RIES - Internet Voting in Action

Engelbert Hubbers, Bart Jacobs and Wolter Pieters

Security of Systems
Nijmegen Institute for Computing and Information Sciences
Radboud University Nijmegen
PO Box 9010
6500 GL Nijmegen
{E.Hubbers,B.Jacobs,W.Pieters}@cs.ru.nl

Abstract. RIES stands for Rijnland Internet Election System. It is an online voting system that was developed by one of the Dutch local authorities on water management. The system has been used twice in the fall of 2004 for in total approximately two million potential voters. In this paper we describe how this system works. Furthermore we do not only describe how the outcome of the elections *can* be verified but also how it *has* been verified by us. To conclude the paper we describe some possible points for improvement.

1 Introduction

RIES, the Rijnland Internet Election System, was developed by the 'Hoogheemraadschap van Rijnland', one of the Dutch local authorities on water management. In the rest of this paper we will refer to this authority by 'Rijnland'. The Netherlands is divided in approximately 35 'waterschappen'. These are local authorities responsible for almost anything that has to do with water in their region: the quality of the water, the quantity of the water, the quality of the dikes and so on. In the Netherlands this is a serious matter.

These authorities have their own elections with typically between a half and one million potential voters. As a local authority these elections do not have to follow the Dutch 'kieswet', the national law of how elections should be arranged in the Netherlands. They are free to use their own system as long as their board has approved it. In order to increase the number of people actually casting their vote and simultaneously to decrease the cost of such an election, Rijnland decided to invest in setting up an internet election system, even though in general it is absolutely not clear whether this can be done securely. See for instance [3] in which a very critical view towards internet voting is presented and the advise is given not to use this technology at all because of inherent vulnerabilities. Rijnland's previous election in 1999 was an election by ordinary mail. The overall turnout was in the order of 22%. Unfortunately for Rijnland, the turnout in 2004 was decreased to 17% of which 33% has voted by the internet. This amounts to 70.000 online votes, making it one of the largest internet elections. There are no figures available yet on the turnout for the second time RIES was used: the elections for the water management authority 'De Dommel.' Since it is not relevant for this paper, we will not address whether the new system was a success or not, from this perspective. We will only address technical matters.

Rijnland started their development by asking a third party to check the security risks involved with setting up an internet voting system. The Dutch company TNO carried out this preparatory research and came to the following conclusions:

- Many risks involved in voting by internet are not higher than in voting by ordinary mail.
- There are some risks typical to internet settings like DDOS attacks and Trojan horses on client machines. However, there exist procedural counter measures for the specific situation of internet voting.
- None of the currently available systems can be applied to Rijnland's election.

See [6] for the complete report.

Based upon this TNO report, Rijnland decided to develop and build its own system. It set up a project team which included one of the co-authors of the TNO report, Piet Maclaine Pont. Based upon the ideas from the master's thesis ([4]) of one of his former students Herman Robers, he designed the RIES system. In order to get some return value for their investment Rijnland and Maclaine Pont have applied for patents on the system. In Section 2 we describe in detail both Robers's system and RIES. At this stage we only describe the distinguishing feature of RIES: its transparency. Before the elections take place all potential outcomes are published. Via clever but elementary use of hashes and secret encryptions each voter can actually check afterwards if his vote has contributed appropriately to the final outcome. See for instance [1] and [5] for cryptographically more advanced systems.

Several independent parties have looked at the RIES system before it was actually used during the elections. This is where the current authors enter the picture, since they were involved in this evaluation. During a public workshop before the elections most of these parties presented their findings.

As independent outsiders the authors have evaluated the RIES system before use, and have critically followed its deployment, including third party counting of the electronic votes. This has led to a national publication [2]. The current paper presents the not widely known RIES system—together with our findings—to an international audience.

2 The systems

Before we flood the reader with many details on the two related systems, we first want to emphasize the main idea. Essential in both systems is that before the elections already a pre-election reference table is published which contains all possible valid votes represented by *key-less* hashes together with a mapping to the corresponding candidates. During the election the legitimate voters build up a post-election table with their votes represented by hashes *using their personal secret key*. The outcome of the election is calculated by computing key-less hashes of each vote in the post-election table. If the vote is valid, its hash value can be found in the pre-election table and the chosen candidate can be determined. And since this hash is a key-less hash anyone can compute it, hence anyone can check the result of the elections.

2.1 Robers's system

The system described by Robers in his thesis [4] was developed for a local election at the Delft University of Technology in the Netherlands. Key assumption here was that all participants own a multi-function smartcard. This smartcard is trusted and takes care of performing the critical operations.

Robers describes three separate entities. In his thesis he uses a quite technical explanation of these entities. Here we opt for a more intuitive description.

