INTERNET VOTING: IMPROVING RESISTANCE TO MALICIOUS SERVERS IN REVS

Ricardo Lebre
INESC-ID Lisboa
Address
ricardo.lebre@gsd.inesc-id.pt

Rui Joaquim
INESC-ID Lisboa
Address
rui.joaquim@gsd.inesc-id.pt

André Zúquete
IEETA/INESC-ID Lisboa/UA
Address
avz@det.ua.pt

Paulo Ferreira
INESC-ID Lisboa
Address
paulo.ferreira@.inesc-id.pt

ABSTRACT

With the explosive growth and consequent usage of the Internet as a medium to offer new services with increased value, it became possible to develop Internet Voting Systems. So far, several have been proposed but few have been implemented. REVS is an Internet Voting System based on blind signatures designed to tackle some of the real-world problems presented by other systems. The main contribution of our work is to improve the robustness of REVS. This is achieved with a scheme that prevents specific denial of service attacks against protocol participants, which are not easily detected. In particular, we address the problem raised by colluded malicious servers preventing voters from voting and the exhaustion of resources on servers. Then, we present a performance comparison of the solutions proposed against the current REVS protocol.

KEYWORDS

Internet Voting, Blind Signatures, Digital Signatures, Denial-of-Service attacks

1. INTRODUCTION

With the explosive growth of the Internet and the improvement on communications it became possible to use this infrastructure to offer new services with increased value. One of such services is Internet Voting, which presents several advantages over the traditional method. For example, it allows the reduction of election's operational costs and may prevent high abstention rates by avoiding the traditional movement to the booth. However, in order to be adopted, voters must be fully confident with this type of systems. Such confidence can only be achieved if they are resistant to faults, either exploited from outside or from inside the voting protocol. It is possible to use auditing systems to detect attacks external to the protocol, and it should be possible to mitigate protocol vulnerabilities. In particular, it should be possible to eliminate completely, or at least to keep under reasonable margins, the collusion of protocol entities willing to prevent authorized voters from voting. On the other hand, the vote submission system should be capable to filter out submitted garbage. Otherwise the system could easily be overflowed by attackers.

Several voting protocols have been proposed [Chaum88, Fujioka92, Cranor97, Herschberg97, Okamoto97, DuRette99, Ohkubo99, Joaquim03], however only few systems have been implemented [Cranor97, Herschberg97, DuRette99, Joaquim03]. REVS [Joaquim03] is an Internet Voting System designed to tackle some of the real-world problems presented by other systems (e.g. communication failures, server failures). It is based on MIT's EVOX [Herschberg97] and EVOX Managed Administrators [DuRette99] but, unlike its predecessor, it presents no single point of failure; in addition, in the presence of a communication failure it allows the voter to resume the process from the state saved in a previous checkpoint.

The main contribution of our work is to improve the robustness of REVS by tackling attacks that cannot be detected by auditing systems while the election stands. This is achieved with some protocol modifications that prevent denial of service attacks against participants. In particular, our solution prevents colluded Administrator servers, which are responsible for validating votes, to prevent voters from voting. We also present a solution to prevent the exhaustion of resources of other protocol servers, the Counters, which are responsible for collecting votes. REVS was not able to check the validity of submitted votes until the end of the election, allowing Counters to be overflowed with garbage. We modified the way votes are produced to enable Counters to filter out garbage submitted during elections.

The remainder of this paper is organized as follows. In Section 2 we present the architecture of REVS highlighting the properties guaranteed and the problems detected. In Section 3 we present our solutions to the problems described before. Section 4 shows some performance results and compares them to the ones presented by REVS. In Section 5 we present a critical overview of current systems. The paper ends with some conclusions in Section 6.

2. REVS SYSTEM

2.1 Properties

In [Cranor97] Cranor presented the four core properties that any voting system should have:

1. **Accuracy:** (i) It is not possible for a vote to be altered; (ii) It is not possible for a validated vote to be eliminated from the final tally; (iii) It is not possible for an invalid vote to be counted in the final tally.
2. **Democracy:** (i) It permits only eligible voters to vote; (ii) It ensures that eligible voters only vote once.
3. **Privacy:** (i) Neither authorities nor anyone else can link any ballot to the voter who casts it; (ii) No voter can prove that he voted in a particular way.
4. **Verifiability:** (i) Anyone can independently verify that all votes have been counted correctly.

