close to the number of late votes actually cast. If we can run the simulation using the data from a given precinct, and the simulated late vote count at closing time is close to the actual late vote count, we would then argue that the time voters spent in line during Election Day would be close to the queue times computed in the simulation. This would provide an analytical picture of the voting process during Election Day on November 6, 2012.

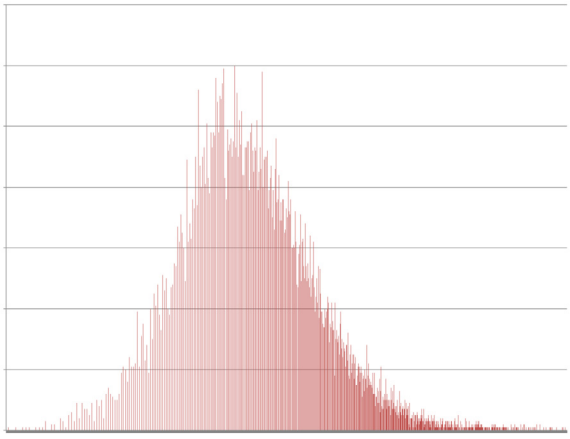
4.1. Computing time-to-vote from the late votes

Polls close in South Carolina at 7pm, but voting continues in the precinct until all voters in line at closing have voted. The distribution of votes cast by hour after closing is in Table I, with a comparison to the Richland County data for the general election in 2010. We note that the last vote cast in Richland County came at 17 minutes, 59 seconds past midnight on the morning of November 7, 2012, in Keels Precinct (number 0336, see Table II), by a voter who had been in line for more than 5-1/4 hours.

As mentioned above, we cannot answer directly the question of how long an individual voter takes to vote. We assume, however, that the time difference between successive EL152 code 1510 "Vote cast by voter" events after closing time represents the actual total time for a single voter to vote, including the time to walk up to the terminal, to have the terminal opened to accept a vote, and the time taken by the voter in voting. We assume that the time between successive votes cast after closing cannot include slack time with no voters present to cast votes, because if there were no voters, then the queue at closing time would have been drained and the precinct would close. In many ways, this is actually better information than just the time spent casting a ballot, at least for the purpose of analyzing the capacity of the precincts to accommodate voters, because it does include the time spent setting up a voter to vote as well as the actual time to vote.

To try to ensure that we are using the most reasonable data, we have trimmed the original (based on EL152 timestamps) set of late votes down to a set of 14,766 differences between successive votes for which both the current and previous vote had a timestamp on November 6, 2012, the current vote was cast after 7pm on Election Day, and the time difference between the current vote and the previous vote was less than 15 minutes. In trimming the vote data in this way, we have

Fig. 1. Histogram of Late Vote Durations (frequency versus log of duration), Richland County, Nov. 2012



eliminated about 1,000 votes from the total set of votes with timestamps after closing. This includes initial votes from a few DREs that were either placed into service after the close of polls or had their internal time reset after the close of polls and then collected more votes. We have assumed that vote times of longer than 15 minutes represent times when the terminal was not being used, not time taken to vote. For example, terminal 5118258 in precinct 0353 collected a vote at 12:08pm, in a time period in which the terminal was recording a number of error messages, and then did not collect another vote until 8:58pm. This terminal had a total of 41 votes in a precinct whose other eleven terminals ranged from 185 to 234 votes collected.

The distribution of the logarithms of the time-to-vote numbers for the 14,766 late votes (as counted above) in Richland County is shown in Figure 1. We have modelled the time to vote with a log normal distribution; for this data the distribution would be modelled as a random variable $X = e^{\mu+\sigma Z}$, with Z a standard normal variable [Parzen 1960], and with $\mu = 5.28282$ and $\sigma = 0.363715$ from the data shown in Figure 1. We note that $e^{5.28282} \approx 197$ seconds. The county-wide data for durations prior to closing results in $\mu = 5.30438$ and $\sigma = 0.387181$, with $e^{5.30438} \approx 201$ seconds.

The votes cast after closing time can be assumed to include no idle time because the process at the polls should simply be one of draining the queue of voters present at closing time. During the day, we might expect idle time. Nonetheless, we have computed the mean and deviation for the votes cast in Richland and Greenville counties prior to closing time: For Richland prior to closing time, we have $\mu = 5.30438$ and $\sigma = 0.387181$, with $e^{5.30438} \approx 201$ seconds. For Greenville, we have $\mu = 5.16941$ and $\sigma = 0.469847$, with $e^{5.16941} \approx 176$ seconds.

