

Kleptographic Attacks on E-Voting Schemes^{*}

Marcin Gogolewski^{1,**,†}, Marek Klonowski^{2,***}, Przemysław Kubiak^{2,†},
Mirośław Kutylowski², Anna Lauks², and Filip Zagórski²

¹ Faculty of Mathematics and Computer Science,
Adam Mickiewicz University

² Institute of Mathematics and Computer Science,
Wrocław University of Technology

Marcin.Gogolewski@amu.edu.pl, {Marek.Klonowski,
Przemyslaw.Kubiak, Mirosław.Kutylowski, Anna.Lauks,
Filip.Zagorski}@pwr.wroc.pl

Abstract. We analyze electronic voting schemes and show that in many cases it is quite easy to implement a kleptographic channel, which is a profound danger for electronic voting systems. We show serious problems with Neff's scheme. We present also attacks on Chaum's visual voting scheme and some related schemes, which work at least when implementation is not careful enough.

Keywords: kleptography, electronic voting, receipt voting, coercion, election integrity, verifiable pseudo-randomness.

1 Introduction

The concept of electronic elections gains popularity nowadays. Electronic voting systems contribute to decreasing costs of elections, provide more efficient procedures of counting and collecting votes and offers more flexibility than traditional voting. Due to growing interest of the topic, many new voting schemes were proposed recently. A collection of them can be found via Ronald Rivest's web page [15].

The most important goal that has to be achieved by the voting scheme is to prevent manipulation of the votes and changing the election result. At the same time, anonymity of the voters should be preserved and possibility of selling a vote by a voter must be excluded. So called *voter verifiable voting schemes* enable a voter to convince herself that her vote has been included in the final tally.

E-voting schemes become more and more sophisticated, with many wonderful tricks. Nevertheless, security analysis often disregards many dangers and does not treat the voting system as a whole. Having in mind the importance of election process and scandals

^{*} partially supported by KBN grant 0 T00A 003 23.

^{**} Author is a holder of a scholarship received from Measure 2.6 of Integrated Regional Operational Programme.

^{***} Author is supported in part by Domestic Grant for Young Scientists Programme from The Foundation for Polish Science.

[†] On a leave from Adam Mickiewicz University.

connected with the existing voting systems [20, 19], one cannot simply assume that companies creating e-voting systems will never try to put any trapdoor in their products. Therefore a voting system should be designed in such a way that each its part can be verified.

It was observed that using randomness in e-voting schemes yields a threat of constructing a subliminal channel by a malicious voting machine. Such a machine can imperceptibly pass on its secret values by generating a random components in a cryptographic way. We point out that the actual attack on an e-voting system might be far more dangerous than a simple subliminal channel. The attack can be mounted in such a way that the information leaked can be retrieved only by a party possessing a certain secret key. Moreover, such a malicious implementation neither changes the protocol executed nor can be detected without reverse engineering of the software running on the device, and even if one reveals malicious code and data inside the device, it remains impossible to perform the same attack on other devices infected in the same way. In other words, technique called *kleptography* [22, 23, 24, 25] may favor a single party over other ones with ability to buy votes, identify its opponents, or even imperceptibly falsify the election results.

2 Previous Works and Our Contribution

Some gaps in the security of verifiable voting protocols have been noticed and described by Karlof et al. in [6]. They proposed various social engineering and subliminal channel attacks on two prominent schemes: Neff's scheme [13] and Chaum's Visual Voting [2]. These attacks enable vote coercion and changing the contents of encoded votes. They considered also denial of service attacks that can be particularly dangerous in the context of electronic voting.

Another important, recent paper about attacks on voting schemes is [17]. In this paper P. Y. A. Ryan and T. Peacock present an extended version of Prêt á Voter scheme and its analysis as well as some other attacks on Chaum's and Neff's schemes not included in [6]. Authors also point to several attacks tailored for Prêt á Voter scheme and design appropriate countermeasures.

In our paper we present attacks on four different verifiable voting schemes: the first one presented by M. Klonowski et al. in [7], the second one presented by D. Chaum in [2], the third one presented by C. A. Neff in [13], and the fourth one presented by P. Y. A. Ryan et al. in [3]. In some sense our work can be regarded as an extension of paper [6] of C. Karlof et. al, but we point to some aspects that seems to be far more dangerous. Namely, we prove that (pseudo) randomness can be a tool not only for creating a subliminal channel (as it was in [6]) but also for constructing a kleptographic trapdoor in the voting schemes. Such a trapdoor can be used only by a particular adversary and, as we have already noticed in the introduction, that is the cause of a huge asymmetry.

Most importantly, the possibility of implementing a kleptographic attacks in e-voting schemes is a strong argument against the point of view presented by some e-voting companies, which assure that systems are secure, since the code was written and audited according to the rigorous procedures and security standards.

Notation: In the subsequent sections Mallet is the name of an adversary and n is a number of candidates. Most systems include similar components: voting machines, registration machines, and a bulletin board, which we will denote correspondingly as VMs, RMs, and a \mathcal{BB} .

3 A Practical Voting Scheme with Receipts

Description of the Scheme. Below we recall the scheme from [7] (see also [21]). Besides, we took advantage of having access to the specification of a test implementation. The system consists of: VMs, RMs, and *tallying authorities*. There are also some *control servers* provided by independent watch dog organizations.

