Protecting Code Voting Against Vote Selling

Rolf Oppliger rolf.oppliger@esecurity.ch

Jörg Schwenk joerg.schwenk@rub.de

Jörg Helbach joerg.helbach@sprint.de

Abstract: Code voting is an appropriate technology to address the secure platform problem of remote Internet voting. In this paper, we elaborate on the susceptibility of code voting to vote selling, argue that the situation is comparable to all-postal voting, and overview and put up for discussion some organizational and procedural protection measures—recast ballots, multiple code sheets, and powerful voting credentials. The measures are to sap the incentive to sell (or buy) votes and hence to protect code voting against vote selling. We also argue in favor of a clear separation of voter authorization and vote casting and discuss possibilities to do so.

1 Introduction

Elections and votes are fundamental processes for a democratic state and its (democratically legitimated) government. In the literature, the term *electronic voting* (or *e-voting*) is used to refer to elections and votes that are supported by electronic means. With the proliferation of the Internet, its use for e-voting has been proposed by many people (mainly politicians) as a way to make voting more convenient and—as it is hoped—to increase participation in elections and votes. The term *Internet voting* is therefore used to refer to election or voting processes that enable voters to cast their ballots over the Internet. This basically means that the ballots must be represented electronically, and that the electronic ballots must be transmitted to election officials using the Internet as a transport medium.

There are many possibilities to implement Internet voting, and *poll-site Internet voting*, *Kiosk voting*, and *remote Internet voting* are usually distinguished in the literature.[1] In this paper, we only focus on remote Internet voting, i.e., Internet voting where the voter (or a third party acting on behalf of him) uses his PC to cast a ballot over the Internet. From a security viewpoint, this is the most challenging possibility to implement Internet voting. In states that support absentee balloting, such as all-postal voting, any other form of Internet voting (i.e., poll-site Internet voting and Kiosk voting) is likely to fail (because they both require that the voter physically visits a voting place). In Europe, for example, a few states have started to employ remote Internet voting—be it in geographically restricted pilot projects (e.g., Switzerland) or for area-wide official use (e.g., Estland).

Against this background, it is possible and very likely that the use of remote Internet voting tends upwards, and that the security of remote Internet voting will become a major

[1] http://www.ss.ca.gov/executive/ivote/

issue. Security, in turn, has many aspects, and there are several security technologies, mechanisms, and services that can be used to address them. As argued in [Opp02], code voting (i.e., voting by providing randomly-looking codes instead of YES or NO in the case of a vote and candidates' names in the case of an election) is an appropriate technology to address the secure platform problem of remote Internet voting. But code voting is also susceptible to vote selling. In this paper, we elaborate on this susceptibility, argue that the situation is comparable to all-postal voting, and overview and put up for discussion some organizational and procedural protection measures—recast ballots, multiple code sheets, and powerful voting credentials. The measures are to sap the incentive to sell (or buy) votes and hence to protect code voting against vote selling. We argue in favor of a clear separation of voter authorization and vote casting and discuss possibilities to do so. In the rest of the paper, we summarize the security requirements of (remote) Internet voting in Section 2, introduce code voting in Section 3, bring up vote selling in Section 4, overview and discuss the organizational and procedural protection measures mentioned above in Section 5, and draw conclusions and give an outlook in Section 6.

2 Security Requirements

There are many investigations and studies that elaborate on the security of Internet voting in general, and remote Internet voting in particular (e.g., [Rub01]). The results all give evidence to the fact that security (including privacy and reliability) is among the most important preconditions for the successful deployment of Internet voting. The current paper ballot systems set a standard that is adopted as a security baseline for Internet voting. They represent certain tradeoffs between voter convenience and protection against fraud and abuse. It is generally required that elections and votes conducted over the Internet are at least as secure as the current paper ballot systems. In states that support absentee balloting in the form of all-postal voting, it is this technology that sets the security standard for Internet voting.

