Ballot stuffing in a postal voting system

Véronique Cortier, Jérémie Detrey, Pierrick Gaudry, Frédéric Sur, Emmanuel Thomé, Mathieu Turuani, Paul Zimmermann

Ballot stuffing in a postal voting system

Véronique Cortier*, Jérémie Detrey*, Pierrick Gaudry*, Frédéric Sur*, Emmanuel Thomé*,
Mathieu Turuani*, and Paul Zimmermann*
*LORIA & CNRS & INRIA & INPL

***Abstract*—We review a postal voting system used in spring 2011 by the French research institute CNRS and designed by a French company (Tagg Informatique). We explain how the structure of the material can be easily understood out of a few samples of voting material (distributed to the voters), without any prior knowledge of the system. Taking advantage of some flaws in the design of the system, we show how to perform major ballot stuffing, making possible to change the outcome of the election. Our attack has been tested and confirmed by the CNRS. A fixed postal voting system has been quickly proposed by Tagg Informatique in collaboration with the CNRS, preventing this attack for the next elections.**

I. Introduction

Voting systems have been invented a long time ago and are one of the keystones of democracy. They are used in many configurations with various issues, e.g., legislative or presidential elections but also elections of representatives of students, in unions and many other institutions. In France, the most current and well known voting system uses paper ballots, transparent ballot boxes and booth stations. This standard voting system has been continuously improved over the time and actually offers a good level of security guarantees. It however requires the voters to attend a polling station, which implies both some effort from the voters and expenses for running a polling station. Therefore, some voting systems also allow voters to vote from home. These systems are typically used for medium-size and mid-level elections. The simplest postal voting system consists of having each voter use two envelopes: the voter chooses his/her ballot and encloses it in a first (anonymous) envelope. He/she then encloses the second envelope with the address of the collector, this envelope being signed by the voter. While convenient, this system can be easily subject to fraud since malicious voters may vote on the behalf of other voters, especially in case they know some voters will abstain (as voters signatures are difficult to check). Moreover, there is no guarantee that the collector will not open both envelopes at the same time, therefore breaking voters' confidentiality.

About a decade ago, electronic voting protocols have been developed, allowing anyone to vote from any computer. Voting machines have also been used in several countries for national scale elections, replacing the old paper ballot system. Such voting machines however still require the voters to attend a polling station. While offering convenient and efficient recording and tallying of the votes, these new voting systems have been criticized and attacked. For example, a catalogue of vulnerabilities and possible attacks has been produced regarding the Diebold machines used in 37 US states [14] or the Diebold AccuVote-TS voting machine [9]. Recently, an electronic voting system about to be used in Washington DC has been attacked in only a few days [11], [20]. Consequently, these electronic voting systems are still under evaluation.

In order to allow voters to vote from their home without using computers, voting systems have been proposed to improve the standard "two-envelope" system. They consist of somewhat hybrid systems, still using paper ballots but with barcodes (to facilitate the tallying phase using barcode readers) and identifiers that should ensure that votes cannot be linked to voters. They are typically used for elections with intermediate issues such as elections of representatives in unions, companies or many councils. Up to our knowledge and surprisingly, these systems have not been submitted to a careful security analysis (nor even design), in contrast to electronic voting protocols. Some official guidelines nevertheless exist. For example in France, the *Commission Nationale de l'Informatique et des Libertés* (CNIL) — which is an independent administrative authority whose mission is to guarantee that data processing complies with human rights, private life, or individual freedoms — has recently issued recommendations about electronic voting [7]. A similar recommendation has been issued for postal voting with barcodes [6].

In this paper, we review such a barcode voting system and show that it suffers from a severe flaw that can be exploited to perform major ballot stuffing. Specifically, we study the postal voting system that has been used in spring 2011 by the French research institute CNRS (*Centre National de la Recherche Scientifique*) in an election involving about 30,000 voters (CNRS active and retired employees). The voting system was designed and deployed by the French company Tagg Informatique, contracted by the CNRS for the organization of this election. We show how a malicious user could perform ballot stuffing without even being noticed by the system. This relies on the fact that the voting material sent to voters contains predictable identifiers. The predictability of the identifiers can then be exploited to forge valid ballots. It is worth noticing that the discovery of the attack has been made without any prior knowledge of the system, simply by observing a few samples of the voting

material distributed to the voters. Moreover the attack can be performed with only a few straightforward computations. We have notified the CNRS and tested our attack by sending 50 (well identified) forged ballots, whose validity was later confirmed by the CNRS. Informed of our discovery, the Tagg Informatique company quickly changed its voting system, preventing our attack scheme from being exploited further.

We believe that our attack exemplifies the need for clear security requirements for voting systems, even for non electronic ones. Here the CNRS had insisted in its security requirements on the confidentiality and anonymity of the votes. Similarly, the recommendations of the CNIL French commission [6] mostly insist on the fact that it should not be possible to link votes to the voters. While prominent and rather obvious, voter eligibility (the fact that only eligible voters should be able to cast at most one vote) seems to have received insufficient attention.

Related work. There is an important amount of work on the design and security analysis of electronic voting systems in particular on voting machines [14], [9] or pure electronic voting protocols such as Helios [1], Civitas [12], [5], or some more theoretical ones [18], [10], [15]. By contrast, we focus here on systems that do not require voters to use a computer. ThreeBallot [19], Scantegrity [3] and Prêt à Voter [4] are the three main examples of voting schemes that are purely paper-based (from the user's point of view). They however require the voters to attend physical voting stations with external authentication processes (such as ID card or voter card). In the system we study, it is an important feature that voters can vote from home.

