SOFTWARE-BASED RECEIPT-FREENESS IN ON-LINE ELECTIONS

Emmanouil Magkos*
Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou, Piraeus, *GREECE*
emagos@unipi.gr

Vassilios Chrissikopoulos
Department of Archiving and Libmry Studies, Ionian University
Corfu, 49100, GREECE
vchris@ionio.gr

Nikos Alexandris
Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou, Piraeus, GREECE
alexandr@unipi.gr

Abstract Electronic elections could be a viable alternative for real-life elections in a democratic society. In traditional elections, a voting booth does more than allow voters to keep their vote secret. The voting booth actually requires that voters vote secretly. If the privacy of the vote was allowed but not required, then a voter could easily sell his vote to a vote-buyer, or be coerced by a coercer. We present a receipt-free election scheme without making any hardware or physical assumptions about the communication channel between the voter and the voting authorities. Our solution is software-based i.e. voters are able to exercise their electoral rights from their home by using a personal computer with Internet access. The only physical assumption we make use of is an untappable channel between the two voting authorities that are employed in our scheme. This scheme satisfies most requirements of a secure electronic election. We make use of well-known cryptographic techniques such as time-lock puzzles and blind signatures.

Keywords: Receipt-freeness, electronic voting, privacy.

*Research supported by the Secretariat for Research and Technology of Greece.

1. Introduction

Due to the rapid growth of the Internet, electronic voting could be a viable alternative for governmental elections, especially in the case of geographically distributed voters with access to open computer networks. If not carefully designed and implemented, e-voting systems can be easily manipulated, thus corrupting election results or violating voters' privacy.

In traditional elections, a voting booth does more than allow voters to keep their vote secret. The voting booth actually requires that voters vote secretly. If the privacy of the vote was allowed but not required, then a voter could easily sell her vote to a vote-buyer, or be coerced by a coercer. All receipt-free schemes met in the literature use hardware assumptions to achieve receipt-freeness. In [15] there are tamper-resistant smartcards that keep some information secret from the voter. Most other schemes 1, 2, 9, 10, 12, 13, 14, 19] make physical assumptions about the communication channel between the voter and the election authorities. More specifically, they assume the existence of :

- *Untappable channels* from the voter to the authority [13, 14]
- *Untappable channels* from the authority to the voter [1, 9, 10, 19]
- *Physical Voting Booths* [2, 12].

In [9], it is argued that ‘‘one-way channels from the authorities to the voters are the weakest physical assumption for which receipt-free voting protocols are known to exist". We believe that these physical assumptions are unsatisfactory: If the underlying communication structure consists of untappable channels between the voting authority and secure dedicated machines (where voters vote), then there is no point of quitting the traditional elections. Real life citizens in a democratic society, who find it inconvenient to go to the polls (and so they finally abstain from the elections) will find it equally inconvenient to cast their vote from a physical voting booth in a dedicated computer network. Note that untappable channels will also force the voter to use specified voting locations.

Our Contribution. We present a software-based receipt-free election scheme, which is secure against a coercer who has tapped all the communication lines between a voter, say Victor, and the voting authorities. Victor's vote is a computational *time-lock puzzle* [17], i.e., it requires a precise amount of time (real time, not CPU time) to be solved, unless a trapdoor information is known in advance. In our election scheme, the trapdoor information is only known to a voting authority. A second authority exists to make sure that votes remain secret until the end of the voting period. A coercer, who wants to find out who Victor voted for, has no other way than running a dedicated computer continuously for a certain amount of time. Even if Victor has incentives to prove his vote to a vote-buyer, there are no means to prove it, since he does not know the trapdoor information. We do not assume any untappable channels between Victor and the voting authority, or any hardware devices. The only physical assumption we make use of is an untappable channel between the two voting authorities that are employed in our system.

The cost paid for receipt-freeness is that the voter constructs her vote inefficiently, by repeatedly squaring a given value, for a significantly large amount of time. However, we believe that this is a minimal tradeoff for a software-based receipt-free solution. To our knowledge, our scheme is the only receipt-free scheme in the literature without the physical assumption of an "untappable" channel between the voter and the voting authority. Voters axe able to exercise their electoral rights from their home by using a personal computer with Internet access. Furthermore, our scheme satisfies most security requirements met in the literature.