There are many lists of security requirements for (remote) Internet voting that can be found in the literature, and there is even a draft for a German e-voting protection profile for the Common Criteria.[2] The requirements comprise the completeness and soundness of the Internet voting protocol(s), the correctness of the results, the authenticity of both the voter and the voting server, the secrecy of the ballots (including the anonymity of the voter), the integrity of the ballots, the non-duplication of ballots, as well as the availability and reliability of the voting process as a whole. Some of these requirements are complementary and don't interact with each other, such as the integrity and non-duplication of the ballots. Other requirements are (or at least seem to be) contradictory. For example, one way to attest the correctness of a voting process is auditability, meaning that the entire voting process can be audited in some way. Auditability, however, may contradict to the secrecy of the ballots. In fact, there is a lot of research going on to address this apparent contradiction and to guarantee ballot secrecy and the correctness of the results simultaneously. Most of this research is based on verifiable secret sharing and secure multi-party computation.

[2] http://www2.dfki.de/fuse

Many security requirements of (remote) Internet voting can be addressed with existing technologies, mechanisms, and services. For example, there are many technologies that can be used to secure the server side. Examples include firewall technologies and intrusion detection or prevention systems. The authenticity of the voter and the voting server can be addressed with public key certificates. Similarly, the secrecy and integrity of the ballots can be guaranteed with cryptographic protocols, such as the SSL/TLS protocol. It is, however, important to note that the use of the SSL/TLS protocol protects the secrecy and integrity of the ballots only during the transmission over the Internet. The ballots are not automatically protected on the client or server side. In fact, additional security technologies, mechanisms, and services are required to protect the the ballots before and after they are transmitted over the Internet. There are additional risks for the secrecy of the ballots related to the use of spyware or remote system administration tools.

Due to the fact that a remote Internet voter uses his PC to cast a ballot and this PC may be subject to malware attacks, the insecurity of the client platform represents the major vulnerability (and Achilles heel) of remote Internet voting. Rivest coined the term *secure platform problem* to refer to the problem of protecting an inherently insecure client platform against malicious software and corresponding attacks [Riv01]. There are not many security technologies, mechanisms, and services that can be used to effectively address the secure platform problem of remote Internet voting. In fact, we think that code voting as introduced next is one (if not the only one) technology that may work on a large scale.

3 Code Voting

The term *code voting* is used to refer to an e-voting technology in which the voter casts his ballot by providing a *voting code* instead of YES or NO (in the case of a vote) or a candidate's name (in the case of an election). The voting code, in turn, looks like a random string. If the alphabet consists of all decimal digits $0 \ldots 9$, then the voting code represents a number. In general, however, any alphabet can be used and the voting codes can be arbitrarily long. To the best of our knowledge, the first code voting system was proposed by Chaum in 2001.[3] In this system, each voter is equipped with a code sheet (i.e., a sheet that itemizes all voting codes) and he must enter the appropriate voting codes to cast his ballot. An exemplary code sheet is illustrated in Table 1. If the voter wanted to vote for Bob, then he would enter "990234" (instead of "Bob").

Due to the fact that voting codes look like random strings, code voting effectively protects against passive and active attacks:

- In a passive attack, the adversary sees a voting code sent over a network (using, for example, a network management or system administration tool), and must then be able to tell whether this code represents YES or NO (in the case of a vote) or to which candidate the code actually refers to (in the case of an election). If the voting codes are chosen with a good random bit generator or a cryptographically secure

[3] The system was named SureVote and it was presented at the first Workshop on Trustworthy Elections (WOTE) in August 2001.

Candidate	Voting code
Alice	236412
Bob	990234
Carol	141290
...	...

Table 1: A code sheet with voting codes

pseudorandom bit generator (PRBG), then the best the adversary can do is guessing. In this case, observing the voting codes sent over the network does not help the adversary.

- In an active attack, the adversary does not only see a voting code sent over a network, but he can also manipulate it. For example, the adversary may employ malware or a client-side remote system administration tool to turn a voting code representing YES into a voting code representing NO (in the case of a vote) or a voting code of one candidate into a voting code of another candidate (in the case of an election). Again, if the voting codes are chosen with a good random bit generator or a cryptographically secure PRBG, then the adversary does not know the other voting codes, and hence the best he can do is again guessing.

In either case, the success probability of an adversary is not better than guessing. There are, however, two conditions that must be fulfilled:

1. As mentioned before, the voting codes must be random, i.e., they must be chosen with a good random bit generator or a cryptographically secure PRBG;
2. The code sheets must be personal and distributed out-of-band,[4] using, for example, a postal mail delivery service.

