REVS – A ROBUST ELECTRONIC VOTING SYSTEM

Rui Joaquim, André Zúquete, Paulo Ferreira
Instituto Superior Técnico (Technical Univ. of Lisbon) / INESC ID
R. Alves Redol, 9 – 6º andar
1000 Lisboa, Portugal
[rui.joaquim, andre.zuquete, paulo.ferreira]@gsd-inesc-id.pt

ABSTRACT

There are many protocols proposed for electronic voting, but only a few of them have prototypes implemented. Usually the prototypes are focused in the characteristics of the protocol and do not handle properly some real world issues, such as fault tolerance. This paper presents REVS, a robust electronic voting system designed for distributed and faulty environments, namely the Internet. The goal of REVS is to be an electronic voting system that accomplishes the desired characteristics of traditional voting systems, such as accuracy, democracy, privacy and verifiability. In addition, REVS deals with failures in real world scenarios, such as machine or communication failures, which can lead to protocol interruptions. REVS robustness has consequences at three levels: (i) the voting process can be interrupted and recovered without weakening the voting protocol; (ii) it allows a certain degree of failures, with server replication; and (iii) none of the servers conducting the election, by its own or to a certain level of collusion, can corrupt the election outcome.

KEYWORDS

Electronic voting, Fault-Tolerance, Robustness, Blind signatures.

1. INTRODUCTION

In the last few years several experiences have been conducted in order to facilitate the voting process in elections. Such facilitations were introduced by new ways of expressing votes besides the traditional paper-based. Examples of new voting interfaces and systems are touch screens, SMS messages from cellular phones and distributed voting systems using the Internet (Monteiro 2001, UK-eDemocracy 2003).

Internet voting systems are appealing for several reasons: (i) people are getting more used to work with computers to do all sort of things, namely sensitive operations such as shopping and home banking; (ii) they allow people to vote far from where they usually live, helping to reduce abstention rates; and (iii) they may support arbitrary voting ballots and check their correct fulfillment during the voting process.

However, Internet voting systems face several problems that prevent their widespread use today (CIVTF 2000, CALTECH-MIT 2001, Cranor 2001, IPI 2001, Rivest 2001, Rubin 2002). The problems can be broadly divided in three main classes.

The first class of problems includes security and fault tolerance issues inherited from the current Internet architecture. Vital services, such as DNS name resolution, can be tampered in order to mislead users into spoofing servers (Lioy et al. 2000). IP routing mechanisms and protocols, managed by many different organizations, should deal with partial communication outages, however communication problems may arise.

The second class of problems includes issues that are specific to voting protocols. These problems derive from the assumptions of the protocols about the execution environment, namely:

- Client machines used by voters must be trusted, in order to act as “trusted agents”, which is hard to ensure in personal or multi-user computers with general-purpose commercial operation systems.
- Servers controlling the voting process cannot (i) fail, (ii) become unreachable or (iii) pervert the voting protocol. The protocol perversion can be done either by not reacting properly to client requests or by trying to influence the election by acting as a voter.
- The voting protocol is not disturbed by communication problems or machine failures.

The third class of problems includes those difficulties that may be created by specific attacks against a voting protocol or a running election. Such attacks may try to get some useful outcome, by subverting the voting protocol, or simply ruin an election using DoS (Denial of Service) attacks against the participating machines or applications. Another kind of attack is the coercion of voters, which can happen if they can vote anywhere without supervision of electoral committees.

REVS is an Internet voting system designed to tackle some of these problems. In particular, the REVS voting protocol, involving several participating machines, supports some types of communication and machine failures by keeping a distributed loosely-coupled state. Each voter keeps a local state, in mobile non-volatile storage, allowing him to stop and resume the election anytime and anywhere. Servers are replicated and only a subset of them needs to be contacted by each voter. Each server keeps a distinct state regarding the participation of each voter in the election, allowing voters to get many times the same answer from each server. Each server alone is not able to act as any voter and cannot provide false replies to voters without being noticed. The collusion of servers in order to interfere with the election (e.g. voting for absentees) is prevented to a certain degree of collusion.

REVS is a blind signature electronic voting system based in DuRette's (1999) work, which improved the EVOX system (Herschberg 1997). DuRette, with the EVOX Managed Administrators (EVOX-MA) system, improved EVOX in order eliminate single entities capable of corrupting the election. Both EVOX and EVOX-MA are very sensible to failures in communication or servers, a problem that we solved with REVS. Furthermore, the EVOX-MA has problems concerning the authentication of voters, allowing an easy impersonation of voters by the servers running the election. In REVS we solved this problem redesigning the voters' authentication algorithm.

A first prototype of REVS was implemented in the Instituto Superior Técnico (Technical Univ. of Lisbon) to support elections, namely surveys for evaluating the quality of courses. The experimental results obtained indicate that the architectural and implementation solutions were adequate to fulfill the goals/requirements of the voting system.

This document is structured as follows. Section 2 presents an overview of previous work stressing the work in which REVS is based on. Section 3 describes the protocol used and how it achieves robustness. In Section 4 we describe the implementation details of REVS. Section 5 presents an evaluation of the system and finally in Section 6 we draw some conclusions.

2. RELATED WORK

Researchers have been working in the electronic voting research area mainly after 1980, with an emphasis in the last decade. Currently there is a consolidate taxonomy for classifying electronic voting systems and well-defined sets of protocols for implementing them. We start with a presentation of the taxonomy of the electoral process and the properties that voting systems should have, then we briefly describe the basic types of voting protocols, and finally we describe EVOX and EVOX-MA systems, upon which we designed REVS.

