A Supervised Verifiable Voting Protocol for the Victorian Electoral Commission

Craig Burton[1], Chris Culnane[2], James Heather[2], Thea Peacock[3], Peter Y. A. Ryan[3], Steve Schneider[2], Sriramkrishnan Srinivasan[2], Vanessa Teague[4], Roland Wen[5], Zhe Xia[2]

[1]Victorian Electoral Commission,
Victoria, Australia
Craig.Burton@vec.vic.gov.au

[2]University of Surrey
Surrey, United Kingdom
{c.culnane, j.heather, s.schneider, s.srinivasan, zhe.xia}@surrey.ac.uk

[3]University of Luxembourg
Luxembourg
{thea.peacock, peter.ryan}@uni.lu

[4]The University of Melbourne
Melbourne, Australia
vjteague@unimelb.edu.au

[5]The University of New South Wales
Kensington, Australia
rolandw@cse.unsw.edu.au

Abstract: This paper describes the design of a supervised, verifiable voting protocol suitable for use for elections in the state of Victoria, Australia. We provide a brief overview of the style and nature of the elections held in Victoria and associated challenges. Our protocol, based on Prêt à Voter, presents a new ballot overprinting front-end design, which assists the voter in completing the potentially complex ballot. We also present and analyze a series of modifications to the backend that will enable it to handle the large number of candidates, 35 +, with ranking single transferable vote (STV), which some Victorian elections require. We conclude with a threat analysis of the scheme and a discussion on the impact of the modifications on the integrity and privacy assumptions of Prêt à Voter.

1 Introduction

Australian elections have distinctive features that create unique challenges for automation. Almost all elections in Australia use preferential electoral systems. Both the alternative vote (AV) and the single transferable vote (STV) are common. Preferential voting offers voters a high degree of freedom to express their choices, but at the same time preferential voting can make it hard for voters to cast binding votes, and it is prone to voter error. Unintentional numbering errors are by far the largest category of errors contributing to informal[1] ballot papers—comprising 50% of the total informal votes in the 2010 Victorian state election.

To help simplify the voting, STV elections often provide voters with the option of selecting 'group tickets', which are predetermined preferences chosen by parties. This can result in large and complex ballot papers. For example in Victorian elections, the Legislative Council ballots have had up to 38 individual candidates and 11 group tickets.

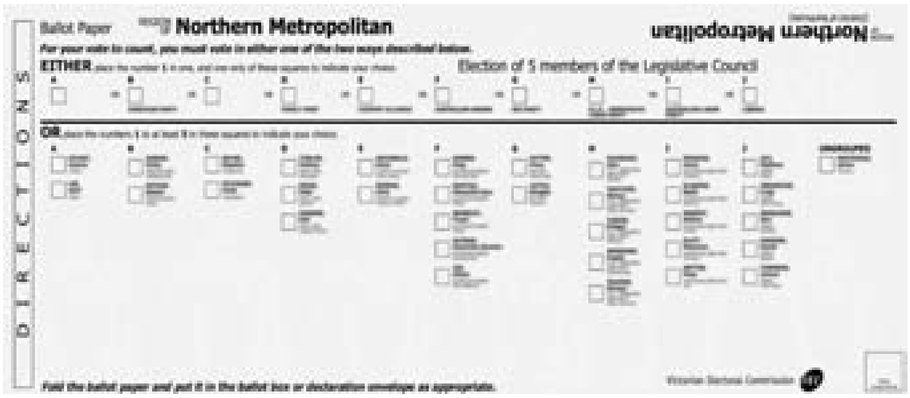
Ballot Paper
Northern Metropolitan
EITHER
Election of 5 members of the Legislative Council
OR
DIRECTIONS

Fig. 1: Ballot paper for the Victorian Legislative Council

A sample ballot is shown in Figure 1. The ballot has a top section where voters can vote for a party or group (known as voting 'above-the-line'), and a bottom section where voters can mark their preferences for individual candidates (voting 'below-the-line').

There is a very tight turnaround for printing and delivering the ballots. Candidate nominations typically close on a Friday with Early Voting commencing at 4pm the same day. Ballots must be printed, checked, and delivered as soon as possible, but no later than the following Monday morning.