2. A Model for Software-based Receipt-Freeness

In our model we assume that a coercer may have tapped the communication channel between the voter and the voting authority. It is clear enough that the vote should be encrypted, for vote secrecy. The *trapdoor* information for the encrypted vote may consist of a secret decryption key and/or the randomness used in a probabilistic encryption scheme. If this trapdoor is in the possession of the voter (e.g. as in [3, 13, 14]) then it could also serve as a receipt for the vote. Even if the voter "lies" about the encrypted vote [l, 9, 10, 19], a coercer who taps the communication channel will eventually find out the value of the vote by eavesdropping on the confidential information exchanged between the voter and the authority. Note that simple encryption does not serve our purposes: a coercer will tap the encrypted message as it is being sent from the voter to the authority (or vice versa), and then require the voter to reveal the trapdoor information. Even worse, the coercer may demand that the voter uses some specific randomness. To summarize: the simplest bit of information that will make the voter's life easier during the construction of the encrypted vote, may also make the coercer's life easier. Thus, software-based receipt-freeness in the presence of a coercer who taps communication lines can *only* be achieved if the voter does not use any secret information other than the vote itself.

We came up with a variation of the *time-lock* concept, as has been described by Rivest, Shamir and Wagner [17]. The idea is based on *preprocessing:* the voter selects a vote from a set of valid votes and constructs a time-lock puzzle of the vote by repetitively squaring a specific value (which is not secret). The number of squarings is also a public parameter. In [17] the user does this efficiently because she knows some trapdoor information. In our model, the user does not know the trapdoor information. The trapdoor information is possessed by a voting authority and it will be used at the end of the voting phase to reveal the cleartext vote. Thus, the voter constructs his vote *inefficiently* by executing an ‘‘intrinsically sequential” process.

A coercer, who taps the communication line between the voter and the voting authority, will get the time-lock puzzle of the vote, as it is being delivered to the authority. Even with the help of the voter, there is no way to reverse the time-lock process: the voter does not know the trapdoor information, so the coercer will have to run a dedicated computer for a specific amount of time. This time can be determined by an independent authority who sets the public parameters, e.g. the number of squarings for each puzzle, so as to prevent massive coercion in a large-scale election: assuming that each voter performs n squarings, a coercer will have to perform nk squarings to coerce k voters. However the voter too constructs his vote inefficiently, but we believe that this is a minimal tradeoff for a software-based receipt-free solution that does not employ untappable channels between the voter and the voting authority.

The only physical assumption we make use of is an untappable channel between the two voting authorities employed in our system. This is acceptable, since our main goal was to abolish the necessity of a physically secure channel between the voter and the authority. We believe that an untappable channel between two authorities that belong to a distributed set of voting authorities, is a minimal physical assumption for a receipt-free scheme. We could remove this physical assumption by requiring that there is only one voting authority, but in that case, and unless full trust was granted to this authority, fairness would have been sacrificed: if the authority possesses the trapdoor information, then votes may be revealed before the end of the voting period.

3. Building Blocks

Our voting scheme makes use of *blind signatures* [4], which is a well known technique, already implemented with the RSA algorithm [16, 20]. Blind signatures are the equivalent of signing carbon-paper envelopes: a user seals a slip of a paper inside such an envelope, which is later signed on the outside. When the envelope is opened, the slip bears the carbon image of the signature. Furthermore, users in our scheme lock their votes in a *time-lock puzzle*. The mechanism is a variation of a well-known technique [17] and is presented below.

