Using Mobile Device Communication to Strengthen e-Voting Protocols

Michael Backes
Saarland University &
MPI-SWS
Saarbrücken, Germany
backes@mpi-sws.org

Martin Gagné
Saarland University
Saarbrücken, Germany
gagne@cs.uni-saarland.de

Malte Skoruppa
Saarland University
Saarbrücken, Germany
skoruppa@cs.uni-saarland.de

ABSTRACT

Remote e-voting protocols strive to achieve sophisticated security properties. However, the inherent complexity of this level of sophistication typically comes at a cost: Protocols must either accept trade-offs in terms of security or are impractical. In this paper, we show how the additional communication capabilities given by the pervasive availability of mobile phones today can be used to strengthen the security offered by remote e-voting protocols. More precisely, the presence of two separate channels between the voter and the election authorities, namely the possibility for voters to communicate with authorities using both their computers and their mobile phones, opens up useful possibilities to significantly improve the security of remote e-voting with little cost in practicality.

We discuss three mobile building blocks that can be plugged into many existing protocols from the literature, and that yield important security properties such as eligibility, resistance against impersonation attacks, inalterability, vote independence and coercion resistance, and even privacy and integrity of votes in the presence of malicious computers, under realistic assumptions.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*; E.3 [**Data**]: Data Encryption—*Public key cryptosystems*

Keywords

remote electronic voting, mobile devices, secure platform problem

1. INTRODUCTION

Protocols for remote e-voting have been studied for about three decades. Nonetheless, the inherent complexity of such systems and seemingly conflicting requirements with regards to their security have left researchers skeptical as to their realizability in large-scaled, say nation-wide, elections. Indeed, existing protocols either do not satisfactorily address the problem of coercion and vote-buying [2, 3, 10], or they are too impractical and involved to be understood by the vast majority of users [5, 11]. In particular, a notable problem is to achieve the property of *individual verifiability* (a voter can verify that their vote was counted as intended) without violating *coercion resistance* (a voter cannot prove to a coercer how they voted). This problem has been addressed in the past by using designated verifier proofs [13], or else by enabling voters to forge fake credentials [5, 11]. These avenues constitute impressive ideas from a cryptographic point of view, yet from a user perspective they are simply too complicated to be deployed in practice. Additionally, the secure platform problem [9], that is, the problem of voting using potentially corrupted computers, remains open. In fact, despite being a realistic threat undoubtedly relevant for governments that wish to implement e-voting [10, 14], this problem is rarely addressed at all in remote e-voting protocols. Instead, most protocols from the literature focus on the voting system itself, and rely on the voter's computer operating strictly according to the voter's wishes (often, a voter and their computer are simply modeled as a single entity). Yet, viruses and trojans are abundant nowadays, and important security properties such as individual verifiability or coercion resistance should clearly not rely on the trustworthiness of voters' computers.

With the pervasive presence and availability of mobile devices today, we show that such problems can be addressed much more efficiently. In particular, due to the presence of two channels between the voter and the election authorities, namely, the possibility for voters to communicate with authorities both using their computers (via the Internet) and using their mobile phones (via the mobile communications network) opens up practical possibilities to significantly improve the security of such protocols.

In practice, it clearly is harder for any potential attacker to eavesdrop on both the Internet and the mobile communications traffic of a voter than to listen on a single channel. Yet, classical protocols only make use of one channel between voter and authorities, typically the Internet. Online banking services have already recognized the improved security given by communicating with their clients on two separate channels: Nowadays, most such services make use of the mobile communications network in order to send transaction authentication numbers (TANs) for confirming financial online transactions directly to their clients' mobiles. While not

WPES'13, November 4, 2013, Berlin, Germany.

ACM 978-1-4503-2485-4/13/11 ...$15.00.
http://dx.doi.org/10.1145/2517840.2517863.

perfect [12], the security of this approach has proven to be reasonably strong in practice.

In addition, the secure platform problem can be much better addressed when both a computer and a mobile phone have to be used for voting. Indeed, the presence of two separate channels between the voter and the authorities yields the possibility that neither device learns all traffic of a particular voter: Clearly, it is a much less realistic attack model that both of these devices owned by a particular voter are malicious and collude, than an attack model where only one of the two devices is malicious and does not operate according to the voter's actual wishes. That is, by requiring both a computer and a phone we avoid the presence of a single point of failure. Assuming that at least one of these devices is honest, we can ensure privacy and integrity of votes even when the other device is corrupted.