Outline of the paper. We provide an informal description of the system in Section II, as it can be observed by any voter. In Section III, we explain how the structure of the identifiers can be understood out of some ballot samples, performing reverse engineering. Based on this reverse engineering phase, we explain how to perform ballot stuffing in Section IV. Further possible security flaws (e.g., regarding confidentiality of the votes) are discussed in Section V. We discuss which lessons should be learned from this case study in Section VI.

II. Description of Barcode Voting

The postal voting protocol at hand has been used by the CNRS in spring 2011, for a nationwide union representative election involving about 30,000 voters. The election was a *list system*: the number of affiliated candidates being elected depends on the number of votes that the list receives, each list corresponding to a different trade union. Let us describe the voting process from the candid user point of view. Each voter receives the documents for the vote at home by post. These documents are made of directions for use, a detachable ballot (Fig. 1), and a sheet with the name of the lists running for the election together with a sticker for each one of them (Fig. 2). More precisely:

- The ballot displays a barcode with a 10-digit number. The directions for use claim that it is "*a random barcode which permits to sign the attendance sheet, while preserving anonymity and secrecy*." A place is left blank to affix the sticker corresponding to the selected list.
- The stickers for the lists display a barcode, with a 8-digit number.

To sum up, once the voter has made his/her decision, he/she affixes the corresponding sticker on the ballot (Fig. 3), and then mails the ballot inside an anonymized envelope.

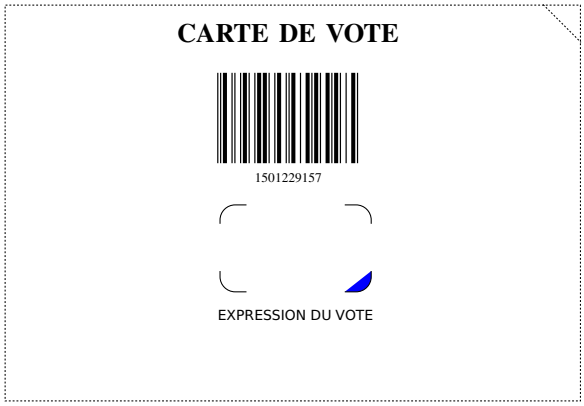

Figure 1. A blank ballot. The upper 10-digit barcode identifies the voter.

III. Reverse engineering

Although we eventually had to study a total of ten samples of voting material in order to guess what we believe to be, with quite some confidence, most of the actual specifics of the voting system, we want to stress that a good deal of that knowledge was obtained after examining only a few samples, and that even a single sample gave away some critical information. This distinction is all the more relevant since the most limiting factor for an adversary to be able to carry out an attack is by far the number of samples he/she will be able to procure.

To illustrate this point, we detail in the following paragraphs how much understanding of the system can be progressively guessed from the knowledge of one, then a handful, and finally ten samples of voting material.

A. From one sample of voting material

1) Common prefix: Taking the example of the voting material presented in Figs. 1 and 2 (sample #1 in Appendix A), one can immediately remark that the 10-digit voter identifier (1501229157) and all of the seven 8-digit list identifiers (15010394, 15011485, and so on) start with the same 4-digit sequence 1501. Whether this sequence is specific to this sample only or to the whole election is unknown at this stage, but we can nonetheless focus our first study on the remaining digits of the identifiers.

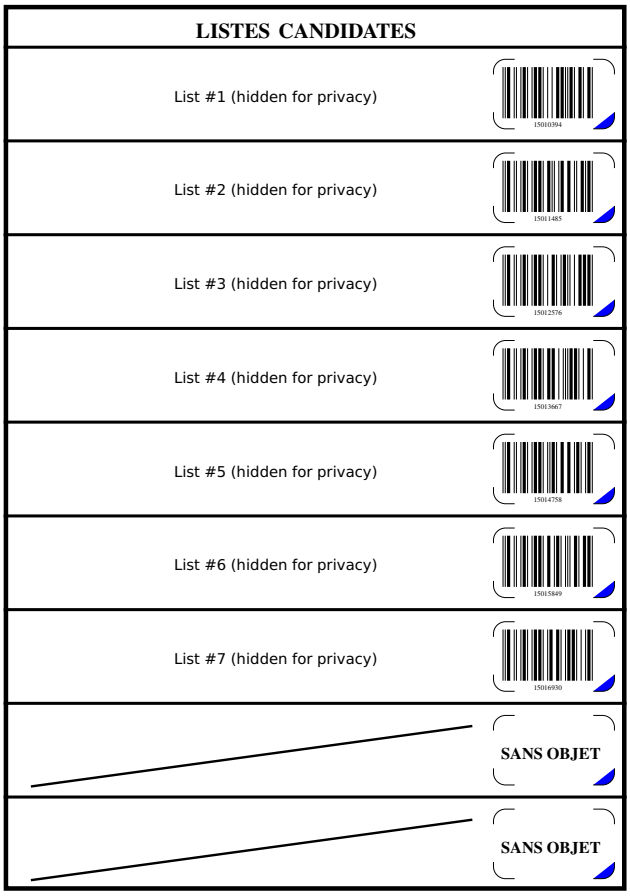

Figure 2. The sheet with the names of the lists running for the election (hidden here for privacy). A sticker with an 8-digit barcode is associated with each list.

2) List identifiers: Let us now consider the last 4 digits of the list identifiers: 0394, 1485, 2576, 3667, 4758, 5849, and 6930. The somewhat regular increasing of these 4-digit sequences, when seen as decimal integers, hints at a kind of arithmetic progression. Indeed, there appears to be a common difference of 1091 between the first 6 numbers. However, trying to extend this guess to the last one, one remarks that $5849+1091$ is 6940 and not the observed 6930: it seems that the first carry digit was not propagated to the second digit. This led us to believe that the last digit had to be considered separately, as would be the case for a check digit, for instance.

