

School of Computing Science,
University of Newcastle upon Tyne



A Simplified Version of the Chaum Voting Scheme

Peter Ryan and Jeremy Bryans

Technical Report Series

CS-TR-843

May 2004

Copyright©2004 University of Newcastle upon Tyne
Published by the University of Newcastle upon Tyne,
School of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK.

A Simplified Version of the Chaum Voting Scheme

Peter Y A Ryan and Jeremy W Bryans
School of Computing Science, Newcastle University, UK
{Peter.Ryan|Jeremy.Bryans}@newcastle.ac.uk

Abstract. We present a simplified version of Chaum's digital voting scheme. This appears to preserve the essential characteristics of the original whilst being significantly easier to understand and implement.

1 Introduction

In [2], Chaum presents a digital voting scheme that enables voter verification. In this paper we outline a variant of the original proposal that sidesteps much of the complexity of the original arising from the use of visual cryptography. In place of the visual cryptography we propose encoding the vote in terms of a pair of aligned strips. This greatly simplifies the presentation and implementation, and we refer to our simplified version as “*Prêt à Voter*”.

The implementation we propose here is related to, and in part inspired by, an approach proposed by van de Graaf, [4], that uses over-printing to obliterate half the ballot information.

The key elements of the Chaum scheme are:

- provide the voter with a receipt showing their vote in encrypted form
- show the voter the decryption of the receipt in the booth
- allow the voter to check that their encrypted vote is entered into the tally
- provide checks to ensure that the tellers correctly decrypt each receipt
- ensure that the ballots are scrambled during the decryption phase (Chaum anonymising mixes) to ensure that no linkage can be established between receipts (voter identity) and decrypted vote.

The design philosophy behind the scheme is to shift, by cryptographic means, the dependence away from the technical components to the socio-technical checking and recovery system surrounding the technical core. Following this philosophy, it is essential that the decryption process in the booth be transparent and not depend on the intercession of any hardware or software devices that might be susceptible to failure or corruption. The implementation proposed here strives to follow this philosophy.

We consider a simple voting process in which each voter is invited to choose between n options (one of which might be a “spoilt ballot” option).

Here we give a sketch of the scheme, omitting much of the cryptographic details. These are closely analogous to those used in the original scheme as described in [2] or [1].

2 Overview of the *Prêt à Voter* digital voting scheme

Our intrepid voter Anne first authenticates herself and registers at the polling station. She now enters a booth and initiates the vote casting process. She might for example be provided with a smart card or one-time-password to enable her to cast exactly one vote.

The booth generates a unique serial number q associated with this voting session. The booth now generates two n long pseudo-random strings. We will describe their construction shortly. One is destined for the upper strip the other for the lower strip.

2.1 Anne casts a vote

To illustrate, let us walk through an example. For concreteness, let us assume that there are 6 options offered to the voters. Anne logs onto the booth, which then generates a unique serial number q and the two 6 bit pseudo-random strings.

Suppose that the upper string z_u is:

0, 0, 1, 0, 1, 1

and the lower, z_l :

0, 1, 1, 1, 0, 0

The booth now inserts these into a table as follows.

1	2	3	4	5	6
0	0	1	0	1	1
0	1	1	1	0	0

We will refer to these bits as the encryption bits. The booth now inserts information bits into the spaces to form an initial, blank ballot: bits inserted in the spaces in the upper strip that match the encryption bits immediately below on the lower strip. Similarly for the information bits on the lower strip:

1	2	3	4	5	6
0	0	1	1	0	1
0	0	1	1	0	1

The booth is now ready to invite Anne to cast her vote. Suppose that she chooses “4”. The booth now flips the information bits in the “4” cell, thus:

1	2	3	4	5	6
0	0	1	0	0	1
0	0	1	1	1	0

The booth now prints a graphical representation of this onto the strips along with the serial number q :

1	2	3	4	5	6	
♥♥	♥♠	♠♠	♥♥	♠♥	♠♥	q
♥♥	♥♠	♠♠	♠♠	♠♥	♠♥	q
1	2	3	4	5	6	

Note that in the “4” cell, the upper and lower symbols complement, for all the other cells the upper symbols match the lower.

If Anne is happy with the encoding of her choice, she signals her approval. The booth now prints some further crypto commitments D_u and D_l to both strips. These are the “Russian dolls” that contain the information that allows the tellers to reconstruct the appropriate encryption string and hence extract the original vote. This will be described in detail shortly.

