A Receipt-Free i-Voting System Based on Blind Signatures and Anonymous IDs

Charles F. de Barros[1], **Diego F. Pimenta**[2]

[1]Departamento de Ciência da Computação (DCOMP)
Universidade Federal de São João Del Rei (UFSJ)
São João Del Rei – MG – Brazil

[2]Universidade Federal de São João Del Rei (UFSJ)
São João Del Rei – MG – Brazil

charlesbarros@ufsj.edu.br, diegof.pimenta@outlook.com.br

Abstract. *In this paper, we propose an i-voting scheme based on blind signatures, with a mechanism to prevent voters from generating vote proofs. The scheme combines blind signatures of the ballots with anonymous IDs also blindly signed by the election authority. After the election is closed, the list of ballots and corresponding signatures is published by the election authority, allowing any interested party to check the integrity of the result.*

1. Introduction

Blind signatures have been applied to the design of e-voting systems, such as [Cranor and Cytron 1996], [Davenport and Woodard 2000], [DuRene 1999], [Fujioka et al. 1993], [Ohkubo et al. 1999] and [Radwin 1995]. A typical scenario is described in [Ibrahim et al. 2003] and involves an electoral committee, which possesses a certified public key and, therefore, is able to issue digital signatures. In order to cast a vote, the voter sends a *blinded* ballot to the electoral committee, who blindly signs it and sends it back to the voter. The voter then removes the blinding factor, thus obtaining a valid signature for his vote. Both the vote and its signature are sent to the tallier, a server responsible for storing and counting the votes. The vote is accepted if and only if the signature is valid. After the election is closed, a list of votes with the respective digital signatures is published, allowing any interested party to verify the integrity of the results.

The problem with this approach is that, moments before the voter casts his ballot, he possesses its digital signature, which can be used as a *receipt*. Hence, he can sell his vote as follows: before sending his vote, he sends the corresponding signature to his favorite candidate, who is capable of checking, after the list of votes and corresponding signatures is published, that the value he received matches the signature of the desired vote.

In this paper, we present a variant of the voting protocol proposed in [Ibrahim et al. 2003], with a mechanism to prevent voters from generating vote proofs. Essentially, our proposed scheme prevents the voter from learning the digital signature of his ballot before casting his vote.

1.1. Roadmap

This paper is organized as follows: in Section 2, we present a brief review on the concept of digital signatures and blind signatures. In Section 3, we describe the i-voting system proposed by [Ibrahim et al. 2003]. Our proposal is presented in Section 4 and, finally, in Section 5 we present our conclusions and final remarks.

2. Digital Signatures

A digital signature is a mechanism for demonstrating the authenticity of a digital document, using a public key scheme. In order to digitally sign a document D, the holder of a secret key sk applies it to the document, generating a value σ, called a signature of D. In order to check the authenticity of the document, one must apply the corresponding public key pk to the signature, verifying that the result is linked to D in such a way that could only be achieved by the holder of the secret key.

Put into more formal terms, a signature is obtained by applying an algorithm `Sign`, which takes as input a document D and a secret key sk and outputs a digital signature σ. In order to verify the validity of the signature, one applies an algorithm `Verify`, which takes as input a document D, a signature σ and the corresponding public key pk and returns `True` if the signature is valid, and `False` otherwise.

The algorithm `Verify` must return `True` if, and only if the signature was generated with the correct secret key. Hence, we expect that only the holder of that secret key is able to generate valid signatures, which means that a signature that passes verification is a clue that the corresponding document is authentic.

2.1. The RSA Signature Scheme

The RSA cryptosystem [Rivest et al. 1978] was the first practical public-key cryptosystem. The idea of using a different key for encryption and decryption had been developed by Diffie and Hellman in their seminal paper [Diffie and Hellman 1976].

An RSA key consists of a modulus $N = pq$, where p and q are large prime numbers, and two integers e and d such that

$$ed \equiv 1 \pmod{(p-1)(q-1)}.$$

The number e is the public exponent, while d is the secret exponent. Hence, an RSA secret key is the pair $\langle d, N \rangle$, while the corresponding public key is $\langle e, N \rangle$.

In order to sign a document m, encoded as an integer modulo N, Alice computes

$$\sigma \equiv m^d \pmod{N}.$$

Now she sends both the document and the signature σ to Bob, who can check, using Alice's public key $\langle e, N \rangle$, that

$$\sigma^e \equiv m \pmod{N}.$$

If the above holds, Bob accepts the document. Otherwise, he rejects. As a matter of fact, it is a more recommended practice to sign not the document, but a cryptographic hash

of it. In the previous example, assume that Alice and Bob agree on a cryptographically secure hash function H. To sign the document, Alice computes

$$\sigma \equiv H(m)^d \pmod{N}.$$

She sends the pair (m, σ) to Bob, who recalculates the hash of m and checks whether

$$\sigma^e \equiv H(m) \pmod{N}.$$

The use of the hash has both the advantages of limiting the size of the signature (since the hash has a fixed length) and preventing some attacks, which could give an adversary the ability to produce valid document-signature pairs, even without knowing Alice's secret key.