Ignoring this last digit, we now observe a perfect arithmetic progression on all the list identifiers, with common difference 109, from $L_1 = 039$ to $L_7 = 693 = L_1 + 6 \times 109$.

Alternatively, one might also look at the last 4 digits of the list IDs individually and remark that each of them forms an increasing sequence, except for the third one which is decreasing. In other words, noting $xyzt$ the last 4 digits of

Figure 3. A cast ballot. The 8-digit barcode for the chosen candidate list (here list #3) is affixed just above the mention "*Expression du vote*".

the list ID #i, then the list ID #$(i+1)$ is $x'y'z't'$, where:

$$\begin{cases} x' = (x+1) \bmod 10, \\ y' = (y+1) \bmod 10, \\ z' = (z-1) \bmod 10, \text{ and} \\ t' = (t+1) \bmod 10. \end{cases}$$

This conjecture is however quickly disproved as soon as a borrow or a carry propagation occurs, as can be observed in samples #2, #4, #6, #8, #9, and #10 in Appendix A. An easy exhaustive enumeration over all possible sets of list IDs (see Appendix C1) shows that the probability for such a carry to appear in a sample of voting material is actually over 81%, which rapidly allows one to find a counter-example.

B. From three or four samples

The knowledge of several samples confirms most of the guesswork that could be obtained from one single sample:

- the 4-digit prefix 1501 is common to all voter and list IDs, and
- out of the 4 last digits $xyzt$ in list IDs, the first 3 (xyz) follow an arithmetic progression of common difference 109, while the last one (t) seems to be acting as a check digit.

1) Check digit: With a few samples available, one can now start investigating how this check digit t is computed. Since it can apparently take any value between 0 and 9, a first idea is that this digit is the remainder modulo 10 of a linear combination of the preceding digits:

$$\begin{aligned} t &= \text{check\_digit}(1501xyz) \\ &= (\delta + x\lambda_5 + y\lambda_6 + z\lambda_7) \bmod 10, \end{aligned}$$

where $\delta = 1\lambda_1 + 5\lambda_2 + 0\lambda_3 + 1\lambda_4$ stands for the constant contribution to the check digit due to the prefix 1051. Putting together the observed list IDs and check digits of n samples therefore yields a system of $7n$ linear equations modulo 10 in the four unknowns δ, λ_5, λ_6, and λ_7. The probability

of this system being of rank 4 modulo 2 and 5 (the two prime factors of 10) is about 27% with $n = 1$ sample, 75% with 2 samples, 94% with 3, and almost 99% with 4 (see Appendix C2).

Solving this system of modular equations gives the check digit coefficients $\delta = 0$, $\lambda_5 = 7$, $\lambda_6 = 5$, and $\lambda_7 = 1$. In other words, we have

$$t = \text{check\_digit}(1501xyz) = (7x + 5y + z) \bmod 10.$$

Note that the system obtained from the first 4 samples in Appendix A is only of rank 3 over $\mathbb{Z}/2\mathbb{Z}$: even though one ends up with two acceptable sets of check digit coefficients instead of a unique solution as above, this is still enough to forge correct check digits with probability $1/2$.

2) Voter identifiers and check digits: Let us now focus on voter IDs. Since they are also barcodes, one can extend the assumption made for list IDs and consider the last digit of voter IDs to be a check digit. We therefore denote the last 6 digits of voters IDs as $abcdek$, where $abcde$ is assumed to be the actual voter ID, while k would be the corresponding check digit. Using the same approach as for the list IDs, one can express the value of this digit as a linear combination of the preceding digits:

$$\begin{aligned} k &= \text{check\_digit}(1501abcde) \\ &= (\epsilon + a\mu_5 + b\mu_6 + c\mu_7 + d\mu_8 + e\mu_9) \bmod 10, \end{aligned}$$

with $\epsilon = 1\mu_1 + 5\mu_2 + 0\mu_3 + 1\mu_4$ being the contribution due to the prefix 1501.

Since each sample of voting material yields only one such modular equation, finding the value of the six unknowns ϵ and μ_5 to μ_9 requires more than just 3 or 4 samples. However, one can make an educated guess and suppose that the check digit algorithms for voter and list IDs have good chances of being the same. In other words, we can assume that the sequence of check digit coefficients is either (a) left-aligned, and that $\epsilon = \delta = 0$, $\mu_5 = \lambda_5 = 7$, $\mu_6 = \lambda_6 = 5$, and $\mu_7 = \lambda_7 = 1$; or (b) right-aligned, and that $\mu_7 = \lambda_5 = 7$, $\mu_8 = \lambda_6 = 5$, and $\mu_9 = \lambda_7 = 1$.

Once again, only a handful of samples of voting material are necessary to disprove assumption (a): given n samples, trying to solve the system of n linear equations of the form

$$7a + 5b + c + d\mu_8 + e\mu_9 \equiv k \pmod{10}$$

will result in an empty solution space with probability 15% with only $n = 1$ sample, 45% with 2, 92% with 3, and 99% with 4 samples (see Appendix C3).

Similarly, assumption (b) can be trusted with a reasonable level of confidence seeing how it is satisfied by all the available samples of voting material: even though the confidence is zero with only one sample, it increases to almost 18% with 2, 49% with 3, and finally reaches 92% with 4 samples (see Appendix C3).