Also, it is important to note that code voting requires a modified voting behavior, and that there may be some legal constraints to consider (not addressed in this paper).

In spite of the fact that code voting as discussed so far is able to provide unconditional or information-theoretic security, it may still be the case that an (active) adversary simply deletes a voting code in transit. To protect against this attack, it may be worthwhile to have the server send back a verification code and have the voter verify this code. Table 2 illustrates an exemplary code sheet with voting and verification codes. Again, if the voter wanted to vote for Bob, then he would enter "990234" and wait for the server to send back the verification code "672345." If another verification code appears, then something illegitimate is going on and the voter is well advised to stop voting (needless to say that some dispute-resolving mechanisms must be put in place here).

[4] It is important that the code sheets must be provided outside the voter's PC. If the code sheets were inside the PC, then malicious software could get and use them to change the ballots. Also, the voting codes must be randomly or pseudo-randomly chosen from a sufficiently large set of possible values to make the probability that malicious software can correctly guess them sufficiently small.

Candidate	Voting code	Verification code
Alice	236412	124355
Bob	990234	672345
Carol	141290	045686
...	...	...

Table 2: A code sheet with voting and verification codes

Candidate	Voting code	Verification code	Confirmation code
Alice	236412	124355	252435
Bob	990234	672345	574546
Carol	141290	045686	124145
...	...	...	...

Table 3: A code sheet with voting, verification, and confirmation codes

If the voter verifies the verification code, then it makes a lot of sense to communicate the result of the verification step to the server (otherwise, the server does not know whether the result is correct). This is where the confirmation code comes into place. Table 3 illustrates an exemplary code sheet with voting, verification, and confirmation codes. In our example, the voter would confirm the successful verification of the verification code "672345" by sending the confirmation code "574546" to the server. At this point, there is no need to continue the recursion (and send more codes back and forth).

The bottom line is that there are many possibilities to implement code voting. In addition to casting a vote by simply entering a voting code, the voter may verify a verification code sent back from the server (to verify that he has casted the vote to an authentic server, and that the vote has been properly registered by the server). Also, the voter may acknowledge proper verification of the verification code by sending out a confirmation code. We think that four possibilities of implementing code voting are meaningful in practice:

- Voting code-only implementation;
- Verification code-only implementation;
- Voting and verification code implementation;
- Full implementation (i.e., voting, verification, and confirmation codes).

In a voting code-only implementation, the voter casts his ballot simply by sending a voting code to the server. In a verification code-only implementation, the voter casts his ballot as usual, but waits for a verification code sent back from the server. It is then up to the voter to verify this code. A verification code-only implementation is particularly interesting, because the voter has to minimally change his behavior, i.e., he can still enter YES or NO and only validate the verification number sent back from the server. This advantage,

however, may also be a disadvantage, because it is possible and very likely that some voters will not care about the validity of the verification codes. As its name suggests, a voting and verification code implementation employs voting and verification codes. Last but not least, a full implementation employs voting, verification, and confirmation codes. It goes without saying that this is the preferred choice from a security viewpoint, and that all other choices represent tradeoffs.

A practically relevant question refers to the length of the codes. Obviously, the length must make the probability to correctly guess a code sufficiently small. For example, if the number includes 10 binary digits (bits), then the probability of correctly guessing a code is $1/2^{10} = 1/1,024 = 0.000975562$. Because the codes cannot be verified offline (without access to the code sheets), this seems to be sufficient. 10 bits can be represented with $\log\ 2^{10} = \log\ 1,024$ decimal digits which is slightly more than 3 digits. Consequently, 4–5 decimal digits can be used to encode a code and some redundancy to detect errors (error detection is particularly important for voting and confirmation codes that are entered by the user). In theory, 10-bit code numbers can be generated with a random function. In practice, however, the code numbers will be generated with a pseudorandom function, such as a keyed one-way hash function (e.g., the HMAC construction specified in RFC 2104). This is not further addressed in this paper.

Last but not least, we note that a guessing attack may have an equalizing effect on the outcome of an election or vote. A candidate who only gets a few votes under normal circumstances may now get an average number of votes. This is because it is equally likely to guess the voting code of an unpopular candidate than to guess the voting code of a popular one. Hence, the outcome of an election or vote that is subject to a guessing attack may be equalized to some extent. Again, this point is not further explored in this paper.

