A pervasively verifiable online voting scheme

Lucie Langer, Axel Schmidt, and Roberto Araújo

Technische Universität Darmstadt

{langer, axel, rsa}@cdc.informatik.tu-darmstadt.de

Abstract: We present an improvement of Ohkubo et al.'s e-voting protocol [OMA$^+$99]: We provide that the validator's signature is attached directly to the vote although the vote has been encrypted before. Thus verifiability does not end before tallying like in the original scheme but is pervasive even beyond the end of the election. This prevents manipulation of the plaintext votes and offers verifiability to the public instead of restricting it to the talliers. Moreover, privacy can be ensured without using blind signatures, which facilitates receipt-freeness. Our variant of the scheme also allows the voter to seemingly comply with the demand of a coercer while actually casting the vote she intended.

1 Introduction

The e-voting scheme we propose is an improvement of the protocol designed by Ohkubo et al. [OMA$^+$99] which is based on the Fujioka et al. scheme [FOO92]. [FOO92] uses a bit-commitment scheme to ensure fairness. This involves voter action during tallying phase, i.e. all voters must wait until the voting stage is completed. [OMA$^+$99] overcomes this obstacle by employing a threshold encryption scheme of several talliers to achieve fairness. This way the "*walk-away-property*" in [OMA$^+$99] is realized, thereby offering much more convenience to the voters. [KKLA01] combine Ohkubo et al.'s protocol [OMA$^+$99] with a PKI which is used for key distribution and registration of the voters.

In [OMA$^+$99], the encrypted votes are signed by a validator to offer verifiability of their validity. This signature is provided after the votes have been encrypted with the tallier's public key. Hence, before tallying, the signatures are discarded due to the decryption process. Thus undetectable manipulation of the plaintext votes is still possible. Verifiability ends before tallying and correctness of the votes afterwards strongly depends on trustworthy talliers and secure storage of the votes. We improve the protocol by providing what we call *pervasive verifiability* for the votes. We therefore modify the signature process: Each vote is first encrypted by the voter with the tallier's public key. Then the vote is signed by the validator. In contrast to [OMA$^+$99] we don't use blind signatures here while the content of the vote is still hidden from the validator. Using the ElGamal cryptosystem combined with RSA signatures we construct the signature process in a way that, despite of the encryption, the vote itself is signed and can be verified after decryption. Hence, in and after the tallying phase all plaintext votes are still protected by the validator's signature. This has two advantages: Firstly, all votes can be verified publicly until the very end of the election process and even beyond. Secondly, the final votes which represent the election result are protected against manipulation.

Under certain circumstances the validator's signature of the vote could serve as a receipt for the voter. To prevent this we ensure that the voter can never obtain the signature of her vote while at the same time she is still capable of verifying her vote before sending it to the bulletin board. Moreover, the voter cannot prove her vote to a coercer or vote-buyer. We explain that in more detail in Section 3.

2 Proposed voting scheme

In this section we first describe the cryptographic primitives we use. After introducing the players of the voting scheme we then describe the steps of the protocol.

2.1 Cryptographic primitives

ElGamal encryption. ElGamal is a probabilistic encryption algorithm for public key cryptography. Let G be a group of order n, g a generator of G, x a private key and $h = g^x$ the corresponding public key. Then a message $m \in G$ is encrypted by choosing a random $r \in \mathbb{Z}_n$ and computing $E(m) = (c_1, c_2) = (g^r, mh^r) \in G \times G$. Decryption is performed by computing c_2/c_1^x. ElGamal offers semantic security, provided that the Decisional Diffie-Hellman Problem (cf. [Bon98]) is hard in G.

Usually the group G is a multiplicative subgroup of $\mathbb{Z}_p$ of prime order q. Since we want to use RSA signatures in our scheme, we will be working with ElGamal encryption with a composite modulus as introduced in [FH96] and [Gol04]: Let p, q be two large primes and $n = pq$. We set $G := \mathbb{Z}_n^*$. ElGamal encryption and decryption then works as specified above. For details see [FH96].

RSA signatures. Let p, q be two large primes and $n = pq$ the RSA modulus. Let d denote a private signing key and e the corresponding public verification key such that $de = 1 \bmod \varphi(n)$. Then a message m is signed by computing $Sig(m) = m^d \bmod n$. Verification is performed by $Sig(m)^e$ which yields m if the signature is correct. Since a deterministic signature scheme facilitates duplication of votes (see also Section 3), we use randomized RSA signatures as specified in [JK03].

Threshold encryption scheme. A threshold encryption scheme distributes the decryption function among a number of decryption authorities to enhance the level of security. The secret key is split into shares and each share is given to one of the authorities. For tallying we use (t, n)-threshold encryption, where any t out of n talliers can decrypt ciphertexts, while any combination of corrupted talliers less than t cannot. Detail can be found for example in [FP01].

