# Defeating Malicious Servers in a Blind Signatures Based Voting System*

Sébastien Canard, Matthieu Gaud, and Jacques Traoré

France Telecom R&D
42, rue des Coutures, BP6243, 14066 Caen Cedex, France
{sebastien.canard, matthieu.gaud, jacques.traore}@francetelecom.com

**Abstract.** In this paper, we present two failures in the blind signatures based voting system Votopia [2] which has been tested during the last World Soccer Cup. We then propose a fix which relies on *fair blind signatures*. The resulting scheme is practical, satisfies the fundamental needs of security in electronic voting, including *public verifiability*, and compares favorably with other like systems in terms of computational cost. As an illustration, our variant of Votopia has been successfully trialed during the French referendum on the European Constitution in May 2005.

## 1 Introduction

A blind signature scheme is a protocol allowing to get a signature from a signer such that the signer's view of the protocol cannot be linked to the resulting message-signature pair.
Blind signatures can be used in applications where anonymity of a message is required such as untraceable electronic cash or electronic voting. One of the standard electronic voting scheme using blind signatures was proposed by Fujioka, Ohta and Okamoto (FOO for short) at Auscrypt'92. Unfortunately, their scheme suffers from several major drawbacks. The main one is that all voters have to participate to the ballot counting process. This means that each voter must stay until all other voters complete the casting stage, which makes the scheme unpractical for real life. In [3], Ohkubo et al. show how to avoid this inconvenience by proposing a variant of FOO's voting scheme with a simple mix-net that allows voters to "vote and go": they need not to make any action after voting. Votopia [2] is a practical implementation of this system and has been tested during the 2002 FIFA World Cup to select the Most Valuable Players.
In this paper, we first focus on the security of Votopia. We describe two failures where the first *mix server* in the system can affect the result of the election in an unnoticeable way. We then show how to repair Votopia [2]. The resulting scheme remains practical for large scale elections, allows voters to "vote and go" and satisfies the fundamental needs of security in electronic voting, including public verifiability (that is, anyone can check the validity of the whole voting process).

* Work partially supported by the French Ministry of Research RNRT Project "CRYPTO++".

*Appeared in* G. Di Crescenzo, A. D. Rubin (Eds.): Financial Cryptography 2006, LNCS 4107,pp. 148–153, 2006.

The key component that makes our voting protocol convenient for voters and publicly verifiable is a (threshold) fair blind signature scheme, a variant of a blind signature scheme that has been introduced in by Stadler et al. at *Eurocrypt'95*. In this variant, the signer can, with the help of a trusted authority (or a quorum of authorities), either identify from the transcript of a signing session the resulting signature (*signature tracing*) or link a message-signature pair to the corresponding signing session (*session tracing*).

## 2 Protocol Failures in Votopia

Five basic entities are involved in the Votopia voting system [2]: the voters ($\mathcal{V}_i$ will denote the voter $i$), an Admin Server $\mathcal{AS}$, the mix servers or *mix-net* $\mathcal{M}$ ($\mathcal{M}_i$ will denote the mix server $i$), the talliers $\mathcal{T}$ ($\mathcal{T}_j$ will denote the tallier $j$) and a bulletin board $\mathcal{BB}$ which, as usual, is publicly readable and which every participant can write to (into his own section) but nobody can delete from. The role of these different entities will be clarified in the sequel. The system makes use of the following cryptographic primitives: a threshold encryption scheme, a digital signature scheme, a blind signature scheme and a *simple* mix-net (i.e. not *universally verifiable*). Any secure implementation of these primitives suits this system. We will therefore use generic notation to describe such primitives: $E_{\mathcal{T}}$ and $D_{\mathcal{T}}$ will denote respectively $\mathcal{T}$'s threshold encryption and decryption schemes whereas $E_{\mathcal{M}}$ will denote $\mathcal{M}$'s "encryption scheme". $B$ and $UB$ will denote respectively the blinding and unblinding functions of the blind signature scheme. In the sequel, we will assume that each eligible voter has a pair of keys of an agreed signature scheme and that the corresponding public key has been certified by $\mathcal{AS}$. $S_i$ (respectively $S_{\mathcal{AS}}$) will denote $\mathcal{V}_i$'s signing function (respectively $\mathcal{AS}$'s signing function), $C_i$ the certificate of the corresponding public key and $\mathcal{V}_i$'s identifier is denoted by $Id_i$.