Another important characteristic of Australian elections is compulsory voting. This introduces numerous logistical challenges. For example, in state elections voters can cast their votes at any polling place in their state, which means that ballot papers for every

[1] by informal we mean any vote that is incorrectly filled and/or somehow ambiguous and non-binding

electorate must be delivered to each polling place before the voting commences, and then completed ballots must be returned to their correct electorates afterwards. Polling places are also set up overseas, usually at Australian embassies.

There is a strong onus on electoral commissions to provide a high level of accessibility for all voters. The complexity of preferential ballots causes difficulties for marginalized voters, in particular for voters with a print disability and voters from non-English speaking backgrounds. Many voters in these categories require human assistance to fill out their ballots, in which case there is no protection of vote secrecy. E-voting has the potential to help solve many of these problems. Although electoral commissions in Australia have generally been cautious about e-voting, there have been strong pushes toward adopting e-voting over the last five years.

The Victorian Electoral Commission (VEC) has been one of the early adopters of e-voting in Australia. In 2006, the VEC conducted a supervised e-voting system provided by a third-party vendor, and the system was rolled out on a larger scale in 2010. The e-voting system offered several benefits for both voters and the VEC. The voting machines alerted voters to numbering errors and could provide instructions in 12 different languages. All machines were equipped with audio facilities to provide guidance and feedback to vision impaired voters. The electronic nature of the ballots helped reduce the administrative overhead and physical security risks of returning the ballots through multiple third parties (couriers for instance); the ballots were submitted to centralized servers via a private network.

However, there were a number of concerns with this system. First and foremost, the system did not provide any meaningful verifiability of the votes. In addition, the proprietary nature of the system meant that none of the design and implementation details could be made public. The necessary, heavy customization of the vendor's core product (for instance to handle preferential ballots) created difficulties in tightly integrating the e-voting system with the VEC's existing election administration process (such as allowing general staff to run the entire system), and in deriving ongoing benefit from the supplier's core solution, which is on another development branch.
To address these shortcomings, the VEC decided to develop its own e-voting system in collaboration with the e-voting community. Academics from several universities are working with the VEC to design a suitable cryptographic e-voting protocol that provides both individual and universal verifiability. The design and the final system will be publicly available for peer review. The VEC's vision is for voters to cast their votes using the machines, which will provide (optional) take-home receipts for voters to verify their votes.

One of the main challenges is in finding the right balance between usability and security, in particular requiring voters to verify large amounts of information in preferential ballots and to perform cryptographic operations such as verifying digital signatures. Our main contribution is not in the proposal of the protocol, but more importantly in highlighting the difficulties and potential trade-offs in practice when applying cryptographic voting schemes to large-scale public elections that have specific requirements.

1.1 Related Works

The present work is based on the Prêt à Voter (PaV) electronic voting system [Rya04, CRS05]. The original PaV scheme has subsequently undergone various adaptations and enhancements, some of which are described elsewhere in this paper. The basic concept remains unchanged and is described as follows.
The voter receives a printed ballot as shown in Fig. 2 below. The order of the candidates is independently randomized for each ballot and the value "7*rJ*94*K*" represents an encryption of the order on the form.

Beta	
Gamma	
Alpha	
	7*rJ*94*K*

Fig. 2: A Prêt à Voter ballot form

At the polling station, the voter is given at random a ballot sealed in an envelope. She takes this to the booth, extracts the ballot form, marks the candidate of choice, separates the right-hand and left-hand sides (RHS, LHS) and destroys the LHS. She can now leave the privacy of the booth with the RHS of the ballot form. In the presence of officials and perhaps observers, the RHS is placed under an optical reader which records the information, that is, the value at the bottom of the strip and the position marked or the preferential rankings. The RHS, or a copy thereof, is retained as a receipt. Note that as the candidate order is randomized and has been destroyed, the receipt does not reveal her vote (except to someone possessing the decryption keys). The decryption keys are shared between a set of parties such that a certain threshold set of these parties is required to perform decryption. This ensures that no single party can decrypt all the ballots. Once all voting has ceased, the receipts are posted on a secure Web Bulletin Board (WBB). Voters can use this facility to confirm that their receipts appear correctly. A set of mix servers then perform a series of robust, anonymizing, re-encryption mixes (e.g. [Nef01, FS01, Wik10]) on the receipts so that the votes can be emitted and counted.