4.2. Distribution of Voters

Loosely following Edelstein and Edelstein and their sources from Columbia County, New York [Dow 2007a; 2007b; Edelstein and Edelstein 2010], we assume that 10% of the voters are queued up at opening, 10% arrive in each of the peak hours 7:00-8:00am, 8:00-9:00am, noon to 1:00pm, 1:00-2:00pm, 5:00-6:00pm, and 6:00-7:00pm, and equal distributions (5% each) for the other hours.

Using these fractions of total vote arriving in hourly intervals during Election Day, we can determine the number of voters to arrive in each hour and generate a voter queue using a standard Poisson interarrival time [Parzen 1960].

4.3. "Effective" Number of Voting Terminals

Although by the end of Election Day there had been 628 terminals that collected votes, it is not the case, as mentioned above, that all terminals were working all the time. Newspaper reports and the

testimony of election officials [Crum 2012; Fretwell and LeBlanc 2012; LeBlanc 2012b; LeBlanc and Lucas 2012; McBride 2012] were that terminals were malfunctioning. Additional terminals were delivered to some precincts during the day. As mentioned above, it is not possible automatically to determine that a terminal is not functioning or that the poll workers have decided not to use that terminal. These issues complicate the determination of the number of voting terminals at a given precinct. In Sandlapper Precinct in Richland County (number 0390), for example, a total of 13 terminals appear in the final data, but five of these were actually delivered and opened for voting *after the polls had closed at 7pm*.

We have determined by detailed manual examination of the logs an independent estimate of the "effective" number of terminals available at a precinct during the day. This number is frequently lower than the physical number of terminals present at the precinct. Our premise in arguing that the simulation is valid is that the number of terminals in simulation that results in the closest match of the number of simulated late votes to the actual number of late votes should be the same as the effective number of terminals we find by manual examination of the event logs and the histograms of votes collected by the terminals.

For all the precincts used in our simulation detailed below in Tables II and III, we have looked carefully at the EL152 data and the histograms derived from that data of votes collected during the day, to try to determine an effective number of terminals in use. Indicators that suggest that terminals might not have been available all the time include the following:

— A terminal that was opened late was clearly not available for the full 12 hours of Election Day, and we prorate the effective number of terminals accordingly.
— A terminal whose total number of votes is much smaller than the rest of the terminals in the precinct bears further scrutiny.
— We have histograms for each terminal of votes collected in 15-minute intervals. If a given terminal is not collecting votes in a precinct where other terminals are collecting votes, then it is either idle due to lack of demand (i.e., voters) or else the poll workers are choosing not to use it, and a detailed examination of the log is warranted.
— We have assumed that terminals whose last recorded vote is much earlier than closing time, in precincts for which other machines were collecting votes at a steady rate, were malfunctioning, and a detailed examination of the log is warranted.

We have used all these indicators to guide us in determining the effective number of terminals by looking at the details of the EL152 logs. Except for the first bullet above, we admit that there is some subjective judgement here. In virtually all cases, however, the phenomenon of few or no votes on a given terminal in a given time interval, or of a last vote cast early in the day, in precincts where and times when other terminals are collecting votes at a steady pace, coincides with events visible in the log. Often what we see are frequent screen calibrations or frequent failed attempts to connect a PEB to the terminal (as is necessary to open the terminal for receiving each vote).

We remark that our estimate of the effective number of terminals is almost certainly always high, which would eventually translate into simulated wait times that were too low. We can know that a terminal not yet opened is not available, but we can only argue that a terminal is not available if it is anomalous in its precinct with respect to the collecting of votes, especially in a precinct with a large number of late votes. What we cannot determine is whether a terminal is "slow" relative to other terminals in the precinct; we always count terminals as available and effective if there is no explicit reason to indicate otherwise. For example, in precinct 0101, the four terminals collected 114, 159, 187, and 199 votes. The 114 votes in terminal 5127549 is lower than the other three terminals, which might indicate that it had not been functioning properly. However, there are no event log messages for that terminal that would indicate that anything was amiss, so we have assumed that it was functioning correctly.

