A coercion-resistant online voting protocol based on blind signatures and fake credentials

Vítor Rezende Silva[1], **Charles F. de Barros**[2]

[1]Curso de Graduação em Ciência da Computação
Universidade Federal de São João del-Rei
São João del-Rei – MG – Brazil

[2]Departamento de Ciência da Computação
Universidade Federal de São João del-Rei
São João del-Rei – MG – Brazil

vitorrez2002@aluno.ufsj.edu.br, charlesbarros@ufsj.edu.br

Abstract. *In this paper, we present an initial proposal for a practical implementation of an online voting system based on blind signatures, to provide both ballot secrecy and integrity, together with universal verifiability. Moreover, the proposed system offers coercion-resistance by means of a credential scheme based on fake passwords, that the voter can use when under coercion. The idea is that fake passwords seem indistinguishable from valid passwords, and by allowing the voter to use an arbitrary number of fake passwords, no adversary is capable of determining if a coerced voter is casting a valid ballot or a fake ballot.*

1. Introduction

Despite the convenience it can bring to voters, online voting faces a series of issues regarding security and reliability. The main reason for this is the lack of control over the voting environment, which may be a voter's personal computer or mobile device. Ballots transmitted over the Internet could be intercepted, tampered or simply lost due to a communication failure. Infection of the voter's device by viruses could compromise ballot secrecy and integrity. Moreover, without a proper isolated environment like a voting booth, the voter is exposed to coercion, although some proposals aim to solve this problem [Araujo et al. 2018, Clarkson et al. 2008, Juels et al. 2005] by allowing the voters to use fake credentials whenever they are under coercion. However, as demonstrated by [Neto et al. 2018], the usability of credentials is limited, and incorrectly managing them could undermine coercion-resistance. The work of [de Sá et al. 2020] offers a viable solution for this issue.

Verifiability is another important aspect of online voting. Systems like Helios [Adida 2008] offer this property by means of zero-knowledge proofs [Chaum and Pedersen 1993], while [Ibrahim et al. 2003] presents a solution based on blind digital signatures [Chaum 1982]. The idea is that the integrity of each ballot can be directly and easily verified by checking the validity of a digital signature, issued by a trusted authority. The use of a blind signature is intended to preserve ballot secrecy.

This paper presents a fusion of these two ideas: offering coercion-resistance by means of a simple credential system based on passwords (inspired by systems like CIVIS), and simple verifiability by checking a digital (blindly issued) signature of the ballots. The

core idea consists of giving voters the opportunity to define, during registration, a valid user password to be used during a normal voting scenario, and allowing them to use any random password to cast fake ballots, whenever they are under coercion. During the voting procedure, if the voter is under coercion, she may authenticate using a random password, which shall be indistinguishable from the valid password to the adversary. The server, after recognizing the fake password, provides to the voter a fake credential. In case the voter provides the valid user password, the server returns a valid credential. This credential (fake or valid) is appended to the ballot and published on a bulletin board. Only the ballots published with valid credentials are included in the final tally.

1.1. Notations and Theoretical Review

Throughout this paper, an RSA key shall be denoted by a triple (d, e, N), where d is a secret exponent, e is a public exponent and N is the modulus. The pair (e, N) shall be referred to as a public RSA key. A cryptographically secure hash function shall be denoted by H.

The core idea in this paper is that every cast ballot must be digitally signed by the election authority. However, to preserve ballot secrecy, the signature must be blind. A blind signature allows a signing entity (in this case, the election authority) to issue a valid signature for a document whose contents remain unknown. It can be achieved, for example, with the RSA signature algorithm. Assume that the owner of a document m wishes to obtain a valid signature of m, but the signing authority cannot view the contents of m. Also assume that the signing authority has a public RSA key (e, N) and a secret signature key (d, N). The owner of the document chooses a random *blinding factor* r and computes

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

where $\mathrm{GCD}(r, N) = 1$. The value of m' is sent to the signing authority, together with some confirmation about the identity of the document's owner. The signing authority issues a blind signature given by

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

The value of σ' is sent back to the owner of the document, who removes the blinding factor in order to obtain the signature of m, given by

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

Note that, because r is chosen at random, the blinded message m' also appears to be random to the signing authority, revealing no information about the contents of m.

2. Main contributions of this paper

This project is inspired by an article published in 2003 [Ibrahim et al. 2003], with some minor changes in the infrastructure. In the 2003 paper, the application was developed for a web environment, while this project is thought for a mobile application, although it could be adapted to a web environment as well. Moreover, in the original article, the voter receives a key pair, which is stored in a diskette. In this project, the voter has no need to store any secret information. A prototype of the mobile application is currently

in the initial stages of development, using the Kivy library for Python. With this library we are able to create an user graphic interface that can be easily deployed on any mobile platform, as well as in a desktop environment if needed.