Voter. A person who is allowed to vote in the election. He uses his smartcard to authenticate himself to the election system and to perform the necessary computations.

Authority. This is the party initiating the election. It calculates a secret key for each voter and distributes this key in advance onto each voter's smartcard. It uses specific crypto-hardware.

Anonymizer. This is a party somewhat inbetween the voter and the authority. Its main function is to publish relevant information, especially the pre- and post-election tables.

2.1.1 The procedure The work is split into three parts: things that need to be done before, during and after the election.

Before the election the authority needs to compose a list of valid voters. The total number of voters must be published in order to be sure that no voters are added or deleted during the procedure. Furthermore the authority chooses a unique identifier ELECTION_ID for the election.

In addition for each of the candidates the authority generates a unique id called CANDIDATE_ID. All of these ids need to be published.

The authority[1] generates a unique DES key K_{voter} for each voter and distributes it to the voter's smartcard. On multi-function cards this loading can typically be done safely after issuing without the need for secure channels. With this secret key and the ELECTION_ID the authority calculates the so-called VOTER_ID:

$$\text{VOTER\_ID} := \text{MDC}(\text{MAC}_{K_{\text{voter}}}(\text{ELECTION\_ID})) \tag{1}$$

Here MDC stands for Modification Detection Code, which is a public hash or one-way function.[2] In other words the $\text{MDC}(x)$ is easy to compute once given x, but there is no way to derive the value x itself given $\text{MDC}(x)$. MDC is used by Robers because this function is a standard function on the IBM multi-function cards used. It can of course be replaced by any other reliable keyless hash function. Furthermore, MAC stand for Message Authentication Code. This is also a hash function, but based upon the secret key K_{voter}. In particular this means that wherever $\text{MAC}_{K_{\text{voter}}}(x)$ appears, this must have been calculated in a place where K_{voter} is known.

Using all these ids, the authority now computes a ballot collection for each voter. Each collection looks like this:

$$\begin{pmatrix} \text{VOTER\_ID} \\ \text{MDC}(\text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID}_1)) \\ \text{MDC}(\text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID}_2)) \\ \vdots \\ \text{MDC}(\text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID}_N)) \end{pmatrix} \tag{2}$$

Note that K_{voter} only leaves the crypto-hardware on the smartcard sent to the voters. The authority itself has no access to K_{voter}. The crypto-hardware should delete these voter keys after distribution.

After sending these ballot collections to the anonymizer, encrypted with a special key K_{anon}, the authority has finished *all* its tasks, even though the election itself has not started yet.

Now we discuss what the anonymizer has to do. It starts by decrypting all the ballot collections it received from the authority. Next it sorts these collections upon the VOTER_ID. This guarantees that there is no link between the order of the collections and the identity of the voters. Finally, it publishes the sorted list of ballot collections. We refer to this list as the *pre-election table*. It makes it possible to link $\text{MDC}(\text{MAC}_{K_{\text{voter}_i}}(\text{CANDIDATE\_ID}_j))$ to candidate j.

During the election the anonymizer receives the votes from the voters. These are encrypted with the anonymizer's public key $K_{\text{hide,public}}$ and hence the anonymizer uses the corresponding private key $K_{\text{hide,secret}}$ to decrypt them. The anonymizer acknowledges this reception by writing some value in each voter's smartcard.

After the election the anonymizer publishes a list of all received ballots. We refer to this list as the *post-election table*.

The third party involved is the voter. During the election he has to use his smartcard to compute his own VOTER_ID, based upon the ELECTION_ID, exactly as the authority already did in (1). Since this requires K_{voter} only he can do this. Furthermore he identifies the CANDIDATE_ID of the candidate of his choice. Then he computes his vote

$$\text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID}) \tag{3}$$

If he wants to check his result he should also compute the value

$$\text{MDC}(\text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID})) \tag{4}$$

[1] Technically, this can also be done by a different party as long as it is a party that uses crypto-hardware. Note that this must be done in an unpredictable way.

[2] To be precise RIES uses the MDC 2 algorithm.

which he compares with the values listed in the published ballot collections at the anonymizer. He now knows whether his vote will be interpreted correctly. He sends his vote to the anonymizer:

$$\begin{pmatrix} \text{VOTER\_ID} \\ \text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID}) \end{pmatrix} \tag{5}$$

The first part of this vote is used to check the authenticity of the voter in an anonymous way. The second part is used to determine the chosen candidate. To ensure that this vote is not recorded during transmission he encrypts his vote with the anonymizer's public key $K_{\text{hide,public}}$ before sending it to the anonymizer. The complete setup of the system is shown in Figure 1.