These should exactly match on the top and bottom strips, e.g.:

1	2	3	4	5	6	
♥♥	♥♠	♠♠	♥♥	♥♠	♠♥	q, D_u, D_l
♥♥	♥♠	♠♠	♠♠	♠♥	♠♥	q, D_u, D_l
1	2	3	4	5	6	

Assuming that these do indeed match Anne signals her okay, at which point the booth offers her the choice between the upper and lower strips. Suppose that she chooses the upper. The booth prints the crypto seed for the upper strip s_u on both strips as well as “Retain, upper” on the upper strip and “Destroy, lower” on the lower strip:

1	2	3	4	5	6	
♥♥	♥♠	♠♠	♥♥	♠♥	♠♥	$q, D_u, D_l, s_u, \text{Retain, Upper}$
♥♥	♥♠	♠♠	♠♠	♠♥	♠♥	$q, D_u, D_l, s_u, \text{Destroy, Lower}$
1	2	3	4	5	6	

Anne now detaches the strips from the printer, and separates them along the middle.

1	2	3	4	5	6	
♥♥	♥♠	♠♠	♥♥	♠♥	♠♥	$q, D_u, D_l, s_u, \text{Retain, Upper}$
♥♥	♥♠	♠♠	♠♠	♠♥	♠♥	$q, D_u, D_l, s_u, \text{Destroy, Lower}$
1	2	3	4	5	6	

On leaving the booth she should present the strip marked for destruction to an official who destroys it in front of her. She is now left with:

1	2	3	4	5	6	
♥♥	♥♠	♠♠	♥♥	♠♥	♠♥	$q, D_u, D_l, s_u, \text{Retain, Upper}$

Finally, she should present her retained receipt to one or more “bar-code” readers to be checked for well-formedness. These readers are able to reconstruct

the encryption string shown on the retained sheets and the corresponding doll from the revealed seed. This allows the Anne to be confident that the booth provided a doll that would yield the correct decryption of her vote, at least for the retained sheet.

Note that knowing the encryption string on the retained strip reveals nothing about the vote in the absence of the information bits on the destroyed strip.

3 Decrypting a Vote

Once the election has closed, the booth passes copies of the retained strips to a web site to be publicly posted along with the other receipts for the election. It is now the task of the tellers to progressively strip off layers of encryption and perform secret shuffles on the batch of receipts. In effect the tellers perform a Chaum anonymising mix on the receipts. To understand this in detail we need to examine more carefully the construction of the encryption strings and the dolls.

3.1 Construction of the encryption strings

The Booth has two independent secret keys SK_u and SK_l that, when applied to the serial number q , generate the seeds for the upper and lower encryption strings respectively:

$$\begin{aligned}s_u &:= \{q\}_{SK_u} \\ s_l &:= \{q\}_{SK_l}\end{aligned}$$

Suppose that there are k tellers. The encryption strings will each be formed by generating $2k$ strings and *xoring* these together. For concreteness we will describe the construction of the upper string, the lower is entirely analogous.

From the upper seed s_u , the $2k$ d -primed values are generated as follows:

$$d'_i := Hash'(s_u, i), \quad \text{for } i = 1, 2, \dots, 2k.$$

These primed, intermediate values will be used in the construction of the dolls, see below. Now the d' values are input into another hash function to yield the layers that will form the final (upper) encryption string:

$$d_i := Hash(d'_i) \quad \text{for } i = 1, 2, \dots, 2k.$$

This second, unprimed hash function should yield strings of length n .

Finally the upper encryption string is formed by *xoring* these $2k$ strings together:

$$z_u := d_1 \oplus d_2 \oplus d_3 \oplus \dots \oplus d_{2k}$$

3.2 Construction of the dolls

Note that, given the initial seed, the corresponding encryption string is determined. It is this fact that enables the barcode readers to verify the correct construction of the encryption strings on a retained strip. In the absence of the seed however the string cannot be determined. However, the d' values are buried in the dolls encrypted under public keys of the tellers. Thus, collectively, the tellers are able to reconstruct the encryption bits on the discarded strip.

Note that, for technical reasons that are presented in [1], each teller performs two Chaum mixes. This is why we need each of the encryption strings to be formed by *xoring* together $2k$ strings.