2.2 The players

The following players take part in the voting scheme: The *registration authority* issues key material to eligible voters. The *voter* encrypts her vote and sends it to the *validator* who checks the voter's eligibility and hereafter validates the ballot with his signature. The votes are posted on the *bulletin board* which is a public channel where data can be published by authorized parties only and, once published, cannot be erased or overwritten by anyone. The *mix net* anonymizes the votes by hiding the relation between incoming and outgoing messages, i.e. encrypted votes. Our scheme employs an ElGamal-based re-encryption mix net (cf. [OKST97]). The n distributed *talliers* count the votes in the end and publish the result of the election.

2.3 Protocol description

In the following we describe the steps of the protocol. Any communication is assumed to be secured by SSL/TLS. For simplicity of subscripts we fix one voter V instead of referring to V_i.

Preparation phase.

This phase comprises issuing keys and certificates to the players. Furthermore, from the registration authority each eligible voter V obtains two factors g^{r_A} and h^{r_A} where g is the generator of the group G used in the ElGamal encryption system, h is the tallier's public key and r_A is a random number used for ElGamal encryption. We assume that g^{r_A} and h^{r_A} are distributed over an untappable channel. The list of eligible voters is signed by an authorized party (e.g. the election board) and published on the bulletin board so that it can be verified by anyone.

Voting phase.

The voter first encrypts her vote with the factor h^{r_A} she obtained in registration phase, i.e. she computes vh^{r_A}. Then she re-encrypts her vote by choosing a random $r_V \in \mathbb{Z}_n$ and computing $vh^{r_A}h^{r_V}$ as well as $g^{r_A}g^{r_V}$. The voter sends the result $E(v) = (g^{r_A+r_V}, vh^{r_A+r_V})$ to the validator over a sender-authenticated channel.

The validator checks the requesting voter's eligibility according to the list on the bulletin board. Then the validator signs the encrypted vote using his private RSA signing key d, i.e. $Sig(E(v)) = ((g^{r_A+r_V})^d, (vh^{r_A+r_V})^d)$ is computed and sent back to the voter. Note that this is an ElGamal encryption of v^d using a modified exponent $(r_A + r_V)d$.

The voter verifies the validator's signature: She first computes $(v^d h^{(r_A+r_V)d})^e = vh^{r_A+r_V}$ and then divides by $h^{r_A} h^{r_V}$ to check whether v is obtained indeed. Then the voter sends the encrypted vote $E(v) = (c_1, c_2) = (g^{r_A+r_V}, vh^{r_A+r_V})$ and its appended signature $E'(Sig(v)) = (s_1, s_2) = (g^{(r_A+r_V)d}, v^d h^{(r_A+r_V)d})$ to the bulletin board.

Mixing phase.

The votes are anonymized using a re-encryption mix net [OKST97]: Each mix server M shuffles and re-encrypts the votes $E(v) = (c_1, c_2)$ including their encrypted signatures $E'(Sig(v)) = (s_1, s_2)$ by choosing random $r_M, r'_M \in \mathbb{Z}_n$ and computing $(c_1 g^{r_M}, c_2 h^{r_M}) = (g^{r_A+r_V+r_M}, vh^{r_A+r_V+r_M})$ as well as $(s_1 g^{r'_M}, s_2 h^{r'_M}) = (g^{(r_A+r_V)d+r'_M}, v^d h^{(r_A+r_V)d+r'_M})$.

Tallying phase.

The talliers jointly decrypt the votes and post them on the bulletin board as pairs $(v, Sig(v))$.

3 Analysis

Our scheme provides *pervasive verifiability* by signing the votes in a way that they can be verified *after* decryption. In more detail, the validator signs the encrypted vote by computing $(vh^{r_A+r_V})^d = v^d h^{(r_A+r_V)d}$, so after decryption by the talliers, we have the signed vote v^d.

The signature process has to be probabilistic and thus non-deterministic because otherwise identical votes would lead to identical signatures and hence could easily be duplicated without the need to forge the signature. Concluding each vote has an individual signature. But then the voter could use the signature as a receipt: As at the end of the election the signed votes are published for verification purpose, a voter could prove her vote by showing the validator's signature of her vote, which the voter could know from the signing process. This way anonymity is broken and vote-selling and coercion would be possible. Hence it is crucial to hide the signature from the voter during the signing process to prevent the link. Nevertheless the voter wants to verify that the signed and encrypted vote she receives from the validator in fact contains her original vote. Both is possible: The voter receives $v^d h^{(r_A+r_V)d}$ from the validator. But as she does not know d she cannot retrieve the signature v^d by division to use it as a receipt. Nonetheless she can verify that her vote v is indeed contained: the voter computes $(v^d h^{(r_A+r_V)d})^e / h^{r_A+r_V} = v$ and retrieves her original vote. Note that division by $h^{r_A+r_V}$ is necessary here because otherwise a malicious validator could have created his own vote $\tilde{v}$ and multiplied it with an according factor to obtain the product $vh^{r_A+r_V}$ which would remain undetected.