3.1. Time-lock Puzzles

Suppose that Alice wants to encrypt a message M so that Bob can decrypt it after a period of T seconds. T is a real (not CPU) time period, given that Alice knows (or approximately assumes) in advance the CPU power of Bob. In [17], Alice generates a composite modulus $n = pq$ as the product of two large primes p and q. Then, Alice computes $\Phi(n) = (p - l)(q - 1)$ and $t = TS$, where S is the number of squarings modulo n per second that Bob can perform. Alice chooses a key K for a symmetric cryptosystem and encrypts M with key K, thus getting $CM = Enc(K, M)$. In order to hide K, she picks a random a modulo n and encrypts K as:

$$C_K = K + a^{2^t} \ (mod\ n) \tag{1}$$

To do this efficiently, Alice uses the trapdoor information $\Phi(n)$ that only she knows: She first computes $e = 2^t (mod\ \Phi(n))$ and then $b = a^e (mod\ n)$. The public output of the puzzle is the set (n, a, t, C_M, C_K). Since Bob does not know the factors p and q, computing $\Phi(n)$ from n is provably as hard as factoring n. Bob has no way of computing a^{2^t}, other than starting with a and perform t sequential squarings, each time squaring the previous result. The computational problem of performing these squarings *is* not parallelizable: having two computers is not better than having one computer.

Our variation. In our model, Alice is the voter and Bob is the coercer. Alice *does not know* the trapdoor information $\Phi(n)$ (if she knew it she could hand it over to Bob, e.g. in a vote-selling scenario), *so* she cannot construct the puzzle efficiently. In addition, there are two voting authorities. The first authority selects n, p and q, and publishes n and t, where t is the number of squarings that Alice has to perform. Alice selects a as previously and computes a^{2^t}. Alice's vote v takes the place of the key K in equation (1), thus yielding:

$$C_V = v + a^{2^t} (mod\ n) \tag{2}$$

The public information will now be the set (n, a, t, Cv). When the time comes, Alice uses a clear channel to submit the time-lock puzzle of her vote to the second voting authority. The first voting authority, who possesses the trapdoor information $\Phi(n)$, will later cooperate with the second authority to decrypt the submitted votes. In Section 4, our voting protocol is presented in detail.

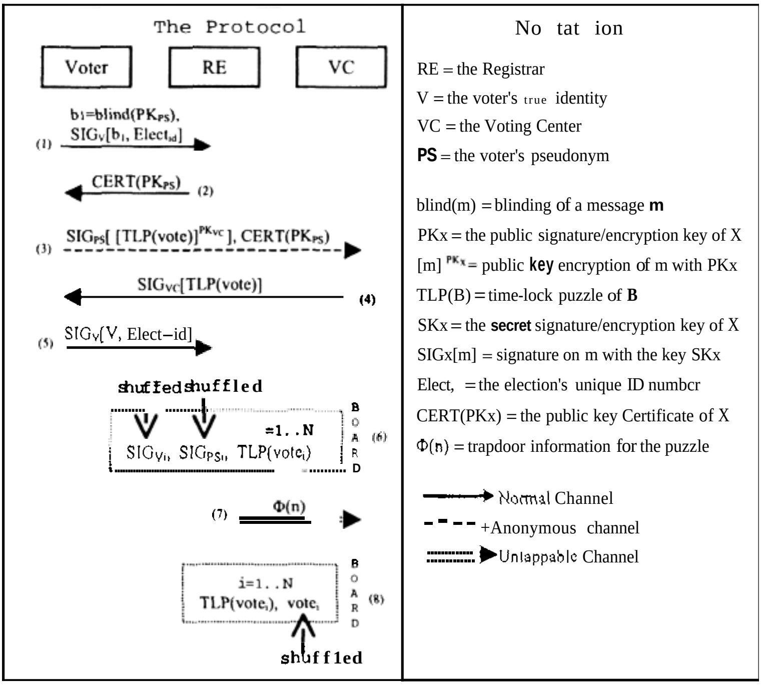
An efficient solution with "secure" hardware. Another solution would be each voter to be equipped with a tamper-resistant smartcard. During an off-line registration protocol, this smartcard would be provided with the trapdoor information $\Phi(n)$. Later, during voting phase, the voter would provide the smartcard with her preferable vote, and the smartcard would use the trapdoor information $\Phi(n)$ to construct the time-lock puzzle in an efficient way. To reveal the vote, the coercer would either have to tamper with the smartcard or solve the time-lock puzzle.

4. A Receipt-free E-voting Scheme

In our protocol there are N voters and two authorities, the Registrar and the Voting Center. The Registrar acts as an intermediate between the voter and the Voting Center, while the Voting Center is responsible for tallying the votes. We assume that each authority is *semi-trusted* [7], i.e., the authority may misbehave but will not conspire with another party. We also make use of a bulletin board, which is publicly readable. Only the Voting Center can write to it, and nobody can delete from it. The Voting Center is committed to everything that is published on the bulletin board.