2.1. The electoral process

The electoral process is all that concerns to the realization of an election. To a better understanding it is useful to divide the electoral process into several phases:

Registration/Preparation: In this phase it is compiled the list of eligible voters; the ballots are prepared and all the logistics arrangements needed for the election are made. This phase also includes any other activity that must be done before the election start.

Validation: Before allowing a voter to vote, the voters' credentials must be verified. Only the registered voters should be authorized to vote, and only once.

Collection: The collection phase consists in collecting the voters' ballots.

Verification: In this phase it is verified the validity of the ballots. Only the valid ballots are used in the tallying phase.

Tallying: The tallying phase consists in counting the valid ballots. At the end the tally is published.

Claiming: This is the phase in which all the claims should be made and investigated. At the end of this phase the final results are published.

2.2. Properties of voting systems

Researchers in the electronic voting field have already reached a consensus pack of four core properties that an electronic voting system should have (Cranor and Cytron 1997):

Accuracy: (1) it is not possible for a vote to be altered, (2) it is not possible for a validated vote to be eliminated from the final tally, and (3) it is not possible for an invalid vote to be counted in the final tally.
Democracy: (1) it permits only eligible voters to vote and, (2) it ensures that eligible voters vote only once.
Privacy: (1) neither authorities nor anyone else can link any ballot to the voter who cast it and (2) no voter can prove that he voted in a particular way.
Verifiability: anyone can independently verify that all votes have been counted correctly.

Accuracy, democracy and verifiability are, in most cases of today's electoral systems, assured by the presence of representatives of opposite parties. The privacy property is currently assured by the existence of private voting booths, allowing voters to cast their votes in secrecy.

The particular nature of the Internet environment requires robust Internet voting systems. We define robustness as the join of the following three characteristics:

Collusion Resistance: no electoral entity (any server participating in the election),or group of entities, running the election can work in a conspiracy to introduce votes or to prevent voters from voting. If all entities conspire this property is not achieved. Therefore, this characteristic should be measured in terms of the total number of entities that must conspire to achieve a successful interference in the election.
Availability: (1) the system works properly as long as the poll stands and (2) any voter can have access to it from the beginning to the end of the poll.
Resume Ability: the system allows any voter who had interrupted his/her voting process to resume it or restart it while the poll stands.

2.3. Basic types of voting protocols

The electronic voting protocols proposed so far are mainly divided in three different approaches:

Blind signatures: protocols based in blind signatures[1] (Fujioka et al. 1992, Sako 1994, Cranor and Cytron 1997, Herschberg 1997, Okamoto 1997, Ohkubo et al. 1999 and DuRette 1999) have the advantages of simplicity, low computational costs and being ballot independent.
Mix-nets: protocols based in mix-nets (Chaum 1981, Park et al. 1993, Sako and Kilian 1995, Ogata et al. 1997 and Jackobsson 1998) require less voter's interactions, but need complex proofs of correctness.
Homomorphic encryption: protocols based in homomorphic encryption (Benaloh and Fischer 1985, Benaloh and Yung 1986, Benaloh 1987, Benaloh and Tuinstra 1994, Sako and Kilian 1994, Cramer et al. 1996, Cramer et al. 1997, Hirt and Sako 2000 and Baundron et al. 2001) have a very complex mathematical structure, therefore, high computational costs. This type of systems also imposes great ballot restrictions.

Each approach has pros and cons. The more flexible and with computationally less demands are those based on blind signatures. Therefore, we chose this type of system for REVS.

Fujioka et al. proposed in 1992 a blind signature protocol, know as FOO, which became the reference in blind signature voting protocols. Later, Cranor and Cytron (1997) with Sensus, and Herschberg (1997) with EVOX, proposed their voting protocols and systems based on FOO. However, in both protocols an Administrator server can introduce votes. In 1999, DuRette proposed the EVOX-MA protocol, an evolution

[1] Blind signatures are a class of digital signatures, consisting in getting a message digitally signed without giving any knowledge about the message to the signer. This is like putting a document and a sheet of carbon paper in a sealed envelope that somebody signs on the outside. After removing the envelope we get the signed document.

of EVOX. The goal of EVOX-MA was to prevent the Administrator from adding votes. REVS is based in EVOX-MA system, but we modified the system in order to be fault tolerant, thus assuring availability. Furthermore, we modified the algorithm for authenticating users because the one used in EVOX-MA has personification problems.

2.4. The EVOX and EVOX Managed Administrators voting systems

In this Section we briefly explain the EVOX and EVOX-MA voting systems. We start by explaining the EVOX system and pointing out the problem that EVOX-MA tries to solve. Then we present the EVOX-MA and point out its problems.

In FOO, the blind signatures reference protocol, voters have to participate in the ballot tallying phase. This means that voters cannot simply vote and walk away. Consequently, a direct implementation of the FOO protocol would result in a system with enormous inconvenience for the voters.

The EVOX voting protocol is a voting system based on FOO but allowing voters to vote and walk away. It works as follows (c.f. Figure 1): first, the voter contacts the Administrator to obtain the ballot (1); then the voter fills the ballot and sends it to the Administrator for signing (2); after receiving the Administrator's signature, the voter sends the ballot and signature to the Counter through the Anonymizer.