4 Vote Selling

In the literature, the term *vote selling* is used to refer to the problem that occurs when a voter can prove to a third party how he voted. If, for example, the voter is given a paper printout of his vote (i.e., a receipt), then such a proof is simple and straightforward. In this case, the voter can sell his vote and prove that he acted accordingly by showing the receipt. Similarly, a voter can also be subject to coercion. Consequently, most e-voting schemes avoid the use of receipts in the first place—this is also true for remote Internet voting.

Against this background, we observe that code voting is also susceptible to a special form of vote selling: the voter may not be able to prove to a third party how he actually voted, but he may sell his voting credentials and code sheets to a person that casts his ballot in the first place (i.e., on behalf of the legitimate voter). One may be scared that people will sell their voting credentials and code sheets for a relatively small amount of money (because they have no economic incentive not to do so). Consequently, we see a possibility that people may try to buy votes or elections simply by offering enough money to some voters.

Taking a closer look at the possibility of vote selling, one recognizes that the situation

is similar to all-postal voting. Here, it is also possible to buy voting credentials and ballots, and to cast them on behalf of legitimate voters (there have even been reported cases in which people have stolen voting credentials and ballots—preferrably from elderly or handicapped persons). Even a person with only modest technical skills is able to scan the filled-in voting sheet and send the resulting image to the vote buyer over the Internet. To significantly influence the outcome of a vote or election, however, it is necessary to cast ballots for a huge quantity of voters. In this case, the attacker must in some way publicly announce the possibility to sell voting credentials and ballots. Since this is an offensive act, police forces have a possibility to prosecute such offerings (if they get aware of them).

In summary, we think that all-postal voting and code voting are similarly susceptible to vote selling, but that the problem may be more severe in the case of code voting (mainly because vote selling can be done more stealthy in the digital world). This is reason enough to overview and discuss some protection measures next.

5 Protection Measures

First, we note that protection against vote selling is difficult to achieve technically for any absentee balloting scheme. This is because a legitimate voter can always have somebody else complete his ballot and cast it himself (it may sometimes even be possible to have somebody else cast the ballot on behalf of the legitimate voter). We therefore focus on organizational and procedural protection measures, i.e., measures that sap the incentive to sell (or buy) votes and therefore protect code voting against vote selling. The adversary we have in mind is not a technically operating one, but rather one that has enough money to buy votes. The protection we try to achieve is not absolute, but the measures can still be used to reduce the vote selling problem.

5.1 Recast Ballots

If a voter can recast his ballot at any point in time (after the ballot box opens but before it closes), then a potential vote buyer cannot be sure that the voter does not recast his ballot (after having sold it) or resell it to somebody else. Consequently, he will offer less money for the ballot, and this, in turn, will reduce the likelihood that voters sell their credentials and code sheets in the first place. We are not able to quantify the level of protection. But we still think that providing the possibility to recast ballots is a simple but effective measure to protect code voting against vote selling. We are also aware of the fact that providing the possibility to recast ballots will have an impact on the way votes and elections take place. People will tend to cast preliminary ballots at some early point in time, but they will eventually recast them in accordance to recent developments and information. This will change the dynamics of votes and elections, and this, in turn, will change the strategies and tactics of political parties and politicians. Whether this is good or bad cannot be told today, but it will be interesting to investigate this point in the future.

5.2 Multiple Code Sheets

If a voter receives multiple similarly-looking and indistinguishable code sheets with only one being valid (e.g., the first one in a batch of sheets), then a potential vote buyer cannot be sure that he has received all code sheets or that the sheet he receives is actually the valid one (this is conceptually related to [JCJ05], where it is infeasible for an adversary to determine if a coerced voter complies with his demands). Codes from invalid sheets can be given to the voting application; they must not be counted, but otherwise the application must behave like with valid codes. Again, it is reasonable to assume that the potential vote buyer will offer less money for such a ballot, and that this, in turn, will reduce the likelihood that voters sell their credentials and code sheets in the first place.

