A Simple Cast-as-Intended E-Voting Protocol by Using Secure Smart Cards

March 12, 2012

Helger Lipmaa

Institute of Computer Science, University of Tartu, Estonia

Abstract. We propose a simple cast-as-intended remote e-voting protocol where the security is based on the use of secure (and trusted) smart cards that incorporate incard numeric keyboards and LCD displays, and can perform a limited number of cryptographic operations (like encryption, signing, and random number generation). The protocol, while very simple, is significantly more secure (in the sense of "cast-as-intended") and convenient to use than the e-voting protocol currently used in Norway. The protocol is developed primarily with the idea of deploying it in Estonia within the next 3 to 10 years. Since in Estonia, a vast majority of the population already has ID-cards with digital signing and authentication functionality, and the use of ID-cards is a required prerequisite to participate in Estonian e-voting anyway, our assumption of every voter having a secure hardware token makes sense in this concrete context.

Keywords. Cast-as-intended e-voting, Estonian e-voting, secure hardware.

1 Introduction

Several countries like Estonia and Norway are either already implementing or considering implementing nation-wide elections in a way that allows the voters to use their own computers for vote casting. For the sake of simplicity, we will call such elections just *e-voting* during this paper. Since voting in general is very important in contemporary democratic society and thus has to be performed in a correct and accountable way, e-voting has many unique security challenges. This has for the last 30 years generated a huge amount of cryptographic research that has however mainly focused on guaranteeing privacy and correctness properties against malicious voting servers (also known as the *counted-as-cast* or *accepted-as-cast* property), see [CGS97,Nef01,DJ01,Gro03,GL07a,BG12,LZ12] for some typical papers in this field.

During the process of preparation of Norwegian e-voting, it was established that — at least in the context of Norway — the largest challenge is the security of voters' computers [AHL+09,HLV10,Gjø11]. Due to the ease of construction of (specifically tailored) malware and the fact that the voter computers very often have inadequate protection against (most of the) malware, one cannot just assume that the voter computers are secure. One should really design e-voting protocols that achieve either privacy or verifiability (cast vote arrived to the voting servers, *cast-as-intended*), or both, in the presence of malicious voter computers. Without at least some degree of verifiability in place, e-voting should not implemented at all.

Prior Work. There exist many cast-as-intended e-voting settings and protocols. Here, by setting we mean the infrastructure that is visible to a voter (for example, whether she has to use a computer or a mobile phone to vote, and whether there is a visible verification code), and by protocol we mean a technical solution that implements secure e-voting given a concrete setting. Obviously, it is much simpler to change the concrete protocol than the setting, since it does not require one to change the voter user experience.

Probably the best known cast-as-intended setting is *code voting* [Cha01], where every voter receives before the election (over some secure prechannel that is independent of the voter computer) a codesheet that contains a voter-specific input and a verification code per every candidate. She inputs the first code to her computer, and "accepts" that her vote was transferred to the voting servers only after she sees the correct verification code (on some secure postchannel that is again independent of the computer). While code voting guarantees both privacy and the cast-as-intended property, it relies on several underlying assumptions that are undesirable in the real-world e-voting. In particular, it means that the voters can only vote if the codesheet delivery is guaranteed.

To alleviate this problem, Norway uses a different setting, called code-verification in [Lip11]. (The first description of this setting is given in [AHL+09], the first concrete protocol is proposed in [HLV10] and the presumably final Norwegian protocol is analyzed in [Gjø11].) In this setting, the codesheet (that is transferred to the voters over a prechannel, which in the case of Norway is the usual postal mail) only contains the verification code. The voters will enter the candidate name (or number) to their computers as usually (thus, this setting does not guarantee privacy against a malicious computer), and then receive the verification code over an independent postchannel (SMS, in the case of Norway). As in the case of code voting, the voters only accept after seeing a correct verification code that their vote has made it to the voting servers. In this setting, the voters can vote without having a copy of the codesheet, and thus do not have to rely on the trustworthiness of the postal service.

