

## **Tear and Destroy: Chain voting and destruction problems shared by Prêt à Voter and Punchscan and a solution using Visual Encryption**

**D. Lundin**

*University of Surrey*

**P. Y. A. Ryan**

*Newcastle University*

**J. Heather**

*University of Surrey*

**H. Treharne**

*University of Surrey*

**S. Schneider**

*University of Surrey*

**Z. Xia**

*University of Surrey*

---

**Abstract.** Prêt à Voter and Punchscan are two electronic voting schemes that both use paper based ballot forms, part of which is detached and destroyed, to provide receipt-free voter verifiability. However, both schemes share the chain voting problem and the part destruction problem. The first is where anyone who can see the ballot form before it is used can coerce a voter who uses it and the latter where a voter who can leave with the complete form can prove to a coercer the contents of the vote.

In this paper we provide a comparison of the schemes from a systems perspective. We also introduce a visual encryption solution to both the above problems.

**Keywords:** Prêt à Voter, Punchscan, election integrity, ballot secrecy, voter anonymity, receipts and coercion resistance

### **1. Introduction**

Prêt à Voter [2] and Punchscan [7, 3] are both electronic voting schemes that provide voter verifiability through paper based ballot forms. A further similarity between the schemes is that one part of the ballot form has to be destroyed in order to conserve receipt freeness and thus safeguard voter integrity and

anonymity [8, 9]. If the voter is somehow able to leave the voting place with both parts of the form then he or she can prove the contents of the vote to a coercer.

As both schemes rely on part of the ballot form to be destroyed and thus leaving an encrypted receipt, in both the secrecy of the vote is compromised if anyone can see the ballot form before it is used by the voter in the booth. Anyone who can note down information from the form can check the contents of the vote using the receipt shown on the web bulletin board after the close of the election.

It is clear that a coercer who can see the form before it is used can use that information to coerce a voter — but even if the form is kept safe in transit, a central organisation has created the form and thus solely holds the contents of the form. A corrupt election authority can therefore compromise the integrity of the election.

The schemes thus share two vulnerabilities: chain voting and enforcing part destruction.

This paper builds on a previous paper [5] which introduced visual encryption of the ballot form in Prêt à Voter. The contribution of this paper is to show how the same visual encryption can be used in Punchscan as well as provide a comparison between Prêt à Voter and Punchscan from a systems perspective.

The following introduces both schemes in more detail. In Section 2 we compare Pret a Voter and Punchscan from a systems perspective. In Section 3 we propose adding a visual encryption element to both Pret a Voter and Punchscan in order to provide a solution to the chain voting and part destruction problem. Section 4 details the visual encryption and scrambling technique and the paper concludes with a summary and future issues to explore.

## 1.1. Prêt à Voter

Based on a paper by David Chaum [1] Prêt à Voter [2] introduces a paper-based ballot form to aid user recognition and simplify use. The ballot form consists of a candidate list in the left of two columns, the order of the candidates in a seemingly random order for each form. To the right of the candidate list is a grid in which the voter marks her choice. Underneath this grid, also on the right hand side of the form, is printed a unique *onion*. This onion encapsulates the order of the candidate list in a number of *germs*, each hidden underneath a layer of encryption. An example of the ballot form is shown in Figure 1.

The ballot form is created in the *pre-election phase* by the election authority which selects all the random germs, performs the shift of the candidates based on the sum of these germs and creates the onion by encrypting each layer under the public key of one of a number of trusted parties, *tellers*.

During the *election phase* the voter marks her choice with an X in the grid next to the intended candidate and detaches the candidate list which is then destroyed. What remains is a grid with an X and the onion, making out an encrypted *receipt*. This receipt is scanned and the position of the X together with the onion is transmitted digitally and displayed on the *web bulletin board* after the close of the election.

In the *post-election phase* the voter can check that the representation of the receipt shown on the web bulletin board corresponds to the one held in paper form. If that is the case the voter can be confident that the vote has been cast as intended.

During the *tallying phase* the receipts are passed as a batch between the tellers, each of which removes its layer of encryption and extracts its germ. It reorders the position of the X based on this germ, mixes the batch and passes it on to the next teller in order. When all tellers have performed

LISA	
MAGGIE	
HOMER	
MARGE	
BART	

a45K1s

Figure 1. Prêt à Voter ballot form

82347
(c) Homer
(a) Marge
(b) Bart
(e) Lisa
(d) Maggie
(d) (b) (e) (a) (c)

Figure 2. The ballot form in Punchscan

these anonymising decryption mixes, what remains is a number of votes in the base order which can be counted.

