Using Prêt à Voter in Victorian State elections

Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, Zhe Xia*

July 17, 2012

Abstract

The Prêt à Voter cryptographic voting system was designed to be flexible and to offer voters a familiar and easy voting experience. In this paper we present a case study of our efforts to adapt Prêt à Voter to the idiosyncrasies of elections in the Australian state of Victoria. The general background and desired user experience have previously been described; here we concentrate on the cryptographic protocols for dealing with some unusual aspects of Victorian voting. We explain the problems, present solutions, then analyse their security properties and explain how they tie in to other design decisions. We hope this will be an interesting case study on the application of end-to-end verifiable voting protocols to real elections.

1 Introduction

End-to-end verifiable election protocols are well studied in the academic literature, but until recently have not been deployed in public elections. In 2011 the Victorian Electoral Commission (VEC) approached the Prêt à Voter team to investigate adapting the scheme to the special requirements of the Victorian parliamentary elections, which use both (single-seat) IRV and 5-seat STV[1]. The first prototypes are available at the time of writing and the development of production systems is underway. Trials on small scale by-elections are expected in 2013, with the goal of having the system ready for the next Victorian State election in November 2014.

The proposed protocol is universally verifiable, meaning that there are no trust assumptions for guaranteeing the integrity of the votes.[2] There are probabilistic assumptions about the number of voters who audit some Prêt à Voter ballots, the number who check that their printout matches their intended vote, and the number who read the Web Bulletin Board (WBB). It also provides voters with evidence of malfeasance, assuming that they check the signature on their receipt before they leave the polling station.

Since this is a polling-station scheme, we do not address eligibility verifiability. Prevention of ballot stuffing is by existing procedural mechanisms.

The main departure from standard Prêt à Voter is the use of a computer to assist the user in completing the ballot. This necessitates trusting that device for vote privacy, which is different from standard Prêt à Voter in which the voter does not need to communicate her vote to any (encryption) device. This modification is necessary for usability, because a vote can consist of a permuted list of about 30 candidates. It seemed infeasible for a voter to fill in a Prêt à Voter ballot form without assistance. Indeed, simply filling in an ordinary paper ballot with about 30 preferences is a difficult task. About 2% of voters accidentally disenfranchise themselves by incorrectly filling in their vote. Computerised assistance is an important benefit of the project, and trusting the device for privacy seemed an almost unavoidable result of that usability advantage.[3] Hence our scheme depends on stronger privacy assumptions than stan-

*Craig Burton is at the Victorian Electoral Commission. Chris Culnane, James Heather, Steve Schneider, Sriramkrishnan Srinivasan and Zhe Xia are at the University of Surrey and are supported on the Trustworthy Voting Systems project by EPSRC grant EP/G025797/1. Thea Peacock and Peter Y. A. Ryan are at the University of Luxembourg and are supported on the SeRTVS project by FNR Luxembourg. Vanessa Teague is at the University of Melbourne and Roland Wen is at the University of New South Wales.

[1] For more information on various election methods, please refer to the appendix in [XCH+10].

[2] Vision impaired voters must assume that at least one device reads accurately to them.

[3] In principle one could use an EBM to fill in a series of ballots and only cast one of them, without telling the device which one. This is too much work for voters.

dard Prêt à Voter. Providing privacy for complex ballots is notoriously difficult, and is further complicated by some details of Victorian elections that are described below. Our system provides privacy and receipt-freeness under reasonable assumptions about the correct randomised generation and careful deletion of secret data, and of course assuming a threshold of decryption key sharers do not collude. It does not fully defend against the "Italian Attack," or all other subtle coercion issues, but neither does the current paper-based system. We make this more precise below.

1.1 Challenges of Victorian Voting

Prêt à Voter was designed originally for first-past-the post voting, in which each voter chose a single candidate [CRS05]. Subsequent papers extended the scheme to more complex types of elections [RS06, Rya08, RBH+09, XCH+10]. However, the state of Victoria, like many other Australian states, allows voters a sort of hybrid between single-choice and ranked-choice voting. Each citizen can vote for both a single representative in the Legislative Assembly (LA) and a set of representatives in the Legislative Council (LC). Each LA representative is elected by IRV with compulsory complete preference listing, with rarely more than 10 candidates. Voters for the Legislative Council (LC) typically choose from among about 30 candidates. They may cast either a standard STV vote of optional length (at least 5, and up to all preferences), or a single selection of a political group. Each political group (of which there are about 12) sets a (complete) STV vote which becomes the meaning of all votes cast for that group. Traditionally, both voting options are presented on the same ballot paper. The single-group selections are presented on top of a thick line, and are referred to as "above-the-line" voting (LC-ATL); the full STV options are shown below the line (LC-BTL).

Each polling place must accept votes from a resident of anywhere in the state. Hence our system must produce Prêt à Voter ballots for every electoral division in both the LA and the LC, available at every polling place. This is a significant challenge for Prêt à Voter, but Prêt à Voter confers the great advantage of verifiability on these votes. The existing methods of verifiable paper counting do not work with this requirement. For the large fraction of people who vote outside their home electorate, completed paper ballots must be sent to the home electorate by courier, usually arriving after the polling-station count has been completed and after observers have departed.

This system will not be responsible for all of the votes cast in the upcoming state election, so it will have to combine with existing procedures for casting and counting ordinary paper ballots. For LA and LC-ATL votes this is straightforward. However, LC-BTL votes are complicated. Even those cast on paper must be tallied electronically—in the existing system they are manually entered and then electronically tallied. The authorities then publish complete vote data to allow observers to check the count.[4]

Preferential elections are vulnerable to coercion through signature attacks [DC07], commonly referred to as Italian attacks, as discussed in Appendix A. The system proposed here does not address this attack, primarily because it will work alongside a paper system that is also susceptible to it. Our system also reveals whether a person voted ATL or BTL. This is unlikely to have political consequences.

Another challenge is producing an accesible solution for voters who cannot fill out a paper ballot unassisted. This is a primary justification for the project, but producing a truly verifiable solution for such voters is extremely difficult, because many of them cannot perform the crucial check that the printout matches their intention (though see [CHPV09] for a verifiable and accessible protocol). We provide a way for them to use any other machine in the polling place to do the check, in which case the cast-as-intended property depends upon at least one of the machines in the polling station not colluding with the others to manipulate the vote.

1.2 Related Work

In the USA, voter verifiable paper trails with auditing (VVPAT) are a common means of achieving software independence. However, this does not solve the problem of secure custody and transport of the paper trail described above. Furthermore, performing rigorous risk-limiting audits seems intractable for IRV [MRSW11, Car11], let alone for 30-candidate STV.