In the initialization phase, a product $\prod_{j=1}^{\lambda} y_j \bmod p$ of public keys (g, y_j, p) of λ tallying authorities is loaded to each VM, as well as the lists of the candidates. For the simplicity of a description of a protocol, we consider single elections with two candidates, namely blue party B and yellow party Y . Shortly before an election starts every VM generates two pairs of keys for signature schemes, with private keys K, K' . In the voting phase the following steps are executed:

1. A VM creates a *virtual ballot* (it exists only in the processor's memory) consisting of random numbers r, q, r_L, r_R and two sides (left and right), with $n + 1$ triples on each side:

$$\begin{array}{ll} (B, B_1^L, B_2^L), & (Y, Y_1^R, Y_2^R), \\ (I, I_1^L, I_2^L), & (B, B_1^R, B_2^R), \\ (Y, Y_1^L, Y_2^L), & (I, I_1^R, I_2^R). \end{array}$$

r is a random ballot identifier (according to the specification r is a 64 bit random number r_s concatenated with VM's DSA signature of r_s). The element I contained in two triples is simply the identifier r . One can see on each side of the virtual ballot there are three columns: in the leftmost one there are the names of the candidates C and identifier I , in the next two columns on each side there are so called *RE-onions*: C_1^X, C_2^X , where $X \in \{L, R\}$, for each candidate C , and RE-onions I_1^X, I_2^X encoding identifier I .

The rows are permuted independently on each side, according to permutations π_L, π_R respectively. Each π_X is obtained deterministically from the contents of all columns on side X . RE-onions are ElGamal ciphertexts

$$(m \cdot (\prod_{j=1}^{\lambda} y_j)^k, g^k), \quad (1)$$

for plaintexts m defined for $X \in \{L, R\}, i \in \{1, 2\}, C \in \{B, Y\}$ as follows:

$$\begin{array}{l} m = (C, r_X, \text{ser}_V, \text{sig}'_{K'}(C, r_X, i)) \text{ for } C_i^X, \\ m = (r, \text{ser}_V, \text{sig}'_{K'}(r, i, X)) \text{ for } I_i^X, \end{array} \quad (2)$$

(ser_V is an identifier of the VM). For each RE-onion $Z_i^X, Z \in \{C, I\}$, the exponents k are computed as follows: the VM creates a signature $\text{sig}_K(q, i, X, Z)$ and uses it as a seed of a pseudo-random generator \mathcal{R} ; then k is taken from the output of

the generator. sig is a deterministic signature scheme (recall that the VM generates its keys itself, such a procedure poses a risk to RSA private keys, compare [25] and references given there).

2. The VM prints a *hash ballot* – it is a commitment to the virtual ballot that contains r , and – in a machine readable form - hashes of r , q , r_L , r_R and of all RE-onions Z_i^X , in the same order as in the virtual ballot.
3. The visualization of the virtual ballot appears on the screen of the VM.
4. The voter chooses a side (say R) and a party for which he votes (say B).
5. VM creates and prints a *voting ballot* that contains a pair (B_1^R, B_2^R) of RE-onions corresponding to the icon chosen and a pair (I_1^R, I_2^R) of RE-onions encoding the identifier I from the same side. These onions are printed in a random order, say given by a permutation π_{vb} . Additionally, the voting ballot contains a VM's signature of the values printed.
6. (Optional step) From the side X not used for voting (i.e. $X = L$ in our example) the voter may choose one column $i \in \{1, 2\}$ and some number of RE-onions Z_i^X in column i . The VM prints a *control ballot* that contains:
 - the RE-onions Z_i^X chosen for verification with their identifiers Z ,
 - the signatures used to generate exponents k in these onions,
 - the string r_X ,

After getting the control ballot the voter should compare the identifiers on the control ballot with the corresponding positions on the screen.

7. The voter comes to a RM and presents the voting ballot. Four RE-onions contained in the ballot are read in and stored for counting purposes, provided that the signature of the voting machine is valid. Simultaneously, the voting ballot is marked as used, and it is retained by the voter.

The voter can control honesty of VM by checking the control, hash and voting ballots through a machine that may read the printed values.

When all ballots are registered the tallying of the votes may start. From our point of view the tallying phase has some important features: there are no intermediate bulletin boards, in particular there is no bulletin board with the onions collected by RMs. Hence it is not necessary to collect voting ballots (as a fake watch dog organization for example) to change election results.

Only the last, final tallying authority publishes the list of completely decoded onions, and the list must contain:

- pairs encoding an identifier: $(r, \text{ser}_V, \text{sig}'_{K'}(r, 1, X)), (r, \text{ser}_V, \text{sig}'_{K'}(r, 2, X))$,
- and the same number of pairs encoding single votes: $(C, s, \text{ser}_V, \text{sig}'_{K'}(C, s, 1)), (C, s, \text{ser}_V, \text{sig}'_{K'}(C, s, 2))$, where $C \in \{B, Y\}$, and s are random strings.

Having her control ballot each voter can check whether the identifier r from the ballot is on the list.

Betraying Voters Preferences via Digital Signatures. We show how a VM may betray voter's preferences, even if the ballots are built according to the protocol. Such information may be available for Mallet on the final bulletin board. The attack is possible if

the signature scheme $\text{sig}'_{K'}$ used to create r from r_s is probabilistic, like DSA in the test implementation. A DSA signature (R, S) is generated as follows:

$$R = (g^\alpha \bmod p) \bmod q, \quad (3)$$

$$S = \alpha^{-1} \cdot (\mu - K' \cdot R) \bmod q, \quad (4)$$

where μ is a message to be signed, K' is the private key of the signer, and α is a random number. First, a VM can leak the signing key K' to Mallet in a kleptographic way [24] using only one signature. For this purpose, the VM must learn only the public key of Mallet. Then in every virtual ballot (except the one used for leaking K'), in all onions encoding candidates, the VM uses numbers α that betray r contained in the onions encoding identifiers. For example, let $\alpha = \alpha' \alpha''$, where

$$H(\alpha', r_s) = \alpha'', \quad (5)$$

and H is a good hash function. For each vote (2) Mallet finds α using equality (4). Then he can match the exponents α and numbers r_s by equality (5). Of course, the exponents α used in (3) as well as S have to be coprime with $\text{ord}g$. One can easily see that most α fulfill these conditions.

