9

RIES – Internet Voting in Action

Engelbert Hubbers, Bart Jacobs and Wolter Pieters
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

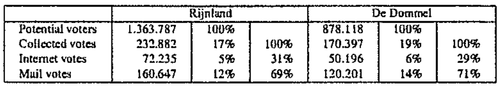
RIES stands for Rijnland Internet Election System. It is an online voting system that has been used twice in the fall of 2004 for in total over two million potential voters. In this paper we describe how this system works. Furthermore we describe how the system allowed us to independently verify the outcome of the elections—a key feature of RIES. To conclude the paper we evaluate possible threats to this system and 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 for 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 (except drinking water): 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 on 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 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 [4], in which a very critical view towards internet voting is presented and the advice is given not to use this technology at all because of inherent vulnerabilities, and [5], in which a brief response is given to [4], mentioning RIES.

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 decreased to 17% of which 31% voted via the internet. Still, this amounts to over 70.000 online votes, making it one of the largest internet elections held so far. The second time RIES was used, during the elections for the water management authority 'De Dommel', showed a similar turnout. See Figure 1 for the precise figures. Since it is not relevant for this paper, we will not address whether the new system was a success or not, from the turnout perspective. We will only address technical and procedural matters.



	Rijnland			De Dommel		
Potential voters	1.363.787	100%		878.118	100%	
Collected votes	232.882	17%	100%	170.397	19%	100%
Internet votes	72.235	5%	31%	50.196	6%	29%
Mail votes	160.647	12%	69%	120.201	14%	71%

Figure 1. Election turnout figures

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 ordinarv mail.
- There are some risks typical to internet settings such as DDOS attacks and Trojan horses on client machines. However, there exist procedural counter measures for the specific situation of internet voting.
- None of the systems available for internet voting at the time were suitable for Rijnland's election.

See [10] 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, Maclaine Pont. Based upon the ideas from the master's thesis [6] of one of his former students, Robers, he designed the RIES system. In order to get some return on their investment Rijnland and Maclaine Pont have applied for patents on the system [2]. In Section 2 we discuss RIES in detail. At this stage we only point out 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 [8] 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 authors of this paper enter the picture, since they were involved in this evaluation. During a public workshop [9] 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 as described in Section 3. An earlier publication, [3], presented RIES to a national audience, but this paper presents it—together with our findings—to an international public. Of course, RIES and its underlying ideas, are not ours. Our contribution in this paper lies in an accessible description and a critical evaluation.

In Section 4 we describe to what extent RIES is vulnerable to general, well known attacks. In Section 5 we describe some of our findings that we think could be done differently to make the system more trustworthy.

2. RIES: the system

Before we provide the reader with the details of RIES, we first want to emphasize the main idea. Essential in the system is that before the election a pre-election reference table is published which contains all possible valid votes represented by *key-less* hashes (MDC) 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* (MAC). This table will also be published. 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.

As mentioned earlier, Robers did his research under supervision of Maclaine Pont. And because his system, based upon keys in smartcards, was not patented in 1998 but published as [6], 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.
- Robers's system makes a strict distinction between several roles within the system: the authority, the anonymizer and the voter. In RIES this distinction is less clear.

These issues will be discussed in more detail below.

2.1. Smartcard replacement

In Robers's system the smartcard is used for two purposes: to hold the secret keys and to perform computation of the key-less MDC hash and key-based MAC.

Distribution of the secret key within RIES is done by printing it in sixteen characters on a ballot and sending it by ordinary mail to the voter. Obviously, the voter must be careful with this paper with his printed secret key. No-one else should be able to copy or memorize the sixteen characters on his ballot. Hence after voting he should make sure that the key is destroyed.