However, the codesheets are still required for verification and thus the Norwegian system still poses several practical problems, like the cost and inconvenience of printing and mailing out codesheets to all the voters, the necessity of knowing the postal addresses (and mobile phone numbers) of all voters, possible privacy (like eavesdroppers observing the codesheet printing or the postal service, or eavesdropping the SMSs) and correctness issues (attackers actually actively modifying the codesheets or SMSs). Thus, while the Norwegian e-voting system guarantees the degree of verifiability not seen in many other systems (including the Estonian e-voting system), it also has several problems that one should try to overcome with.

There are several directions one can take instead of the code-verification setting. For example, one setting currently considered in Estonia [HMVW12][1] makes it possible for the voters to use their mobile devices (e.g., smartphones or tablets) for verification process. While this eliminates many problems in Norwegian e-voting, it introduces several new problems. The first problem is smartphone security: it is well-known that at least Android smartphones are currently plagued (see [LMS] for just one of the papers in this area) with many security issues. (One but not only reason for this is the fact that Android apps are not carefully screened before they make it to the "Google Play".)

If the verification device (a smartphone) is not more secure than the vote casting device (a computer), introducing verifiability does not really give us much. Obviously, the attacker now has to attack two different devices (the voting device and the verification device) with — at this moment — different operating systems, and this will make her work considerably more complicated. However, this is only the current situation, and it may be that in 5 years many users use either (say) Android, Windows 8 or iOS both on their computers and mobile devices. In such a case, it will be equally simple to attack the voting device and the mobile verification device.

Secondly, smartphones (or any mobile devices, capable of verification as envisioned in [HMVW12]) are still luxury devices[2], and it is wrong to assume that a significant part of the electorate will have them. Here, the problem is not only in the cost of mobile devices, but also in the perceived unnecessity of the devices by many users who prefer to have feature phones — and are not interested in say tablets either — for whatever reasons. (While the authors of the current paper are not among them, they have many such friends — even and somethimes especially among technically savvy people — who by principle do not want to have a smartphone or a tablet.)

We feel strongly that such a verification system will introduce strong "verification non-uniformity", where only sufficiently affluent voters (who have money to buy smartphones or say tablets) — or voters, who do not object to having a smartphone — will be able to verify their votes.[3] This introduces a risk of a directed malware that only infects computers of say poorer voters (where some machine learning algorithms are applied to detect the probability that the owner of a computer has a smartphone). Such

[1] Currently only available in Estonian, except some slides, see e.g. `http://satoss.uni.lu/seminars/srm/pdfs/2012-Jan-Willemson.pdf`. The English version [WH13] is available from the authors. See App. A for a short description. We emphasize that despite of the full description of this system not being publicly available in English, it is currently being implemented for a pilot project

[2] To execute the protocol of [HMVW12], they must have a reasonable camera, a possibility to install applications, and enough computing power to decode a QR code and perform the verification (which may require exhaustive encryption of all candidate numbers by using the same randomness). If there are hundreds of candidates — like in Estonian parliamentary elections —, the latter may require significant processing power.

[3] We note that while verification uniformity is not a notion that is currently mentioned in the election laws of any countries (up to our knowledge), this is partially since verification by itself is not supported by current election mechanisms. If verification will be supported (in particular, in law), also the notion of uniformity should be updated to cover verification.

machine-learning algorithms — though in different contexts — exist and have put to use, an example is a recent high-profile case where Target knew that a teen is pregnant (based on her consumption patterns) before her father did[4]

The end-result will create a system that is detrimental to universal suffrage that most of the developed world has enjoyed for 100 years. It is also our strong belief that the costs of a election should be paid by the state and not by the individual voters.

Thirdly, such a system does still not offer privacy against malicious voter computer. While achieving such privacy is nigh-impossible in most of the settings, the setting proposed in the current paper allows a voting scheme to enjoy such privacy.[5]

Our Contributions. Compared to the Norwegian e-voting system, we are interested in a system that is cheaper and more efficient to maintain, and poses less security risks. Compared to the smartphone-verifiable systems, we want to achieve better verification uniformity, and also better security. In addition, we would like the e-voting system to be private against the malicious computer.

