

Gesellschaft für Informatik (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into the fields of

- Seminar
- Proceedings
- Dissertations
- Thematics

current topics are dealt with from the fields of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure the high level of the contributions.

The volumes are published in German or English

Information: <http://www.gi-ev.de/service/publikationen/lni/>

ISSN 1617-5468

ISBN 978-3-88579-225-3

The 2008 Conference on Electronic Voting took place in Castel Hofen near Bregenz at the wonderful Lake Constance from 6th to 9th August.

This volume contains 17 papers selected for the presentation at the conference out of more than 30 submissions. To assure a scientific quality, the selection was based on a strict and anonymous double-blind review process.



Robert Krimmer, Rüdiger Grimm (Eds.): Electronic Voting 2008

# GI-Edition

## Lecture Notes in Informatics

**Robert Krimmer, Rüdiger Grimm (Eds.)**

### 3<sup>rd</sup> international Conference on **Electronic Voting 2008**

**Co-organized by Council of Europe,  
Gesellschaft für Informatik and E-Voting.CC**

**August 6<sup>th</sup>- 9<sup>th</sup>, 2008  
In Castle Hofen, Bregenz, Austria**

# Code Voting With Linkable Group Signatures

Jörg Helbach<sup>1</sup>, Jörg Schwenk<sup>2</sup>, Sven Schäge<sup>3</sup>

Chair for Network and Data Security  
Ruhr-University Bochum  
Universitätsstr. 150  
D-44780 Bochum

<sup>1</sup>[joerg@helbach.info](mailto:joerg@helbach.info)

<sup>2,3</sup>[{joerg.schwenk|sven.schäge}@rub.de](mailto:{joerg.schwenk|sven.schäge}@rub.de)

**Abstract:** Code Voting is an appropriate technology to deal with the “Secure Platform Problem” [Riv02]. However, code voting as proposed by Chaum [Cha01] is vulnerable to vote selling and other flaccidities. In this paper we describe the vulnerabilities of code voting and propose to extend code voting to prevent vote selling. For this purpose we combine code voting with linkable group signatures and vote updating. We analyze the security properties of this new approach.

## 1 Introduction

Regarding remote online voting systems one of the major issues is the security of the voting client, i.e. the personal computer of the voter, as it cannot be considered to be trustworthy. Due to this fact in 2002 Ronald Rivest coined the term “Secure Platform Problem” [Riv02]. Even though different cryptographic voting protocols exist, the problem is that the voting client could be infected with malicious software, which is nowadays a widespread problem. Some estimates say that between 15% and 25% of all computers on the Internet are infected with malware bots [Web07], i.e. they have been under the complete control of an adversary. Hence, the voter cannot be sure that his electronic ballot is submitted faultless and unmodified to the voting server. Some methods for resolving this problem have been analyzed in [Opp02]. A good approach to overcome this problem is to use code voting as introduced by David Chaum in 2001 [Cha01]. Instead of a candidate's name, the voter only submits a voting transaction number (voting TAN) to the voting server. There is no correlation between the chosen candidate and the voting TAN on the voting client. So even if malware is installed on the client, it cannot identify the decision of the voter.

In this paper we will describe the code voting approach in detail and show that it is vulnerable against vote selling and denial of service attacks regarding the voting client. We then propose to improve code voting to deal with those vulnerabilities. Furthermore we combine code voting with vote updating and linkable group signatures. At last we describe the security properties of the new introduced approach.

## 2 Election Requirements

### 2.1 Security Requirements

In general voting systems used for (political) elections have to be free, universal, secret and equal. Much research has been done to adopt those requirements to remote online voting systems. Regarding Germany respectively Europe two important studies are the catalogue of requirements of the Physikalisch-Technische Bundesanstalt (PTB) [PTB04] and the recommendation of the Committee of Ministers of the Council of Europe [CoE04]. In 2006 Grimm et.al. analyzed those requirements and developed a protection profile for non-political elections according to the Common Criteria [GKM+06].