Another two non-core properties are additionally guaranteed by REVS protocol:

5. **Collusion Resistance:** (i) No electoral entity or group of entities can work in a conspiracy to introduce votes; (ii) No electoral entity or group of entities can work in a conspiracy to prevent others from voting.
6. **Availability**: (i) The system works properly as long as the poll stands; (ii) Any voter can participate from the beginning to the end of the poll.

Since the great majority of the existing systems are only theoretical they don't take into account operational aspects. So, REVS [Joaquim03] has introduced a new functional property:

7. **Resume Ability:** The system allows any voter who had interrupted his voting process to resume or restart it as long as the poll stands.

The first four properties are guaranteed by the blind signature scheme [Chaum82] used by REVS. The solution proposed by [Joaquim03] for collusion resistance and availability imposes a trade-off that we will explain latter.

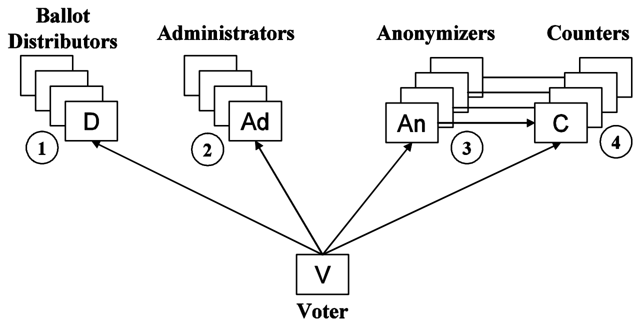

Figure 1 - REVS current protocol.

2.2 Protocol

Figure 1 presents REVS' protocol steps, which are explained in more detail bellow.

1. **Ballot distribution:** The Voter contacts a Ballot Distributor to get a ballot for a given election. The Ballot Distributor returns him the requested ballot, the election's public key and the election's operational configuration (e.g. servers' keys, servers' locations), all signed by the election's Commissioner. This is done in two steps: the voter contacts a Ballot Distributor and provides his ID to receive the list of elections in which he can participate. Then the voter chooses the election and requests a ballot for it from a Ballot Distributor.

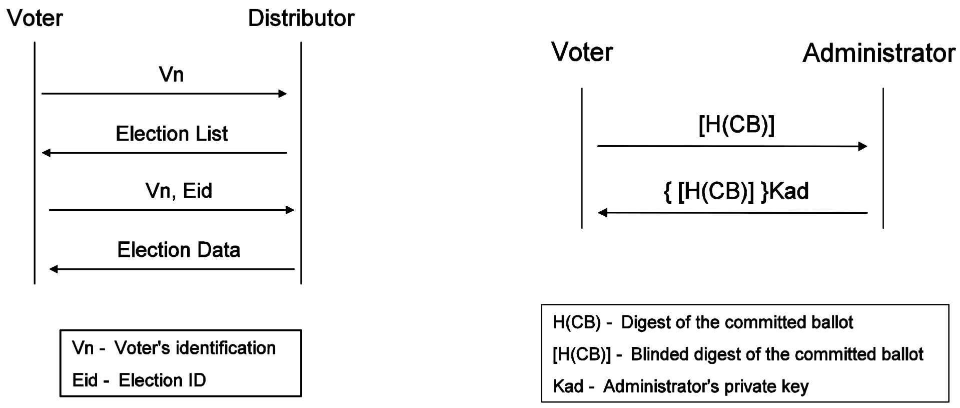
Voter
Distributor
Vn
Election List
Vn, Eid
Election Data
Vn - Voter's identification
Eid - Election ID
Voter
Administrator
[H(CB)]
{ [H(CB)] }Kad
H(CB) - Digest of the committed ballot
[H(CB)] - Blinded digest of the committed ballot
Kad - Administrator's private key

Figure 2 - REVS Ballot distribution and Ballot Signing.