4.4. The Queueing Model

We sample from a standard normal distribution and exponentiate to determine a log-normally-distributed time-to-vote for each voter. For the precincts of Table II with a large number of late votes, we use both the mean and deviation of vote times for votes cast in each precinct before closing time and the mean and deviation for vote times cast in each precinct after closing time. For the precincts of Table III with no late votes and for the precincts in Greenville County, we use the mean and deviation for vote times cast before closing time. We admit that neither time is free from caveats. The times before closing will include idle time between voters, if any idle time existed, and might thus be too large. The times after closing will not include idle time, but might also be larger as a consequence of fatigue on the part of either voters or poll workers.

We simulate voting in each precinct for a range of terminal numbers centered on the number given by our manual independent analysis of the effective number of terminals for each precinct. We compare the simulated number of late votes with the actual number. If, for the simulation using the manually-computed estimate of effective numbers of terminals, these two numbers are close, we conclude that the model is reasonably accurate and thus that the simulated wait times are a good estiamte of the actual wait times experienced by the voters.

4.5. Capacity Issues

Edelstein and Edelstein [Edelstein and Edelstein 2010] argue that voting terminals should be allocated at no more than half their maximum capacity. Their analysis is done using Maryland's rules, with a 13-hour Election Day and with voters permitted to take five minutes to vote. South Carolina has a 12-hour Election Day and officially permits a voter a maximum of three minutes to vote. If for projection and allocation purposes we assume that all voters take three minutes, then one terminal used for a 720-minute Election Day has a capacity of $V_{possible} = 720/3 = 240$ votes. The half-capacity argument of Edelstein and Edelstein would then require the allocation of sufficiently many terminals to ensure that no terminal was used beyond half its capacity, or an actual vote total V_T of 120 votes, so that a capacity bound $C = V_T/V_{possible} < 0.5$ is maintained.

Richland County, at 192 votes per terminal county-wide, far exceeded that, as did Greenville County at 209 votes per terminal in the precincts on Election Day.

4.6. Best and Worst Precincts, Richland County

We select as the worst twelve precincts in Richland County the set union of the top ten for the largest number of late votes and the top ten for the largest fraction of late votes. To this we have added two precincts that had substantial late votes, 0340 and 0378, and for which we have our own first-hand information about the effective number of terminals at the precinct. As can be seen in Table II, the mean times for the late votes for these worst precincts are both larger and smaller than the county mean of 210.68 seconds, which is 3.511 minutes.

At the other end of the spectrum, we have randomly selected fourteen precincts from among the 41 precincts that had at most one vote per terminal after closing time, and we have run the same simulation.

5. SIMULATION RESULTS

We ran our simulation twenty times for each terminal count 2 through 14. In each case we use as our mean vote duration, for the precincts in Table II with the worst records for late votes, the observed precinct mean vote durations for votes before closing and after closing. And in each case we use as our mean vote duration, for the precincts in Table III and for the precincts in Greenville County for which few or no late votes were observed, the observed precinct mean vote duration time for votes before closing.

5.1. Richland County

Our analysis begins with the Richland County precincts with late votes; these results are presented in Table II. All times in Tables II and III are in minutes.

The second column is the actual number of terminals present and the effective number of terminals present at the precinct. The 10/9 for precinct 0318, for example, indicates that there were 10 terminals but only 9 and not 10 twelve-hour days of working terminals. The third column is the number of terminals used in the simulation. The fourth and fifth columns are the means and deviations of the vote durations after closing. Columns six and seven are the means and deviations of the number of simulated votes, and column eight is the actual number of late votes. Column nine has the mean time spent by a voter in the queue in the simulation, and column ten is the maximum time spent in the queue in the simulation. Finally, the last column is the capacity quotient $C = V_T / V_{possible}$ that Edelstein and Edelstein argue needs to be less than 0.5 to prevent long lines.

We have included as many lines of simulated terminal counts in each precinct as seemed appropriate for presenting both the fixed simulation result for that many terminals and the sensitivity of the simulated late vote count to the number of terminals simulated. As must be the case mathematically, capacity quotient C exceeds 1.0, late votes must appear, and we note the dramatic change that frequently appears across the boundary of $C < 1.0$ and $C > 1.0$.

We note also that the mean vote durations in these precincts are both larger than and smaller than the county-wide means both for the "before" times and the "after" times. Nine of the fourteen precincts have "before" times larger than the county mean and five have times smaller than the county mean. This is reversed for the "after" times: nine precinct times are smaller than the man and five are larger.