All the mentioned desiderata seem to be impossible to achieve simultaneously unless one has access to some secure hardware token: if the voter cannot trust the computer, there must be some other device that is more trustworthy. In the case of Norwegian election systems, it is for example assumed that the postal service is more trustworthy than Internet. This is an assumption that may work in Norway but does not hold true not in every country.

Since this work is written in the context of trying to develop a new verifiable e-voting system in Estonia, we can make use of the wider context in Estonia. In particular, in Estonia every citizen must have an ID-card which is a device for visual identification, but it also has a capability for electronic authentication and digital signing. At this moment, a very large majority of Estonian population has such ID-cards[6]. Since ID-cards are distributed by the state, it can be assumed that the state will be able to reduce their prices to individual voters. Moreover, importantly, ID-cards are multipurpose devices that can be used to log in to e-banking, and they are already used for authentication in Estonian e-voting. (That means in particular that one cannot currently cast an e-vote without first having an ID card[7].) Thus, the assumption that one has to use ID-cards (or their extended versions) for e-voting and verification does not reduce verification uniformity in the Estonian context. As we already mentioned, in the current Estonian e-voting protocol, a voter must already have an ID-card to authenticate herself and thus to cast a vote.

In the next section we propose a very simple ID-card based e-voting protocol. It makes some assumptions on the ID-card that are not satisfied by the current Estonian ID-card:

- the ID-card must have an LCD display to display a candidate number and a verification code (preferably but not necessarily at the same time). The verification code does not have to be too long to guarantee acceptable security level (say 3 or 4 decimal digits)[8],
- the ID-card must have a numerical keyboard so that the voter can input the candidate number,
- the ID-card must be able to perform a small number of cryptographic operations (certificate verification, random number generator, encryption — by possibly using a less standard cryptosystem —, signing, etc).
- finally, the ID-card must be capable to implement simple cryptographic protocols based on inputs on the incard keyboard (in particular, inputs from the computer should be ignored), without leaking the intermediate results to the voter computer.

[4] See for example `http://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/`..

[5] One could argue that also here, one can use machine learning to get to know the preference of the voter so that additional privacy does not help here. It however does: the difference is between the voter being likely to vote for some candidate, and he or she actually doing it.

[6] See `http://www.id.ee/?lang=en`

[7] See `http://www.valimised.ee/internet_eng.html#C1`

[8] Such cards have existed for many years, see `http://www.schneier.com/blog/archives/2006/09/oncard_displays.html`

1. The voter takes the card out from the card reader, if it was there, and performs the following steps:
   (a) The voter obtains the number of her preferred candidate from a trusted source.
   (b) The voter inputs the candidate number using the incard keyboard. The card verifies that this candidate number is in the correct range.
   (c) The card generates a random new ℓ-digit verification code v and displays it on the incard LCD display. The voter memorizes (or writes down) the code.
2. The voter i inserts the card to the card reader, and then the next steps are performed:
   (a) The card generates new random numbers r_1 and r_2 as required by the cryptosystem,
   (b) Let id be the unique identifier of the concrete election. The card encrypts the candidate number c (by using the tallier's public key T and randomness r_1) and the random verification code v together with voter's identifier i (by using the vote collector's public key VC and randomness r_2), and then signs a message that contains id, two ciphertexts