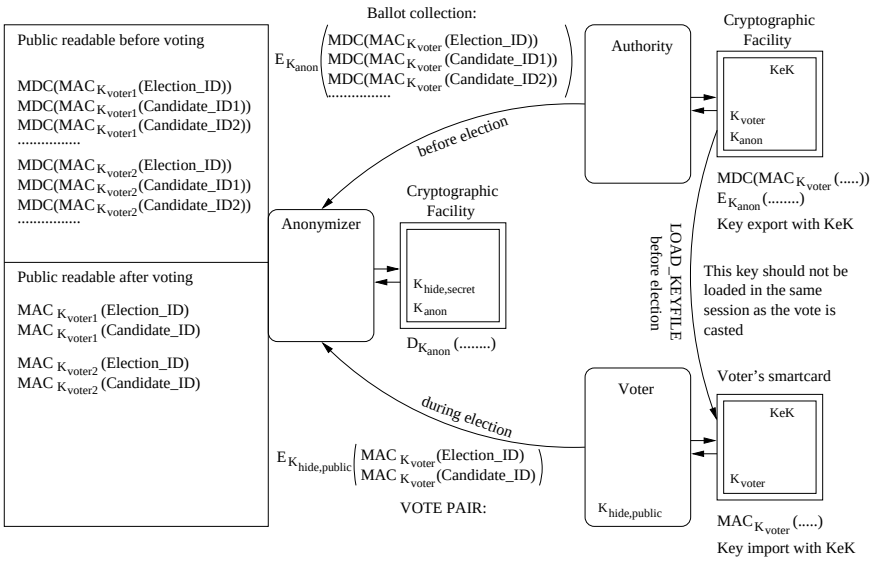

Fig. 1. This diagram is an exact copy of [4, Figure 1.3].

After the election the outcome can be calculated by everyone without using any secret keys. All one needs to do is compute MDC(VOTER_ID) and MDC($\text{MAC}_{K_{\text{voter}}}$(CANDIDATE_ID)) for all the received votes published by the anonymizer in the post-election table. Note that this actually only requires the computation of the outer MDC hash values. These hashes can then be looked up in the pre-election table. In this way one can determine for which candidate each vote should count. This transparency and verifiability make the whole approach very attractive.

2.2 RIES

As mentioned earlier, Robers did his research under supervision of Maclaine Pont. And because the system was not patented in 1998 but published as [4], it could be used as a starting point for RIES. However there are some major differences:

- Because of the cost aspect it was out of the question to give each potential voter a multi-function smartcard. Therefore RIES uses a different system for key management and authentication.
- Robers's system is a purely electronic voting system. RIES is not, since it also provides the possibility to vote by regular mail.
- The strict distinction between the parties authority, anonymizer and voter indicated by Robers is not all that clear for RIES.

These three points will be discussed in more detail below.

2.2.1 Smartcard replacement Because of the high cost it was no option in the water management elections to give each potential voter a personal smartcard. Therefore Robers's system had to be adapted. In his system the smartcard is used for two purposes: to hold the secret keys and to perform computation of the MDC and MAC.

Distribution of the secret key within RIES is done by printing it on paper and sending this paper by ordinary mail to the voter. The key is printed in sixteen digits of the so-called AN34 format. This is a regular number system just like the decimal or hexadecimal system, but now the base is 34. Its digits are the ciphers 0,...,9 and the characters a,...,k,m,n,p,...,z. The 'l' and 'o' are not used because their appearance is too similar to '1' and '0'. Using a high base like 34 means that one can store higher values in fewer digits. And fewer digits printed on the card means that it will be easier for the voter to enter the digits on his computer.

Obviously, since the secret key is simply printed on paper, the voter must be careful with the paper. No-one else should be able to copy or memorize the sixteen digits on his ballot. Hence after voting he should make sure that the key is destroyed.

The cryptographic computations that the smartcard would have performed are now done by the client's computer using JavaScript. If a voter wants to vote, his browser connects to a webserver and downloads a page that contains JavaScript. Within these scripts there are routines available to compute the MDC and MAC values. Of course letting the client's computer do these computations implies a certain risk: the JavaScript code can easily be modified in order to send arbitrary data to the server trying to impersonate legitimate voters. However, in order to cast a valid vote, a client's computer should either be lucky enough to guess both a valid VOTER_ID as well as a valid $\mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Candidate\_ID})$ or it must operate as a virus and read the secret key from the voter as he enters it. The first situation is quite unlikely and the second situation can be detected if the voter checks his vote afterwards.[3]

2.2.2 Integration with mail voting system The merging of the electronic votes and the ordinary mail votes comes down to a transformation of the latter ones to the so-called technical votes already used by the electronic votes. On each paper ballot there are some special numbers from which the VOTER_ID and the $\mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Candidate\_ID})$ can be computed. The algorithm used for this has not been made public, but obviously the K_{voter} needs to be in those numbers somehow. Hence after this transformation the mail votes are handled the same way as internet votes.