We observe that the simulated number of late votes in all the "no late vote" precincts (Table III), using the "before" time for vote duration, are within a 95% confidence level of the number of actual late votes. Only four of the 14 "late vote" precincts (Table II) are within a 95% confidence level[3]. We note that in all these other precincts except 0390, the 95% confidence level would come, in our simulation, for a non-integral simulated number of terminals only slightly larger (by less than one full terminal) than our manually-determined effective number, and in all these precincts our manual analysis results in an effective number smaller than the actual physical number of terminals.

We argue that the results in Tables II and III suggest that our model is valid. In precincts 0324, 0327, 0340, 0358, 0359, 0363, 0378, 0380, 0388, and 0392 the simulated count of late votes using both the before times and the after times is closest to the actual number of late votes either for the same effective number of terminals as determined by our independent examination, or for some fractional effective number of terminals no more than one larger. In precincts 0318 and 0336 the before times match up with our independent estimate of effective number of terminals, but the after times do not. In precinct 0353 the opposite is true, in that the after time matches up but the before time suggests more terminals might have been effective. Only precinct 0390 is truly anomalous; the actual late vote counts are worse than suggested by the simulation. Although we cannot tell that fewer than 6 terminals were genuinely usable during the day, this was the precinct where five terminals were delivered after closing. Perhaps there were other complications that did not show up in the logs.

We compare these late-vote precincts with the precincts whose data is presented in Table III. In this table, we have used in the fourth and fifth columns the means and deviations of time to vote computed from votes cast in that precinct during the day. This number is almost surely too large, in that it will likely include idle time. Nonetheless, with only two possible question marks for precincts 0103 and 0362, the simulated counts of late votes are consistent with the observed data of zero or one late votes. Precinct 0103 had three terminals working all day, with a fourth that was opened at 2:45pm (thus $3 + 4.25/12 \approx 3.4$ effective terminals). Precinct 362 had three terminals working all day, and a fourth that was opened late at 9:30am (thus $3 + 9.5/12 \approx 3.8$ effective terminals).

[3]That is, the actual number of late votes falls within the interval $[mean - 1.96 \cdot dev, mean + 1.96 \cdot dev]$, where *mean* and *dev* are the mean and deviation of the late votes from the simulation.

Table II: Results for Precincts with Late Votes, Richland County SC

Pct	Real T	Sim T	Mean Time	Dev Time	Sim Late Mean	Sim Late Dev	Act Late	Mean Queue Time	Max Queue Time	Capacity Quotient
318	10/9	8	3.691	1.789	665.600	11.504	484	189.910	306.108	1.429
(before)		9			478.850	16.898		134.057	196.281	1.270
		10			277.650	16.469		86.673	127.675	1.143
		11			81.750	19.470		48.712	105.609	1.039
		12			5.150	6.806		28.395	86.992	0.952
318	10/9	8	3.157	0.984	403.650	10.341	484	115.124	159.318	1.222
(after)		9			172.450	16.280		65.309	114.633	1.086
		10			5.000	7.443		32.971	91.910	0.977
		11			8.350	8.850		21.071	73.922	0.889
324	3/2	2	2.944	1.207	450.400	7.908	309	364.830	661.051	1.930
(before)		3			203.550	9.271		132.781	197.380	1.286
		4			3.150	5.247		29.500	86.790	0.965
324	3/2	2	2.797	0.965	425.300	5.469	309	332.051	595.119	1.834
(after)		3			171.850	7.485		114.283	161.924	1.223
		4			3.350	4.704		23.281	80.151	0.917
327	4/3	3	3.379	1.272	468.850	6.858	355	297.482	527.801	1.741
(before)		4			260.300	11.555		145.465	220.285	1.306
		5			43.650	12.881		49.842	106.124	1.045
		6			3.950	4.330		19.676	71.873	0.870
327	4/3	3	2.931	0.943	374.150	6.901	355	216.809	360.514	1.510
(after)		4			124.000	9.110		81.127	124.950	1.133
		5			5.350	7.761		22.436	76.723	0.906
336	6/5	5	5.009	2.962	500.100	9.914	434	288.396	511.124	1.688
(before)		6			360.600	11.749		187.181	306.554	1.406
		7			212.750	13.386		112.026	157.530	1.206
		8			72.350	13.890		57.675	112.472	1.055
		9			6.250	8.318		27.010	85.130	0.938
336	6/5	5	4.394	2.122	386.350	13.774	434	204.354	336.055	1.480
(after)		6			221.200	15.949		115.166	163.512	1.234
		7			52.900	12.837		51.473	107.404	1.057
		8			2.900	4.471		23.236	80.432	0.925
340	6/4	5	3.363	1.592	138.950	16.539	138	80.418	123.559	1.141
(before)		6			5.850	8.278		26.493	83.061	0.951
340	6/4	5	3.382	1.208	157.100	8.843	138	86.854	126.736	1.148
(after)		6			2.950	5.287		27.777	84.503	0.957
353	12/9	9	4.530	1.976	963.250	16.658	714	275.482	482.744	1.683
(before)		10			801.800	14.918		215.149	360.490	1.515
		11			647.750	13.145		167.779	263.675	1.377
		12			487.250	16.807		126.828	180.872	1.262
		13			328.550	13.098		91.410	130.277	1.165
353	12/9	9	3.794	1.304	694.200	13.113	714	181.212	294.264	1.410
(after)		10			506.150	14.051		131.217	193.239	1.269
		11			311.650	15.743		89.334	130.911	1.153
		12			118.550	18.710		54.001	108.290	1.057
		13			8.150	9.645		33.771	91.432	0.976
358	8/4.7	4	3.746	1.437	530.250	11.580	415	282.043	496.914	1.696
(before)		5			340.500	12.655		161.959	255.202	1.357
		6			139.950	10.618		77.812	123.192	1.131
		7			4.250	6.090		30.600	88.841	0.969
358	8/4.7	4	3.460	1.116	470.900	7.641	415	238.353	407.163	1.567
(after)		5			254.350	10.036		121.783	178.498	1.253
		6			52.250	9.591		49.265	105.117	1.045
		7			2.500	5.545		20.720	73.784	0.895