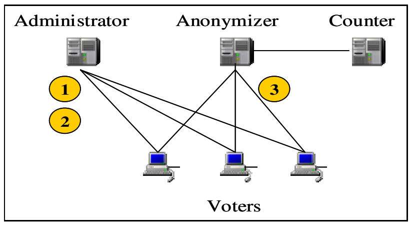

Figure 1 – EVOX voting protocol

In EVOX, as in many others blind signatures based voting protocols, the Administrator can insert votes for the absentee voters without being noticed. This is exactly the problem that EVOX-MA tackles. The idea is to ensure accuracy by sharing the power of the election Administrator among several servers: there are $\boldsymbol{N}$ Administrators and it is required signatures from $\boldsymbol{t}$ of them, $t \leq N$ to make a ballot valid. An additional signature from the Manager ensures democracy.

Here is an overview of the protocol (c.f. Figure 2): the voter contacts the Manager and obtains a ballot (1), fills it and contacts t Administrators for signing his ballot (2). Then the voter gets the Manager's signature over all the signatures previously obtained from the Administrators (3) and, finally, he submits his vote to the Counter through an Anonymizer, which protects the voter's privacy from the Counter (4) [2].

If the Manager is not trusted then, in theory, the value of t should be $t \geq N/2 + 1$. However, in practice this threshold depends on the implementation and may not be enough to prevent Manager's frauds. In fact, the EVOX-MA system uses one password per voter for all Administrators and for the Manager. None of these entities knows the password in advance, because a UNIX-like validation is used. However, a small set of Administrators, colluded with the Manager, can generate false votes using a voter's password once they get it. The fraud may be performed as follows: x colluded Administrators use a voter's password to get signatures from all the Administrators not contacted by the voter and send to the Manager a signed vote that it could accept and send to the Counter. With N Administrators and $N/2+\Delta$ required signatures, x is equal to 2Δ. If Δ is a low value, 1 or 2 for improving performance, the possibility of attack is not negligible. If t, the number of required signatures, is less than $N/2$, the Manager, by itself, can introduce votes without the participation of any other entity. But in both cases the Manager must actively participate in the fraud to drop legitimate votes and replace them by votes produced by the colluded Administrators or by itself. Thus, in practice, even with $t \geq N/2 + 1$ the Manager, colluded with a small number of Administrators, can impersonate voters.

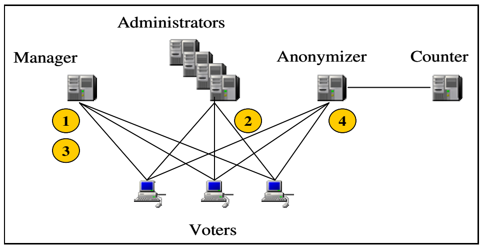

Figure 2 – EVOX Managed Administrator

[2] Note that in both protocols the signatures are obtained using the blind signature method.

In REVS we tried to eliminate this weakness. Consequently, the authentication method in REVS is also username/password based but uses a different password for each server, preventing voter's personification by less than *t* colluded Administrators. The details about the authentication scheme are presented in Section 4.

Another problem, but now respecting the robustness of the systems, is the existence of singular points of failure, namely the Administrator in EVOX and the Manager in EVOX-MA (the authors propose some replication of the other servers involved).

The Administrator, in EVOX, and the Manager, in EVOX-MA, must sign all ballots and are also responsible for ballot distribution. Any malfunction on those servers leads to an interruption of the election, as they are singular points of failure. Thus, both systems do not ensure availability. Furthermore, those servers are performance bottlenecks when considering large-scale elections.

Single points of failure should be avoided to increase availability, one of the properties that a robust electronic system must have (c.f. Section 2.2). REVS has this property by allowing the deployment of voting systems without singular points of failure.

The possibility to run several elections simultaneously is given in both EVOX and EVOX-MA systems. This is done distributing the right ballots to the right voters. However, nothing was done to prevent voters from exchanging ballots, or even the use of steal ballots from other voters. Thus, it is possible for voters to participate in elections for which they should not be authorized to participate.

3. REVS VOTING PROTOCOL

In REVS we have five types of servers: Commissioner, Ballot Distributor, Administrator, Anonymizer and Counter. There is also a Voter Module, that is used by voters to support their participation in elections, performing all the proper interactions with election servers (get the ballot, get it signed by election servers, validate and submit the ballot, etc.).

Commissioner: Similarly to EVOX and DuRette's systems, in REVS we also have a Commissioner. In all three systems the Commissioner is the election supervisor, receiving complains made by any voter or electoral server. If the received complains raise any suspicion, an investigation is made to find out the causes.

The Commissioner in REVS has also the responsibility of preparing the election, generating and keeping secret the election's keys, signing the ballot questions and defining the operational configuration for the election (addresses and public keys of servers, number of required signatures, etc.).

Ballot Distributor: The Ballot Distributor is responsible for the distribution, for the voters, of the data set up by the Commissioner for the election: ballots and operational configuration. This procedure is expensive in terms of data exchange, so we decided to introduce this dedicated server in REVS.

All the information distributed by a Ballot Distributor must be signed by the Commissioner, in which all voters trust. Thus, there may be several Ballot Distributors. This replication, besides reducing the work load on each Ballot Distributor, improving efficiency in large-scale elections, provides protection to communication or machine failures affecting Distributors, bringing robustness to the distribution process.

Administrators: The Administrators are the electoral entities that have the power to decide upon the acceptability of a ballot from a voter. A ballot is acceptable for the final tally of the election only if it has a minimum set of signatures from different Administrators. If N is the total number of Administrators a voter must get $t > N/2$ valid signatures from different Administrators to make its ballot acceptable. This makes impossible for any voter to get two valid votes.