Contributions. While some governments have started using mobile devices in their remote e-voting protocols [6, 10], the security properties offered by such methods have never been carefully analyzed. In this work, we study three mobile building blocks that are modular enough to be easily plugged into many existing remote e-voting protocols. We show that they help to achieve several noteworthy security properties, and outline the underlying assumptions.

The first of these building blocks are *humanly computable one-time pads*. These are one-time pads sent to a voter's mobile phone that a voter uses to pre-encrypt their vote before they enter it into a computer. Assuming that the mobile device is honest and that an attacker cannot eavesdrop on the mobile channel, the encryption of the vote yields perfect secrecy, even when the voter's computer is corrupted.

Second, we study the use of *return codes* sent to a voter's mobile phone and used to guarantee vote integrity. While this idea has been proposed as an isolated mechanism in the past [1, 10], we investigate how it combines with the humanly computable one-time pads mentioned above. We study its impact on security and how it interplays with the privacy of a voter's ballot, in particular towards the voting servers, who must necessarily compute the return codes.

Finally, we investigate the possibility of having a voter confirm their vote using a *transaction authentication number* that is also sent to their mobile. Thereby, a voter's mobile effectively works as an additional credential, offering an improved resistance against impersonation attacks that arise when voting using a corrupted computer, or when credentials sent to a voter before the election are otherwise leaked.

Using these building blocks, we obtain a high degree of protection with regards to a corrupted platform. Additionally, we remove much of the cryptography that makes protocols error-prone and hard to understand from a user perspective.

2. PREREQUISITES

The techniques discussed in this paper are based on the assumption that a voter who wishes to interact with the voting servers has access to a computer and owns a personal mobile phone, whose number is known to the authorities. As depicted in Figure 1, we keep our interaction model as general as possible, so that it may be instantiated with potentially any remote e-voting protocol from the literature. In order to capture the possibility of a potentially corrupted platform, in our model a voter does not directly interact with the voting servers herself; rather, she interacts with her computer and mobile phone, and these devices interact with the voting servers. We assume bidirectional links between the voter, computer and server, but only unidirectional links from server to phone and from phone to voter, to model the fact that a phone is generally a poor input device. None of our techniques requires any computation from the phone – although we will see that security is greatly improved by, say, the presence of a secure application that deletes the information received from the server as soon as it has been used. We denote a voter by V, their computer by C_V, and their mobile phone by M_V. For simplicity and generality, and since we focus on the client-side, we model the voting servers as a single protocol participant, denoted by S.

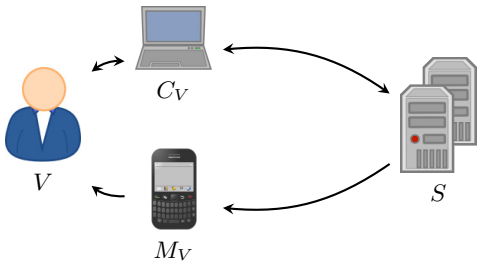

Figure 1: Interaction between voter and authorities

We define n as the number of eligible candidates, and $\mathsf{Cand} := \{0, \ldots, n-1\}$ as the set of *candidate identification numbers* corresponding to the eligible candidates. The voting system S owns a key pair (sk, pk), and the public key pk is propagated to any computer used by a voter. Accordingly, there are encryption and decryption algorithms $\mathcal{E}(\cdot,\cdot)$ and $\mathcal{D}(\cdot,\cdot)$ to be used for the encryption of submitted votes.

3. MOBILE BUILDING BLOCKS

3.1 Humanly computable one-time pads

Our first building block is a form of pre-encryption of ballots that yields strong privacy guarantees, even in the face of corrupted devices or coercers.

The essential idea is the following. When a voter V wishes to cast a ballot, the voting system S sends a canonical mapping f from candidate names into the set Cand to the voter's computer C_V, as well as a randomly selected *one-time pad* $i \in \{0, \ldots, n-1\}$ to the voter's mobile phone M_V. The voter's computer thus presents the voter with candidate names and the associated candidate identification numbers. The voter is then asked to add the one-time pad i to the candidate identification number of their preferred candidate (modulo n), and send the resulting number back to the voting system through their computer. Alternatively, S can pre-compute $f(v) + i \mod n$ for all v, and send a corresponding table to M_V. Then V needs not perform any computation, but the message gets clumsier. This interaction is depicted in Figure 2, where v represents the voter's chosen candidate.

