Eindhoven Institute for the Protection of Systems and Information

Description and Analysis of the RIES Internet Voting System

version 1.0
June 24, 2008

Engelbert Hubbers[1] e.hubbers@cs.ru.nl
Bart Jacobs[1,2] bart@cs.ru.nl, bjacobs@win.tue.nl
Berry Schoenmakers[2] berry@win.tue.nl
Henk van Tilborg[2] h.c.a.v.tilborg@tue.nl
Benne de Weger[2] b.m.m.d.weger@tue.nl

[1]Digital Security group
Institute for Computing and Information Sciences (ICIS)
Radboud University Nijmegen
Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands
http://www.ru.nl/ds/

[2]Eindhoven Institute for the Protection of Systems and Information (EiPSI)
Faculty of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
http://www.win.tue.nl/eipsi/

(this page intentionally left blank)

Samenvatting

RIES is een reeks systemen (RIES-2004, RIES-KOA, RIES-2008) voor elektronische verkiezingen via het Internet. Het is in de praktijk gebruikt voor middelgrote verkiezingen voor Waterschappen, en voor in het buitenland verblijvende kiezers bij parlementsverkiezingen.

We geven een beschrijving en analyse van de veiligheid van RIES, gebaseerd op de ons ter beschikking gestelde documentatie. Het doel is RIES begrijpelijker te maken voor alle belanghebbenden: beleidmakers, wetenschappers, verkiezingsambtenaren, implementatoren, en het grote publiek. Dit document maakt expliciet wat de aannames zijn bij RIES, welke beperkingen er aan kleven, welk veiligheidsniveau wordt bereikt, enz., gericht op beveiligingsaspecten, zowel de technische als de organisatorische en procedurele.

RIES integreert stemmen per Internet met stemmen per post, en is in die specifieke context ontwikkeld. Dat legt het kader vast van de eisen volgens welke RIES ontworpen is, namelijk in vergelijking met systemen voor stemmen per post. Vandaar dat enkele redelijke vereisten voor verkiezingen (zoals stemvrijheid) vanaf het begin niet gehaald konden worden. Een gevolg is dat de algemene vereisten voor stemsystemen zoals geformuleerd door de Commissie Korthals Altes niet allemaal gehaald worden: zowel stemvrijheid als vertrouwelijkheid worden niet structureel door het ontwerp van RIES gegarandeerd.

RIES bouwt op cryptografische primitieven, zoals eenmalige handtekeningen. Sleutels voor kiezers worden centraal aangemaakt. Er zijn geen anonieme kanalen. De structurele bescherming en beveiliging die de cryptografie kan bieden is daardoor nogal beperkt. Veel garanties van RIES zijn daardoor gebaseerd op organisatorische maatregelen, met name de aanmaak van kiezers-sleutels, de productie van stembrieven, aanvallen door ingewijden (vooral op de server), integriteit en authenticiteit van de software, en helpdesk-procedures.

RIES-2008 is in een open sfeer ontworpen en gebouwd. De broncode en documentatie zullen binnenkort publiek beschikbaar komen voor analyse. Daarnaast hebben de ontwerpers en organisatoren veel werk gestoken in het publiek bespreken van hun systeem.

De technische en organisatorische inrichting lijkt zorgvuldig ontworpen te zijn. Er zijn echter pragmatische elementen in het systeem—zoals het gebruik van vervangende stempakketten—die aangegrepen kunnen worden voor manipulatie en misbruik, met name door ingewijden. Het RIES-systeem voor stemmen per Internet kent mogelijk ook gevaarlijke manieren voor het manipuleren van verkiezingen, die in principe op grote schaal toepasbaar zijn en afwijken van die bij verkiezingen per post.

Een van de onderscheidende aspecten van RIES is dat onafhankelijke tellingen van het eindresultaat mogelijk is, alsook individuele stemcontrole. Dit is een interessante en nuttige eigenschap van RIES, die echter de structurele zwakheden niet compenseert.

Wij zien RIES (m.n. RIES-2008) als een project dat waardevolle praktische ervaring en expertise oplevert over hoe een elektronische verkiezing georganiseerd en gehouden kan worden. RIES-2008 is niet geschikt voor gebruik buiten een context van verkiezingen per post, in het bijzonder niet voor 'algemene' verkiezingen (zoals voor nationale of Europese parlementen of plaatselijke of regionale raden en staten). We juichen verder onderzoek, ontwikkeling en experimenten toe om daarmee meer ervaring op te doen op dit gebied.

Summary

RIES is an evolving family of systems (RIES-2004, RIES-KOA, RIES-2008) for electronic elections via the Internet. It has been used in practice for medium scale elections for the Dutch District Water Control Boards and for expatriates in national parliament elections.

We describe and analyze the security of RIES, based on the documentation made available to us. The aim is to make RIES easier to understand for all parties involved: policy makers, scientists, election officials, implementors, and the general public. This document makes explicit what the assumptions in RIES are, what kind of restrictions apply, what level of security is achieved, etc., focussing on the security aspects, both technical and organizational / procedural.

RIES provides integration of Internet voting and voting by regular mail, and has been developed in that specific context. This has set the framework of requirements for the design of RIES to comparison with postal voting systems. Hence certain reasonable goals for elections (like vote freedom) have been out of scope from the start. Consequently, the general voting requirements formulated by the Korthals Altes Committee are not all satisfied: not only vote freedom but also vote integrity and confidentiality are not structurally guaranteed in the RIES design.

RIES is built on certain cryptographic primitives, like one-time signatures. Keys for individual voters are generated centrally. There are no anonymous channels. The structural protection and safeguards offered by cryptography are therefore rather limited. Many of the guarantees in RIES thus rely on organizational controls, notably with respect to (voter) key generation, production of postal packages, insider attacks (especially at the server), integrity and authenticity of the software, and helpdesk procedures.

RIES-2008 is designed and built in an open spirit. Its source code and documentation will shortly be available openly for inspection and analysis. Additionally, the designers and organizers have put considerable effort in publicly explaining and discussing their system.

The technical and organizational set-up seems carefully designed. There are however pragmatic elements in the system—such as the use of replacement packages—that are open to manipulation and abuse, notably by insiders. The RIES Internet election system also offers potentially dangerous ways for manipulation of elections, in principle applicable on a large scale and different from attacks on postal elections.

One of the distinguishing aspects of RIES is that it allows independent recounts of the final outcome and individual checks to see if own votes have been included. This interesting and useful feature does however not compensate for the structural weaknesses.

In a larger context we see RIES (esp. RIES-2008) as a project that yields valuable hands-on experience and expertise on how to organize and run electronic elections. We do not think RIES-2008 is a suitable system for use outside a context of postal elections, and in particular not for 'general' elections (like for national/European parliaments or local/regional councils). We do encourage further research, development and experiments to gain more experience in this area.

Contents

1 Introduction

1.1 The origin of RIES

In 2004, voting over the Internet has been presented as an option (next to the traditional voting by mail) by the Dutch District Water Control Boards *Rijnland* and *De Dommel*. For this purpose RIES-2004 was developed, the first practical implementation of the RIES concept. RIES stands for *Rijnland Internet Election System*. The main reason for introducing Internet voting was to increase the turnover, which traditionally was quite low for these District Water Control Board elections. About 120 000 voters (out of 2.2 million eligible voters) used RIES-2004 to cast their votes via the Internet.

The basic idea of RIES comes from a master thesis [26] by a student Herman Robers, written under supervision of Piet Maclaine Pont. This system was based upon smartcards (for holding the voter's secret key and for cryptographic operations) and was used as a starting point for RIES, of which Maclaine Pont is the main designer. 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.

One of the main distinguishing features of RIES is that it enables voters to verify after the election is closed that their own votes have been counted correctly, and that the result of the tally corresponds to the cast votes.

In 2006, a modified version, called RIES-KOA[1], was made available to Dutch voters outside the Netherlands, to enable them to use the Internet to take part in the November elections for members of the Parliament. Almost 20 000 voters used RIES-KOA to vote by Internet.

This year (2008), an enhanced version, called RIES-2008, is being developed for Internet and regular mail elections for all Dutch District Water Control Boards, which will take place in November 2008, with an estimated number of 12.3 million eligible voters. With an expected turnover of 20-40% of which 50-75% may use Internet voting, the number of Internet voters using RIES-2008 might very well considerably exceed 1 million. At the time of writing (June 2008) a decision on its actual use still has to be made.

RIES is patented [14] by Maclaine Pont and the Rijnland District Water Control Board. All RIES applications were designed and implemented by Arnout Hannink's company Magic Choice, owning the source code. An intentional agreement has been signed in which all

[1] KOA stands for *Kiezen Op Afstand*, i.e. *Voting Remotely*.

RIES intellectual property rights, including patents and source code, will be transferred to the *Unie van Waterschappen* (the union of District Water Control Boards), and made publicly available under the GPL version 3 conditions at `www.openries.nl`. The RIES server and network infrastructure is hosted and managed by SURFnet.

1.2 Aim and scope of this report

Late last year, *Het Waterschapshuis*, the ICT-organization of the Dutch District Water Control Boards, decided to call for an evaluation and analysis of RIES-2008. Soon, it was clear that to be able to do that, a proper description of the RIES concept was also very much needed. Existing documentation was fragmented over many different manuscripts, often written for different purposes and with different intended readers. For an in-depth understanding of the intricacies the only available documentation was extensive and technically very detailed, written mainly for implementors.

The successive improvements to the RIES concept, as well as the ongoing development of RIES-2008, complicated a clear security analysis; changes were still been made, details were further filled in, etc. Moreover, in the existing documentation, design criteria are not always given and design choices not always discussed.

The aim of this report is given by its title: *Description and Analysis of the RIES Internet Voting System*.

By providing this report, we also hope to realize a secondary goal, namely to give the system a transparency that will make the workings of RIES much easier to understand for all parties involved: policy makers, scientists, election officials, implementors, and the general public interested in a proper understanding of the security of the RIES voting system. This document makes explicit what the assumptions in RIES are, what kind of restrictions apply, what level of security is achieved, etc., focussing on the security aspects, both technical and organizational / procedural. Issues of availability, robustness and reliability however are only touched upon, not studied in detail.

This report focuses on the RIES concept in general and RIES-2008 in particular, where it should be realized that at the time of this writing the development of RIES-2008 was still not finalized. Some major differences between the various versions will be discussed. It is not an audit report, but a design study, as much as possible based on written documentation only.

It may be good for the reader to know that this report has been commissioned to LaQuSo[2] and EIPSI[3] by the Waterschapshuis, but with the explicit provision of a completely independent judgement. Several of the authors had some involvement in earlier RIES elections. The Radboud University Nijmegen (RUN) Digital Security group staged an independent vote counting during RIES-2004 and RIES-KOA as well as a website where voters could

[2] LaQuSo (`www.laquso.com`) is the *Laboratory for Quality Software*, a research institute of the Eindhoven University of Technology (TU/e) and the Radboud University Nijmegen (RUN).

[3] EiPSI (`www.win.tue.nl/~eipsi`) is the *Eindhoven Institute for the Protection of Systems and Information*, the TU/e information security research institute.

verify that their vote was counted correctly, thereby making use of its own software but relying on the same input data (votes) as RIES (see also [6]). They came to exactly the same outcome for both elections. The RUN group also performed a small audit of the server used for RIES-2004. One of the Technische Universiteit Eindhoven (TU/e) authors acted as independent umpire during RIES-KOA, but, as it turned out, nobody called for his intervention.

The outline of this report is as follows. Chapter 2 discusses the possible requirements and properties that one may like to impose on Internet voting schemes. Chapter 3 describes and discusses the cryptographic characteristics of Internet voting schemes as can be found in the literature, and places RIES in this spectrum. Chapter 4 gives a detailed description of the RIES design (phases, components, cryptographic protocol, etc.). Chapter 5 follows with an elaborate security analysis. Overall conclusions are gives in Chapter 6.

2 Requirements for Internet voting systems

There are several formulations of requirements for (e-)voting systems. Here we shall take as starting point the eight requirements formulated in the report [11] from 2007 of the committee *Korthals Altes*, commissioned by the Dutch government. It lists the following eight requirements.

1. **transparency** The election process should be organized in such a way that the structure and organization is clear, so that everyone in principle can understand it. There must be no secrets in the election process: questions must be able to be answered, and the answers must be verifiable.

2. **verifiability** The election process should be objectively verifiable. The verification tools may differ, depending on the method of voting that is decided upon.

3. **fairness / integrity** The election process should operate in a proper manner, and the results must not be capable of being influenced other than by the casting of lawful votes.

4. **eligibility to vote** Only persons eligible to vote must be allowed to take part in the election.

5. **equal suffrage / unicity** Each voter, given the Dutch election system, must be allowed to cast only one vote in each election, which must be counted precisely once.

6. **free suffrage / vote freedom** Every elector must be able to choose how to vote in complete freedom, free from influence.

7. **secret suffrage / vote secrecy** It must be impossible to connect the identity of a person casting a vote to the vote cast. The process should be organized in such a way that it is impossible to make a voter indicate how he or she voted.

8. **accessibility** Voters should be enabled as far as possible to participate directly in the election process. If this is impossible, there must be a way of taking part indirectly, *i.e.* by proxy.

This list is based on recommendations of the council of Europe [3]. There are many other sources, such as [1].

The precise meaning of these requirements is non-trivial and already involves interpretations. Some of the requirements seem relatively clear, like unicity: no voter should be able to vote more than one time. But does this also involve the requirement that votes cannot be duplicated? Or is non-duplication part of the integrity requirement?

The transparency requirement also seems obvious at first sight. But how far should it go? In electronic elections based on cryptographic techniques there are usually certain secret keys that should not be transparant, at least not during the actual operation of the election. Does the transparency requirement automatically mean that open source software must be used? Note that the formulation used above does not prescribe the absence of secrets in general, but only of secrets in the election *process*.

Apart from questions about the precise meaning of such requirements there are concerns about conflicts (or tensions) between them. Resolving these conflicts is precisely what makes (e-)voting such a challenging discipline. For instance, there is a tension between eligibility and vote secrecy: identification of voters is needed to ensure that only eligible voters participate, but there should be no way to connect voters' identities to their votes in order to achieve vote secrecy. Similarly there is a tension between vote secrecy and verifiability: if all the individual steps can be logged in detail it is not so difficult to ensure correctness of the outcome. In general there is a tension between convenience (as in accessibility) and security (secrecy, integrity).

In this paper we do not wish to discuss these requirements at the highest level of generality. After all, we focus on one particular system only, namely RIES-2008. Therefore we can further explicate the requirements in the concrete context of RIES-2008, if needed.

The basis of RIES-2008 was developed in the context of regional District Water Control Board elections within the Netherlands. Voting for these boards is usually done via ordinary mail. Comparison with such mail elections has thus been leading during the design of RIES-2008: the system was required to be at least as 'good' or 'secure' as postal elections.

One of the main concerns in postal elections is vote freedom: voters can be coerced to cast a particular vote, either by immediate presence of the coercer, or by demanding (a copy of) the posted vote afterwards. A related concern is that voters may sell their vote. The RIES-2008 system does not (try to) address these concerns, simply because of the context of postal voting in which it was developed. Hence we can already draw the easy conclusion that RIES-2008 does not fully satisfy the requirements of [11]. A more refined analysis is needed.

3 Cryptographic characteristics of Internet voting systems

3.1 General characteristics

Cryptography is an important provider of mechanisms for secrecy, integrity and authentication. Therefore it is not surprising that cryptography is at the heart of the security design of many Internet voting systems. RIES is an example of this. In this chapter we give a brief overview of the major cryptographic characteristics of Internet voting systems (or, more generally, remote voting systems in which votes are cast through some public network).

A voting system typically centers around the following cryptographic protocols. The most visible protocols are the voting protocol, which is used by voters to cast their votes, and the tallying protocol, which is used to count the votes. A further major protocol is usually the set-up or initialization protocol (which also may involve registration of the voters).

We limit the discussion to voting protocols which do not require any interaction between the voters, hence each voter will be able to cast its vote independently, and possibly in parallel, to other voters. A useful cryptographic view of such an Internet voting system is obtained by focusing on the information stored by the voting servers upon completion of the voting protocol by each voter. We will refer to this information as the *voted ballot*. The voted ballots are used by the tallying protocol to determine the election result.

For voted ballots we consider three main characteristics:

- voted ballot contains vote in the clear vs. encrypted form;
- vote or voted ballot is authenticated or not;
- voted ballot is anonymous vs. non-anonymous.

We briefly comment on each of these characteristics in turn.

The most common case is that voted ballots contain the votes in encrypted form. To this end, public key encryption is usually employed, avoiding the need for voters to share a secret key with the voting servers (or tallying authorities). Decryption of the voted ballots will only be performed as part of the tallying protocol. One of the major reasons for the use of encryption is to block the computation of intermediate election results, as this is not allowed in many elections.

Also, voted ballots (or the votes therein contained) are usually authenticated, thereby showing that the voted ballot originates from a registered voter. (In some voting systems, however, voters may simply log on to the voting server and the voting server will basically store the (possibly encrypted) vote; this puts a lot of trust in the server.) The strength of authentication mechanisms employed varies a lot between voting systems. Authentication must be sufficiently strong to ensure that voters cannot cast more than one vote (and, of course, that only eligible voters can vote at all).