The most closely related project is the groundbreaking use of Scantegrity II in binding local government elections in Takoma Park, MD [CCC+10]. Our project has very similar privacy and verifiability. However, both the number of votes and the complexity of each ballot are greater for our system. Obviously we are describing a proposal for a system,

[4] These procedures are also under review and improvement, but are out of the scope of this paper.

while the Takoma Park project has already happened. Although the Scantegrity II scheme appears to have been highly successful in the context of the Takoma Park elections, we decided that Prêt à Voter was more appropriate than Scantegrity II for our application. Scantegrity II is inherently for single-candidate selections. It has been adapted to IRV in Takoma Park by running a separate single-candidate election for each preference, but would be difficult to adapt to 30-candidate preference lists. Even with computer assistance, a 30 by 30 grid of invisible ink bubbles seems too complicated for most voters.

Wombat [RTsRBN], VoteBox [SDW08] and several other polling-station end-to-end verifiable voting schemes guarantee integrity by using "Benaloh challenges," [Ben06] which require filling in the vote more than once. This would be time-consuming for 30-candidate STV. It would perhaps be possible to make challenges easier (for example, by letting the device remember the last vote), but the integrity guarantees still depend on the voter performing quite a subtle randomised protocol. We have opted for Prêt à Voter, in which voters may audit the unvoted ballot form. This audit is independent of the vote size and can be completed with assistance without compromising privacy. It also provides full accountability: there is no need to take the voter's word for how they voted.

1.3 Prior work and paper overview

In a previous paper [BCH+12] we gave an overview of this project, including the context of Victorian voting and some ideas about how we would implement the protocol.

Here we present all of the protocol, including both the cryptographic protocol and the human procedures to be followed in the polling place and at the electoral commission. Our aim is for a comprehensive analysis of the protocol's security, including the assumptions on which privacy depends, a precise explanation of the kind of verifiability achieved, and a clear statement of the issues that remain.

The next section provides a detailed protocol description, including both human and cryptographic elements; the final section contains a preliminary threat analysis. A more comprehensive and rigorous threat analysis will be performed as the details of the design, including surrounding procedures etc, are finalised.

2 Protocol description

The main roles for computers are:

- **the web bulletin board (WBB)**, a broadcast channel with memory,
- **candidate list mixers** who generate ballot information,
- **candidate list key sharers** who deliver ballot information to printers.
- **printers** who print ballots,
- **electronic ballot markers (EBMs)** who mark votes onto ballots,
- **vote mixers** who shuffle votes, and
- **election key sharers** who share the key used to decrypt votes at the end of the election.

In practice we will use the same set of servers for mixing and decryption, so the election key sharers will be the same as the vote mixers, and likewise the candidate-list key sharers will be the same as the candidate-list mixers. These roles are explained below.

2.1 Ballot Form

The ballot form is a 240×120mm card with a perforation down the middle. The front face (as shown in Figure 1) lists the LA section as well as the LC-ATL section, while the back face (shown in Figure 3 of Appendix A) lists the LC-BTL section. The ballot on each face is upside down with respect to the other. On both faces, the right hand side (RHS) lists the candidate/party names and the left hand side (LHS) allows the voter to mark her choices. Because both the LA and the LC-BTL sections are using ranked elections, the candidate list on the ballot for these two sections needs to be randomly permuted rather than just cyclic shifted. Otherwise, the LHS with the ranking choices will reveal too much information. For example, if adversaries know a voter's most/least favorate candidate, then they can figure out this voter's other preferences.

Each ballot has a serial number on the LHS. Moreover, in the LHS of the ballot form, there is an encrypted value, called an "onion", associated with each candidate. If an onion is properly decrypted, it will represent the corresponding candidate in the RHS. However, the onions are not directly printed on the

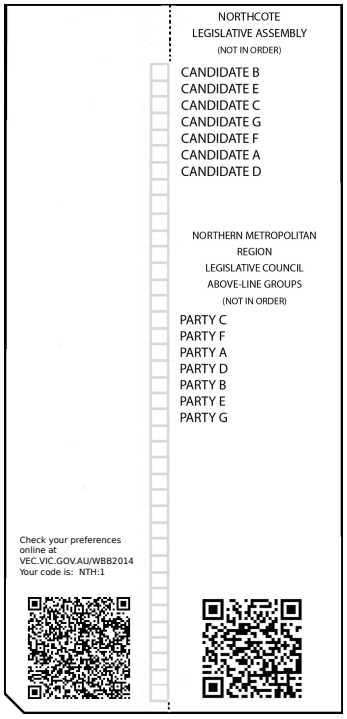

Figure 1: A slip example—the front face

ballot. Instead, they are recorded on the WBB, and there is a QR code on the LHS front face of the ballot to refer to the corresponding onions. On the front face of the ballot form, there is another QR code in the RHS which records the permutation of the candidate ordering of the entire ballot. Each QR code merely reproduces in machine-readable form exactly the information that is available in human-readable form on the same side of the ballot.

We now describe how the onions are constructed in different sections of the ballot form. In this paper, we use $\hat{E}_{pk}(m)$ to denote that m is encrypted using exponential ElGamal, and $E_{pk}(m)$ denotes that m is encrypted using normal ElGamal.

For the LA section, we use the Baudron counter [CFSY96, BFP+01, BCH+12] to encode these onions as follows: suppose there are k candidates in the LA election, we first select a value M where $M > k$ (e.g. $M = k+1$). Then we associate M^0 with the first candidate in the ballot draw order, M^1 with the second candidate, and so on. The onion for the i-th candidate will be encrypted using the exponential ElGamal cipher as $\hat{E}_{pk}(M^{i-1}) = (g^{M^{i-1}} y^r, g^r)$. This allows us to absorb all these onions as well as their associated preferential rankings into a single ciphertext using the homomorphic property. Hence it will speed up the tallying process.

For the LC-ATL section, we simply select a value in G_q to represent each party/group name, and the onion is encrypted using the ElGamal cipher $E_{pk}(m) = (my^r, g^r)$. For example, if α, β, γ are values in G_q to represent the parties $\mathcal{A}, \mathcal{B}, \mathcal{C}$ respectively, their corresponding onions will be $E_{pk}(\alpha), E_{pk}(\beta), E_{pk}(\gamma)$.

For the LC-BTL section, we use the Baudron counter again to encode the onions. Suppose there are l candidates in the LC-BTL section, we select a value L where $L > l$ (e.g. $L = l + 1$). Then we associate L^0 with the first candidate in the ballot draw order, L^1 with the second candidate, and so on. The onion for the j-th candidate will be encrypted as $\hat{E}_{pk}(L^{j-1}) = (g^{L^{j-1}} y^r, g^r)$. However, we will show in a later section that the tallying method for this section has to be slightly different.

2.2 Ballot Generation

In Prêt à Voter, privacy depends on maintaining the secrecy of the candidate order that corresponds to a particular receipt. Since a printer actually has to print both sides of the form, and hence can recognise the receipt subsequently and recall the candidate order, privacy depends on very strong assumptions about the printer's data being properly generated and destroyed. We emphasise that this affects privacy, not integrity, because the correctness of printing can be audited.