The underlying intention of this approach is that an attacker or a corrupted device needs to learn messages from both channels in order to break vote privacy. This particular pre-encryption presents several advantages: *(i)* it is easily computable by a human; *(ii)* it is as strong as a one-time-pad; and *(iii)* it is easy for a voter to lie to a coercer about the value i, such that, assuming that a coercer cannot eavesdrop on the mobile channel, a voting scheme implementing this ap-

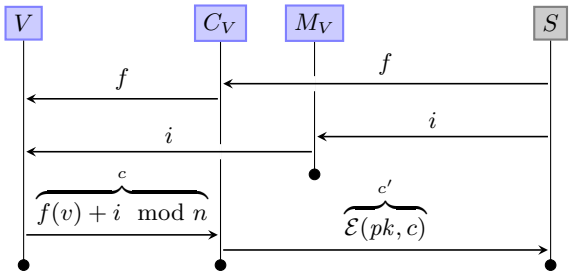

Figure 2: Ballot submission with one-time pads

proach obtains a form of coercion resistance. More precisely, this mechanism yields the following privacy-type guarantees.

Vote privacy. Even in the case where a voting computer is corrupted, it will not learn how a voter voted. Indeed, while the computer knows the mapping f from candidates to identification numbers and the value c cast by the voter, it does not learn the value i that was sent to the voter's mobile. This value effectively works as a one-time pad, and hence each vote is equally probable from the computer's perspective. This is true assuming that the mobile device is honest and does not collude with the computer, and that the computer is not able to otherwise eavesdrop on the channel between S and M_V. Conversely, if the computer is honest but the mobile is not, then the mobile does not learn the voter's vote, assuming that it cannot eavesdrop on the channel between S and C_V, since it simply does not learn the value c.

Coercion resistance. This property states that even when a voter interacts with a coercer during the voting process, the coercer cannot ensure that the voter followed his instructions [11]. To achieve coercion resistance, it is necessary that the voter can, at some point during the voting process, communicate with election authorities via an untappable channel. For instance, JCJ [11] assumes an untappable channel during the registration phase between the voter and a trusted registrar, while the scheme proposed by Lee [13] assumes an untappable channel between the voter and an external, tamper-resistant device owned by the voter (both schemes have been shown to be coercion resistant [4, 7]). In our case, we assume that the communication on one of the two channels between voter and authorities remains secret to the adversary. Let us first assume that the mobile is honest and that a coercer cannot eavesdrop on the mobile channel. Then, a voter could easily fool a coercer by pretending that the value i was the one that would have resulted in the vote demanded by the coercer. As a result, a coercer cannot be sure whether the voter is telling the truth. We note that additional mechanisms should be implemented to ensure that the message sent to the mobile is securely deleted after the voter has seen its content (such as a specialized app required for receiving or decrypting the message), so that the voter cannot be forced into revealing the value i. Similarly as for vote privacy, if we conversely assume that the mobile is corrupted, but the computer is honest and the coercer cannot eavesdrop on the channel between S and C_V, then coercion resistance easily follows since a coercer never learns the voter's encrypted ballot, so the voter can lie about the ciphertext they sent.

Vote independence. This notion means, among other things, that a vote cast by a voter cannot be copied by someone observing the messages sent and received by that voter [8]. In our scenario, assuming that an observer cannot eavesdrop on a voter's mobile channel, then he cannot copy that voter's vote. Indeed, if the observer simply cast the very same ciphertext as the other voter, then, as the observer would himself obtain an independent, random one-time pad i, he would cast a vote for any candidate with equal probability. In the case where the phone is corrupted, but the computer is honest and the channel between system and computer is untappable, the argument for vote independence is analogous to the one for vote privacy and coercion resistance.

As we see, this building block yields strong privacy guarantees against eavesdroppers on one of the channels. Yet, it clearly does nothing about the vote's integrity. For instance, it would be easy for a corrupted computer to perform randomization attacks. In order to ensure vote integrity, we investigate another mobile mechanism: return codes.

3.2 Return codes

Return codes are human-readable, personalized and unforgeable codes sent to a voter via an out-of-band channel – such as a voter's mobile phone – that allow a voter to verify that their vote was correctly recorded by the system. The idea is not novel; it has been proposed as early as [1], and is indeed used in systems that are deployed in practice [10]. Here we mean to analyze the impact of return codes on security properties, and its interaction with the one-time pads discussed above.