2. **Ballot signing:** After expressing his will on the ballot, the voter commits to the ballot with a random bit string and generates a digest of the committed ballot. Then it generates a random blinding factor, applies it to the digest and sends the result to a subset of the Administrators for signing. The voter's module can save 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 may affect the voter's privacy. An Administrator, after receiving a request for signing, verifies if it had already signed a blind digest for the requesting voter and given election. If not, he signs it and saves that signature; if he had signed before, it returns the previously saved data, i.e. the digest's blind signature. After receiving the signature, the voter removes the blinding factor and verifies its correctness using the original ballot digest and the Administrator's public key. This process is repeated until a required number of signatures *t* is collected. Note that, because of the blinding factor, the Administrators cannot link the signed ballots to the signatures they provided for (blind) ballot digests.

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 mobile storage. This is an optional step, because it helps improving the resume ability property, but again may affects privacy. Then he submits this package to a Counter through an Anonymizer, encrypted with a hybrid cryptosystem using a random symmetric session key and the election's public key, 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 his destination. Although this introduces some fault tolerance, it introduces also a new problem. A badly intentioned voter could try to make a denial of service attack to any Counter, as we will see latter.

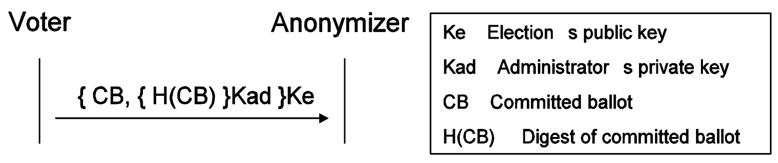

Figure 3 - REVS Ballot submission.

4. **Counting Phase:** After the end of the election the Commissioner publishes the election's private key. Then the counting process is performed by the Counters and involves the following steps: (i) Decipher the submission packages with the election's private key; (ii) Verify that all required *t* signatures from the Administrators are present; (iii) Remove repeated votes, which are the ones with the same bit commitment; (iv) Tally the remaining votes; (v) When using multiple Counters, the master Counter collects all previous verified votes and proceeds with the final tally. All the Counters publish the contents of all received submission packages and the Administrators publish all the blindly signed ballots.

After this publication the voter can verify if his vote was counted. If the vote isn't present at the tally he can reclaim, presenting anonymously the previously saved vote. Also, anyone may verify the relation between the total number of votes in the final tally and the number of votes signed by the Administrators.

2.3 Problems Detected

Being an Internet voting system, REVS is subjected to two distinct classes of attacks: **external** and **internal**. External attacks are the ones that by targeting others than protocol entities may corrupt any of the protocol properties (e.g. OS or network denial of service, IIS buffer overflow). On the other hand, internal attacks target directly the protocol, trying to take advantage of any protocol vulnerability. External attacks, although more common, may be detected using monitoring mechanisms such as IDSs (Intrusion Detection Systems). With respect to internal attacks the problem is more serious because there is no way to report or even to detect them.

We have detected two vulnerabilities in the current REVS protocol that can lead to internal attacks. This section describes those vulnerabilities and the problems they may create.

2.3.1 Interference of malicious Administrators

A **defective** Administrator is one that fails to follow the protocol. That may happen either because (i) it is a fail-stop server or (ii) it is a server that deliberately fails to follow the protocol. We call the first a **faulty** Administrator and the second a **malicious** Administrator.

The threshold parameter *t* in the REVS protocol controls two different properties of the voting process: (i) the degree of fault tolerance and (ii) the collusion of Administrators necessary to impersonate voters. Increasing *t* decreases fault tolerance, because the number of allowed faulty Administrators is N-t. On the other hand, increasing *t* increases the tolerance to the impersonation of voters by colluded malicious Administrators, because they must get *t* signatures for producing a valid, impersonated vote. For example, if we have 11 Administrators and $t = 9$, the protocol tolerates at most 2 simultaneous, faulty Administrators and at least 9 servers must cooperate to generate a valid vote for an impersonated voter.

However, a malicious Administrator can behave like a faulty one without notice, leading to denial of service scenarios. In the ballot signing phase of the REVS protocol an Administrator can answer the voter with garbage instead of the correct blind signature. The voter can detect the problem by checking the returned signature, but has no way to prove to a third party that the Administrator has misbehaved, maliciously or not. So, an Administrator can maliciously and selectively deny its service to selected voters, while for the affected voters it looks like a faulty server. This means that at least N-t+1 malicious Administrators need to be colluded to prevent a voter from voting by returning false signatures. Recalling the example above, only 3 malicious Administrators need to be colluded to prevent voters from voting. Increasing *t* makes it worse, because fewer colluded Administrators are needed.