Having disproved assumption (a) while verifying that assumption (b) had good chances of being valid, one can now try and solve the resulting system of n linear equations of the form

$$\epsilon + a\mu_5 + b\mu_6 + 7c + 5d + e \equiv k \pmod{10}$$

in the three unknowns ϵ, μ_5, and μ_6. Using $n = 3$ samples, this system can be completely solved modulo 2 and 5 with probability 26%. With 4 samples, this probability increases to 56%, with an expected number of solutions of 2 (see Appendix C3). Although this is not enough to gain full understanding of the check digit mechanism, it is sufficient to know how to forge correct check digits for some chosen voter IDs, as discussed in Section IV. For instance, under assumption (b) and using the samples #1 to #4 given in Appendix A, one obtains the two following solutions:

$$(\epsilon, \mu_5, \mu_6, \mu_7, \mu_8, \mu_9) = \begin{cases} (2, 4, 2, 7, 5, 1), \text{ or} \\ (2, 9, 2, 7, 5, 1), \end{cases}$$

which allows one to forge valid check digits, as long as the digit a of the corresponding voter ID is chosen to be even.

3) Range of voter IDs: After looking at several samples of voting material, one can also remark that all voter IDs seem to fall in the range $\{0, \ldots, N\}$, where N is the number of voters taking part in this election (about 30,000), while the 5 digits of voter IDs would allow for a possible 100,000 values. The probability of this happening for all of the n available samples is therefore $(3/10)^n$, i.e., less than 1% for $n = 4$ samples. Hence, it seems reasonable to make the assumption that all the voter IDs in this election form a contiguous range between 0 and N.

When forging ballots, making this assumption should allow one to pick valid voter IDs with very high probability.

4) Link between voter and list IDs: Let us now consider the first two samples of voting material from Appendix A: their voter IDs — omitting the prefix 1501 and the check digit — are $V = 22915$ and $V' = 05166$, respectively. When seen as decimal numbers, the difference between the two is $V - V' = 17749$, which is congruent to 91 modulo 109. Now looking at the identifier of list #1 for these two samples, namely $L_1 = 039$ and $L'_1 = 057$, we also observe that $L_1 - L'_1 = -18 \equiv 91 \pmod{109}$. Pushing this investigation further, we find that

$$L_1 = (V \bmod 109) + 14 \text{ and } L'_1 = (V' \bmod 109) + 14.$$

This construction can then be checked for consistency with the other available samples.

Even though this may not be the only linear congruence which can relate the voter and list IDs of 3 or 4 different samples of voting material, the fact that this particular congruence happens modulo 109, which is also the common difference in the arithmetic progression of list IDs, is a very

good indicator that this is a correct assumption. Indeed, since the identifier L_i of list #i is computed as

$$L_i = L_1 + 109(i-1) = (V \bmod 109) + 109(i-1) + 14,$$

we have that $L_i \equiv V + 14 \pmod{109}$ for all lists. When tallying up the votes, this enables the organizer of the election to check very quickly that the voter and list IDs from a cast ballot are coherent, i.e., that they come from the same voting material.

C. From ten samples

The more samples of voting material one is able to collect, the higher the confidence one will have in the conclusions of the above reverse engineering. Using the 10 actual samples we secured, we were thus able to verify all the assumptions made in the previous paragraphs. With a level of confidence close to absolute certainty, we can now describe what we believe to be the full internal works of the voting system:

- 5-digit voter IDs V are taken from a contiguous range between 0 and N, where N is the number of voters (i.e., about 30,000);
- noting $abcde$ the 5 digits of V, the full voter ID is $1501abcdek$, where the check digit k is computed as
$$k = (4a + 2b + 7c + 5d + e + 2) \bmod 10;$$
- the 3-digit identifier L_i for list #i, $1 \leq i \leq 7$, is computed as
$$L_i = (V \bmod 109) + 109(i-1) + 14;$$
- noting xyz the 3 digits of L_i, the full identifier for list #i is $1501xyzt$, where the check digit t is computed as
$$t = (7x + 5y + z) \bmod 10.$$

IV. Ballot stuffing

A. Forging ballots

From the complete description of the voting system given in Section III-C, an attacker is able to forge up to N valid ballots for the list of his/her choosing with a probability of success of 100%: picking any voter ID V in the range $\{0, \ldots, N\}$ and constructing the corresponding list ID and check digits as explained above is guaranteed to produce a valid ballot.

However, gaining full knowledge of the voting system with high confidence requires procuring around 10 samples of voting material, which may not be possible in the context of an attack, especially since this would imply either (a) colluding with several voters, at the risk of one of them reporting the attack in progress to the organizer of the election; or (b) intercepting several samples of voting material, which is hardly practical at all since they are sent directly by post to the home address of the voters.

Nevertheless, as we showed in Section III-B, the knowledge of only 3 or 4 samples, along with a few reasonable assumptions, is enough to retrieve most of the specifics of the voting system and to forge ballots. Indeed, even if one cannot find the unique solutions for computing the check digits of the voter and list IDs, and ends up with m possible solutions instead, one is still able to forge at least N/m ballots by picking only the voter and list IDs on which the m solutions agree.