Let us describe this more precisely, For concreteness, we will describe the construction of just one of the dolls, in this case the lower as this will be used by the tellers to reconstruct the lower encryption string and so decrypt the information bits on the retained, upper strip. In analogy to Russian dolls, the dolls of Chaum's scheme are formed by nesting encryptions. The inner layer is formed by encrypting d'_{2k} by the second public key of the k th teller. The next layer is formed by concatenating d'_{2k-1} with the inner layer and encrypting this with the first public key of the k th teller. d'_{2k-2} is concatenated with this and the concatenation is encrypted with the second public key of the $k-1$ th teller. We continue in this manner until we finally form the outer layer as the encryption under PK_{T_1} of d'_1 concatenated with all the inner layers. Thus:

$$D_l := \{d'_1, \{d'_2, \{d'_3, \dots, \{d'_{2k-1}, \{d'_{2k}\}_{PK_{T_{2k}}}\}_{PK_{T_{2k-1}}}\dots\}_{PK_{T_3}}\}_{PK_{T_2}}\}_{PK_{T_1}}$$

3.3 The role of the tellers

Once all the receipt string/doll pairs have been posted to the web site, a transformation is performed on them before they are passed to the first teller.

Each of the receipt strings, which we will denote by R_u and R_l , will be $2n$ symbols and comprises an alternation of (a symbolic representation of) encryption and information bits, the slice depending on whether the receipt was derived from the upper or lower strip (as indicated by the upper/lower flag). The encryption symbols are now useless (their role was to perform the well-formedness checks) and are discarded. Thus all the receipt strings are transformed into n long strings comprising only information bits. The serial number, doll for the retained strip, seed, "Retain" and the upper/lower flag should also be discarded at this stage. Finally, the symbols should revert to the bit representation for ease of manipulation, and the cell numbers can also be discarded.

Thus, in our example, Anne's ballot receipt would go from:

1	2	3	4	5	6	
♥ ♥	♥ ♠	♠ ♠	♥ ♥	♠ ♥	♠ ♥	$q, D_u, D_l, s_u, Retain, Upper$

To:

0 0 1 0 1 1 D_i

i.e. the information bits of the upper strip along with the doll for the tellers, to reconstruct the corresponding encryption bits of the lower strip.

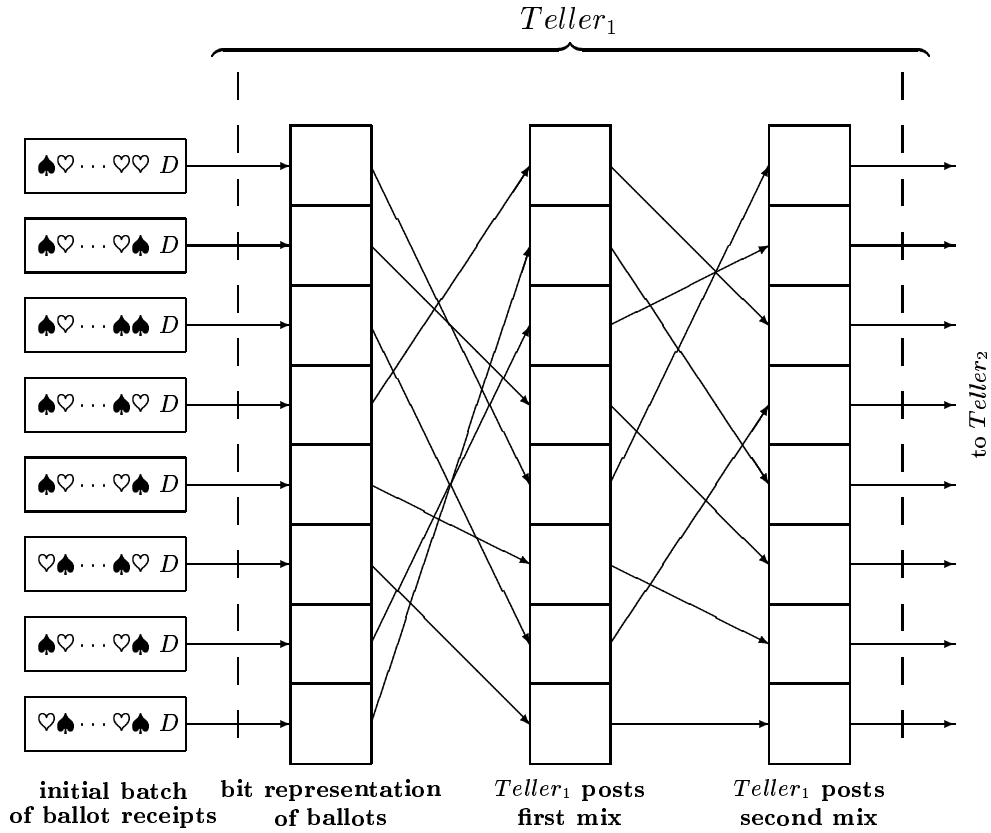


Fig. 1. $Teller_1$: permutations

$Teller_1$ now collects the resulting receipt/doll pairs and performs its anonymising mixes, as depicted in Fig 1. Note that the dolls are unique to each receipt.