There is a certificate infrastructure and all participants are legally bound by their signatures. Voters and authorities possess a private/public key pair for signature and encryption as well as the corresponding certificates, issued by a trusted Certification Authority. We also assume that there is an untappable channel between the Registrar and the Voting Center. Communication between voters and authorities takes place through an *anonymous channel*: voters can send/accept messages that cannot be traced (e.g., by using traffic analysis). For example, e-mail anonymity can be established with Mixmaster re-mailers [5], and HTTP anonymity can be established with services such as the Onion-Routing system [8]. The election protocol is depicted on Figure 1. It is split into four stages, the *Authorizing stage* (Steps 1-2), the *Voting* stage (Steps 3-5), the *Claiming stage* (Step 6) and the *Tallying stage* (Steps 7-8).

Authorizing Stage. A voter, say Victor, wishes to get a certified pseudonym that will identify him to the Voting Center. Victor creates a private/public key pair $(SKps, PKps)$, blinds PK_{PS} (the public tallying key) to create the blinding b_1 and then signs a message consisting of b_1 and the unique election identification number $Elect_{id}$ (Step 1).



Notation

RE = the Registrar
V = the voter's true identity
VC = the Voting Center
PS = the voter's pseudonym

blind(m) = blinding of a message **m**
PKx = the public signature/encryption key of X
$[m]^{PKx}$ = public key encryption of m with PKx
TLP(B) = time-lock puzzle of **B**
SKx = the secret signature/encryption key of X
SIGx[m] = signature on m with the key SKx
Elect, = the election's unique ID numbcr
CERT(PKx) = the public key Certificate of X
Φ(n) = trapdoor information for the puzzle

→ Normal Channel
- - - +Anonymous channel
▶ Unlappable Channel

Figure 1. A receipt-free election scheme (software-based)

Victor sends these to the Registrar and gets the blinding signed by the Registrar (Step 2). Victor unblinds the Registrar's signature on b_1 and **is** left with a certificate of the public tallying key, *CERT(PKps)*.This certificate will be used later by the Voting Center to verify signatures that are made with the secret tallying key, *PKps* The public key PK_{PS} will be Victor's official pseudonym.

Voting Stage. In Step 3, Victor, who has already constructed a time-lock puzzle of his vote, *TLP(vote)*,encrypts it with the public key of the Voting Center, and signs the result using his secret tallying key, thus producing $SIG_{PS}[[TLP(vote)]^{PK_{VC}}]$. He anonymously sends this to the Voting Center, along with the certificate of his public tallying key. The Voting Center verifies the signature, decrypts the message, stores the time-lock puzzle in a local database and returns, in Step 4, a signature on the puzzle, $SIG_{VC}[TLP(vote)]$. This can be seen **as** a receipt that the Voting Center has accepted the time-lock puzzle of the vote. At some time later, in Step 5, Victor uses his authentic signature key to sign a message consisting of his true identity *V* and the $Elect_{id}$ number. He then sends the signature to the Voting Center.

Claiming Stage. In step 6, the Voting Center publishes on the board, in random order, the list of the authentic and pseudonymous signatures SIG_{V_i} and SIG_{PS_i}, $i = 1,...,$ N. The Voting Center also publishes all the time-lock puzzles $TLP(vote_i)$ of the votes that have been successfully submitted. In case Victor's time-lock puzzle is not published, he can protest by broadcasting $SIG_{VC}[TLP(vote)]$, with no need to reveal in which way he actually voted. This is called an "open objection to the tally", introduced by Sako in [18].

Tallying Stage. In Step 7, the Registrar sends the secret trapdoor $\Phi(n)$ to the Voting Center, by using an untappable channel. No one, except the Voting Center, can have access to $\Phi(n)$. The Voting Center uses $\Phi(n)$ to solve the time-lock puzzles of the votes. In Step 8, the Voting Center publishes in clear the results of the election, i.e. the list of the votes $vote_i$, $i = 1,...,$ N. The Voting Center also publishes a list with the corresponding time-lock puzzles of the votes, $TLP(vote_i) = [TLP(vote_1),...,TLP(vote_n)]$.