**Voting Stage.**

1. $\mathcal{V}_i$ selects the vote $v_i$ of his choice and encrypts $v_i$ with $\mathcal{T}$'s public key of the threshold encryption scheme as $x_i = E_{\mathcal{T}}(v_i)$. $\mathcal{V}_i$ blinds $x_i$ as $e_i = B(x_i, r_i)$, where $r_i$ is a randomly chosen blinding factor. $\mathcal{V}_i$ signs $e_i$ as $s_i = S_i(e_i)$ and sends $(Id_i, C_i, e_i, s_i)$ to $\mathcal{AS}$.
2. $\mathcal{AS}$ checks that the signature $s_i$ is valid and that it comes from a registered voter who has not already submitted a blind ballot. If all these verifications are valid (the protocol is aborted otherwise), then $\mathcal{AS}$ signs $e_i$ as $d_i = S_{\mathcal{AS}}(e_i)$ and sends $d_i$ to $\mathcal{V}_i$. At the end of the voting phase, $\mathcal{AS}$ announces the number of voters receiving $\mathcal{AS}$'s signature, and publishes the final list $L_{\mathcal{AS}}$ of $(Id_i, C_i, e_i, s_i)$.
3. $\mathcal{V}_i$ retrieves the desired signature $y_i$ of ballot $x_i$ by $y_i = UB(d_i, r_i)$. $\mathcal{V}_i$ encrypts $(x_i, y_i)$ with the "encryption key" of the mix-net as $c_i = E_{\mathcal{M}}(x_i, y_i)$. $\mathcal{V}_i$ signs $c_i$ as $\sigma_i = S_i(c_i)$ and sends $(Id_i, C_i, c_i, \sigma_i)$ to $\mathcal{BB}$.
4. $\mathcal{BB}$ checks the signature of the posted message and checks that $Id_i$ appears in $L_{\mathcal{AS}}$. $\mathcal{BB}$ publishes the list $L_{\mathcal{BB}}$ of $(Id_i, C_i, c_i, \sigma_i)$.

*Appeared in* G. Di Crescenzo, A. D. Rubin (Eds.): Financial Cryptography 2006, LNCS 4107,pp. 148–153, 2006.

**Counting Stage.**

1. $\mathcal{M}$ decrypts the list of $c_i$ and outputs the list $L$ of $(x_i, y_i)$ in random order.
2. $\mathcal{T}$ checks the signature $y_i$ of $x_i$. If the verification fails, $\mathcal{T}$ claims that $y_i$ is not a valid signature on $x_i$ by publishing $(x_i, y_i)$. If more than $t$ (the threshold) talliers claim about the same $(x_i, y_i)$, the mix servers have to reveal the corresponding $c_i$ and prove in zero-knowledge that $(x_i, y_i)$ is the correct result of decryption of $c_i$ (we call *back-tracing* such procedure). Each tallier checks the proofs issued by each mix server. If the checks fail, the mix server that issued a wrong proof is disqualified. If all proofs are valid, it means that the voter casts an invalid vote. Thus, the vote is excluded from further steps of the counting stage. After excluding the invalid results of mix processing, $\mathcal{T}$ (cooperatively) decrypts ballots $x_i$ and retrieves vote $v_i$ as $v_i = D_{\mathcal{T}}(x_i)$. $\mathcal{T}$ then publishes the result of the election.