$$(C_1, C_2) = (E_T(id||c; r_1), E_{VC}(id||v||i; r_2))$$

   and possibly some additional election-specific auxiliary data. Let σ be this signature and M be the signed message $M = (id, E_T(id||c; r_1), E_{VC}(id||v||i; r_2), \dots)$.
   (c) The card sends the message M and the signature σ to the computer, who forwards both to the vote collector.
   (d) The vote collector verifies the signature, and if the verification succeeds, stores both the message $M = (id, C_1, C_2, \dots)$ and the signature σ. He also decrypts the verification code $id||v'||i \leftarrow D_{VC}(C_2)$. If id corresponds to the current election and i is the number of the voter whose card signed M, VC sends a signed $(id||v'||i, S_{VC}(id||v'||i))$ back to the ith voter's computer.
3. The voter takes the card out of the card reader. The computer displays the code v' to the voter, who then compares this to the code that is still visible on the incard LCD display.
   Alternatively, comparison can be done by the card itself. (In this case, no verification code will not be shown to the the voter at all.)
4. The code on the display will be erased at some point after the vote was cast to make vote buying and coercion more difficult to organize.

Fig. 1. Single voting attempt protocol

Moreover, to be able to satisfy the previous assumptions, software on the ID-card must be securely updateable. Such cards are available on the market[9], and they can also importantly used in other applications (like say e-banking, e-cash, or just to make digital signing more secure).

The idea of the protocol is very efficient, which is in fact very beneficial due to the small processing power of the current smart cards. It is also conceptually simple, which makes it easy to understand and implement also by non-cryptographers. We will give the full description of the protocol in the next section.

2 New Protocol

We first describe the basic new cast-as-intended e-voting protocol, in the following discussions we will also detail some potential strengthenings. Let ℓ be the verifiation code length (say $\ell = 4$) in digits. We assume that (G, E, D) is a secure public-key cryptosystem and that (G_s, S, V) is a secure signature scheme (see Sect. 2.1 for a more precise decription of the underlying cryptographic primitives).

The protocol consists of the initialization protocol and of the voting (attempt) protocol. The initialization protocol is performed only once before any voting attempt is made, and the voting protocol is performed once per every voting attempt. The initialization protocol is as follows (we assume that the problem of obtaining the root certification authority public key, and the voting application are solved

[9] See for example `http://www.nidsecurity.com/products/details-306.html`, that incorporates a secure contactless interface to load an OS, application, proprietary algorithm, key files, etc through ISO 14443 Type B protocol. Whether this concrete card is applicable for our needs has to be seen, but we think that it is a reasonable assumption that if a country of Estonia's size will start a procurement or a acquisition process, there will be vendors interested in providing necessary hardware.

separately, possibly by physically visiting a designated office of the government — or a bank, as it can be currently done in Estonia):

1. The card obtains the election description from the computer. The description is signed by the root certification authority, and contains certificates of the public keys of the vote collector and the tallier. It also contains auxiliary information about the election, including a unique identifier *id* of this election, starting and ending dates of the e-voting period, the full candidate list, etc.

(We note that for efficiency reasons, it is reasonable to assume that the certificate will not contain too many levels of hierarchy.)

To obtain full security, before the election, the card should obtain from the computer a certificate on two required public keys. Thus, the card must be able to do certificate verification. Here we need to assume that the hardware and software is sufficiently secure so as the information from the card is not leaked to the computer, and computer's bogus inputs are not accepted by the card.

A single voting attempt consists of the steps, described in Fig. 1. We emphasize that to achieve security, *id* must be a unique election identifier. Moreover, the card should *never* encrypt or sign any message, sent to it by the computer (even in the context of this election), that starts with prefix *id*: prefix *id* really means that the encrypted candidate number has been entered to the card *by the incard keyboard* within the context of the concrete election. To guarantee backwards compatibility with the current Estonian digital signature practices (namely, currently the ID card only receives a hash of the message to be signed, and therefore cannot verify whether the message starts with *id* or not), it may be easier to assume that one employs a different signature key in e-voting, or more generally, contains two signature keys, one of which signs messages as currently (signs a hash, without prefixing an *id*), while the second key signs a hash, prefixed by an *id*.