Therefore, in order to prevent malicious Administrators to act like faulty ones, it is necessary to allow independent third parties to control the correctness of Administrators' replies,. We are assuming that in REVS implementation there are supervision authorities that may inspect all interactions between voters and Administrators in order to detect other simpler denial of service attempts, such as not responding to voters or returning false error messages.

2.3.2 Invalid vote submission

In REVS votes are submitted anonymously and encrypted to protocols servers. Valid votes may even be repeatedly submitted for fault tolerance reasons. Note that such votes won't be counted in duplicate because of the bit commitment that makes the vote unique.

However, Anonymizers and Counters cannot check the correctness of submissions before the end of the election, when the decryption key is disclosed. Thus, these servers cannot prevent anyone, authorized voter or not, to submit garbage. Consequently, Counters may overflow with useless data that can only be dropped at the end of the election, eventually leading to denial of service scenarios.

One possibility for reducing this risk is to allow submissions only from authorized voters. However, this access control must be done carefully, as Anonymizers and Counters cannot authenticate voters, otherwise they could link voters to votes. Hence, Anonymizers/Counters should be able to check the correctness of votes submitted anonymously. This can be achieved by adding to votes an anonymous authentication credential that only authorized voters can get.

3. PROPOSED SOLUTIONS

The general solution for the two problems previously presented is to authenticate data with digital signatures. Voters sign the data they send to Administrators and Anonymizers/Counters. This way, Administrators cannot forge voter's requests and Anonymizers/Counters can check if the data was sent by an authorized voter.

The critical aspect now is the management of the key pairs for signing and validating voter's requests. Validating keys must be certificated to ensure their correctness. Additionally, the certification must ensure anonymity when voters interact with Anonymyzers/Counters.

We decided to extend the functionality of Administrators in order to certificate voters' public keys. Such certification is different from the standard certification provided by ordinary PKIs (Public Key Infrastructures). Namely, (i) public keys must be certified by at least *k* Administrators and (ii) the certification protocol may involve the exchange of normal or blind signatures.

The new threshold parameter *k* is, in theory, independent from *t*. It allows controlling separately the tolerance to faulty servers and the tolerance to malicious servers attempting to denial service to voters.

3.1 Double Signing Authentication

As previously referred, we modified the blind signing phase of REVS in order to authenticate voter's requests using digital signatures. Thus, Administrators cannot forge voter's requests in order to reply with garbage, preventing them from voting. Figure 4 describes the process that is explained in more detail below.

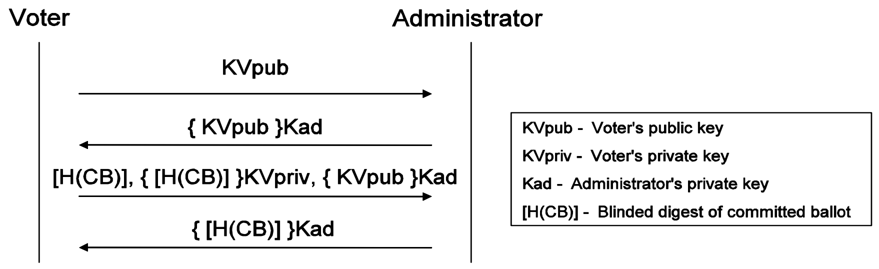
Voter
Administrator
KVpub
{ KVpub }Kad
[H(CB)], { [H(CB)] }KVpriv, { KVpub }Kad
{ [H(CB)] }Kad
KVpub - Voter's public key
KVpriv - Voter's private key
Kad - Administrator's private key
[H(CB)] - Blinded digest of committed ballot

Figure 4 - Double Signing Authentication

Each voter generates a pair of asymmetric keys {*KVpub*, *KVpriv*}. Then he certifies his key *KVpub* with the signature of *k* Administrators. This process can be repeated as many times as he wants with any *KVpub*. After acquiring the required signatures, the protocol is similar to REVS. The voter sends the blinded digest of the committed ballot for signing, but now signed with *KVpriv*. In the same message goes the *KVpub* certificate, i.e. the key *KVpub* and its *k* Administrators' signatures.