Because the mail voters are not getting any feedback on this transformation to a technical vote, they are not able to check what has been done with their vote. Only in case they cast both electronically as well as by mail they can use the electronic feedback to see what happened with their paper vote. However, in the previous election which was done entirely by ordinary mail, it was also not possible for the voter to check his vote, hence this drawback does not make the system any worse than the previous one.

2.2.3 Roles within RIES Whereas Robers emphasizes a clear distinction between who does what, such compartmentalization is not so clear in the RIES system. Main party in the actual elections in the fall of 2004 is a company called TTPI which consists of the architect of RIES, Maclaine Pont, and the main developer Arnout Hannink. For instance they take care of creating the secret keys, publishing the reference tables, merging the mail votes with the internet votes and computing the final outcome. In particular this means that this TTPI company knows all the ins and outs of the system, including the secret keys.

Other parties involved in RIES are the board of Rijnland, SURFnet and of course the potential voters.

[3] There are also other alternatives to protect against such viruses such as using candidate-identities that are different for each voter, so that the virus does not know which identity to select. But this is not part of RIES.

2.2.4 The details Because the separation in roles is not as clear as in the original system by Robers, we will describe the RIES details by looking at the different phases of the procedure: before, during and after the voting.

Before the voting Most of the work before the actual voting takes place is done by TTPI. It starts by generating a DES key K_i for each voter i. These keys are printed on the ballots. As mentioned before they are represented on the ballots in AN34 format. Furthermore TTPI uses these keys to generate the same ballot collections we have seen in (2) in Robers's system. By combining all these ballot collections the so-called reference table or pre-election table is created. This table is published on the internet in the form of a two level `.zip` file. See Figure 2 for an example. In principle it shouldn't matter whether these reference tables are presented as `.zip` files or just as a huge `.txt` file. However, we have noticed a small problem with the use of `.zip` files in combination with the MD5 hashes, which are used to prove that files are not modified. We will get back to this in Section 4.

```
Archive:  01010204.zip
  Length      Date    Time    Name
 --------    ----    ----    ----
     2172  08-25-04 09:32   01010204/RT_0.zip
     4017  08-25-04 09:32   01010204/RT_1.zip
     2173  08-25-04 09:32   01010204/RT_2.zip
     1865  08-25-04 09:32   01010204/RT_3.zip
     2789  08-25-04 09:32   01010204/RT_4.zip
     3097  08-25-04 09:32   01010204/RT_5.zip
     2787  08-25-04 09:32   01010204/RT_6.zip
     1559  08-25-04 09:32   01010204/RT_7.zip
     1559  08-25-04 09:32   01010204/RT_8.zip
     2480  08-25-04 09:32   01010204/RT_9.zip
     2784  08-25-04 09:32   01010204/RT_A.zip
     3405  08-25-04 09:32   01010204/RT_B.zip
     2785  08-25-04 09:32   01010204/RT_C.zip
     1867  08-25-04 09:32   01010204/RT_D.zip
     1559  08-25-04 09:32   01010204/RT_E.zip
     3403  08-25-04 09:32   01010204/RT_F.zip
        0  08-25-04 08:51   01010204/
 --------                   -------
    40301                   17 files
```

```
Archive:  RT_0.zip
  Length      Date    Time    Name
 --------    ----    ----    ----
      220  08-25-04 09:31   008AB1E98AEDFBA450A1813DDC153553
      220  08-25-04 09:31   08677B73378E1D59153DE30263A3C47C
      220  08-25-04 09:31   06CAC042AF7D6940DD8A51814E68DFF8
      220  08-25-04 09:31   00FEA51461FBF7B406554EEF2E23554D
      220  08-25-04 09:31   05C02BD8E3863DB24D6C332A17B78EFB
      220  08-25-04 09:32   070C60BFFC06B7355425E6FFADBBED30
      220  08-25-04 09:32   034C37BA687E21477D38A110954207B8
 --------                   -------
     1540                   7 files

008AB1E98AEDFBA450A1813DDC153553:

vervangend=0
verstrekt=1
vervallen=0
AC94983743058334B25452E0F63A9C20=0101020401
B0015BAC8ECF766DB67825592DC10957=0101020402
ACE42133255CA8184D18E0293FEF7EE8=0101020403
358AAB0C934757ACCF071A1CD732EDEA=0101020499
```

Fig. 2. Reference table format. On the left we see the first level within `01010204.zip`. On the top right we see the second level: all hashed VOTER_IDs starting with `0` are archived into `RT_0.zip`. On the bottom right we see the ballot collection for the voter with MDC(VOTER_ID) = $008AB1E98AEDFBA450A1813DDC153553$. It contains three lines with status bits indicating whether the ballot is a replacement, used or revoked. Because this particular election only had three real candidates (0101020401, 0101020402, 0101020403) and one blank (0101020499) there are only four entries found after the status bits.