Whether voted ballots are anonymous or non-anonymous is a major characteristic of an Internet voting system, which is strongly related to the way ballot secrecy is ensured by the system. In general, anonymous voted ballots are used to hide 'who is voting' and non-anonymous (identifiable) ballots are used to hide 'what someone is voting for'. Normally, an anonymous voted ballot is required to be delivered through an anonymous channel to hide data such as the IP address of the voter's computer.

3.2 Characteristics of the RIES protocol

In terms of the above cryptographic characteristics, RIES is a voting system in which voted ballots (i) contain the votes in the clear, (ii) are authenticated by means of a digital signature, and (iii) are required to be anonymous. The use of cryptography is in fact limited, as the anonymity of the voted ballots depends completely on trust (or, in any case, non-cryptographic procedures). In the context of the RIES protocol, cryptography is used solely for the authentication of the voted ballots.

3.3 Succinct cryptographic description of the RIES protocol

The type of authentication used in RIES is best understood in terms of so-called *one-time digital signature schemes*. These digital signature schemes are well-known in cryptography and were introduced in the mid 1970s, at the same time when public key cryptography was invented by Diffie and Hellman. See [21, Section 11.6] for a few example schemes and references.

A one-time digital signature schemes shares the same functionality as an ordinary digital signature scheme (such as schemes based on RSA or DSA), hence consists of the following three algorithms:

- **Key generation.** A probabilistic algorithm that on input of a security parameter k, generates a key pair (sk, pk) consisting of a private key sk and a public key pk.
- **Signature generation.** A (potentially) probabilistic algorithm that on input of a message m and a private key sk, outputs a signature s.
- **Signature verification.** A (usually) deterministic algorithm that on input of a message m, a public key pk, and a signature s, determines whether s is a valid signature on m with respect to public key pk.

The difference with an ordinary signature scheme is that the unforgeability of a one-time digital signature scheme is only guaranteed as long as no more than one signature is generated for a given key pair.

The restricted use of a key pair will provide no problems for application in voting systems, as each voter is supposed to cast a single vote only in the first place (voters will use newly generated key pairs in each election). The implication of the restricted use, however, is that

one-time signature schemes can be implemented much more efficiently than ordinary signature schemes. For example, the Lamport one-time signature scheme (see [13], mentioned and attributed to Lamport already in [4, p.650]) can be instantiated using just a cryptographic hash function such as SHA-1 or SHA-256 (without the need for number-theoretic one-way functions such as modular exponentiation).

To illustrate the basic idea of a one-time signature scheme we will describe the most basic one, namely a signature scheme which can be used to sign messages consisting of a single bit only. The scheme is defined in terms of a cryptographic hash function h, mapping bit strings of arbitrary length into bit strings of length 160, say. The algorithms are as follows:

- **Key generation.** Pick two k-bit strings $x_0, x_1 \in \{0,1\}^k$ uniform at random. The private key is $sk := (x_0, x_1)$ and the public key is $pk := (y_0, y_1) := (h(x_0), h(x_1))$.
- **Signature generation.** Given a bit $m \in \{0,1\}$, the signature is set to $s := x_m$.
- **Signature verification.** Given a message m, a public key $pk = (y_0, y_1)$, and a signature s, verify that $h(s) = y_m$ holds.

This basic scheme can easily be extended to accommodate larger messages. Also, as is immediate from the construction, it is possible to compute the message m from the signature s, with little effort only; hence, message recovery is achieved in a natural way. Furthermore, in practice one generates the private key pseudorandomly from a short seed, thus it is easy to limit the size of private keys. (The pseudorandom generator can be defined in terms of a cryptographic hash function as well.) In combination with Merkle's authentication trees (introduced at the end of 1970s; see [22] and references therein), the size of public keys can be limited as well. In general, many trade-offs are possible between the speed of signature generation, verification, and the size of the keys; see, e.g., [2] for a rather general treatment.

In terms of a one-time digital signature scheme $\mathcal{S}$, the RIES protocol can now be described succinctly as follows. Basically, a unique key pair is generated for each voter using the key generation algorithm of $\mathcal{S}$. This is done before the election. During the election each voter will cast its vote by generating a one-time signature using scheme $\mathcal{S}$, where the vote plays the role of the message. The signed vote (voted ballot) can be verified (using scheme $\mathcal{S}$) once its received by the voting server, or later, when all valid votes are added together. Note that anyone can count the votes as no encryption is used.

At this level of detail we can already make the following observations regarding the cryptographic properties of the RIES protocol. First of all, although the voter should be the only party knowing the private key (as in ordinary signature schemes), the RIES protocol in fact generates the key pairs in a central place, from which the key pairs are then distributed to the voters. This gives rise to several major issues: (a) voters must trust that they are the only ones knowing their private keys, (b) in case private keys leak anyway, voters cannot dispute this fact, as anyone knowing a private key is equally powerful and may cast votes on behalf of the voters, (c) voters must trust that no one keeps track of who gets which key pair, as this would directly break ballot secrecy. Thus not only the generation of the key pairs, but also the delivery to the voters relies on trust in the parties involved (delivery

can be thought of as done via an anonymous channel; the anonymous channel is, however, not done cryptographically but physically, using printers and envelopes).

Furthermore, the votes should actually be sent to the voting server using an anonymous channel, as the votes are cast in the clear. An anonymous channel, however, is a very complicated and expensive cryptographic primitive. Therefore, the RIES protocol simply assumes that the voting server can be trusted not to record the IP addresses from which the voters submit their votes (via an SSL connection). The voting server must also be trusted not to compute intermediate election results, based on the votes received so far.

The use of cryptography is thus rather limited in the RIES protocol. Moreover, as the key pairs are generated in a pseudorandom way from a master key, the effectiveness of the one-time signature scheme is reduced, as knowledge of the master key allows one to cast votes on behalf of anyone.

Consequently, the security of the RIES system will mainly depend on the use of administrative procedures and trust in these procedures. Any security requirement beyond the scope of the RIES cryptographic protocol should be handled by other measures.

4 Description of the RIES design

This chapter gives a detailed description of the design of RIES-2008, based on the available documentation. For easy referring we start with listing the data structures that are central to the RIES design. Then we give a description of all that happens during the different phases of a RIES election, the main components and architecture are described, and the RIES cryptographic protocol which is at the heart of the security design is studied in detail.

4.1 Data structures

This section describes the data structures relevant for our detailed description of RIES-2008, such as identifiers, cryptographic keys, codes derived by cryptographic means, paper forms and digital files. Each data structure has an attribute to denote its level of confidentiality: "public" if anyone is allowed to access it; "voter" if only one specific voter should have access, or "system" if no persons at all, or at most only RIES officials, may have access. Data structures that contain values that are specific for a voter are denoted by a subscript i, data structures related to replacement and test votes are denoted by a subscript i', while a subscript j denotes an individual candidate.

The following cryptographic mechanisms are used:

- `DES`, single DES encryption, used in `DESmac` and as the block cipher inside `MDC-2`,
- `3DES`, double key triple DES encryption, used in `DESmac`, to encrypt sensitive system files, and for key derivation,

- `DESmac`, Message Authentication Code based on `DES` or `3DES`, used for authentication of voters and ballots (when applied to one 8-byte block of data `DESmac` is just DES-ECB encryption),
- `MDC-2`, Modification Detection Code MDC-2, a hash function based on `3DES`, used for one-way hashing to anonymize sensitive data in order to enable public verification,
- `RSA`, RSA public key encryption, used for wrapping / unwrapping of `3DES` keys and for certification of public keys,
- `SHA-1`, SHA-1 hash algorithm, used to generate public commitment values for public file contents.

The MAC of message `m` under key `k` will be denoted by `DESmac(k,m)`. String concatenation will be denoted by $\|$.

4.1.1 Identifiers

Table 1 lists identifiers that are not derived by cryptographic means.

name	**status**	**size**	**description**
`ElID`	public	2 bytes	Election Identifier
`ElCd`	public		Election code, human-friendly form of `ElID`
`BalBxID`	public	4 bytes	Ballot box identifier, expanded version of `ElID`
$\texttt{Vn}_i$	voter, system		Voter name and address
$\texttt{VnID}_i$	voter, system	5 bytes	Voter Identifier (simple encoding of the Dutch Social Security Number (BSN) or another identifier (A-number) from the citizen administration (GBA))
$\texttt{VrID}_{i'}$	voter, system	5 bytes	Replacement Package Identifier, similar format as $\texttt{VnID}_i$
$\texttt{VtID}_{i'}$	system	5 bytes	Test Package Identifier, similar format as $\texttt{VnID}_i$
`ParGp`	public	1 byte	Participation Group (to allow for voter groups, in RIES-2008 this is one fixed value)
`ExtParGp`	public	2 characters	Extended Participation Group, human-friendly version of `ParGp`
$\texttt{AbelPI}_i$	voter, system	8 bytes	Abuse Elimination Personal Information (per voter), this sometimes refers to the last two digits of the voter's year of birth, and sometimes to an 8-byte value derived from it
$\texttt{Cm}_j$	public	6 bytes	Candidate Identifier

Table 1: Identifiers.

4.1.2 Cryptographic Keys

Tables 2 and 3 list symmetric and asymmetric cryptographic keys. Also codes that are essentially keys in a different representation are listed here.

The asymmetric key pairs are RSA key pairs with a modulus size of at least 2048 bits [17]. Public keys are authenticated in X.509 certificates. The Certification Authority conforms to the requirements of the Dutch Government PKI[4]. SSL key pairs, used for remote access to the Portal (both by voters and RIES operators), are not mentioned in this section, as they are not directly used in the RIES core protocol. These SSL key pairs are managed by SURFnet, under responsibility of the Waterschapshuis.

name	**status**	**key length**	**description**
`Kgenvoterkey`	system	112 bits	`3DES` key for generating voter keys $\texttt{Kp}_i$
$\texttt{Kp}_i$	voter	56 bits	Voter key, `DES` key, generated at initialization and distributed to the voter in the election package; to obtain $\texttt{Kp}_i$ the 8-byte message "$\texttt{VnID}_i \| \texttt{ElID} \| \texttt{ParGp}$" is encrypted using `3DES` with the key `Kgenvoterkey`, note that replacement and test values $\texttt{VrID}_{i'}$, $\texttt{VtID}_{i'}$ are treated similarly in the generation of replacement and test voter keys
$\texttt{VPID}_{1,i}$, $\texttt{VPID}_{2,i}$	voter, system	56 bits	Voting code, two halves of $\texttt{Kp}_i$, encoded to 16 characters from an alphabet of size 34 by a simple alphanumeric encoding scheme called 34AN (to make them human-friendly as the voters have to enter these values on screen, the splitting into two halves seems not essential)
$\texttt{OCR}_i$	voter, system		Machine readable encoding of the `3DES` encryption of the message "$\texttt{Kp}_i \| \texttt{AbelPI}_i$" by the key `Kkpocr`, used to transform postal into electronic vote
`Kkpocr`, `Kocrkp`	system	112 bits	`3DES` key, used as `Kkpocr` for encryption of $\texttt{Kp}_i$ into $\texttt{OCR}_i$, and as `Kocrkp` for decryption
`Kc10`	system	112 bits	Printer `3DES` session key, used for encryption and decryption of `WV-STUF-C10.xml`
`Kbbs_0`	system	112 bits	`3DES` key for generating receipts
`KabelPi`	system	112 bits	`3DES` session key for encrypting and decrypting the Abel-part of $\texttt{OCR}_i$
`Kpretal`	system	112 bits	`3DES` session key for encrypting and decrypting pre-tally result reports

Table 2: Symmetric keys.

[4]PKI Overheid, see `www.pkioverheid.nl`.

name	status	key length	description
PKpsbc10, SKpsbc10	public, system	1024 bits	Printer public RSA key pair, used for wrapping resp. unwrapping Kc10, the private key is to be destroyed at least 24 hours before starting the election
PKpsbttp, SKpsbttp	public, system	1024 bits	TTP public RSA key pair, used to certify the origin of PKpsbc10
PKnotpretal, SKnotpretal	public, system	1024 bits	Notary public RSA key pair, used for wrapping resp. unwrapping Kpretal
PKnotttp, PKnotttp	public, system	1024 bits	TTP public RSA key pair, used to certify the origin of PKnotpretal

Table 3: Asymmetric key pairs.

4.1.3 Cryptographically derived codes

Table 4 lists codes that are derived by cryptographic means but that are not keys themselves.

name	status	size	description
$\texttt{RnPID}_i$	public	8 bytes	Reference pseudo voter identity, as computed at initialization and stored in the pre-election table, $\texttt{RnPID}_i = \texttt{MDC-2}(\texttt{VnPID}_i)$
$\texttt{RnCx}_{i,j}$	public	8 bytes	Reference potential vote per voter and per candidate, stored in the pre-election table, $\texttt{RnCx}_{i,j} = \texttt{MDC-2}(\texttt{VnCx}_{i,j})$, for not only the actual $\texttt{VnCx}_{i,j}$ but for each possible values of it
$\texttt{RnpotVote}_i$	public		All potential votes per voter, i.e. for given i all possible $\texttt{RnCx}_{i,j}$
$\texttt{ReSPID}_i$	voter, system	8 bytes	Alternative reference pseudo voter identity as computed at initialization and stored in the status tracking file, the MDC-2 of the DESmac of the padded message "ElID‖ExtParGp" with the key $\texttt{Kp}_i$
$\texttt{VnPID}_i$	voter	8 bytes	Pseudo voter identity, as computed by the voter as the DESmac with the key $\texttt{Kp}_i$ of the padded message "BalBxID")

continues on next page

continued from previous page

name	status	size	description
$\texttt{VnCx}_{i,j}$	voter	8 bytes	Actual vote, as computed by the voter as the DESmac with the key $\texttt{Kp}_i$ of a message based on $\texttt{Cm}_j$ (for the voter's choice of the candidate) and $\texttt{AbelPI}_i$
$\texttt{VirtualBallot}_i$	voter, system	16 bytes	Virtual Ballot, consisting of $\texttt{VnPID}_i$ and $\texttt{VnCx}_{i,j}$
$\texttt{RnRecVote}_i$	public	16 bytes	Reference value of a recorded vote, consisting of the concatenation of the MDC-2's of $\texttt{VnPID}_i$ and $\texttt{VnCx}_{i,j}$
$\texttt{VotRecCon}_{i,j}$	system	8 bytes	Voter Receipt Confirmation, computed by the ballot box server as the DESmac of the message "$\texttt{VnPID}_i \| \texttt{VnCx}_{i,j}$" with the key Kbbs_0
$\texttt{VotRecConSvr}_{i,j}$	system	4 bytes	Receipt for the umpire, the upper 4 bytes of $\texttt{VotRecCon}_{i,j}$
$\texttt{VotRecConCnt}_{i,j}$	voter	4 bytes	Receipt for the voter, the lower 4 bytes of $\texttt{VotRecCon}_{i,j}$
VotValVal	voter	20 bytes	The receipt string sent back to the voter, equal to $\texttt{VnPID}_i \| \texttt{VnCx}_{i,j} \| \texttt{VotRecConCnt}_{i,j}$

Table 4: Cryptographically derived codes

4.1.4 Paper forms

Table 5 lists paper forms and election packages.

name	status	description
Postal Ballot Form	voter	On the ballot form $\texttt{OCR}_i$ is shown in machine readable form, containing in encrypted form the voter key $\texttt{Kp}_i$ (equivalent to voting code $\texttt{VPID}_{1,i}$, $\texttt{VPID}_{2,i}$) and $\texttt{AbelPI}_i$
Voting Card	voter	On the voting card the voting code ($\texttt{VPID}_{1,i}$, $\texttt{VPID}_{2,i}$, equivalent to the voter key $\texttt{Kp}_i$) and the ElID are shown in human readable form

continues on next page

continued from previous page

name	status	description
ElPac	voter	Election Package, containing the Postal Ballot Form and Voting Card for voter $\texttt{VnID}_i$ in a closed envelope, $\texttt{Vn}_i$ is printed on the outside of the envelope
RepElPac	voter	Replacement Election Package, containing a Postal Ballot Form and Voting Card for replacement voter $\texttt{VrID}_{i'}$ in a closed envelope, $\texttt{VrID}_{i'}$ is printed on the outside of the envelope
TstElPac	system	Test Election Package, containing a Postal Ballot Form and Voting Card for test voter $\texttt{VtID}_{i'}$ in a closed envelope, $\texttt{VtID}_{i'}$ is printed on the outside of the envelope

Table 5: Paper forms.