For instance, considering the first 3 samples from Appendix A, the linear system for computing the check digit t of list IDs of the form xyz is only of rank 3 modulo 2, and thus yields two solutions:

$$t = \begin{cases} (7x + \quad 6z) \bmod 10, \text{ or} \\ (7x + 5y + \ z) \bmod 10. \end{cases}$$

Following assumption (b) from Section III-B2, the check digit coefficients for the last 3 digits of voter IDs should also be either $(7, 0, 6)$ or $(7, 5, 1)$. However, the former is disproved by the voter IDs from samples #1 and #3, which allows us to conclude that only the latter is valid, and that $t = (7x + 5y + z) \bmod 10$. We can then solve the linear system for computing the check digit k of voter IDs of the form $abcde$ and obtain the two solutions

$$k = \begin{cases} (4a + 2b + 7c + 5d + e + 2) \bmod 10, \text{ or} \\ (9a + 2b + 7c + 5d + e + 2) \bmod 10. \end{cases}$$

Consequently, we just need to pick only voter IDs whose first digit a is even in order to successfully forge the corresponding check digit. Assuming that N is around 30,000, we are still able to forge around 20,000 valid ballots under this restriction.

Combining the expected number of solutions to each system as computed in Appendices C2 and C3, the expected number of valid ballots one is able to forge using knowledge from 3 samples of voting material is over 6,200, and over 14,500 using 4 samples.

B. Ballot stuffing attack

In order to assess the feasibility of a ballot stuffing attack using forged ballots as above, one also has to take into account the way duplicate ballots are detected and handled by the election organizer when tallying up the votes. Indeed, since each forged ballot corresponds to an actual voter, this voter might also have cast his/her vote, resulting in a duplicate ballot.

The recommendations of the CNIL on this matter [6] stipulate that "*the voting system should include a mechanism for rejecting a ballot which has already been processed*." One can then reasonably assume that this is the case for the election at hand, and that all forged ballots corresponding to voters who actually voted will be discarded. Since the voter turnout for such elections is usually pretty low, around 40% [2], this means that more than half of the forged ballots will be accepted and counted. This probability goes up to 80% if we assume that in the case of duplicates our forged

ballot has the same probability of being counted as the legitimate ballot.

Finally, we remark that the highest-ranking list in such elections usually receives around 25% of the votes, i.e., 3,000 out of the 12,000 cast ballots [2]. We can then conclude that the proposed ballot stuffing attack has a very good probability of changing the outcome of the election and bringing about the desired result, as it is able to generate at least 6,200 valid ballots, out of which between 60% (3,720) and 80% (4,960) will actually be taken into account in the tally.

C. Experiment

To convince the CNRS and Tagg Informatique of our attack, we have sent 50 (well identified) forged ballots, all casting a vote for the first list. These forged ballots have been scanned in a test phase before the actual tallying phase. We received confirmation that our forged ballots were considered to be valid by the system and would have been counted in the tallying phase.

V. Further potential weaknesses

Our findings led the company in charge of the preparation of the voting material (Tagg Informatique) to modify their procedures. From what we have been able to observe, the modifications carried out are twofold:

- The voter ID is two digits longer (7 significant digits), still accompanied with a 4-digit prefix identifying the election, and a check digit. The barcode for the voter IDs thus encodes $4 + 7 + 1 = 12$ digits.
- The list IDs are no longer prefixed with an election identifier. The 8-digit number encoded by the barcode in this modified version still has a check digit, thereby leaving 7 significant digits (instead of only 3 previously). We did not find any particular regularity in these numbers.

Four samples of such modified voting material are reproduced in Appendix B.

To have a guarantee of security, the system should choose the voter ID randomly. We make here the assumption that this is indeed the case, and that the list IDs are also generated randomly. However, the use of "random" data requires some caution. We wish to stress that the use of a *cryptographically strong* random number generator is necessary to thwart any attempt of ballot stuffing. We show that should this not be the case, then under plausible assumptions the complete list of voter IDs and list IDs can be retrieved, with very serious consequences.

A cryptographically strong (pseudo-)random number generator ensures that a sequence of generated numbers is indistinguishable from random noise. In particular, a probabilistic polynomial-time algorithm which tries to guess one bit output by the random number generator, based on the information of the complete previous output, has to succeed no more often than a coin flip.

Practical examples of pseudo-random number generators for which this strong hypothesis is believed to hold in practice can be obtained using essential building blocks such as hash functions and stream ciphers [16, Chap. 5]. Examples of random number generators which typically do *not* meet these requirements are those whose intent is originally to model randomness from a *statistical* point of view, which is a significantly weaker requirement. Most random number generators encountered in computer applications and programming languages essentially care about this second requirement only (e.g., most `C` programming language libraries implement a `rand()` function which is barely satisfactory from a statistical point of view, and certainly not suitable for cryptographic use).

In order to explain further potential weaknesses, let us assume that the generation of voting material is done with some pseudo-random number generator. The output of such a generator is entirely determined by the initial value (seed) of its *internal state*. When cryptographic strength is not a design requirement, pseudo-random number generators with an internal state of 50 bits or less are commonly encountered. This is the case, for example, of the pseudo-random number generator used by the Microsoft Excel program. Its `RAND()` function is described in [17] — following the design proposed in [21] — and relies on floating-point arithmetic. The internal state of this function is 45-bit wide. We consider that this function is likely to have been used for generating random values in the modified voting system under scrutiny.

In such a situation, assuming the attacker has the knowledge of the voting material for one or several voters, it is feasible to simulate the random number generator by exploring all the successive values taken by the internal state. For a poll with N voters and k lists, a sequence of $N(k + 1)$ values taken by the internal state determines the complete identifier database. Testing whether the known ballot values are encountered among $N(k+1)$ successively generated values is sufficient to gain confidence in the guessed initial internal state. Upon achievement of this complete regeneration of the identifier database, it is trivially possible to forge valid ballots. Furthermore, if it occurs that the identifiers are generated successively for the voters in a publicly available, or easily guessable voters list[1], it is also possible to break anonymity of votes.