We stress that return codes must be personalized and secret: Otherwise, it may be possible for an attacker who managed to alter a voter's electronic ballot to send back the return code corresponding to the original vote, making the voter believe that their vote was cast as intended. Furthermore, such a personalized table must be sent to the voter prior to the election, for instance along with their ballot paper or their credentials. Sending this table to either the voter's computer or their mobile during the voting process is not possible: Such an approach would be vulnerable to randomization attacks where a malicious device shuffles the association between candidates and return codes – in such a case, a vote could be maliciously altered on the way to the voting servers, and the associated return code sent from the servers would match the displayed return code of the intended candidate with non-negligible probability.

We therefore assume that a table of return codes was sent to each voter before the election, say, by mail. The main remaining problem is that on the one hand, the voting servers have to generate a return code that corresponds to a candidate, while on the other hand we would like to avoid that any of the voting servers learns how the voter voted before the tallying phase. Therefore, a reasonable approach seems to be to use an encryption scheme with homomorphic properties in order to compute return codes *inside* an encrypted vote. As we want to avoid having to trust the devices on the client side, we adopt a similar idea as that proposed in [10]: First, for ballot encryption we use the additive homomorphic encryption scheme ElGamal in some cyclic group G with generator g. The secret key sk is an element $x \in \{0, \ldots, |G|-1\}$, and the public key pk is the value $g^x \in G$. For each voter V, let $s_V \in \{0, \ldots, |G|-1\}$ be a secret exponent generated uniformly at random, and $d_V(\cdot)$ be a pseudo-random function that maps elements of G to a human-readable form (such as a string of 6 or so characters). Furthermore, assume that the voting system S is composed

of at least two servers S_1 and S_2. Neither server knows the decryption key x, but they each know a piece x_1 or x_2 such that $x \equiv x_1 + x_2 \mod |G|$. Additionally, S_1 knows the one-time pads sent to voters after submission of a ballot and the personalized secret exponents s_V, while S_2 knows all pseudo-random functions d_V. As in Section 3.1, assume that a pre-encrypted ballot has the form $c \equiv f(v) + i \mod n$. Then, the voter's computer C_V encrypts the ballot c by computing $\mathcal{E}(pk, c) = (g^r, g^{xr} g^c)$ (where r is randomness), and sends this to S_1. Note that since $i \in \{0, \ldots, n-1\}$, $f(v)+i \mod n$ is either $f(v) + i$, or it is $f(v) + i - n$. Hence, S_1 and S_2 will jointly compute two return codes, one for each case. First, S_1 computes both $w_1 = g^{xr} g^c g^{-i}$ and $w_2 = g^{xr} g^c g^{-i+n}$. Thus, either w_1 or w_2 equals $w := g^{xr} g^{f(v)}$, i.e., the encryption of the vote stripped of the one-time pad. Next, S_1 computes $(\hat{x}, \hat{w}) := ((g^r)^{s_V}, (w g^{-x_1})^{s_V}) = (g^{r s_V}, g^{x_2 r s_V} g^{f(v) s_V})$, i.e., S_1 homomorphically adds in the secret exponent s_V, but strips out the exponent x_1. Note that S_1 does not know whether $w = w_1$ or $w = w_2$, however it can perform the above computation twice, once using w_1 and once using w_2, resulting in two pairs $(\hat{x}, \hat{w}_1)$ and $(\hat{x}, \hat{w}_2)$ respectively. Accordingly, either $\hat{w} = \hat{w}_1$ or $\hat{w} = \hat{w}_2$. Now, S_1 sends $(\hat{x}, \hat{w}_1)$ and $(\hat{x}, \hat{w}_2)$ to S_2, which computes $\rho_1 := \hat{w}_1 \hat{x}^{-x_2}$ and $\rho_2 := \hat{w}_2 \hat{x}^{-x_2}$. Thus, either ρ_1 or ρ_2 equals $\rho := g^{f(v) s_V}$. Finally, S_2 sends the values $d_V(\rho_1)$ and $d_V(\rho_2)$ back to the voter's mobile M_V, who checks both codes against the personalized table sent to them prior to the election: This table maps all candidates v to the value $d_V(g^{f(v) s_V})$. If one of the codes matches, the voter can be sure that their vote arrived at the election authorities unchanged. Note that the other return code will either be $d_V(g^{(f(v)-n) s_V})$ or $d_V(g^{(f(v)+n) s_V})$ and therefore encrypt a vote *out of range*, i.e., it will simply be an invalid return code that does not correspond to a candidate.