Betraying Voters Preferences via Ordering on the Ballot. Note that the voting ballot is constructed after a voter made her choice. We show that a kleptographic channel can be created, if an implementation allows to permute at random onions O_1, O_2, O_3, O_4 to be placed on the voting ballot. It can carry $4!$ messages and point, for instance, to the choice of the voter or leak secret keys of the VM (which would allow to prepare votes outside the VM).

Assume that Mallet has a public key y_M and a private key x_M such that $g^{x_M} = y_M$. Let O_1, O_2, O_3, O_4 be the list of onions for the voting ballot after sorting them lexicographically. Assume that g^k is the second component of an RE-onion O_1 . Then VM computes $z = y_M^k$ and uses a few initial bits of $H(z, O_1, O_2, O_3, O_4)$ to determine a permutation π' on $\{1, 2, 3, 4\}$. Let π'' be the message-permutation on $\{1, 2, 3, 4\}$ to be hidden. Then the VM puts the onions on the voting ballot so that the i th onion gets position $\pi'(\pi''(i))$ for $i = 1, \dots, 4$. Note that Mallet is able to recover π'' . Indeed, he computes $z := (g^k)^{x_M}$, and then π' by its definition. Then finding π'' is straightforward. Note that a third party cannot find z and therefore π'' remains hidden.

Another trick is to use permutations π_L, π_R of rows on both sides of the virtual ballot. In the specification the permutation π_X depends on all columns on the side X , and thus its validity cannot be verified on the control ballot, where one column from side X is missing. Hence π_L, π_R itself might have any form convenient for Mallet. For example α from (3) might be a compressed point P^β (cf. [18]) of an elliptic curve E which is a part of Mallet's public key now defined as (E, P, Y_M) , where $Y_M = P^{x_M}$. As a result Mallet might determine $\pi_X = H(X, Y_M^\beta)$ on the basis of α , and then might check with help of the hash ballot whom the vote has been cast for. Interestingly, in the above case of Diffie-Hellman protocol it suffices, instead of the point P^β , to pass only its x -coordinate (see [11]).

Selling Votes via Random Parameters. "Random" numbers r_s, q, r_L, r_R can be used in a malicious way; recall that the way of generating them is not controlled in the

protocol. Let y_M, y be components of DSA public keys of Mallet and the VM respectively, with appropriate private keys x_M, K' . As a result, the key $K^* = y_M^{K'}$ for a symmetric encryption scheme might be established according to the Diffie-Hellman protocol and the numbers r_s, q, r_L, r_R might be ciphertexts addressed to Mallet. Another option to generate K^* is to use α from the previous paragraph.

Changing Votes Cast by RM. Now we assume that a RM cooperates with a malicious VM. The method taking advantage of a permutation on the voting ballot can be used to transfer the secret keys K, K' from the VM to the RM in a kleptographic way. Also, the VM may generate the parameters q through a pseudorandom generator with a secret seed. Again, the seed can be transferred to the RM in a kleptographic way.

Even if the onions on voting ballots are not permuted in step 5 of the scheme, it is still possible to transfer one bit per ballot using permutations π_L, π_R . Because $\pi_{vb} = id$ now, then the onions from each exemplary pair $(B_1^R, B_2^R), (I_1^R, I_2^R)$ are not separated, and the order of the pairs on the voting ballot is determined by their order on the virtual ballot. Hence if bit 1 will be transmitted, each $\pi_X, X \in \{L, R\}$, might be determined according to ascending order of the values y_M^k for k from (1), where (1) are the second onions in consecutive rows on side X . When bit 0 need to be sent, descending order is created. Because exponents k on both sides of the virtual ballot are different, the names of candidates look to be permuted randomly and independently on each side.

Once RM got the keys it can modify the votes. It is facilitated by the fact that voting ballots are presented to the RM in about the same order as they are generated. Having appropriate q , the RM tries to “open” the onions containing the votes from the voting ballot. That is, for a given q , for each $i \in \{1, 2\}, X \in \{L, R\}, C \in \{B, Y\}$ the RM computes k used to construct an onion, Then it computes g^k and checks which onion has this number as the second component. If it is so, then the RM can retrieve the plaintext encoded by the first component by dividing it by $(\prod_{j=1}^{\lambda} y_j)^k$. Then the RM can replace the discovered onions containing a vote by a pair of new onions with a different choice – this is possible, since the RM has the necessary keys. Note that the replacement is done yet before the pools close, and without any cooperation with tallying authorities.

4 Chaum’s Visual Voting Scheme

Description of the Scheme. Due to space limitation we describe this scheme only briefly (for more details see [2]). The system consists of: VMs, *tallying authorities* (mixes) and a \mathcal{BB} . Let us sketch the voting procedure:

1. On a VM, a voter chooses a monochrome picture with the name of the candidate chosen. This *ballot image* is encoded as a matrix B of pixels.
2. The VM deterministically computes pseudo-random binary matrices W^t and W^b based on deterministic signatures $s^t(q)$ and $s^b(q)$ respectively, where q is a ballot serial number. Then it determines L^t and L^b based on B, W^t and W^b so that $L^t \otimes L^b = B$, and it is possible to obtain an image of the vote from L^X and W^Y for $X \in \{t, b\}, Y \in \{t, b\} \setminus \{X\}$. Namely every second bit of L^X is from W^X , the other half of bits is denoted by R^X , and $B^X = R^X \otimes W^Y$, where B^X is composed from every second bit of B . Separately, each of L^t, L^b gives no information about B ,

just like one-time-pad. Each image L^X , $X \in \{t, b\}$, will be printed on a transparent layer, and both layers X will be laminated together during the print.