The computational cost of the attack described above is dominated by the number of possible values of the internal state. This brute-force approach is clearly enabled by the small size of this number. For the case of Microsoft Excel's `RAND()` function, this induces 2^{45} trials, which is not

[1]Such a list is indeed publicly available for the poll considered in our study. We do not know however whether it has been used in order for generating the voting material.

frightening, especially given the distribution opportunities of such an approach.

We also notice that if some additional conditions hold regarding the procedure for generating the identifier database, retrieving the internal state of the random generator can happen to be significantly easier. For example, Microsoft Excel's `RAND()` function is essentially a linear congruential generator[2]. If a given list ID is obtained as the truncation of $\texttt{RAND}() \cdot 10^7$, it is straightforward to list the roughly 2^{24} possibly corresponding internal states. If list IDs are generated in order for each candidate, one can cross-check each of these guesses against the next random draw, and thus quickly validate the right guess. From this data, all previous and future output of the random generator can be deduced[3]. In this case, a mild computational load (2^{24}) incurs the same devastating consequences already described.

Although potentially very effective, these attacks have however not been attempted for two reasons. First, although we believe our guesses, including the use of Microsoft Excel, to be plausible, these are only guesses. Lacking the precise specification of the procedure, mounting the attack is not very tempting, since potentially unfruitful.

Furthermore, in order to mount such an attack, it is necessary to assume that Microsoft Excel follows some well-defined standard for floating-point arithmetic, such as the IEEE-754 standard. Unfortunately this is not the case. As remarked in [13], Microsoft Excel employs *cosmetic* rounding, which is nowhere defined. This feature is certainly not meant as a security countermeasure, and can quite probably be reverse-engineered as well, but we have not tried to do so.

VI. Conclusion

We have shown how to attack (with very few computational resources) a barcode-based postal voting system, with the access to only 4 samples of voting material, our guesses being fully confirmed when accessing to 10 samples of voting material. Due to the fact that the ballots are completely predictable, it is possible to forge the whole voting material sent to the voters and therefore to use it to perform major ballot stuffing.

We believe that our attack demonstrates two main points. Firstly and obviously, the postal voting system we studied was clearly designed in a rather naive manner. Secondly and more importantly, anonymity and confidentiality, albeit very desirable, are not the only security goals a voting system should ensure. The current research on electronic voting has however already allowed to better understand what a good voting system should offer in terms of security (see for instance [8]). Such works point out that voting systems should of course guarantee the confidentiality of the votes (no one should know that a voter has voted in a particular way) but also eligibility (only registered voters can vote, at most once), fairness (the result reflects the actual votes), and verifiability (voters can check that their votes have been counted). In our case study, we probably attacked what is called *eligibility verifiability*, i.e., the fact that voters can check that only eligible voters have voted. Future requirements on barcode voting systems should probably include such properties, in order to prevent, at the very least, ballot stuffing.

As discussed in Section V, we also would like to emphasize that the use of random generators as required by the CNIL or the CNRS does not suffice *per se* for guaranteeing security, in particular anonymity of the voters. Instead, it is necessary to make use of *unpredictable* random generators, that is cryptographically strong random number generators.

Our security analysis was focused on possible threats of *outside attackers*, who do not have any prior information on the system. Even the proposed fixed version of this postal voting system is clearly not robust to *inside attackers*. Indeed, the person responsible for mailing the ballots out can duplicate voting material and stuff the ballot box. Similarly, a dishonest employee of Tagg Informatique could have access to the seed (or key) used in the random generator and could re-generate the voting material. A (standard) way to defend against inside attackers would be to distribute the sensitive information (like the seed of the random generator) among several authorities who are assumed not to collude.

[2]The n-th iterate of this function is $\{x_n\} = x_n - \lfloor x_n \rfloor$, where $x_n = 171^n/p_1 + 172^n/p_2 + 170^n/p_3$, the p_i's being 30269, 30307, 30323. This is equivalently written as $x_n = g^n(p_1p_2 + p_2p_3 + p_1p_3)/(p_1p_2p_3)$, where the numerator arithmetic is performed modulo $p_1p_2p_3$, and g is obtained with the Chinese Remainder Theorem.

[3]Previous output can be deduced because the internal state iteration function is reversible.

Acknowledgements

Firstly, we would like to thank our colleagues who kindly agreed to provide us a copy of their voting material, which allowed us to confirm our initial guesses. We also thank the election group (*Organisation des élections*) at CNRS for their reactivity and their willingness to understand, test, and correct the flaws in the voting system in use. We are also grateful to the anonymous reviewers for their helpful comments and suggestions.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865, project ProSecure, and the ANR-07-SeSur-002 AVOTÉ project.

References

[1] Ben Adida. Helios: Web-based Open-Audit Voting. In *Proc. of the 17th USENIX Security Symposium (USENIX Security'08)*, pages 335–348. USENIX Association, 2008.

[2] Centre National de la Recherche Scientifique. Élection 2011, conseil d'administration du CAES du CNRS, March 2011. http://www.dgdr.cnrs.fr/elections/caes/resultats/resultats_CAES2011_T2.pdf.

[3] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Security and Privacy*, 6(3):40–46, 2008.

[4] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical Voter-Verifiable Election Scheme. In *Proc. of the 10th European Symposium On Research In Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

[5] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *Proc. of the 29th Security and Privacy Symposium (S&P'08)*, pages 354–368. IEEE Computer Society, 2008.