Summarized one can specify the following list of security requirements, which is not intended to be complete or comprehensive:

- Completeness and soundness of the Internet voting protocol(s),
- Correctness of the results
- Authenticity of both the voter (or the voting client acting on behalf of the voter, respectively) and the voting server,
- Secrecy of the ballots (including, for example, anonymity of the voter),
- Integrity of the ballots (including, for example, protection against malicious software)
- Non-duplication of the ballots,
- Availability and reliability of the voting process (including, for example, protection against denial of service attacks)

Even though single of those requirements are easy to fulfil, it is quite difficult to achieve all requirements concurrently, for some of them are contradictory. Furthermore appropriate cryptographic methods exist to deal with single requirements. E.g. to guarantee the secrecy of the ballot, one can use asymmetric encryption technologies. But as the encryption has to be computed on the voter's local client computer (in case of a remote online voting system), it is possible that malicious software forges the encryption process. As the local voting client is an important part of a remote online voting system and malware is an increasing problem on personal computers, it is difficult to ensure the client's security and integrity. As mentioned, therefore, in 2002 Ronald Rivest introduced the term "Secure Platform Problem" [Riv02]. We think that code voting, which we will describe in the next section is one (if not the only one) approach that works on a large scale.

## 2.2 Other requirements

Additionally to the security requirements there are further requirements, which a (electronic) voting system has to or should fulfil.

As mentioned above a political election has to be free, i.e. the voter must be able to vote for his favoured candidate without the fear of oppression or other disadvantages. The secrecy of the voter's ballot protects the freeness of his or her vote. Due to these facts a vote has to be anonymous, i.e. an attacker must not be able to correlate a (intercepted) ballot to a voter. Furthermore the voter must not be able, voluntary or nonvoluntary, to prove his vote to a third person to prevent vote selling or coercion of the voter. In the literature this property is named receipt-freeness.

Another relevant property of voting systems is the verifiability of the election process. In our democracy it is very important that the voter trusts this process and its result. This trust is often addicted to the possibility to check the election process in general and the calculation of the tally in particular. We distinguish between two kinds of verifiability, individual and universal verifiability. A voting system is individual verifiable, if the voter can check that his or her ballot has been computed in the tally correctly. Certainly the voter must be the only one, who can check his own vote. A voting system is universal verifiable, if it is individual verifiable and additionally all voters can check that the tally was calculated correctly. The particular challenge regarding individual and universal verifiability is not to compromise the receipt-freeness of the voting system.

## 3 The Secure Platform Problem

There is a simple attack against most of the remote voting systems proposed in the literature: If the attacker is able to control the communication channel between PC and voter, he can present the voting options in a different order, intercept the choice of the voter and redirect it to a voting option of his choice. This approach is similar to recent attacks on online banking systems [Gri03] [SW07] [LS07]. All cryptographic primitives employed can protect the voter's choice only from the point where it has been entered into the PC. There are two major options to solve this problem:

- Securing the PC against malware, e.g. by using Trusted Computing techniques.
- Using a separate channel from the voting authority to the voter, e.g. by snail mail, or by using a stand-alone security token.

We propose to use code voting [Cha01], where the separate channel is instantiated by snail mail. However, this scheme is vulnerable to vote selling attacks, so to be able to use this scheme in political elections, we have to add additional functionality. This additional functionality will be a linkable group signature scheme that is executed inside the untrusted PC. This may at first seem contradictory, but the adversary does not gain an advantage by manipulating the GS scheme, as long as the private key of the group member remains secret.

## 4 Code Voting

The term code voting was introduced in 2001 by David Chaum [Cha01]. Each eligible voter is issued a code sheet as shown in table 1.

Candidate	Voting TAN
Alice	738747987
Bob	983293774
Clark	192851911
...	...

Table 1: Printed Code Sheet.

As with many remote online voting systems, the voter connects to the remote voting server, but instead of submitting the name of his or her favoured candidate the voter only enters the appropriate voting TAN, i.e. if a voter wants to vote for Bob he just enters 983293774 into the voting application. Using code voting we assume

- a trustworthy voting authority, which issues a valid code sheet to every eligible voter, and
- the according voting servers and databases to be reliable and secure.