2.2. Blind Signatures

A blind signature is simply a signature in which the signer does not know the contents of the document which is being signed. The concept was developed Chaum in [Chaum 1983]. It may seem counterintuitive that someone would agree on signing a document without knowing its contents. But imagine, for instance, that Alice wishes to send Bob a digitally signed classified document, but she does not have a certified public key. Hence, she asks a trusted *signing authority* to digitally sign her document. Obviously, since the document is classified, she does not intend to reveal its contents to the signing authority.

Remark that, prior to issuing a blind signature for Alice, the signing authority must gain some assurance about Alice's identity. Hence, there must be some kind of authentication mechanism between Alice and the signing authority. Otherwise, anyone could impersonate Alice by sending a digitally signed document to Bob. In other words, the signing authority is blind with respect to the contents of the signed document, but not to the person who is requesting the blind signature.

Blind signatures are quite simple using the RSA scheme. Let $\langle e, N \rangle$ be the certified public key of the signing authority, and d the corresponding secret exponent. Alice has a document m, which she wants to digitally sign. Since this document is classified, she adds a *blinding factor* to it, generating the *blinded document*

$$m' \equiv r^e H(m) \pmod{N},$$

where H is a cryptographically secure hash function, and r is a randomly chosen integer modulo N such that $\gcd(r, N) = 1$. Now, Alice sends m' to the signing authority, who signs it to obtain the blind signature

$$\sigma' \equiv m'^d \pmod{N} \equiv rH(m)^d \pmod{N}.$$

The signing authority sends back to Alice the blind signature σ'. Since she knows the value of r, she removes the blinding factor r to obtain

$$\sigma \equiv \sigma' r^{-1} \equiv H(m)^d \pmod{N}.$$

Now she has a valid signature of the document m, which was issued by a signing authority who does not know the content of m. Bob can check the authenticity of the document by verifying that

$$\sigma^e \equiv H(m) \pmod{N}.$$

One could argue that simply sending the hash of m to the signing authority would be sufficient to blind the document. However, blind signatures must satisfy a condition of *unlinkability*, which means that the signing authority cannot be able to relate the value that it signs to any existing signed document, even if it is later revealed.

3. An i-Voting System Based on Blind Signatures

Blind signatures have an interesting application to electronic voting systems, allowing a voter to submit a *blinded ballot* to the electoral authority, which blindly signs the vote without breaking its secrecy.

In this section, we briefly describe the scheme presented in [Ibrahim et al. 2003], upon which ours is based. The scheme has four servers: an Administrator, a Registrator, a Validator and a Tallier. Prior to the elections, a voter logs into the Registrator server. If the voter is eligible, he receives an RSA key pair generated by the system. The public key is stored in the voters database, and the corresponding secret key must be stored by the voter in a safe place. Voting procedures follow the steps below:

1. At the election day, the voter sends his identification to the Validator;
2. The Validator checks whether the voter is eligible to vote. If so, it sends the voter a ballot with a unique ID;
3. The voter casts his ballot, which is then blinded and sent to the Validator;
4. The Validator blindly signs the ballot, sending the blind signature back to the voter;
5. The voter removes the blinding factor, checks the integrity of the ballot and sends both the ballot and its signature to the Tallier;
6. The Tallier checks the integrity of the ballot using the Validator's public key and sends the voter a confirmation code, signed with the Tallier's secret key.

After the election is closed, the collected ballots and the corresponding signatures are published. Hence, anyone can check the integrity of any published ballot by verifying the digital signature using the Validator's public key.

As remarked by [Kucharczyk 2010], the main drawback of this protocol is the voter's ability to produce a vote proof. In fact, after he receives the blind signature of his ballot from the Validator, he is able to remove the blinding factor, obtaining the digital signature of his ballot, which may be used as a receipt.

4. Our Contribution: Using a Second Blinding Factor

In order to prevent voters from being able to produce vote proofs, we propose a modification on the described voting scheme, which consists of adding a second blinding factor to the ballot. This blinding factor is unknown to the voter, hence he never learns the signature of his own vote.

We also propose the use of only two servers: an Authentication Server, responsible for authenticating voters and signing ballots, and a Voting Server, responsible for collecting and counting the ballots. There is no communication between these two servers. The

voter interacts with these servers through an app, which can be installed in his personal computer or mobile device. All communications are encrypted with session keys. In this paper, we consider RSA signatures are employed, but the scheme can make use of any other signature system that allows blind signatures.