The cryptographic computations in RIES are done by the client's computer using JavaScript. If a voter wants to vote, his browser connects to a web server 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. We will get back to this in Section 4. However, in order to cast a valid vote, a client's computer should either be lucky enough to guess both a valid VOTER_ID and a valid identifier for the chosen candidate, 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 since both identifiers are sixteen hexadecimal characters long. The second situation can be detected if the voter checks his vote afterwards. But then it is too late to change the vote. See (1) later on for the definition of VOTER_ID.

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 same format as used by the electronic votes: the so-called technical vote. See (3) later on. On each paper ballot there are some special numbers from which the VOTER_ID and the $\mathrm{MAC}_{K_{\text{voter}}}(\textsc{Candidate\_ID})$ can be computed. The algorithm used for this has not been made public, but obviously the secret voter key needs to be in those numbers somehow. Hence after this transformation the mail votes are handled the same way as internet votes.

In the original setup of RIES the mail voters were not getting any feedback on this transformation to a technical vote, and hence they were not able to check what had been done with their vote. However, to overcome this drawback, RIES has used some kind of shadow system to be used after the elections. This shadow system offers the possibility to compute the technical vote after the elections, if the voters kept their original ballot with the keys on it. Obviously, this implies that mail voters should use some internet system after all to check their vote, which is not very likely to happen. If they would have felt comfortable with internet they probably would have voted by internet to begin with. However, in the previous election which was done entirely by ordinary mail, it was not possible at all for the voter to check his vote, hence this drawback does not make the system any worse than the previous one.

2.3. Roles within RIES

Whereas Robers emphasizes a clear pattern of who does what, such a 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 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, which makes it a very powerful player.

Other parties involved in RIES are the board of Rijnland, SURFnet (for the server infrastructure) and of course the potential voters.

2.4. The details

We will describe the RIES details by looking at the different phases of the procedure: before, during and after the voting.

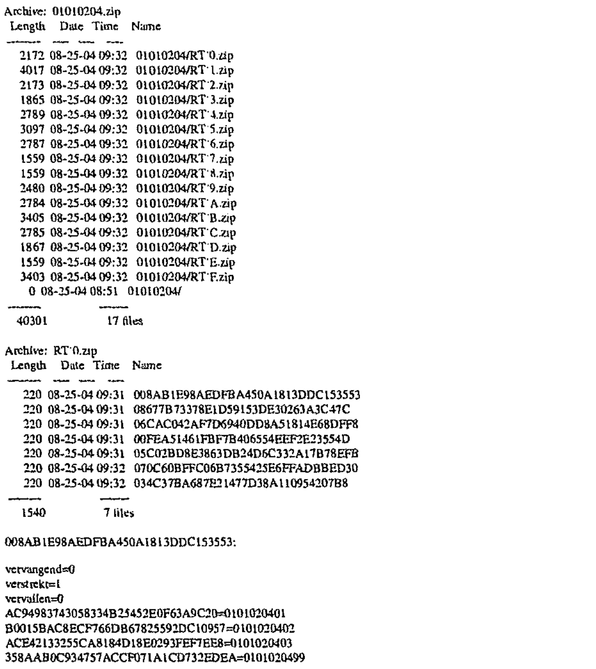
Before the voting Most of the work before the actual voting takes place is done by TTPI. It starts by generating a unique ELECTION_ID for the upcoming election. Next, a DES key K_i for each voter i is generated. These keys are printed on the ballots. Furthermore TTPI uses these keys to generate the VOTER_ID

$$\textsc{Voter\_ID} = \mathrm{MAC}_{K_{\text{voter}}}(\textsc{Election\_ID}) \qquad (1)$$

and the complete ballot collections shown in (2).

$$\begin{pmatrix} \mathrm{MDC}(\mathrm{MAC}_{K_{\text{voter}}}(\textsc{Election\_ID})) \\ \mathrm{MDC}(\mathrm{MAC}_{K_{\text{voter}}}(\textsc{Candidate\_ID}_1)) \\ \hfill \simeq \textsc{Candidate\_ID}_1 \\ \mathrm{MDC}(\mathrm{MAC}_{K_{\text{voter}}}(\textsc{Candidate\_ID}_2)) \\ \hfill \simeq \textsc{Candidate\_ID}_2 \\ \vdots \\ \mathrm{MDC}(\mathrm{MAC}_{K_{\text{voter}}}(\textsc{Candidate\_ID}_N)) \\ \hfill \simeq \textsc{Candidate\_ID}_N \end{pmatrix} \qquad (2)$$