## 1.2. Punchscan

The Punchscan system [7, 3] was invented by David Chaum and based on his early voting system [1] which relied on a visually encrypted receipt printed by the Direct Recording Electronic voting machine (DRE). Punchscan uses a ballot form printed on two pages, one on top of the other, before the start of the election. The top page carries a list of the candidates and associated symbols. It also has a series of perforations through which the same set of symbols printed on the second page can be seen. An example of the Punchscan ballot form is shown in Figure 2.

During the *election phase* the voter indicates his or her choice by colouring the symbol on the second page which corresponds to the candidate on the first page. As the colouring is done using a "dauber", bingo marker, the mark is made over the symbol on the second page but also around the edge of the perforation of the first page.

If the two pages are detached after the choice has been indicated in this fashion they each separately represent an encrypted receipt of the vote. One randomly selected page is destroyed making it impossible for anyone other than the election authority to decrypt the vote. The remaining receipt is scanned in, interpreted and transmitted electronically.

In the *pre-election phase* the election authority creates the data from which the ballot forms are printed. This consists of a ballot form serial number with the associated ordering of the symbols on the top page and the ordering on the bottom page.

Also before the election begins the authority creates a table of decryption data, holding values for two permutation functions. If a particular receipt is permuted using these functions and the decryption data, it will be changed into canonical form and thus be possible to count. Cryptographic commitments are published for all of the data in the decryption table. This means that the decryption data is put through

a publicly available hash function and the result is published.

As a pre-election audit step, auditors select half of these potential ballot forms, requiring the authority to display the underlying values so that anyone can check that the commitments are valid. The remaining forms are printed onto paper and can be used in the election, the assumption being that if the election authority is cheating then the likelihood of this being detected in the pre-election audit is substantial.

At least one chained decryption table must exist, each of which changes the order of the receipts in the table so as to obscure the identity of the voter. During a post-election audit the election authority is required to show some of this decryption data, although not enough to link a particular receipt to a vote. The more decryption tables that exist, the more data can be revealed in the audit, making it increasingly hard for the election authority to cheat.

In the *tallying phase* the election authority performs the permutations on the ballot forms as dictated by the decryption tables and then publishes the result in a result table, consisting simply of a number of canonical form receipts.

## 2. Comparison from a systems perspective

We have already noted that chain voting and part destruction are common vulnerabilities for the two schemes: Prêt à Voter (PAV) and Punchscan (PS). This section compares the techniques from a systems perspective in order to identify further similarities and differences. The structure of the section follows the structure of Sections 5, 6 and 7 of [9].

### 2.1. Subliminal channels

Subliminal channels are where for example the machine that creates the encrypted receipt includes extra information such as the contents of the vote or the identity of the voter in the receipt, in a form only legible to an accomplice. As the ballot forms of both schemes are printed in the pre-election phase and a ballot form is selected at random by the voter, no link can be made to voter identity or intention when the cryptographic commitments are made. The random subliminal channels described in [4] do thus not exist.

### 2.2. Social engineering attacks

Neither scheme uses a voting machine to create the receipt, in both the receipt is created by removing part of the plain ballot form. Attacks where the voting machine reorders steps in the algorithm in order to learn information which it should not learn before committing to a receipt are therefore not applicable.

### 2.3. Denial of service attacks

To allow ballot forms to be audited PAV uses the tellers in *oracle mode* during the election phase. This is the analogue of the cut-and-choose method in PS and other schemes. If the tellers are not all available no audit can take place.

During the tallying phase PAV is vulnerable to the loss of one or more tellers (unless re-encryption mixes are used as in [10]) but PS relies solely on one election authority to perform the decryption and

tabulation. Whether this makes PS more resistant against denial of service attacks is a matter of discussion.

Furthermore both schemes are susceptible to malfunctioning or compromised voting machines that submit several copies of valid receipts or do not pass on submitted receipts.

#### **2.4. Discarded receipts**

A receipt discarded by the voter indicates to a cheating entity that that particular receipt will not be checked for inclusion or validity on the web bulletin board. Both schemes are vulnerable to the election authority or a voting machine removing or changing the contents of those votes, be it randomly.

In [9] one suggestion to overcome this problem is using a Verifiable Encrypted Paper Audit Trail to allow for independent checking of all or a random sample of receipts.

#### **2.5. Invalid digital signatures**

To allow a voter to successfully accuse a malicious voting machine the cast ballot forms in both schemes are digitally signed. However, a voting machine may use a non-existent signature or one of another, properly functioning machine. The receipt then appears to be falsified by the voter who cannot prove that the machine is malfunctioning. Independent checking of the signature in the voting place by independent organisations is a possible solution to this problem.