4.1.5 Files

Table 6 lists all files and their record structure. All public files are published on a website, and their integrity is protected by publishing the SHA-1 hash value in a newspaper.

name	status	description
`WV-STUF-B10.xml`	system	Management data about an election
`WV-STUF-C10.xml`	system	Input for the printer, contains per voter $\texttt{VnID}_i$, $\texttt{Vn}_i$, $\texttt{VPID}_{1,i}$, $\texttt{VPID}_{2,i}$, the file is encrypted using `3DES` with a fresh random key, this `3DES`-key is encrypted using `RSA` with `PKpsbc10`, replacement and test packages data are present in the same file; this extremely sensitive file (with all its copies) is to be destroyed at least 24 hours before starting the election
`WV-STUF-K10.xml`	system	Voter Registry, contains per voter $\texttt{VnID}_i$, $\texttt{Vn}_i$ and the day of birth from which $\texttt{AbelPI}_i$ can be derived, this file is privacy sensitive
`WV-STUF-K11.xml`	system	Similar to `WV-STUF-K10.xml`, containing records for both replacement voters and test voters
`WV-STUF-K30.xml`	system	File that is being used to transfer the information from scanned postal ballots (a.o. $\texttt{OCR}_i$, $\texttt{AbelPI}_i$, and the candidate number) to RIPOCS who converts this information into technical votes

continues on next page

continued from previous page

name	status	description
WV-STUF-K50.xml	public	Party and candidate registry, contains one 'blanco' candidate
WV-STUF-H00.xml	system	The data files WV-STUF-B10.xml, WV-STUF-C10.xml, WV-STUF-K10.xml, WV-STUF-K30.xml and WV-STUF-K50.xml are always accompanied by a WV-STUF-H00.xml file that contains a hash value of the data file, in order to enable validating the integrity of the data file
Status tracking file	system	Contains status values, indexed by $\texttt{ReSPID}_i$
Reference table	public	Contains all possible votes in the form $\texttt{RnpotVote}_i$, will be published on the web before the election starts; its SHA-1 hash value will be published in a newspaper
Virtual Ballot Box	system	Contains all actual votes in the form $\texttt{VnPID}_i \| \texttt{VnCx}_{i,j}$
IRecVote file	public	Internet Received Votes, published on the web after the election is closed, its SHA-1 hash value will be published in a newspaper
PRecVote file	system	Postal Received Votes
RecVote file	public	Merge of IRecVote and PRecVote, published on the web after the election is closed, its SHA-1 hash value will be published in a newspaper
RecPostBal file	system	Postal Ballots
RepElPac File	public	Replacement Election Package file, contains $\texttt{VrID}_{i'}$'s of issued replacement packages, published on the web after the election is closed, its SHA-1 hash value will be published in a newspaper
TstElPac File	system	Test Election Package file, contains $\texttt{VtID}_{i'}$'s
Mutations File	public	Mutations to reference file due to issuing replacement packages, published on the web after the election is closed, its SHA-1 hash value will be published in a newspaper

Table 6: Files.

4.2 Phases

Before we will look at the specific phases in RIES-2008, we start by defining a list of actors inside the system. Sometimes these actors are real persons, sometimes they are systems. For background see [19].

UvW From the current documentation it is not really clear who is actually responsible for which part of the elections. Basically each District Water Control Board is responsible for its own election. However some tasks have been delegated to this *Unie van*

Waterschappen, for instance by signing a treaty that describes how the different District Water Control Boards will work together on this project. Furthermore, the UvW has on its turn delegated tasks to HWH. Since this paper is more about the technical issues of the RIES protocol we will not further address the issue of responsibilities.

DWCB The Netherlands consists of 26 District Water Control Boards (DWCBs). Each authority is responsible for its own election. This means that they have to check all the information with respect to the list of people entitled to vote, but also that they have to have their own helpdesk in order to administrate and distribute replacement forms.

HWH *Het Waterschapshuis* is a special ICT department that works for all these 26 DWCBs. Its tasks are both technical (e.g. developing software and infrastructure for the election) and organizational (e.g. making sure that all DWCBs know how to deliver their data). They control the overall process of the elections.

Bestandendienst An outsourced service, which is taking care of all the files that have to do with the people entitled to vote. They seem to cooperate only directly with HWH and therefore we will consider them as part of HWH.

RIPOCS This is the isolated server for sensitive operations. For instance, it is used for generating the file that links voter keys to persons.

ROCMIS This is the so-called init and cloning server.

PORTAL This is not a person but a system. The system can be seen as a sort of workflow manager. It facilitates services in such a way that parties like HWH and DWCB can actually do the things they have to do. Its tasks include integration of all received votes, prepare publications and offer them to HWH and many other tasks.

PSB Printing service bureau; the company that takes care of printing and mailing the ballot forms.

DPSB Desktop publishing service that basically supports PSB. Its task is mainly to make sure that specific stuff like all the appropriate logos for each DWCB is correct. Because their task is pretty small, we will see them as part of PSB.

VPSB The bureau that takes care of interpreting the votes that were cast by mail. If possible this is done by OCR techniques, but if needed this will include some manual input of ballot forms.

SURFnet This is basically the Dutch service provider for institutes of higher education. In RIES-2008 its task is mainly to support the technical infrastructure of both the PORTAL as well as the VotWin.

UMPIRE The umpire is a trusted third party that will look into all the complaints of voters after the election. He has specific technical possibilities to check whether complaints are valid or not.

VOTER An individual voter. Besides his usual role of casting a vote, the voter also has the ability to check his vote after the final tally process. If sufficiently many voters use this right, fraud should become detectable.

VotWin The Voting Window server. The application that takes care of actually receiving the Internet votes.

NOTARY The notary is used to act as a safe and trusted place for storing secret information that shouldn't be destroyed.

As with most electronic voting systems also RIES-2008 can be divided into three stages. The first stage is the so-called initialization phase, which takes care of everything that needs to be done before the actual elections start. The second stage is the time slot in which the actual voting takes place. The third and last stage is the so-called tallying phase. This phase includes all steps that are needed in order to derive the outcome of the election in such a way that the result is correct and verifiable.

4.2.1 Initialization – overview

This stage is also known as the *preparation stage*. It starts as soon as it is decided that an election will take place and end at the moment that the voting window is opened. The main tasks that need to be done are

- Establish an official election authority with an appropriate mandate.
- Determine the rules that entitle people to vote.
- Establish a list of people entitled to vote.
- Establish a list of all the candidates and their associated parties.
- Generate all essential cryptographic keys.
- Generate the personalized codes for each entitled voter and include them in the so-called pre-election reference tables. In addition, include some anonymous codes as well that can be used later on as so-called replacement codes.
- Publish the reference table together with its hash to make sure that each modification of this table can be detected.
- Generate the personalized information needed to print all ballot forms.
- Send all the printed forms to the voters.

Note that some of these actions have clear time (and order) constraints, while others can de done in parallel, independently. These constraints will become clear in the next section.

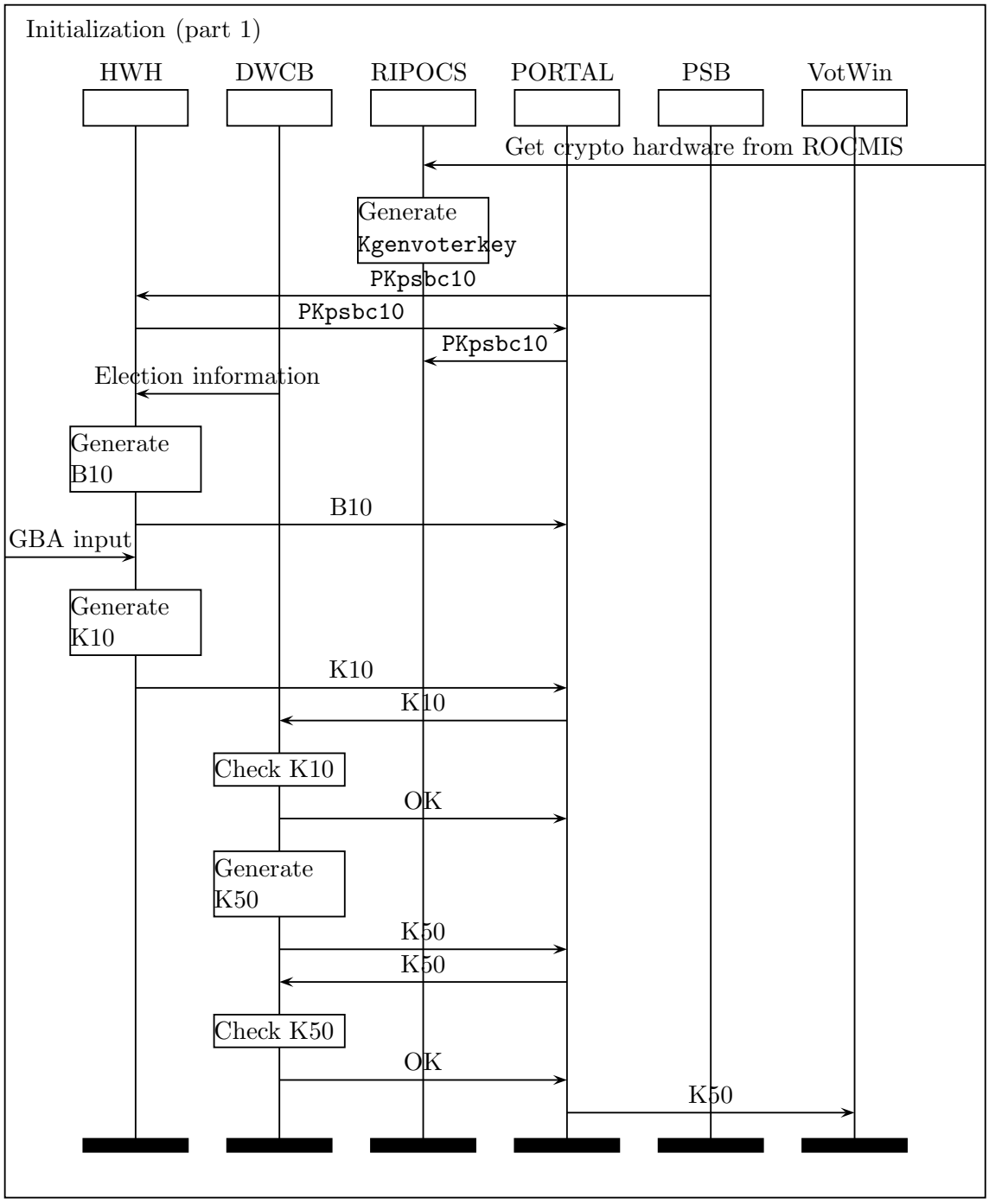

Figure 1: Initialization phase, part 1. Recall that B10 contains administrative information, K10 contains the list of voters and K50 the list of candidates.

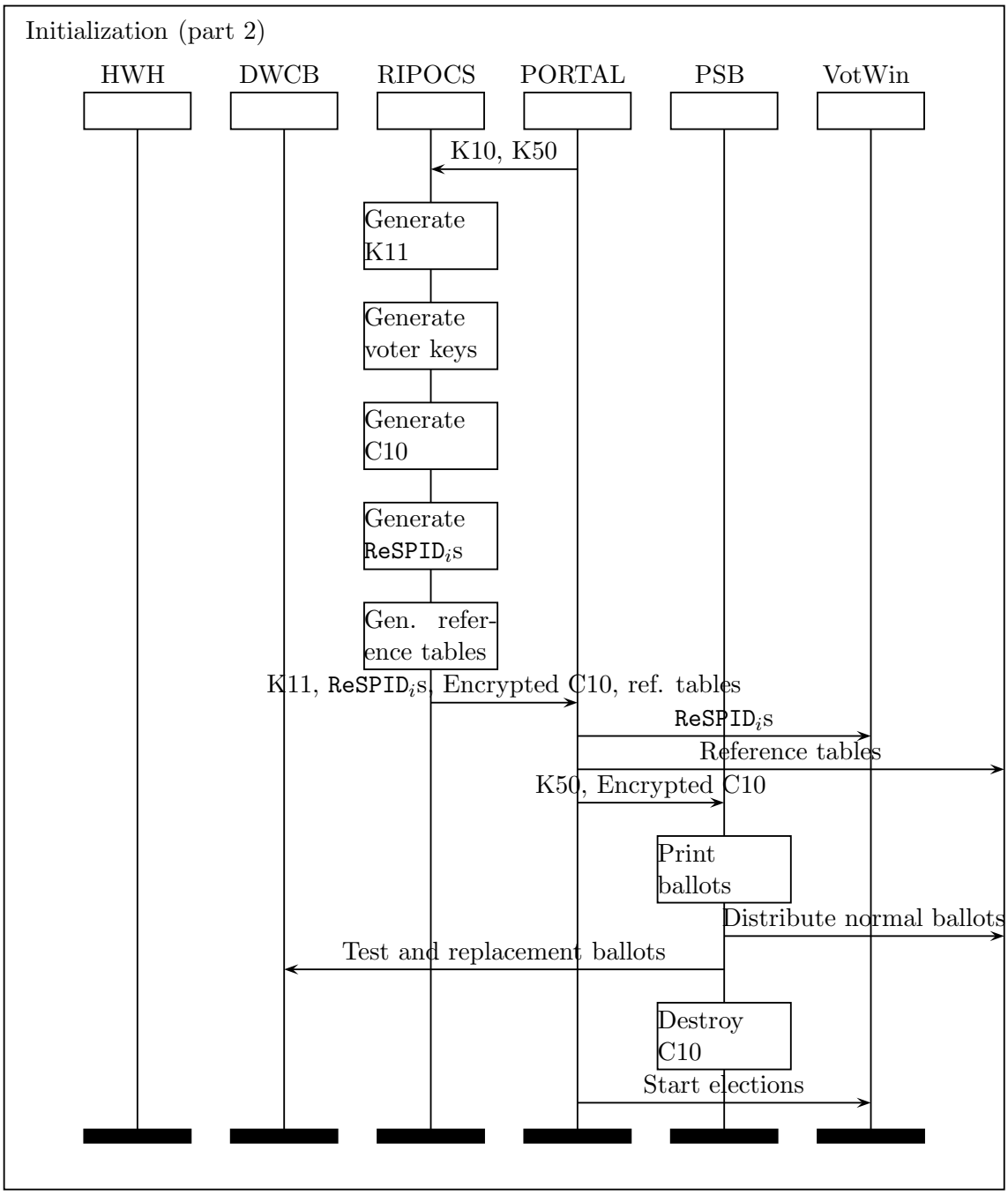

Figure 2: Initialization phase, part 2. Recall that K11 contains the replacement voters and test voters.

4.2.2 Initialization – in detail

In Figures 1 and 2 the important tasks in this phase are shown. Note that for some actions the exact time that they are performed compared to actions by other actors is not always relevant. However, the arrows that connect two actors always act as a time for synchronization. Furthermore, for reasons of simplicity, we do not include the `WV-STUF-H00.xml` file all the time. Basically, every time a file is transported from one system to another, the contents of the file are checked by computing a hash of the file and comparing it with the appropriate value that is sent in `WV-STUF-H00.xml`.

At the top of the MSC in Figure 1 we see that RIPOCS gets crypto hardware from ROCMIS. Although it is depicted as a message, it is not really a message that is transported over some channel. Before this preparation phase, in an off-line situation ROCMIS has been used to generate the device master key `KM` and 'copy' it onto different instances of the hardware crypto by cloning the device. See [16] for more details. Note that this preparation ends by setting the crypto cards in a state that they cannot be cloned anymore afterwards. One of these instances is the RIPOCS system. Or actually, RIPOCS is not just one machine but there are always three of them in order to detect possible mistakes. Actions by RIPOCS are only accepted if at least two of these three systems give the same result. However, from a conceptual point of view, RIPOCS is just one entity and hence we pretend as if it really is only one machine.

And now that RIPOCS has its master key `KM`, RIPOCS can generate the key `Kgenvoterkey` which will be used later on to generate the specific voter keys. Note that this cannot be done at the time that the same hardware is still in ROCMIS, because although the `Kgenvoterkey` cannot be exported from the card, it could be abused by ROCMIS. Furthermore, PSB creates its pair of public key `PKpsbc10` and private key `SKpsbc10` on its special crypto import pc [17]. It obtains a certificate on its public `PKpsbc10` from a certification authority. After that the certificate containing the public key `PKpsbc10` is sent to HWH. On its turn HWH installs `PKpsbc10` and the certificate on PORTAL, from which it is transported to RIPOCS.

Also quite at the top of the MSC we see that the DWCB sends some administrative information to the HWH like when the election exactly starts, when it stops, its short name etc. This is done by each of the 26 DWCBs. All this information is combined into one `WV-STUF-B10.xml` file by the HWH which is sent to PORTAL to define all the elections on the PORTAL.

A bit lower we see that HWH receives information from the Dutch GBA (citizen administration) containing the name and address information of all the people entitled to vote. Technically this is not done by HWH, but by Bestandendienst. Based upon these files and the rules that explain who is entitled to vote in a specific election, the file `WV-STUF-K10.xml` is created. Basically this file contains a list of all the voters, identified by a unique ID which is known in the system as $\texttt{VnID}_i$. Typically the Dutch BSN (comparable to the social security number in the US) or the so-called A-number (used in the GBA) is used here. However, it can be any unique number. See [19]. Within this list of voters, for each voter a list is created for which ballot boxes this voter is entitled to vote. Although there is only one

election, each election can consist of different ballot boxes. This `WV-STUF-K10.xml` is sent to the PORTAL, which sends it in its turn to the corresponding DWCB. It is the task of the DWCB to check whether this list is correct.

Besides checking the `WV-STUF-K10.xml`, the DWCB is also responsible for generating its `WV-STUF-K50.xml`. This is the file that contains all the parties and their associated candidates. In particular this file determines exactly how the names will show up on the printed ballot form, the ballot form on the screen, the result of the election, etc. Hence it is clear that it is very important to check that the names are written correctly. This is exactly why we see first that DWCB generates this file, sends it to PORTAL, which sends it back to DWCB, who checks it and finally indicates to PORTAL that it is OK. Although this may seem strange in the MSC, this is only because inside this MSC we do not distinguish between the different roles inside the DWCB. One role has responsibility for generating the file, another role later on has responsibility for checking the file.

When `WV-STUF-K50.xml` is approved, it will be sent to the VotWin, so it can be used to create the screens with all the candidates. This is the last line of Figure 1.