Inalterability. This property essentially states that a ballot cannot be changed when sent to the voting system, i.e., it is a form of vote integrity. Return codes provide us with exactly the means we need in order to guarantee inalterability. If a network attacker were to exchange the ciphertext sent by a voter to the voting system S in transit, or a malicious voting computer encrypted a vote other than the one intended by the voter, both return codes computed by the servers S_1 and S_2 would be different from the one associated with the desired candidate on the voter's table. Then, a voter could either *(a)* complain to the authorities, or *(b)* not confirm their vote, as required by the protocol in consideration. Again, we emphasize that a message containing sensitive information such as return codes should be securely deleted after the voter has seen it, so as to ensure that the voter cannot reveal their ballot to a third party. In the case where the mobile is corrupted, it could display fake return codes to the voter. However, without the knowledge of the table, it would be unable to predict valid return codes, i.e., the fake return codes would be identified as such with high probability, and therefore strongly indicate that one of the voting devices is corrupted. In such a case, a voter might consider trying to fix their mobile, or using another mobile (with their original SIM card), and start over the vote casting process, in order to ensure that their vote is cast as intended.

Coercion resistance (continued). It is worth considering how return codes affect coercion resistance, as they, along with the table of return codes, can be used as a receipt to prove how a voter voted, in particular in the case where a mobile is corrupted and thus the return codes are not securely deleted after having been displayed to the voter, or are even sent directly to a coercer. If a coercer additionally gets hold of the original table of a voter, coercion resistance is clearly broken. However, without knowledge of the table the coercer would not obtain any information about how the voter voted. Furthermore, if asked for this table by the coercer, it would be easy for a voter to forge a table of return codes that would satisfy the coercer, e.g., by swapping the return code of their intended vote with the return code of the vote demanded by the coercer.

Concluding this section, return codes ensure a certain degree of vote integrity, as a network attacker or a malicious voting device cannot forge them. We recommend that a voter should confirm to the voting system that the return code was correct before the vote is actually counted. For this purpose, we suggest to use TANs.

3.3 Transaction authentication numbers

Transaction authentication numbers (TANs) have been used for quite some time by online banking services in order to increase the security of financial online transactions. Essentially, a TAN is a one-time password that the user must enter in addition to their usual credentials to authorize a specific transaction. Thus, they form a kind of two-factor authentication, intended to improve security over a simple one-factor authentication (such as for instance a password or a digital signature). Different methods for TAN distribution and generation are used in practice: TANs may be pre-distributed on indexed lists, sent to the voter's mobile, or generated by both sides separately using some secret information. In the spirit of exploring security properties arising from using mobile devices, we focus on the method where a TAN is sent to a voter's mobile phone.

The mechanism is straightforward. Upon reception of a vote from a computer C_V and verification of the voter's credentials (such as a digital signature), the voting system S generates a reasonably large random number (say, 6 or 8 digits). The received ballot is stored in a database along with the TAN and marked as *not fully authorized*. The TAN is then sent to the voter's mobile M_V (ideally along with return codes as discussed in Section 3.2). The voter enters this TAN into their computer and sends it back to S to confirm their ballot. The voting system now checks that the TAN matches the one stored in the database, and marks the associated ballot as *authorized*, to be counted in the tallying phase.

This mechanism improves the verification of a voter's eligibility in a voting protocol with respect to corrupted computers or stolen credentials. In particular, it improves the following security properties of e-voting protocols.

Eligibility. Requiring a TAN to confirm a vote in effect promotes a voter's mobile to an additional credential. This may even remove the need for a PKI established before the election (such as the need for each voter to have a registered signing key). Setting up such a PKI is difficult and costly, yet most protocols from the literature simply assume that one is in place. While this may be an acceptable assumption in high-stakes elections, a PKI is less realistic for small elections, say in small online communities. In a system without PKI, a voter would not be required to sign a ballot sent to S. Then, although an attacker might submit a ballot for another voter to the system, the attacker would not be able to confirm it without the correct TAN. This is assuming that an attacker

cannot listen on mobile channels. In practice, mobiles should only serve as an additional, but not as the only credential. This is particularly important if we assume that the mobile is corrupted (but the computer is honest). Then, such a corrupted mobile could not vote on the voter's behalf, as it would lack some secret information, such as a signing key known only to the computer. Note however that in the case where a mobile is corrupted, it may still perform a denial-of-service attack by not displaying the TAN, or displaying an incorrect one. Similarly as for the case where a mobile displays fake return codes, here a voter would have to either fix their mobile, or use another mobile with their original SIM card. Another possibility would be to allow voters to vote both electronically and with traditional paper ballots, where the traditional way takes precedence over electronically cast ballots – as is done in Norway [10]. Then, if a malicious phone prevents a voter from casting their ballot, that voter could still revert to paper ballots, as a backup solution.