We remark that i-voting systems are subject to coercion by design, because the voter casts his vote from a totally uncontrolled environment, such as his home or work. A coercer may simply observe how the voter casts his vote. However, it is probably infeasible for a coercer to observe a large number of voters while they cast their votes. Thus, a voting scheme which allows the voter to produce vote proofs is vulnerable to large-scale coercion and/or vote selling, since the coercer does not need to be present in the exact time the voter casts his ballot. Instead, he may require a large number of voters to send him (by email, for example) a vote proof (a receipt), confirming that the voter cast his ballot according to the coercer's will. Hence, it is crucial for the voting scheme to be receipt-free.

In the voting scheme we propose, election is divided into four phases, which occur at separate periods of time. At the end of each phase, the election authority publishes a list containing the partial result of that stage, allowing voters to check the integrity of the electoral process.

4.1. First Phase: Election Preparation

Two RSA key pairs are generated: $(\langle N_1, d_1\rangle, \langle N_1, e_1\rangle)$ for the Authentication Server and $(\langle N_2, d_2\rangle, \langle N_2, e_2\rangle)$ for the Voting Server. The election authority issues a certificate for each of these public keys, and the corresponding secret keys are securely stored in each server.

Prior to the election, all eligible voters are registered in a database stored in the Authentication Server. After downloading and installing the election app, the voter creates an account by choosing a user name, associated to his identity in the database, and a strong password. Alternatively, two factor authentication may be employed. After being installed, the app automatically downloads the public keys of the election servers.

Candidates are registered by the election authority in a database, which is also stored in the Authentication Server. This database will be accessed later to generate ballots for the election.

At a public ceremony, the election authority computes the digital signatures of the two databases, which are published together with a list of all eligible voters and candidates. No sensitive voter data, such as identification numbers, are published.

4.2. Second Phase: Ballot and Anonymous ID Signature Request

In the second stage, the voter can choose the option **Generate Anonymous ID** in the app. A random number id and a random blinding factor f are generated by the app, which computes

$$x \equiv f^{e_1} H(id) \pmod{N_1},$$

where H is a cryptographically secure hash function.

The voter logs into the Authentication Server, using his previously defined user

name and password, and chooses the option **Request Ballot and Anonymous ID Signature** in the app.

If that voter has already requested a ballot and anonymous ID signature, the app shows a warning and prevents the voter from sending another request. Otherwise, the value x is sent to the Authentication Server, which generates a blind signature

$$y \equiv x^{d_1} \equiv fH(id)^{d_1} \pmod{N_1}$$

and sends it back to the voter, together with a digital ballot containing all the election data. The app removes the blinding factor by computing

$$z \equiv yf^{-1} \equiv H(id)^{d_1} \pmod{N_1},$$

which is the digital signature of the anonymous ID id. The pair (id, z) is locally stored by the app, encrypted with a local key. The Authentication Server marks that voter with a tag, indicating that he has already requested a ballot and an anonymous ID signature. After the second phase is over, the election authority publishes a list of all voters that have requested a ballot and anonymous ID signature.

If the voter has made the request, but his name is not on the list, he may open a dispute with the election authority. After proving that he had made a ballot request, the voter will have the opportunity to make a new request, thus obtaining his ballot and a signature for a new anonymous ID. After all disputes are resolved, an updated list of voters who requested ballots and anonymous IDs signatures is published.

4.3. Third Phase: Second Blinding Factor Request

In the third election stage, the voter logs into the Voting Server by sending the pair (id, z) obtained in the previous phase. The server checks the signature, generates a random blinding factor k and hides it with the Authentication Server's public key, obtaining the value

$$a \equiv k^{e_1} \pmod{N_1},$$

which is sent to the voter. The Voting Server temporarily stores the pair $(H(id), k)$ while it waits for the voter to cast his ballot. After a certain period of time, the election authority publishes a list of the hashes of all anonymous IDs that made a second blinding factor request. Each voter can check if his ID is on the list. Once again, if a voter has made a request but his ID is not on the list, he may open a dispute. After disputes are resolved, an updated list is published and the third phase is officially closed.