We now continue with what we see in Figure 2. This MSC starts when `WV-STUF-K10.xml` and `WV-STUF-K50.xml` are sent to RIPOCS. Although, technically, 'are sent' is not what happens. Because of security reasons it is not possible to connect to RIPOCS. Only RIPOCS itself is able to connect to PORTAL and look at a specific place to see whether the appropriate files are available and waiting to be copied to RIPOCS itself.

Based upon the statistics of `WV-STUF-K10.xml` RIPOCS decides how many test forms and how many replacement forms need to be created. These special ballots are described in the file `WV-STUF-K11.xml`, which looks a lot like `WV-STUF-K10.xml`, but doesn't contain real names, see also Section 4.2.6. Furthermore, these special voters are identified by an ID called $\texttt{VrID}_{i'}$ and not by $\texttt{VnID}_i$. These $\texttt{VrID}_{i'}$s can be chosen freely as long as they start with a 9 for replacement ballots and 99 for test ballots and are unique within the list of all $\texttt{VnID}_i$s and $\texttt{VrID}_{i'}$s together. After `WV-STUF-K11.xml` is created, RIPOCS uses its master key `Kgenvoterkey` to generate voter keys $\texttt{Kp}_i$ for each voter listed in `WV-STUF-K10.xml` and `WV-STUF-K11.xml`. To be more precise,

$$\texttt{Kp}_i = \texttt{3DES}(\texttt{Kgenvoterkey}, (\texttt{VnID}_i \| \texttt{ElID} \| \texttt{ParGp}))$$

for real voters in `WV-STUF-K10.xml`. For the replacement and test voters in `WV-STUF-K11.xml`, the computation is basically the same but $\texttt{VnID}_i$ is replaced by $\texttt{VrID}_{i'}$. Note that after $\texttt{Kp}_i$ is calculated in the rest of the process of generating the reference table there is no distinction between reference values for real voters or for replacement or test voters. The difference can only be seen by looking at the status bits in the reference table. Now RIPOCS is going to create `WV-STUF-C10.xml`. This is a very important file since it contains a lot of confidential information. For each voter the following information is linked:

- $\texttt{VnID}_i$ (or $\texttt{VrID}_{i'}$),
- Name,

- Address,
- Day of birth,
- $\texttt{Kp}_i$, split in two parts $\texttt{VPID}_{1,i}$ and $\texttt{VPID}_{2,i}$ and encoded in AN34,
- $\texttt{OCR}_i$.

If someone gets hold of this file he will in principle be able to vote on behalf of each registered voter. Furthermore, after the election he will be able to determine who voted for which candidate.

Therefore this `WV-STUF-C10.xml` is encrypted with the randomly generated 3DES CCA encipher key `Kc10`, which is generated as exportable key in crypto hardware and wrapped within the public key `PKpsbc10` to make sure that only PSB is able to decrypt the file. In particular it is not possible to use this `Kc10` to decrypt `WV-STUF-C10.xml` on RIPOCS.

Furthermore, these voter keys are used by RIPOCS to compute three important values:

- $\texttt{ReSPID}_i = \texttt{MDC-2}(\texttt{DESmac}(\texttt{Kp}_i, (\texttt{ElID}\|\texttt{ExtParGp})))$, the reference pseudo voter identity which is used for the status in VotWin.
- $\texttt{VnPID}_i = \texttt{DESmac}(\texttt{Kp}_i, f(\texttt{BalBxID}))$, the voter's pseudo identity,
- $\texttt{VnCx}_{i,j} = \texttt{DESmac}(\texttt{Kp}_i, f(\texttt{AbelPI}_i, \texttt{Cm}_j))$, the candidate's identity for this voter.

Here f is a simple padding function. The last two items are in their turn used to compute the

- $\texttt{RnPID}_i = \texttt{MDC-2}(\texttt{VnPID}_i)$,
- $\texttt{RnCx}_{i,j} = \texttt{MDC-2}(\texttt{VnCx}_{i,j})$.

These last two hashed values are used to build the reference table. Basically, the reference table is a table that contains all possible valid votes, including the votes that could be cast by a replacement or test voter.

By publishing the reference table before the election, it can be assured that votes are counted for the proper candidate. Or in other words, by publishing the reference tables before the elections, it can be made clear that the link between cryptographic numbers and candidates are fixed before the elections. Technically, the reference table consists of two parts. One part, RDT.zip, contains all the links between candidates and numbers for each voter. The other part, RDS.zip, contains status bits, indicating whether a certain ballot is used for test, as a replacement or for a normal voter. RDT.zip can never be modified, but RDS.zip can be modified during the election. This will be indicated later on.

At the end of all this work by RIPOCS, all files are copied to PORTAL again by RIPOCS.

The file `WV-STUF-K11.xml` stays at PORTAL. It is used later by the helpdesk. The $\texttt{ReSPID}_i$s are sent to VotWin, because they are needed to keep track of the progress of voting by the

voters in the status table. The reference table is published by PORTAL, secured by hashes that are published in media so that they cannot be changed anymore.

The encrypted `WV-STUF-C10.xml` and unencrypted `WV-STUF-K50.xml` are sent to PSB. They will decrypt the file and use it to generate the ballot forms together with the envelopes and the addresses. The ballots for regular voters are distributed by mail. The replacement ballots are presumably delivered to the corresponding DWCB, since it is the helpdesk of each DWCB that has to use them. Finally, the `WV-STUF-C10.xml` file must be destroyed.

When all this is done the election can be started. Technically, the election could have been started automatically using the information in the `WV-STUF-B10.xml` file, but it has been decided that HWH will create an official start through a PORTAL application order to start the elections.

4.2.3 Vote collection – overview

The main tasks during this stage are:

- Opening and closing of the ballot box.
- Receiving and storing votes by Internet and mail.
- Operate the helpdesk: deal appropriately with people claiming that they didn't receive their ballot form.
- Generate reports of all actions taken.
- Publish the modified reference table together with its hash to protect it against further modification.

4.2.4 Vote collection – in detail: Casting a vote by Internet

In Figure 3 we show the flow of information that describes a vote being cast by Internet. The MSC starts with two basic items. First the voter $\texttt{Vn}_i$ should have received his ballot and second the election must have been started by PORTAL.

Now $\texttt{Vn}_i$ connects to the website, which is automatically set up as an SSL connection. Because the document [18] is not really up to date at this moment, we don't know exactly which screens are sent. However, by private communication we were told that some screens are sent where the voter does nothing besides reading the information and clicking 'next'. Therefore these screens are combined in the diagram with the first screen that really needs user input.

Screen A000 asks the voter to enter the election code `ElCd`, which he can find on his ballot. This `ElCd` is nothing but the `ElID`, but written in AN34 and with input validation characters added. The JavaScript program transforms `ElCd` into `ElID` and sends it to VotWin. Now VotWin checks whether this is a valid election id and if so, it returns screen A010 which is personalized based upon the indicated election.

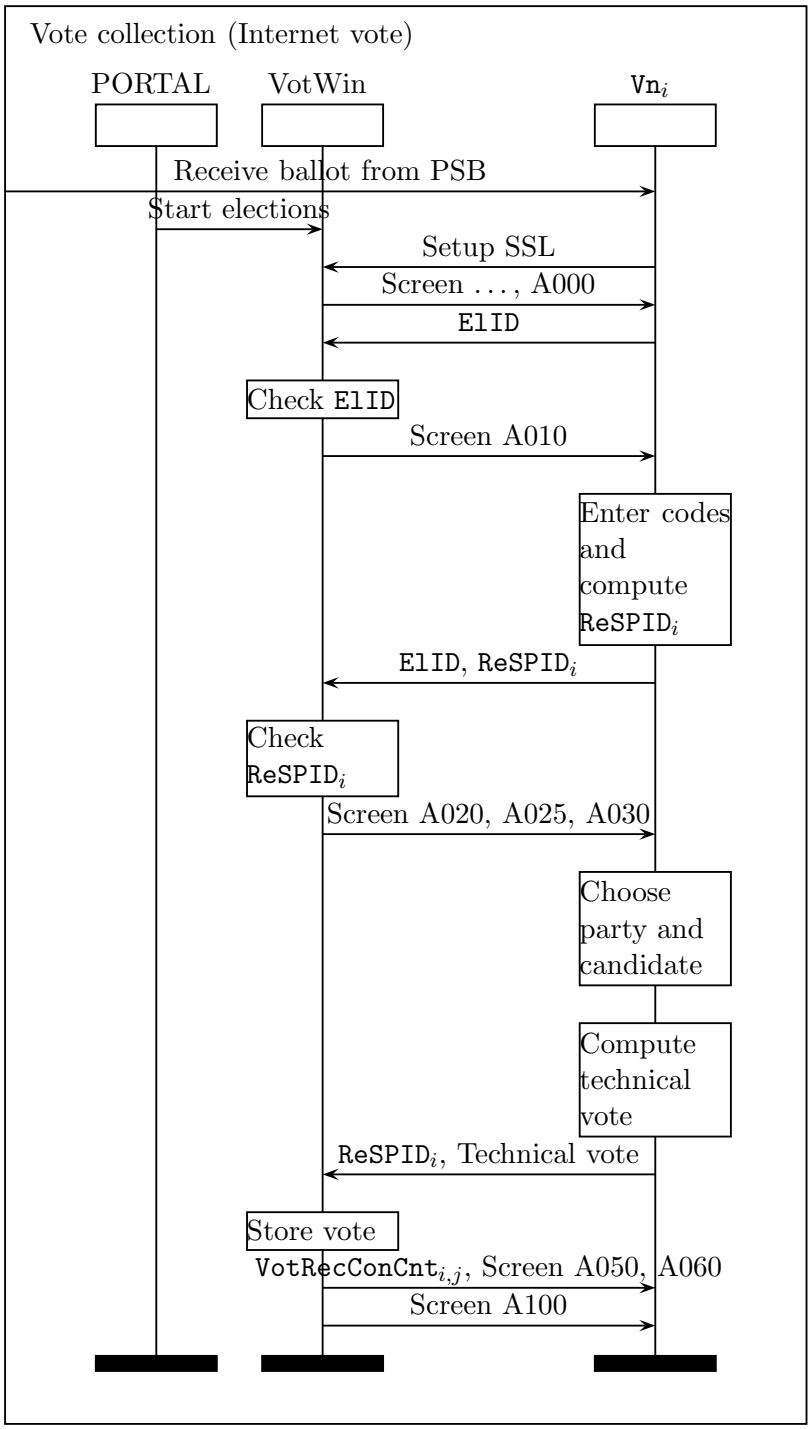

Figure 3: Vote collection if voted by Internet

Now the voter has to enter his so-called $\texttt{VPID}_{1,i}$, $\texttt{VPID}_{2,i}$ and $\texttt{AbelPI}_i$. The first two values are written on the ballot and are nothing but the voters personal key $\texttt{Kp}_i$, the last value is something that he should know. It is used to reduce the possible abuse of ballot forms. Currently this field is nothing but the last two digits of the voter's year of birth, which is obviously not a very strong code. It can be guessed with a reasonable chance of success. Both the $\texttt{VPID}_{1,i}$ and $\texttt{VPID}_{2,i}$ contain checksums and are checked locally by the client PC via JavaScript. If correct, they are used to compute $\texttt{ReSPID}_i$. At the end of this stage this $\texttt{ReSPID}_i$ is sent to the server, together with (again) the ElID. Note in particular that $\texttt{VPID}_{1,i}$ and $\texttt{VPID}_{2,i}$ are not sent over the Internet.

When the VotWin receives the ElID and the $\texttt{ReSPID}_i$, the system checks whether this is a valid combination. This is most likely done using the table 'status', which has a primary key consisting of these two values. Furthermore, this same table contains a field 'status' that indicates whether a voter has already or has not yet voted for a certain election. In case he hasn't voted the server will now present the voter with screen A020 that shows all the parties competing in this election, including an option for a blank vote. The voter simply chooses one and the server now presents screen A025, a list with all the candidates within this party. Although this sounds like a straightforward way to do it, there is something special that needs to be mentioned here. If the information that the server sends to the voter only contains the candidates of the party chosen by him, by looking at the length of this encrypted package, it could be possible to make an educated guess about his favorite party. Information leakages is prevented by sending all parties and candidates in one large message and using JavaScript on the client to make sure that the voter sees the appropriate screens A020 and A025. After selecting the candidate, screen A030 is presented which shows only the candidate that has been chosen previously. This screen also contains the final vote button.

Once the voter presses this button the JavaScript engine starts to compute the so-called technical vote. In particular this means that by using his secret $\texttt{Kp}_i$, the values $\texttt{VnPID}_i$ and $\texttt{VnCx}_{i,j}$ are computed. The first indicates the pseudo identity of the voter, the second indicates the pseudo identity of the candidate chosen by the voter. Note that we have seen these values before: RIPOCS has computed them as a necessary intermediate step to compute all hash values $\texttt{RnPID}_i$s and $\texttt{RnCx}_{i,j}$s, that were published in the reference table. Now that this technical vote is computed, it is sent to VotWin together with the voter's $\texttt{ReSPID}_i$. Note that all these messages between $\texttt{Vn}_i$ and VotWin are done via SSL and hence encrypted, furthermore this SSL session takes care of communication integrity and server authentication: the voter can check by clicking on the browser 'padlock' that he is communicating with a proper RIES server.

The server VotWin is going to store this technical vote. However, before it actually stores the vote it seems that it checks the $\texttt{ReSPID}_i$ to be sure that the vote corresponds to one from a legitimate voter. The technical vote itself is not checked. Now assume that the $\texttt{ReSPID}_i$ is valid. This technical vote is stored in two different ways. As was the case with the previous version RIES-KOA, the technical vote is appended to a simple text file on the server, typically in a file that used to be called 'ssXvotes.txt' where the X denotes the number of the server being used. However, in addition to this RIES-2008 also stores the

vote in a MySQL database referred to as 'Stembus'. And to be more precise, the server tries to insert the vote not only into one database, but in a cluster of databases, all on different physical locations. This is for safety reasons if something goes wrong with one of the databases. Only if the server receives the message that the insert has succeeded from two different databases, the server is going to mark this vote as being cast. In that situation, the server is going to update the status of the user who sent the vote (using the $\texttt{ReSPID}_i$ value). Furthermore it is going to compute a receipt that the voter can use to prove that he voted. This receipt is actually only half of the value that the server computes. Technically, the VotWin computes $\texttt{VotRecCon}_{i,j} = \texttt{DESmac}(\texttt{Kbbs\_0}, \texttt{VnPID}_i \| \texttt{VnCx}_{i,j})$. The key Kbbs_0 is stored in the crypto hardware inside the server. This $\texttt{VotRecCon}_{i,j}$ is an eight byte value, which is stored in the database, in the 'status' field of the 'status' table. Note that this table only contains one row for each $\texttt{ReSPID}_i$ and therefore if a voter decides to vote twice or more, only the last $\texttt{VotRecCon}_{i,j}$ will be stored in the database. However, this $\texttt{VotRecCon}_{i,j}$ is split in two four byte values, such that $\texttt{VotRecCon}_{i,j} = \texttt{VotRecConSvr}_{i,j} \| \texttt{VotRecConCnt}_{i,j}$. And only the last four bytes $\texttt{VotRecConCnt}_{i,j}$ are sent to the voter. After the election, the voter can use this value if his vote doesn't show up in the tally results. In order to do that he needs to go to the so-called UMPIRE, who is able to check whether the four bytes that the voter has are in fact part of a valid confirmation receipt or not. See Section 4.2.10 for more information about this process.

In the diagram we see that besides this receipt confirmation value, also the screens A050 and A060 are sent as one message. Screen A050 shows to the voter wether the vote was received or not. Furthermore, it contains a button that needs to be pressed in order to actually see the important values $\texttt{VotRecConCnt}_{i,j}$ and the technical vote as presented on screen A060. The values are important because they are needed to check one's vote in the final tally result. And they are also important because if someone gets to know the technical vote of a voter, he doesn't need any cryptographic key anymore to discover on which candidate the voter voted. This switch between pages is done via JavaScript. The value $\texttt{VotRecConCnt}_{i,j}$ is computed, stored and sent to the client by VotWin. In particular it is possible to ask the server to resend the receipt value in a later stage, based upon the voter's $\texttt{ReSPID}_i$. This will be done in case a voter repeats his voting process after the successful casting of his vote in an earlier stage. However, it is not possible to retrieve the technical vote from the server by sending the $\texttt{ReSPID}_i$. This would be an easy way for people to find out to which candidate a voter has cast his vote. That is the reason the browser uses JavaScript to keep the technical vote in memory. Note that via private communication we heard that there were ideas to present the $\texttt{VotRecConCnt}_{i,j}$ and the technical vote to the voter by means of a .pdf file, however, since this file is likely to be generated on the server, this would introduce a method for sending out technical votes by the server and that is something that should be prevented, so we advise against this idea.

The final screen A100 is simply a message in which the voter is being thanked for casting his vote.