Formally speaking, let $\mathcal{V}_R$ be a voting scheme where each voter receives a batch of $N + r$ similarly-looking and indistinguishable code sheets, with $N, r \in \mathbb{N}$, N constant, and r randomly chosen and equally distributed in $\{0, \ldots, R-1\}$ or $\{1, \ldots, R\}$, i.e., each possible value occurs with the same probability $1/R$. Without loss of generality, we assume that the first code sheet is the valid one, whereas all remaining $N + r - 1$ code sheets are invalid. If the vote buyer does not know r, then he doesn't know how many code sheets a voter actually possesses. Consequently, he is going to ask for $N + i$ code sheets for some value i between 1 and R. We assume that the vote seller tries to sell all of his code sheets except the valid one. In such a setting, Theorem 1 applies.

Theorem 1: If a vote buyer buys $N+i$ with $i \in \{1, \ldots, R\}$ code sheets from a vote seller in $\mathcal{V}_R$, then the success probability, i.e., the probability that he buys a valid sheet is $1/R$ (and hence independent from i).

Proof: If the vote buyer contacts a voter who has recieved $N + j, j \neq i$ code sheets, then this voter is either not able to sell his code sheets (if $j < i$) or he can keep the valid code sheet (if $j > i$). Only if $j = i$ is the vote buyer successful, and this happens with a probability of $1/R$.

Having Theorem 1 in mind, one may be tempted to ask for the most cost-efficient strategy (from the vote buyer's viewpoint). Obviously, this strategy occurs if the vote buyer minimizes the number of invalid code sheets that he buys (and spends money for). There are several assumptions (and simplifications) that one can make to address the problem. For example, one may assume that a vote buyer pays the same amount of money for every batch of code sheets (sold by a vote seller). In this case, Theorem 2 applies.

Theorem 2: If the vote buyer pays the same amount of money for every batch of code sheets from $\mathcal{V}_R$, then the most cost-efficient strategy (from the vote buyer's viewpoint) is to go for (and only buy) maximum-sized batches, i.e., batches of $N + R$ code sheets.

Proof: According to Theorem 1, the success probability for the vote buyer is $1/R$ for every $i \in \{1, \ldots, R\}$. Since for $i = R$ the vote buyer only spends money for batches that comprise a valid code sheet, the optimal strategy is to go for (and only buy) batches of $N + R$ code sheets.

Alternatively, one may consider the case in which the vote buyer pays the same amount of money for every code sheet (instead of every batch of code sheets) and—due to money

restrictions—he can buy up to K code sheets. In this case, S_i refers to the strategy of always buying $N + i$ code sheets (per vote seller) for $i = 1, \ldots, R$. For small values of i, the costs per vote seller are small, but the number of incomplete batches is large. For example, if $i = 1$, then the buyer can buy K batches of which $1/R$ comprise a valid code sheet. Consequently, the vote buyer ends up with K/R valid code sheets. For larger values of i, the costs per vote seller increase but the number of incomplete batches decreases. For example, if $i = R$, then the buyer can buy K/R batches of which all of them comprise a valid code sheet. Consequently, the vote buyer again ends up with K/R valid code sheets. All intermediate values for i, i.e., $i = 2, \ldots, R - 1$, perform worse. More formally, if the vote buyer acts according to S_i, then he ends up with

$$\frac{K}{i} \cdot \frac{1}{R - i + 1} = \frac{K}{i(R - i + 1)}$$

valid code sheets. The results of this function are summarized in Table 4. Note that the function returns maximal values for $i = 1$ and $i = R$ (the result is K/R in either case), and that it returns smaller values for all other values for i. Also note that the function is symmetric in the sense that it returns the same value for i and $R - i + 1$.

i	Valid code sheets
1	$K/(1(R-1+1)) = K/1R = K/R$
2	$K/(2(R-2+1)) = K/(2(R-1))$
3	$K/(3(R-3+1)) = K/(3(R-2))$
$\ldots$	$\ldots$
$R-2$	$K/((R-2)(R-(R-2)+1)) = K/((R-2)3)$
$R-1$	$K/((R-1)(R-(R-1)+1)) = K/((R-1)2)$
R	$K/(R(R-R+1)) = K/(R \cdot 1) = K/R$

Table 4: Number of valid code sheets for S_i

Things get more involved (and the strategy can be exacerbated) if one changes the probability distributions or the cost functions. Interdependencies like this are being explored in *rational cryptography* [Nie07]. Anyway, we think that providing multiple code sheets is a simple and effective measure to protect code voting against vote selling. Our major concern is usability: we are not sure whether multiple code sheets can be handled properly by the legitimate voters. It might be the case that legitimate voters are confused by the multitude of code sheets.