3. The VM provides 4-tuples: $\langle L^b, q, D^t, D^b \rangle$ for the *bottom layer* and $\langle L^t, q, D^t, D^b \rangle$ for the *top layer*, where D^Y (based on $s^Y(q)$) for $Y \in \{t, b\}$, are deterministic onions containing information necessary to decrypt W^Y , and hence to obtain B^X from one layer X only (recall that B^X is a subset of pixels of $L^X \otimes W^Y$). Each 4-tuple is printed on a separate layer.
4. The voter verifies that both layers encode the ballot image $B = L^t \otimes L^b$ and the last three components of the 4-tuples are identical on both layers.
5. The voter either aborts (i.e. if verification fails), or selects the top or the bottom layer. Henceforth, the selected layer shall be denoted by X .
6. The system makes two digital signatures and provides them as a tuple:

$$\langle s^X(q), o^X(L^X, q, D^Y, D^X, s^X(q)) \rangle, \quad (6)$$

where o^X is called *overall signature*.

7. The voter separates two layers. The layer $Y \neq X$, unselected in step 5, will be shredded by a poll worker (we call him *Shredder*). Its digital counterpart in VM's memory will be destroyed as well. The layer X is a receipt for the voter. Its electronic version is used for vote counting purposes.
8. The voter can perform a consistency check to ensure that the digital signatures of the tuple are correct, i.e. $s^X(q)$ correctly determines D^X and the half of the pixels of L^X . He can also check that his vote is included in the receipt batch.
9. At the end of the election day the VM's receipt batch is transferred to the \mathcal{BB} (we assume that there is no additional subliminal channel, so e.g. the ordering of receipts is deterministic).
10. Later the receipt batch is mixed and partially decoded by successive trustee-operated mixes. When the original images are revealed the election results are calculated.

Attacks on Serial Numbers. The description of the scheme [2] does not specify how the serial numbers q are created. We show that if it is admitted that the serial numbers are random, then we can install a kleptographic channel through which a VM can betray its signing key to an arbitrary party observing the first \mathcal{BB} . This is a threat not included in [6] in the list of potential weaknesses of the scheme.

Namely, let N_M, e_M be a public RSA key of Mallet ("public" in the sense that VM knows that it is the key of Mallet). The VM will transmit not the secret signing key k , but $z = k^{e_M} \bmod N_M$. Therefore nobody but Mallet will be able to recover k . Note that once Mallet obtains the signing keys, he will be able to buy votes (from q and (6) he can reconstruct the vote cast) or to make fake votes and claim that election results are manipulated. We assume also that VM and Mallet share a secret key s (this key will not be sufficient to recover k).

In order to create a vote the VM chooses q until it finds a proper one encoding some digit of z expressed in radix α system (for example $\alpha = 2^2$). Let $\ell = \lfloor \log_\alpha N_M \rfloor + 1$. Hence for the current q the VM takes the first $\lfloor \log_2 \ell \rfloor + 1$ bits of $H(q, s)$ as an index i of the digit z_i , where $z = (z_{\ell-1}z_{\ell-2} \dots z_0)_\alpha$. Then the VM compares z_i with the last $\lfloor \log_2 \alpha \rfloor + 1$ bits of $H(q, s)$. If they disagree, then q is discarded. Recovering the digits of z is a random process. Each q appearing on \mathcal{BB} brings information on some α -ary

digit of z . Notice that the use of Mallet's key does not change the official behaviour of the VM, which is still consistent with the primary protocol.

Leaking a secret through the \mathcal{BB} might be possible even if numbers q published on the \mathcal{BB} after closing the polls must turn out to be consecutive. The point is that VM may use the numbers q in a non-consecutive way (at least at the beginning of the election day). Each voting ballot will carry one bit of the secret. Due to a limited number of voters perhaps only a part of the secret could be transferred in such a way, but this still poses a threat — mind the lattice attacks, which are practical, if a fraction of the RSA secret exponent is known to an attacker [1].

Let us describe the general idea. The image printed on each side depends on q and the picture B of a candidate chosen by a voter. Namely, the signature $s^Y(q)$ of VM under q determines W^Y , which together with B^X , i.e. the half of the picture B , determines R^X , i.e. the half of L^X . We shall fix some position (i, j) on R^t . It follows from the encoding scheme that on L^b a pattern of two black squares is printed - their positions at $W_{i,j}^b$ depend on q only, and not on the voter's choice. On the second layer at position $R_{i,j}^t$ VM prints the squares in exactly the same way - if the image B is white at (i, j) , or the mirror image, if B is black at this place. In the second case by superimposing both transparents we get a black spot on B , while in the first case we get a spot with black and transparent pixels. We fix position (i, j) so that for about half of the votes at this point should be a black spot. Thus there are two possible configurations of the squares at the position $R_{i,j}^t$ - they encode one bit. The point is that this bit is determined by q and the choice of a candidate C . So when a voter casts a vote by choosing the picture B_c , VM takes the first number q such that the bit $q - 1 \pmod{\ell}$ of the ℓ bit secret is not transferred yet onto \mathcal{BB} , and B_c together with q encodes on $R_{i,j}^t$ the value of this bit. If we are lucky that the voter chooses the upper layer, then a bit of a secret will be transmitted. (A slight change of the protocol ensures that there will be no strategy for the voter to choose a layer that does not leak the secret.)

The attack is probabilistic in the sense that we have no control which candidates will be chosen by the voters. However, two popular candidates C would suffice (their pictures B_c must differ at some position (i, j)) to transfer onto \mathcal{BB} some portion of the first bits of the secret. Moreover, the above method might be used to indicate to Mallet whether the vote is cast for some particular candidate C ($R_{i,j}^t = 1$) or not ($R_{i,j}^t = 0$).

In the next attack Mallet shall cooperate with Shredder, who observes the ordering of serial numbers q on unselected layers that he is given to shred. To make the attack more realistic we assume that Shredder is able to remember for a short time only two last digits of q (after a while he can write them down somewhere). On a voting ballot the serial number is printed over the bar-code representing it, and is expressed in base b system.