Now we show why the last computation cannot serve as a proof for a coercer or vote-buyer. The voter uses the term $h^{r_A+r_V}$ to verify her vote. Assume an coercion scenario where a coercer forces the voter to cast the vote v_c. If an attacker coerces the voter to reveal the verification term $h^{r_A+r_V}$, the voter can trick him by using a fake term instead: The voter gives $c := (vh^{r_A+r_V})/v_c$ to the coercer who consequently believes to verify the vote v_c. The coercer cannot differentiate if c is the correct term $h^{r_A+r_V}$ or not due to two facts. First, if c is not a power of the public key h, the coercer would realize being cheated. But he cannot decide this due to the Decisional Diffie-Hellman Assumption (see [Bon98] for details). Secondly, the coercer could force the voter to reveal the random exponent $r_A + r_V$ as a proof of secret knowledge about the term $h^{r_A+r_V}$. For this reason the exponent is a secret shared between the voter (r_V) and the registration authority (r_A). The voter only knows h^{r_A}, but not r_A. Hence the voter cannot reveal the exponent completely. Tricking the coercer is possible because he cannot obtain h^{r_A} since we assume an untappable channel from the registration authority to the voter. Re-encryption by the voter using h^{r_V} is necessary since otherwise a malicious registration authority could reveal the vote and thus break anonymity and fairness.

In a vote-buying scenario the voter wants to prove her vote to a buyer. Of course she could reveal $h^{r_A+r_V}$ but this is not a proof as tricking by using a fake term c could not be detected by the buyer for the above reasons.

Our scheme provides individual verifiability in the following sense: Each signed and encrypted vote which is sent to the bulletin board is randomized due to the random factors used in the encryption. So the voters can check by comparison whether the encrypted votes they cast are actually on the bulletin board. This is secure as the votes are encrypted. In the next step, the votes are shuffled and re-encrypted in the mix net. The output of the mix net cannot be linked to the encrypted votes seen before on the bulletin board. The mix net cannot modify the encrypted votes, as this would invalidate the signature. It cannot duplicate votes as they are randomized, thus identical votes are highly unlikely. Finally the mix net cannot erase votes undetected as the output and input of the mix net are published on the bulletin board and thus can be compared easily.

The same holds for the talliers. After decryption of the votes, their validity can be checked by verifying the signatures. We use a distributed tallier system to provide fairness. A single tallier could start decrypting and tallying before the end of the election. In the distributed system the talliers have to decrypt jointly, so up to $t - 1$ malicious talliers cannot decrypt on their own.

4 Conclusion

We have proposed an online voting scheme which improves the protocol [OMA$^+$99] by providing pervasive verifiability: In our variant of the scheme votes remain signed by the validator during tallying and even beyond the end of the election. This prevents manipulation of the plaintext votes and offers verifiability to the public instead of restricting it to the talliers. The validator's signature is hidden from the voter and hence cannot be used as a receipt. Still, the voter is able to verify that her vote was cast as intended. This verification process cannot be misused by a coercer or vote-buyer since the voter is not able to prove her vote.

References

[Bon98] Dan Boneh. The Decision Diffie-Hellman Problem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[FH96] Matthew K. Franklin and Stuart Haber. Joint Encryption and Message-Efficient Secure Computation. *Journal of Cryptology*, 9(4):217–232, 1996.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In Jennifer Seberry and Yuliang Zheng, editors, *ASIACRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[FP01] Pierre-Alain Fouque and David Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2001.

[Gol04] Philippe Golle. Reputable Mix Networks. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 51–62. Springer, 2004.

[JK03] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) 1. RFC 3447, 2003.

[KKLA01] Kwangjo Kim, Jinho Kim, Byoungcheon Lee, and Gookwhan Ahn. Experimental Design of Worldwide Internet Voting System using PKI. In *Proceedings of SSGRR International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, August 2001.

[OKST97] Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault tolerant anonymous channel. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *ICICS*, volume 1334 of *Lecture Notes in Computer Science*, pages 440–444. Springer, 1997.

[OMA$^+$99] Miyako Ohkubo, Fumiaki Miura, Masayuki Abe, Atsushi Fujioka, and Tatsuaki Okamoto. An Improvement on a Practical Secret Voting Scheme. In Masahiro Mambo and Yuliang Zheng, editors, *ISW*, volume 1729 of *Lecture Notes in Computer Science*, pages 225–234. Springer, 1999.