#### **2.6. Insecure web bulletin board**

Many electronic voting schemes, including PAV and PS, incorporate a secure web bulletin board, most commonly in append-only mode. Security issues are in how to make such a bulletin board secure and ensure that it presents its contents correctly to all parties.

#### **2.7. Undermining public confidence in the secrecy of encrypted receipts**

This is a psychological attack that is separate from the implementation of the scheme as such, therefore both schemes are vulnerable as both use encrypted receipts. In short, a party might question the system without providing evidence that it is flawed. This may cause concern in voters who are not familiar with the inner workings of electronic voting schemes.

#### **2.8. Side-channel attacks**

As the voting machine does not learn the intention of the voter in either scheme, neither scheme is vulnerable to an attack where information about that intention is communicated via other channels from the machine. However, both systems are vulnerable to cameras, invisible marks made on ballot forms and so forth.

#### **2.9. Kleptographic channel attacks**

As described by [9], PAV is vulnerable to a corrupt election authority creating ballot forms with an onion conveying information about the candidate list order in some other way than through its full decryption.

For example, a hash of the onion value could indicate the order to a colluding compromised voting machine.

It seems likely that it is hard for the election authority to create such a channel in PS because the ballot form has a serial number instead of an onion. However, one possible channel might be where the election authority selects the order of the symbols in such a way that the order on the bottom page can be deduced from the order on the top page and vice versa.

The solution to this problem in both schemes is distributed creation of the ballot form.

## **2.10. Chain voting**

As we have already noted both schemes are liable to chain voting — in its simplest form described that anyone who can see the ballot form before it is used can check the plain text contents of the receipt as it appears on the web bulletin board. In PAV, anyone who can link a candidate list order to an onion and in PS, anyone who can note down the order on both pages as connected to a ballot form serial number, is able to check the contents of the vote as the receipt appears online.

Anyone who is able to derive the contents of the vote from the receipt posted on the web bulletin board without going through the cryptographic steps of the tallying phase is able to coerce a voter.

The solution to this is to keep the forms secure so that no-one can read their contents before they are used by the voter in the booth.

## **2.11. Authority knowledge**

In both schemes a central election authority creates the ballot form by selecting the values representing the shift of the candidate list or the symbols. This authority has to be trusted not to leak this information or use it to compromise the election. For example, even if the ballot forms are kept secure in transit, the election authority does hold the information needed to perform the chain voting attack detailed previously.

The solution to this must be to create the ballot form in a distributed fashion, sharing the secret of the contents of the form between a number of parties. One way of doing this using re-encryption mixes in PAV is presented in [10] and another solution based on visual encryption in both PAV and PS is presented in this paper.

## **2.12. Enforcing the destruction of part of the ballot form**

The conversion of a plaintext ballot form into an encrypted receipt is in both schemes done by removing part of the form. In PAV the candidate list is detached and in PS one page is removed and destroyed. There may be problems involved in enforcing this destruction and it might be possible for a voter to leave with both parts of the ballot form. In such a case the voter is able to prove to a coercer the contents of the vote.

There may exist physical solutions to this but this is based on trust in election officials who may in fact be colluding with a coercer. Another solution based on visual encryption is presented in this paper.

### 2.13. Confusion of teller modes

To audit ballot forms in PAV the order of the candidate list is re-created by the tellers in real time working in oracle mode during the election phase. A form that has been audited in this way must be destroyed to safeguard ballot secrecy. Similarly, if a ballot form already used to cast a vote can be audited this reveals the contents of the vote. There may be physical solutions to this problem such as two onions, one used for audit and the other for casting the ballot.

This problem does not arise in PS as this type of audit is not performed.

### 2.14. Summary

From the above it appears that PAV and PS are very alike when these vulnerabilities are considered — but one distinct difference is that PAV does place trust in a number of trusted parties where PS places all trust in the election authority.

## 3. Distributed creation of the ballot form using visual encryption

In order to prevent any one organisation from learning the content of the ballot form before it is used and potentially compromising the secrecy of the election its content has to be created by a number of trusted parties all working together but the final ballot form should not be visible to any one of these parties. In this section we review extensions to Prêt à Voter and propose a new extension to Punchscan.

### 3.1. Prêt à Voter

In original Prêt à Voter [2] the candidate list shift from the base ordering is dependent on a number of *germs* selected by the election authority and encapsulated under a number of layers of encryption, one for each of a number of tellers, in the onion. In this scheme the secret of the ballot form is held by the election authority which can compromise the election by leaking it.

In a recent update to incorporate re-encryption mixes [10] the ballot forms can be created in a distributed fashion by a number of clerks. By printing two separate onions onto the form one can be decrypted by either those clerks or the voting machine, revealing the candidate list order which is printed onto the form within the booth. This onion is detached along with the candidate list and the remaining receipt consists of the position of the X and the remaining onion, which can only be decrypted by the tellers.