In [2], Kim et al. emphasize on the fact that their system satisfies the "vote and go" property. Here, we will show that if we really let the voters "vote and go" then their system doesn't satisfy the *accuracy* requirement (that is the impossibility to alter a cast ballot). More precisely, we will show that the first mix server (and only this mix) can modify the result of the election in an unnoticeable way. Indeed, since the ballots sent to $\mathcal{BB}$ are signed by the voters (see step 3 of the voting stage), this first mix can easily recognize or substitute the ballots that come from voters who are members of a political party different from its own. We distinguish two cases: the case where the number $n$ of (encrypted) ballots sent to $\mathcal{BB}$ is smaller than the number $N$ of voters who interact with $\mathcal{AS}$ (which could correspond to the case where some voters obtained their ballots from $\mathcal{AS}$ and finally decided not to cast it) and the case where $n$ is equal to $N$.
**1)** $n < N$**.** Suppose that the first mix server, denoted by $\mathcal{M}_1$, has $m \leqslant N - n$ accomplices. $\mathcal{M}_1$ can ask its $m$ accomplices to execute step 1 and step 2 of the voting stage (and consequently to obtain valid signed ballots from $\mathcal{AS}$) but not the following steps (in other words, they will not send their ballots to $\mathcal{BB}$). $\mathcal{M}_1$ can then replace $m$ valid ballots of targeted voters by the $m$ ballots of its accomplices. As the latter ballots are valid (they contain $\mathcal{AS}$'s signature) there will be no anomaly in the list $L$. The back-tracing procedure will consequently not be executed and no one will detect the subterfuge. This fraud will thus allow $\mathcal{M}_1$ and its accomplices to affect the result of the election.
**2)** $n = N$**.** As in the previous case, we suppose that $\mathcal{M}_1$, has $m$ $(m < N)$ accomplices. $\mathcal{M}_1$ asks its accomplices to obtain valid signed ballots from $\mathcal{AS}$. But this time, the accomplices will not abstain from casting their ballots. Rather, they will send dummy ballots to $\mathcal{BB}$, while keeping the valid ballots provided by $\mathcal{AS}$ for future use. $\mathcal{M}_1$ can then replace $m$ valid ballots of its choice by the $m$ ballots of its accomplices. Obviously, the dummy ballots will be detected and discarded in the counting stage. (Note that the back-tracing procedure will not detect $\mathcal{M}_1$'s substitution). So $\mathcal{T}$ will decrypt the remaining ballots (after having discarded the invalid ones) and tallies the final result. Again, this subterfuge will allow $\mathcal{M}_1$ and its accomplices to modify the result of the election.

*Appeared in* G. Di Crescenzo, A. D. Rubin (Eds.): Financial Cryptography 2006, LNCS 4107,pp. 148–153, 2006.

Another issue is what should be done in case of a "suicide attack" : suppose that the first mix server substitutes an invalid ballot for the valid one posted by a targeted voter. The substitution will be detected after the decryption of the full list of ballots in a provisional tally, and the cheating mix-server identified. But what should be done now? Either the excluded vote is added in the final tally, in which case it will be exposed as the difference between the provisional and final tally or else the vote cannot be counted!

## 3 Our Electronic Voting Scheme

Our aim in this section is to repair Votopia. Before describing our main proposal, we envision several approaches that seem possible.

**Possible approaches:** to detect the frauds described above, we could require the active participation of the voters in the counting stage to verify that their votes were counted (i.e., that their pairs $(x_i, y_i)$ appear in the list $L$). However, this would clearly contradict the "vote and go" property and it would be impractical for large scale elections to force all voters to check the result. Furthermore, although each voter can check that his or her vote has been correctly counted, no voter can be assured that all ballots have been tallied correctly. Such solution would only provide *individual verifiability* and not *public verifiability*. Moreover, it is not clear how a voter can complain to the scrutineers or officials of the election without taking the risk of compromising the privacy of his or her vote.
Another option is to have the first mix server provide a proof of correct mixing. But this doesn't solve the problem anymore. Indeed, if the first mix server colludes with the second one, as well as with malicious voters, they will still be able to change valid votes in an unnoticeable way (in a similar manner at what was done by the first mix server in section 2). This remark remains valid even if we assume that the first $k$ mix servers (among the $l$ servers) provide a proof of correct mixing (with $k \leqslant l - 2$). In this case a collusion of malicious voters and the $k + 1$ first servers will still be able to manipulate votes.
A radical solution to overcome such shortcomings is therefore to require that all mix servers prove that they have correctly mixed the set of encrypted ballots. In other words a solution would be to use a *universally verifiable mix-net*. But if verifiable mixes are used anyway, there is no need for blind signatures at all! So we cannot assume that the mixes are verifiable when considering the use of blind signatures. Still, the security must be ensured.
We try to solve this seemingly paradox in the next section by using a threshold fair blind signature scheme and two *simple* mix-nets (care must be taken however on the choice of these mix-nets), though *robust* against server failures (which means that when a server is unavailable, it is possible to replace it by another one). However, we would like to emphasize that in most cases, the mix servers will not have to prove that the mixing was done correctly. Our solution can then be seen as an *optimistic mix-type voting scheme* [1]. As we will see, a cheating mix server will always be detected. Therefore, if the penalties for