5.3 Powerful Voting Credentials

If a voter must employ powerful voting credentials to cast his ballot, then it is reasonable to assume that he will not hand them over to arbitrary (and possibly unknown) vote buyers. If, for example, a voter must employ an *electronic ID* (eID) card to authenticate himself to

the voting system, then it is unlikely that he will give his eID card to a vote buyer. Alternatively, one can think of using weaker credentials, but combine them with the privileges to look into private data (e.g., tax forms). Also, it may be possible to use credit card information or to have the voter deposit money that he can withdraw at some later point in time. There are some challenges related to the use of financial information or credentials, but they still represent possibilities one may consider. The idea is to reduce the incentive (or increase the price) for a voter to hand over his voting credentials to a vote buyer.

As a politically relevant example, we consider the use of eID cards that represent smart cards. Such cards are mainly used for identification, i.e., they uniquely identify their holders and are non-anonymous. Consequently, the use of such cards for remote Internet voting must be considered with care. In particular, one must separate remote Internet voting into two phases:

1. Voter authentication and authorization;
2. Vote casting.

The eID card can only be used in phase 1. Phase 2 must be anonymous and must not depend on the voter's eID card. A simple solution would be to store, after successful authentication and authorization in phase 1, a symmetric key common to all voters on the eID card. This key could then be used to encrypt the code chosen from the code sheet or to compute a message authentication code (MAC) for it. This approach is anonymous, but it allows a vote buyer to submit all votes he has bought. Consequently, we think that a more sophisticated remote Internet voting scheme must be employed. Ideally, such a scheme $\mathcal{V}_{eID}$ works in four phases.

- Initialization phase: Before the vote or election starts, each voter receives a code sheet by postal mail. Anonymity is guaranteed by manually shuffling the code sheets before sending them to specific mail addresses. Also, voters are allowed to exchange code sheets at will.
- Authentication phase: Before casting a vote, a voter must authenticate himself to an authentication server AS using his eID card. Only a small number of authentications to AS is allowed per eID card.
- Vote casting phase: The voter enters the selected code from his code sheet into the client application, and the client application sends out the code for transport through an anonymous network.[5]
- Vote counting phase: As codes received can be grouped according to the code sheets they originate from, vote updating can be detected. The last code from each code sheet is counted, by looking up the corresponding candidate from the secure database, and incrementing the counter.

[5]There are several technologies to implement anonymous networks. Examples include Chaum mixes, mix networks, and onion routing—a technology employed by The Onion Router (TOR) project.

$\mathcal{V}_{eID}$ employs code voting and is resistant against malware attacks. Also, the authentication process is secure because the voter's secret or private key is stored on the eID card. Large scale vote buying is difficult, because each legitimate voter can cast only a small number of votes. We therefore think that powerful voting credentials are effective to protect code voting against large-scale vote selling. The disadvantage is related to the fact that the management of voting credentials is generally in line with their power, i.e., the more powerful voting credentials are the more expensive their management is. In the most extreme case, we think that the management of eID cards is prohibitively expensive. In fact, protecting code voting against vote selling is certainly not reason enough for a state to launch an eID card program (but it may represent a welcomed side effect).

6 Conclusions and Outlook

Code voting is an appropriate technology to address the secure platform problem of remote Internet voting. In this paper, we elaborated on the susceptibility of code voting to vote selling, argued that the situation is comparable to all-postal voting, and overviewed and put up for discussion some organizational and procedural protection measures that are to sap the incentive to sell (or buy) votes and hence to protect code voting against vote selling. Recast ballots, multiple code sheets, and powerful voting credentials are useful and can be combined to some extent. Obviously, the first two measures don't work if the voter sells all the materials he receives in the first place (i.e., without opening them). Consequently, one must force the voter to open the materials. This can be achieved, for example, by sending it together with materials that are personally important for the voter, such as tax information.