Ballot generation must satisfy two main requirements:

- The ballot's candidate ordering and the values used for encryption must be random and not generated by any single party. (Otherwise a malicious printer can use the receipt to leak information about the votes via a kleptographic attack [GKK+06].)
- As much as feasible, the ballot's random data, and the plaintext candidate list corresponding to each RHS, must be secret.

We will use the distributed ballot generation of [RT10], in which the candidate list mixers successively shuffle a list of encrypted candidate names for each vote. This protocol guarantees the first condition above if at least one participant is honest. This produces a list of encrypted ballots on the WBB, each one consisting of a serial number, the list of

LHS onions, and the corresponding list of encrypted candidates for the printer together with a proof of correspondence. The printers' candidate lists are encrypted under a threshold key shared across a set of candidate list key sharers, distinct from the election key sharers. Thus no individual server can obtain the cleartext candidate lists.

It is not strictly necessary to make the election key sharers and candidate list key sharers distinct. Note however that there are two distinct modes of operation here: threshold decryption for ballot printing and audit (on a per ballot basis) on the one hand, and tabulation mixing/decryption on the other (a batch operation). Even though the capabilities required are the same, it seems wise to assign these to separate entities to make mode confusion less likely.

Although some possibilities exist for distributing the printing step [ECHA09], these are not mature enough to use yet.

Printers obtain the candidate list by means of blinding [Cha82], using a protocol similar to that of [ZMSR05] but with the printer generating the blinding factor. We outline it as follows: to obtain a list for printing, the printer encrypts a blinding factor under the candidate list public key, and passes it, with a proof of knowledge, to the list server which combines it with the encrypted list. The resulting blinded list is decrypted and passed back to the printer, which removes the blinding factor to obtain and print the candidate list.

2.3 The Voting Ceremony

2.3.1 Casting a vote

The voter presents herself to an official at a polling station and her name is marked off a register. The official sends the print station a request for a ballot of the appropriate LA and LC division. The print station runs the protocol with the WBB to retrieve the candidate permutation for the assigned look-up code. It then prints the ballot. It is important that no-one except the voter sees the association between the candidate order and serial number on the ballot, so printing should be private.

Check 1: Confirming ballot correctness. Once she has obtained her ballot, she should decide whether she wishes to run a confirmation check on it or use it to vote. A confirmation check, called "auditing" in previous versions of Prêt à Voter, means checking that the encrypted list of candidates on the WBB matches the plaintext candidate ordering on the RHS of the ballot. In our version the QR codes must be checked too. Ballot confirmation ensures that the ballot is well-formed and hence would correctly encode a vote. We describe the ballot confirmation procedure below in Section 2.3.2. She can repeat the ballot confirmation procedure as many times as she wants, each time obtaining a fresh ballot, until proceeding to vote using the last obtained, unaudited ballot. This implements an iterated cut-and-choose protocol: not knowing which option the voter will choose before committing to the printed ballot serves to counter any attempts by the system to manipulate votes by issuing mal-formed ballots. Confirming ballot construction necessarily reveals encryption information, so a ballot that has been confirmed should not be subsequently re-used for voting.

Assuming that she is now happy to proceed to casting her vote, the voter takes the last obtained ballot to the booth. In standard Prêt à Voter she would now proceed to fill in her preferences directly on the ballot. However, given that the LC-BTL section contains about 30 candidates, it is not reasonable to expect the voter to enter her ranking preferences using a permuted candidate list. Instead we propose to use a touch screen Electronic Ballot Marker (EBM) that will display the candidates in standard order, as previously introduced in [BCH$^+$12]. The voter enters her preferences via the screen in the standard way, the EBM will take care of the permutation as we will see shortly. This means that we have to sacrifice one of the pleasing features of standard Prêt à Voter: that no device directly learns the voter's choices. This seems unavoidable for such expressive ballots if the system is to be usable.

She inserts the ballot into the EBM and selects her preferred language and can run through a training module on the machine to learn about the whole voting procedure, verification and tallying. The voter is now offered the choice of sequence in which she votes that is, the Legislative Assembly (LA) or Legislative Council ballots, and for the latter she can vote either "above the line" (ATL) or "below the line" (BTL). For the LA ballot, there is also a "how to vote card" option if she wishes.[5] Note that although the voter can vote at any polling station, the LA ballot is specific to the region in which she is registered. She must however, fill in both a LA and LC ballot and will be prompted by the EBM to ensure that she does this.[6]

[5]This is an opportunity for candidates to give their supporters helpful recommendations on how to arrange their other preferences.

[6]Exact rules on ballot spoiling are yet to be specified. This

For each ballot (LA or LC), the EBM scans the the QR code which represents the permutation of the candidate ordering on her ballot and displays the candidates in canonical order. Once the voter enters her choices, she is asked to confirm her choices and when she does so they are overprinted on the LHS of the ballot. Note that the EBM knows the permutation on the ballot and so re-orders the voter's selection accordingly.

Check 2: EBM vote printing. The voter should check that the overprinting matches the preferences she told the EBM. Note also that the EBM can assist the voter by pointing out syntactic errors, for example, duplicate rankings etc.

Once both the LA and LC ballots are complete, the voter extracts the ballot from the EBM and separates the left and right sides of the ballot. To ensure receipt-freeness, she must insert the RHS in a disposal bin as without the candidate order on her ballot, she cannot later prove how she voted.

The voter takes the LHS to the scanner and scans both faces. The scanner submits the preferences and the look-up QR code to the WBB, which registers the vote, generates a hash value of the received information and sends the digital signature of the hash value back to the scanner. The scanner now overprints the signed hash as a further QR code onto the LHS, which can be taken away by the voter as her receipt.

Check 3: Signature. The voter can check the signature using a purpose-built smart phone app or on-site services provided by helper organisations. This must of course incorporate a check that the data signed by the WBB is the same as the data printed on the paper.

The voter is easily able to produce multiple copies of her receipt, either using a photocopier or because they are automatically printed by the scanner. This combats the "trash attack," [BL11] and also allows others to check her recipt on the WBB.

Check 4: Receipt appears on WBB. After a given time period, the voter can use her receipt to check that the information is correctly recorded on the WBB.

is a matter of user interface: voters could easily be allowed to cast incomplete or invalid preference lists, as long as they are warned. The receipts would then reveal their decision to spoil their ballot. Alternatively, there could be a candidate called "spoiled ballot" who would be the first preference of any invalid ballot. Subsequent preferences would be meaningless, but could be filled in to make the receipt look like that of a valid vote.

We now describe the ballot confirmation process in more detail.

2.3.2 Confirming ballot correctness

To perform Check 1, confirming ballot correctness, the ballot can be taken back to the printer.[7] The candidate list key-sharers are asked to decrypt the candidate list directly, and to publish a proof of decryption. The WBB must also be notified that the ballot has been audited, and therefore not to accept any vote cast with that ballot form. As part of the confirmation process, a clear "AUDITED—NOT TO BE USED TO VOTE" message (which must be visible on the LHS) is printed on the ballot form.