With the two additional requirements

- all voting codes are random and unique for every code sheet and every candidate, and
- the code sheets must not be distributed by electronically means

we can consider code voting secure against active and passive attacks [HS07]. In a passive attack the adversary can read the submitted voting TAN. As this voting TAN is random and there is no correlation between the voting TAN and the chosen candidate, the best the attacker can do is guessing the vote. In an active attack the adversary not only can read, but also could modify or discard the submitted voting TAN. For the attacker neither knows the corresponding candidate nor can calculate a new voting TAN, the best he can do is guessing again. However, code voting is vulnerable to unnoticeable denial of service attacks, as the attacker could prevent the voting client from submitting the chosen voting TAN to the server either by simply discard the voting TAN or modifying the voting TAN, so that it is invalid. The voter has no possibility to discover that his vote wasn't delivered to the voting server. For this purpose a possible extension of the basic code sheet is to introduce a confirmation TAN, which is displayed after the voting TAN was delivered correctly to the voting server as shown in table 2.

<b>Voting TAN</b>	<b>Candidate</b>	<b>Confirmation TAN</b>
738747987	Alice	332676873
983293774	Bob	676476488
192851911	Clark	301287123
...	...	...

Table 2: Printed code sheet with confirmation TAN.

After the voter entered the voting TAN and it was successfully delivered to the voting server, the server responds with the confirmation TAN. This confirmation TAN is also random and unique for every code sheet and every candidate, so the voter has evidence, that his chosen voting TAN was delivered correctly to the voting server. However, one has to think about the voter's claiming possibilities in case of a faulty or missing confirmation TAN. With this solution one possible (averaging) attack is to prevent the voter from voting by means of a denial of service attack, i.e. the voter enters the chosen voting TAN, but malware on the client computer prevents from submitting the voting TAN to the voting server. Then the malware either answers with a random, faulty confirmation TAN or doesn't answer at all. We then can assume, that the voter would enter another voting TAN (in particular when vote updating is allowed) to check, if his code sheet is correct. Presumably, the voter would then enter a voting TAN corresponding to an outsider candidate, which the malware allows to pass. However, this problem that neither the sender nor the receiver of a TAN could know, if his message was delivered successfully, is comparable to the two army problem [AEH75][Gra78], which illustrates the problems and challenges of attempting to coordinate an action of two parties over an unreliable communication channel. However, though one can show that the two army problem has no solution, often as a solution approach a three-way handshake is used, as e.g. used in TCP. According to this approach we propose a 3-step scheme by adding a third TAN, the finalization TAN (see table 3). The voting server only counts the vote, if the finalization TAN has been entered by the voter. With the attack described above, one can assume that the voter wouldn't enter the finalization TAN, if he or she doesn't receive the correct confirmation TAN.

<b>Voting TAN</b>	<b>Candidate</b>	<b>Confirmation TAN</b>	<b>Finalization TAN</b>
738747987	Alice	332676873	442367810
983293774	Bob	676476488	123456789
192851911	Clark	301287123	520172861
...	...	...	...

Table 3: Printed code sheet with confirmation and finalization TAN.

## 5 Vote Selling

However, even with a finalization TAN, code voting is vulnerable to vote selling, as the voter could simply sell the code sheet or a scanned copy thereof to an attacker. Even if vote updating is allowed and the vote seller tries to update his or her vote, he or she is racing with the vote buyer, and the vote buyer can arrange to be almost certain to win the race, since the vote buyer can re-perform the update as many times as needed. We have to assume that the vote buyer probably has more resources and patience than the vote seller, and, for instance, can automate the process of repeatedly sending updates.

In the following sections we will improve code voting with group signatures and vote updating to deal with vote selling.

## 6 Code Voting With Linkable Group Signatures

### 6.1 Group Signatures

In 1991 Chaum and van Heyst presented the concept of a group signature scheme [CH91]. A group signature is used to allow every member of a group sign messages on the group's behalf. In most cases those signatures are anonymous, i.e. it is not possible to identify which member of the group has signed a particular message. In addition, one cannot check if two signed messages were signed by the same group member. However, only a designated group manager exists who manages the membership list of the group and who can reveal the identity of the signer of a message.