A voter uses a different password to get a signature from each Administrator. And because one Administrator cannot derive any other passwords from the one it knows, as is shown in Section 4.2, it cannot alone impersonate a voter.

Anonymizer: The Anonymizer server provides anonymity to the voter's machine, preventing a Counter from associating a ballot to the machine owner. A voter can choose any number of Anonymizers from the ones participating in the election. The Anonymizer hides the voter's location and randomly delays and shuffles several submitted ballots before sending them to Counters. The randomization of ballot submissions prevents time analysis trying to discover voters' ballots from the hour that have voted.

Counter: The Counter is the server who verifies the validity of the ballots, verifying that all required signatures are on the ballot. Then the Counter removes the repeated ballots verifying a bit commitment (made by the voter in the ballot signing phase (see, c.f. Section 3.1) and performs the tally.

Voters send their final ballots to Counters through Anonymizers encrypted with the public key of the election preventing Anonymizers and Counters from watching votes during the election. Counters can only analyze the votes when the election ends and the Commissioner releases the election key (private key).

REVS allows configurations with no single points of failure. So, in those configurations there must be several Counters, each one of them reachable through an independent Anonymizer.

In the case of the existence of several Counters the voters can send theirs ballots to any Counter (or, more correctly, to any Anonymizer), or even to several Counters. This means that different Counters will get different sets of votes at the end of the election, and those sets may even contain repeated votes. A selected master Counter obtains the final tally after gathering all the valid votes from the several Counters and discarding the repeated ones. Any person with access to the ballots collected by all Counters can act as a master Counter. This increases the confidence in the election outcome.

3.1. The Protocol

The REVS protocol allows a flexible replication of all servers involved in the election process. With the introduction of Ballot Distributors we separated every logical function into a different server, leading to a modular system. To prevent double voting by the voters the number of required signatures to make a ballot valid, *t*, must be greater than *N/2*, where *N* is the number of Administrators.

From the voters' point of view, the REVS protocol is divided in three steps (see Figure 3, detailed messages in Figure 4):

1. **Ballot distribution:** The voter contacts a Ballot Distributor to get a blank ballot for a given election. The Ballot Distributor returns the requested ballot, the election public key and the election's operational configuration, all signed by the Commissioner public key. This is done in two phases. First, the voter contacts a Ballot Distributor and provides the voter ID to receive the list of elections in which he can participate. Then the voter chooses an election and requests a ballot for it from a Ballot Distributor.

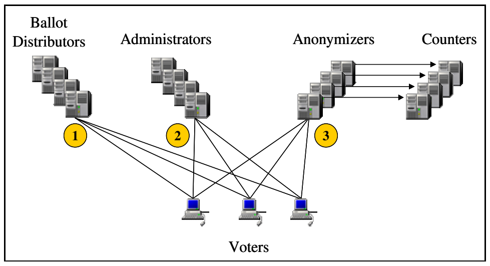

Figure 3 – REVS protocol

2. **Ballot signing:** After expressing his will on the ballot, the voter commits to the ballot digest with a random bit string and blinds the committed digest with a random blinding factor. Then the voter sends his blinded committed ballot digest to the Administrators for signing.

The voter's module saves the answers, the bit commitment and the blinding factor into a non-volatile storage, preferably provided by a mobile media, before using them. This enables the voter to stop and latter resume its participation in the election, but affects the voter's privacy.

Each Administrator, after receiving a request for signing, verifies if it had already signed for the requesting voter. If not, he signs and saves the signature; if he had signed before, the Administrator returns the previously saved data, i.e. the signature of the blinded committed ballot digest.

After receiving a signature (BB signed) the voter updates it using the blinding factor and verifies its correctness using the original ballot digest and the Administrator's public key. This process is repeated until all required *t* signatures are collected. Note that Administrators cannot link the signatures they provided to the vote with the signatures that the voter gets after applying the blinding factor. Therefore, concerning the Administrators, the privacy property is assured.

3. **Ballot submission:** in this step the voter constructs the ballot submission package, joining the ballot, its signatures and the bit commitment. At this time the voter can save this data into secure storage. This is an optional step, because it helps improving accuracy but affects privacy. Then he submits this package, ciphered with a hybrid cryptosystem using a random symmetric session key and the election's public key,

through the Anonymizer, concluding the voting protocol. The voter can submit the same package to any Counter as many times as he feels necessary to be sure that the ballot has reached its destination.

After the end of the election, the Commissioner publishes the election private key. Then the counting process is performed by the Counters and involves the following steps:

a) Decipher each submission package with the election's private key.
b) Verifying for each package that all required *t* signatures from Administrators are present.
c) Remove the repeated votes, which are the ones with the same bit commitment. If the length of the bit commitment is large enough (160 bits in REVS) the danger of collisions is negligible.
d) Tallying the votes that successfully pass steps two and three.
e) When using multiple Counters, the master Counter collects all previous verified votes. Then checks for repeated votes using the bit commitment, and proceeds with the final tally.

All the Counters also publish the contents of all received submission packages, and the Administrators publish all the blindly signed ballot digests. After this publication the voter, with the information saved during the voting protocol, can verify if his vote was counted. If the vote is not present at the tally he can reclaim presenting anonymously the previously saved vote. In addition, anybody can verify the relation between the total number of votes and the total possible valid votes signed by the Administrators.