In this protocol, as one can see, the card must be able to perform a very small number of cryptographic operations (certificate verification, random number generation, signing and encrypting) together with some UI operations (inputting the candidate number and displaying the code) and with some auxiliary operations (like basic string operations). While all this may be complicated, it is difficult to imagine an alternative smart card-based protocol that would skip using any of the mentioned subroutines:

- Obtaining certificates of the public keys of voting servers cannot be avoided, since otherwise a malicious computer may act as a vote collector.
- Encryption and signing functionalities are clearly necessary.
- Having an incard keyboard and a display will conceal information from the computer, and protect against the attacks where the computer initiates signing without an input from the voter.
- Executing the whole protocol inside a card is necessary, since otherwise at least some private information would be leaked to the computer.

Moreover, a card that already supports the mentioned operations can clearly also be used in some other contexts. For example, in Estonia currently one has to type in a PIN to the computer to be able to sign something with the ID-card. This is a clear security bottleneck of the current system. Obviously all ID-card applications (digital signing, authentication, etc) will be much more secure when the PIN is directly entered to the smart card.

We note that in the previous protocol, it was assumed that the voter receives the candidate number from a trusted source. However, also the ID-card obtains the candidate numbers, digitally signed, from a trusted authority. Ideally, the ID-card should able to certify to the voter the name of the candidate with the given number, that is, also to display at least a part of the candidate's name.

Finally, the version where comparison is made by the card (and not by the voter, see Step 3 in Fig. 1) is actually better for the following reasons. First, it requires the voter to perform less operations, and thus introduces less human errors. Second, it does not give the voter a possibility to falsely claim that the code did not go through.[10] Third, one can imagine that a code displayed by the card and known by the voter finds a way to the computer (by a designated malware asking for this code). Therefore, the e-voting protocol will be also more secure if the voter does not see the code.

[10] In several countries like Belgium and Estonia, the possibility that a voter performs such publicity attacks has been seen as a reason *not* to allow cast-as-intended verification.

2.1 More on Required Cryptographic Primitives

In principle, the proposed protocol only requires an IND-CCA secure public-key cryptosystem [RS91] (like RSA-OAEP [BR94]), and a secure signature scheme [GMR88] (like RSA-PSS [BR96]). However, we do recommend against using RSA for the simple reason that the current recommended RSA key length is 3248 on security level 2^{128}, see [ECR11], and will become larger in every few years. Thus, after a few years, using RSA will just become too expensive. Instead, one should use more efficient — and possibly, secure — cryptosystems (like Cramer-Shoup [CS98]) and signature schemes (like ECDSA) implemented on top of elliptic curves. According to [ECR11], elliptic-curve cryptography has recommended key length of 256 bits on the same security level, and therefore offers significantly better throughput. This is very important, especially since the smart card cannot be assumed to be computationally powerful, have large internal memory, or have a very long battery life.

Moreover, using RSA is not compatible with all cryptographic protocols. For example, if a future version of the e-voting system is also made accepted-as-cast secure by say implementing a secure shuffle, one should instead of RSA use a homomorphic cryptosystem (like Elgamal [Elg85], Paillier [Pai99], or BBS [BBS04]).

2.2 General Security Assumptions

The new protocol will be secure if the required cryptographic primitives are secure and some additional security assumptions hold on the hardware. We will briefly describe the additional assumptions, though it may come up that in practice one will need even more assumptions. We omit formal proofs: the security of the minimal cryptographic part of the new protocol is obvious. Full modelling the non-cryptographic part of the protocol will be left for a future work.

Assumptions on Smart Card. The smart card must be completely trusted. That is, it is assumed that the smart card is a secure black box that will perform all computations as required, will not accept bogus inputs from the computer or the voter, and will not leak any secret information to the outside world. In practice it means that one has to trust the smart card issuer and the software on the card (including the updating process). In addition, the smart card must be sufficiently secure not to allow for any side channel attacks.

Assumptions on Voter Computer. The voter computer can be completely untrusted. If the smart card is secure, the computer will not obtain any private information. The only reasonable attacks the computer can perform are the same the untrusted Internet can perform (packet removal, injection, etc). One cannot obviously vote at all if the computer does not send away the encrypted ballot. However, this will be detected by the voter since she will then not obtain the correct code. Since all information is encrypted and signed, the computer can also not mount any other undetected attacks without damaging the e-voting security.