Consequently, Shredder is able to completely describe any permutation of any b^2 consecutive numbers. Thus the VM might transmit digits of z mentioned above, where z is expressed in radix $(b^2)!$ system (detailed method of coding numbers into permutations and vice versa can be found in [8]). To transfer 2048 bits in case of $b = 16$ one permutation from S_{256} and one permutation from S_{76} are enough, when for $b = 10$ three "digits" from S_{100} and one from S_{93} are needed.

Countermeasures. The attacks described above become impossible, if deterministic sequence of issuing serial numbers is guaranteed. A simple and effective solution we

propose is to link subsequent receipt by “linear-linking” similar to the one used in a time-stamping system (see [5]). Let \mathcal{L}_i be the i th receipt issued (6), and let the next serial number should be $q_{i+1} = h(\mathcal{L}_i)$, where h is some collision free hash function. We may start with \mathcal{L}_0 equal to the serial number of the VM. In this procedure VM learns the serial number q_{i+1} only after step i .

Of course, linking does not solve all problems with VMs. A user may transmit a secret code on the touch-screen or may select and cancel successive candidates from a secret sequence. This might trigger a mechanism in which the VM makes itself voter’s choices and the colluding voter simply collects a valid receipt, which indicates one of $2n$ possibilities (the choice of a layer and a candidate). The set of ℓ such voters obtains one of $(2n)^\ell$ possible messages. Moreover, if a choice of a candidate is cancelled no signature is printed on the layers (so the receipt will be impossible to verify without all tallying authorities), but the printout can already contain a ciphertext of the secrets.

Note also that to prevent homomorphic attack [4] one should avoid naive implementation of a signature scheme $s^X(q)$. If for example the RSA signatures $s^t(q_1)$, $s^t(q_2)$ are available on \mathcal{BB} , then $s^t(q)$ for $q = q_1^{\alpha_1} q_2^{\alpha_2}$ and any $\alpha_1, \alpha_2 \in \mathbb{Z}$ can be easily calculated by an attacker. Accordingly, any receipt (6) with such a number q and $X = b$ might be opened.

5 The Neff’s Scheme

Scheme Description. Below we follow a draft description [13]. The voting infrastructure includes voting machines (to be consistent with other subsections we call them VM instead of *DRE* [6] or *voting device* [13]), a \mathcal{BB} , and a verifiable mix-net.

The encoding techniques used are as follows: let ℓ be a security parameter ($10 \leq \ell \leq 15$), a *verifiable choice* (VC) is a $n \times \ell$ matrix of *ballot mark pairs* (BMP). Each row of VC represents a single candidate. A BMP is a pair (b_L, b_R) of ElGamal ciphertexts $(b_X = (g^{\omega_X}, m_X y^{\omega_X}), X \in \{L, R\})$, where pair (g, y) is a public key for the mix-net, $m_L, m_R \in \{Y, N\}$, and symbols Y, N represent, respectively, a fixed element $G \in \langle g \rangle$ and the group’s $\langle g \rangle$ neutral element. Each BMP in the row representing the candidate chosen contains two ciphertexts of the same symbol - i.e. $(m_L, m_R) \in \{(Y, Y), (N, N)\}$. All other rows of VC contain BMPs with $(m_L, m_R) \in \{(N, Y), (Y, N)\}$.

Each BMP (b_L, b_R) in can be partially *opened*, according to a bit ϵ . If $\epsilon = 0$, then VM reveals the plaintext of b_L by showing the random exponent ω_L . If $\epsilon = 1$, then the right ciphertext is opened.

Finally we might describe the voting procedure:

1. VM shows a list of all candidates C_1, C_2, \dots, C_n to the voter.
2. The voter chooses a candidate C_i .
3. Let $S = \{0, 1\}^\ell$ be the set of all ℓ -bit strings. VM prepares a VC representing a vote for the candidate C_i . It chooses $x_j \in S$ for $1 \leq j \leq n$ at random. For $j \neq i$, if $(x_j)_k = 0$, then VM encrypts (N, Y) in the j th row and the k th column of VC. If $(x_j)_k = 1$, then (Y, N) are encrypted. If $(x_i)_k = 0$, then the k th BMP in the i th row contains (N, N) , otherwise (Y, Y) . Next the VM commits prepared VC by

printing it or its hash on a receipt with a *ballot sequence number* BSN (BSN is present in the documentation of the *VoteHere* project based on the scheme).

4. The voter chooses strings $c_j \in S$ for $j \neq i$.
5. The VM computes *pledges* $p_j := c_j \oplus x_j$ for $j \neq i$ and $p_i := x_i$. VM commits sequences of strings $\{p_k\}_{k=1}^n$ in such a way that they cannot be changed but the voter gains no knowledge of the pledges.
6. The voter chooses c_i - a challenge to the row representing the candidate C_i .
7. For $j = 1, 2, \dots, n$ and $k = 1, 2, \dots, \ell$, VM opens BMP of VC in the j th row and the k th column according to bit $\epsilon = (c_j)_k$ as it was described before. VC with opened BMPs is called *opened verifiable choice* (OVC).
8. Values $\{(C_j, c_j)\}$ for $1 \leq j \leq n$ are printed on the receipt.
9. The voter gets the receipt containing BSN, the hash of VC, and (C_j, c_j) for $j = 1, 2, \dots, n$.
10. After closing the polling station OVC is sent to the \mathcal{BB} together with the associated BSN.

All ballots collected on the \mathcal{BB} are then processed according to a *verifiable shuffle protocol* [14].

Attacks on the Scheme. Karlof et al. [6] describe a “random subliminal channel attack” on the Neff’s scheme. They suggest to use the same ω for both encryptions in a given BMP. Since only one ciphertext per BMP is opened on OVC, the VM can send $n\ell \cdot \log_2(\text{ord}g)$ bits in such a channel. Of course, usage of the same ω clearly indicates that the two exponents in the BMPs are not randomly chosen. We repair this shortcoming in a kleptographic way and extend the attacks to N votes simultaneously.