By combining all these ballot collections the so-called reference table or pre-election table is created. It contains all possible outcomes. The $\simeq$-sign represents the link between the hashed value and the candidate. This reference table is published on the internet in the form of a two level .zip file. See Figure 2 for an example.



```
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
```

Figure 2. Reference table format.

The first block represents the top level of the reference table for the election 01010204. The second block shows the next level: all hashed VOTER_IDs starting with 0 are archived into RT_0.zip. At the bottom 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 secret DES keys and destroys them. Checking that this actually happens (before the elections) is a procedural matter. In Figure 3 we have presented the actions of the parties involved by means of a message sequence chart.

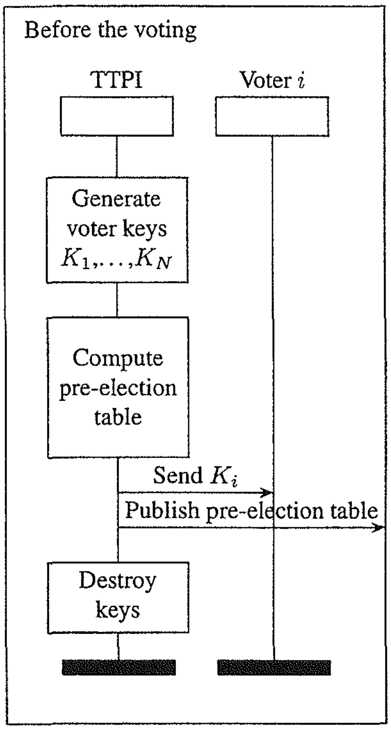

Before the voting
TTPI
Voter i
Generate voter keys $K_1,\ldots,K_N$
Compute pre-election table
Send K_i
Publish pre-election table
Destroy keys

Figure 3. Phase 1: before the voting

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 www.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*:

$$\begin{pmatrix} \textsc{Voter\_ID} = \mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Election\_ID}) \\ \mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Candidate\_ID}) \end{pmatrix} \tag{3}$$

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 and that the information in this technical vote alone cannot identify a voter.

Therefore, if the server receives such an SSL-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 cryptographic 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 (3) in order to perform a check afterwards. The receipt confirmation needs to be stored by the voter in order to prove afterwards that his vote was received by the server. See Figure 4.

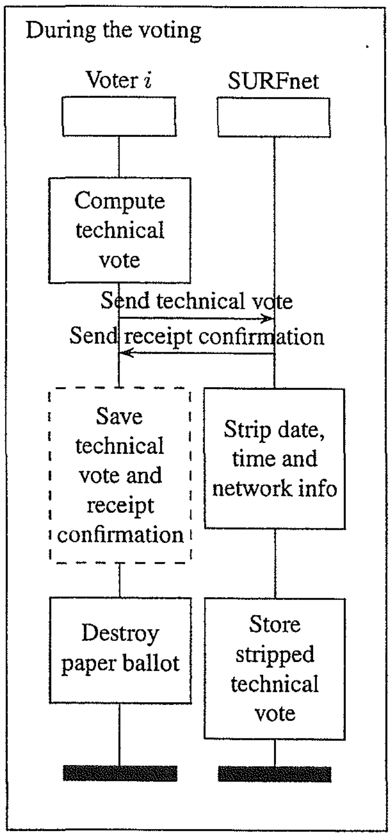

During the voting
Voter i
SURFnet
Compute technical vote
Send technical vote
Send receipt confirmation
Save technical vote and receipt confirmation
Strip date, time and network info
Destroy paper ballot
Store stripped technical vote

Figure 4. Phase 2: during the voting

Note that we slightly modified the interpretation of the mesthe same format as used by the electronic votes: the so-called technical vote. See (3) later on. 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 secret voter key needs to be in those numbers somehow. Hence after this transformation the mail votes are handled the same way as internet votes.