This change does ensure that no entity involved in the process of creating the ballot form learns the order of the candidate list and it hides the candidate list from view until the form is used to cast a vote within the booth. It does not however address the problem of ensuring that the candidate list is detached and destroyed.

As an alternative approach to ensuring that no party learns the contents of the ballot form we presented a solution based on visual encryption of the candidate list in [5]. A basic overview of this solution is provided here.

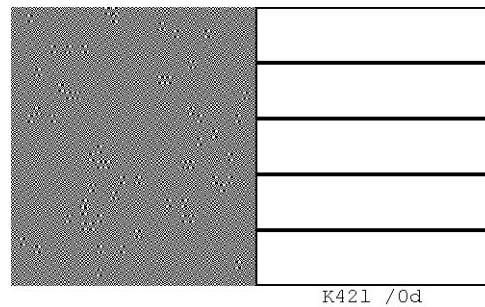


Figure 3. The visually encrypted ballot form in Prêt à Voter

### 3.1.1. Ballot form creation phase

The creation of the ballot form is started by the election authority which creates an image of the candidate list in the base order. It then visually encrypts this image by splitting it into two layers, using the technique demonstrated in [6]. See Section 4 for details.

For each ballot form the top layer of this encryption is fed to the first of a number of trusted parties, tellers. The teller selects a nonce and uses this to shift the visual encryption of the candidate list without knowing which candidate is which. It stores the nonce by encrypting it into the onion. In order to hide the reordering performed, the visual encryption is also scrambled using a mask derived from the nonce.

The resulting onion and visual encryption layer is printed onto a ballot form which may be printed and distributed by a third party if necessary.

### 3.1.2. Election phase

The voter selects a ballot form at random, inside or outside the booth. The form is placed on the voting machine which reads in the onion. The onion and the bottom layer of the visual encryption of the candidate list (in the base ordering, created by the election authority before the election) are passed to the last of the tellers in the sequence.

Each teller then strips off its layer of encryption and extracts its nonce. The nonce is used to shift the candidate list as well as perform scrambling. When all tellers have done this the resulting visual encryption layer is returned to the voting machine which displays it on a screen underneath the ballot form. When light passes from the screen through the ballot form the candidate list becomes human-legible.

At seeing the candidate list in plain text the voter can be confident that the ballot form is well formed and indicates her choice by marking an X in the grid next to the preferred candidate. The machine reads in this mark and stores and transmits the position of the X together with the onion to the election authority which publishes it on the web bulletin board. The voter is allowed to leave with the ballot form as the candidate list is visually encrypted and therefore does not have to be destroyed. From this point we regard the candidate list as detached and destroyed. After the election the voter can check that the receipt shown on the web bulletin board is exactly the same as the one held.

An example of a visually encrypted ballot form in Prêt à Voter is shown in Figure 3. A similar visual encryption of the ballot form in Punchscan is outlined in the next section.

Top page		
Bottom page	consists of	top layer bottom layer

Table 1. Two pages and two layers in Punchscan

### 3.2. Punchscan

Punchscan [7] relies heavily on a single election authority preparing the election and the ballot forms. It thus requires a number of modifications to distribute the secret contents of the ballot form over a number of trusted parties. Recall that a ballot form comprises of two pages. We propose to change the bottom page by visually encrypting it.

One main difference between Prêt à Voter and Punchscan is that the former uses an onion which encapsulates all the information needed to decrypt the vote and the latter uses a ballot form serial number which is used by the election authority to locate the decryption information in its secret database. Instead of introducing an onion to Punchscan we suggest that the serial number is used in the distributed creation also.

#### 3.2.1. Ballot form creation phase

In order to keep the contents of the ballot form secret at all times except to the voter in the booth, the two pages of the ballot form must not be seen together by anyone except the voter in the booth. There may exist a number of ways of ensuring this, for example the form might be created on demand by the voting machine.

Our solution must entrust the secret contents of the ballot form to a number of parties involved in the creation of it. As these trustees are not mentioned by [7, 3] we here name them similarly to the Prêt à Voter nomenclature: *tellers*. Each teller must contribute to the secret which determines the correct decryption of the encrypted receipt. In other words, each teller must be the guardian of a decryption table as described earlier.

We propose that each teller, in the ballot creation phase, creates a set of decryption data for the full set of pre-election potential ballot forms. Each teller publishes commitments to this data as described in [7].