After publication of these reference tables together with their MD5 hashes, TTPI no longer needs the keys and destroys them. Checking that this actually happens is a procedural matter. See Figure 3.

During the voting During the actual voting two parties are active. The vote server which is operated by SURFnet, the national internet service provider for universities in the Netherlands, and of course the voter.

Voter i copies the codes printed on his ballot into the appropriate fields of the web page `internetstemmen.nl`. In particular this means that he hands over his personal key K_i to the JavaScript engine of his browser. If he managed to do this without mistakes he can click on his favorite candidate j. The JavaScript engine in his browser will compute the so-called *technical vote* which consists of two values: his VOTER_ID which is equal to MAC_{K_i}(ELECTION_ID) and MAC_{K_i}(CANDIDATE_ID$_j$). Exactly the two values we have already seen before in (5). This vote is sent to the vote server through SSL, and hence it is encrypted and cannot be revealed by other parties besides the voter and the vote server. Note in particular that the secret key K_i is not sent over the internet!

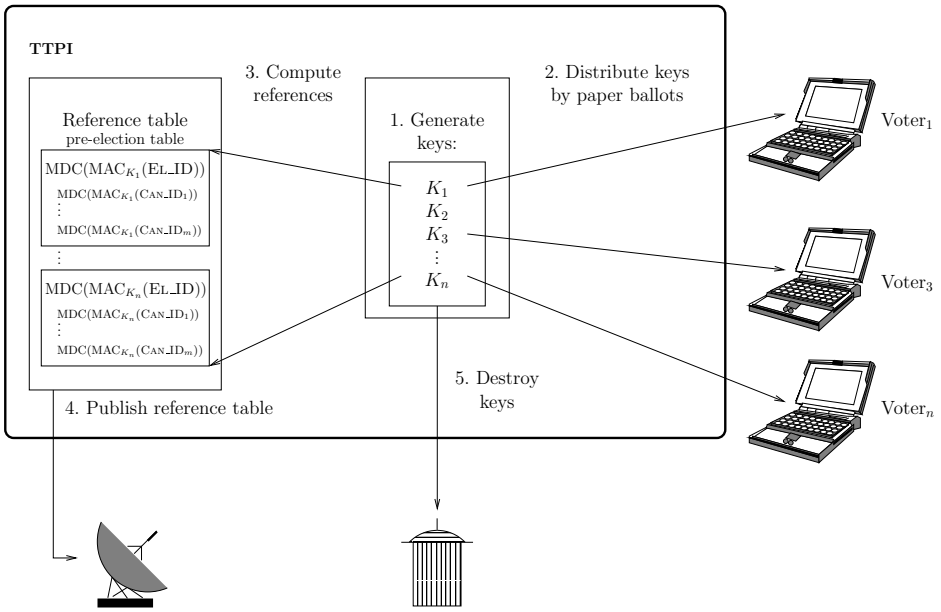

Fig. 3. Phase 1: before the voting

If the server receives this encrypted vote, it decrypts it and strips all meta information like time, date and network address from the vote before storing it. It computes a receipt confirmation and sends this back to the voter. After receiving this confirmation, the voter should carefully destroy his ballot with his secret key. Furthermore he should store his technical vote (5) in order to perform a check afterwards. See Figure 4.

After the voting After the elections are closed, three parties come into action. First, SURFnet hands over all collected technical votes to TTPI. TTPI starts by computing an MD5 hash over these files in order to prove that they did not modify the votes from the server.[4] Next, TTPI computes the total outcome and the official voting office publishes it.

Before TTPI starts working on the technical votes given to them by SURFnet, they transform the scanned paper ballots into technical votes and add them to the files received from SURFnet. From this point on they are treated as internet votes as well. Hence if we talk about technical votes they can originate either from an internet vote or from a mail vote.

TTPI computes the outcome of the election by computing for each technical vote the MDC hash on both parts. In order for a vote to be valid, the combination of these hashes needs to be somewhere in the reference table. Votes that do not comply with this rule are automatically marked as invalid. Furthermore, if the hashes do represent a real vote, TTPI checks whether the vote might be invalid because of some other reason. E.g. if one voter has cast votes for different candidates. If a vote is declared invalid, a log entry is created indicating why it was invalid and hence not counted. A later check can then show what happened with a particular vote. After filtering out all invalid votes, the valid votes that appear more than once are also reduced to one occurrence. Finally, the actual counting is done by looking up the hashes in the reference table and appoint the correct number of votes to the indicated candidates. See Figure 5.