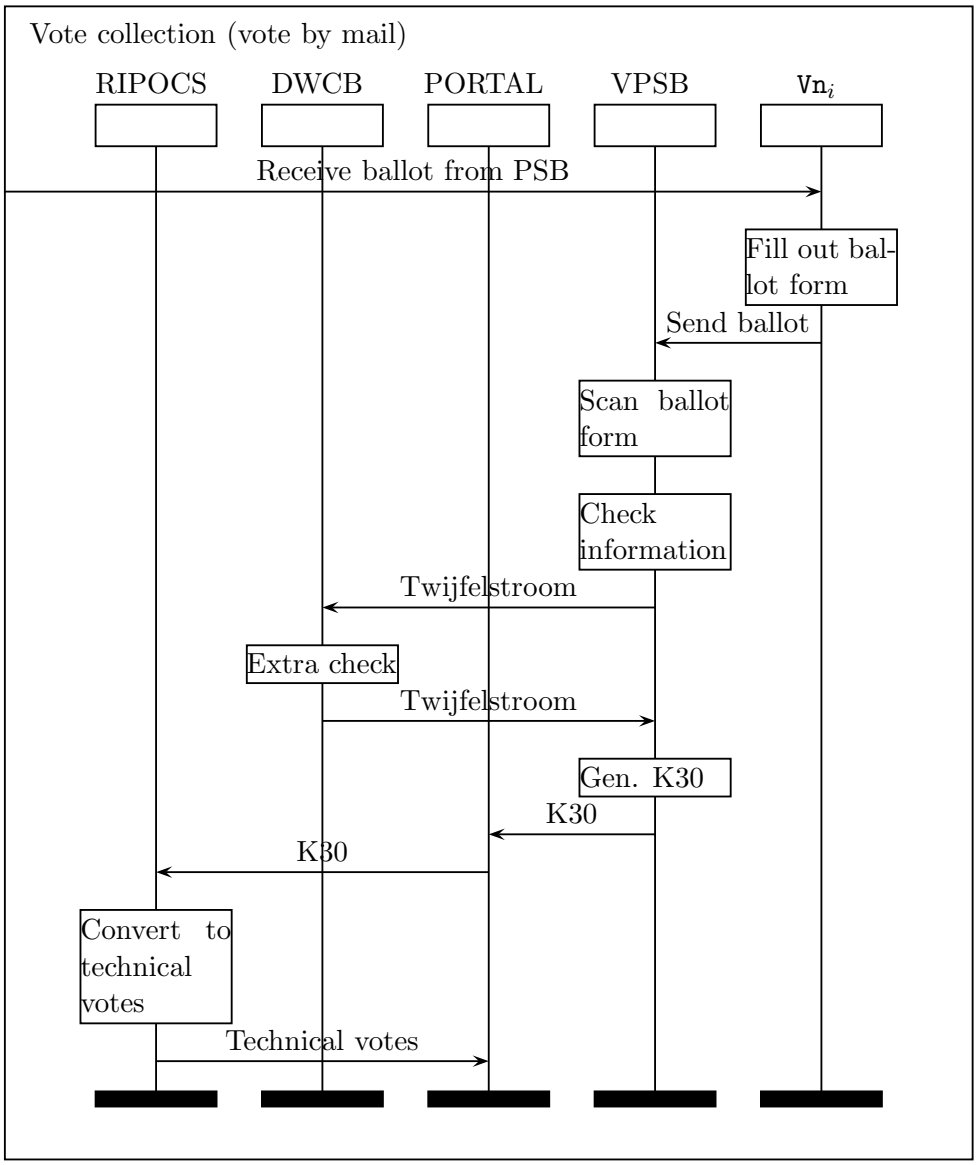

Figure 4: Vote collection if voted by mail

4.2.5 Vote collection – in detail: Casting a vote by mail

Figure 4 describes the second way of voting: sending the ballot forms by regular mail. Obviously, the main prerequisite is that voter $\texttt{Vn}_i$ has received his ballot form. Note that it is not needed that the election is officially started as with voting by Internet. The voter just needs to make sure that he sends his ballot in time; he can't send it too early.

Filling out the form requires three steps. First he has to write down the last two digits of his year of birth. This will result in the so-called $\texttt{AbelPI}_i$ information. Furthermore, he has to mark his favorite group, followed by the candidate number within that group. In case a voter only selects a group and no candidate, the first candidate of that group will receive the vote. The ballot forms do not show the names of the individual candidates on the ballot form. They only show generic names like 'Candidate 12'. This is done to allow for an A4 size postal ballot, in spite of the fact that the voter can choose out of over thirty groups, each with a maximum of fifteen candidates. To figure out who is 'Candidate 12' a separate list with the names and numbers is provided. This seems like a system that is likely to introduce mistakes.

The ballots are received by VPSB. They expect thousands of forms everyday and hence they will not wait till the election is closed, but will use batch processing to handle the ballots. They start by scanning the forms. The forms contain special OCR lines that somehow represent the pseudo identity of the voter. More precisely the OCR lines contain the $\texttt{Kp}_i$, but only encrypted by the 3DES key `Kkpocr`. In particular the decrypt version of this key `Kocrkp` is needed to actually compute $\texttt{Kp}_i$ itself. And even then one would probably need `Kgenvoterkey` as well to compute personal information like $\texttt{VnID}_i$, which is typically the BSN. These lines are for instance used to check whether the voter has made copies of his ballot and submits it more than once. Furthermore, these lines are also used to help VPSB with recognizing the handwritten $\texttt{AbelPI}_i$ on the ballot forms. Since it is difficult for character recognition software to recognize a text of only two characters, VPSB gets some help from the system. Using the so-called Abel-PI-Black-box, a module that contains a smartcard with the key `KabelPi` on it, the correct $\texttt{AbelPI}_i$ is computed for each postal vote that was received by VPSB. If the handwritten characters cannot be identified unambiguously, this real $\texttt{AbelPI}_i$ will be used for determining what was written. For instance if the normal software cannot decide whether a 1 or 7 is written by the voter, but in the real $\texttt{AbelPI}_i$ a 7 is written on the same place, then the software will decide that a 7 was probably written. Obviously it would have been easier to give VPSB a list of all $\texttt{AbelPI}_i$s, but using this method with Abel-PI-Black-box it is enforced that VPSB only gets to know the $\texttt{AbelPI}_i$ of ballots that have been received. In addition, note that VPSB only transforms the postal ballots into digital information and never marks votes as invalid, even though they can see that the written $\texttt{AbelPI}_i$ was wrong. Whether a vote is valid or not will be determined later during the tally process. See [25] for more details.

If a ballot cannot be handled automatically, a long chain of responsible persons will look at it, starting at VPSB, but eventually moving to the corresponding DWCB. And as soon as a DWCB is involved the ballot is part of the so-called 'twijfelstroom'.

With the information form a batch of scanned ballot forms a file `WV-STUF-K30.xml` is

created by VPSB. Each (complete) ballot gets transformed into a record containing the ballot box number, the OCR lines, the $\texttt{AbelPI}_i$ information, the selected party and the selected candidate and a reference to the stored image of the scan. Besides these records, each `WV-STUF-K30.xml` batch file also contains statistics about how many ballots are transformed in this batch and also for accumulated for all batches so far. VPSB sends this file to PORTAL.

RIPOCS checks whether a `WV-STUF-K30.xml` is available on PORTAL and if so copies it. After that it decrypts the information in each OCR line using the key `Kocrkp` which is only available on RIPOCS. After decrypting these lines, RIPOCS now knows both the $\texttt{Kp}_i$ and the $\texttt{AbelPI}_i$. Together with the choice made by the voter, RIPOCS now computes the technical vote that corresponds to this choice. This is the same technical vote that would have been sent to the server if the voter voted by Internet. The resulting list of all these technical votes is sent by RIPOCS to PORTAL again, where it can be used in the tally process.

Note that Figure 4 does not really indicate that many of the tasks are done in parallel. The receiving and scanning of ballots and hence also the transformation to technical votes is a repeating process that cannot be modeled properly in this type of diagram.

4.2.6 Vote collection – in detail: Helpdesk

The main task of the Helpdeks is providing information and assistance to voters on how to exercise their voting rights. Here we shall concentrate on the most security-sensitive aspect, namely providing replacement votes, on request.

As already described the file K11 contains both test and replacement forms. They are not related but handled in the same file (K11) because of similarities in shape. Both are different from ordinary vote forms: their UId's starts with 9 or 99. The number of these (test and replacement) forms is determined on the basis of information in the file K10 with ordinary voting forms.

Test forms are meant for monitoring the system by various parties involved (HWH, DWCB, TTPI and SURFnet, and possibly others). These forms can be followed throughout the system and process. They do not contribute to the outcome.

Replacement forms are sent to the Helpdesk, from where they can be sent to voters that complain that their original forms are missing. These replacement forms do contribute to the outcome and are therefor extremely sensitive: they allow everyone who holds them to add votes to the system, while disabling some ordinary vote form. Replacement votes are added to RIES for pragmatic reasons, in order to make the system more user friendly, at the expense of introducing vulnerabilities. These replacement vulnerabilities are accepted and covered via procedural measures, mainly at the Helpdesk.

During operation of RIES-KOA (for expatriates) a substantial number of replacement forms was sent, due to the poor reliability of international postal services. However, during regional District Water Control Board elections the number is limited. Replacement forms can be provided only once per voter. When all replacement forms have been handed out,

requests can no longer be answered: these forms are produced only during the initialization phase.

In Figure 5 it looks like the helpdesk (part of DWCB) sends out the new ballot before RIPOCS has modified the status bits of the reference table. This coincides with the description in [15]. Each mutation by the helpdesk is immediately sent to PORTAL, but RIPOCS only starts modifying the status bits in the reference tables when the election is closed. However, there is something strange here. RIPOCS does send a status report to PORTAL after it tried to handle all mutations, but it is not clear what can be done with this if the result parameter in this report is ERROR. What happens if RIPOCS fails to modify the status bits but the ballot is already sent? Since both the helpdesktool and RIPOCS use the same `WV-STUF-K10.xml` and `WV-STUF-K11.xml` files this should not happen. Furthermore, the fact that actually three RIPOCS machines are computing these changes, reduces the chance that this ERROR value will be returned. But nevertheless, if it does happen, there is a serious problem with the outcome of the elections since some votes might be counted that were invalid and some valid votes might not be counted.

Now let's have a look at how it works. Each DWCB has its own helpdesk, that voters should contact (by phone) if they didn't receive a ballot form or if they received a damaged form or envelope. According to [15] this is what the helpdesk should do.

When replacement forms are requested the Helpdesk authenticates the requester by asking for personal identity or DWCB-tax related data. Unfortunately, most of these data are more or less public knowledge. With the answers to these questions, the Helpdesk uses its support software (helpdesktool) to link the claimed identity to an eligible $\texttt{Vn}_i$ and to the corresponding identity $\texttt{VnID}_i$, see [18]. The tool will mark the status bits for $\texttt{VnID}_i$ as 'withdrawn' in the reference table, or more precisely, in the second part of the reference table RDS.zip where status information is kept.

Furthermore it chooses one of the replacement ballot forms and sends it to the complaining voter. This form is identified by the so-called $\texttt{VrID}_{i'}$. This $\texttt{VrID}_{i'}$ is now reported as 'issued' to PORTAL and subsequently collected by RIPOCS. This ends the work of the helpdesk.

Of course, this can only be done during the voting phase, via the PORTAL.

In principle the number of 'withdrawn' and 'issued' status bits must coincide. However, in practice the number of 'issued' bits may be higher: someone may complain whose name is not in the list of eligible voters. This omission may turn out to be a mistake, after contacting the responsible citizen administration authorities. In that case new voting forms are issued. The number of replacements is published after the elections, so that the scale of these changes becomes known.

A paper log of each replacement step is kept in the Helpdesk, which is signed by the individual Helpdesk member that handles it. This log should not contain any link between the identity ($\texttt{Vn}_i$ / $\texttt{VnID}_i$) and the identifier $\texttt{VrID}_{i'}$ of the replacement form.

In [15] there is an explicit sentence that we would like to quote here:

> Note is taken that the combination $\texttt{VnID}_i$ and $\texttt{VrID}_{i'}$ is NOT recorded in any way (e.g. this can be handled by two different, separated elements of the helpdesk).

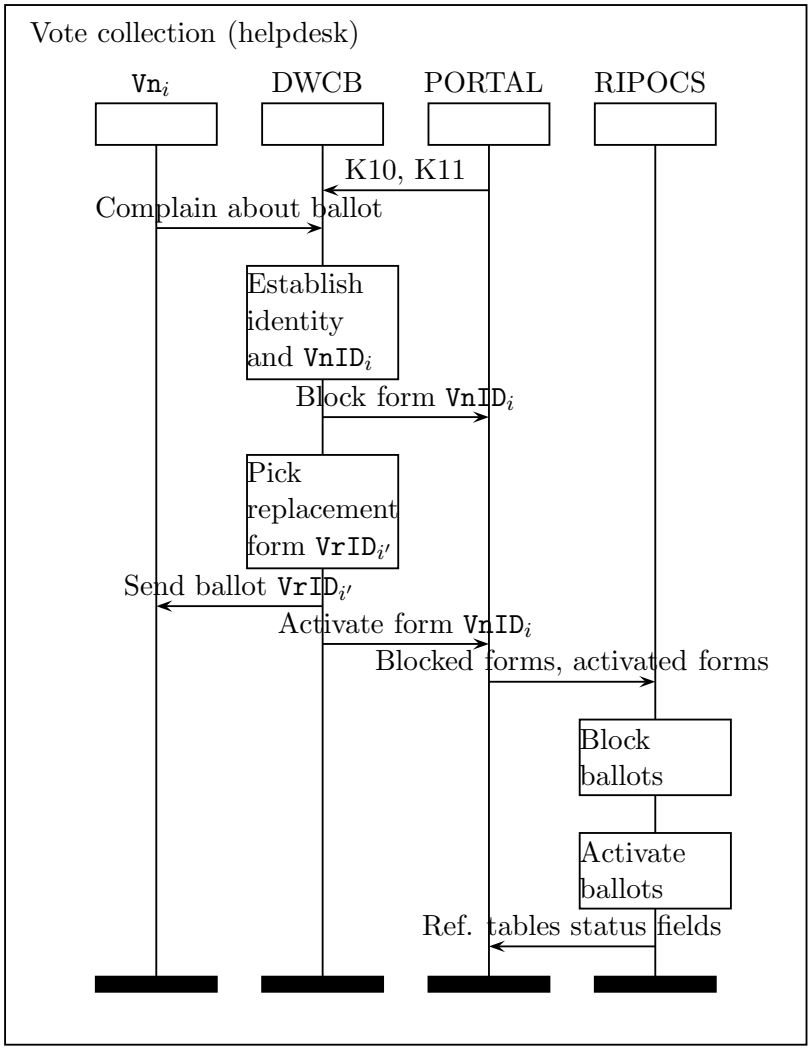

Figure 5: Procedure for replacing ballot forms

This is to ensure voter secrecy. Furthermore, any personal information in relation to the mailed $\texttt{VrID}_{i'}$ is not needed, since any voter can only receive one replacement package.

4.2.7 Vote collection – in detail: Monitoring

HWH is the central organization that keeps track of the whole election for all 26 DWCBs. Main source for HWH is PORTAL, which provides several systems of monitoring the elections in use on PORTAL. On its turn PORTAL typically gets its information from the collection of all VotWins. The communication for this is described in [5] as XML/RPC calls for control messages and `scp` for file transport over the VPN protected RIES-control network, always originating from PORTAL. Each VotWin is configured so that connections to this specific URL are only allowed from PORTAL.

The first system uses so-called PortalMessages. They are used to give feedback to operators of functions that are performed. A PortalMessage is sent by PORTAL to specific operators, who can only read them when they log on to PORTAL. A second system uses SMS. Main difference is that these messages also reach their target when the operators are not logged in to the system. For each operator a phone number is stored in the system.

A third system of monitoring is the idea of 'current tally'. While the election phase is still running it is possible to perform a tally process on the votes already cast. Because it is not legal to see a partial election result during the election this report will be stored encrypted. Its use lies in the fact that after the election has been closed it can be decrypted. And, if a series of these current tallies are available, it might be possible to detect where something has gone wrong.

Encryption of the confidential data is done by a randomly generated 3DES encipher key `Kpretal`, which is only exportable under a public key `PKnotpretal` with the proper certificate. For practical and performance reasons PORTAL doesn't use crypto hardware, but computes everything in software. This software has been reviewed by an independent, but for us unknown, company in the Netherlands. So we have to trust that the software used here only keeps the clear pre-tally result in memory.

One of the issues here is how PORTAL performs this current tally job. First it gets the current votes from VotWin. This is a critical part of the task. As we have seen before, typical calls from PORTAL to VotWin are done via XML/RPC or `scp` over the protected RIES-control network.

Once the votes are available on PORTAL, the current tally job is basically the same as the final tally job, which will be described later. Besides the encrypted report it also creates an unencrypted report with status information like the number of votes cast.

A fourth system for monitoring the elections is again based upon voting statistics. PORTAL periodically sends requests to each VotWin to get a list of the numbers of votes cast for each election. This information is used by PORTAL to generate graphics.

4.2.8 Tallying – overview

The following tasks are handled during the tallying stage.

- The votes are being counted by the system.
- The election authority publishes the provisional results.
- The election authority deals with all incoming objections against the provisional result. This may include advice by the so-called umpire.
- Eventually the final results are published.

4.2.9 Tallying – in detail: tally process

In Figure 6 we have described how the tally process works. Formally DWCB is in control. But since the election is normally stopped by the pre-installed time-value from `WV-STUF-B10.xml`, the last batch of postal ballots is sent by VPSB and the last helpdesk mutations are received, there is no actual action to be taken by DWCB to start the tally process and normally that would be best. But there might be unexpected situations to postpone the start of tally, which in general would not be desirable. Therefore HWH has to perform a final OK action to start the tally. This is in line with HWHs task to overview the entire organization of all DWCB elections.