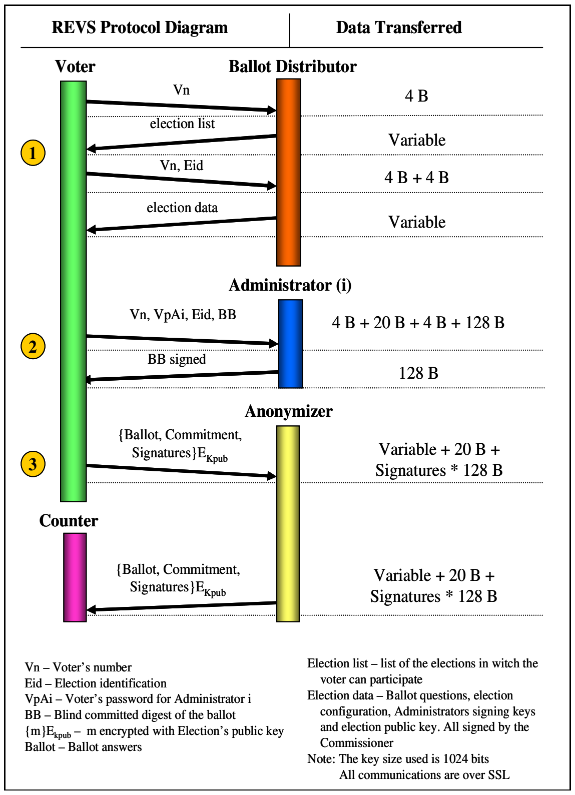
REVS Protocol Diagram
Data Transferred
Voter
Ballot Distributor
Vn
4 B
election list
Variable
1
Vn, Eid
4 B + 4 B
election data
Variable
Administrator (i)
Vn, VpAi, Eid, BB
4 B + 20 B + 4 B + 128 B
2
BB signed
128 B
Anonymizer
{Ballot, Commitment, Signatures}E_{Kpub}
Variable + 20 B + Signatures * 128 B
3
Counter
{Ballot, Commitment, Signatures}E_{Kpub}
Variable + 20 B + Signatures * 128 B
Vn – Voter's number
Eid – Election identification
VpAi – Voter's password for Administrator i
BB – Blind committed digest of the ballot
{m}E_{kpub} – m encrypted with Election's public key
Ballot – Ballot answers
Election list – list of the elections in witch the voter can participate
Election data – Ballot questions, election configuration, Administrators signing keys and election public key. All signed by the Commissioner
Note: The key size used is 1024 bits
All communications are over SSL

Figure 4 - REVS protocol details

4. IMPLEMENTATION

REVS was fully implemented in Java, enabling it to be installed and executed on any computational platform. REVS protocol does not impose any restriction to the ballot format; therefore we decided to define the ballots in XML for more flexibility, see, c.f. Section 4.1. For encryption we used the logi.crypt crypto package (version 1.1.2). We also used a database back-end in REVS servers, namely MySQL (version 3.23.53-max).

4.1. Ballots and Answers

Like all blind signatures based voting protocols, REVS is ballot independent. Taking advantage of this property we have implemented the ballots and answers in XML. This way we bring several others advantages to the system, such as:

- It is easy to build ballots; it can be done in any text editor.
- Anyone who knows a little of XML can easily read the ballots and answers.
- Provides additional interoperability capabilities to the system. For instance it can be easily build an application to analyze graphically the tally of any election or survey.

In Figure 5 it is presented the general XML construction of ballots and answers. A ballot is composed by a description and several groups of questions. A group of questions has a description and several questions. A question is composed by a description, the question it self, and by the possible answers.

```
<!-- General ballot example -->

<ballot electionCode="123">
    <ballotDescription>
        <line>Test Ballot description line one</line>
        <!-- more lines -->
    </ballotDescription>
    <group code="1" description="Group one">
        <!--type tag can also have the values Multiple, OpenS or OpenM->
        <question code="1" type="Single">
             <questionDescription>First Question</questionDescription>
            <answer code="1">First Answer</answer>
            <!-- more answers -->
        </question>
        <!-- more questions -->
    </group>
    <!-- more groups -->
</ballot>
```

```
<!-- General ballot answer example -->

<ballotAnswer electionCode="123">
    <group code="1">
        <!-- Single  type -->
        <question code="1" answer="1"></question>
        <!-- Multiple  type -->
        <question code="2" answer="1-3-7"></question>
        <!– OpenS  type (selecting the open answer)-->
        <question code="3" answer="0">Open answer</question>
        <!– OpenS  type (selecting another answer )-->
        <question code="4" answer="5"></question>
        <!– OpenM  type -->
        <question code="5" answer="0-2-4">Open answer</question>
        <!-- more answers -->
    </group>
    <!-- more groups -->
</ballotAnswer>
```

Figure 5 - XML general ballot and answer

Currently four types of questions are supported: **Single**, the answer must be one and only one of the presented choices; **Multiple**, we can choose any number of choices for our answer; **OpenS** (open single) and **OpenM** (open multiple) types are similar to the Single and Multiple types respectively, but it is also possible to give another answer. The OpenS and OpenM types of question are very useful for surveys.

4.2. System security

In EVOX and EVOX Managed Administrator systems it is possible to run several elections simultaneously. This facility is controlled by distributing the right ballots to voters, but nothing was done to prevent the voters from exchanging ballots, or even stealing ballots from other voters. Thus, it is possible for voters to participate in elections for which they are not authorized to. In REVS every Administrator has an asymmetric key pair for each election, so even if the voters manage to exchange the ballots the signatures will not match and the ballot will be discarded.

The voter's module represents the voter, so it's important that the voter trusts it. In REVS the Commissioner signs the voter's module, ensuring its integrity.