359	7/7	6	3.627	1.332	734.550	9.500	543	256.919	441.554	1.619
(before)		7			527.550	10.888		171.194	272.991	1.387
		8			332.950	12.986		110.082	152.009	1.214
		9			128.000	14.188		60.582	111.150	1.079
		10			2.950	6.430		32.392	90.738	0.971
359	7/7	6	3.203	1.028	573.850	11.616	543	187.910	306.370	1.429
(after)		7			351.750	14.652		115.777	159.233	1.225
		8			123.150	14.964		60.432	113.174	1.072
		9			3.700	5.951		29.031	87.547	0.953
		10			1.850	3.351		19.034	67.862	0.858
363	7/4	4	3.702	1.405	575.700	13.039	375	301.889	533.794	1.746
(before)		5			380.450	13.079		177.113	285.708	1.397
		6			189.150	10.716		93.654	134.302	1.164
		7			6.150	6.350		37.914	98.576	0.998
363	7/4	4	3.077	1.068	419.050	7.612	375	197.445	324.777	1.451
(after)		5			186.600	9.260		93.862	133.083	1.161
		6			4.700	7.356		33.388	92.487	0.967
		7			4.250	5.873		17.616	64.789	0.829
378	6/4	4	2.808	0.890	257.950	10.689	181	125.967	181.465	1.256
(before)		5			10.250	8.209		39.647	98.866	1.005
		6			4.500	7.527		17.542	64.789	0.837
378	6/4	4	2.864	1.068	275.850	10.795	181	134.164	195.342	1.281
(after)		5			25.850	10.268		43.979	102.652	1.025
		6			6.600	6.981		18.685	69.140	0.854
380	7/4	4	2.962	1.107	469.150	11.778	459	209.004	348.499	1.487
(before)		5			222.050	11.800		100.099	137.291	1.190
		6			6.500	6.667		35.470	93.388	0.992
		7			4.750	5.735		18.206	66.704	0.850
380	7/4	4	3.080	1.059	505.400	10.027	459	229.753	388.958	1.546
(after)		5			270.800	11.830		117.924	166.222	1.237
		6			32.550	9.785		44.483	102.000	1.031
		7			4.800	6.882		20.139	72.559	0.884
388	10/7	7	4.112	1.695	699.750	11.502	561	238.715	406.235	1.576
(before)		8			513.300	13.214		165.021	262.246	1.379
		9			343.750	16.938		112.961	156.970	1.226
		10			172.750	12.091		70.146	118.117	1.103
		11			8.100	9.181		38.201	95.987	1.003
388	10/7	7	3.869	1.310	622.000	12.462	561	206.457	344.653	1.483
(after)		8			439.550	9.282		140.689	213.042	1.298
		9			247.300	11.576		87.976	128.838	1.154
		10			68.900	14.254		48.051	103.753	1.038
		11			5.100	8.677		27.034	84.378	0.944
390	12/6	6	3.902	1.560	462.350	12.587	520	185.266	300.297	1.425
(before)		7			276.300	14.990		113.310	157.732	1.222
		8			89.250	11.567		57.688	111.245	1.069
		9			3.650	4.497		29.241	87.354	0.950
390	12/6	6	3.796	1.271	433.950	9.184	520	172.750	272.731	1.387
(after)		7			245.350	9.881		102.368	141.211	1.189
		8			53.500	12.963		48.977	107.095	1.040
		9			4.350	6.460		24.609	80.954	0.924
392	6/5	5	4.286	1.895	339.850	8.856	325	179.375	289.127	1.411
(before)		6			166.150	12.039		92.947	131.349	1.176
		7			9.950	8.071		37.887	96.889	1.008
392	6/5	5	4.179	1.665	319.600	10.017	325	166.984	262.676	1.376
(after)		6			144.600	10.418		85.896	129.087	1.146
		7			2.750	3.590		33.650	92.054	0.983