Assumptions on Voting Servers. A malicious vote collector can clearly perform the same attacks as the untrusted Internet (e.g., packet removal and injection). As in the case of a malicious voter computer, such attacks will be detected, though the voter will not be able to decide who was the attacker. The vote collector does not see the candidate number, so a malicious vote collector cannot violate voter's privacy or mount selective (that is, based on the actual ballot — selective attacks based on the voter's social background are obviously possible) attacks.

A malicious tallier can obviously output a wrong tally and also violate voters' privacy. To defend against this, one should in addition use cryptographic protocols (e.g., a shuffle) to defend against attacks by a tallier. Such protocols are well-known [Nef01,Gro03,GL07b,GL07a,BG12,LZ12], and we will not describe them in the current paper.

2.3 Pros and Cons of the New Protocol

We list briefly some pros and cons of the new protocol, as compared to the Norwegian setting [HLV10,Gjø11] and the mobile-device based protocol of [HMVW12]. First, the positive sides are:

– Need to only trust ID-cards, no computers. Secure ID-cards are arguably more secure than the personal computers (and smartphone-like mobile devices), partially since security is one of their most important intended properties.
– No need for prechannels and postchannels. The need for prechannels and postchannels makes the Norwegian solution somewhat costly and cumbersome. (The government needs to know the postal addresses and mobile numbers of all voters, the codes have to be securely printed and distributed to the voters, etc.)
– The code is completely random, that is, unique to every voting attempt[11], and it does not depend on candidate. This solves several issues present in the Norwegian e-voting system. (For example, in the Norwegian case, a malicious vote computer might not know for whom the voter votes, but can say observe that the voter voted three times, for some candidates A, B, and A. In the last attempt, the voting computer can vote for B instead, and then collaborate with some other malicious party who will show the voter the verification code for A.)
– Extended ID-cards can have other applications (banking), which means that their deployment costs can hopefully be amortized, and one may be able to get funding from say banks to support the deployment of such ID-cards. Moreover, introducing extended and more secure ID-cards will also make other applications (starting from digital signing) more secure, which will benefit the society in general.
– Uniform verification: the deployment costs are covered by the state, unlike in the mobile-device based protocol of [HMVW12]. This makes verification more fair and uniform.
– Privacy: the voter's computer never sees the vote, and the vote cannot also be obtained by combining access to the prechannel and the postchannel. This is an important benefit, not shared by almost any of the other cast-as-intended protocols, except the code voting.

The drawbacks of this solution are:

– Need to trust the ID-cards (or, their manufacturer) (see Sect. 3 for more discussion).
– Initial cost of deployment (see Sect. 4 for more discussion).

An additional drawback may be the cost of securely updating the software on the ID-card. The current Estonian ID-card has to be changed to a new one every five years, and it would be desirable if the voter did not have to visit a designated office to update the election software for every election. (This can probably be solved by using existing secure update mechanisms, but it has to be studied further.)

3 Reducing Trust in ID-Cards

In the solution, proposed above, one needs to trust the ID-cards almost completely. While this is a much smaller issue than trusting the personal computers (e.g., when the computer is infected, one does not know whether it was because of the manufacturer or it was just some malware), it may still be problematic in practice.

We outline below some possibilities of improving over the "trivial" protocol above. Our outline is very brief and omits many details. The simplest idea here seems to be combining extended ID-cards with other verification methods. While such combinations would eliminate many of the benefits of the extended ID-card based solutions (e.g., it will add an additional and costly infrastructure element), they would make it possible to achieve some security even in the case the ID-cards are not secure.