When implementing a secure application, aspects such as intervenient authentication and security in communications must be taken into account. In REVS we use RMI over SSL to provide security in the communications, namely authentication. When establishing a connection with a server the voter uses the server public key to authenticate it within the SSL handshake. The authentication of data exchanged between voters and servers is provided by the SSL transport protocol.

For authenticating voters we used the well-known username/password method. The voter must use a different password for each different Administrator for preventing impersonation. But for keeping the authentication user-friendly we should not force the voter to memorize all passwords. Thus, we designed an algorithm, similar to the HMAC digest algorithm (Krawczyk et al. 1997), for generating all necessary passwords from a small set of secrets.

It uses two secrets (see Figure 6): a strong password (non trivial password, like a large random alphabetical string) and an activation PIN that should be memorized by the voter. The voter introduces these two passwords in the voter's

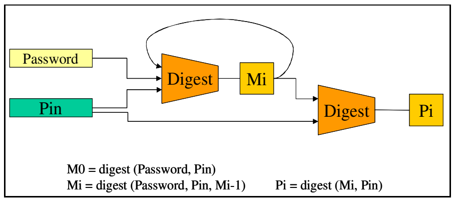
Password
Pin
Digest
Mi
Digest
Pi
M0 = digest (Password, Pin)
Mi = digest (Password, Pin, Mi-1)
Pi = digest (Mi, Pin)

Figure 6 – Password generation

module and it computes all required passwords, one for each Administrator. For being user-friendly both passwords should be chosen by the voter. Another possibility is to give the passwords to the voters and allow them to choose new ones.

The two secrets used to produce Pi password are never disclosed to electoral servers. At the registration phase a voter gives his *N* Pi passwords, in an Unix-like encoding, that will be used to authenticate itself to the Managers. Because the algorithm uses digest functions, which are not invertible, an Administrator *Ai* knowing a voter's password *Pi*, cannot compute any other password Pj, $j \neq i$, known by Administrator *Aj*.

4.3. Election preparation

Before we can start using REVS an election must be prepared. This preparation consists in generating the election key pair, registration of voters and configuration of the servers. The registration of voters consists in defining their identification (username) and collecting their passwords for all Administrators.

In REVS prototype the election key pair is generated by the Commissioner. The public key is distributed in the ballot distribution phase, and the private key is only released, by the Commissioner, after the election voting period ends. This means that the Commissioner together with any Counter can produce intermediate tallies. To prevent this possibility it could be used a threshold election key pair[3] generated by all Counters. At the end of the election period a subset of all Counters would cooperatively decrypt the votes.

The configuration of the servers consists in installing the servers (this includes the generation of theirs public and private keys), and setting up theirs databases with the election and voter's information.

The voter's module can be distributed to the voters in the registration process or can be downloaded later from the election official site or from some Ballot Distribution server. The voter's module comes with the Commissioner public key.

5. EVALUATION OF THE FUNCTIONALITY OF REVS

The evaluation of the functionality of REVS is made under several assumptions. First we clarify those assumptions, and then evaluate REVS considering the characteristics presented in Section 2.1 and also considering its usability and scalability. The assumptions are:

- The cryptographic algorithms used are hard to break. In REVS there are three cryptographic algorithms used: (i) RSA, for producing and checking blind and non-blind signatures and encrypting the keys used to encrypt the submission packages; (ii) Triple-DES, used to encrypt the submission packages; and (iii) SHA-1, for all required digest computations.
- The communications are secure. All communications in REVS use SSL and servers are authenticated with their public keys.
- The servers, voters' computers and communication gateways are not vulnerable to attacks, such as DoS or infection by Trojan horses or viruses.
- The Anonymizers used are honest and work correctly.
- The required number of signatures respects $t > N/2$.
- The voter decides to save his data in secure non-volatile storage in steps 1 and 2 of the voting protocol (see Section 3.1).
- The Commissioner only publishes the election private key after the end of the election.

We now analyze how REVS fulfills each voting requirement (stated in Section 2.2).

[3] In a threshold cryptosystem there are *N* parties sharing the same private key *Kpri*, in such a way that at least *t* parties must collaborate to decrypt *E(Kpub,m)*, the encryption of the arbitrary message *m* with the public key *Kpub*. This kind of system is called (*t,N*)-threshold cryptosystem. Threshold cryptosystems usually include two protocols: (1) a key generation protocol, where all the *N* parties are involved in the generation the shared, private key *Kpri*; and (2) a verifiable decryption protocol, that allows *t* parties to cooperatively decrypt an encrypted message *E(Kpub,m)* in a way that everyone can verify that the decryption was performed correctly. This process should not give anyone the ability to decrypt any other messages.

Accuracy: The vote cannot be altered because that would destroy all Administrators' signatures. A voter can verify if his vote was eliminated from the final tally, examining the list of received votes published by Counters, and can correct this sending his submission package anonymously. The elimination of votes when using several Counters is harder than when using just one Counter because it implies the elimination of the vote from all Counters. Since the signatures can be verified by anyone and are published with the votes, it is impossible for an invalid vote to be part of the final tally. Therefore, all three aspects of accuracy are respected.

Democracy: Each voter can only vote once in each election because $t > N/2$ (a voter cannot obtain two valid votes). All voters can vote as long as t Administrators and one Anonymizer are available. Therefore, the two aspects of democracy are guaranteed as long as the system works.

Privacy: While the assumptions stand, the first part of privacy (neither authorities nor anyone else can link any ballot to the voter who cast it) is guaranteed. The second part (no voter can prove that he voted in a particular way) as in most voting protocols proposed so far, isn't accomplished, allowing the voter to prove his choice.