The voter can also check the proof of decryption later on any other machine, including at home, so we are not trusting the polling-place machines for confirmation of ballot construction.

When the day's bulletin board becomes available (see Section 2.5), it shows which serial numbers were audited and displays a proof of what the candidate ordering should be. (It also shows which ones were voted and what the preferences were.)

Ensuring the mutual exclusion of audited and cast ballots is vitally important. There must be a realtime check that the same ballot is not both audited and voted. The entity that does this (which could be the printer, or the tellers, or the WBB, depending on exactly how the CTs are opened) is trusted for privacy, but not for integrity because violations are detectable.

2.3.3 Resolving disputes

We have not specified exactly what happens when any of these checks fail. It is challenging in any voting system to recover from errors. A failure of check 1, 3 or 4 is immediately demonstrable (assuming Check 3 is performed on the spot in the polling place) and proves malfeasance by election authorities. This would have serious implications for the trustworthiness of the election result. It is less clear how seriously to regard a failure of Check 2. Unfortunately there will be some rate of false alarms, in which voters claim their vote was misrecorded when they simply misremembered it or changed their minds. Hence a zero-tolerance policy is unworkable, even though any tolerance increases the chances for vote manipulation. Whatever the level of tolerance, it is important that

[7] A scanner with printing facilities would also work.

ballots spoiled in this way remain secret, or the process can introduce opportunities for coercion.

2.4 The Voting Ceremony—the case for vision impaired voters

2.4.1 Casting a vote

We assume that the vision-impaired voter has registered at a polling place and had her name marked off. The voter should be able to print her ballot by following the procedure documented in the previous section, assuming that the printing device is accessible. As before, printing needs to be private to ensure confidentiality of the RHS. Notice (Figure 1) that the slip has a clipped corner on the lower LHS. This is to assist correct insertion into the device for overprinting, as described shortly.

The vision impaired voter takes the slip to an Electronic Ballot Marker (EBM). At the EBM, she inserts the slip, clipped corner first. The system is set so that she has an audio-only session in her preferred language and the touch screen is laid out like a keypad. For example, the four corners when touched render 1, 3, $*$ and #, the middle top and bottom give 2 and 0, and so on.

The session is similar to the one described previously in Section 2.3 in that the voter has to fill in ballots for her LA and LC (ATL/BTL) votes, but this time she indicates her choices by touching the appropriate parts of the screen and has voice prompts to guide her. When she had filled in all required parts of the slip, she is given a voice confirmation of her vote choices and if she agrees with them, she can finish the voting part of the ceremony by touching the designated part of the screen.

She then inserts the form into the device which overprints her choices on the LHS of the slip.

This voter is unable to perform by sight the crucial check that the overprinted values match her intended vote. Hence she may take her form (with both sides still joined) to another EBM[8] which scans the RHS QR code and the printed preferences, and reads her vote back to her. In this case we are trusting for integrity that she can find at least one EBM in the polling place that does not collude with the first one she used.

By this point we can be confident that the printed preferences match the voter's intentions. She then separates the two sides of the slip down the lengthwise perforation and destroys the RHS. As before, the LHS is scanned and the WBB returns a signed hash of the vote information which is printed on the receipt.

The EBMs could also speak the preference orders on the slip so the voter can note them down (with a blind note-taker device or with memory), and likewise the ScanStation could speak the numbers it reads and submits. This helps the voter to check the ScanStation unassisted but does not really affect privacy or verifiability because she must still check that her vote is printed as she requested, and recorded on the WBB as it is printed, rather than trusting the EBM and scanner to tell her the truth. She could do the WBB check with assistance from a print reader or from a sighted person without jeopardising privacy.

Note that the only steps that need to be private are the mark-up by the EBM and check with a second EBM. All the other verification steps: confirmation of the ballot, confirmation of the receipt signature and of correct posting of the receipt to the WBB can be performed with assistance without jeopardising ballot privacy.

If she has performed a confirmation check on a ballot, the voter can still go home and use her screen- or print-reader, with the same confirmation-checking software as everyone else, to make sure her candidate list matches the onion. The only important detail is that she has to make sure she knows what the cleartext candidate order is. She must either ask several people or use (a) print reader(s). Neither of these impacts upon privacy: there are no privacy implications for anyone in confirming ballots.

2.5 The Web Bulletin Board

A number of voting schemes require some form of append-only Web Bulletin Board (WBB). However, specific details of how to design or implement such a service are often lacking. In this section we do not aim to propose a generic WBB, only to define one that will work within the constraints we have and offer the properties we need. We will approach the problem from a pragmatic point of view and rather than try and define a service that provides the properties itself, we will define a process that provides the

[8]This represents the ideal scenario. In practice it may be too hard to equip every EBM with an OCR enabled scanner, so we may have to either use the ballot-submission scanner for this purpose, or use a dedicated machine. Using the ballot-submission scanner is not ideal because it is online, so should preferably not learn the contents of votes. Of course, the voter could also ask for assistance from a sighted person, but that would compromise her privacy and defeat most of the purpose of using this system.

same assurances. The fundamental requirements we have of a WBB are

- that every observer gets the same information, and
- that the data written to it cannot be changed or deleted without detection.

We assume that when the voter submits their choices to the WBB, the WBB creates a digital signature of the choices and returns it to the voter. The voter then checks the validity of the signature and that the contents match the choices that were submitted. Initially we will only discuss a single WBB, we will discuss later how this creates a robustness assumption, but does not directly impact on integrity. Unlike some previous proposals for a WBB, the WBB we utilise will not be accessible to the public during the run of the election. At the close of the election the WBB commits to its contents by constructing a hash of it and digitally signing it. This hash is then distributed to public organisations and media outlets. At this point all the data is also accessible and those organisations are free to verify that the hash is correct and the signature is valid. Likewise, a voter is free to perform those checks should they wish to. Following this commitment the WBB becomes accessible to the public for them to verify the receipts they have and check that their votes are included in the hash.

If an adversary, or even the WBB itself, attempts to delete or modify the data following the commitment it will be detected when the hash values do not match. If an attempt is made to delete or modify the data prior to the commitment it will be detected when the voters verify their receipts on the WBB. A key point here is that information cannot be removed from the WBB after a voter has checked that it is present. It is worthwhile noting that this assumes a significant enough number of people do verify their receipts. We do not have to trust the WBB for integrity or privacy. We do have an assumption that the WBB is robust. The loss of data would potentially result in the election having to be re-run.

Due to the nature of the election, and the early voting phase lasting two weeks, we need to address the practicality of the WBB running for that period and the reasonableness of asking voters to wait two weeks to check their vote on the WBB. To mitigate against this problem we will make a commitment of the contents of the WBB on a nightly basis. This will allow the WBB to potentially be shutdown and maintained overnight when no voting is taking place. It additionally allows voters to check the presence of their vote, in the committed list, at most 24 hours after they cast their vote. From an abstract point of view this can be thought of as if we have an individual WBB for each day and the votes submitted to the mix-net are the combination of all the WBBs, which can be verified via the previously made commitments.