Each Administrator only signs a voter's blinded digest if all its *k* signatures are correct, and any supervision authority can verify the correctness of such decision. Likewise, an incorrect reply from an Administrator can be detected by a supervision authority.

This is repeated until all the required *t* Administrators signatures are obtained. The Administrators store the complete voters' requests as a proof for future claims. Note that in both interactions the voter authenticates himself to each Administrator contacted with the current username/password scheme of REVS.

3.1.1 Solution Analysis

With our solution, we are only modifying the ballot signing part of the protocol. There is no influence on the accuracy property since it's also required the validation of votes using Administrators' signatures. The authentication process is maintained, so democracy remains unchanged. The blind signature mechanism that assures verifiability and the privacy properties was not compromised by our changes on the protocol. The resume ability is guaranteed because: (i) The key certification process is idempotent; (ii) A voter resuming the voting process may use different *KVpub* for each signed vote.

Resistance to Defective Servers The collusion resistance depends on the two threshold parameters *t* and *k*. The first parameter controls the degree of collusion necessary for the impersonation of voters by Administrators. The second parameter controls the degree of collusion of Administrators for preventing voters from voting.

The lowest value of *t* and *k* controls the system's tolerance against colluded malicious Administrators attempting to interfere with the voting process. On the contrary, the highest value of *t* and *k* controls the system's tolerance against **faulty** Administrators. If *t=k*, then either value controls the system's resistance against **defective** Administrators.

3.2 Vote Validation

We modified the ballot submission phase of REVS in order to authenticate submission packages using also digital signatures. This way, only authorized voters that may generate such signatures, can interact successfully with Anonymizers and Counters.

We propose a solution that is similar to the one previously used for signing voter's requests sent to Administrators (described in Section 3.1). A submission package is signed by another voter's private key and the corresponding public key is certified by *k* Administrator's blind signatures. The blind signatures prevent Anonymizers and Counters from cooperating with Administrators in order to link votes to voters. Since only authorized voters can get the required blind certificate, only authorized voters may successfully interact with Anonymizer/Counters.

The key pair used for authenticating votes **must** be different from any of the key pairs used for authenticating voter's requests to Administrators. Otherwise Administrators could easily link votes to voters by checking the results publicly presented by Counters at the end of elections. For optimizing the protocol, both certificates, this blind certificate and the one referred in Section 3.1, can be obtained simultaneously.

Note that this solution does not prevent authorized voters from submitting correctly signed random data. But this behavior can be detected keeping some extra state in Counters and using auditing systems in Administrators. If Counters store blind certificates in order to filter out different submissions using the same certificate, then malicious voters are forced to require several blind certificates from Administrators. This way they reveal their potentially malicious attempts and can easily be identified by election authorities, because the interaction with Administrators requires identification of voters. Another possibility is to allow Administrators, under a proper supervision, to slow down blind certifications for voters requiring them very often.

4. RESULTS AND ANALYSIS

Since REVS was designed to work on a real-world environment, such as the Internet, it was given a great deal of importance to its performance. Typically the bottleneck on the Internet is on the client side, since it is where the slowest connections and fewer processing capability are. In REVS case, its design allows the easy addition of new servers to handle processing limitations. Most time-consuming operations are done offline (e.g. key generation), so we opted to evaluate the communication penalties imposed by our solution.

4.1 Current REVS

Being REVS a ballot independent voting system we must make some assumptions before analyzing the data transferred. In our tests, the election list is 375B long, the election data is divided in a fixed part with 2.5KB (ballot choices, election configuration and election public key) and a variable part with N*128 Bytes long (signing keys for Administrators running the election). Finally the ballot (answers) is 32 Bytes long. These values were retrieved from a real voting experience at Instituto Superior Técnico – Technical University of Lisbon.

Table 1- Data transferred in REVS (approximately).

	Ballot Distribution	Ballot Signing	Ballot Submission	Total (1 voter)
N=5; t=4	2.3KB	1.1KB	1.1KB	4.5KB
N=7; t=6	2.5KB	1.7KB	1.6KB	5.8KB
N=9; t=8	2.8KB	2.2KB	2.1KB	7.1KB

Looking at Table 1 we can see that the data transferred in the Ballot Distribution phase has a small growth as *N* grows. This is because of the necessary operational data transferred concerning the new Administrators. The Ballot Signing and Ballot submission phases have a more relevant growth since the number of signatures transferred grows linearly with *t*. The total data transferred is not relevant considering today's Internet connections.