According to the documentation [12] of *VoteHere* project, the numbers BSN are “unpredictably assigned to voters” (if there were no BSNs we would assume $N = 1$). The way of the BSNs assignment might be known to Mallet, hence he would be able to find on the \mathcal{BB} all consecutive N -tuples of ballots issued by the VM.

Let $y_M = g^{x_M}$ is Mallet’s public key. By $\text{BMP}_{t,i,j}$ we denote the BMP in the i th row of the j th column on the t th ballot. Let g^{ω_L} and g^{ω_R} be chosen at random as the first components of the ElGamal ciphertexts in the $\text{BMP}_{N,n,\ell}$ (this BMP is selected arbitrarily, e.g. $\text{BMP}_{1,2,1}$ could be taken as well). Let

$$K_X^* = h_X(y_M^{\omega_{\sigma(L)}}, y_M^{\omega_{\sigma(R)}}) \quad (7)$$

for $X \in \{L, R\}$, where $\sigma : \{L, R\} \rightarrow \{L, R\}$ is a permutation such that $g^{\omega_{\sigma(L)}} \leq g^{\omega_{\sigma(R)}}$ and h_L, h_R are some good one-way functions.

Having calculated the keys K_L^*, K_R^* the VM might prepare the exponents for creating BMPs. Let $\omega_{t,i,j}$ be a block of a message that has to be hidden kleptographically in $\text{BMP}_{t,i,j}$. Then the exponents used for creating $\text{BMP}_{t,i,j}$ are $E_{K_L^*}(\omega_{t,i,j})$, $E_{K_R^*}(\omega_{t,i,j})$, for $t = 1, \dots, N$, $i = 1, \dots, n$, $j = 1, \dots, \ell$, and $(t, i, j) \neq (N, n, \ell)$, where E denotes a secure symmetric encryption scheme. To prevent repetition of the block-values $\omega_{t,i,j}$ for different triples (t, i, j) , compression of the whole message can be made (note that $\text{ord}g$ is a large number).

When the OVC’s are published on the \mathcal{BB} , Mallet can retrieve the secret messages. Namely, he reconstructs the key K_X^* from $\text{BMP}_{N,n,\ell}$ contained in OVC_N by putting

$g^{\omega_L}, g^{\omega_R}$ in ascending order, raising them to power x_M and applying, respectively, h_L, h_R to the pair obtained. Since one of the exponents $E_{K_L^*}(\omega_{t,i,j})$ and $E_{K_R^*}(\omega_{t,i,j})$ is included in OVC, Mallet can decrypt it and obtain $\omega_{t,i,j}$.

If VM is forced somehow to use really random exponents for ElGamal encryption, then it is also possible to hide the choice of the voter. Namely, the strings ω_L, ω_R used for encryption in a fixed BMP are discarded until the system provides a number such that $[H(y_M^{\omega_L}, y_M^{\omega_R}) \bmod n] + 1 = i$, where C_i is the candidate chosen by the voter. Then the choice of the voter can be easily detected by Mallet, while for anybody else the information encoded in $g^{\omega_L}, g^{\omega_R}$ is impossible to retrieve.

Additional kleptographic channels might be mounted thanks to BSN numbers assigned to ballots. It is reasonable to assume that each VM has some scope of at most N_{\max} numbers BSN. If the method of issuing the numbers is not specified, then a VM may release BSNs in a manner that additionally hides $\omega_{t,i,j}$. Suppose that Mallet knows the BSN_1 . Then BSN_t may be calculated from BSN_{t-1} as the $(r+1)$ st yet unused “random” number from the scope, where

$$r = H(y_M^{E_{K_L^*}(\omega_{t-1,n,\ell})}, y_M^{E_{K_R^*}(\omega_{t-1,n,\ell})}) \bmod (N_{\max} - (t - 1))$$

for some good one-way function H . Again, only Mallet would be able to recover the order of the ballots issued. Furthermore, instead of using the same pair (K_L^*, K_R^*) of keys to all $\omega_{t,i,j}$, distinct pairs of subkeys could be used: for example $K_{X,t,i,j}^* = f(K_X^*, t, i, j)$ for some function f and $X \in \{L, R\}$. Moreover, the argument t may be replaced by a kind of linear linking [5] of the values $BSN_t, BSN_{t-1}, \dots, BSN_1$.

Another source of attacks are the numbers x_j used by the protocol. They are useful for our purposes for instance at the moment when the flaws related to random exponents become patched. Note that the numbers x_j are shown by OVC’s. Indeed, if $j \neq i$ where C_i is the candidate chosen by the voter, then we can reconstruct $(x_j)_k$ for $k \leq \ell$ as follows. If the b_L is opened and it contains N , or b_R is opened and it contains Y , we have $(x_j)_k = 0$. Otherwise, $(x_j)_k = 1$. In the case of x_i the above rule provides flipped values, for the k ’s where b_R is opened. VM may choose all bit-strings x_j according to $H(y_M^{\omega_L}, y_M^{\omega_R})$ for some $g^{\omega_L}, g^{\omega_R}$ being the first components of ElGamal ciphertexts of some BMP at established position. Then with probability $1 - 1/2^\ell$ we can detect the index i where the x_i computed according to the rule disagrees with the value obtained from $H(y_M^{\omega_L}, y_M^{\omega_R})$, and so the choice of the voter.

Let us remark that it is possible to build yet another kleptographic channel. Again, suppose that BSNs are used (if not, then take $N = 1$). VM determines the exponents ω_j in advance, for $j = 1, 2, 3, \dots, 2n\ell N$, computes g^{ω_j} and sorts them. Then VM encodes a secret message as an ordering in which permuted numbers g^{ω_j} are used in consecutive ciphertexts. So there is room for $(2n\ell N)!$ messages. To make Mallet the only addressee of the message the VM determines a permutation $\pi' = H(y_M^{\omega_1}, y_M^{\omega_2}, \dots, y_M^{\omega_{2n\ell N}})$, where the arguments of H are ordered lexicographically, and instead of a permutation π'' encoding the secret message VM can order the numbers g^{ω_j} according to the permutation $\pi = \pi' \circ \pi''$.