The tallying phase officially starts when the election is ended. The elections are ended automatically by PORTAL, based upon the end date, time and delay that was defined in the configuration file `WV-STUF-B10.xml`. It signals VotWin to close the ballot boxes. Next step is that PORTAL retrieves all the technical votes from the VotWins.

Before the actual counting can start the last batch of postal votes received by VPSB needs to be processed. As we have seen in Figure 4 VPSB doesn't wait with processing all the forms to the moment that the election is closed. This is scheduled as a periodical batch process and hence only the last batch needs to be processed. The scanning part is done by VPSB, but via PORTAL the `WV-STUF-K30.xml` file gets to RIPOCS who does the actual conversion to technical votes and sends the result back to PORTAL.

In a similar way, also the last mutations of the helpdesk need to be taken care of before the counting can start. In particular this also requires action by RIPOCS to block or activate certain ballot forms by changing their status in the reference table. Since these status fields are being used during the counting process this is obviously something that has to be done before the actual counting can start.

At this stage PORTAL has all the technical votes: both the ones that come from postal votes as well as the ones that come from Internet. Since the format of both type of votes is the same they can be easily merged. Important is that they get a so-called priority label, indicating whether they were cast by post or by Internet. This is needed to deal with the fact that voters can try to vote twice by sending in an Internet vote and a postal vote.

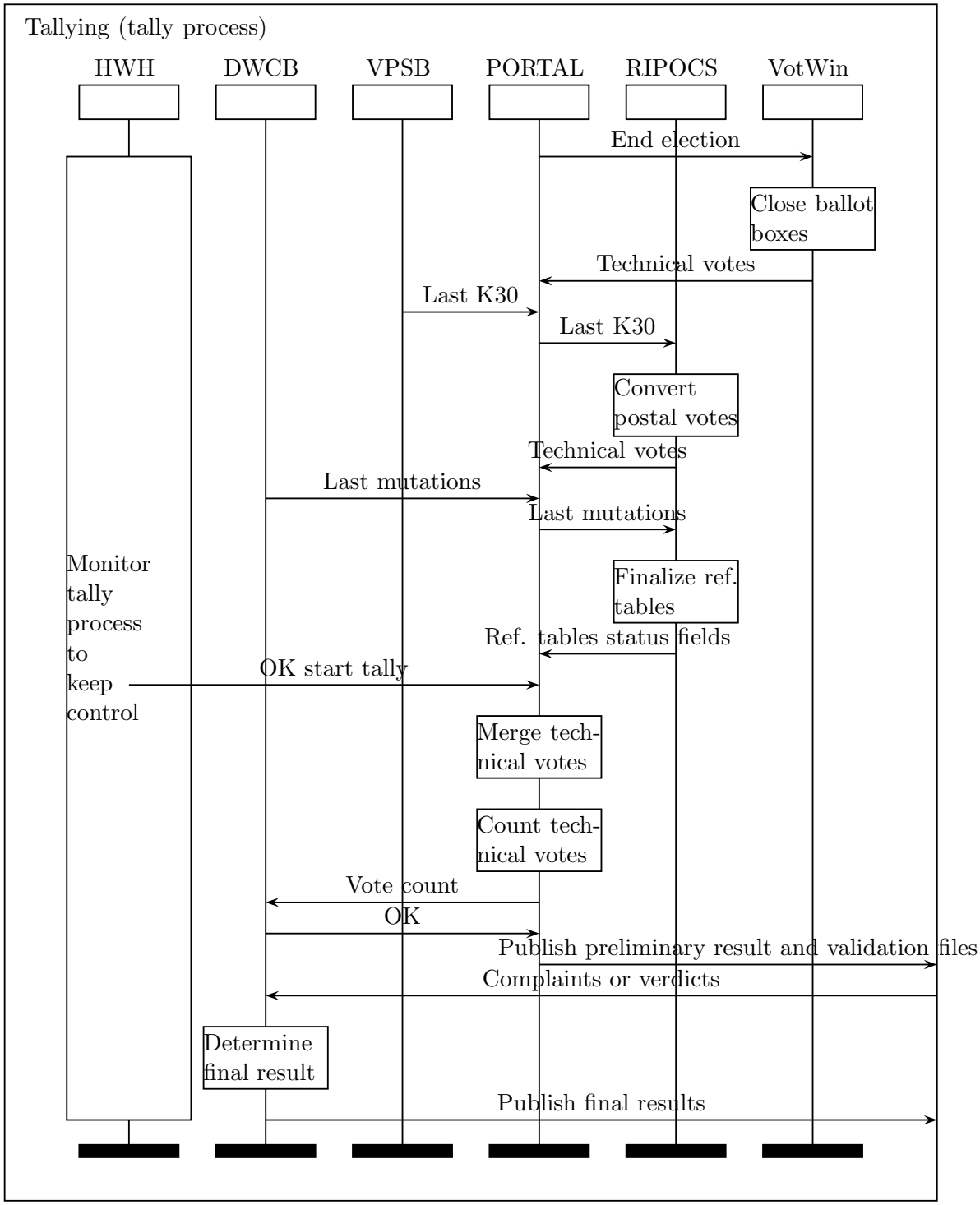

Figure 6: Determining the outcome of the elections

Based upon the priority the system is able to determine which of these votes should be counted in the tally.

The actual counting process is basically easy: for each technical vote the $\texttt{RnPID}_i$ and $\texttt{RnCx}_{i,j}$ values are computed using the key-less hash `MDC-2`. These are the values that are in the reference table and form the link between a vote and actual candidate. The result of this action is sorted on $\texttt{RnPID}_i$, priority and $\texttt{RnCx}_{i,j}$. Now for each cluster of equal $\texttt{RnPID}_i$s the following steps are done.

- If the $\texttt{RnPID}_i$ is not valid, i.e. it is not in the reference table, all votes in this cluster are marked invalid with a reference to this reason.
- Otherwise the status of this $\texttt{RnPID}_i$ is checked. If one of the three status bits indicate that the vote should not be counted, again all votes in this cluster are marked as invalid for the appropriate reason.
- Now that the voter is considered valid, each vote is checked. Note that they are ordered by priority from high to low. The first check is on the validity of $\texttt{RnCx}_{i,j}$. If it is not valid this vote is marked invalid.
- If it is valid, it is checked whether there are more votes within the same priority. If this is not the case, all other votes are marked invalid because of this.
- If there are other votes with the same priority, the $\texttt{RnCx}_{i,j}$ in the next vote is checked. If it is invalid then this next vote will be marked as invalid. If it is valid then it is checked whether this next $\texttt{RnCx}_{i,j}$ is equal to the current $\texttt{RnCx}_{i,j}$. If it is, the next vote is marked invalid for double vote, but the current vote is still considered valid. If it is not, both the current vote and this next vote are marked as invalid for the reason of contrary votes within priority.
- If a vote is valid, the corresponding candidate as defined in the reference table will get one more vote.

Eventually each vote in the list of technical votes is either marked as counted or as not counted. If it is counted, it is indicated for which candidate the vote is counted. If it is not counted, the reason why not is listed. This file is referred to as the 'performed votes' file.

Now the preliminary vote count is handed over to DWCB, who should formally accept this preliminary result before it can be published. Later on DWCB has to decide whether this will be the final outcome or not. Before they can do that, they wait for a short time frame to see whether some complaints from voters or verdicts from UMPIRE (see Section 4.2.10) are filed. Obviously this means that voters should have the chance to check the outcome of the elections. In order to support this, a lot of files are published on the Internet:

- The updated reference tables,
- A report with all mutations with respect to the status changes in the reference table,

- A list of the plain technical votes received by Internet,
- The 'performed votes' file as described above.
- A list of receipt confirmation values. This list contains for each vote the $\texttt{ReSPID}_i$, the ElID and the $\texttt{VotRecConSvr}_{i,j}$, which is the first half of the receipt confirmation value $\texttt{VotRecConCnt}_{i,j}$. In particular this is not the same value as $\texttt{VotRecConCnt}_{i,j}$, which was sent to the voter and should only be known to the voter.

If no objections are filed in time the DWCB will publish the final results.

4.2.10 Tallying – in detail: UMPIRE

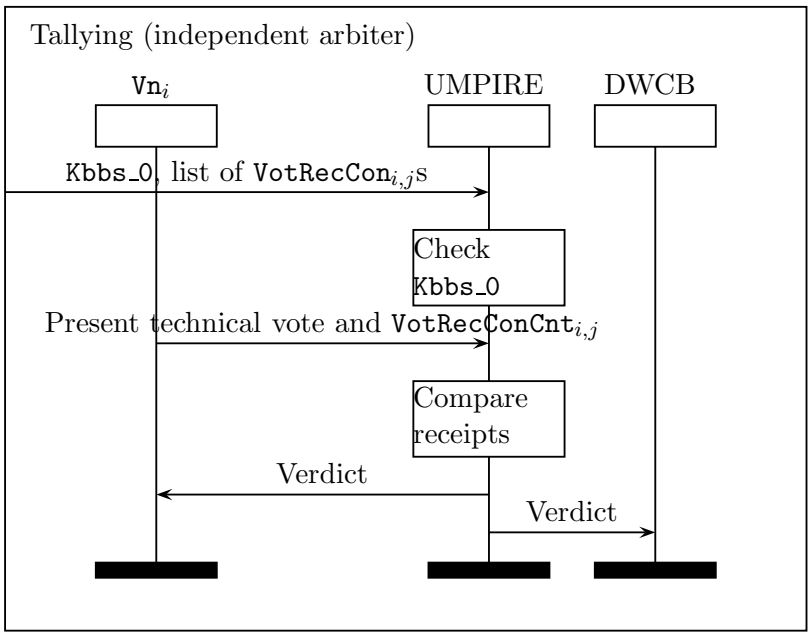

Figure 7: Role of the UMPIRE

Part of the official tally process is that voters can check whether their vote shows up properly in the list of votes after the election. This scenario is depicted in Figure 7. If it doesn't, they can use their receipt confirmation to complain about this at an independent arbiter, typically called UMPIRE.

Before UMPIRE can perform his tasks with respect to voters who complain. he first needs to do some checks on the material that he has received. For instance, UMPIRE gets a machine with crypto hardware that contains the Kbbs_0 key, which is supposed to be used by the VotWins to compute the $\texttt{VotRecCon}_{i,j}$. According to [16] this is a machine that is also called ROCMIS, but contains a crypto card that has been taken out of one of the

VotWins. It is not really clear whether he is supposed to write his own software to compute all the receipts or whether the system comes with software that deals with this. For an independent role the first option would probably be better, but for practical reasons the second option is more likely. Now he needs to be sure that this key has indeed been used by those servers. Besides this key UMPIRE only has a list of all the technical votes and all published $\texttt{VotRecConSvr}_{i,j}$ values sent through Internet since these lists are publicly available.

Now the first check that he should perform is to see whether the list of published receipt values corresponds with the technical votes received. For instance, the number of receipts should be less than the number of technical votes. This is due to the fact that only the receipt that belongs to the technical vote submitted last is stored and hence only this last one shows up in the list whereas all submitted technical votes show up. Furthermore, he should compute the $\texttt{VotRecCon}_{i,j}$ of all technical votes. The first half of these $\texttt{VotRecCon}_{i,j}$s should either show up in the list that was published or belong to a voter who submitted more than one vote. In the latter case this should make up for the difference in length of the lists assuming that double appearances of technical votes are treated properly. If this check is passed then UMPIRE knows that the key he has in his hardware really forms the link between the technical votes and the list of receipts.

However, in order to be sure that this key has actually been used during the election and not only after the election to generate this list, UMPIRE needs to check some votes of people that aren't complaining. Did those people in fact have a $\texttt{VotRecConCnt}_{i,j}$ that is indeed the second half of the $\texttt{VotRecCon}_{i,j}$ that UMPIRE computes for them? If so it means that `Kbbs_0` was indeed used during the elections.

Now if a voter complains that his technical vote doesn't show up in the list of votes, this voter should present both his technical vote and his receipt confirmation to UMPIRE. Note that such a complaining voter can come up with basically any technical vote he wants, since they are publicly available in the reference tables. However, since only a list of $\texttt{VotRecConSvr}_{i,j}$s is published, he will not be able to derive a valid $\texttt{VotRecConCnt}_{i,j}$ from this information. Furthermore, the UMPIRE already knows that if the receipt came from the server, he can simply reproduce the $\texttt{VotRecCon}_{i,j}$ that the server would have computed for this technical vote. Now if the receipt confirmation of the voter doesn't coincide with the second half of the confirmation value that UMPIRE computes, the latter one knows that the $\texttt{VotRecConCnt}_{i,j}$ that the voter presented to him cannot have been generated by the vote server and hence the verdict will be that the voter has no valid complaint. If the voter's $\texttt{VotRecConCnt}_{i,j}$ does coincide then the voter has a valid complaint and the outcome of the elections cannot be trusted anymore.

4.2.11 Finalization

After closing the elections and publishing the preliminary result some things need to be done. Here we list the ones that we consider important.

- Vote and tally verification by a voter or an independent party:

If a voter did not store his technical vote somewhere he will not be able to check that his vote was counted correctly. If he did store it, the procedure supported by RIES says the following.

- Download the post-election table, listing all the received votes.
- Search for his $\texttt{VnPID}_i$ and $\texttt{VnCx}_{i,j}$ in this file.
- On the same line both $\texttt{RnPID}_i$ and $\texttt{RnCx}_{i,j}$ are listed.
- Furthermore, the candidate number and the status of the vote are given.
- A voter sees directly whether his vote was counted or not.

If the voter doesn't find his vote, he can go to UMPIRE with his receipt $\texttt{VotRecConCnt}_{i,j}$. If his vote is there, but given to some other candidate he should complain at the DWCB. See Section 4.2.10 for details.

Besides the voter also independent parties can verify the outcome of the election. They should do the following.

- Download all received technical votes.
- For each technical vote, compute the `MDC-2` of both parts.
- Search the pre-election reference table for these values.
- Each combination should lead to exactly one vote on a specific candidate.
- Special attention should be given to the rules that need to be followed when double votes occur.

- The procedure for voter objections:

  The procedure for complaining when a voter cannot find his vote in the list of collected votes is already described in Section 4.2.10. However, the procedure for complaining if a voter finds his vote in the list, but sees that the vote is counted for a wrong candidate or sees that the vote is marked invalid for some reason is not completely described. The only thing written is that the voter should contact the DWCB as soon as possible and that its helpdesk will support the voter in filing an official complaint. It is not clear what kind of evidence the voter should provide in order to support his claim besides the technical vote and the confirmation receipt. However, due to the nature of the system, it is not always possible to prove that such a voter's claim is correct or incorrect. In such a situation the DWCB decides what to do.

- The destruction of sensitive material:

  Unfortunately the documentation doesn't seem to describe what happens with sensitive information after the tally process. Only for PSB it is marked explicitly that the `WV-STUF-C10.xml` file needs to be destroyed. For other actors it is not clear how they archive their material.

  With respect to the key management it is claimed that ROCMIS, RIPOCS and VotWin all work in application stages, where one of the last stages takes care of

erasing all keys. For ROCMIS and RIPOCS it is indeed indicated in [16] that they have a requirement to erase essential keys after completion of election tasks. For VotWin this is not indicated exactly, but it is mentioned that it has at least functionality to erase keys. However, it is not indicated how strict this 'after completion of election tasks' is implemented. Is it within a minute after all is done? Or within a week or month? For the keys like `Kgenvoterkey` that are no longer needed once the election is closed, they could be destroyed immediately after closing the elections. For the keys like `Kbbs_0` that are needed by independent parties for research it seems logical that the keys are destroyed when those parties decide that they have finished their work.

4.3 Components

4.3.1 Voter PC and Internet Connection

RIES is designed in such a way that voters should be able to use their own PC with standard browser to express their votes. The design document [18] says that 99% of the voters should be able to cast their vote without adaptation of their PC. This percentage is formulated on the basis of earlier helpdesk experiences and logs.

The (cryptographic) operations on the client side are done within the voter's browser using the scripting language JavaScript. Since any voter can inspect the source of the webpages, the client side software is *de facto* open source. It includes, in particular, an implementation of `DES` in JavaScript, taken from an open library[5].

The following features are assumed about the client's browser.

- JavaScript must be enabled; if not, the user is directed to a special page where the options are explained: switch on JavaScript, use another computer, or vote with the postal ballot,
- cookies are not used,
- SSL must be supported, with very specific version requirements that are comparable to Internet banking,
- screen resolution of 800×600, minimally.

The webpages of the RIES system have been developed following common webstandards, not for a specific browsers. Pages have been tested for Internet Explorer (versions 5, 6, 7), Firefox, Mozilla Suite, Safari, Opera. It is reported to have been used also on PDAs, even though not officially supported.

Interfacing is considered by some to be the most important issue in e-voting, in order to avoid a bias in the presentation of the options towards certain parties or candidates.

[5] Namely `http://tero.co.uk/des/` of Paul Tero, July 2001.

Independent consultation—on a previous candidate-based version, and not on the most recent party-based one—has contributed to the usability of the webpages of RIES (also for the disabled). The focus has been put mostly on clarity and unambiguity, and not so much on lack of bias.

Some voters may go directly to the webpage of their preferred candidate and thus reveal their choice to the server. Such a possible leak of information has been addressed in RIES by sending all candidates in bundled form to the voter's machine. In practice this involves a modest amount of information (only text).