5.2. Queue Times

We argue that these two tables largely validate our model and thus that we can infer queue times for the precincts of Richland County. For most of the precincts in Table II, the mean queue times

Table III. Results for Precincts with No Late Votes, Richland County SC

Pct	Real T	Simu T	Mean Time	Dev Time	Sim Late Mean	Sim Late Dev	Act Late	Mean Queue Time	Max Queue Time	Capacity Quotient
101	4/4	4	3.901	1.748	1.650	2.903	1	22.568	79.589	0.893
103	4/3.4	3	2.875	1.024	83.650	8.788	5	73.335	118.566	1.119
		4	2.875	1.024	2.650	4.028	5	16.032	61.227	0.840
116	3/3	3	2.671	1.280	0.500	1.118	1	24.807	81.076	0.944
130	3/3	3	4.585	2.606	2.100	3.727	1	14.956	63.371	0.819
133	3/3	3	4.010	1.851	2.050	4.489	0	30.554	92.210	0.958
307	6/6	6	3.895	1.763	1.400	3.056	0	26.057	84.105	0.956
309	8/8	8	4.049	1.544	5.100	8.426	0	23.012	79.254	0.918
310	6/6	6	4.383	2.077	2.400	3.625	0	29.120	89.250	0.951
321	3/3	3	3.086	1.249	1.950	2.783	0	28.736	86.602	0.971
343	3/3	3	4.230	1.913	1.900	3.419	0	25.170	84.766	0.950
351	3/3	3	3.619	1.444	2.700	3.703	0	33.666	94.436	0.990
362	4/3.8	3	4.333	2.188	58.650	9.707	1	74.846	120.870	1.127
		4	4.333	2.188	1.750	3.562	1	16.355	64.693	0.846
368	10/10	10	3.231	1.223	2.300	5.367	1	33.071	91.662	0.981
370	3/3	3	3.258	4.443	3.300	4.196	0	31.842	92.809	0.884

are in excess of one hour and more commonly in excess of two hours, with maximum queue times of four hours or more in eleven of the fourteen precincts. These wait times in the simulations for the late-vote precincts are not out of line with anecdotal and news media reporting of wait times ranging from four to seven hours in several precincts [Fretwell and LeBlanc 2012; LeBlanc and Lucas 2012].

Our simulations show that in just the 14 precincts of Table II, there were nearly 19,000 voters who suffered queue times greater than one hour and more than 14,000 voters whose wait times were in excess of two hours. County-wide, our simulation indicates that more than 53,000 voters, about 44% of the county total, had wait times in excess of an hour, and more than 20,000 voters, about 17% of the total, had waits in excess of two hours. Even in the precincts of Table III, with essentially no late votes, the simulation indicates that more than one-fourth of the 11,558 voters in those precincts had wait times in excess of one hour. These are conservative estimates. Our simulation runs only for integral numbers of terminals, and for the precincts for which we have fractional estimates of the effective number of terminals, we have chosen the smaller queue times so as to underestimate the wait times.