In the original setup of RIES the mail voters were not getting any feedback on this transformation to a technical vote, and hence they were not able to check what had been done with their vote. However, to overcome this drawback, RIES has used some kind of shadow system to be used after the elections. This shadow system offers the possibility to compute the technical vote after the elections, if the voters kept their original ballot with the keys on it. Obviously, this implies that mail voters should use some internet system after all to check their vote, which is not very likely to happen. If they would have felt comfortable with internet they probably would have voted by internet to begin with. However, in the previous election which was done entirely by ordinary mail, it was not possible at all for the voter to check his vote, hence this drawback does not make the system any worse than the previous one.

2.3. Roles within RIES

Whereas Robers emphasizes a clear pattern of who does what, such a 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 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, which makes it a very powerful player.

Other parties involved in RIES are the board of Rijnland, SURFnet (for the server infrastructure) and of course the potential voters.

2.4. The details

We will describe the RIES details by looking at the different phases of the procedure: before, during and after the voting.

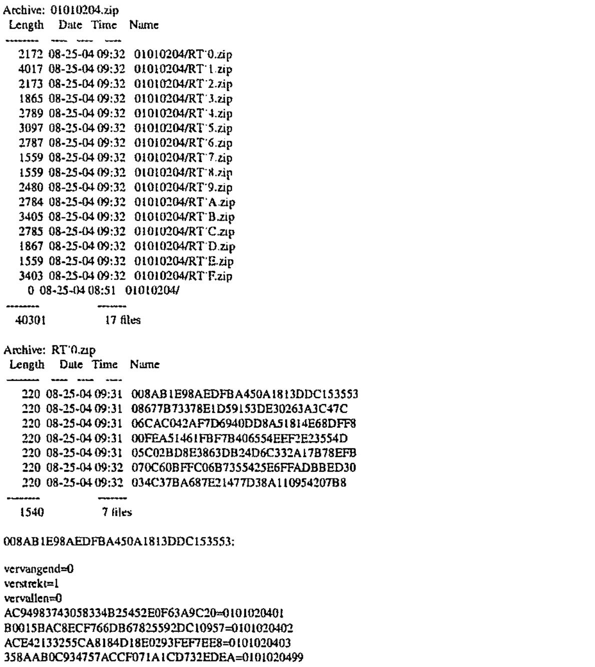
Before the voting Most of the work before the actual voting takes place is done by TTPI. It starts by generating a unique ELECTION_ID for the upcoming election. Next, a DES key K_i for each voter i is generated. These keys are printed on the ballots. Furthermore TTPI uses these keys to generate the VOTER_ID

$$\textsc{Voter\_ID} = \mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Election\_ID}) \qquad (1)$$

and the complete ballot collections shown in (2).

$$\begin{pmatrix} \mathrm{MDC}(\mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Election\_ID})) \\ \mathrm{MDC}(\mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Candidate\_ID}_1)) \\ \qquad \simeq \textsc{Candidate\_ID}_1 \\ \mathrm{MDC}(\mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Candidate\_ID}_2)) \\ \qquad \simeq \textsc{Candidate\_ID}_2 \\ \vdots \\ \mathrm{MDC}(\mathrm{MAC}_{K_{\mathrm{voter}}}(\textsc{Candidate\_ID}_N)) \\ \qquad \simeq \textsc{Candidate\_ID}_N \end{pmatrix} \qquad (2)$$

By combining all these ballot collections the so-called reference table or pre-election table is created. It contains all possible outcomes. The $\simeq$-sign represents the link between the hashed value and the candidate. This reference table is published on the internet in the form of a two level .zip file. See Figure 2 for an example.



```
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
```

Figure 2. Reference table format.