Similarly, it may be impractical to expect voters to download the entire WBB contents in order to recompute the hash. This could easily be addressed by generating a hash tree and publishing the root, then giving each individual their list of neighbours. They would recompute the hashes along their branch and this would demonstrate the inclusion of their vote.

The WBB will be in operation prior to the start of voting, in order for generated ciphers and various audit data to be submitted and stored on it in preparation for the election. It seems sensible for the WBB to make an initial commitment, prior to the start of the election, which can then be used to verify that the lookup data and ciphers have remained on the WBB throughout.

2.5.1 Further work

As we mentioned above, with a single WBB we are making a robustness assumption and are particularly dependent on voters checking their receipts on the WBB, neither of which is ideal. We have investigated a number of different approaches and present one possible approach here. To improve the robustness we could construct a distributed, peered, WBB. It will operate almost identically to the single WBB except the digital signature will be a threshold signature. When a vote is submitted to the WBB it is sent to all peers simultaneously. Each peer constructs its individual share of the signature and distributes it to the other peers. Once a peer has the threshold number of signature shares it can combine them and return it to the voter on their receipt. This improves the robustness assumption, since we no longer need to assume a single machine remains online and intact, we just need to assume a threshold number of peers remain intact. Due to the redundancy in the data it may also be possible to bring a peer back online following an attack or failure and for that peer to be able to verify it has a valid copy of the data.

Whilst having a peered WBB does not directly change the integrity assumptions, it may impact on the assumptions the voter ends up having. To this degree the assumptions the voter has to make reduce, the more work they do:

1. If the voter casts their ballot and does not check the receipt or signature they have an integrity assumption on the equipment in the polling station and the WBB.

2. If the voter verifies their receipt in the polling station they have an integrity assumption on the WBB.

3. If the voter checks the WBB later on they are free from any integrity assumption.

The obvious question is whether voters would undertake step 3 in significant numbers or not. As such, we can improve the assumption in step 2 by distributing the trust the voter has in the WBB amongst multiple peers. So from a practical point of view there may be a security benefit to peering the WBB, purely because voters are unlikely to utilise the full verifiability on offer.

2.6 Vote tabulation, decryption and export

Votes are marked on the WBB with the day and polling place they were cast (or confirmed). Poll workers must check that the number of votes recorded each day matches the number of people who submitted ballots to be scanned.

After all votes are received, there will be two types of votes on the WBB: one containing rankings in the LA section and a single choice in the LC-ATL section, and the other containing rankings in both the LA section and the LC-BTL section. For all these votes, the LA section will be tallied together, but the ATL votes and BTL votes will be tallied separately.

2.6.1 LA + LC-ATL pre-processing

We take the approach described in [BCH+12] to processing the votes. The first type of votes are illustrated in Figure 2, where $r_1, r_2, \ldots, r_k$ are the ranking preferences in the LA section (note that the columns might be in different orders, but the tallying method will not be affected):

For the above received vote, the onions and their corresponding rankings in the LA section can be packed into a single ciphertext using the homomorphic property as:

$$\hat{E}_{pk}(m) = \prod_{i=1}^{k} \hat{E}_{pk}(M^{i-1})^{r_i}$$

where $m = \sum_{i=1}^{k}(r_i \times M^{i-1})$.

Moreover, for the LC-ATL section, the onion next to the voter's mark $E_{pk}(\beta)$ will be picked out.

LC-BTL votes are preprocessed similarly to LA votes, except that instead of turning 38 preferences into one ciphertext we pack the first t preferences into one ciphertext, then the next t in to a second ciphertext and so on. Using $t = 6$ is about the right tradeoff between reducing data size and reducing discrete log (dlog) lookup time.

2.6.2 Mixing and tallying the votes

Each type of vote (LA, LC-ATL and LC-BTL) is mixed separately. Unpacking is done by dlog computation based on a lookup table. LA votes have at most 10 candidates, so the maximum size of the LA look-up table is $10! \approx 2^{22}$ which is perfectly reasonable. More than 90% of LC votes are ATL, which is simply decrypted without any need for a dlog. For BTL votes, if there are 38 candidates and we pack every 6 onions into a single ciphertext, the size of the look-up table is $P_6^{38} = 38!/(38-6)! \approx 2^{30}$. This is feasible—see Section 2.7 for an estimate of how long it takes.

2.7 Timing analysis

At the core of the implementation is the Verificatum Mix-net [Wik12] developed by Douglas Wikstrom. Verificatum offers both fast and optimised mixing and decryption as well as efficient proofs. The details of this implementation and the interface between Verificatum and Prêt à Voter are described in Appendix B. Here we give an estimate of the computation time required for mixing and decrypting the votes. The estimate excludes ballot generation, but includes setting up the mixnet, generating the distributed key shares, mixing the votes, and distributed decrypting with discrete log lookup. We assume one mixnet for each of LA, LC-ATL and LC-BTL, with LC-BTL votes packed as described in Section 2.6.

The timings provided are based on the demo mixnet settings provided in Verificatum. As such, this consists of three mix servers with a threshold of two. The threshold impacts on both the mixing and decryption phase. For decryption it refers to the minimum number of mix servers that must be online in order to successfully decrypt. If all mix servers are online then all mix servers will be used during decryption. For mixing this refers to the number of mix servers that will perform an actual mix. Thus,

	LA				LC-ATL		
Onion	$\hat{E}_{pk}(M^0)$	$\hat{E}_{pk}(M^1)$	...	$\hat{E}_{pk}(M^{k-1})$	$E_{pk}(\alpha)$	$E_{pk}(\beta)$	$E_{pk}(\gamma)$
Marking	r_1	r_2	...	r_k		X	

Figure 2: LA + LC-ATL votes

the first two mix servers will perform a mix, but the third will not. Therefore the timings given below for mixing are for two mix servers, whilst the decryption is for three. In practice we are likely to use more mix servers and a higher threshold.

These estimates have been extrapolated from our testing on Intel Core i7 machines with 8GB RAM and 1TB Hard Disks. The timings are meant as a guide rather than scientifically significant evaluations of time complexity.

The timings are based on a hypothetical district of 100,000 votes. Most of the timings scale linearly as the size of the election changes. Where that is not the case we will explicitly state it. Unless otherwise stated the timings are given in milliseconds. We have estimated an election with 38 candidates and 8 parties, with 10% of votes being BTL and 90% ATL. These details are summarised in Table 1.