The election authority creates a set of ElGamal encryptions, one for each ballot form, with the associated ballot form serial numbers. The encryption is made under the public key of the election authority. It also creates a visual encryption of the symbols on the bottom page in the base ordering as described in Section 4. The bottom page is thus split into two layers as shown in Table 1. The ElGamal encryption and the top layer of the visual encryption are then passed between all tellers, each of which adds a value to the ElGamal encryption representing its shift of the symbols on the top page and modifies the visual encryption to represent its shift of the symbols on the bottom page. The ElGamal encryption is described in Figure 4.

When all tellers have contributed to the ElGamal and visual encryptions the election authority decrypts the ElGamal encryption using its secret key and inserts this data in its ballot form table. The associated visual encryption is also inserted in the table.

---

Suppose  $x$  is the secret key,  $y$  is the public key

$$y = g^x \quad (1)$$

The election authority generates an ElGamal encryption based on the base order  $D_0$

$$c = (h^{D_0}y^{r_0}, g^{r_0}) \quad (2)$$

The first teller contributes its germ  $D_1$  to the ElGamal encryption

$$\begin{aligned} c' &= (h^{D_0}y^{r_0}, g^{r_0}) * (h^{D_1}y^{r_1}, g^{r_1}) \\ &= (h^{D_0+D_1}y^{r_0+r_1}, g^{r_0+r_1}) \end{aligned} \quad (3)$$

The second teller contributes its germ  $D_2$

$$\begin{aligned} c'' &= (h^{D_0+D_1}y^{r_0+r_1}, g^{r_0+r_1}) * (h^{D_2}y^{r_2}, g^{r_2}) \\ &= (h^{D_0+D_1+D_2}y^{r_0+r_1+r_2}, g^{r_0+r_1+r_2}) \end{aligned} \quad (4)$$

Each of the remaining tellers then contributes its germs until finally

$$\hat{c} = (h^{D_0+\hat{D}}y^{r_0+\hat{r}}, g^{r_0+\hat{r}}) \quad (5)$$

where  $\hat{D} = D_1 + \dots + D_n$  and  $\hat{r} = r_1 + \dots + r_n$

The election authority who has possession of the secret key  $x$  can decrypt  $\hat{c}$

$$h^{D_0+\hat{D}} = \frac{h^{D_0+\hat{D}}y^{r_0+\hat{r}}}{(g^{r_0+\hat{r}})^x} \quad (6)$$

---

By searching the field of  $h$  the authority can retrieve  $D_0 + \hat{D}$

Figure 4. Distributed creation of the ballot form in Punchscan using ElGamal encryption

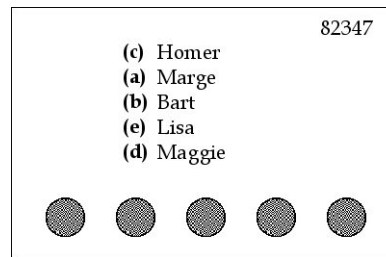


Figure 5. The visually encrypted ballot form in Punchscan

At this point all tellers must work together to show the contents of the bottom page, therefore the ballot form has been created in a distributed fashion.

During the pre-election audit the auditors require the election authority and each of the tellers to reveal the data associated with a subset of the ballot forms, for example a randomly selected half of the set. The tellers show their decryption data. Anyone can now check that all published commitments are valid and that the sum of the decryption data for the bottom page of the ballot form does indeed result in a plain image of the symbols, instilling trust that the remaining ballot forms are also valid and can be used in the election.

The election authority now prints the symbols on the top page in the order it has gained from the decryption of the ElGamal encryption and the associated visual encryption on the bottom page. These forms can be printed and distributed by a third party if needed.

### 3.2.2. Election phase

The voter selects a ballot form at random either in or outside the booth. The form is placed on the voting machine which reads in the serial number. The bottom layer of the visual encryption of the symbols on the bottom page in the base ordering is then passed to all tellers in order together with the serial number.

Each teller performs the same reordering and scrambling of this layer of the visual encryption as it did to the other layer in the ballot creation phase. When all tellers have performed their operations the resulting layer is returned to the voting machine which displays it on a screen underneath the form. When the light from the screen passes through the ballot form, the result is a human-readable list of symbols on the bottom page. The ballot form can now be used exactly as in [7].

An example of the visually encrypted ballot form in Punchscan is shown in Figure 5.

## 4. Visual encryption

The visual encryption, or division of the image into two layers, is the same as presented in [1], in turn derived from [6]. It is based on the division of each plain image pixel into four sub-pixels, each of which is one of two different pixel symbols as shown in Figure 6.