The first block represents the top level of the reference table for the election 01010204. The second block shows the next level: all hashed VOTER_IDs starting with 0 are archived into RT_0.zip. At the bottom 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 secret DES keys and destroys them. Checking that this actually happens (before the elections) is a procedural matter. In Figure 3 we have presented the actions of the parties involved by means of a message sequence chart.

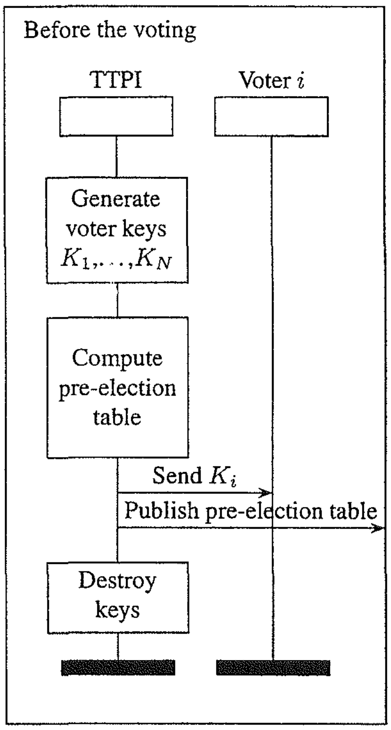

Before the voting
TTPI
Voter i
Generate voter keys $K_1,\ldots,K_N$
Compute pre-election table
Send K_i
Publish pre-election table
Destroy keys

Figure 3. Phase 1: before the voting

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 www.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*:

$$\begin{pmatrix} \text{VOTER\_ID} = \text{MAC}_{K_{\text{voter}}}(\text{ELECTION\_ID}) \\ \text{MAC}_{K_{\text{voter}}}(\text{CANDIDATE\_ID}) \end{pmatrix} \quad (3)$$

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 and that the information in this technical vote alone cannot identify a voter.

Therefore, if the server receives such an SSL-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 cryptographic 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 (3) in order to perform a check afterwards. The receipt confirmation needs to be stored by the voter in order to prove afterwards that his vote was received by the server. See Figure 4.

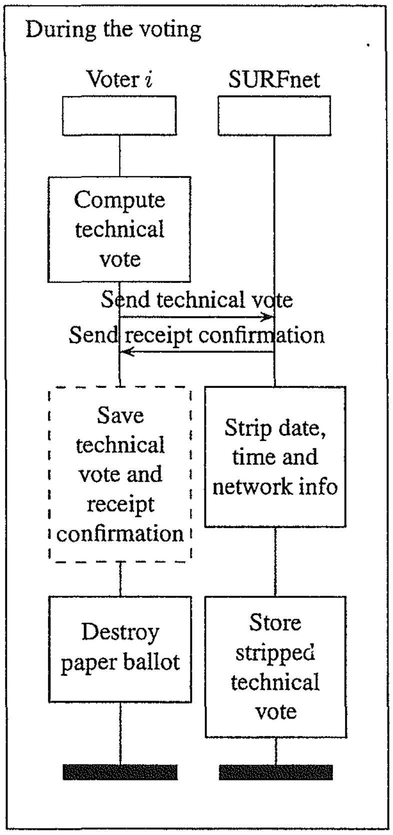

During the voting
Voter i
SURFnet
Compute technical vote
Send technical vote
Send receipt confirmation
Save technical vote and receipt confirmation
Strip date, time and network info
Destroy paper ballot
Store stripped technical vote

Figure 4. Phase 2: during the voting

Note that we slightly modified the interpretation of the message sequence: we used a dashed action to indicate that it is optional.

Note also that the task of the server is not very complex. In fact, because there is no session information shared with the clients, and because storing votes is based upon file storage in the current setup, backup and the use of multiple servers is extremely simple.