5. Security Analysis

We evaluate the security of our scheme by examining some basic requirements, which most researchers seem to agree upon [6, 20]:

Eligibility. (Only authorized voters are able to vote). In Step 1, Victor signs a message using his authentic signature key. The Registrar checks the eligibility of each user who submits a tallying key for certification.

Unduplicability. (No one is able to vote more than once). The Registrar will not issue more than one tallying keys for each voter. In Step 6, all the authentic signatures of the voters are published. Consequently, it is not possible to exist more tallying keys than authentic public keys, so the Registrar cannot misbehave without being caught.

Untraceability. (All votes remain anonymous). When Victor submits a tallying key for certification, he signs a message and the Registrar checks his identity. However, the tallying key is blindly signed by the Registrar in Steps 1-2. Consequently, the Registrar cannot trace any signature SIG_{PS_i} published in Step 6, back to Victor's real identity. Furthermore, Victor in Step 3 uses an anonymous channel to submit his validated time-lock puzzle. The puzzle cannot be traced back to its sender, since it is signed under a certified pseudonym (the tallying key).

The link between Victor's pseudonym and his real identity cannot be done by either authority.

Fairness. (All ballots remain secret while voting is not completed). The trapdoor information necessary to solve the puzzle, is in the possession of the Registrar. Victor encrypts the time-lock puzzle of his vote with the public encryption key of the Voting Center, and sends it to the Voting Center. The Voting Center will not publish the time-lock puzzles until the end of the voting period. Fairness is achieved, as long as the Registrar and the Voting Center do not combine their knowledge. Neither the Registrar nor the Voting Center can break fairness by themselves. Since the Registrar and the Voting Center are assumed to be semi-trusted, this requirement is satisfied.

Accuracy. (No one is able to alter/delete anyone else's vote). In Step 6, the Voting Center commits to the time-lock puzzles of all the votes and cannot alter them, according to the properties of the bulletin board. Every voter, whose time-lock puzzle has not been taken into account, can make an ''open objection to the tally".

Atomic Verifiability. (Voters are able to verify that their vote has been counted correctly). In Step 6, all the time-lock puzzles of the votes are published by the Voting Center. Victor can check that his time-lock puzzle has been published on the board. If not, Victor makes an open objection: he anonymously broadcasts the receipt that was sent to him in Step 4.

Receipt-Freeness. (No voter is able to prove the value of its vote). The receipt freeness property is separately discussed in Section 2. It must be noted that the scenario of a coercer who observes the voters at the moment they vote, is not addressed at all. This attack cannot be prevented by any e-voting scheme and is rather unrealistic in large-scale elections.

Responsibility. (Eligible voters who have not voted can be identified). This is an optional requirement, desirable in Australian elections [11]. All voters, who receive in Step 4, an acknowledgment of their votes from the Voting Center, sign a message by using their authentic signature keys and send this message to the Registrar, in Step 5. The Registrar has already received, in Step 1, the authentic signatures of all eligible voters, so he is able to identify, by comparing the corresponding lists, the eligible voters who have not voted.

6. Discussion

We have presented a receipt-free election scheme, which satisfies most requirements of a secure election. We do not assume any hardware devices or untappable channels between the voter and the voting authorities. We make use of well-known cryptographic primitives that have been implemented. Time-lock puzzles, while being very difficult in their solution, are quite efficient in their construction. The problem with our scheme is that we sacrifice efficiency in order to achieve software-based receipt-freeness. While the computations *during* the election are done quickly and in few steps, the computations made by the voter *before* the election (the preprocessing for the time-lock puzzle) are not done in a reasonable amount of time. This time is determined by an authority, and has to be long enough to discourage massive coercion of voters. Yet, as noted in Section 3.1, our scheme could be relaxed to become an efficient scheme with smartcards. In such case, however, the scheme would be a hardware-based solution.

References

[1] D. Alpert, D. Ellard, O. Kavazovic, M. Scheff. *Receipt-Free Secure Elections 6.857 Final Project,* 6.857 Network and Computer Security, 1998, http://www.eecs.harvard.edu/~ellard/6.857/final.ps.