Table 1: Scale of example election

Estimate Australian Election	
Number of candidates	38
Number of parties	8
Number of ballots	100000
Number of ATL Votes	90000
Number of BTL Votes	10000

Table 2: Timings for example election

	100,000 Ballots	**Time in Hours**
Cipher Generation	142481320	39:34.41
MixingATL	7200000	2
DecryptionATL	729000	0:12.9
MixingBTL (packing of 6)	5600000	1:33.20
DecryptionBTL	567000	0:9.27
DL Lookup	3430000	0:57.10

Table 2 shows the timings for our example election. The time for Cipher Generation is clearly the largest. One of the reasons this is so large is the requirement to generate enough BTL ciphers for everyone to vote BTL if they wish. We pre-compute the ciphers and commit them to the WBB before the start of the election. As such, we cannot generate new ballots on the fly. This dramatically increases the amount of encryption that is required during the cipher generation because we are producing 46 ciphers per ballot.

It is clear to see that the mixing and decryption of the ATL vote is quite quick. The mixing of the BTL votes looks equally efficient, however, it should be remembered that we assume only 10% of the vote will be BTL. The efficiency saving of packing the votes is a factor of 6, without which it would be infeasible to handle such a large number of candidates in the required time. The DL Lookup (Discrete Log Lookup) is extremely quick. This is based on looking up a precomputed and sorted table. The demo table handles a packing of 6 ciphers in 38 and took approximately 15 hours to generate and sort, and is 29GB in size. The fast lookup time is achieved through optimised memory and disk caching. We believe the time could improved even further were we to use Solid State Drives instead of standard hard disks. The lookup can also be performed in parallel and distributed across multiple machines if required.

3 Security Analysis

For reasons of space we focus here on the principal threats, particularly with respect to the key requirements of integrity and privacy.

3.1 Security Guarantees of the protocol

3.1.1 Integrity for sighted voters

The protocol makes no trust assumptions for integrity, apart from trusting that each eligible voter is allowed to cast at most one vote, and that only eligible voters can vote. It does of course rely on voters to perform some checks, which are detailed in Section 2.3. Invalid ballots, in which the candidate list

doesn't match the onion, are detected at ballot confirmation by Check 1. Check 2 detects incorrect vote printing by the EBM. Vote substitution by the scanner before WBB submission is detected by Check 3. Check 4 detects vote substitution by the WBB. Incorrect mixing or decryption would be detected because the proofs of correct mixing and decryption are public.

3.1.2 Receipt freeness and privacy

Privacy depends on the assumption that at least one mix server generates randomness correctly and keeps it secret. This applies to randomness used in both ballot construction and tallying. Further, that a threshold set of those who share the keys is honest.

Provided that the two assumptions hold, the system has some defence against kleptographic attacks on the receipt [GKK+06]. This is because the receipt's random data is generated in a distributed way, and the entities that do the printing (the printer and the EBM) are deterministic. Thus information cannot be leaked in the ballot data itself, though it could be subtly leaked in slight font changes or other printing effects.

There is privacy of the contents of each receipt, meaning that the tallying protocol does not add any information about the link between a receipt and its vote, except whether it was ATL or BTL. The system is also receipt-free, meaning that the receipt itself does not allow a person to prove how she voted.

(However, there are coercion attacks on this protocol, including the "Italian attack," which are described in Section 3.3).

3.1.3 Integrity in the case of vision-impaired voters

The vision-impaired voter is unable to do Check 2, that the EBM printed the correct ballot. She cannot ask for human assistance without destroying privacy. This leads to a distribution of trust over the machines in the polling place: she can check her vote on as many machines as she likes, and must assume that at least one of the machines she uses is honest.

3.2 Threats Ameliorated By Procedural Controls

3.2.1 Integrity

There is potential for ballot stuffing by authenticated parties. For example, the scanner and the WBB can both submit ballots that did not originate with voters. This is mitigated by the pollworkers reconciling the WBB against the list of attendees as described in Section 2.6.

Procedures must prevent voters from taking someone else's ballot off the printer and hence voting in the wrong division.

3.2.2 Privacy

As the voter inputs her choices into the EBM, the device necessarily "learns" how she voted. The potential for the EBM to leak vote information clearly raises privacy issues.

Possible countermeasures are to ensure that the EBM is "stand alone" and offline at least during the voting phase and is therefore unable to communicate vote information to other colluding entities. Any data stored in the EBM's memory should be deleted, ideally after each session, but at least before the EBM goes back on-line if this is required for any reason.

Prêt à Voter introduces a privacy threat that does not exist for either standard paper voting or for DRE's with VVPAT: someone may discover and record an unvoted ballot's candidate order and look up code, then learn the vote choices when they are later posted on the WBB. Therefore there should be procedural controls to protect both the paper printout and the electronic data on the printer from observation by anyone but the voter.

The threat of using the confirmation process to expose the contents of a ballot that has been voted on is ameliorated by the electronic locking process described in Sec 2.3.2.

Note that the EBM does not learn the look up code or the onion, so it cannot align votes with receipts unless the preference ordering is unique (which unfortunately, with 30 candidates is possible).

As voters may ask for assistance at any point during the ceremony, there is still a chance that an official may make a connection between candidate order and look-up code. This threat to privacy however, exists in the current system.

3.3 Threats Remaining: general voting ceremony

We now discuss remaining threats to the system.

3.3.1 The general voting ceremony

An analysis of the system was carried out by a walk-through, assessing the potential for known threats against actions required of the voter, officials and machinery and the interactions between them [KSW05, RP05].

There are possible threats to coercion, such as forced randomisation, where a coercer demands the ballot to be filled out in a particular order, effectively producing a random vote. For FPTP elections, the voter could keep confirming ballots until she finds one that satisfies the coercer, at the same time allowing her to vote as she wishes. This is infeasible in complex voting systems such as this, though it might let the voter choose her first preference. Coercion to vote above the line is possible, but unlikely to have any political consequences. We have already mentioned the"Italian attack". Although this exists in the current system, it is perhaps exacerbated by publishing the vote, rather than releasing it only to a small number of trusted organisations, which is current practice.

The possibility exists for an official to learn voter's choice if she asks for assistance in the booth.

If the voter leaves the polling place with the RHS of her voting slip, she could prove how she voted. This problem in Prêt à Voter has already been noted. Unfortunately, having dummy and discarded RHS freely available at the polling station as previously suggested, will not work here. With preference voting, especially with a large number of preferences and if the attacker demands an unusual ranking, there is only a tiny chance that a coerced voter will find a suitable RHS among the dummy/discarded slips.

It is important that destruction of the RHS is enforced, and that the voter does this before she leaves the privacy of the booth. Even then, there is still a possibility she may capture her RHS on a mobile device.

A possible alternative is to provide facilities for voters to generate "pseudo" RHS's. A coercer would not then be able to rely on the voter's complicity. Admittedly, this adds further complexity to the system, and is not being actively considered for the VEC system.

"Psychological" attacks are a potential threat. As an example, a coercer manages to convince voters that he is able to decrypt their receipts and find out how they voted [RP05]. Voter education could mitigate this attack; however psychological attacks will be a problem for virtually any end-to-end verifiable system.

Chain voting is a possibility in Prêt à Voter, including this version. This problem exists already in the ordinary paper voting system, and is not regarded by the VEC as a significant threat.