### 6.2 Procedures in Group Signature Schemes

The group signature setting comprises three parties, namely the group manager  $M$ , the group members  $u_i$  and one or more verifiers  $w_j$ . In a group signature scheme, these parties participate in several polynomial-time algorithms (Fig. 1)<sup>50</sup>:

- **GMKEY**: a probabilistic algorithm that generates the private keys  $isk$  (issuing key) and  $opk$  (opening key) for  $M$  together with a group public key  $gpk$ .
- **GUKEY**: a probabilistic key generation algorithm that provides each user  $u_i$  with a public key pair  $(upk_i, usk_i)$ . The key  $upk_i$  is also referred to as membership key or pseudonym and should only be known to  $u_i$  and  $M$ .
- **JOIN**: an interactive algorithm in which  $M$  computes a membership certificate  $v_i$  on  $upk$  for user  $u_i$  using  $isk$ . Using  $v_i$ ,  $u_i$  can prove to any verifier  $w_j$  that he is a member of the group administered by  $M$ .

---

<sup>50</sup> We here omit the JUDGE procedure, assuming that each identity determined by the OPEN algorithm is accompanied with a proof of that fact.

- **SIGN**: a probabilistic algorithm in which  $u_i$  generates a signature  $s$  on an arbitrary message  $m$  using a membership certificate  $v_i$  and a secret key  $usk_i$ . Essentially, for a group signature scheme, no party can learn from  $s$  which  $v_i$  was used to generate it nor determine if any two signatures  $s$  and  $s'$  have been generated by the same group user.
- **VERIFY**: given  $gpk$ ,  $m$ , and  $s$  a verifier  $w_j$  can use this deterministic algorithm to determine if a received signature  $s$  has actually been signed by a group member.
- **OPEN**: given  $opk$  and a message  $m$  with a corresponding group signature  $s$ , this deterministic algorithm can identify the originator of  $s$ .

A secure group signature scheme must guarantee the following (informal) security properties<sup>51</sup> [ACJT00]:

- **Correctness**: A group signature  $s$ , which has been correctly generated by a group user, is always accepted by a verifier.
- **Unforgeability**: only group users can generate valid group signatures.
- **Anonymity**: no one (except  $M$ ) can learn the identity of the originator of a valid group signature.
- **Unlinkability**: no one (except  $M$ ) can decide whether two signatures have been issued by the same user.
- **Traceability**: the group manager  $M$  can associate all valid group signatures with their originator.
- **Coalition-resistance**: a set  $C$  of malicious group users cannot work together to successfully create valid group signatures, which are associated to a user  $u_i$  who is not a member of  $C$ .

In [BSZ05] the security requirements of group signature schemes are reduced to just four basic properties (including correctness). Each property is then formalized in an attack experiment. Accordingly, a group signature scheme is called secure with respect to a certain security property if no polynomial-time attacker can win the corresponding experiment with a non-negligible probability.

---

<sup>51</sup> We remark that some authors even consider further security properties.