[2] J. Benaloh, D. Tuinstra. *Receipt-free secret-ballot elections,* 26th Annual ACM Symposium on the Theory of Computing, Proceedings, 1994, pp. 544-553.

[3] R. Canetti, C. Dwork, M. Naor, R. Ostrovsky. *Deniable Encryption,* Advances in Cryptology - CRYPTO 97, Proceedings, Lecture Notes in Computer Science, Vol. 1294 , Springer-Verlag 1997, pp. 90-104.

[4] D. Chaum. *Blind Signatures for Untraceable Payments*, Advances in Cryptology - CRYPTO 82, Proceedings, Plenum Press 1982, pp. 199-203.

[5] L. Cottrell. *Mixmaster and Remailer Attacks,* http://obscura.obscura.com/~loki/remailer/remailer-essay. html.

[6] L. Cranor, R. Cytron. *Sensus: A security-conscious electronic polling system for the Internet,* Hawaii International Conference on System Sciences, Proceedings, 1997, http://www.research.att.com/~lorrie/pubs/hicss/hicss.html.

[7] M.Franklin, M. Reiter. *Fair exchange with a semi-trusted third party,* 4th ACM Conference on Computer and Communications Security, Proceedings, ACM 1997, pp. 1-6.

[8] D. Goldschlag, M. Reed, P. Syverson. *Onion Routing for Anonymous and Private Communications,* Communications of the ACM, Vol. 42(2), pp. ACM 1999, pp. 39-41.

[9] M. Hirt, K. Sako. *Eficient Receipt-fie Voting Based on Homomorphic Encyption,* Advances in Cryptology - EUROCRYPT 2000, Proceedings, Lecture Notes in Computer Science, Vol. 1807, Springer-Verlag 2000, pp 539-556.

[10] B. Lee, K. Kim. *Receipt-free Electronic Voting through Collaboration of Voter and Honest Verifier,* JWISC 2000, Proceedings, 2000, pp. 101-108.

[11] Y. Mu, V. Varadharajan. *Anonymous Secure e-voting over a network,* 14th Annual Computer Security Application Conference, Proceedings, IEEE Computer Society 1998, pp. 293-299.

[12] V. Niemi, A. Renvall. *How to prevent Buying of Votes in Computer Elections,* Advances in Cryptology - ASIACRYPT 94, Proceedings, Lecture Notes in Computer Science, Vol. 917, Springer-Verlag 1994, pp. 141-148.

[13] T. Okamoto. *An Electronic Voting Scheme,* IFIP '96, Proceedings, Advanced IT Tools, Chapman & Hall 1996, pp. 21-30.

[14] T. Okamoto. *Receipt-Free Electronic Voting schemes for Large Scale Elections,* Workshop of Security Protocols '97, Proceedings, Lecture Notes in Computer Science, Vol. 1163, Springer-Verlag 1996, pp. 125-132.

[15] A. Riera, J. Borrell, J. Rifa. *An uncoerciblevverifiable electronic voting protocol,* 14th International Information Security Conference IFIP/SEC'98, Proceedings, 1998, pp. 206-215.

[I6] R. Rivest, A. Shamir. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,* Communications of the ACM, Vol. 21, ACM 1978, pp. 120-126.

[17] R. Rivest, A. Shamir, D. Wagner. *Time-Lock Puzzles and Timed-Released Crypto,* LCS Technical Memo MIT/LCS/TR-684, 1996, http://www.theory.Ics.rnit.edu/ ~rivest /RivestShamirWagner-timelock.ps

[18] K. Sako. *Electronic Voting Scheme Allowing Open Objection to the Tally,* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Proceedings, Vol. E77-A(1), 1994, pp. 24-30.

[19] K. Sako, J. Killian. *Receipt-Free Mix-TypeVoting Schemes - A practical solution to the implementation of voting booth,* Advances in Cryptology - EUROCRYPT 95, Lecture Notes in Computer Science, Vol. 921, Springer-Verlag 1995, pp. 393-403.

[20] B. Schneier. *Applied Cryptography - Protocols, Algorithms and Source Code in C,* 2nd Edition, 1996.