The main structure of the project is based upon four server applications, each one with its main functionalities: registrar, administrator, tallier and validator. The servers are implemented using the socket library in python. The registrar is the server application responsible for registering and then validating each voter during the election setting phase. The administrator is responsible for defining the election parameters and registering candidates. The validator is intended to check the eligibility of voters, as well as issuing and signing the ballots. The tallier receives the ballots, checks their validity and computes the election result. All the four server applications run simultaneously. We are able to do this by using the threading library in python.

The novelty of this work, that is not addressed in the 2003 paper, is the inclusion of a coercion-resistance mechanism, based on fake passwords. During registration, the voter defines a valid login password, but whenever he is under coercion, he may use a fake password (which shall seem indistinguishable from the valid password) to generate a fake ballot (also indistinguishable from a valid ballot).

3. Election Stages

The entire election process can be separated in three main stages: setting, voting and tallying. At each step, the election authority publishes relevant information to the election bulletin board, allowing voters and any interested party to audit the whole voting process.

3.1. Election Setting

The first step consists of creating the election and setting its basic parameters, including the election unique identifier, the public key of the election authority and the list of valid credentials.

1. The administrator creates the election, defining its unique identifier, date and time, and generates an RSA key (d_a, e_a, N_a). It also generates a set of random valid credentials $V = \{v_1, v_2, \cdots, v_m\}$, where m is at least equal to the number of voters. Each credential consists of an alphanumeric code.
2. The election authority commits to the valid credentials as follows: for each v_i, a random salt r_i is chosen and the commitment $c_i = H(v_i||r_i)$ is computed, where H is a cryptographic hash function.
3. The list of commitments $c_1, \cdots, c_m$ is signed and published in the election bulletin board. The values of v_i and the respective r_i are kept secret.
4. Voters and candidates are also registered during this phase. The voter provides his personal info and chooses a username and a login password p_l. Some sort of in person registering procedure may be necessary, so that the voter is able to define his login information without interference of an adversary. After registration, the voters and candidates information is stored in the election database.

After the election is created, voters must download the election app, which is digitally signed using the election authority secret key. After downloading the app, the voter follows the steps below:

1. The voter enters his login information defined during registration. We assume that the voter can perform this step privately.
2. The voter fills a form in order to confirm basic information present in the election database. This information includes his name, a number of registration, and the identifier of the election he wishes do participate, together with any other relevant personal information.
3. The registrar validates the voter's information (by comparing the data sent through the app with the data previously stored during the registration phase).
4. When the information is validated, the voter is added to the list of eligible voters.

At the end of the election setting, candidates and voters are registered. The server stores the following information:

1. Basic voters' personal information: name, registering number, address, etc. (the amount of personal information may vary from one election to another);
2. Voter's Public RSA key (e_v, N_v), in the form of a digital certificate;
3. Hash of the valid login password $H(p_l||s_l)$ of each voter, where s_l is a random salt;

3.2. Voting procedures

After a period of time, predetermined by the election authority, the election setting stage is closed, and no voter or candidate is able to register anymore. The full list of eligible voters and candidates is published to the bulletin board. After a period of dispute, voting is officially opened.

1. In the election app interface, the voter may choose in which election he wants to participate. He is allowed to mark his options offline.
2. After marking his options for the desired election, the application encodes the voter's choices as a byte string b, chooses a random blinding factor r and computes the blinded ballot $b' = r^{e_a} b \pmod{N_a}$, using the election authority public key.
3. The voter is asked to authenticate to the validator using his login password. If he is under coercion, he may use any random fake password. Otherwise, he authenticates using the valid password p_l defined during the registration phase.
4. The voter sends his identification, the election identifier and the blinded ballot
5. The validator checks if the voter is eligible for that election and issues a blind signature $\sigma' = (b')^{d_a} \pmod{N_a}$ and sends it to the voter, together with an encrypted credential, which is signed to guarantee its integrity and authenticity.
6. If the voter used the valid login password, he receives the encryption of a valid credential v_i, for some value of i. Otherwise, he receives the encryption of a random fake credential (simply a random alphanumeric code, not contained in the list of valid credentials).
7. The voter removes the blinding factor and checks the validity of the signature.
8. The ballot is encrypted using the election authority public key. The encrypted ballot and the encrypted credential (with the corresponding signature) are sent to the tallier.
9. After checking the validity of the encrypted credential signature, the tallier computes the hash of the encrypted ballot, concatenated with the corresponding encrypted credential, and sends it to the voter. This hash serves as a tracking number.