If each voter is forced to properly authenticate and authorize himself before casting his ballot, then an adversary can cast only one ballot for each voting credential he is able to capture. Consequently, we argue in favor of a clear separation of voter authorization and vote casting, and we think that strong authentication technologies help protecting remote Internet voting against fraud. This argument has been the catalyst for Section 5.3. Also, private authentication technologies as, for example, porposed in [Aba02] can be used to improve the privacy in remote Internet voting. The primary goal is to ensure that a voter can vote only once. This can be ensured, for example, by a(n authentication) server AS that authenticates and authorizes a voter and—in the positive case—issues voting credentials of the form

$$(c_{vote}, hash(c_{vote}, ID_{voter}), sig_{sk_{AS}}(c_{vote}, hash(c_{vote}, ID_{voter}))),$$

where c_{vote} is a constant value for the vote or election, $hash$ is a cryptographic hash function, ID_{voter} is an identifier for the voter, sk_{AS} is AS's private signing key, and $sig_{sk_{AS}}$ refers to the signing function with this key. In this setting, the voter can verify that only the one-way hash of his identity ID_{voter} is included in the voting credentials, and the voting server can only link different credentials anonymously (except for an exhaustive search over all possible voter IDs). Against this background, there are a couple of schemes related to pseudo-voter identities [CD07] and anonmyous credentials (e.g., [Cha82, Bra00])

that come into play. Alternatively, one may also think of using group signatures [Cv91] to anonymously attribute ballots to legitimate voters. To be able to detect that two or more ballots are signed by the same voter, one must require that the group signatures are linkable (this is in contrast to the usual requirement of unlinkability). Last but not least, we think that having an authentication server issue Security Assertion Markup Language (SAML) tokens that serve as voting credentials provides an interesting possibility to invoke contemporary identity management technologies, such as Liberty Alliance, Shibboleth, and Microsoft CardSpace.

As already mentioned in the Introduction, several states have started to employ remote Internet voting, but as of this writing only a few projects employ code voting. To the best of our knowledge, none of these projects cares about vote selling. Consequently, it will be interesting to see whether vote selling really represents a problem, and if it does, whether the protection measures overviewed and discussed in this paper are useful.

References

[Aba02] Martin Abadi, "Private authentication," *Proceedings of the Second Workshop on Privacy-Enhancing Technologies (PET 2002)*, Springer-Verlag, LNCS 2482, 2003, pp. 27–40.

[Bra00] Stefan A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates—Building in Privacy*, MIT Press, Cambridge, MA, 2000.

[CD07] Orhan Cetinkaya and Ali Doganaksoy, "Pseudo-Voter Identity (PVID) Scheme for e-Voting Protocols ," *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES 2007)*, IEEE Computer Society, Los Alamitos, CA, 2007, pp. 1190–1196.

[Cha85] David Chaum, "Security without identification: transaction systems to make big brother obsolete," *Communications of the ACM*, Vol. 28, No. 10, 1985, pp. 1030–1044.

[Cv91] David Chaum and E. van Heyst, "Group signatures," *Proceedings of EUROCRYPT '91*, Springer-Verlag, LNCS 547, 1991, pp. 257–265.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson, "Coercion-resistant electronic elections," *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, Alexandria, VA, USA, ACM Press, 2005, pp. 61–70.

[Nie07] Jesper B. Nielsen (Ed.), *Summary Report on Rational Cryptographic Protocols*, ECRYPT Report D.PROVI.7, March 2007, http://www.ecrypt.eu.org/documents/D.PROVI.7-1.0.pdf.

[Opp02] Rolf Oppliger, "How to Address the Secure Platform Problem for Remote Internet Voting," *Proceedings of the 5th Conference on "Sicherheit in Informationssystemen" (SIS 2002)*, Vienna, Austria, 2002, vdf Hochschulverlag, pp. 153–173.

[Riv01] Ronald L. Rivest, "Electronic Voting," *Proceedings of Financial Cryptography 2001*, Springer-Verlag, LNCS 2339, 2002, pp. 243–268.

[Rub01] Aviel D. Rubin, "Security Considerations for Remote Electronic Voting over the Internet," *Proceedings of the 29th Research Conference on Communication, Information and Internet Policy (TPRC2001)*, October 2001.