4.4. Fourth Phase: Casting and Signing the Ballot

During the fourth election stage, the voter accesses the app and casts his ballot, which is encoded as an integer b. After confirmation, the election app generates a blinding factor r and computes a blinded ballot

$$u \equiv r^{e_1} aH(b) \pmod{N_1}.$$

He logs into the Authentication Server and chooses the option **Request Ballot Signature**. If a signature has already been requested by that voter, the app shows a warning and

prevents him from making a new request. Otherwise, the value u is sent to the server, which blindly signs it by computing

$$s \equiv u^{d_1} \equiv rkH(b)^{d_1} \pmod{N_1}.$$

This value is sent back to the voter, who partially unblinds the ballot, revealing

$$t \equiv sr^{-1} \equiv kH(b)^{d_1} \pmod{N_1}.$$

The app checks whether

$$t^{e_1} \equiv aH(b) \pmod{N_1}$$

and, if this equivalence holds, it sends the triplet (id, b, t) to the Voting Server, which computes $H(id)$, fetches the corresponding value of k and computes

$$\sigma \equiv tk^{-1} \equiv H(b)^{d_1} \pmod{N_1},$$

the final ballot signature. The Voting Server then checks whether

$$\sigma^{e_1} \equiv H(b) \pmod{N_1}.$$

If this is the case, it stores the pair (b, σ), discards the values of id, t and k, generates a confirmation code c and sends the pair

$$(c, c^{d_2} \bmod N_2)$$

to the voter. These two values are temporarily stored by the Voting Server. Now, the voter must log into the Authentication Server and send his confirmation code, together with its signature. The Authentication Server signs the confirmation code using his own secret key and sends the signature to the voter, who must now send this value to the Voting Server. After checking the confirmation code signature, the Voting Server includes the voter's ballot in the final tally. After that, the confirmation codes and their signatures are discarded by the Voting Server.

The election authority publishes a list of all voters who made a ballot signature request to the Authentication Server and their corresponding voting confirmation codes, signed by both the Voting Server and the Authentication Server. Any disputes will be resolved by the election authority, who publishes an updated list after the dispute resolution period is closed.

In the last stage, the final tally is published by the election authority, together with the list of ballots and corresponding digital signatures. Vote secrecy is preserved because the list does not contain any voter identification, and is not in the same order as the list of anonymous IDs previously published in the third phase.

Receipt-freeness comes from the fact that no voter is able to produce a vote proof. In fact, all he learns during the voting process are the values of k^e, s and t. Neither of these values allow him to retrieve his own ballot signature σ, because he does not know the second blinding factor k, which was generated by the Voting Server and discarded after the ballot was cast.

Finally, the scheme has universal verifiability, because any interested party is able to verify the integrity of the election result. If any ballot is modified, the signature won't match and the modification will be detected. If any ballot is erased or not included in the final tally, the total number of published ballots won't match the number of confirmation codes. If any confirmation code is missing, the corresponding voter is able to detect it.

5. Conclusion

We proposed an i-voting scheme based on blind signatures and anonymous IDs. The scheme is a modification of an existing i-voting system, preventing voters from generating vote proofs. Hence, the presented voting scheme has the property of receipt-freeness. This property is achieved by adding to the ballot a second blinding factor, generated by one of the election servers, preventing the voter from learning the digital signature of his vote. This signature is only disclosed after the ballot is sent to the voting server, hence the voter cannot use it as a receipt.

The proposed scheme achieves universal verifiability, as anyone is capable of checking the integrity of the election result, by verifying the digital signatures of each ballot published by the electoral authority. The implementation of the proposed scheme is currently in progress, and the main goal of this research is to assess its efficiency and security in a real-world scenario.

References

Chaum, D. (1983). Blind signatures for untraceable payments. In Chaum, D., Rivest, R. L., and Sherman, A. T., editors, *Advances in Cryptology*, pages 199–203, Boston, MA. Springer US.

Cranor, L. and Cytron, R. (1996). Design and implementation of a practical security-conscious electronic pollind system. Technical report, Washington University.

Davenport, Ben, A. N. and Woodard, J. (2000). Creating a secure digital voting protocol for campus elections. http:www.princeton.edu/usgvotehechnicallpaper.html.

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

DuRene, B. (1999). Multiple administrators for electronic voting. http://lltheory.Ics.mut.edu/čis/voting/voting.html.

Fujioka, A., Okamoto, T., and Ohta, K. (1993). A practical secret voting scheme for large scale elections. In Seberry, J. and Zheng, Y., editors, *Advances in Cryptology — AUSCRYPT '92*, pages 244–251, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ibrahim, S., Kamat, M., Salleh, M., and Aziz, S. R. A. (2003). Secure e-voting with blind signature. In *4th National Conference of Telecommunication Technology, 2003. NCTT 2003 Proceedings.*, pages 193–197.

Kucharczyk, M. (2010). Blind signatures in electronic voting systems. In Kwiecień, A., Gaj, P., and Stera, P., editors, *Computer Networks*, pages 349–358, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ohkubo, M., Miura, F., Abe, M., Fujioka, A., and Okamoto, T. (1999). An improvement on a practical secret voting scheme. In *Information Security*, pages 225–234, Berlin, Heidelberg. Springer Berlin Heidelberg.

Radwin, M. J. (1995). An untraceable, universally verifiable voting scheme. Seminar In Cryptology.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126.