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. 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 received by mail 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 reveal what happened to 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 assigning the correct number of votes to the indicated candidates. See Figure 5.

3. Verification of election outcome

We have stated already in the introduction 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 appear. With

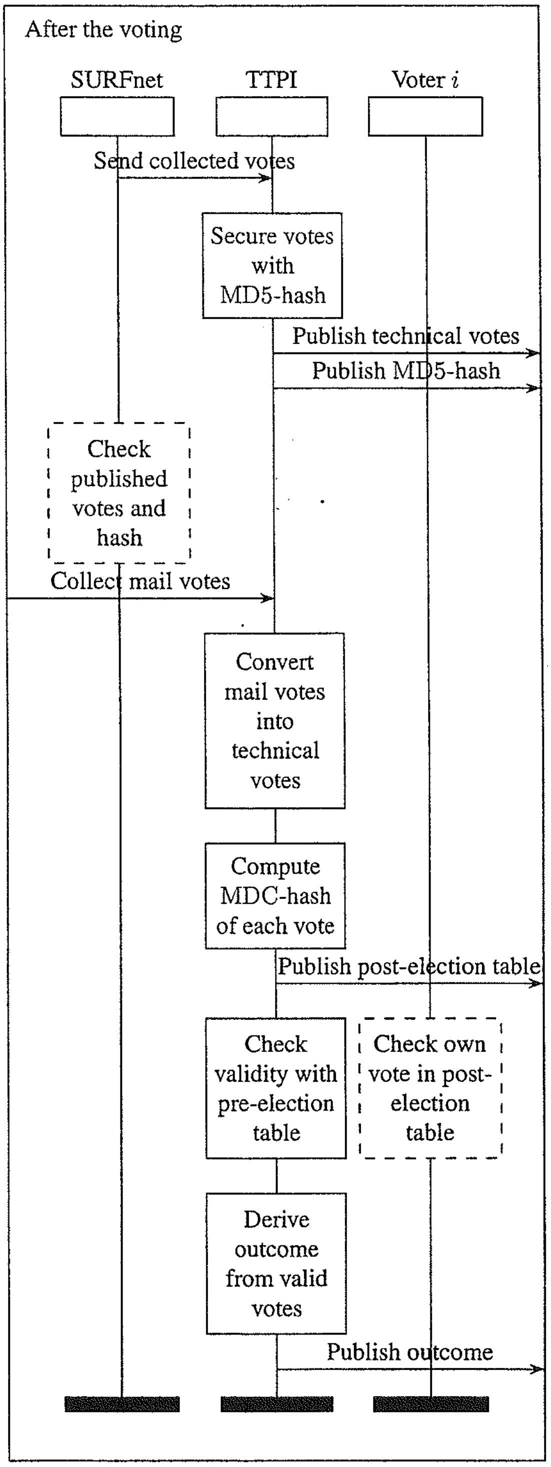

After the voting
SURFnet
TTPI
Voter i
Send collected votes
Secure votes with MD5-hash
Publish technical votes
Publish MD5-hash
Check published votes and hash
Collect mail votes
Convert mail votes into technical votes
Compute MDC-hash of each vote
Publish post-election table
Check validity with pre-election table
Check own vote in post-election table
Derive outcome from valid votes
Publish outcome

Figure 5. Phase 3: After the voting

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 or third parties available that can do this.

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 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 come from the votes sent by ordinary mail. Therefore we were forced to use a file which was generated by TTPI and hence our tally still depends on theirs. Currently there is no way to avoid this dependency. Fortunately, we were able to see that at least the internet votes in the file we used matches one to one with the SURFnet files.

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. Threats

In this section we mention some general well known threats to internet voting systems and explain to what extent they are indeed real threats to RIES. The threats are listed in random order.