Although seemingly simple on the surface, the underlying protocol offers many of the properties desirable in voting systems such as ballot secrecy, individual and universal verifiability, and receipt-freeness. As PaV has a certain similarity to traditional pen-and-paper, booth-based voting, the user experience is familiar, making the scheme is readily adaptable to real-world situations.
The original scheme was designed for First-Past-The-Post (FPTP) voting as currently used in the UK, but it is clear that it adapts easily to ranked, AV, etc.: the voter simply adds further marks to the ballot. However, if done naively, this opens up possibilities of "Italian"-style attacks (see page 10). This has been addressed in [TRN08, XCH10], which introduce new mixing and tallying algorithms.
Certain fielded, verifiable voting systems, such as Scantegrity II [CCC08] and Civitas [CCM08], have the potential to accommodate ranked voting. However, it is unclear how they would perform with a large number of candidates. The checkerboard-style ballots in Scantegrity II would be impractical with 35 + potential candidates. Encoding vote preferences in Civitas could incur a significant processing overhead when accounting for a sizeable candidate base. Furthermore, Civitas is a remote rather than supervised

scheme. Wombat (http://www.wombat-voting.com/) is currently implemented as an FPTP-supervised system, but again, it is unclear how it would handle a large number of ranked-vote choices. There could also be privacy issues connected to the plaintext audit trail provided by Wombat ballots.

With the PaV implementation for the VEC, we note that although workable solutions have been found for the moment, many research challenges remain. Whilst a formal security analysis has yet to be carried out, security of the scheme remains a primary concern throughout the development process and is being continuously monitored and discussed by all parties involved.

2 Front-End Design

We will now describe the proposed system.

2.1 Electronic Ballot Marking

In this section, we introduce the procedures of vote casting, in other words, how to record the voter's intent with an encrypted vote and how to verify that the encrypted vote has been correctly recorded by the election system.

Echo	θ_E
Bravo	θ_B
Alpha	θ_A
Delta	θ_D
Charlie	θ_C
$\{P\}$	

Table 1: Ballot form with voter's intent

An example ballot is shown in the above table. It contains a vertical perforation down the middle so that the two halves can be separated. The LHS lists the candidates in a random order. At the bottom of the LHS, is an unencrypted representation P of the candidate order, e.g., a computer-readable barcode. The RHS is left blank for the voter to mark her rankings. Moreover, on the RHS an encrypted value called an *onion* is associated with each candidate. If it is decrypted, its plaintext will represent the corresponding candidate in the LHS. The encoding of the onions is explained in section 3.

In contrast to the traditional PaV protocol, the voter does not mark her preferential rankings on the ballot directly. This is because the state of Victoria's upper house election contains around 36 candidates, and ranking so many candidates using a candidate list in the random order is obviously not user friendly. Instead, we will use a voting device called an *Electronic Ballot Marker* (EBM) to help the voter mark her rankings. The EBM is a standalone, isolated computer device with a barcode reader and touch screen. To cast a vote, the voter first inserts the ballot into the EBM, which will read the permutation information P in the bottom of the LHS. The EBM displays the ballot on its touch screen interface such that the candidate list is in the official draw order. The user interacts with the touch screen to give her preferential rankings. Note that the EBM can also assist the voter by pointing out an invalid vote. Once the vote is confirmed, the EBM sorts the voter's rankings according to the permutation information P and prints the results on the RHS of the ballot.

The voter takes her completed ballot paper to a scanner. As with the conventional PaV, she separates the ballot along the perforation, destroys the LHS, and then feeds the RHS into the scanner. The scanner submits the voter's preferences and onions to the WBB, which will then generate a hash value of the received information and send the digital signature of the hash value back to the scanner. The scanner would then overprint the signed hash onto the RHS, which can then be taken away by the voter as her receipt.