4.2 REVS with Double Signing Authentication

Our changes affected only the ballot signing part of the protocol with the new ballot signing mechanism. So, all other data exchanges are kept the same. Table 2 presents the new values for the data transfer with double signing authentication.

Table 2 - Data transferred with Double Signing Authentication (approximately).

	Ballot Distribution	Ballot Signing	Ballot Submission	Total (1 voter)
N=5; t=4; k=3	2.3KB	3.1KB	1.1KB	6.5KB
N=7; t=6; k=4	2.5KB	5.2KB	1.6KB	9.3KB
N=9; t=8; k=5	2.8KB	8.0KB	2.1KB	12.9KB

Table 2 shows that we have solved the problem posed by malicious servers with an acceptable performance penalty. The values for the new ballot signing phase are of the same order of magnitude as in the current REVS.

4.3 REVS with Double Signing and Vote Validation

Considering that the new key validation interaction is part of the ballot submission phase, all other data exchanges are kept the same. Table 3 presents the data transferred in REVS with both Double Signing and Vote Validation.

Table 3 - Data transferred with Double Signing and Vote Validation (approximately).

	Ballot Distribution	Ballot Signing	Ballot Submission	Total (1 voter)
N=5; t=4; k=3	2.3KB	3.1KB	1.9KB	7.3KB
N=7; t=6; k=4	2.5KB	5.2KB	2.7KB	10.2KB
N=9; t=8; k=5	2.8KB	8.0KB	3.5KB	14.3KB

Again, the solution proposed imposes no severe penalty on the Ballot Submission phase.

Concluding, the Double Signing and the Vote Validation phases increase slightly the data transferred between voters and protocol servers. Both phases impose also a delay of k message exchanges. However, because these exchanges may be done in parallel the delay is not significant.

5. RELATED WORK

The reference protocol using blind signatures, known as the FOO protocol, was proposed in 1992 by Fujioka, Okamoto and Ohta [Fujioka92]. In this protocol the voters cannot vote and walk away; they must wait until the end of the election to submit their vote-opening keys, therefore bringing no convenience to voters. There have been some implementations of protocols closely based on FOO, such as the Sensus [Cranor97] and Evox [Herschberg97]. In both proposals the FOO protocol was changed to allow voters to vote and walk away, however in both protocols it is quite easy to reveal intermediate results. To solve this problem an improvement on FOO was proposed [Ohkubo99]. The improvement consists in using a *(t,N)* - threshold cryptosystem. In this protocol the voter encrypts his signed ballot and sends it anonymously to a bulletin board. There are *N* counters, so if *N-t+1* of them are honest the results are only available at the end of the election.

One of the unsolved problems of the previously presented protocols is robustness. In 1999, DuRette proposed the Evox Managed Administrators [DuRette99], an improvement of Evox by the addition of more Administrators. This improvement addressed the problems of collusion resistance and availability. However, it still presents singular points of failure and uses a weak voter's authentication mechanism that simplifies the impersonation of voters by Administrators, therefore leading to a much less robust system than the author claims.

The REVS protocol [Joaquim03] is an improvement of DuRette's work. It uses a new authentication mechanism that avoids the easy impersonation of voters and a new architecture that presents no singular points of failure. These changes improved considerably the robustness of DuRette's work. But, as shown previously in this paper, REVS also suffers from some problems concerning collusion resistance and invalid vote submission. Furthermore, the current REVS implementation still uses a single key pair – the election key pair, generated and managed by the Commissioner – for hiding votes during the election. But the system can easily be improved to use a *(t,N)* - threshold cryptosystem for generating and using the election's key pair. Note that this is mainly an implementation issue, since the votes will likewise continue to be hidden with the election's public key and disclosed with the election's private key.