If two of these pixel symbols are placed in different layers they will yield one of three possible results, shown in Figure 7, when placed immediately above and below each other. The resulting pixel symbol that is completely black is perceived by the human eye as black and the two resulting symbols



Figure 6. The two pixel symbols

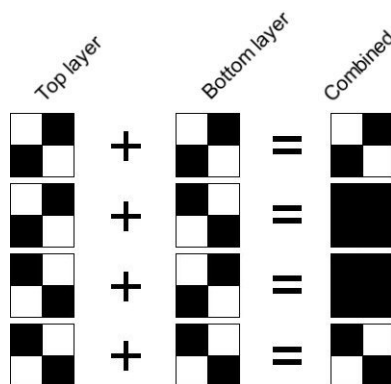


Figure 7. The three resulting pixel symbols

that have white sub-pixels are perceived as white. Thus, the plain image is in fact simply represented by white pixels on a black background.

It is easy to see from Figure 7 that a white pixel in the plain image is represented by the same pixel symbol in both layers; it can be either symbol as long as they are the same in both layers. The black pixel is similarly represented by different symbols in the two layers, but which symbol is in which layer is not dependent on the result (nor vice versa).

#### 4.1. Mathematical notation of the visual encryption

A simple mathematical model for the visual encryption is presented here to provide completeness.

The pixel symbols are represented by the integers 0 (white), 1 (white) and 2 (black) and thus in this representation the following is true:

$$0 + 0 = 0 \quad (7)$$

$$1 + 1 = 1 \quad (8)$$

$$0 + 1 = 2 \quad (9)$$

$$1 + 0 = 2 \quad (10)$$

It is easy to see that we can use this system to calculate the contents of the plaintext image when the two layers have been overlaid. The first constituent is the top layer, the second is the bottom and the resultant is the plain-text image.

HOMER
MARGE
BART
LISA
MAGGIE

Figure 8. The image of the candidate list

## 4.2. Example of visual encryption

By the following example<sup>1</sup> it should be more clear how the visual encryption of the image of the candidate list is done. An image of the list is created and shown in Figure 8. In simple terms we start by creating a bottom, random layer where both dimensions are twice those of the original image and the area has been randomly filled with the pixel symbols in Figure 6, resulting in the layer shown in Figure 12.

Because we randomise the visual contents of the bottom layer, this means that the top layer will not be random but dependent on the bottom. From a cryptanalytic perspective one might put forth that the fact that all the information is in the top layer that is printed onto paper and thus no information about the contents on a ballot paper can be derived from the pixel symbols used in the bottom layer that is handled electronically. This could be argued to enhance the security of the system because the layer handled electronically stands a slightly higher chance of being stolen.

We now create a representation of the original image, expanding each pixel into one of the pixel result symbols in Figure 7. For each pixel, if the current pixel is white then the pixel symbol used in this representation must be the same as the symbol in that particular place in the random bottom layer. Otherwise the symbol is simply the completely black. This complete representation can be found in the final image in Figure 12.

From the complete representation of the image and the random bottom layer we can create the top layer simply by going through each pixel and checking which symbol is in place in the complete representation. If that pixel is black then the pixel used in this layer must be the opposite to the one used in the bottom layer. Otherwise the pixel in the top layer must be the same to that of the bottom layer. The resulting top layer is shown in the middle image in Figure 12.

Thus, the superimposing of the top layer upon the bottom layer is shown in the final image in Figure 12.

## 5. The visual transformations

The transformations applied to the layers by a teller do not have to be reversible. In order to mitigate the buffering problem described above, they do however have to be possible to apply in any order and still yield the same result. One can say that the scrambling of the image is performed in order to mask the reordering at each stage.



Figure 9. Illustration of how the candidate list is split into smaller images

### 5.1. Reordering of the list

The teller treats the image of the candidate list as a set of vertically stacked smaller images (as shown in Figure 9), each of which contains the name of one candidate. By reordering these smaller images the teller also reorders the candidates, though without knowing which image has within it the name of which candidate. The basis for the reordering is of course the germ selected by that same teller in the ballot creation phase.

In this first instance the reordering of the candidate list is based on cyclic shifts.

### 5.2. Scrambling of the image

Also based on the germ created by that teller, the teller performs a scrambling of the image so that the reordering is not apparent to a spectator. If the scrambling is not performed, it is trivial to simply reorder the image of the base order list until a match is found. The theory is also that if the same transformations are applied to both layers, the final output will be a different but still legible candidate list.

One requirement on the scrambling of the list is that the transformations must be possible to apply in any order and still yield the same result, that is to say that the top layer should be possible to create in a forward teller order and the bottom layer in a reverse teller order and still yield a legible list.

To accomplish this the image is divided into a number of smaller images along the vertical axis, one for each candidate. The same scrambling is then applied to the same pixels of all these smaller images. If the scrambling described in Section 5 is applied to all these smaller images, it is evident that their ordering in the larger image does not matter but the result is the same.