For each receipt/doll $Teller_1$:

- applies its first secret key to the outer doll D_1 to reveal the d'_1 value and the enclosed doll D_2 .
- computes $\text{Hash}(d'_1)$ and *xors* this with the receipt string R_u to form R'_u .
- applies a secret permutation to all the R'_u/D_2 pairs.
- posts the resulting primed pairs to the web page.

Teller 1 now repeats this process using its second secret key. Note that the outer two layers are encrypted under its public keys so that it is able to strip both of these layers off.

$Teller_2$ now takes the R_u''/D_3 pairs posted by $Teller_1$ and performs its two mixes in the same fashion.

Finally, when $Teller_k$ performs its mixes this yields a set of receipt strings that show the votes in the clear. More precisely, a vote for candidate v will be revealed as a 1 in the v th place and 0's in all the other places. In the case of Anne's vote ("4") we get:

000100

To see this, recall the construction of the encryption bits and information bits. The encryption bits are formed by *xoring* the $2k$ d strings together. The corresponding information bits were then chosen to complement in the position corresponding to the choice of candidate and to match in all other positions. The result of the teller's operations on the receipts is to *xor* all the d strings into the information string. This is equivalent to *xoring* the encryption string into the information bit string so yielding the appropriate canonical representation of the vote.

The overall effect then is to have posted on the web site, in the left hand column say, the batch of initial receipts along with serial number and associated decryption doll as posted by the booth. In the right hand column we will have the fully decrypted ballot images. There will also be a set of columns in between with the intermediate, partially decrypted sets of receipts/dolls. Each column will be some secret permutation of the previous one. Note that the encryption prevents the permutation being reconstructed by simple matching of elements.

Assuming that all the tellers perform their transformations correctly, there will be a one-to-one correspondence between the elements of each column and the next. The exact correspondence, which receipt is the decrypt of which receipt in the previous column, will be hidden and known only to the teller who performed the transformation between those columns. Thus, the receipts will have undergone multiple, secret shuffles between the first column as posted by the booth and the final decrypted column. This ensures that no voter can be linked to their vote, so ensuring voter anonymity.

The fact that several tellers are used gives several layers of defence with respect to voter privacy: even if several of the tellers, but not all, are compromised, the linkage of voters with their votes will remain secret.

The decrypted votes will all be available in the final column output of the last teller and so the overall count will be checkable by anyone.

4 Checking on the booth

The description so far has assumed that all the players, the booth and the tellers, have behaved correctly, in accordance with the rules of the scheme. If

everyone obeys the rules we can be sure that the election will be both accurate and private. But this is a big “if”. Should the booth or any of the tellers cheat then the accuracy and privacy could be undermined. We really don’t want to have to put such a level of trust in the components of the scheme.

Anne should perform two checks that serve to detect attempts by the booth to cheat. She should run her receipt through a reader device that checks that the receipt has been correctly formed by the booth. Such devices should be readily available at the voting station for example and provided by independent organisations, for example, The Electoral Reform Society or similar. These devices, given the information on the receipt, can reconstruct the revealed encryption string z_u and the corresponding doll. Thus any attempt to decouple these by the booth (and so corrupt) a vote, will be detected.

The computations performed by the checker are as follows:

Apply the booth’s upper public key to the revealed seed and check that this yields the correct q .

$$q = \{s_u\}_{PK_u} = \{\{q\}_{SK_u}\}_{PK_u}$$

Note that both PK_u and PK_l are publically known.

From s_u the d' values can be computed. From these, the doll for the retained strip can be reconstructed. Note that the public keys of the tellers are publically known. This reconstructed value for the doll can be checked against the value printed on the retained ballot strip.

Now the unprimed d values can be computed by applying *Hash* to the primed values computed earlier. These can then be *xored* together to give the appropriate encryption string, z_u in our example, and this is now checked against the value printed by the booth.