Finally, note that if the scope for BSNs is large enough, then the BSNs may carry messages — we may apply the same method as in Section 4.

6 Chaum, Ryan, Schneider's Scheme

Short Description of the Protocol. For the sake of simplicity, the authors of [3] illustrate a single race with v candidates. The scheme includes: RMs, \mathcal{BB} and k tellers ($k \geq 3$). Each of the tellers operates two Chaum's mixes, and the i th mix, $i = 0, 1, \dots, 2k-1$, has a pair of keys: a secret key SK_{T_i} and the corresponding public key PK_{T_i} . There is also an authority which is responsible for generating the ballots.

For each ballot the authority prepares a seed which is a random number D_0 and a sequence of $2k$ random values g_i taken from the set $\{0, 1, \dots, 2^{32} - 1\}$. Let us define

$$D_{i+1} := \{g_i, D_i\}_{PK_{T_i}}, \quad \theta_i := \left(\sum_{t=0}^i h(g_t) \right) \bmod v,$$

where h is a good hash function. In this way the authority computes an onion D_{2k} and a cyclic offset $\theta = \theta_{2k-1}$. A voting card consists of two columns: in the left column there is a list of the candidates after applying a cyclic shift by θ positions, the last cell of the column remains empty. All but the last of the cells of the right column are empty (one of them will be chosen by a voter). In the last cell of the right column the code of D_{2k} is printed.

In the voting phase a voter selects a voting card at random. She marks her candidate by making a sign \times in the appropriate cell on the right side, just as for the traditional voting procedures. Let r_{2k} be the index of a cell with mark \times , $r_{2k} \in \{0, 1, \dots, v-1\}$, and define $r_i := r_{i+1} - h(g_i) \bmod v$. Note that $r_0 = r_{i+1} - \theta_i \bmod v$ for any $i \in \{0, 1, \dots, 2k-1\}$, where r_0 is the cell reflecting the voter's choice on non-shifted list of candidates. Then the voter detaches the left and the right hand side, the left hand side is destroyed, the right hand strip, containing (r_{2k}, D_{2k}) , is fed into the RM. The RM marks the strip as being used and returns it to the voter as her ballot receipt.

Later, during tallying phase the i th mix transforms (r_{i+1}, D_{i+1}) into (r_i, D_i) using its private key SK_{T_i} for a deterministic encryption scheme. The input and the intermediate values of the tallying process are presented on \mathcal{BB} .

Attack on the Random Seed. We assume that the authority uses a secret hash function H (a hash function with a secret key) and for every ballot repeats the following procedure:

1. randomly selects D_0 and germs g_0, \dots, g_{2k-1} ,
2. deterministically computes onions D_1, \dots, D_{2k} ,

until the *Collision Condition* (CC) holds for $j = 2k - 1$:

$$CC_j : \quad h(g_j) + \theta_{j-1} \bmod v = H(D_{j+1}) \bmod v,$$

i.e. until $H(D_{2k}) \bmod v = \theta_{2k-1} (= \theta)$.

Then the authority outputs a card with the offset θ and the onion D_{2k} . Obviously, the CC_j might be applied for any $j = 2k - 1, \dots, 1$ (for $j = 0$ it is useless). Hence the teller who is operating the j th mix server and knows hash function H will be able to

compute vote values, i.e. r_0 . Accordingly, if the teller recognize partial election results as unfavorable, a DoS attack may be launched.

Note that for the attack exploiting CC_{2k-1} , everyone who knows the secret hash function H and sees a voter's ballot receipt, gets immediately knowledge about the voter's choice. Especially, it concerns members of the commission at the polling station and the RM.

As one can easily see, the expected number of tries needed to find the collision in CC_{2k-1} is equal to v . The same complexity bound applies for an attack on the extension of the scheme, where θ is a permutation, not an offset. If the authority knows the base ordering, i.e. the order of candidates on the list not permuted yet, then it is able to point out the position of the supported candidate on the list permuted according to θ_{2k-1} . The D_{2k} is accepted, when $H(D_{2k}) \bmod v$ indicates the same position as θ_{2k-1} . Hence anyone who knows H is then able to point out the cell where the sign \times should be put for the candidate supported. It is easy to see that in more general case of elections which allow to vote for u out of v candidates the complexity of a single ballot preparation grows to $\frac{v!}{u!(v-u)!}$ trials on average, or to $\frac{v!}{(v-u)!}$ if a voter must rank chosen candidates, and the ranking is also important for an attacker.

Suppose now that the attack is launched on some fixed layer j , and H is a keyed hash function with secret key K^* . Then K^* might be transmitted kleptographically encoded in D_{j+1} , the same value that satisfies the condition CC_j . Namely assume that Mallet possesses a public key of ElGamal elliptic curve cryptosystem. Then the length of the ciphertext z containing K^* equals about 340 bits if the points are compressed. Let us consider a secret hash function H' , which is known to Mallet, and the value $H'(D_{j+1})$. Suppose that z is expressed in radix α system and let $\ell = \lfloor \log_\alpha(2^{340} - 1) \rfloor + 1$. Then the first $\lfloor \log_2 \ell \rfloor + 1$ bits of $H'(D_{j+1})$ indicate an index i of some digit z_i , where $z = (z_{\ell-1} \dots z_1 z_0)_\alpha$, and the last $\lfloor \log_2 \alpha \rfloor + 1$ bits of $H'(D_{j+1})$ should be equal z_i (if are not, then given D_{j+1} is discarded). On average, one out of α strings D_{j+1} properly describes one digit of z . Consequently, the average complexity of a single ballot preparation increases to $\alpha \cdot v$ trials. As one can see (cf. [9], "the occupancy problem"), for any $c > 1$ the number $c\ell \ln \ell$ of values D_{j+1} suffice to receive the complete z with probability at least $1 - (\frac{1}{\ell})^{c-1}$. Mind that the mix servers are supposed to operate much more than $2\ell \ln \ell$ ballots. Once Mallet gets z he decrypts K^* and then is able, on the basis of CC_j , read θ_j from all the D_{j+1} he obtained.