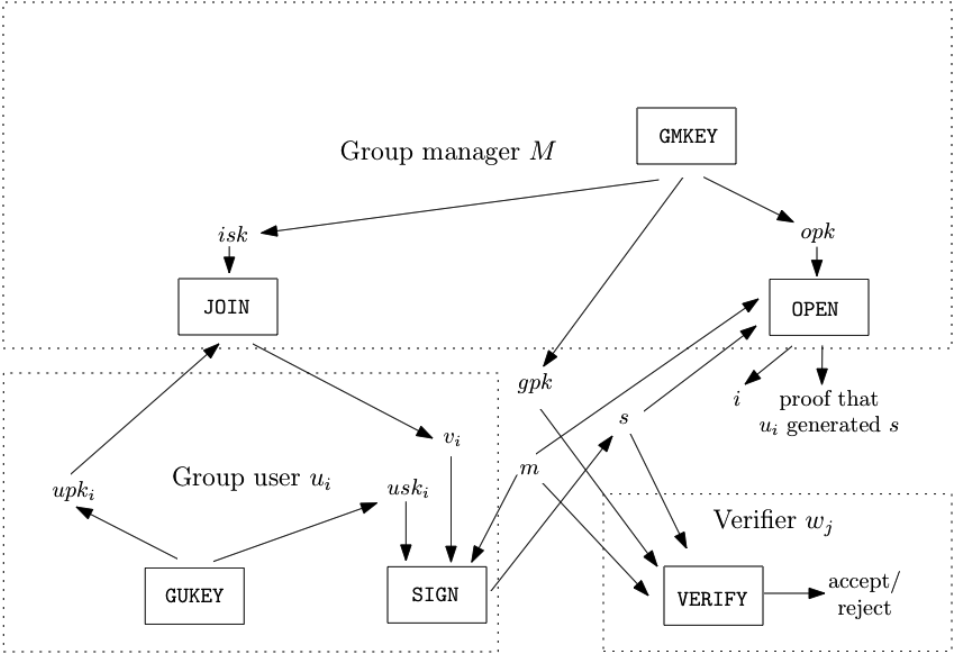


Figure 1: Group signature scheme.  $gpk$ : group public key;  $isk$ : (private) issuing key;  $opk$ : (private) opening key;  $usk_i$ : (private) user key;  $upk_i$ : membership key;  $m$ : message to be signed;  $v_i$ : membership certificate;  $s$ : group signature on  $m$ ;  $i$  user identity

### 6.3 Signatures of Knowledge

Signatures of knowledge are among the most important building blocks for group signature schemes. They are based on zero-knowledge protocols in which a prover can convince a verifier that he possesses a certain secret without revealing any information on that secret. Basically, usual 3-move zero-knowledge proofs of knowledge are made non-interactive using the Fiat-Shamir heuristic by replacing the verifier in the first two protocol steps with a hash function. Accordingly, the output of the hash function is interpreted as one or more challenges for the prover. The input to the hash function consists of the random commitments of the prover along with additional public information. In a signature of knowledge, these values are concatenated with the message to be signed. Signatures of knowledge can be proven secure in the random-oracle model. As a result, a signature of knowledge convinces a verifier that its issuer knows a certain secret while at the same time not revealing any information on that secret. Similar to [CS97] we denote signatures of knowledge rather descriptive than technical. According to this, a signature of knowledge of the fact that the issuer knows, for example, the discrete logarithms of  $y$  to the base  $g$  is denoted as:

$$SK_{\{(\alpha) : y = g^\alpha\}}(m).$$

Such signatures of knowledge can easily be constructed using Schnorr signatures [Sch91]:

Let  $H: \{0,1\}^* \rightarrow \{0,1\}^k$  be a collision-resistant hash function with a  $k$ -bit output and  $G = \langle g \rangle = \langle h \rangle$  be a cyclic group of prime order  $p$ . Then, a signature of knowledge of the above fact is a pair

$$(c, d) \text{ in } \{0,1\}^k \times \mathbb{Z}_p^*$$

satisfying

$$c = H(m || y || g || g^d y^c).$$

Signatures of knowledge can also be used to prove more complex statements about secrets, like

$$SK\{(a, \beta): y = g^a \text{ and } z = h^\beta\}(m)$$

$$SK\{(a, \beta): y = g^a \text{ or } z = h^\beta\}(m)$$

$$SK\{(a): y = g^a \text{ and } a \text{ is in } [A, B]\}(m).$$

The security properties of signatures of knowledge make them suitable for the design of SIGN algorithms. To show that he is a group user of  $M$ 's group,  $u_i$  has to prove that he (i) possesses a group membership certificate  $y_i$  issued by  $M$  and (ii) that he knows the private key  $usk_i$  corresponding to the public key  $upk_i$  certified in  $v_i$ . By showing his membership certificate or his membership key directly to a verifier, the user would make his signatures linkable. Using signatures of knowledge  $u_i$  can show possession of both values without actually revealing them. Essentially,  $u_i$  exploits that signatures of knowledge can be randomized (just like interactive zero-knowledge proofs of knowledge) by the prover. The group user only has to choose a new random commitment (corresponding to the first protocol move in an interactive zero-knowledge proof) every time he issues a group signature. In this way  $u_i$  can guarantee that no two signatures are equal, thus making the group signature scheme unlinkable.