Verifiability: The final tally can be made by anyone, verifying the signatures on the votes and summing all votes. Each voter can verify if its own vote is correct, using the information saved during the voting protocol, and assumes that the other votes are correct because of the signatures they have.

Collusion Resistance: This characteristic depends on the number of Administrators, N, and required signatures, t. To cast a vote one needs the cooperation of t Administrators (increases as t grows). To prevent a voter from voting $N–t+1$ Administrators must conspire, preventing the voter from obtaining the required t signatures (decreases as t grows). So, it's obvious that it's necessary to make a trade-off between the two parts of this property.

Availability: The system is available as long as there is a minimal set of servers running correctly. The minimal set is actually one Ballot Distributor, t Administrators, and one Anonymizer/Counter pair. The two last properties intrinsically depend on the configuration of the system.

Resume Ability: As explained in Section 3.1, the voter can recover from an interruption in the voting protocol as long as the voter keeps its voting data, i.e. the bit commitment and the blinding factor.

Robustness: REVS is a robust system because achieves the last three properties, as defined in Section 2.2.

5.1. Implementation evaluation

REVS was designed to support large scale elections. In this section we evaluate the prototype of REVS concerning implementation decisions, time consumed in cryptographic functions, both by the voter's module and by Administrators, and amount of data transferred.

As seen before, REVS can run without singular points of failure, therefore avoiding bottlenecks. All the servers have a database back-end and were implemented in a way that, if necessary, a cluster can be easily implemented to improve performance. With this design and implementation considerations we believe that REVS can easily support large scale elections.

To evaluate the prototype performance we have made some tests using a computer with a Pentium III processor, 384 MB of RAM running Windows XP Professional.

Regarding the voter's module we determined that it would take less than half a second to compute 1000 passwords; the blinding process is done in less than 200 ms; and the verification of a blind signatures is done in less than 30 ms. So, when using REVS in a configuration requiring 5 signatures the voter's module would compute all cryptographic functions in about a second.

The Administrator is the other entity that must compute cryptographic functions (signing blinded ballot digests). In our tests we verified that an Administrator takes less than 200 ms to verify the voter identity and sign the blinded ballot (about 15% for the first action and 85% for the second). An Administrator, with the test configuration, can produce 5 signatures per second. For getting better performance results we can use a more powerful computer or deploy the Administrator as a cluster of machines sharing a high-performance database.

Besides servers' performance it's also necessary to analyze the amount of data transferred in the protocol (c.f. Figure 3). Being REVS a ballot independent voting system we must make some assumption before analyzing any data transfers: we assumed that the elections list is 1 KB long; the election data is divided in a fixed part with 2 KB (ballot questions, election configuration and election public key), and a variable part with N * 1024 bits long (public keys of the Administrators running the election), finally we assume that the vote (the ballot answers) is 256 bytes long.

Table 1 – Data transferred in REVS with the voter submitting to one Anonymizer-Counter chain

	Ballot Distribution	Ballot Signing	Ballot Submission	Total (one voter)	Total (one million voters)
N = 3; t = 2	≈ 3.3 KB	≈ 0.5 KB	≈ 1.1 KB	≈ 4.9 KB	≈ 4.9 GB
N = 5; t = 3	≈ 3.4 KB	≈ 0.8 KB	≈ 1.3 KB	≈ 5.5 KB	≈ 5.5 GB
N = 7; t = 4	≈ 3.5 KB	≈ 1.0 KB	≈ 1.5 KB	≈ 6.0 KB	≈ 6.0 GB

From the data presented in Table 1 it is easily seen that REVS offers a good data transfer performance. For instance we can compare the data transferred for each voter with the size of the Google's main page, which is about 16 KB. REVS also provides a good tradeoff between increased security and data transfer, about 0.5 GB for each additional signature required. Note that all calculations do not take into account the additional traffic generated by the communications over SSL.

Another important aspect to observe from Table 1 is that 60% to 75% of all data is relative to the Ballot Distribution phase. Therefore it was a good option to use dedicated servers for this operation.

Regarding the required computation and data transfer aspects, the previously presented data allows us to conclude that our prototype of REVS is efficient and can be used in large scale elections.

5.1.1. The first experiment

The first prototype and experiment of REVS was done in the Instituto Superior Técnico (Technical Univ. of Lisbon) to support elections, namely surveys for evaluating the quality of courses. To this particular scenario, REVS servers were deployed and managed by separate entities, namely central computer services, several departments and students' organizations, in order to reduce collusion. A set of trusted machines were made available to voters, but they could use any other machine to participate in the elections.

The students were able to vote during a period of two weeks. During the survey period it was achieved 100% of availability, even in the presence of a few server and communications failures. This first experiment proved the robustness of REVS.

6. CONCLUSIONS

We have presented REVS, an electronic voting system able to perform well in faulty environments such as the Internet. REVS is a robust electronic voting system based on blind signatures that tolerates failures in communications and servers while maintaining all desired properties of a voting system.

Another important characteristic of REVS is the ballot independency; which facilitates its use in any kind of elections or surveys.

The implementation of REVS was carefully designed for assuring scalability and availability in large-scale elections. In particular REVS was fully implemented in Java, and can use replication to avoid bottlenecks and single points of failure.

Concerning future work, REVS can beneficiate from a more sophisticated anonymity mechanism. The authentication mechanism could also be substituted with one based on public keys, but only after this technology becoming largely available to guarantee the usability of the system. A future use of threshold election keys for concealing submitted votes during elections would also improve the overall security of the system.