Changing Votes. The attack below seems to be problematic due to the number of cooperating parties, but fully explores the possibilities given by CC_{2k-1} .

Let us assume that a RM is cooperating with a fake or dishonest watch-dog organization (WDO) who collects receipts from voters. Let us assume that votes collected by WDO will not be checked by voters (they do not have receipts now). A WDO can pass information about onions collected to the RM and RM is now free to modify electronic vote representation, provided that the Mercuri method [10], considered as some possible extension of the scheme, is not implemented. If it is not, then spoiling election results of the most popular candidate is possible for RM even without knowing H : it may randomly change r_{2k} for votes collected by WDO.

Audit Procedure. Note that the ballots are formed properly, so the only possibility to catch cheating authority is to prove that the entropy used in the generation process is low. But let us observe that for any fixed j the sample space size for (θ_j, D_{j+1}) is larger than $(2^{32})^{j+1}$. Instead of sampling from the space of that size, the authority mounting the attack on onions D_{j+1} chooses (still independently at random) from the space which size is larger than about $(2^{32})^{j+1} / (\alpha \cdot v)$ (we have omitted the size of D_0). For reasonable $\alpha \cdot v$ this room is still too large to detect the fraud regarding the number of votes investigated during the audit phase.

It must be noted that to minimize the possibility of the above attacks in new, distributed procedures of ballot cards generation outlined in [17] and [16], the mixes (called *clerks*) should not be delivered from the same source.

7 Conclusions

A variety of attacks on the main election schemes have been proposed in [6, 17] and in our paper. Most of the attacks are possible because too much trust is put in a single party of the protocol, for example in a Voting Machine. We conclude that designs of schemes should try to avoid using randomness. Use of deterministic signatures and encryption schemes and Chaum's MIX like style of communication (with messages passed in lexicographic order) facilitates verification and reduces the possibility of existence of subliminal channels and kleptographic attacks.

References

1. Dan Boneh, Glenn Durfee, and Yair Frankel. An attack on RSA given a small fraction of the private key bits. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 1998.
2. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy Magazine*, 2(1):38–47, January/February 2004.
3. David Chaum, Peter Y.A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.
4. George I. Davida. Chosen signature cryptanalysis of the RSA public key cryptosystem. Technical Report TR-CS-82-2, Dept of EECS, University of Wisconsin, Milwaukee, 1982. Available from: <http://www.uwm.edu/~davida/papers/chosen/>.
5. Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.
6. Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, pages 33–50, 2005.
7. Marek Klonowski, Mirosław Kutylowski, Anna Lauks, and Filip Zagórski. A practical voting scheme with receipts. In *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 490–497. Springer, 2005.
8. Donald Ervin Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, Reading, Massachusetts, 3rd edition, November 1998.
9. Boris Koldehove. Simple gossiping with balls and bins. *Stud. Inform. Univ.*, 3(1):43–60, 2004.

10. Rebecca Mercuri. Government: a better ballot box? *IEEE Spectr.*, 39(10):46–50, 2002.
11. Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.
12. C.Andrew Neff. Detecting malicious poll site voting clients. [online]. September 2003 [cited 10 January 2006]. Available from: <http://www.votehere.com/vhti/documentation>.
13. C.Andrew Neff. Practical high certainty intent verification for encrypted votes. [online]. October 2004 [cited 10 January 2006]. Available from: <http://www.votehere.com/vhti/documentation>.
14. C.Andrew Neff. Verifiable mixing (shuffling) of ElGamal pairs. [online]. April 2004 [cited 03 March 2006]. Available from: <http://www.votehere.com/vhti/documentation>.
15. Ronald L. Rivest. Voting resource page. [online, cited 10 January 2006]. Available from: <http://theory.lcs.mit.edu/~rivest/voting/index.html>.
16. Peter Ryan. Socio-technical trade-offs in cryptographic voting schemes. Workshop on Electronic Voting and e-Government in the UK, 27th–28th February 2006. Slides. Available from: <http://www.nesc.ac.uk/action/esi/contribution.cfm?Title=639>.
17. Peter Y.A. Ryan and Thea Peacock. Prêt à voter: a systems perspective. Technical Report 929, University of Newcastle upon Tyne, School of Computing Science, September 2005. Available from: <http://www.cs.ncl.ac.uk/research/pubs/trs/papers/929.pdf>.
18. Scott A. Vanstone, Ronald C. Mullin, and Gordon B. Agnew. Elliptic curve encryption systems. US patent 6141420, October 2000. Available from: <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6,141,420>.
19. Clint Curtis affidavit. [online, cited 10 January 2006]. Available from: http://www.buzzflash.com/alerts/04/12/images/CC_Affidavit_120604.pdf
20. The e-voting machine fraud. [online, cited 10 January 2006]. Available from: http://www.linkcrusader.com/vote_machines.htm
21. The e-voting project web page. [online, cited 10 January 2006]. Available from: <http://e-voting.im.pwr.wroc.pl>.
22. Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 1996.
23. Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In *EUROCRYPT*, volume 1109 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1997.
24. Adam Young and Moti Yung. Bandwidth-optimal kleptographic attacks. In *CHES: International Workshop on Cryptographic Hardware and Embedded Systems, CHES, LNCS*, pages 235–250, 2001.
25. Adam Young and Moti Yung. Malicious cryptography: Kleptographic aspects. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 7–18. Springer, 2005.