This check ensures that if a malicious booth shows the voter one decryption, then creates a false doll which will produce a different decryption by the tellers, it has a one in two chance of being caught out.

Note that the algorithms for these checks are publicly known, so in principle, anyone could construct such a checker and make it freely available. Similarly anyone would be able to examine such a checker to establish that it was performing correctly. Indeed, if she is really enthusiastic, Anne may choose to run several such independent checks.

Once all the receipt batches have been posted to the web site, Anne should also check that her receipt is accurately recorded there.

5 Checking on the Tellers

Some checks must be performed to ensure that the tellers themselves perform all these transformations accurately, i.e. they do not alter, remove, inject or corrupt votes.

Recall that each teller performs two decryptions on each duo, and therefore each teller performs two permutations on each batch. They post each of these mixes.

However, if no further information about the process was revealed, tellers could alter votes without fear of detection. For example, if the last teller falsified all the final decrypted votes as being in favour of a particular candidate, it would be difficult to prove that anything illegal had taken place. The solution to this problem is derived from [3]. Tellers are required to provide *some* linkage information between the votes: enough to make the possibility of successful corruption vanishingly small, but not enough to allow any of the final decrypted votes to be traced back to the original votes cast.

For each posting half of the input links and half of the output links are revealed. Some external authority choses a random half of the duos in the first posting. For each chosen duo (R, D) the responsible teller must reveal: (1) the d' extracted from the D , and (2) the target duo in the subsequent posting (i.e., the *link*).

In the next posting all the duos not pointed to by links opened in the first batch have their outgoing links opened. No full link from a single final vote to a single original vote can therefore be drawn.

For each of the revealed links, if the duo has been transformed correctly we should have:

$$R_{i-1}, D_{i-1} \longrightarrow R_i, D_i$$

where

$$\begin{aligned} D_{i-1} &:= \{d'_i, D_i\}_{PK_i} \\ R_i &:= R_{i-1} \oplus d_i \\ d_i &:= h'(d'_i) \end{aligned}$$

Note that, along with the links, the corresponding d'_i is revealed. Auditors can therefore compute $Hash(d'_i)$, and check that this equals

$$R_{i-1} \oplus R_i$$

The PK_i key is public so the auditor can also calculate

$$\{d'_i, D_i\}_{PK_i}$$

and check that this equals D_{i-1} .

The fact that the d' is revealed along with the link is significant in that it foils a potential attack on the secrecy: knowledge of the d'_i value is essential in performing the checks. As this is the pre-image of a crypto-hash it should be intractable to compute this from knowledge of the final output, the d_i value.

If the checks could be performed knowing only the putatively linked duos then the scheme would be vulnerable to a guessing attack: Given a putative link, compute the checks. If the checks work you have, with high probability, identified a valid link. In this fashion you could, in principle, reconstruct as much of the secret permutation as needed.

Without knowledge of the d'_i pre-images, such an attack is intractable.

6 Conclusions

We have presented a simplified variant of the Chaum digital voting scheme. It preserves the essential features of the original whilst sidestepping the complexity of the visual cryptography of the original. Whether it could be a viable, practical alternative to the original would require investigation. The presentation of the encoding on the vote to the voter is arguably less obvious than in the original, in which visual cryptography is used to present the voter with a 2 dimensional image which could be used to spell out the candidates name for example. Another possible issue is that this scheme is probably only really suitable for elections etc in which there is a smallish number of options. There would seem to be quite a large class of elections and referenda that would qualify, but it would be unsuitable for US style "butterfly" ballots.

7 Acknowledgements

The authors would like to thank David Chaum for many helpful clarifications regarding details of his scheme and Jeroen van de Graaf for useful discussions.

References

1. Jeremy Bryans and Peter Ryan. A Dependability Analysis of the Chaum Voting Scheme. Technical Report CS-TR-809, Newcastle University School of Computing Science, 2003.
2. David Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security and Privacy*, 2(1):38–47, Jan/Feb 2004.
3. M. Jakobsson, M. Juels, and R. Rivest. Making Mix Nets Robust for Electronic Voting by Randomised Partial Checking. In *USENIX'02*, 2002.
4. Jeroen van de Graff. Adapting Chaum's Voter-Verifiable election scheme to the Brazilian system. 2004. Presented at WSeg 2004 IV Workshop em Seguranç de Sistemas Computacionais, Brazil.