## 6.4 Linkable Group Signatures

In 1997 Camenish and Stadler introduced the first efficient group signature scheme. Using this group signature scheme the length of the public key is independent from the size of the group. Even if a new member joins the group it is not necessary to calculate a new public key. Furthermore, in this scheme it is possible to assign the two different roles of the group manager (issuer of membership certificates and opener of group signatures) to different parties, which is a very desirable property regarding electronic voting systems. Since then, a large number of group signature schemes have been proposed [GW07]. We will show how to change such schemes to linkable GS schemes using the high-level description from [CS97]. Our starting point is to force each group user not to randomize his signatures of knowledge:

- The group manager  $M$  computes a key pair  $(\text{sig}_M, \text{ver}_M)$  of a digital signature scheme, and a public key encryption key pair  $(\text{enc}_M, \text{dec}_M)$ , and publishes the two public keys.

- Alice joins the group by choosing a random value  $x$ , sending her membership key  $z=f(x)$  ( $f$  a one-way function) in an authenticated way to  $M$ , and receiving in return her membership certificate  $v=\text{sig}_M(z)$ .

- Alice signs a message  $m$  by encrypting  $(m,z)$  using the group managers encryption key, i.e.  $d=\text{enc}_M(m,z)$ . (Note: We omit the random number here to make the GS linkable.) She computes a signature of knowledge that she knows the values  $x'$  and  $v'$  satisfying the following equations:  $d=\text{enc}_M(m,f(x'))$  and  $\text{ver}_M(v',f(x'))=\text{true}$ .

To protect the private key  $(x,z,v)$  against the attacker controlling the PC, this key can e.g. be bound to a TPM chip (which is much easier than to secure the whole platform using TPM technology), or it can be stored on a smart card (e.g. an electronic passport).

## 6.5 Vote Updating

To prevent vote selling in some voting systems, vote updating is used. That is, the voter could cast his or her ballot as often as he or she wants to, but only the last cast ballot is computed in the tally. The basic idea is that even if a voter sells his ballot to an attacker, he could easily update his or her vote. Hence the vote buyer never can be sure that the vote seller will not update his vote, after he has proven his choice to the vote buyer. E.g. the Estonian voting system, which was employed for the first political election over the Internet, uses vote updating [Est05]. Besides the advantages some disadvantages also exist. These advantages and disadvantages that are also different types of vote updating, are not further addressed in this paper, but are analyzed and discussed in [VG06]. However, we think that vote updating is a good method to prevent vote selling, but cannot be the only measure and therefore has to be facilitated by other measures [OSH08]. In this paper we will use vote updating as a part of a measure against vote selling, independent from the type of implementation of vote updating.

## 6.6 Improved Voting Scheme

To deal with vote selling and the secure platform problem, we propose to improve code voting. We assume a trustworthy voting authority, which consists of representatives of all parties that are supervising each other<sup>52</sup>. We further assume a group signature scheme as described in section 6.4. The voting authority is divided into different groups, which are responsible for the following tasks:

- Printing and issuing the code sheets to the eligible voters.

---

<sup>52</sup> In the literature, this property is called Separation of Duties.

- Operating the voting servers and the according databases, which we assume to be reliable and secure.
- Managing the group signature scheme by issuing the private keys to the eligible voters.
- Managing the group signature scheme by opening the signed voting TANs, i.e. verifying that every eligible voter only casts one ballot.

Each member of the voting authority should only belong to one of those groups.

The improved voting scheme works as follows: Prior to the election, each eligible voter is issued a private key according to the group signature scheme. Additionally, in a second step, the voting authority prints code sheets as seen in table 3. We assume that for every voter and every candidate the voting TANs, the confirmation TANs and the finalization TANs are randomly chosen using a good PRNG algorithm. Since the printing procedure links the voting TANs to the candidates, this process has to be monitored not only by the members of the code sheet issuing group, but by all voting authority members. For that purpose the different parties and their representatives in the voting authority then can check a control sample if the correlation between voting TAN and candidate is correct for the valid code sheets.