5.3. Greenville County

We turn now to Greenville County, the largest of South Carolina's counties, which cast 171,146 votes in precincts on Election Day. There were actually more votes on average per terminal in Greenville than in Richland, and yet there were only 403 late votes, less than one-half of one percent of the total. Those late votes are clustered in precincts 0208 (35 late votes), 0247 (86), and 0298 (81). Precinct 0208 had eight terminals and 35 late votes, which is fewer than five late votes per terminal. These three precincts of the 151 total account for half the late votes in the county.

We have computed the mean times to vote during regular hours for each of the precincts and run our simulation on Greenville County to produce what would be the analogs of Tables II and III.

In no precinct except 0247 and 0298 do we get a simulated count of more than ten total late votes in a precinct. It is true, however, that in all but six precincts the Edelstein and Edelstein capacity quotient is greater than 0.8, the mean queue times are less than 15 minutes in only two of those six precincts, and the maximum queue times range from one to two hours. Indeed, in all but precincts 0208, 0247, and 0298, the simulated number of late votes falls within a 95% confidence level of the actual number of late votes. For 0247 and 0298, the simulation results in slightly too few late votes to fall within a 95% confidence interval. The histogram of votes collected in 15-minute intervals does not show obvious down time, but a close examination of terminals shows that in each of these two precincts there was at least one terminal with a significant number of problem events

Table IV. Mean Number of Ballot Choices, Richland and Greenville Counties SC

	Richland	Greenville
Contested	6.53	4.04
Uncontested	6.50	7.32
Combined	13.02	11.36

recorded. These were usually failues of the PEB to connect to the terminal, as well as an instance of recalibration of the terminal. Even a slight slowdown of vote collection (that could alternatively be viewed as a failure to function at 100% capacity) would account for the statistical discrepancy between the simulation and the actual number of late votes.

6. ANALYSIS

6.1. Issues of Capacity

Starting with the capacity of a given terminal, we can begin to understand why long lines appeared in Richland County. In order for 121,206 votes to be cast in Richland County in 12 hours with an average vote taking 190 seconds (ten seconds for setup plus the official 180 seconds allowed to a voter), a total of 533 voting terminals would be needed, working nonstop, with no slack or idle time, no morning, noon, or after-work surges, and with no uneven distribution of voters or of voting terminals in the various precincts. If the 190 seconds is increased to the 217 seconds observed county-wide in votes after closing, the minimal number of terminals in order to finish voting in 12 hours is increased to 608. Our analysis of the "vote cast" events in the EL152 file, however, shows that only 539 terminals were open in Richland at 7:00am, only 555 terminals were open by 8:00am, and during no quarter-hour period of the day were there more than 578 terminals collecting votes. A total of 628 terminals collected *some* vote during the day, although about 75 of these were either opened late due to malfunction or delivered late when the enormity of the catastrophe had become apparent. Even under the best possible queueing circumstances, voting could not have been accomplished in 12 hours if the average time between cast votes was 217 seconds.

Greenville County was positioned almost as precariously as Richland; at 190 seconds per vote, its 171,146 votes would have required 752 terminals. At its actual mean time before closing of 198 seconds, 784 terminals would be needed. However, Greenville had more than 820 terminals in operation through most of the day, with more than 810 collecting votes for all but 90 minutes of the 12 hours of Election Day. As the queueing analysis shows, small changes in resources that would move from the safe to the unsafe side of maximal capacity can have a marked impact for the presence or absence of queues.

6.2. Issues of Ballot Complexity and Precinct-Precinct Variation

On the one hand, the fact that Richland had quite literally too few terminals to accommodate the turnout and Greeville at least avoided that problem provides one reason that the late vote experiences of the two counties were quite different. However, the issue of ballot complexity has also been an issue in affecting the time taken for voters to vote. The Greenville County mean time to vote of 198 seconds during regular hours is noticeably smaller than the 217 seconds in Richland, and the increase of only a few seconds in transaction time can have a significant effect on the queueing in conditions (as in both counties) where resources are inadequate or just barely adequate.

We have examined the ballots in both counties, and present the data in Table IV. Weighting the number of choices faced by a voter with the number of voters making that number of choices, we find that the average voter in Richland County would have had 6.53 choices in contested races and 6.50 choices in uncontested races, for a total of 13.02 choices on the ballot. The Greenville County ballot was shorter in contested races by nearly 2.5 choices. In Richland, there was a mean time of 16.88 seconds per choice (contested and uncontested) with a deviation of 2.61, and in Greenville the mean and deviation per choice was 17.72 and 2.35.