Possible malicious software on the voter's side is not addressed in RIES. It is left to the individual voter to be able to use an appropriately protected machine.

Bandwidth is not an issue: the webpages that have been used so far contain only a modest amount of graphics. They work well even on a slow (modem or mobile) connection.

4.3.2 RIES System Architecture

Here we'll give a short description of the system architecture used in several parts of RIES.

PORTAL

The PORTAL is the web service that facilitates the workflow of an entire election process. Typically there is only one PORTAL machine available, however, a stand in machine is available if the original PORTAL goes down. Since PORTAL is not a machine that has a high load during the election process it doesn't seem necessary to mirror it. It uses an Apache Tomcat 5.5 servlet container. Furthermore it uses a MySQL database. Authentication to this server is done by SMS, using a HTTPS connection to a telcom provider. Communication with VotWin is done via XML-RPC or `scp` over the RIES-control network. Communication with RIPOCS is done via SCP, initiated by RIPOCS. Also important is the file system. Within the normal local file system, a special area is called the Election File Workspace. This directory is used for all the critical files needed for the system. Technically, it is divided in subtrees. For instance each DWCB has its own subspace. Furthermore, a special area within this Election File Workspace is dedicated for file transmission between PORTAL and RIPOCS. This workspace is protected by the normal user access control systems that the operating system provides.

Private Network

All servers are connected via the SURFnet 6 backbone. The different VotWins are located in Tilburg, Utrecht, Amsterdam and Nijmegen. Besides the fact that the backbone used is one of the fastest and most trustworthy networks currently available, we also need to mention here that the network is completely glass fibre and if there is one network problem, traffic is automatically redirected and hence all servers will remain reachable. In fact, before no one of the four servers is available, there need to be at least eight severe network problems at

the same time. So availability of at least one server should not be a problem. Furthermore, four servers are used to make sure that also during peak hours the voters will not notice slow response times.

Besides the physical network that we have described above, we should also say something about the way the network is used. As we have seen there are four locations for VotWins. Furthermore, there is also one location for PORTAL, which might coincide with one of the previously mentioned locations. In addition some laptops are available for doing remote management on the servers. Communication between different locations always goes over a VPN dedicated to the RIES system. However, since both VotWin and PORTAL are not just one machine, also communication within the locations is needed. This communication always takes place over a private LAN. For instance PORTAL needs to communicate with three RIPOCS systems. And each VotWin needs to communicate between the so-called front-end server that is used for presenting static webpages and the back-end server that is used for dynamical content and for instance also contains the database. Each VotWin has one designated database-server ('Stembus') to deliver the casted vote. This database-server handles the storage of the casted vote locally and on all (but at least one) database-servers on the other locations. Communication between VotWin and the designated database-server is, under normal conditions, over the private LAN and communication between the database-servers is over the dedicated RIES-VPN.

For communication typically XML/RPC calls over HTTP are used for control messages. Because of the VPN protection between locations and the private network within a location, it was an explicit choice not to use the slower HTTPS. If files need to be transported from one machine to another, this is always done using secure copy over SSH, usually referred to as `scp`.

The communication between VotWin and a voter client is restricted to the casting of the technical vote by the client and the return of the receipt to the client. In all other parts of the protocol there is no dependency of the client on a specific VotWin. In case of service interruptions on one VotWin, the casting of votes will continue almost uninterrupted to each client by the other VotWins. In case of doubt on the side of the voter about the successful completion of casting his vote, he can just repeat his voting action. He either receives the status of his former vote casting (as voter status is synchronized among all VotWins) or can complete his cast the second time.

Cryptographic Hardware

In order to prevent that one organization has all the important keys, RIES-2008 uses hardware crypto for doing the critical computations. The hardware used is of the type IBM 4764 PCI-X, which is FIPS PUB 140-2 Level 4 certified by NIST [23]. See [16] for more details on this device. One of the important features is that for each key it can be explicitly specified whether it can be exported or not. However, because some of the keys need to be available at several places, it needs to be possible to clone a complete hardware crypto device. This is possible with the IBM 4764. The standard cloning procedures from IBM's *Common Cryptographic Architecture* [8], [7] are used. Typically all three RIPOCS

machines have a crypto card. Furthermore each of the four VotWins has such a crypto card. Also the UMPIRE has a machine that is equipped with a crypto card, but PORTAL does all its crypto in software. Because of the fact that once the cloning session is ended it is no longer possible to clone a card again, some spare cards are created and stored in a safe as backup for serious hardware failure.

PSB - Printing Service Bureau

The PSB is a critical node in the whole RIES setup where sensitive data (`WV-STUF-C10.xml`) are handled in the clear, namely in order to print name and address information, together with personal keys. With this information one can not only vote on behalve of someone else, but also, if copied, check later who voted what. The PSB receives `WV-STUF-C10.xml` in encrypted form, via its public key `PKpsbc10`, from PORTAL. After finishing its printing job, destruction of `WV-STUF-C10.xml` is required. If and how such destruction is actually supervised is not clear.

In RIES-2008 the PSB is a foreign (non-Dutch) commercial company.

VPSB - Postal Votes Processing Bureau

Processing postal votes to convert them into technical votes that can be tallied by the RIES tallying process is outsourced to a company.

Backup / redundancy

As stated before there are four redundant VotWins and three redundant RIPOCS systems all online. For PORTAL there are two redundant machines, but only one of them is online. The second one is configured as hot-standby and can be made active when needed. Due to the sensitive nature of the data on the servers no backups are made. In case of the loss of a server it will be rebuilt from scratch using a provisioning server. Data loss is prevented by several mechanisms, specifically using redundant hardware and synchronization between locations.

4.4 The cryptographic protocol

The one-time signature scheme used by RIES is described as follows. Rather than generating the key pairs for each voter independently, which usually means that the private keys are generated uniformly at random and independent of each other, all key pairs are generated from a master key, called `Kgenvoterkey`. The individual key pairs are derived from `Kgenvoterkey` as diversified keys using additional inputs such as a voter ID etc. We model the presence of the master key by an additional set-up algorithm in the one-time signature scheme, to stress that all keys are actually related.

The one-time signature scheme thus becomes:

- **Set up.** A probabilistic algorithm that generates a 112-bit 3DES-key, called $\texttt{Kgenvoterkey}$.
- **Key generation.** A deterministic algorithm that on input of the master key $\texttt{Kgenvoterkey}$, a list of messages $\{\texttt{Cm}_j\}_j$, and a diversifier consisting of a voter ID $\texttt{VnID}_i$, an additional $\texttt{AbelPI}_i$, a voter group $\texttt{ParGp}$, and an election ID $\texttt{ElID}$, outputs a unique private key $\texttt{Kp}_i$, defined as

  $$sk := \texttt{Kp}_i = (\texttt{VPID}_{1,i}, \texttt{VPID}_{2,i}) = \texttt{3DES}(\texttt{Kgenvoterkey}, \texttt{VnID}_i \| \texttt{ElID} \| \texttt{ParGp}),$$

  and a corresponding public key defined as

  $$pk := (\texttt{RnPID}_i, \{\texttt{RnCx}_{i,j}\}_j),$$

  where

  $$\begin{aligned} \texttt{RnPID}_i &= \texttt{MDC-2}(\texttt{DESmac}(\texttt{Kp}_i, f(\texttt{BalBxID}))), \\ \texttt{RnCx}_{i,j} &= \texttt{MDC-2}(\texttt{DESmac}(\texttt{Kp}_i, f(\texttt{AbelPI}_i, \texttt{Cm}_j))). \end{aligned}$$

  Here f is a simple padding function.
- **Signature generation.** A deterministic algorithm that on input of a message $\texttt{Cm}_j$ and a private key $sk = \texttt{Kp}_i$, outputs as signature the pair

  $$s := (\texttt{VnPID}_i, \texttt{VnCx}_{i,j}),$$

  where

  $$\begin{aligned} \texttt{VnPID}_i &= \texttt{DESmac}(\texttt{Kp}_i, f(\texttt{BalBxID})), \\ \texttt{VnCx}_{i,j} &= \texttt{DESmac}(\texttt{Kp}_i, f(\texttt{AbelPI}_i, \texttt{Cm}_j)). \end{aligned}$$

- **Signature verification.** A deterministic algorithm that on input of a message $\texttt{Cm}_j$, a public key $pk = (\texttt{RnPID}_i, \{\texttt{RnCx}_{i,j}\}_j)$, and a purported signature $s = (a, b)$, determines whether s is a valid signature on $\texttt{Cm}_j$ w.r.t. public key pk by checking that

  $$\begin{aligned} \texttt{MDC-2}(a) &= \texttt{RnPID}_i, \text{ and} \\ \texttt{MDC-2}(b) &= \texttt{RnCx}_{i,j} \end{aligned}$$

  holds.

Given one signature for a key pair, it is not feasible to find a second signature (for a different message) for the same key pair. The first part of each signature, however, is equal, hence breaking the signature scheme amounts to inverting exactly one application of the $\texttt{MDC-2}$ function.

An election is run as follows. The set-up algorithm is run to generate the master key $\texttt{Kgenvoterkey}$. This is the only probabilistic part of the election protocols. Subsequently,

the key generation algorithm is run centrally, for each voter (plus some additional ones). All the secret keys and public keys are written in two files, respectively. The file with public keys is made publicly known; the file with secret keys is used to produce the printed ballot papers. There is a built-in link between the voter's identities $\texttt{VnID}_i$ and voter's name and addresses $\texttt{Vn}_i$, as $\texttt{VnID}_i$ is derived from the voter's BSN, hence linked to the real-life identity of a voter. The printed ballot paper containing $(\texttt{VPID}_{1,i}, \texttt{VPID}_{2,i})$ will be sent to $\texttt{Vn}_i$.

A voter will use $(\texttt{VPID}_{1,i}, \texttt{VPID}_{2,i})$ to compute a valid vote $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ for the chosen candidate $\texttt{Cm}_j$. The signature $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ is stored by the voting server. Note that in principle there is no reason why a signature should consist of two pre-images rather than just one. That is, normally the component $\texttt{VnPID}_i$ should be redundant in the presence of $\texttt{VnCx}_{i,j}$ (and in the presence of $\texttt{ReSPID}_i$, see below). However, the cryptographic mechanisms as employed by RIES necessitate the use of two pre-images (as the pre-images are only 64 bits each; this is much worse than a single pre-image of 128 bits).

Upon successful completion of the voting protocol, a voter also gets a receipt. The receipt cannot be verified by the voter though. The voter needs to store or record the receipt for later use, if desired, when checking the election results. A receipt is actually a MAC of the following form:

$$\texttt{VotRecCon}_{i,j} = (\texttt{VotRecConSvr}_{i,j}, \texttt{VotRecConCnt}_{i,j}) = \texttt{DESmac}(\texttt{Kbbs\_0}, \texttt{VnPID}_i \| \texttt{VnCx}_{i,j}),$$

where Kbbs_0 is a master key used by the voting server. In other words, the voter's vote consisting of the one-time signature $s = (\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ is MACed using the key Kbbs_0. The voter only gets the $\texttt{VotRecConCnt}_{i,j}$ half. Note that a receipt can only be verified with knowledge of the master key Kbbs_0.

Tallying is basically done by comparing the received pairs $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ with the public keys of all voters. Each time a match is found (i.e., when $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ is a valid signature w.r.t. a given public key and a candidate $\texttt{Cm}_j$), the score for the corresponding candidate is incremented by 1. (In the actual RIES system it is also possible that more than one vote is issued per voter, e.g., one by Internet and one by postal mail. A paper ballot is converted by the RIES system into a digital vote, by reconstructing the voter's key $\texttt{Kp}_i$ and then simply computing $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ for the candidate indicated by the voter on the ballot. The double votes are then resolved according to some pre-defined rules.)

All received pairs $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$ are published, such that anyone can recompute the election result, also using the public keys of all voters. In addition, the $\texttt{VotRecConCnt}_{i,j}$ halves of voter receipts are published as well, next to the corresponding votes $(\texttt{VnPID}_i, \texttt{VnCx}_{i,j})$.

In addition to the main cryptographic protocol, the following mechanism is used to facilitate handling of voters during an election. To keep track of a voter's status the following value is used, which is derived directly from the voter's private key $\texttt{Kp}_i$ as follows:

$$\texttt{ReSPID}_i = \texttt{MDC-2}(\texttt{DESmac}(\texttt{Kp}_i, \texttt{ElID} \| \texttt{ExtParGp})).$$

There is no cryptographic reason why an MDC-2 is applied to the MAC value in this case; note that an 8-byte DESmac is expanded to a 16-byte MDC-2 value. Also, the value of $\texttt{ReSPID}_i$ is known to several parties throughout the system so it is not really used as a secret value, but rather as a value unique to each voter.

5 Security analysis of RIES

In this chapter we analyze the security of RIES. We start with a few security issues we have come across that we have not seen addressed before. Then we analyse RIES against the criteria discussed in Chapter 2.

Previous papers analyzing security aspects of (variants of) RIES are [6], [12] and [9].

5.1 Some Issues

In this section some concrete issues and attacks are described, which are not addressed by the cryptographic protocol nor by the procedural steps.

5.1.1 Forging Votes

Prior to the election the list $L = \{(\texttt{RnPID}_i, \{\texttt{RnCx}_{i,j}\}_j)\}_i$ is published. Individual votes $\texttt{VnPID}_i, \texttt{VnCx}_{i,j}$ will be accepted only if the verification against this list succeeds. The problem is, however, that both $\texttt{VnPID}_i$ and $\texttt{VnCx}_{i,j}$ are only 64 bits in length. (The fact that these values are actually MAC values is irrelevant to the forgery attack, as described here.)

An attacker with access to the public list L, but without access to the voter's private keys or any other secret values, may proceed as follows. Suppose for the purpose of illustration that the public list consists in total[6] of 2^{31} different hash values (`MDC-2` images). it follows that if one picks a 64-bit string x uniformly at random, that with probability 2^{-33} it will hold that $\texttt{MDC-2}(x)$ appears on the list L. Therefore, if we create a list of length 2^{33} of different random 64-bit strings, with probability 1, we see that we find a pre-image for one of the hash values appearing on list L.

Since the `MDC-2` pre-images are already random (as these are MAC values), it suffices for the attacker to deterministically try the hash values of $x = 0$, $x = 1$, $x = 2$, and so on. The attacker may actually prepare a database of values $\{\texttt{MDC-2}(x) \mid 0 \leq x < B\}$ for some bound B, and put this database in sorted order. There is no need to store the full `MDC-2` value of 128 bits; storing 64 bits or even less should be fine. Without any further optimizations, the storage requirement would be 2^{32} times 64 bits, or 32 GBytes of storage.

With probability close to 1, the pre-image x corresponds to a vote different than cast by the legitimate voter, and the attacker may thus replace such a vote by its own vote. The amount of work is thus very limited.[7] The attacker cannot control in advance for which candidate or party the replaced vote will be, but repeating the attack a sufficient number

[6] In the Dutch District Water Control Boards elections with 180 candidates (on average; 15 candidates per party, 12 parties per sub-election) and 12 million eligible voters, we get in total about $2.2 \times 10^9 \approx 2^{31}$ hash values. The expected number of collisions in this list is below 1.

[7] `MDC-2` requires 2 applications of `DES` to process one input block. COPACOBANA is claimed (CHES 2007) to achieve 400×10^6 `DES` encryptions per second per 40 euro FPGA at 100 MHz, versus 2×10^6 `DES` encryptions per 80 euro Pentium-4 at 3 GHz. See `http://www.copacobana.org/`.

of times will soon lead to success. The attack can be repeated as many times as desired, such that a sizeable fraction of the votes can be forged. The total effort will still be much lower than a brute-force DES key search.

Note that it does not matter that an election is actually divided into sub-elections. The attack takes as input all published hash values, gathered from all sub-elections. The attack is also independent of any keys used, possibly specific to each sub-election. This is due to the fact that casting a legitimate vote boils down to producing correct one-time signatures, which consist of two 64-bit values: $\texttt{VnPID}_i, \texttt{VnCx}_{i,j}$. The attacker only needs to forge the second part, copying the first one from the vote cast by the legitimate voter. Also, note that the bulk of the attack can be prepared beforehand: it's a matter of computing and storing a large table of the form $\{\texttt{MDC-2}(x) \mid 0 \leq x < B\}$ in sorted order. Once the list of public keys of all voters is published, one can simply run a matching algorithm. The attack can be performed trivially in parallel, using disjoint subsets of $\{\texttt{MDC-2}(x) \mid 0 \leq x < B\}$.[8]

An attack as described above depends on access to the received votes, hence probably to the voting servers, before the tally phase starts. Probably the easiest way to achieve this is with insider help. It is conceivable that the attack might even be mounted in such a manner that the vote receipt will be calculated for the replaced vote, in stead of for the original vote. In that case, a complaining voter will not be able to prove that his intended vote was different. As a consequence, the handling of the votes (voted ballots) must be done much more securely, as the cryptographic protection is insufficient.

Countermeasures to prevent this attack are not hard to find and implement. A simple counter to diversify the $\texttt{VnPID}_i$ values for one voter over all candidates suffices.

5.1.2 Secure PCs

RIES assumes that the voter's PCs are secure. Attackers may however employ malware or even 'man-in-the-browser' attacks to capture voter's PCs. Powerful attackers may thus change votes, and so this involves a unique potential risk for Internet elections.