In a third step the valid code sheets are shuffled, put into anonymous envelopes and then they are distributed to the eligible voters. After the election has been started the voter connects to the voting server and enters the voting TAN for the desired candidate into the voting software and signs it with his private key according to the linkable group signature scheme. This signature could include information about e.g. the electoral district. The signed voting TAN is sent to the voting server over a MIX net. After the voting server has answered with the correct confirmation TAN, the voter approves his vote with the (signed) finalization TAN. In the improved code voting scheme we allow vote updating, i.e. every voter could submit several valid voting TANs to the voting server, but only the last submitted voting TAN approved with the corresponding finalization TAN counts in the tally. With the aid of the group signature, the responsible group of the voting authority ensures that a single voter can cast only one valid voting TAN regardless of how many code sheets he may have bought.

Therefore, this voting authority group opens the group signature to check if the voter already cast a ballot<sup>53</sup>. The finalized voting TAN is stored in the database, where older required voting TANs will be removed. If the voter gets either no or a faulty confirmation TAN, the client may be infected with malicious software, and he may vote again using another voting client, if the group signature scheme is transferable<sup>54</sup>.

---

<sup>53</sup> For further research it would be interesting to analyze if threshold schemes could be applied in conjunction with the linkable group signature scheme, so that  $m$  out of  $n$  group managers are needed to open a signature.

<sup>54</sup> This is e.g. the case if the private key is stored on a smart card.

It is an open question whether at the end of the election the voting authority should publish the submitted voting TANs: since they are checkable also by a coercer, even if vote selling is impossible, the adversary may control the voting decision of certain voters.

## 7 Security Properties

A passive adversary is only able to attack the secrecy of an election. He can observe the TANs entered into the web browser. Since those TANs were chosen at random, the best an attacker can do is guess the vote. Additionally, in our voting scheme the voter sends his signed voting TAN to the server. As his or her vote is sent over a MIX net, an allocation between IP address and the submitted voting TAN is not possible. Even though malware on the voting client could just read the voting TAN, it cannot identify the chosen candidate. So our voting scheme is secret.

Further, our voting scheme is equal because the group signature is linkable by the group managers, i.e. for every eligible voter only one signed voting TAN is counted in the tally. If the voting scheme doesn't publish the submitted voting TANs, the proposed voting scheme is receipt free, but it is not verifiable. Because our voting scheme uses linkable group signatures, a vote buyer can only cast as many ballots as he has different group signature keys. For that purpose, the group managers have to issue a private key, which a voter presumably would not give to an attacker, e.g. a private key according to an ePass.

## 8 Summary

In this paper we proposed to use code voting as a reasonable measure against the “Secure Platform Problem” that is a major threat to most of the proposed electronic voting schemes. As the code voting model is vulnerable against vote selling, we extended code voting using vote updating and linkable group signatures to prevent vote selling attacks regarding the voting client and described some security properties of the new approach.

## References

- [ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *Advances in Cryptology – CRYPTO 2000*, vol. 1880 of Lecture Notes in Computer Science, pages 255–270. Springer 2000.
- [AEH75] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some Constraints and Tradeoffs in the Design of Network Communications. In *ACM SIGOPS Operating Systems Review*, volume 9, pages 67–74, 1975.
- [BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *Topics in Cryptology - CT-RSA 2005*, volume 3376 of Lecture Notes in Computer Science, pages 136–153. Springer-Verlag, 2005.
- [Cha01] D. Chaum. Sure Vote: Technical Overview. In *Proceedings of the workshop on trustworthy elections (WOTE '01)*, <http://www.vote.caltech.edu/wote01/pdfs/surevote.pdf>, 2001.