3.3.2 The case of the vision impaired voter

In addition to the previously noted threats, a vision impaired voter may be vulnerable to an eavesdropper learning her choice if the audio feature is faulty/corrupt. There is a greater risk of an official learning the voter's choice. Arguably, a vision impaired voter is more likely to ask for assistance during the ceremony. There is greater trust in the machinery performing as intended than for sighted voters. A blind voter will be relying on voice prompts to perform the ceremony correctly. There are no easy solutions to these problems and again, security has to be offset against system requirements.

3.4 Other issues

We finally draw attention to other positive properties of the system.

There is robustness against less than a threshold of authorities stopping the protocol. This ensures that a result will always be output. The mix servers do not need any secret information to do the shuffle. Hence if some of them are absent or refuse to perform the shuffle, they can simply be replaced or even ignored. The threshold of honest tellers ensures decryption will always be properly carried out. The robustness assumption for the WBB should also be addressed by distributed implementation.

There is accountability for receipt misrecording. Voter checking of the signature on their receipts in the polling place should detect malicious behaviour on the part of the scanner or WBB.

Finally, the system provides evidence of malfeasance: if the voter has a properly-signed receipt that does not appear on the WBB, she can prove it.

4 Conclusion

One lesson from our attempt to adapt Prêt à Voter to a real election is that not all issues can be perfectly addressed in a way that retains usability and computational feasibility. This system has unconditional integrity but does introduce some coercion possibilities that do not exist for paper voting. The design

problem is to identify and address the issues that really are important and easy enough to address; the political problem is to maintain honesty about the ones that remain.

Acknowledgements Thanks to Jason White, Jeremy Clark, and the reviewers for many helpful comments. Thanks are also due to Douglas Wikström for his support and advice with respect to the Verificatum system. We are grateful to the FNR Luxembourg for funding the SeRTVS project, and to the EPSRC for funding through the Trustworthy Voting Systems project under grant EP/G025797/1.

References

[BCH+12] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, and Zhe Xia. A supervised verifiable voting protocol for the victorian electoral commission. In *Proc. 5th International Conference on Electronic Voting*, 2012.

[Ben06] Josh Benaloh. Simple verifiable elections. In *Proc. 1st USENIX Accurate Electronic Voting Technology Workshop*, 2006.

[BFP+01] O. Baudron, P.-A. Fouque, D. Pontecheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *Symposium on Principles of Distributed Computing*, pages 274–283. ACM, 2001.

[BL11] Josh Benaloh and Eric Lazarus. The trash attack: An attack on verifiable voting systems and a simple mitigation. Technical Report MSR-TR-2011-115, Microsoft, 2011.

[BMN+09] Josh Benaloh, Tal Moran, Lee Naish, Kim Ramchen, and Vanessa Teague. Shuffle-sum: coercion-resistant verifiable tallying for STV voting. *IEEE Transactions on Information Forensics and Security*, 4(4):685–698, 2009.

[Car11] David Cary. Estimating the margin of victory for instant-runoff voting. In *USENIX Accurate Electronic Voting Technology Workshop Workshop on Trustworthy Elections*, 2011.

[CCC+10] 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. Scantegrity ii municipal election at takoma park: The first e2e binding governmental election with ballot privacy. In *Proc. USENIX Security*, 2010.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT*, pages 72–83, 1996.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[CHPV09] David Chaum, Benjamin Hosp, Stefan Popoveniuc, and Poorvi L. Vora. Accessible voter-verifiability. *Cryptologia*, 33(3):283–291, 2009.

[CRS05] D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[DC07] Roberto Di Cosmo. On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack, 2007.

[ECHA09] Aleks Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. How to print a secret. In *HotSec*, 2009.

[GKK+06] Marcin Gogolewski, Marek Klonowski, Przemyslaw Kubiak, Miroslaw Kutylowski, Anna Lauks, and Filip Zagórski. Kleptographic attacks on e-voting schemes. In *ETRICS*, pages 494–508, 2006.

[Hea07] J. Heather. Implementing stv securely in prêt à voter, 2007. Computer Security Foundations 2007.

[KSW05] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, number 3444 in Lecture

Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.

[MRSW11] Thomas R. Magrino, Ronald L. Rivest, Emily Shen, and David Wagner. Computing the margin of victory in IRV elections. In *USENIX Accurate Electronic Voting Technology Workshop Workshop on Trustworthy Elections*, 2011.

[RBH+09] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.

[RP05] P.Y.A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.

[RS06] P.Y.A. Ryan and S. Schneider. Prêt à Voter with Re-encryption Mixes. In *European Symposium on Research in Computer Security*, number 4189 in Lecture Notes in Computer Science. Springer-Verlag, 2006.

[RT10] Kim Ramchen and Vanessa Teague. Parallel shuffling and its application to prêt à Voter. In *Proc. USENIX Accurate Electronic Voting Technology Workshop*, 2010.

[RTsRBN] Alon Rosen, Amnon Ta-shma, Ben Riva, and Jonathan (Yoni) Ben-Nun. Wombat voting system.

[Rya08] Peter Y. A. Ryan. Prêt à Voter with paillier encryption. *Mathematical and Computer Modelling*, 48(9-10):1646–1662, November 2008.

[SDW08] Daniel R. Sandler, Kyle Derr, and Dan S. Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *Proc. 17th USENIX Security Symposium*, 2008.

[Wen10] Roland Wen. *Online Elections in Terra Australis*. PhD thesis, School of Computer Science and Engineering, The University of New South Wales, 2010.

[Wik12] Douglas Wikström. Verificatum, 2012. http://www.verificatum.org/verificatum/.

[XCH+10] Zhe Xia, Chris Culnane, James Heather, Hugo Jonker, Peter Y. A. Ryan, Steve A. Schneider, and Sriramkrishnan Srinivasan. Versatile prêt à voter: Handling multiple election methods with a unified interface. In *INDOCRYPT*, pages 98–114, 2010.

[ZMSR05] Lidong Zhou, Michael A. Marsh, Fred B. Schneider, and Anna Redz. Distributed blinding for distributed elgamal re-encryption. In *ICDCS*, pages 815–824, 2005.

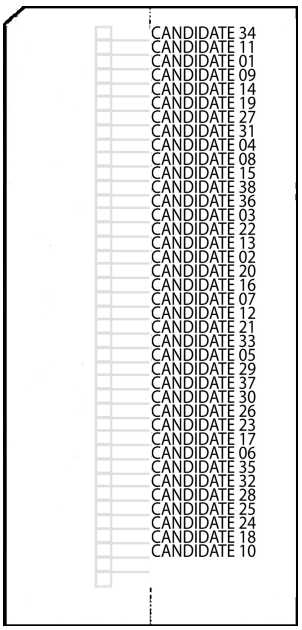

Figure 3: A slip example—the back face

A Signature Attacks

Here we explain signature attacks (a.k.a. "Italian Attacks"), why we did not address them in this project, and how we could do so in future. Any election with a complex ballot is vulnerable. In preferential elections, a coercer can instruct a voter to cast a particular signature vote (i.e. a particular permutation of candidates) and then check to see whether it appears in the final tally. Since the number of possible votes

(around 30! in this case) is much greater than the number of votes actually cast, a coerced vote is unlikely to appear unless the voter obeys. The system described in this paper ends with complete decryption of all the Prêt à Voter votes, and so it does not address this attack.