*Appeared in* G. Di Crescenzo, A. D. Rubin (Eds.): Financial Cryptography 2006, LNCS 4107,pp. 148–153, 2006.

cheating are severe this will preclude any attempt.
In the sequel, we will denote by $\mathcal{M}$ and $\mathcal{TM}$ the two sets of mix-net servers (where $\mathcal{TM}_j$ will denote the mix server $j$) and by $E_{\mathcal{M}}$ and $E_{\mathcal{TM}}$ their respective encryption scheme. The "private key" of $\mathcal{M}$ will be denoted by $SK_{\mathcal{M}}$ and the one of $\mathcal{TM}$ by $SK_{\mathcal{TM}}$. We will denote by $\mathcal{J}$ the revocation authorities of the threshold fair blind signature scheme ($FBSS$ for short) and by $REV_{\mathcal{J}}$ the corresponding signature tracing mechanism.

We saw that the problem of Votopia comes from the fact that some votes can easily be removed or substituted by other valid ballots (by valid we mean a correctly formed ballot that has been signed by the Admin Server). We want to repair Votopia by making everybody sure that, before tallying the result of the election, the ballot box doesn't contain any fraudulent ballots. We consider that fraudulent votes can be deployed either by mix servers and/or voters and from various types of actions: adding, removing or replacing a valid ballot by another one. Owing to space limitations, we will only give a sketch of our fix.

**Voting Stage.** This stage is similar to the one of Votopia (see section 2). The main differences are that $x_i = E_{\mathcal{TM}}(v_i)$ instead of $x_i = E_{\mathcal{T}}(v_i)$ and that $y_i$ is a (threshold) fair blind signature of $x_i$ rather than a conventional blind signature. (We would also like to stress that, as usual, the voter should prove that he knows the plaintext of $c_i$ in order to prevent *vote-duplication*). At closing time of the poll, the two lists $L_{\mathcal{BB}}$ and $L_{\mathcal{AS}}$ are compared. If $Id_i$ appears in $L_{\mathcal{BB}}$ but not in $L_{\mathcal{AS}}$, which means that $\mathcal{V}_i$ has not requested a fair blind signature, then $(Id_i, C_i, c_i, \sigma_i)$ is removed from $L_{\mathcal{BB}}$. If a voter $\mathcal{V}_i$ has requested a fair blind signature to $\mathcal{AS}$ but has not submitted (deliberately or owing to a network failure for example) a ballot to $\mathcal{BB}$ (which means that there is an entry $(Id_i, C_i, e_i, s_i)$ in $L_{\mathcal{AS}}$ but not a corresponding one $(Id_i, C_i, c_i, \sigma_i)$ in $L_{\mathcal{BB}}$), then the anonymity of $e_i$ is revoked. The value $f_i = REV_{\mathcal{J}}(e_i)$ is then put in a black list $RL$ so that everybody can recognize the message-signature pair $(x_i, y_i)$ later. Note that by using $REV_{\mathcal{J}}$, we do not compromise the privacy of the vote $v_i$ of $\mathcal{V}_i$: depending on the fair blind signature scheme used, the revocation authorities can at most obtain $y_i$ but not $x_i$!