The voter can choose to audit either the entire vote casting procedure or just a part. Here we explain how the complete auditing process should be carried out:

- *Audit the ballot:* This audit checks whether the ballot has been correctly generated. In other words, whether each onion on the RHS correctly encrypts the corresponding candidate on the LHS and whether the permutation information P contains the correct candidate order. A ballot either be audited or cast but not both. The auditing method is the same as the traditional PaV [CRS05].
- *Audit the EBM:* The EBM transfers the voter's rankings with respect to both the candidate list in the canonical order and to the candidate list printed on the ballot. This audit checks that the transformation is done properly. For example, the voter can randomly note down some or all of the candidate-preference pairs from the EBM's touch screen surface and then compare whether these pairs are consistent with those printed on the ballot.
- *Audit the vote recording:* This audit ensures that the encrypted vote has been correctly recorded by the WBB. To perform the audit, the voter calculates a hash value of the preferences and onions in her receipt and then checks whether the signed hash from the WBB is valid.

2.2 Digital Signature Issues

One of the fundamental principles of PaV is the issuing of a receipt that the voter can use to verify that their vote has been correctly recorded onto the WBB. It is this checking that assures the voter that their vote is being included in the count. If anything is amiss, the information on the receipt is incorrect or the information is missing from the WBB altogether, the voter can challenge the authorities. As such, the veracity of the receipt is vitally important.

A valid receipt provides protection for two parties: it provides the voter with evidence to launch an appeal while simultaneously protecting the system from false accusation. It is therefore essential that any issued receipt is verified by the voter when received. If it is invalid or false, the voter must appeal at that point in time. Once the voter has left the polling station, his or her right to appeal false receipts will have elapsed.
The difficulty is that it is easy to verify a digital signature on a computer but impossible for a human to perform such a calculation mentally. While at the polling station, the voter is virtually devoid of any trusted hardware and therefore does not have the ability to check the veracity of the digital signature in a way that is reassuring.

Alternative approaches have been suggested ([CBH11, Rya11]) that either augment or entirely do away with the digitally-signed receipt. Such schemes are based on verifying codes to ensure that the vote has been accurately recorded on the WBB. Such schemes have the desirable property that, upon leaving the polling station, voters will have already completed their verification step. However, such schemes do require a higher level of trust in the WBB, although there already has to be a certain degree of trust in the WBB due to the digital signatures. The bigger disadvantage is that the codes used to verify the recording of the vote must be distributed to the voter. The typical suggestion is to include them on the ballot form issued to the voter. However, this places a chain of custody requirement on those ballots, which, if breached, could potentially undermine the election's integrity. There may be situations where such a chain of custody already exists or where it is a preferred compromise to the digital signature approach.

The final and preferred option is to permit voters to use their mobile phones to verify the digital signature. Constructing a phone application to perform such a task is relatively easy: multiple organizations could work on providing such an application, allowing voters to use an app from an organization they trust or perhaps even build their own. Such an approach does require that the voter be in possession of a smartphone and that they sufficiently trust the device and the application to perform the operation. There is growing concern about malware on mobile devices, but currently the average user is likely to trust such a device. This approach also causes concerns about disenfranchising the poor or seniors, both groups that tend not to own smartphone devices. While this may be true, the validity of the system only requires a small number of people to check their receipt. Unless the machine/system can know in advance whether someone has a smartphone, it cannot risk cheating in case it gets caught. There may also be legislative problems with allowing phones and photographic devices to be used in a polling station; however, provided that the process is well-managed and audits be performed in a designated area, such concerns should be mitigated. It is worth noting that checking the signature can be performed at the polling station, in public, with assistance if necessary.

3 Back-End Design

In this section, we discuss how to tally the received encrypted votes into the election result.
We use the Exponential ElGamal cipher [ElG85] in our protocol. A plaintext message m will be encrypted as $E(m) = (g^m y^r, g^r)$. In the ballot form, there will be a ciphertext next to each candidate. Suppose there are k candidates in the election, the i -th candidate will be encoded as $E(M^{i-1})$, where M is a value larger than k (e.g. $M = k + 1$). A received vote will look similar to the following table (note that the columns might be in different orders, but the tally methods will not be affected):

Ciphertext	$E(M^0)$	$E(M^1)$	…	$E(M^{k-1})$
Ranking	R_1	R_2	…	R_k

Table 2: Received votes

3.1 Tally Method 1

We first sort the ciphertexts within the above table according to their rankings. The result will be a k -ciphertexts tuple $\{c_1, c_2, \dots, c_k\}$ ranked in the canonical order. We then treat each of the ciphertext tuples as an input to the mix-nets (e.g. Verificatum [Wik10]). After the shuffle, all ciphertexts in the outputs are decrypted, and the election result will be calculated. However, the biggest drawback of this method is that the computational cost for the shuffle and decryption phase will be expensive if the number of candidates is large. Hence it is not ideal for elections with large numbers of candidates.