[6] Commission Nationale de l'Informatique et des Libertés. Délibération 98-041 du 28 avril 1998 portant recommandation sur l'utilisation des systèmes de vote par codes-barres dans le cadre d'élections par correspondance pour les élections professionnelles, 1998. http://www.cnil.fr/en-savoir-plus/deliberations/deliberation/delib/71/.

[7] Commission Nationale de l'Informatique et des Libertés. Délibération 2010-371 du 21 octobre 2010 portant adoption d'une recommandation relative à la sécurité des systèmes de vote électronique. *Journal Officiel de la République Française*, 0272, 2010. page texte 29, http://www.cnil.fr/en-savoir-plus/deliberations/deliberation/delib/249/.

[8] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[9] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. Security analysis of the Diebold AccuVote-TS voting machine. http://itpolicy.princeton.edu/voting/, 2006.

[10] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Proc. of the Workshop on the Theory and Application of Cryptographic Techniques (AUSCRYPT'92)*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[11] Alex Halderman. Hacking the D.C. internet voting pilot. http://www.freedom-to-tinker.com/blog/jhalderm/hacking-dc-internet-voting-pilot, 2010.

[12] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *Proc. of the 4th Workshop on Privacy in the Electronic Society (WPES'05)*, pages 61–70. ACM Press, 2005.

[13] William Kahan. Floating-point arithmetic besieged by "business decisions". http://www.cs.berkeley.edu/~wkahan/ARITH_17.pdf, July 2005. Keynote talk at ARITH'17 symposium on Computer Arithmetic.

[14] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. In *Proc. of the 25th IEEE Symposium on Security and Privacy (SSP'04)*, pages 27–28. Comp. Soc. Press, 2004.

[15] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In *Proc. of the 6th International Conference on Information Security and Cryptology (ICISC'03)*, volume 2971 of *LNCS*, pages 245–258. Springer, 2004.

[16] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.

[17] Microsoft corporation. Description of the RAND function in Excel. http://support.microsoft.com/kb/828795, 2010. Microsoft knowledge base article.

[18] Tatsuaki Okamoto. An electronic voting scheme. In *Proc. of the IFIP World Conference on IT Tools*, page 21–30, 1996.

[19] Ronald L. Rivest and Warren D. Smith. Three voting protocols: Threeballot, vav, and twin. In *Proc. of the Electronic Voting Technology Workshop (AVT'07)*, Boston, MA, 2007.

[20] Stateline. http://www.stateline.org/live/details/story?contentId=522635, October 2010.

[21] Brian A. Wichmann and I. David Hill. Algorithm AS 183: An efficient and portable pseudo-random number generator. *Applied Statistics*, (31):188–190, 1982.

APPENDIX

A. Set of 10 samples of voting material used for reverse engineering the system

Note that for the sake of clarity, the prefix 1501, common to all voter and list identifiers in this election, has been omitted.

Sample #	1	2	3	4	5
Voter ID	229157	051665	068705	215452	211409
List IDs	0394	0572	0172	0866	1179
	1485	1663	1263	1957	2260
	2576	2754	2354	3045	3351
	3667	3845	3445	4136	4442
	4758	4936	4536	5227	5533
	5849	6024	5627	6318	6624
	6930	7115	6718	7409	7715

Sample #	6	7	8	9	10
Voter ID	073818	007667	002062	081461	140109
List IDs	0927	0172	1113	0949	0727
	2015	1263	2204	2037	1818
	3106	2354	3290	3128	2909
	4192	3445	4381	4219	3995
	5283	4536	5472	5300	5083
	6374	5627	6563	6396	6174
	7465	6718	7654	7487	7265

B. Samples of corrected voting material

Below are four samples of voting material including the corrective modifications by Tagg Informatique after they were made aware of our findings. Note that, in the following table, all voter IDs should be prefixed with 2943, omitted here for concision.

Sample #	1	2	3	4
Voter ID	3247782 9	8619722 5	8971889 5	2909147 4
List IDs	5381135 9	2989495 8	3719965 6	8497343 2
	0886456 1	3566740 9	3560045 3	8392673 2
	5739954 4	7977436 1	0008984 3	5204306 7
	7664888 8	0517221 3	0015821 1	0468390 0
	5588005 3	8561033 8	2989942 5	1617446 6
	7332815 6	3436714 0	0192921 0	2486192 8
	6988239 8	8680259 0	1971011 8	5821665 5

C. Details of probability estimates

1) Carry propagation in list IDs: As the identifier of the first list L_1 is computed as $(V \bmod 109) + 14$, where V is the voter ID, we have that $L_1 \in \{14, \ldots, 122\}$. There are therefore exactly 109 possible values for L_1, and thus 109 different possible sets of list IDs. Out of these 109 sets, only 20 will not generate any carry. This gives an $89/109 = 81.7\%$ chance of getting a sample of voting material with such a carry, hence detecting the arithmetic progression of common difference 109 as described in Section III-A2.