5.1.3 Status of cryptanalysis for the cryptographic mechanisms

RIES uses single `DES`, rather than the more secure `3DES`, for `DESmac`-operations to be performed by the voter. The reason is that these keys have to be typed in by humans of varying typing capabilities, and thus should be not too long to keep the typing error rate low. As a result the effective key length of operations with the voter key is 56 bits. In the cryptographic community this is commonly seen as rather weak. Apparently in the RIES design this is seen as acceptable.

According to [24], `MDC-2` based on `DES` allows collision attacks using 2^{55} hashing operations (which more or less matches the key size of 56 bits), and preimage attacks using 2^{83} hashing

[8] Activists may discredit the elections by publishing all the hits found, and anyone interested can join to compute further hits.

operations. Though `DESmac` based on `DES` is not the strongest MAC available, the known attacks are not an immediate threat.

The hash function `SHA-1`, used for computing commitments to tables published for verifiability of the election process and results, is known to be vulnerable to collision attacks of, presently, about 2^{60} compression function operations [20]. Again, known attacks on `SHA-1` are no immediate threat, but it should have almost no impact on the operation of RIES to replace `SHA-1` by a hash function that still does meet its design criteria, such as SHA-256.

5.2 Verification of the requirements

In this section we analyse to what extent the design of RIES-2008 meets the criteria described in Chapter 2, and to what extent RIES-2008 has reached improvements over RIES-KOA.

1. ***transparency*** *The election process should be organized in such a way that the structure and organization is clear, so that everyone in principle can understand it. There must be no secrets in the election process: questions must be able to be answered, and the answers must be verifiable.*

   The structure and organization of RIES-2008 are reasonably clear. Extensive documentation is provided by the designers and organizers (especially about the system design [15, 18]). Understanding the details requires a certain amount of technical expertise, but there are no secrets (except, necessarily, particular cryptographic keys). Design decisions regarding cryptographic details are however at various places ill-motivated and hard to understand for a cryptographic expert.

   The source code of the system will shortly become openly available for inspection. The value of this availability is of course limited, because it is hard to verify that the published code is also the one running on the system. Proper auditing procedures should be in place to increase the required trust.

   In general the designers and organizers have an open attitude, have discussed the working of their system openly with everyone interested and have organized several workshops about RIES, in its various incarnations.

   In the transition from RIES-KOA to RIES-2008 separation of duties and responsibilities has improved: where in RIES-KOA the design and development team also had operational duties, this is not allowed anymore in RIES-2008.

2. ***verifiability*** *The election process should be objectively verifiable. The verification tools may differ, depending on the method of voting that is decided upon.*

   Verifiability in RIES-2008 is organized at three levels.

   - Each individual voter can verify, on the basis of his technical vote, whether his actual vote has been taken into account. This works by looking up the technical vote in the tables of performed votes (preliminary) or of final results. This form

of verifiability requires that sufficiently many voters actually check their votes. There is no reliable information about what percentage of people really do this. In particular, eligible voters who do not cast a vote, will in general not verify that no vote has been registered for their $\texttt{Kp}_i$. And if they do and notice that a vote has actually been cast, then they have no way to prove that they didn't actually cast the vote (of course, they don't have a receipt).

When the voter receives the confirmation receipt it is not possible to check whether it is appropriate—since it is computed by the vote server using a secret key (namely `Kbbs_0`). Therefore, disputes can be resolved only to a limited extent, as briefly illustrated above.

- The calculation of the end-results can be done by everyone interested, on the basis of the tables with final results (as provided by the organizers). However, one cannot check that the results appearing in these tables are actually the results as intended by the voters.
- The umpire can check and recalculate various steps in the whole process and pass his own judgement. But the umpire can handle only a limited type of disputes. The use of the receipt $\texttt{VotRecConCnt}_{i,j}$ is limited. A voter has no way to check the validity of $\texttt{VotRecConCnt}_{i,j}$ when it is sent to the voter at the conclusion of the voting protocol. The reason is that verification requires access to the secret key `Kbbs_0`. Second, a voter may present a fake $\texttt{VotRecConCnt}_{i,j}$ and claim that it was sent by the voting server. Also, the voter may claim that even though the receipt is present, that the voter actually voted for a different candidate but the system apparently recorded it wrongly, and even delivered a wrong receipt to the voter (i.e., a correct receipt but for a different vote). The umpire can notice and report anomalies, but it is not clear that he can resolve them all.

Independent implementations of individual voter verification and calculation of tally results can be used shortly after the election closure, making use of the published reference and performed votes tables. For RIES-KOA such an implementation was provided by the Radboud University Nijmegen. Verification of an individual vote by such a service implies that the voter has to expose his vote to the service. One could imagine that political parties are interested in setting up verification services.

3. ***fairness / integrity*** *The election process should operate in a proper manner, and the results must not be capable of being influenced other than by the casting of lawful votes.*

   This requirement is not fulfilled because of what must be regarded as the greatest structural weakness in the design of RIES: at a central level crucial secret keys (the $\texttt{Kp}_i$) are generated—and available in the clear at certain stages (notably printing)—for all individual voters. Also via the replacement packages it is possible to manipulate votes. Hence it is, in principle, possible that the results are 'capable of being influenced other than by the casting of lawful votes'.

   It should however be noted that RIES-2008 has improved considerably over RIES-

KOA in this aspect, because all important master keys are now generated and operated in tamper-resistant cryptographic hardware, and cannot be exposed to humans.

Also, since votes are not encrypted, the voting server sees all votes as soon as they are cast. Therefore, the voting server is able to perform intermediate tallies[9], which is not allowed, as intermediate election results may influence the proceedings of the election. The fact that the voting server receives and stores the votes in the clear is also a serious weakness.

In Section 5.1 a forging attack is described, that requires cooperation of an insider, and then enables an attacker to forge a notable amount of votes.

The measures to mitigate these risks in RIES-2008 are organizational. Ideally, this requirement is enforced by the cryptographic structure of the voting system.

4. ***eligibility to vote*** *Only persons eligible to vote must be allowed to take part in the election.*

   Essentially the procedure for RIES is the same for as for other elections for public bodies in the Netherlands: citizens receive their ballot on the basis of the citizen administration (GBA). However, RIES adds one thing: because the reference table is published in advance, the maximal number of possible votes is fixed in advance. It is thus in principle possible to trace this number back to the source (the GBA), making fraudulent addition of voters, if any, visible.

5. ***equal suffrage / unicity*** *Each voter, given the Dutch election system, must be allowed to cast only one vote in each election, which must be counted precisely once.*

   The tally-system helps in enforcing this property, once one assumes that an eligible voter gets hold of precisely one ballot. However, note that a voter's key $\texttt{Kp}_i$ is used at various places in the system and one must assume that no insider takes advantage of its access to a such a copy of the voter's key. With access to $\texttt{Kp}_i$, one may issue a second vote (e.g., by Internet for a voter who only used a postal ballot), as a consequence of which the voter's original vote may not be counted.

6. ***free suffrage / vote freedom*** *Every elector must be able to choose how to vote in complete freedom, free from influence.*

   Within postal and Internet systems like RIES this requirement is the sole responsibility of the individual voter. A voter can, in principle, sell his vote, or can be coerced to vote for a particular candidate. A voter can simply send a copy of $\texttt{Kp}_i$ to a coercer, e.g. by email. (Note that $\texttt{Kp}_i$ is printed in a human-friendly manner in the material sent to a voter.). The coercer does not have to see the voter's ballot papers at all, because the validity of $\texttt{Kp}_i$ can be checked against the public list $\{(\texttt{RnPID}_i, \{\texttt{RnCx}_{i,j}\}_j)\}_i$. This is done simply by generating some test votes, and checking the validity of these—if

[9] Actually, this is even standard procedure in RIES, in the so-called *pre-tallying* process, for monitoring reasons. Pre-tally results are encrypted under a public key for which the corresponding decryption key is available only to a notary.

desired, the coercer simply runs test votes for all candidates to be absolutely sure that $\texttt{Kp}_i$ is valid.

The fact that voters can later prove what they have voted aggravates this point, since a coercer may demand to see the technical vote, as proof.

Hence this requirement is not fulfilled within RIES, like for postal votes.

7. ***secret suffrage / vote secrecy*** *It must be impossible to connect the identity of a person casting a vote to the vote cast. The process should be organized in such a way that it is impossible to make a voter indicate how he or she voted.*

   The situation is partly as for the previous requirement: in the personal environment of the voter, the secrecy of the connection between voter and the vote that is cast is the responsibility of the voter. Once the vote is expressed via the Internet and stored inside the system, secrecy depends on the way the personal keys $\texttt{Kp}_i$ are handled. As stated before, secrecy of these keys depends on organizational matters. Additionally, vote servers can link the originating IP address to the vote that is cast (using the reference table): there is no anonymous channel. Again, organizational measures must prevent such linking.

8. ***accessibility*** *Voters should be enabled as far as possible to participate directly in the election process. If this is impossible, there must be a way of taking part indirectly, i.e. by proxy.*

   In comparison to (ordinary) elections in poll stations one may argue that accessibility is better in RIES because physically disabled people can vote at home. In general voters are not forced to use the Internet, since they can still vote via postal ballots.

6 Conclusion

RIES is an evolving family of systems (RIES-2004, RIES-KOA, RIES-2008) for electronic elections via the Internet. It has been used in practice for a number of District Water Control Board elections (involving millions of potential voters, but hundreds of thousands actual voters) and for expatriates in national parliament elections (involving hundreds of thousands of potential and tens of thousands actual voters). It is among the 'largest' Internet voting systems worldwide, in terms of actual use, and maybe even the largest. This deployment of RIES has resulted in valuable experience and expertise in how to run Internet elections.

RIES has been developed in a specific context, in which postal (District Water Control Board) elections form the framework of comparison. Hence certain reasonable goals for elections (like vote freedom) have been out of scope from the start. Also, this framework did not allow any (technical) requirements on the voter's side, such as availability of smart cards: only a standard, reasonably up-to-date PC could be used, that is assumed to be trusted. As discussed in Section 5, the general voting requirements formulated by the

Korthals Altes Committee [11] are not all satisfied: not only vote freedom but also vote integrity and confidentiality are not structurally guaranteed.

RIES is built on certain cryptographic primitives, like one-time signatures, which are realized via hash functions and encryption with centrally generated keys for individual voters. There are no anonymous channels. The structural protection and safeguards offered by cryptography are rather limited. Many of the guarantees thus rely on organizational controls, notably with respect to (voter) key generation, production of postal packages, insider attacks (especially at the server), integrity and authenticity of the software, and helpdesk procedures.

RIES-2008 is designed and built in an open spirit. Its source code will shortly be available openly for inspection. Also much (design) documentation will become publicly available. Additionally, the designers and organizers have put considerable effort in publicly explaining and discussing their system. The actual running of RIES elections also involves an independent umpire that has access to the internals and can run his own checks after the election.

One of the distinguishing aspects of RIES is that it allows independent recounts of the final outcome and individual checks to see if own votes have been included. This is an interesting and useful feature in itself, but does not compensate for the structural weakness of the limited use of cryptography, and the conclusions that can be drawn from these individual checks are limited.

The whole technical and organizational set-up is, as far as we could see from a study of the documentation, carefully designed. Various controls are in place to oversee the proper execution of all the required steps. There are however pragmatic elements in the system—such as the use of replacement packages—that are open to manipulation and abuse, notably by insiders. The RIES Internet election system also offers potentially dangerous ways for manipulation of elections, in principle applicable on a large scale and different from attacks on postal elections.

In a larger context we see RIES (esp. RIES-2008) as a project that yields valuable hands-on experience and expertise on how to organize and run electronic elections. RIES does not use the best that cryptography has to offer in this area, but, to be fair, there is currently no agreed standard on what techniques to use for such elections. We do not think RIES-2008 is a suitable system for use outside a context of postal elections, and in particular not for 'general' elections (like for national/European parliaments or local/regional councils), given the current technical state of affairs and the reservations formulated in [11]. We do encourage further research, development and experiments to gain more experience in this area: Internet elections are increasingly used for various forms of elections (like within organizations) and using Internet elections for public bodies will probably remain looming on the political agenda.

Acknowledgements

The authors are grateful to the Waterschapshuis (especially Simon Bouwman) for financial support and encouragement, and to Piet Maclaine Pont (MullPon), Arnout Hannink (Magic Choice), Xander Jansen (SURFnet) and Marco Rijkschroeff (Waterschapshuis) for providing a wealth of information and enlightening discussions.

References

[1] Rachid Anane, Richard Freeland, and Georgios K. Theodoropoulos. e-voting requirements and implementation. In *CEC/EEE*, pages 382–392, 2007.

[2] D. Bleichenbacher and U. Maurer. Directed acyclic graphs , one-way functions and digital signatures. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 75–82, Berlin, 1994. Springer-Verlag.

[3] Council of Europe, Recommendation Rec(2004)11 of the Committee of Ministers to member states on legal, operational and technical standards for e-voting, 2004. `https://wcd.coe.int/ViewDoc.jsp?id=778189`.

[4] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[5] Arnout Hannink, Mark Dobrinic, and Suze Maclaine Pont. Documentatie RIES-2008 TTPI Applicatie. version 1.0, February 2008, 2008.

[6] E. Hubbers, B. Jacobs, and W. Pieters. RIES. Internet Voting in Action. In R. Bilof, editor, *Proceedings of the 29th Annual International Computer Software and Applications Conference, COMPSAC'05*, pages 417–424. IEEE Computer Society, 2005.

[7] IBM, CCA Basic Services Reference and Guide for the IBM 4758 PCI and IBM 4764 PCI-X Cryptographic Coprocessors, 2008. 19th ed., `http://www-03.ibm.com/security/cryptocards/pcixcc/library.shtml`.

[8] D.B. Johnson, G.M. Dolan, M.J. Kelly, A.V. Le, and S.M. Matyas. Common Cryptographic Architecture - Cryptographic Application Programming Interface. *IBM Systems Journal*, 30(2):130–150, 1991.

[9] H.L. Jonker and M. Volkamer. Compliance of RIES to the proposed e-voting protection profile. In A. Alkassar and M. Volkamer, editors, *E-Voting and Identity (First International Conference, VOTE-ID 2007, Bochum, Germany, October 4-5, 2007, Revised Selected Papers)*, volume 4896 of *Lecture Notes in Computer Science*, pages 50–61. Springer, Berlin, 2007.

[10] Gerjon Kobus, Jacques Schuurman, Paul Dekkers, Xander Jansen, and Suze Maclaine Pont. Documentatie RIES-2008, SURFnet, Deel 2: documentatie voor 'technisch geïnteresseerden'. version 1.0, February 1, 2008.

[11] F. Korthals Altes, J.M. Barendrecht, B.P.F. Jacobs, C. Meesters, and M.J.C. van der Wel. Voting with Confidence, 27 sept. 2007. Report of the national Election Process Advisory Commission, available at: `www.minbzk.nl/aspx/download.aspx?file=/contents/pages/90517/votingwithconfidence.pdf`.

[12] Lucas Kruijswijk. Internetstemmen met RIES onder de loep, 2006. `http://www.wijvertrouwenstemcomputersniet.nl/Internetstemmen_met_RIES_onder_de_loep`.

[13] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, 1979.

[14] P.G. Maclaine Pont. Systeem en werkwijze voor een electronische verkiezing. Octrooi NL C 1023861, Octrooicentrum Nederland, 2005.

[15] Piet Maclaine Pont. Design Information for Evaluation purposes about RIES, the Internet Election System to be used by Het Waterschapshuis. version 0.92, March 4, 2008.

[16] Piet Maclaine Pont. RIES-2008: HW-CRYPTO, Cryptographic Architecture for RIES-2008 and IBM 4764. version 0.95 draft, January 31, 2008.

[17] Piet Maclaine Pont. RIES-2008-Prepare: PSB C10 uitwisseling, Hoofdelementen voor planning ontwikkeling en test. version 0.3 draft, January 18, 2008.

[18] Piet Maclaine Pont, Arnout Hannink, Jacques Hoeienbos, Marco Rijkschroeff, and Jacques Schuurman. RIES-2008, Functioneel Ontwerp. version 1.0-concept, February 19, 2008.

[19] Suze Maclaine Pont, Piet Maclaine Pont, and Arnout Hannink. RIES WV-STUF 2008, Standaard Uitwisseling Formaat Waterschapsverkiezingen RIES-2008. version 1.1, February 6, 2008.

[20] Florian Mendel, Christian Rechberger, and Vincent Rijmen. Update on SHA-1. Crypto 2007 Rump Session, August 21, 2007, `http://www.iaik.tu-graz.ac.at/aboutus/people/rechberger/talks/Rechberger_SHA1BOINC_V07.pdf`.

[21] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[22] R. Merkle. A digital signature based on a conventional encryption function,. In *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378, Berlin, 1987. Springer-Verlag.

[23] NIST, FIPS PUB 140-2, Security Requirements for Cryptographic Modules, 2001. Federal Information Processing Standards Publication, Information Technology Laboratory, National Institute of Standards and Technology.

[24] B. Preneel. Hash functions and MAC algorithms based on block ciphers. In M. Darnell, editor, *Cryptography and Coding, 6th IMA International Conference*, volume 1355 of *Lecture Notes in Computer Science*, pages 270–282, Berlin, 1997. Springer-Verlag.

[25] Bijlagen RIES WV-STUF, 2008. version 11.1.

[26] Herman Robers. Electronic Elections employing DES Smartcards. Master's thesis, Delft University of Technology, December 1998. `www.iscit.surfnet.nl/team/Herman/election.ps`.