REFERENCES

Baudron, O. et al., 2001. Practical Multi-Candidate Election System. *Proc. of the 20th ACM Symp. on Principles of Distributed Computing.* Philadelphia, USA, pp. 274-283.

Benaloh, J., 1987. *Verifiable Secret-Ballot Elections*. Yale University PhD thesis, YALEU/DCS/TR-561.

Benaloh, J. and Fischer, M., 1985. A Robust and Verifiable Cryptographically Secure Election Scheme. *Proc. of 26th IEEE Symp. on the Foundations of Computer Science.* Los Angeles, USA, pp. 372-382.

Benaloh, J. and Tuinstra, D., 1994. Receipt-Free Secret-Ballot Elections (Extended Abstract). *Proc. of 26th ACM Symp. on the Theory of Computing.* Montreal, Canada, pp. 544-553.

Benaloh, J. and Yung, M., 1986. Distributing the power of Government to Enhance the Privacy of Voters. *Proc. of 5th ACM Symp. on Principles of Distributed Computing.* Calgary, Canada, pp. 52-62.

CALTECH-MIT, 2001. Voting-what Is, What Could Be. Available at http://www.vote.caltech.edu/Reports

Chaum, D., 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Comm. of the ACM,* 24(2).

CIVTF, California Internet Voting Task Force, 2000. A Report on the Feasibility of Internet Voting. Available at http://www.ss.ca.gov/executive/ivote

Cramer, R et al., 1996. Multi-Authority Secret-Ballot Elections With Linear Work. *Proc. of Advances in Cryptology – EUROCRYPT '96.* Berlin, Germany, LNCS 1070, pp. 72-83.

Cramer, R. et al., 1997. A Secure and Optimally Efficient Multi-Authority Election Scheme. *European Trans. on Telecommunications,* 8(5), pp. 481-490.

Cranor, L., 2001. Voting After Florida: No Easy Answers. *In Ubiquity,* 1(47).

Cranor, L. and Cytron, R., 1997. Sensus: A Security-Conscious Electronic Polling System for the Internet. *Proc. of the Hawaii International Conference on System Sciences.* Wailea, Hawaii.

DuRette, B., 1999. *Multiple Administrators for Electronic Voting.* MIT Bs.C thesis.

Fujioka, A. et al., 1992. A Practical Secret Voting Scheme for Large Scale Elections. *Proc. of Advances in Cryptology – AUSCRYPT '92*, Queensland, Australia, LNCS 718, pp.244-251.

Herschberg, M., 1997. *Secure Electronic Voting Using the World Wide Web.* MIT Ms.C thesis.

Hirt, M. and Sako, K., 2000. Efficient Receipt-Free Voting Based on Homomorphic Encryption. *Proc. of Advances in Cryptology – EUROCRYPT '00*. Bruges, Belgium, LNCS 1807, pp. 539-556.

IPI, Internet Policy Institute, 2001. Report of the National Workshop on Internet Voting: Issues and Research Agenda. Available at http://www.internetpolicy.org/research/e_voting_report.pdf

Jakobsson, A., 1998. A Practical Mix. *Proc. of Adv. in Cryptology – EUROCRYPT '98,* Espoo, Finland, LNCS 1403.

Krawczyk, H. et al., 1997. HMAC: Keyed-Hashing for Message Authentication. RFC 2104.

Lioy, A. et al., 2000. DNS Security. *Proc. of TERENA Networking Conference*. Lisbon, Portugal, pp. 22-25.

Monteiro, A. et al., 2001. Sistemas de Votação Electrónica. FCT, Lisbon University, TR-01-X. (In portuguese)

Newmann, P., 1993. Security Criteria for Electronic Voting. *Proc. of 16th Nat. Computer Security Conf.* Baltimore, USA.

Ogata, W. et al., 1997. Fault Tolerant Anonymous Channel. *Proc. of Information and Communications Security ICICS '97.* Beijing, China, LNCS 1334, pp.440-444.

Ohkubo, M. et al., 1999. An improvement on a practical secret voting scheme. *Proc. of ISW: International Workshop on Information Security*, LNCS Springer-Verlag.

Okamoto, T., 1997. Receipt-Free Electronic Voting Schemes for Large Scale Elections. *Proc. of Workshop on Security Protocols '97.* Paris, France, LNCS 1361, pp. 25-35.

Park, C. et al., 1993. Efficient Anonymous Channel and All/Nothing Election Scheme. *In Advances in Cryptology – EUROCRYPT '93*. Lofthus, Norway, LNCS 765, pp.248-259.

Rivest. R., 2001. Electronic Voting. *Proc. of Financial Cryptography '01.* Grand Cayman, Cayman Islands.

Rubin, A., 2002.Security Considerations for Remote Electronic Voting Over the Internet. *In Comm. of ACM,* 45 (12).

Sako, K., 1994. Electronic Voting Schemes Allowing Open Objection to the Tally. *Trans. of IEICE,* E77-A(1), pp. 24-30.

Sako, K. and Kilian, J., 1994. Secure Voting Using Compatible Homomorphisms. *Proc. of Advances in Cryptology – CRYPTO '94*. Santa Barbara, USA, LNCS 839, pp. 411-424.

Sako, K. and Kilian, J., 1995. Receipt-Free Mix-Type Voting Scheme – A Practical Solution to the Implementation of a Voting Booth. *Proc. of Advances in Cryptology – EUROCRYPT '95*. Saint-Malo, France, LNCS 921, pp. 393-403.

UK-eDemocracy, 2003. http://www.edemocracy.gov.uk