[4] We think that it would have been more trustworthy if SURFnet computed this hash before handing the files over to TTPI.

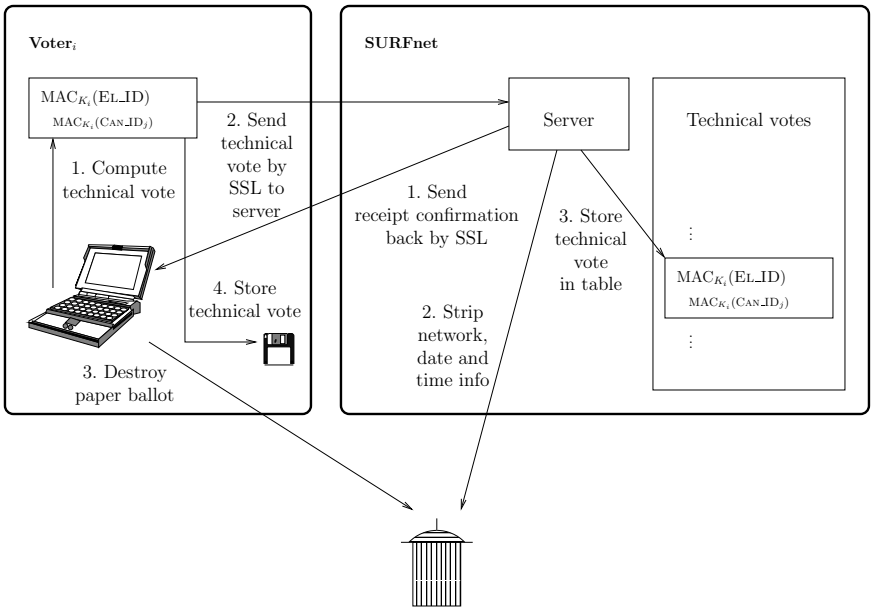

Fig. 4. Phase 2: during the voting

3 Results verified

We have stated in the introduction already that one of the distinguishing features of RIES is that it is transparent. Each voter can check what has happened to his personal vote and anyone who is interested can verify the tally process. In particular this means that also people who were not allowed to vote can check the results.

3.1 Voter specific check

A voter can check his vote because he sees his technical vote on his screen during voting. If he saves this information he will later be able to search for his vote in the post-election table. In this list next to his technical vote also the MDC hashes of the two parts of this vote are listed. With those values he can check in the reference table that his vote was indeed given to his favorite candidate. In the current implementation, there is a drawback to this check system. It is completely based upon the service provided by TTPI: they have already computed the hashes! So if a voter wants to be really sure TTPI did not mess with his vote, he will have to compute the hashes himself. Fortunately there are programs available that can do this. For instance recent versions of `openssl` can compute this MDC hash. But the voter can also implement this function in his favorite programming language. In the JavaScript he already downloaded in order to vote, he can find a JavaScript implementation of this function which can be used as an example. Alternatively, third parties might offer this verification service.

Note that the fact that this transparency feature also introduces a potential privacy problem: anyone able to find your stored technical vote will be able to determine which candidate you voted for.

3.2 General outcome check

The outsider tally verification is also based upon the fact that the computation of the MDC hash can be done by anyone. The authors have written a Java program that uses the files available for