7. CONCLUSIONS

We argue that our queueing model is a reasonable approximation to what actually happened in Richland County on Election Day, and that we have simulations to show that large numbers of voters were in line for several hours. We know from news reports and television footage that this was so, but if our simulation is correct we know that nearly half of Richland County voters waited in line more than an hour.

We would certainly prefer to get better and more complete data from the voting terminals. However, we nonetheless argue that the existing, imperfect, data can be used to model the voting process so that conclusions about queue times and late votes can be drawn.

REFERENCES

American Bar Association Standing Committee on Election Law. 2013. Election Delays in 2012. http://www.americanbar.org/content/dam/aba/administrative/election_law/2012_election_delays_report.authcheckdam.pdf (2013).

Patrick Baxter, Anne Edmundson, Keishla Ortiz, Ana Maria Quevedo, Samuel Rodriguez, Cynthia Sturton, and David Wagner. 2012. Automated analysis of election audit logs. *Proceedings, EVT/WOTE* (2012).

E. Crum. November 26, 2012. Testimony of Board of Elections and Voter Registration chair Elizabeth Crum. Richland County Legislative Delegation Hearing. (November 26, 2012).

K. Dow. 2007a. Study of Voter Flow at the 2006 General Election, Columbia County, NY. *Columbia County Board of Elections* (2007).

K. Dow. 2007b. Voter Flow During Each 2-Hour Interval Throughout Election Day. *Columbia County Board of Elections* (2007).

William A. Edelstein and Arthur D. Edelstein. 2010. Queuing and Elections: Long lines, DREs, and paper ballots. *Proceedings, EVT/WOTE* (2010).

Sammy Fretwell and Clif LeBlanc. 11/07/2012. S.C. turnout heavy; voters in Richland wait hours. The State (Columbia SC). (11/07/2012).

Cheryl Goodwin. (deputy elections director for operations, Richland County) personal communication. (????).

Steven W. Hamm. 6/27/2013. Final Summary Report on the November 6, 2012, Richland County General Election . (6/27/2013).

Benjamin Highton. 2006. Long Lines, Voting machine availability, and Turnout: The case of Franklin County, Ohio, in the 2004 Presidential Election. *PSOnline* (2006), 65–68.

Clif LeBlanc. 11/08/2012b. Richland mess 'violated law'; why so few voting machines? The State (Columbia SC). (11/08/2012).

Clif LeBlanc. 11/11/2012c. Vote-counting expert, former voting machines' technician detail possible missteps in Richland County's election. The State (Columbia SC). (11/11/2012). http://www.thestate.com/2012/11/11/2515647/vote-counting-expert-former-voting.html#storylink=misearch.

Clif LeBlanc. 12/04/2012a. Election data show no evidence of bias. The State (Columbia SC). (12/04/2012).

Clif LeBlanc and Mindy Lucas. 11/10/2012. Richland recount halted; S.C. high court intervenes; election results in limbo. The State (Columbia SC). (11/10/2012).

L. McBride. November 26, 2012. Testimony of Richland County Election Director Lillian McBride. Richland County Legislative Delegation Hearing. (November 26, 2012).

John Monk. 11/14/2012. Richland County vote count resumes today. The State (Columbia SC). (11/14/2012). http://www.thestate.com/2012/11/14/2518148/ /sc-democrats-agree-to-drop-recount.html#storylink=misearch .

Larry Norden. 2013. How to Fix Long lines. *Brennan Center for Justice* (2013).

Barack Obama. 2013. State of the Union Address. (2013). http://www.whitehouse.gov/the-press-office/2013/02/12/remarks-president-state-union-address .

E. Parzen. 1960. *Modern Probability Theory and Its Applications*. John Wiley & Sons. pp. 251ff.

South Carolina State Election Commission. Election Results, last accessed 3/17/2012. (????). http://www.enr-scvotes.org/SC/Richland/42553 /112776/en/summary.html .

Charles Stewart III. March 23, 2013. Waiting to vote in 2012. In *The Voting Wars: Elections and the Law from Registration to Inauguration*. University of Virginia Law School, Charlottesville, Virginia.

Chris Whitmire. 12/4/2012. (public relations officer, SCSEC) by email from the SCSEC. (12/4/2012).