### 5.3. Scrambling of the smaller images

The scrambling of each of the smaller vertical images is simple. The teller uses its germ to create a map of the image with a *true* or *false* value for each pixel. The pixel symbols in the positions with a *true* value are switched to the respective other symbol and those in positions with a *false* value are simply left as they are.

If such scrambling is performed in the same manner to both layers that make up the plain image, this yields the same plain image as if no such scrambling had been performed. This is because two of the

<sup>1</sup>This example is from Prêt à Voter but the application to Punchscan is very similar.

Layer contents	Change	Result
0	0 (no)	0
0	1 (yes)	1
1	0 (no)	1
1	1 (yes)	0

Table 2. Pixel symbols that are changed and the results

same pixel symbol result in a white pixel and two different pixel symbols result in a black. So if the pixel symbols in both layers are switched to the corresponding other symbol, the result will be the same.

### 5.4. Mathematical expression of the scrambling

The upper layer  $L_2$  and the bottom layer  $L_1$  are represented by two two-dimensional arrays. From Section 4.1 we know that the two pixel symbols in these layers are represented by the integers 0 and 1. Thus two examples of these layers are

$$L_1 = \begin{matrix} 1 & 0 & 0 & 1 \\ 1 & \ddots & & \\ 0 & & \ddots & \\ 0 & & & \ddots \end{matrix}, L_2 = \begin{matrix} 0 & 1 & 1 & 1 \\ 1 & \ddots & & \\ 1 & & \ddots & \\ 1 & & & \ddots \end{matrix} \tag{11}$$

The sum of these layers is thus

$$L_0 = \begin{matrix} 2 & 2 & 2 & 1 \\ 1 & \ddots & & \\ 2 & & \ddots & \\ 2 & & & \ddots \end{matrix} \tag{12}$$

Each teller that performs a transformation of the image creates a two-dimensional array which is a map of the cells that will be changed. The contents of this array is dependent on the teller's germ and some secret function. In the array, the integer 0 indicates that the pixel symbol will not be changed and the integer 1 indicates that the pixel symbol will be changed to the other. The following is an example of such an array:

$$F = \begin{matrix} 1 & 0 & 0 & 1 \\ 1 & \ddots & & \\ 0 & & \ddots & \\ 1 & & & \ddots \end{matrix} \tag{13}$$

The changes that are performed are shown in Table 2 and we can see from it that the resulting pixel symbol in the layer is determined by the XOR function. We can annotate this in the following way where  $G(x, y)$  is the layer being modified,  $F(x, y)$  is the modifier and  $R(x, y)$  is the resulting layer:

$$R(x, y) = G(x, y) \oplus F(x, y) \quad (14)$$

The following two arrays are (11) with (13) applied to them:

$$L'_1 = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & \ddots & & \\ 0 & & \ddots & \\ 1 & & & \ddots \end{matrix}, L'_2 = \begin{matrix} 1 & 1 & 1 & 0 \\ 0 & \ddots & & \\ 1 & & \ddots & \\ 0 & & & \ddots \end{matrix} \quad (15)$$

We now add together layers  $L'_1$  and  $L'_2$  and form the following resulting image:

$$L'_0 = \begin{matrix} 2 & 2 & 2 & 0 \\ 0 & \ddots & & \\ 2 & & \ddots & \\ 2 & & & \ddots \end{matrix} \quad (16)$$

Because the integers 0 and 1 are used to represent the “white” pixels, we find that those pixels are found in the same positions in (12) and (16). We can thus deduce that we have altered the contents of the layers  $L_1$  and  $L_2$  but the resulting layers  $L'_1$  and  $L'_2$  still yield the same visual contents to the human eye.

### 5.5. Example of scrambling of the image

A Java application has been written to perform the visual encryption of the image described earlier and the manipulations used in this section as examples of how the scrambling might work. The application takes an image such as Figure 8, encrypts it by splitting it into two layers and then allows the user to perform any number of manipulations as described in this section, saving the results to files. In a test the top and the bottom layer went through reordering and scrambling with the same seeds but in different order. Figure 10 shows the final top layer superimposed upon the final bottom layer — displaying the candidate list in the legible form only ever occurring within the voting booth.

### 5.6. Election phase visual well-formedness check

When the voter places the ballot form on the voting machine the onion is electronically read and sent to the tellers, in reverse order to the ballot form creation phase, as illustrated in Figure 11. The first teller to receive the onion removes its layer of encryption from the onion and extracts its germ. It then takes the bottom layer of the original visual encryption of the candidate list and using the germ it then reorders the candidate list and performs the transformations described in Section 5.