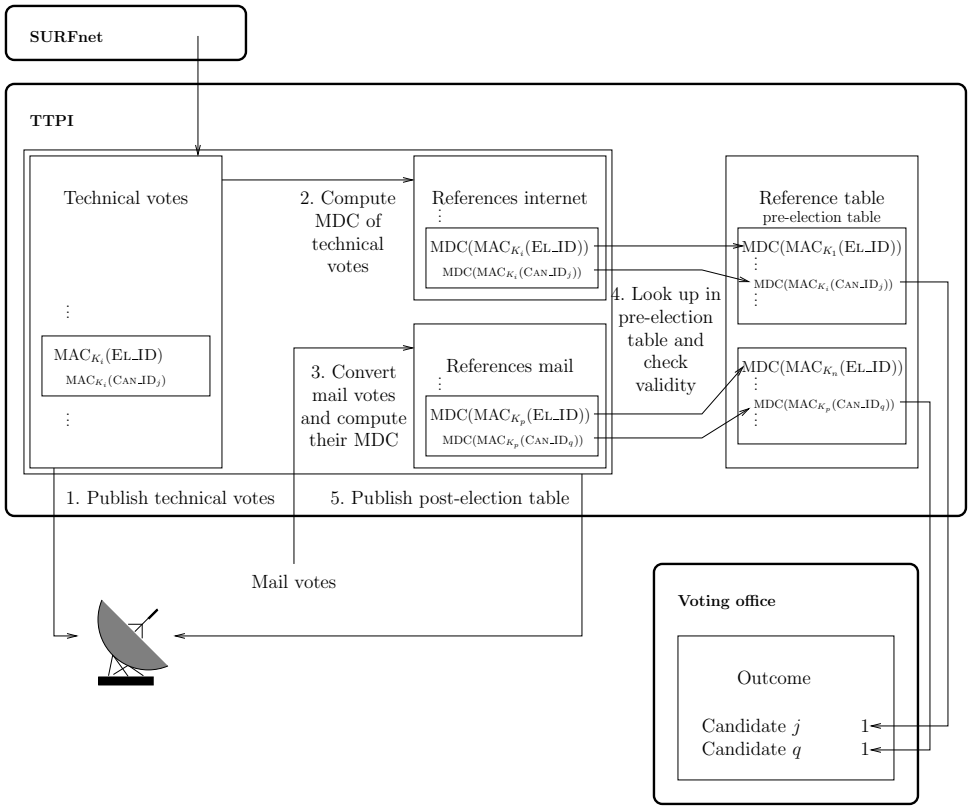

Fig. 5. Phase 3: After the voting

download at the website to compute the final result. Though conceptually easy, we encountered some problems while writing this program.

First of all there is a problem with the files to start with. Theoretically we should start with the files handed over by SURFnet. They are available for download, but the problem is that these files do not include the technical votes that are transformed from the votes sent in by ordinary mail. Therefore we were forced to start with the file created by TTPI which already contains all the status bits on validity checks. Hence this means that our tally still depends a bit on TTPI's work, which is something we don't want. Fortunately, we can check that this TTPI file has a one-to-one correspondence to the SURFnet file when it comes to electronic votes. So at least we know for sure that all internet votes have been counted correctly. Since there is no way that we can check the validity of the imported ordinary mail votes, we always have to trust TTPI on this part. Note that if RIES would have been used in a purely electronic voting session, this would not have been a problem and the system could have been checked completely independent of TTPI.

The second problem we encountered was in the rules determining whether a vote is valid or not. On implementing the rules presented by the RIES project team, we had to make some choices on the order of performing the different validity tests. Obviously, for the outcome of the tally it is not important to know why a specific vote has been declared invalid. The only thing important is that the same set of votes is declared illegal in the different tally programs. However, the choices we made incidentally happened to declare votes invalid for exactly the same reason as the original tally software from TTPI. And hence the outcome of our tally was exactly the same as TTPI's outcome, which was used as official outcome of the elections.

4 Critical remarks

Both Robers's system and RIES present a practical way to set up safe internet elections. Safe in the sense that it is possible to detect fraud. As we have seen in Section 3 internet voters can check what happens to their own vote. We would like to stress that it is important that voters indeed use this possibility. Unfortunately, voters have complained that in the actual use of the RIES system the procedure to check their vote is quite complicated, hence reducing the chance that these checks will really be carried out.

Some critical remarks are appropriate, however. They can contribute to an even better system.

- The current, mixed system is not completely transparent because of the parallel election by ordinary mail. People who voted by mail do not really have the opportunity to check what happens to their vote. In particular TTPI, the party that merges mail votes with internet votes, should really be trusted. In principle they have the possibility to tamper with the mail votes. This can only be prevented by procedural checks.
- Since TTPI knows everything about the system it has a lot and maybe too much power. Not because we have reason to believe that they abuse their powers, but mainly because in general a separation of powers, compartmentalization, is wise, we would like to see that other parties take over some of their responsibilities.
- In Figure 2 we have seen that the ballot collection for each voter also contains three status bits. These bits indicate whether the corresponding vote ballot is actually being used or revoked and so on. When these reference tables are published before the elections, the MD5 hash over the `.zip` files are computed. However, if a voter complains that he did not receive a ballot, he can ask for a replacement. This means that the status bits for his first ballot will change from used to revoked and the status bits for his second ballot will change from replacement to used. For the tally process it is essential that these changes are recorded. Otherwise it would be possible to cast valid votes with revoked ballots! Therefore the reference tables need to be modified after the election. Obviously, since the status bits are within the files itself such a modification will cause the MD5 hash of the complete `.zip` file to change as well. The problem here is that the hashes are meant to detect whether the MDC hashes inside the files are modified or not. Hence by looking at the hashes only it is not possible to determine whether a reference table `.zip` was modified only in its status bits or also in its content. In order to verify whether the content has not been changed one really needs to go into the files and compare each ballot collection in the modified tables to the corresponding collection in the original tables. The authors tried to automate such a check for the Rijnland election using a '`diff`' tool. In principle the check worked, but we did not foresee that the order of the MDC hashes in the files could have changed as well. Therefore our script reported a lot of false positives: `diff` reported permutations of the rows as modifications whereas these modifications are harmless with respect to the RIES system. Adding a simple sort into the script would solve this problem, but because of the fact that we were roughly talking about 200 million files to check, we did not run the script again. We manually inspected a random set of them and found that they were all indeed false positives.