The first problem addressed by our work was to prevent colluded Administrator servers to stop voters from voting without being detected. In all types of voting systems (i.e. based on blind signatures, homomorphic encryption or mix-nets) if a PKI is used to manage voter's asymmetric key pairs then voters are able to sign server requests and it is impossible for servers to forge voter's requests without being detected. The great majority of Internet Voting Systems assume (implicitly or explicitly) the use of PKIs [Cranor97, Fujioka92, Ohkubo99, Park93, Sako95a, Cramer97, Sako95b]. However, using PKIs is not convenient for voters because they have to get and store securely a long-term private key. To our knowledge we have been the first ones to handle this problem using asymmetric key pairs but without requiring a PKI. Instead, our solution produces on-demand, per voter, temporary key pairs. The certification of the public components of those key pairs is part of the voting protocol and the authentication of voters requiring that certification uses the existing REVS username/password authentication mechanism. Thus, voters can sign their requests to prevent false repudiation by electoral servers and do not have to manage a long-term asymmetric key pair, just a password.

In Internet Voting Systems based on blind signatures voters must submit votes anonymously in order to keep their vote intentions secret. If votes are submitted anonymously we have shown that can lead to denial-of-service scenarios. Blind signature systems usually rely on anonymous channels or mix-nets to provide anonymous submissions and completely ignore this denial-of-service problem. We have proposed a scheme based on token authentication that enables electoral servers to mitigate the risk. Again, to our knowledge our solution is unique.

6. CONCLUSIONS

In this paper, we analyzed the REVS protocol and highlighted two important denial of service vulnerabilities. Namely, the possible collusion between malicious Administrators in order to prevent voters from voting and the invalid vote submission in order to exhaust Counter's storage capacity. We presented a solution that uses digital signatures to guarantee both non-repudiation and authentication on votes. This solution improves the resistance to malicious servers and also authenticates anonymously votes submitted to scrutiny. We also presented a performance analysis of the improved protocol comparing the data transferred against the current REVS protocol. Performance results obtained with real data dimensions and several configurations for the operational environment showed that we have improved substantially REVS robustness with acceptable performance penalties.

REFERENCES

[Chaum82] Chaum, D., 1982. Blind signatures for untraceable payments. *Proceedings of Advances in Cryptology – CRYPTO '82.* New York, pp. 199-203.

[Chaum88] Chaum, D., 1982. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. *Proceedings of Advances in Cryptology– Eurocrypt'88.* pp. 177-182.

[Cramer97] Cramer, R. et al, 1997. A Secure and Optimally Efficient Multi-Authority Election Scheme. *Proceedings of Advances in Cryptology – EUROCRYPT '97.* pp. 103-118.

[Cranor97] Cranor, L. and Cytron, R., 1997. Sensus: A security-conscious electronic polling system for the Internet. *Proceedings of the Hawaii International Conference on System Sciences.* Wailea, Hawaii.

[DuRette99] DuRette, B., 1999. Multiple administrators for electronic voting. *Bachelor's thesis.* Massachusetts Institute of Technology.

[Fujioka92] Fujioka, A. et al, 1992. A practical secret voting scheme for large scale elections. *Proceedings of Advances in Cryptology - Auscrypt'92.* pp. 244-251.

[Herschberg97]Herschberg, M., 1997. Secure electronic voting over the world wide web. *Master's thesis.* Massachusetts Institute of Technology.

[Hirt00] Hirt, M. and Sako, K., 2000. Efficient receipt-free voting based on homomorphic encryption. *Proceedings of Advances in Cryptology– EUROCRYPT'00.* pp. 539-556.

[Joaquim03] Joaquim, R. et al, 2003. REVS - A Robust Electronic Voting System. *Proceedings of IADIS International Conference e-Society 2003.* Lisbon, Portugal. pp. 95-103.

[Ohkubo99] Ohkubo, 1999. An improvement on a practical secret voting scheme. *Proceedings of ISW: International Workshop on Information Security.*

[Okamoto97] Okamoto, T., 1997. Receipt-free electronic voting schemes for large scale elections. *Proceedings of 5th International Computer Security Conference.* pp. 25-35.

[Park93] Park, C. et al, 1999. Efficient Anonymous Channel and All/Nothing Election Scheme. *Proceedings of Advances in Cryptology – EUROCRYPT'93.* pp. 248-259.

[Sako95a] Sako, K., 1995. An improved universally verifiable mix-type voting System.

[Sako95b] Sako, K. and Kilian, J., 1995. Receipt-Free Mix-Type Voting Scheme: A Practical Solution to the Implementation of a Voting Booth. *Proceedings of Advances in Cryptology – EUROCRYPT '95.* pp. 393-403.