2) Finding the check digit t in list IDs: It is important to note that the seven list IDs from a single sample of voting material can be enough to solve the system of linear equations modulo 10 in order to find the coefficients of the check digit t. First of all, the first identifier $L_1 = xyz$ and its check digit t give the equation

$$\delta + x\lambda_5 + y\lambda_6 + z\lambda_7 \equiv t \pmod{10}.$$

Furthermore, the difference between two consecutive list IDs xyz and $x'y'z'$ can also produce linearly independent equations of the form

$$(x' - x)\lambda_5 + (y' - y)\lambda_6 + (z' - z)\lambda_7 \equiv t' - t \pmod{10}.$$

More precisely,

- if no carry or borrow occurs, assuming $x' = x + 1$, $y' = y + 1$, $z' = z - 1$, then $t' \equiv t + 1 \pmod{10}$ and we have the equation
$$\lambda_5 + \lambda_6 + 9\lambda_7 \equiv 1 \pmod{10};$$
- if a borrow occurs between z and y (i.e., when $z = 0$, as for instance between lists #3 and #4 of sample #6), then $t' \equiv t + 6 \pmod{10}$ and we have
$$\lambda_5 + 9\lambda_7 \equiv 6 \pmod{10}; \text{ and}$$
- if a carry occurs between y and x (i.e., when $y = 9$ and $z \neq 0$, as for instance between lists #5 and #6 of sample #2), then $t' \equiv t + 8 \pmod{10}$ and we have
$$2\lambda_5 + \lambda_6 + 9\lambda_7 \equiv 8 \pmod{10}.$$

Out of the 109 possible sets of list IDs, 30 produce a system of rank 4 both modulo 2 and 5. This represents a $30/109 = 27.5\%$ chance of being able to completely solve the system with a single sample of voting material. On average, solving the system obtained with one sample will yield 24.04 solutions. Enumerating for up to four available samples, we obtain the following probabilities of success:

Nb. of samples (n)	1	2	3	4
Prob. of being of rank 4 modulo 2	27.5%	79.1%	95.1%	98.8%
Prob. of being of rank 4 modulo 5	27.5%	89.7%	99.1%	99.9%
Prob. of success	27.5%	75.1%	94.5%	98.8%
Expected nb. of solutions	24.04	2.09	1.10	1.02

3) Finding the check digit k in voter IDs: Let us first consider the validity of assumption (a), submitted in Section III-B2, which supposes that the check digit k of a voter ID $abcde$ is computed as

$$k = (7a + 5b + c + d\mu_8 + e\mu_9) \bmod 10.$$

Consider a set of n voter IDs $a_1b_1c_1d_1e_1$ to $a_nb_nc_nd_ne_n$ along with the corresponding check digits k_1 to k_n. For these check digits to satisfy assumption (a), the modular system in μ_8 and μ_9

$$\begin{cases} d_1\mu_8 + e_1\mu_9 \equiv k'_1 \pmod{10} \\ \quad\vdots \qquad\qquad \vdots \\ d_n\mu_8 + e_n\mu_9 \equiv k'_n \pmod{10}, \end{cases}$$

where $k'_i = k_i - 7a_i - 5b_i - c_i$, needs to have at least one solution. Taking the $n \times 3$ matrix of the homogenized system

$$\mathbf{M} \equiv \begin{pmatrix} d_1 & e_1 & -k'_1 \\ \vdots & \vdots & \vdots \\ d_n & e_n & -k'_n \end{pmatrix} \pmod{10},$$

the previous condition is equivalent to checking that $\mathbf{M}$ has at least one vector of the form $(\mu_8, \mu_9, 1)^T$ in its kernel, both modulo 2 and 5. This is finally tantamount to ensuring that the vector $(0, 0, 1)$ does not lie in the row space of $\mathbf{M}$ modulo 2 and modulo 5.

Assuming that voter IDs are taken uniformly in the range $\{0, \ldots, 29999\}$, for $n = 1$ sample of voting material, 25410 of these voter IDs satisfy the above condition (7 out of 8 satisfy it modulo 2, and 121 out of 125 modulo 5). Therefore, the probability of being able to disprove assumption (a) by finding a counter-example using only one sample is $1 - 25410/30000 = 15.3\%$. The probabilities for more samples are given below:

Nb. of samples (n)	1	2	3	4
Prob. of finding a counter-example modulo 2	12.5%	32.8%	58.8%	77.3%
Prob. of finding a counter-example modulo 5	3.2%	18.6%	80.8%	96.0%
Prob. of disproving (a)	15.3%	45.3%	92.1%	99.1%

A similar computation allows one to compute the level of confidence one can have in assumption (b) given a set of n samples of voting material verifying it. For instance, for $n = 2$ samples, out of the 30000^2 possible pairs of voter IDs, 161000000 (i.e., 17.9%) would provide a counter-example if assumption (b) were wrong. Therefore, after checking that the 2 available voter IDs satisfy the condition, one can have a 17.9% confidence in this assumption: in other words, one was not able to disprove (b) despite the 17.9% chance of

this happening. The probabilities for larger values of n are given in the following table:

Nb. of samples (n)	1	2	3	4
Prob. of finding a counter-example modulo 2	0.0%	13.9%	35.4%	60.5%
Prob. of finding a counter-example modulo 5	0.0%	5.3%	23.2%	81.2%
Confidence in (b)	0.0%	17.9%	48.9%	92.5%

Finally, being able to completely solve the system of n modular equations in the three unknowns ϵ, μ_5, and μ_6 arising from assumption (b) only depends on the rank of the corresponding matrix modulo 2 and 5. For instance, with $n = 3$ samples of voting material, there are $30000^3 = 2.7 \cdot 10^{13}$ possible matrices, out of which only $7.2 \cdot 10^{12}$ (i.e., 26.7%) are of rank 3 modulo 2 and modulo 5. This corresponds to an expected number of solutions of 4.41. The probability of solving the system increases with the number of available samples, as shown in the table below:

Nb. of samples (n)	3	4	5	6
Prob. of being of rank 3 modulo 2	33.3%	59.3%	75.6%	85.4%
Prob. of being of rank 3 modulo 5	71.1%	92.4%	98.0%	99.4%
Prob. of success	26.7%	56.9%	75.0%	85.3%
Expected nb. of solutions	4.41	2.03	1.41	1.19

D. Examples of fake ballots