  As a side effect our script showed that the total number of status bit combinations in the `.zip` files was exactly as published by TTPI.
- Using hashes in combination with `.zip` files can also lead to false positives for other reasons. For instance it should be possible to build up the modified reference tables by starting with the old ones, unzipping them and applying the changes as recorded by the voting office while handing out new ballots. After zipping the new tables it would be nice if a check on the MD5 hashes showed that this construction indeed leads to the same tables published by TTPI. However, due to different `zip` programs it is possible that files which are equal when unzipped will not be equal if zipped. Hence such a check is likely to fail whereas it should not.
- The system depends on collision free hashes. If two valid candidates or voters are mapped onto the same hash value, it is no longer possible to determine which candidate was the chosen one. However, since these collisions can already be noted by the authority after generating

the reference tables, it seems that the authority should be able to replace the keys for the particular voter causing the collision. However, with a good hash function such collisions are extremely rare.
- Besides TTPI also SURFnet needs to be trusted. Since they are able to compute the MDC hashes on each vote they received, they can detect for which candidate each vote is intended. And in particular this means that they can delete votes for candidates they don't like. Since the MD5 hash on their received votes will only be computed when the election has been closed and the votes are handed over to TTPI, it is difficult to detect such fraud. An independent party cannot detect it for instance. Only if each internet voter checks his own vote, he can detect this kind of fraud with his vote.
- Note that it is not possible for SURFnet to add valid votes: they need the secret keys for that. However, since TTPI is calculating the MD5 hash to secure the post-election table, and they had the secret keys before the election, they are in a position to alter or add votes in favor of specific candidates. Note that they can only do this if they offended the policy to destroy the keys after distributing them! Fortunately SURFnet can detect fraud like this. As long as they do not destroy the files with the received votes before TTPI publishes their list, SURFnet can detect any modification with respect to the internet votes.
- The concept known as 'family voting' is always a risk in internet elections. However in this particular RIES setting, it is accepted as a risk, like it already was in the previous election which was done entirely by ordinary mail.
- In general it is good to have open source software for electronic voting systems. Because of the JavaScript used in the RIES system, most of the code is automatically open source. Currently the code running on the server that collects the votes and the tally software is not open source. However, because of the transparency of the system, this is less important. It does not really matter what this software does as long as the final outcome is correct. And this is something that can be checked independently.
- DDOS remains a concern, but has not turned out to be a problem in the actual elections. SURFnet has taken technical measures to handle heavy traffic.

5 Conclusion

This paper has presented a critical account of the actual use of a little known internet voting system RIES. The system itself is very interesting because its verifiability: fraud can be detected. Independent recounts have indeed taken place—leading to the same outcome as the official one. The procedural issues surrounding the organization of the elections based on RIES leave room for improvement. Especially the merging of the ordinary mail votes with the internet votes needs to become more verifiable. The designers have already announced the intention to implement such a verification system. Hence the RIES system gives us a more positive feeling towards the future of internet voting than the authors of [3].

References

1. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology–EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer-Verlag, 1997.
2. E.-M.G.M. Hubbers and B.P.F. Jacobs. Stemmen via internet geen probleem. *Automatisering Gids*, 42:15, 2004.
3. D. Jefferson, A.D. Rubin, B. Simons, and D. Wagner. Analyzing Internet Voting Security. *CACM*, 47(10):59–64, 2004.
4. Herman Robers. Electronic elections employing DES smartcards. Master's thesis, Delft University of Technology, December 1998. `http://www.iscit.surfnet.nl/team/Herman/election.ps`.
5. B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Advances in Cryptology–CRYPTO'99*, volume 1666 of *LNCS*, pages 148–164. Springer-Verlag, 1999.

6. M.P. van Esch-Bussemakers, J.M.E. Geers, P.G. Maclaine Pont, and H.J. Vink. ELS: Beveiligings- en gebruikersaspecten van elektronisch stemmen voor het Hoogheemraadschap van Rijnland. TNO-rapport TM-02-C066, TNO Technische Menskunde, 2002. http://www.rijnlandkiest.nl/contents/pages/00000109/rapporttno.pdf.