3.2 Tally Method 2

Alternatively, for a particular vote, we can use the homomorphic properties of the exponential ElGamal cipher to first absorb all the ciphertexts and their corresponding rankings into a single ciphertext as follows[2]:

$$E(m) = \prod_{i=1}^{k} E(M^{i-1})^{R_i} \quad \text{where} \quad m = \sum_{i=1}^{k} ([R]_i * M^{i-1}).$$

[2] Note that in order to ensure the correctness of the election result, we need to ensure that m is always smaller than q which is the order of g . For 128-bit, 256-bit and 512-bit q , we can handle at maximum 27, 47 and 81 candidates respectively.

Then for each vote, we input the ciphertext $E(m)$ into the mix-nets. After the shuffle, all the ciphertexts will be decrypted. Hence, somewhere in the outputs, there will be a value g^m. In order to retrieve m from g^m, we can compile a look-up table for all ($m : g^m$) value pairs in advance (e.g. even before the tally phase starts). After the decryption, we search the table to retrieve the value m , and the ranking choice for this vote can be calculated using the value m .

This method is superior to *tally method 1* because the computational cost for the shuffle and decryption phase has been reduced to the minimum: for each vote, there is only one ciphertext to be shuffled and decrypted. However, the disadvantage is that we need to build a look-up table in order to retrieve the plaintext. For an election with k candidates, the look-up table will contain $k!$ different ($m : g^m$) values. So for elections with small numbers of candidates (e.g. Victoria's lower house election with around 7 candidates), to build such a look-up table is perfectly reasonable. But for elections with large numbers of candidates, it would be infeasible to build such a look-up table. For example, Victoria's upper house election will have $35+$ candidates, and the size of the look-up table for 36 candidates is $36! \approx 3.72 * 10^{41} \approx 2^{139}$.

3.3 Tally Method 3

The third tally method can be considered as a trade-off between the above two methods. It is specially designed for elections with a large number of candidates. We use Victoria's upper house election as an example to demonstrate the idea (we assume there are 36 candidates).
Similar to the *tally method 1*, for a received vote as shown in the table above, we first sort all its ciphertexts into a *k*-ciphertexts tuple $\{c_1, c_2, \ldots, c_k\}$, which is ranked in the canonical order. Now, starting with the first ciphertext in the tuple, we treat every t ciphertext as a group. Hence for the VEC election, if we set the size of the group $t = 6$, we can separate all 36 ciphertexts into $\frac{k}{t} = 6$ groups. As follows, we treat each group as t ciphertexts ranked from 1 to t .

The following processes will be similar to the *Tally Method 2*. For each of the t -size groups $\{c_{j*t+1}, c_{j*t+2}, \ldots, c_{j*t+t}\}$ where $j \in \left\{0, 1, \ldots \frac{k}{t} - 1\right\}$, we will absorb all the t ciphertexts into a single ciphertext using the homomorphic property as follows:

$$E(m_j) = \prod_{i=1}^{t} (c_{j*t+i})^i$$

Hence, we have packed a k -ciphertexts tuple into $\frac{k}{t}$ tuples of t -ciphertexts each as

$$\left\{E(m_0), E(m_1), \ldots, E\left(m_{\frac{k}{t}-1}\right)\right\}$$

Then, for each received vote, we input its $\frac{k}{t}$ and t -ciphertexts tuples into the mixnets. After the shuffle, all ciphertexts in the outputs are decrypted. Note that after the decryption, somewhere in the outputs, we only obtain $\left\{g^{m_0}, g^{m_1}, \ldots, g^{m_{\frac{k}{t}-1}}\right\}$, and we still need one look-up table to retrieve their plaintexts {$m_0, m_1, \ldots, m_{\frac{k}{t}-1}$}. This time, the size of the look-up table is $P_t^k = \frac{k!}{(k-t)!}$ which is much smaller than $k!$. In our case ($k = 36$ and $t = 6$), the size of the table is $P_6^{36} \approx 1.4 * 10^9 < 2^{31}$.