- [CS97] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.
- [CvH91] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.
- [Cyb05] Cybernetica. General description of the estonian e-voting system, online available under <http://www.cyber.ee/english/services/eGovernment/evoting.html>, 2005.
- [GKM+06] R. Grimm, R. Krimmer, N. Meißner, K. Reinhard, M. Volkamer, M. Weinand, and J. Helbach. Security Requirements for Non-political Internet Voting. In *Proceedings of the 2nd International Workshop on Electronic Voting 2006*, volume 86 of *Lecture Notes in Informatics*, pages 203–212, 2006.
- [Gra78] Jim Gray. Notes on Data Base Operating Systems. In *Lecture Notes in Computer Science*, volume 60, pages 393–418, 1978.
- [Gri03] Roger A. Grimes. An SSL trojan unmasked. <http://www.infoworld.com/article/06/03/03/75970\100Psecadvise\1.html>, 2003.
- [HMR04] Volker Hartmann, Nils Meißner, and Dieter Richter. Online Voting Systems for Nonparliamentary Elections – Catalogue of Requirements. Technical Report PTB-8.5-2004-1, online available under [http://www.berlin.ptb.de/8/85/LB8\\_5\\_2004\\_1AnfKat.pdf](http://www.berlin.ptb.de/8/85/LB8_5_2004_1AnfKat.pdf), Physikalisch-Technische Bundesanstalt, April 2004.
- [HS07] Jörg Helbach and Jörg Schwenk. Secure Internet Voting with Code Sheets. In *EVoting and Identity, First International Conference, VOTE-ID E-Voting and Identity, First International Conference, VOTE-ID*, pages 166–177, 2007.
- [LS07] H. Langweg and J. Schwenk. Schutz von FinTS/HBCI-Clients gegen über Malware. In *Proceedings of D-A-CH Security*, pages 227–238, 2007.
- [oE04] Council of Europe. Legal, operational and technical standards for e-voting. [http://www.coe.int/t/e/integrated\\_projects/democracy/02\\_Activities/02\\_e-voting/01\\_Recommendation/Rec%282004%2911\\_Eng\\_Evoting\\_and\\_Expl\\_Memo.pdf](http://www.coe.int/t/e/integrated_projects/democracy/02_Activities/02_e-voting/01_Recommendation/Rec%282004%2911_Eng_Evoting_and_Expl_Memo.pdf), 2004.
- [Opp02] R. Oppliger. How to Address the Secure Platform Problem for Remote Internet Voting. In *Proceedings of the 5th Conference Security in Information Systems (SIS 2002)*, pages 153–173, [http://www.ifi.unizh.ch/~oppliger/Docs/sis\\_2002.pdf](http://www.ifi.unizh.ch/~oppliger/Docs/sis_2002.pdf), 2002. vdf Hochschulverlag.
- [OSH08] R. Oppliger, J. Schwenk, and J. Helbach. Protecting Code Voting Against Vote Selling. In *SICHERHEIT 2008 - Sicherheit, Schutz und Zuverlässigkeit*, volume 128 of *Lecture Notes in Informatics*, pages 193–204, 2008.
- [Riv02] R. Rivest. Electronic voting. In *Financial Cryptography '02*, volume 2339 of *Lecture Notes in Computer Science*, pages 243–268. Springer-Verlag, 2002.
- [Sch91] C. P. Schnorr. Efficient Signature Generation by Smart Cards. In *Journal of Cryptology*, volume 4, pages 161–174. Springer-Verlag, 1991.
- [SW007] Secure Works: Win32.Grams E-Gold Account Siphoner Analysis. <http://www.secureworks.com/research/threats/grams/>, 2007.
- [VG06] M. Volkamer and R. Grimm. Multiple Casts in Online Voting: Analyzing Chances. In *Proceedings of the 2nd International Workshop on Electronic Voting 2006*, volume 86 of *Lecture Notes in Informatics*, pages 97–106, 2006.
- [Wan07] Guilin Wang. Bibliography on Group Signatures, <http://icsd.i2r.a-star.edu.sg/staff/guilin/bible/group-sign.htm>, 2007.
- [Web07] T. Weber. Criminals 'may overwhelm the web', <http://news.bbc.co.uk/1/hi/business/6298641.stm>, 2007.