- A virus on the local PC modifying the vote. Such a virus can read the personal key K_i and can compute a valid vote for any candidate it wants, since this list of candidates is known before the elections and the IDs are equal for all voters. However, fraud like this can be detected. If the virus shows the technical vote to the voter corresponding to the voter's choice, the voter will see that this vote is not present in the post-election table. The same holds if the virus shows a random technical vote to the voter. If the virus shows the technical vote of the virus's choice, the vote will be listed in the post-election table, but comparing it with the pre-election table will reveal that the vote is given to a different candidate.

  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.

- A virus on the local PC compromising privacy. Although fraud by a virus with respect to the value of the vote can be detected by the voter afterwards, a virus can compromise the privacy of a voter. Parallel to sending the correct vote to the vote server, a virus can also send this vote to a different server and publish the vote together with some personal information like ip-address, user name and other circumstantial evidence of the voter's identity.

- Compromise privacy from outside. Because the secret keys used are only known inside the PC, it is not possible to link a specific vote from its hashed value back to the original candidate from outside. Of course this implies that the order used to list the VOTER_IDs in the pre-election reference table should not be correlated with the identity of the voters. As usual key management is important. Both TTPI as well as the voter should destroy the keys after they used them.

- Family voting. This is a serious problem with all systems where the votes are cast outside a controlled environment. If the voter is forced to vote for someone else, he can make this vote invalid afterwards by casting another vote to a different person. However, the person responsible for the forced vote is also able to detect that the original vote is not counted. Hence the voter loses his right to vote properly and runs the risk of actions against him after all.

- Buying votes. The problem here is more or less equal to the family voting issue. If you buy a vote from someone, he is very well able to double cross you. But you can detect this afterwards.

- Compromise secrecy. Because of the 128bit SSL connection between the browser and the server it will not be possible to eavesdrop on the line and decrypt the messages between the voter's browser and the server. Assuming that a good implementation of the protocol is used of course.

- Compromise identity of the vote server. The official vote server www.internetstemmen.nl uses a certificate to identify itself. A fake server will not be able to do this as well. However, if the voter has no knowledge about these certificates he can be easily fooled. If someone manages to hack a DNS server to redirect http://www.internetstemmen.nl to his own server, the voter will not know that something is wrong unless he already knows that he should be redirected to the secure address https://www.internetstemmen.nl. Therefore it would be wise to put this https address in the vote ballot and not the http address. Because of the use of certificates it is not possible to direct the voters to fake vote servers.

- Compromise integrity of the vote server. The server might be compromised by script kiddies or more professional hackers. This can never completely be prevented. However, the authors did a serious test on the server setup by SURFnet and concluded that SURFnet had taken this risk very seriously.
- DDOS attack. This will always be an inherent problem to internet elections and hence also to RIES. It is a matter of money: how much do you want to spend in order to keep the system up and running. During the elections there were no problems with DDOS attacks. SURFnet had taken technical measures to handle heavy traffic. Two servers were used, providing an overcapacity of 97%.
- Key size of 56 bits. We have not looked at the strength of the algorithm to generate the keys. Or to the strength of keys in general of this length, although it is known that keys of 56 bits can be broken. However, the cryptographic issues of RIES have been reviewed by a team of the Cryptomathic company in Aarhus, Denmark [7]. Their general conclusion was that RIES reflects the state of the art in commercial e-vote systems and implemented unusually much security compared to the available budget.
- Insider attacks. The current setup is vulnerable with respect to attacks from the inside. SURFnet is able to delete votes for candidates they don't like. The personnel handling the conversion from mail votes into technical votes might have the algorithm to extract the secret keys from the codes on the paper forms and hence derive valid votes for the candidates they want. And of course TTPI might abuse the secret keys they generated or even the master key used to generate these secret keys.

5. Critical remarks

As we have seen in the previous sections, RIES presents a practical way to set up safe internet elections, in the sense that voters can detect fraud. Moreover, the designers have paid attention to usability aspects. Much time has been spent on assessing the capabilities of the potential users, and adapting the system to their needs. Sometimes this meant sacrificing some high-tech security, but transparency is at least an equally important factor in gaining trust.