Above, we have shown a special case where $t|k$. In the case $s = k\ (mod\ t)$ where $s \neq 0$, the above method still works. Now, we can group the k ciphertexts into several t -sized groups and the remaining s ciphertexts are treated as a group. In such a case, we need to build two look-up tables, one with size $P_t^k = \frac{k!}{(k-t)!}$ to look up the t -sizes ciphertext groups, and the other with size $P_s^k = \frac{k!}{(k-s)!}$ to look up the s -sizes ciphertext group.

Therefore, thanks to this tally method, we are able to handle elections with a large number of candidates. We can carefully choose the value of t (how many ciphertexts should be absorbed into a single ciphertext) so that the size of the look-up table P_t^k is reasonable. Meanwhile, the shuffle and decryption phase is t -times faster than the *Tally Method 1.*

4 Discussion

In the previous sections, we tried to clarify the fundamental design ideas in a simple manner, leaving out some technical details and design decisions. In this section, we will discuss some of these issues.

- *Where are the onions stored? :* In section 2, we mention that on the RHS, an encrypted value, called an *onion,* is associated with each candidate. This implies that the onions are printed on the RHS. However, in order to achieve the proper security level, the size of each onion will be around 1KB. Obviously, it will be impractical to print 36KB data on the paper ballot. To solve this problem, we suggest that onions be recorded on the WBB and that they are linked to a particular ballot using a unique serial number.

- *Italian attack:* There are two kinds of an "Italian attack". The first type works for elections in which the voter can express her preference in a large number of ways. Coercers can force a voter to cast her vote in a unique way that no one else might use. Thus, if coercers find out that no one has cast a vote in this way, the voter will be caught. The second type works for elections in which the transfer history is revealed. Coercers can force a voter to rank an unpopular candidate before a popular candidate. Therefore, if the unpopular candidate is eliminated but there is no vote transfer to the popular candidate, the voter will be caught. The tally methods in this scheme are not able to prevent either kind of Italian attack, but this is a design decision; a tradeoff between security and efficiency. According to some recent works, several new schemes (e.g. [TRN08, BMN09, XCH10]) can prevent Italian attacks; however, their computational costs prevent them from being implemented in practice at the moment.
- *Ballot validity proof:* Generally speaking, in verifiable elections with homomorphic tallying, every ballot should contain some validity proof, which proves that each ciphertext encodes one of the pre-defined values. Otherwise, a faulty ballot could ruin the election result by introducing thousands of extra votes. In our design, although the homomorphic property has been used in the tally phase, it is only used to encode preferences within the ballot itself, not encode preferences across different ballots. Hence the ballot validity proof is not required. Any invalid ballot can only ruin itself: it could neither introduce extra votes nor ruin the other ballots.
- *Impact of the different tallying methods:* In section 3, although we have introduced three different tallying methods, the first two are just special cases of the last method. The major difference lies in how many ciphertexts can be absorbed into a single packing. Election authorities should choose this parameter based on different circumstances, and the selection will only affect the computational cost in the tallying phase rather than the security properties.
- *Vote packing using small primes:* There is an alternative method to pack the ranking information using small primes [PABL04]. For example, $p_1, p_2, \dots, p_k$ are small primes representing each of the candidates, and $r_1, r_2, \dots, r_k$ are their rankings respectively. Then the vote can be packed as $v = p_1^{r_1} * \dots * p_k^{r_k}$. However, compared with the method we have introduced in the paper, this method has two drawbacks. First, when using small primes as counters, the aggregated value will grow very quickly as the number of candidates increase. If the said value is larger than p, it will be wrapped around by p, and we will still need a look-up table when retrieving the ranking choices. Moreover, this could also cause collision problems. Second, safe primes (primes of the form $p = 2q + 1$) need to be used so that small primes in G_q can be selected as the counters. However, this will result in a much larger q, making many calculations much slower. With our method, primes of the form $p = kq + 1$ where $k > 2$ can be used to speed up ballot generation and tallying without affecting security.