Victorian LC elections currently use a naive (non-cryptographic) e-counting system. First the votes are entered into the system through manual data entry. Then the counting system checks for invalid votes and counts the valid votes.

A concern with the existing e-counting system is that the result is not universally verifiable. To help address this problem election administrators are considering publishing all the vote data to facilitate public verification of the counting software. (Note this still leaves verification gaps, for instance in the data entry process and the filtering of invalid votes.) The dilemma is that publishing the vote data makes it possible to carry out signature attacks.

Recently several cryptographic STV counting schemes have been proposed to improve vote secrecy whilst also providing universal verifiability [Hea07, Wen10, BMN+09]. However their high computational and communication cost would make them infeasible for our application. Moreover the proposed schemes are for simplified versions of STV, and would need substantial changes to adapt them to the STV variant used in Victorian LC elections. Such modifications would likely reduce vote secrecy and/or increase the complexity costs. Also the schemes in the literature perform only the counting and do not process invalid votes.

So although cryptographic counting is desirable, it may not be feasible in the short term. Also, if the conventional paper votes are published, then a coercer could simply demand that coerced voters use conventional paper voting instead. Hence it seems that the most practical approach for now is to continue the use of naive e-counting, despite the risk of signature attacks. The encrypted votes cast via the e-voting system will be mixed and decrypted to publicly reveal the plaintext votes. These will then be combined with the conventional votes and fed into the e-counting system.

This issue is worth revisiting if all the LC-BTL votes are ever cast by computer, or if there is reason to believe that signature attacks are being performed.

B Implementation details

This section describes in detail the use of Verificatum. The selection of Verificatum as the mix-net influences other implementation decisions. We will use Verificatum to generate the joint public key, and corresponding private key shares. Ballot generation and collection will be performed in bespoke components as will the tallying of the decrypted votes. As such, we need to interface with Verificatum to retrieve the public key data and group description needed during the construction of the ballot ciphers and the discrete log look-up table required for efficient mixing. In this section we will provide more details about how the implementation will be undertaken and provide an estimate of timings for various different sizes of mix.

B.1 Setting up the mix-net

Prior to being able to generate the joint keys in Verificatum we have to undertake some configuration steps. This involves selection of the group we are going to be operating over, the IP address of the machines in the mix-net, relevant SSH login information and various mixing options. The process is initially performed on each individual mix server before they then run a protocol to jointly agree on the protocol properties. We will use a group with a 4096bit modulus and 256bit subgroup.

B.2 Key Generation

The keys for the mix-net will be generated using Verificatum. The Verificatum key generation produces a distributed key where each share is then threshold shared through a verifiable secret sharing scheme. Full details of the key generation are available in Verificatum, a summary is provided at http://www.verificatum.org/verificatum/prot.html. Full details of the key generation are available on Verificatum website [Wik12].

B.3 Cipher Generation and Candidate Identifiers

Having constructed a key pair in Verificatum we extract the public key for use by the Election Manager. For efficiency reasons we will construct three instances of the Verificatum mix-net. Each will be entirely independent and thus have its set of keys. The reason for doing this is to allow us to perform

one mix for the lower house, one for the upper house ATL votes and one for the upper house BTL votes.

The following stages will be performed for each of sub-elections. Where a different process is followed we will highlight it, otherwise the same process is performed three times. Again, for efficiency reasons these three runs of the Election Manager may be performed in parallel. The Election Manager is the component that will perform the cipher generation. The Election Manager should be run on a diskless workstation and be observed by independent observers. The first step for the Election Manager is to construct the candidate identifiers. For the ATL ciphers the Election Manager randomly selects elements from the group which the key was constructed over. These identifiers are recorded next to each candidates name. In the case where vote packing is being used, as discussed in Section 2.6, the candidate identifier is calculated based on the maximum ranking. Once the candidate identifiers have been created they are publicly committed to on the WBB.

Ballot ciphertexts are generated as described in Section 2.2. Each ballot also requires a serial number, to enable the submissions to be identified on the WBB during verification. The serial number need not be random or unpredictable. However, if the Election Manager is being run in parallel it is important that there is no overlap between the different sub-elections. As such, the serial number will consist of a prefix indicating the election that it is for (LA, LC-ATL or LC-BTL) and a sequential index. The permutation of the candidate names is sent for printing along with the serial number and a digitally signed copy of the serial number. This is signed using the Election Managers private signing key. The permuted cipher texts are committed to the WBB along with the relevant serial number. It is important to note that the digital signature of the serial number is not committed to the WBB. It should only appear on the actual printed ballot and is a safeguard against a rogue machine attempting to ballot stuff.

B.4 Vote Submission and WBB

The front-end submission of votes has already been covered in Sections 2.3 and 2.4. whilst a design for the WBB has been provided in Section 2.5. For completeness we will mention that the WBB will issue a digital signature for the submitted vote. The voter must check that both the vote preferences are correct and that the correct serial numbers are included in this signature.

B.5 Vote Verification on the WBB

The voter takes their voting receipt home and can then check the presence of their vote on the WBB once the WBB has committed to its contents, as discussed in Section 2.5. To check the presence of the vote the voter must enter their relevant serial number into the relevant box on the WBB screen. At this point the WBB will retrieve the submitted information and relevant signatures. It will display this information to the voter so they can both verify the presence of it and check the contents are unchanged.

B.6 WBB Export and Mixing

Prior to being able to run the votes through the Verificatum mix-net they need to be exported from the WBB. This will be a publicly available option. When exporting the ATL votes the submitted preferences are mapped to the corresponding cipher texts. The cipher texts are then ordered by preference and put into a column-wise structure. The ballots are combined in a row-wise structure. With votes that are being packed the relevant vote packing strategy must be performed and committed to publicly. This vote packing can be performed by each mix-net individually or just once centrally and committed to the WBB. Again, the ciphers for each ballot are organised into columns and each ballot is represented by a row.

Each mix-net server needs its own copy of the ciphers that are being submitted to the first mix. We assume that pre-configuration steps have already been performed by the mix-net, since these would need to have been done in order to generate the keys used during cipher generation. The mix servers then commence the mixing protocol. During the run the other mix servers check the proofs produced by each other, as such when the mixing phase completes the proofs have already been checked by the mix servers. At this stage an independent check of the proofs can take place or the decryption process can be started. The output of the decryption service is a list of either plain texts in preference order or values to lookup in the discrete log table. The output and the proof information from each mix server is committed to the WBB.