Resistance against impersonation attacks. This notion deals with the problem that credentials, such as passwords or even signing keys, may be leaked to an attacker unbeknownst to the voter. For instance, they may get stolen when sent to the voter, or a corrupted computer may simply store them and vote on the voter's behalf. With mobile TANs, credentials are not limited only to something that a voter *knows*, but additionally to something that a voter *has*, in this case their mobile. For instance, in the Norwegian or Estonian protocols, a voter may re-cast an electronic vote, overriding a possible previous electronic vote [6, 10]. A malicious computer C_V in possession of a voter's credentials may thus re-cast a ballot after the voter V has already cast their ballot. Similarly, C_V may alter the voter's intended ballot in the first place. Although return codes provide some protection here, they require the voter to take further action in such situations, such as re-voting on another device or complaining to the authorities. Using TANs as an additional line of defense would remove that burden from voters and prevent corrupted computers or attackers who stole a voter's credentials from voting on the voter's behalf. Clearly, if an attacker stole a voter's credentials *and* corrupted that voter's mobile, he could break even this defense, but such an attack would clearly be quite involved and unlikely to pass unnoticed when performed on a large scale.

4. CONCLUSION

We have proposed three practical mechanisms that help to improve the security of remote e-voting protocols by exploiting the possibility to communicate with voters on distinct channels, thanks to today's widespread availability of mobile phones. These mechanisms, despite their simplicity, can provide strong security guarantees even when one of the voter's devices is corrupted, a problem that has been mostly ignored in previous protocols. Combining these techniques with existing voting systems, we may finally be close to obtaining truly practical, secure remote e-voting systems.

Acknowledgments

We thank the anonymous reviewers for their comments. This work was supported by the German Ministry for Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA) and the initiative for excellence of the German federal government.

5. REFERENCES

[1] E-voting security study. CESG, United Kingdom, July 2002. Issue 1.2.

[2] B. Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th USENIX Security Symposium – USENIX Security 2008*, pages 335–348. USENIX Association, 2008.

[3] B. Adida, O. D. Marneffe, O. Pereira, and J.-J. Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *Proceedings of the 2009 Conference on Electronic Voting Technology Workshop/Workshop on Trustworthy Elections – EVT/WOTE 2009*. USENIX Association, 2009.

[4] M. Backes, C. Hriţcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium – CSF 2008*, pages 195–209. IEEE Computer Society, 2008.

[5] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *Proceedings of the 29th IEEE Symposium on Security and Privacy – S&P 2008*, pages 354–368. IEEE Computer Society, 2008.

[6] E. N. E. Committee. Estonian i-voting system: General description. Online, 2010. http://www.vvk.ee/voting-methods-in-estonia/engindex/reports-about-internet-voting-in-estonia/.

[7] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, Dec. 2009.

[8] J. Dreier, P. Lafourcade, and Y. Lakhnech. Vote-independence: A powerful privacy notion for voting protocols. In *Foundations and Practice of Security: 4th Canada-France MITACS Workshop – FPS 2011*, pages 164–180. Springer, 2011.

[9] E. Gerck, C. A. Neff, R. L. Rivest, A. D. Rubin, and M. Yung. The business of electronic voting. In *Financial Cryptography, 5th International Conference – FC 2001*, pages 234–259. Springer, 2001.

[10] K. Gjøsteen. The norwegian internet voting protocol. In *E-Voting and Identity: 3rd International Conference – VoteID 2011*, pages 1–18. Springer, 2012.

[11] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society – WPES 2005*, pages 61–70. ACM, 2005.

[12] E. Kalige and D. Burkey. A case study of eurograbber: How 36 million euros was stolen via malware. Online, Dec. 2012. http://www.cs.stevens.edu/~spock/Eurograbber_White_Paper.pdf.

[13] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Information Security and Cryptology – ICISC 2003*, pages 245–258. Springer, 2004.

[14] R. Oppliger. E-voting auf unsicheren plattformen. *DIGMA – Zeitschrift für Datenrecht und Informationssicherheit*, 8(2):82–85, June 2008.