After the election is closed, the encrypted ballots, together with their respective encrypted credentials and digital signatures, are published in the bulletin board. The data can be released in order, for example, the first voter id, the first tracking number and the first encrypted ballot belong to voter A, next to voter B and so on. This is possible since all data is encrypted, thus keeping the ballot secrecy. Voters can use their tracking numbers to verify that their ballots were correctly registered. The list of voters who cast their ballots is published. We note that voters are allowed to vote multiple times using fake passwords. However, when a voter casts a ballot using his valid password, any other ballot he tries to cast using the valid password again will be regarded as an invalid vote. As soon as the server detects that the voter has already used his valid password, it will keep responding any ballot signing requests with fake credentials, so that submitted ballots will not be included in the final tally.

3.3. Tallying

After the end of the voting stage, the tallier opens the commitments c_i, revealing the valid credentials v_i and the respective values of salt r_i, so that any interested party can check the validity of the credentials by recomputing $H(v_i||r_i)$ and comparing the result with c_i. Ballots and the respective credentials are decrypted, and the list of decrypted ballots is mixed, so that no voter is able to prove how they voted by associating the decrypted ballot with the encrypted one, whose tracking number could be used as a receipt. Disclosing the election data is a way to ensure that the election has not been altered by an external agent. At this stage we can guarantee that no ballot has been altered, and the number of valid ballots corresponds to the number of voters present in the final list. Next, the tallier counts the votes for each candidate and publishes the result. Any interested party can verify the result by recounting the ballots associated to valid credentials. The authenticity and integrity of these ballots can also be verified by checking the corresponding digital signatures.

4. Conclusions and final remarks

This paper presented the initial proposal for an online voting system based on blind signatures, intended to provide verifiability and integrity, and fake credentials, which allows voters to cast fake ballots that seem indistinguishable from valid ballots, whenever they are under coercion. One of the limitations of the proposed protocol is that voters are subject to forced-abstention attacks, and we must assume that the voter's device is trustworthy. Another drawback is related to the voter's ability to memorize passwords. It is a well known fact that users commonly forget their passwords. Moreover, because the coercion-resistance mechanism prevents the system from displaying any warning when an invalid password is entered, accidentally mistyped passwords (when the voter has the intention to enter a valid password) could be erroneously interpreted as a coercion situation, leading the system to provide the voter with a fake credential. As a result, the voter would cast an invalid ballot believing it was a valid one. A possible solution for these limitations could be to allow the voter to define a set of fake passwords, together with the valid password, so that any other password (probably mistyped) would be detected by the system as invalid. To help the voter to remember his passwords, a scheme of colored passwords could be adopted.

Future works include the conclusion of the implementation and usability tests with

both password mechanisms (the user defines only a valid password, and any other password is regarded as a fake password, thus yielding a fake credential, or the user defines a set of fake passwords with the aid of a colored password mechanism). We also intend to address in more details some security solutions. For instance, in order to avoid timing attacks, we must guarantee that the time taken by the server to respond to a request is constant, regardless of the voter having typed a valid password or a fake password. In addition, DoS attacks must be properly addressed. To avoid them, a mechanism to limit the number of requests, and the time interval between requests, must be included in the final implementation.

Acknowledgments

We would like to thank to all reviewers for their valuable suggestions and comments that contributed to improve the quality of this paper.

References

Adida, B. (2008). Helios: Web-based open-audit voting. In van Oorschot, P. C., editor, *USENIX Security Symposium*, pages 335–348. USENIX Association.

Araujo, R., Neto, A., and Traoré, J. (2018). Civis - a coercion-resistant election system. In *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 29–42, Porto Alegre, RS, Brasil. SBC.

Chaum, D. (1982). Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum.

Chaum, D. and Pedersen, T. P. (1993). Wallet databases with observers. In Brickell, E. F., editor, *Advances in Cryptology — CRYPTO' 92*, pages 89–105, Berlin, Heidelberg. Springer Berlin Heidelberg.

Clarkson, M. R., Chong, S., and Myers, A. C. (2008). Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 354–368.

de Sá, M. O. L., Araujo, R., Sobrinho, A. C. L., Neto, A. S., Maximino, G. S., and Traoré, J. (2020). How colored passwords can improve the usability of coercion-resistant internet voting systems. In *Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems*, IHC '20, New York, NY, USA. Association for Computing Machinery.

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

Juels, A., Catalano, D., and Jakobsson, M. (2005). Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES '05, pages 61–70, New York, NY, USA. Association for Computing Machinery.

Neto, A. S., Leite, M., Araújo, R., Mota, M. P., Neto, N. C. S., and Traoré, J. (2018). Usability considerations for coercion-resistant election systems. In *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*, IHC 2018, New York, NY, USA. Association for Computing Machinery.