**Counting Stage.** $\mathcal{M}$ decrypts the list of $c_i$ and outputs the list $L$ of $(x_i, y_i)$ in random order.
**Case 1:** if all pairs $(x_i, y_i)$ are found to be correct (i.e., no pair $(x_i, y_i)$ contains an invalid signature $y_i$, "belongs" to $RL$ or is duplicated) then $SK_{\mathcal{TM}}$ is revealed (which means that all the mix servers $\mathcal{TM}_i$ have to reveal their own private keys). The ballots $x_i$ are decrypted (using $SK_{\mathcal{TM}}$). $\mathcal{TM}$ outputs the corresponding votes $v_i$ and then publishes the result of the election.
**Case 2:** otherwise for each incorrect pair $(x_i, y_i)$, the back tracing algorithm (see section 2, step 2 of the counting phase of Votopia) is used to determine whether this anomaly comes from a mix server or a voter.
***Case 2.1:*** if a mix server cannot prove that it has correctly decrypted and permuted such a suspicious pair $(x_i, y_i)$, it is then disqualified. $SK_{\mathcal{M}}$ is revealed,

*Appeared in* G. Di Crescenzo, A. D. Rubin (Eds.): Financial Cryptography 2006, LNCS 4107,pp. 148–153, 2006.

which means that all the mix servers $\mathcal{M}_i$ have to reveal their own private keys (as we use a robust mix-net, it is then possible to retrieve the key of any malicious mix server even if the latter refuses to cooperate). The list of $c_i$ is decrypted using $SK_{\mathcal{M}}$ and a new list $L$ containing all the decrypted $(x_i, y_i)$ is sent to $\mathcal{TM}$. $\mathcal{TM}$ decrypts and randomly permutes the list of $x_i$ (but this time, the mix servers have to prove that they correctly decrypt and mix their inputs, which is costly), outputs the corresponding votes $v_i$ in random order and publishes the result of the election (we thus solve the issue of *suicide attacks*).
***Case 2.2:*** if no mix server cheats this means that the fraud comes from a voter. The misbehaving voter is identified (thanks to the back tracing algorithm) and the anonymity of the blind ballot $e_i$ is revoked. The pair $(x_i, y_i)$ is removed from $L$ and the revoked value $f_i = REV_{\mathcal{J}}(e_i)$ is put on the black list $RL$. We then redo the counting stage with the new lists $L$ and $RL$.

At the end of the protocol, the lists $L_{\mathcal{AS}}$, $L_{\mathcal{BB}}$, $RL$, $L_0$, $L$ and every step of the counting phase (as well as the intermediate lists outputted by the mix-servers and the back-tracing procedures) are made public. Therefore, anybody can check that only invalid ballots are discarded and that the outcome of the election is consistent with the valid cast ballots (public verifiability), provided however that all the voting entities will not collude. Indeed, if the mix-servers and the admin servers collude, they can produce as many valid ballots as they wish and substitute the ballots of legitimate voters with the ones they fraudulently produced.

## 4 Conclusion

In this paper, we have shown some weaknesses of Votopia [2] and proposed a heuristic method, relying on fair blind signatures, to defeat our attacks. Our solution, which can be seen as an optimistic mix-type voting system based on fair blind signatures, provides, almost for free, both individual and public verifiability so that everyone can be convinced of the correctness of the voting result. In terms of efficiency, it appears that our solution, when all goes well (which is a priori always the case), is better than existing mix-net based solutions. We therefore believe that fair blind signatures could represent a more promising alternative for secure on-line voting than ordinary blind signatures and an appealing direction for future work.

## References


1. P. Golle, S. Zhong, D. Boneh, M. Jakobsson and A. Juels, Optimistic Mixing for Exit-Polls, *Asiacrypt'02*, volume 2501 of LNCS, pages 451-465, 2002.
2. K. Kim, J. Kim, B. Lee and G. Ahn, Experimental Design of Worldwide Internet Voting System using PKI, *SSGRR2001*, 2001.
3. M. Ohkubo, F. Miura, M. Abe, A. Fujioka and T. Okamoto, An Improvement on a Practical Secret Voting Scheme, *Information Security'99*, volume 1729 of LNCS, pages 225-234, 1999.



*Appeared in* G. Di Crescenzo, A. D. Rubin (Eds.): Financial Cryptography 2006, LNCS 4107,pp. 148–153, 2006.