5 Security Properties

In this section we will briefly discuss how the modifications made to standard PaV impact the security properties normally associated with PaV. There are a number of security properties that are important to an electronic voting scheme. They are:

- Integrity
- Privacy
- Receipt-freeness
- Coercion Resistance
- Verifiability
- Usability

The integrity and receipt-freeness properties of the proposed system are identical to that of standard PaV. The manner in which the ballot form is filled out has changed, but not the underlying casting process or receipt construction. Likewise, the verifiability properties are transferable, provided that the voter performs the necessary checks, namely checking the overprinting and the digital signature. It could be argued that this is a more difficult task with the proposed system given the quantity of information that needs checking. However, the system does make it is easier to correctly complete the complex ballot form. The complexity of checking is a consequence of the complexity of the election, not the underlying system. While usability has improved in one sense, filling out the ballot, it may suffer in terms of how the overprinting approach will work. This requires further analysis and trials to determine how easy and reliable it is for the voter to perform.

The issue of robustness has been constantly considered and has influenced the design with aspects like the WBB peered among different parties. The robustness of the system is dependent on both the technology and the procedures surrounding it and is still being refined. The issue of requiring a network connection throughout the election in order to submit votes to the WBB and receive digital signatures back is a possible weakness. Various fallback options are being discussed and analyzed to determine the best compromise.

It is the privacy property that is most affected by the proposed changes. The system now utilizes an EBM that "learns" the vote. Strategies for mitigating this have been included, for example, enforcing that the EBM be offline and wiped clean at the end of the election. However, there is a new trust assumption here, that the EBM has been honestly setup and has not been compromised in any way to record and transmit the votes.

The issue of coercion resistance is impacted by the changes in privacy. Coercion resistance is far more complicated, since it also covers the perception of the voter. A weakening of privacy guarantees would likely reduce coercion resistance; such a discussion is beyond the scope of this paper.

6 Conclusion

In this paper we have presented an end-to-end verifiable voting scheme that would be suitable for use in a Victorian state election. We have detailed the modifications we would need to make to standard PaV in order to comply with the requirements of scale, usability, and legislation. In trying to move from theory to practice, modifications and compromises are a necessity. The challenge is choosing the right compromises and being able to adequately justify them. While some of these modifications are specific to the state of Victoria, for example above-the-line and below-the-line voting, the process we have undertaken is transferable to alternative scenarios.

Acknowledgements

This work has been partially funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under project 'TVS: Trustworthy Voting Systems' (EP/G025797/1) and the Luxembourg National Research Fund (FNR) under project SeRTVS-C09/IS/06.

Bibliography

[BMN09] 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), 2009.

[CBH11] Chris Culnane, David Bismark, James Heather, Steve Schneider, and Sriramkrishnan Srinivasan. Authentication codes. Proceedings of the 6th USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'11), 2011. San Francisco, CA.

[CCC08] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. Proceedings of the 3rd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'08), 2008. San Jose, CA.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: toward a secure voting system. 2008 IEEE Symposium on Security and Privacy, 2008.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. Proceedings of the 10th European Symposium on Research in Computer Science (ESORICS'05), pages 118–139, 2005. LNCS 3679.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on IT, 31(4):467–472, 1985.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. Advances in CRYPTO'01, pages 368–387, 2001. LNCS 2139.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. Proceedings of the 8th ACM Conference on Computer and Communications Security (CSS'01), pages 116–125, 2001.

[PABL04] Kun Peng, Riza Aditya, Colin Boyd, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In Advances in Cryptology - Indocrypt 04, pages 61–72, 2004. LNCS 3348.

[Rya04] Peter Y. A. Ryan. A Variant of the Chaum voter-verifiable scheme. Technical Report of University of Newcastle, CS-TR:864, 2004.

[Rya11] Peter Y. A. Ryan. Prêt à Voter with confirmation codes. Proceedings of the 6th USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'11), 2011. San Francisco, CA.

[TRN08] Vanessa Teague, Kim Ramchen, and Lee Naish. Coercion-resistant tallying for STV voting. Proceedings of the 3rd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'08), 2008. San Jose, CA.

[Wik10] Douglas Wikström. Verificatum, 2010. http://www.verificatum.org/verificatum/.

[XCH10] 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: 11th International Conference on Cryptology in India, 2010. LNCS.