When all tellers have performed this decryption, reordering and transformation in order the result is passed to the voting machine where it is displayed on a screen underneath the printed copy of the top layer, yielding a legible candidate list.



Figure 10. The result of the scrambling

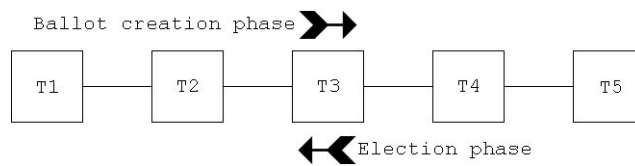


Figure 11. Teller communication order

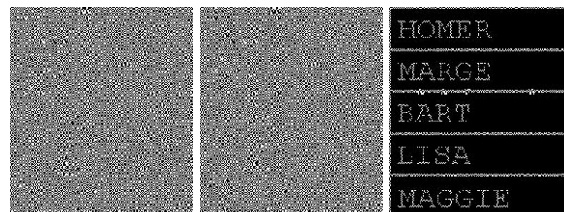


Figure 12. Top layer, bottom layer and both overlain

## 6. Discussion

### 6.1. Summary

In this paper we have given an overview of how visual encryption can be used in the Prêt à Voter and Punchscan electronic voting schemes in order to mitigate the chain voting and part destruction problems.

### 6.2. Future work

A number of open issues were identified in [5] and thus they are only listed here:

- Tellers perform correct task with the correct layer at the correct time — in order not to allow an encrypted receipt to be run through the tellers and thus check its contents the tellers must somehow be able to check that they work on only the correct layer of the visual encryption during each phase of the election.
- Ballot form shown in plain text must be cast or destroyed — if a ballot form is placed on the machine and thus displayed in plain text it must be used to cast a vote or destroyed in order to preserve coercion-freeness.
- Onion must be unique — the voter uses the onion (in Prêt à Voter) to search for the receipt on the web bulletin board and thus this must be unique for the ballot form, a check for this has to be incorporated into the ballot form creation process.
- Aligning the form properly on the display may be hard — there are physical issues in aligning the visually encrypted ballot form on the voting machine so that the layers combine to perform the decryption.

Furthermore, the visual encryption of the rather small symbols in Punchscan may render these illegible. Because of the way the voter marks her choice it is not possible to make these larger and therefore it may be problematic to introduce the visual encryption into the scheme.

### Acknowledgements

We would like to sincerely thank the FEE reviewers for their comments, in particular those regarding the practical problems of introducing visual encryption in Punchscan.

### References

- [1] Chaum, D.: Secret-ballot receipts: True voter-verifiable elections, *IEEE SECURITY & PRIVACY*, **2**(1), 2004, 38 – 47.
- [2] Chaum, D., Ryan, P. Y. A., Schneider, S.: A practical voter-verifiable election scheme, *COMPUTER SECURITY - ESORICS 2005, PROCEEDINGS*, **3679**, 2005, 118 – 139.
- [3] Fisher, K., Carback, R., Sherman, T.: Punchscan: Introduction and System Definition of a High-Integrity Election System, *PRE-PROCEEDINGS, IAVoSS Workshop On Trustworthy Elections*, 2006.

- [4] Karlof, C., Sastry, N., Wagner, D.: Cryptographic Voting Protocols: A Systems Perspective, *USENIX Security Symposium in Lecture Notes in Computer Science*, (3444), 2005, 186 – 200.
- [5] Lundin, D., Treharne, H., Ryan, P., Schneider, S., Heather, J.: Distributed creation of the ballot form in Prêt à Voter using an element of Visual Encryption, *PRE-PROCEEDINGS*, IAVoSS Workshop On Trustworthy Elections, 2006.
- [6] Naor, M., Shamir, A.: Visual Cryptography, *ADVANCES IN CRYPTOLOGY in Lecture Notes in Computer Science*, (950), 1994, 1 – 12.
- [7] Popoveniuc, S., Hosp, B.: An Introduction to Punchscan, *PRE-PROCEEDINGS*, IAVoSS Workshop On Trustworthy Elections, 2006.
- [8] Ryan, P. Y. A., Peacock, T.: *Prêt à Voter: a Systems Perspective*, Technical Report CS-TR-929, University of Newcastle, 2005.
- [9] Ryan, P. Y. A., Peacock, T.: A Threat Analysis of Prêt à Voter, *PRE-PROCEEDINGS*, IAVoSS Workshop On Trustworthy Elections, 2006.
- [10] Ryan, P. Y. A., Schneider, S.: Prêt à Voter with re-encryption mixes, *COMPUTER SECURITY - ESORICS 2006, PROCEEDINGS, TO APPEAR*, 2006.