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

- 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.

  An important procedural issue here is the fact that if there is only one voter who can prove with his cryptographic confirmation receipt that he did cast his vote correctly, but that it doesn't show up correctly in the post-election table, the entire election will become invalid.
- We have seen before that a virus on the voter's computer might change the vote sent to the server without the voter knowing this. He will be able to detect this fraud afterwards, however, he will probably not be able to prove that he cast his vote to a different candidate. Since the algorithm for the cryptographic confirmation receipt is not made public, it is not clear what will be in this receipt, but most likely this will include a reference to the chosen candidate, which in case of such a virus will be for the wrong candidate. Hence the system would definitely be strengthened by using per voter different identifiers for candidates.
- Since TTPI knows all ins and outs of the system it has a lot and maybe too much power. Especially since they are the ones who generate the secret keys and we need to trust them in destroying these keys at the right time. 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. The idea of this hash is that it can be used to show that no id-values of the entries inside the file have been modified during the election. However, during the elections it might be necessary to modify the status bits. And

hence also the hash over the file is changed. This violates the original idea: the hash cannot be used anymore to detect easily whether the id-values have been modified or not. It gives false positives if only the status bits have been modified.

- Using hashes in combination with .zip files can also lead to false positives for other reasons. If one builds up the modified reference tables by unzipping the old ones, applying the changes as recorded, and zipping them again, it might still lead to differences in the hashes. Due to different zip programs it is possible that files are equal when unzipped will not be equal when zipped.
- The system depends on collision free hashes. When checking the validity (and the corresponding candidate) of a technical vote, MDC(VOTER_ID) works as a primary key and MDC(MAC(CANDIDATE_ID)) as a secondary key. 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 TTPI while generating the reference tables, it can replace these problematic keys already before the key distribution. With a good hash function such collisions are nonetheless 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 implying they can delete votes as they 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! However, if TTPI would add, delete or modify votes after the election is closed, SURFnet can detect this fraud. But, if TTPI would add votes during the election by sending them to the vote server the way normal voters do, SURFnet cannot detect this fraud. It can only be detected if they happen to add votes for voters that really did participate in the elections. But looking at the turnout figures, collisions like these are not very likely to occur.

  Obviously it would have looked more trustworthy if SURFnet computed the MD5 hash before handing the files over to TTPI, because they never had the secret keys in their possession.
- In general it is good to have open source software for electronic voting systems. Within RIES not all code is open source at the moment. Fortunately, this is not a big issue here. Since the outcome can be checked, it is not necessary to know exactly how the software derives this outcome.

6. Conclusion

This paper has presented a critical account of the actual use of a little known internet voting system named RIES. The system itself is very interesting because of 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. In general we can say that RIES gives us more confidence towards the future of internet voting than the authors of [4] provide.

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] Hoogheemraadschap van Rijnland. System and Method for Electronic Voting, 2005. WO2005004023.

[3] E.-M. Hubbers and B. Jacobs. Stemmen via internet geen probleem. *Automatisering Gids*, 42:15, 2004.

[4] D. Jefferson, A. Rubin, B. Simons, and D. Wagner. Analyzing Internet Voting Security. *CACM*, 47(10):59–64, 2004.

[5] W. Pieters and J. Kiniry. Internet Voting Not Impossible. *CACM*, 48(2):12, 2005.

[6] H. Robers. Electronic elections employing DES smartcards. Master's thesis, Delft University of Technology, Dec. 1998. www.iscit.surfnet.nl/team/Herman/election.ps.

[7] G. Salomonsen. Cryptomathic. www.cryptomathic.com.

[8] 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.

[9] SURFnet. Workshop over internetstemmen (RIES) voor specialisten op het gebied van internetverkiezingen. www.surfnet.nl/bijeenkomsten/ries.

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