As an example, one could combine the ID-card based solution with mobile-device based verification. The ID-card would additionally (to $E_T(id||c;r_1), E_{VC}(id||v||i;r_2)$) encrypt the candidate number c by using a public key MD, where the corresponding secret key is only known to the mobile device. Let the resulting ciphertext be $C_3 = E_{MD}(id||c;r_3)$, where r_3 is a fresh randomness. The mobile device will also add a non-interactive zero-knowledge proof π that the two ciphertexts $C_1 = E_T(id||c;r_1)$ and $C_2 = E_{MD}(id||c;r_3)$ of c really encrypt the same id and the same candidate number. If one uses the Elgamal cryptosystem, such a zero-knowledge proof is standard and very efficient in the random oracle model. One can also implement this zero-knowledge proof efficiently (by using pairing-based cryptography [GS08]) without assuming the random oracle model.

[11] We recall briefly that in Estonian e-voting, one can vote many times. We call every such voting a voting attempt.

The vote collector will send $(C_1, C_2, C_3, \dots)$, together with a signature, to the mobile device. (Note that clearly the signature σ will now be over three ciphertexts.) The mobile device will then decrypt the ciphertext C_3 and show it to the voter. A problem behind this solution is that the mobile device is required to have a special secret key, which is usually not the case. However, in Estonia one could possibly use the Estonian "mobile-ID" solution (details omitted).

Whether this extension is useful in practice or not can be debated, and should be further studied. However, it shows that at least in principle one can decrease required trust in the ID-card.

4 Additional Practical Concerns

We will briefly list *some* of the practical concerns that have to be analyzed in the future. There are most definitely more questions to answer, but we will leave it for the future work.

The cost of deployment of security-enhanced smart cards. We think that the state should distribute such security-enhanced ID-cards to the voters with a highly reduced cost, to help to achieve uniform verifiability. Currently, Estonian citizens and residents have to pay 25 euros to obtain a new ID card (`http://www.politsei.ee/et/teenused/riigiloivud/riigiloivu-maarad/isikut-toendavad-dokumendid/index.dot`), and we guess that new extended ID cards will be somewhat more expensive. As said, a part of this new expense should be covered by the state. In addition, we do hope that part of the costs will be covered by say banks who currently rely on their own — and not cheap — technology (PIN calculators, one-time code sheets) to provide secure authentication. Whether this is realistic, is a good question.

As an example, Estonian ID-cards are currently used for authentication and digital signing. According to Estonian laws, in the context of legal documents, digital signatures are equivalent to the usual signatures. To perform digital signing, the user currently inserts her card to a card reader, selects the document to sign on the computer, and then inputs her PIN code to the computer. The computer then interfaces with the card, obtaining a digital signature on the document. The user has to completely trust the computer not only on which document was signed, but also on how many documents were signed, and on whether the computer did not store the PIN for the future use. This poses a serious potential problem, since in Estonia digital signatures are legally binding and thus equivalent to the handwritten signatures. Given a security-enhanced smart card, one can at least guarantee that the PIN is not leaked to the possibly infected computer. (Though it is difficult to guarantee which document is signed, since the real document does not fit to the incard display.) (We note that Estonian ID-cards probably have to be replaced anyway during the next few years, due to the use of shorter than recommended (2048-bit) RSA keys, and their insecurity against the Bleichenbacher attack [BFK+12].)

Another possible application is e-banking. Some of Estonian banks sell PIN calculators to the customers, while other banks send them one-time codesheets. The PIN calculator provides a (not strictly) subset of functionality required for the e-voting: after the user inputs her long-term PIN code to the calculator, the calculator displays an ethereal (short-term) PIN code that the user then can use to log into her bank account. Clearly, an ID-card that can execute the e-voting functionality can also execute this functionality, though the short-term PIN code should be some determined functionality of the user's secret key and (say) the current time (say, a signature of the current time). In addition, one could imagine that in e-banking, before confirming a payment, both the payee bank account and the sum will be displayed on the incard display, after which the payer has to input her PIN on the incard keyboard.

Obviously, the number of included functionalities has to be very tightly restricted so as not to create unnecessarity vulnerabilities and make ID-cards as insecure as the computers.

Distribution and updating the code on smart cards. This includes how the root certification authority's public key should be included to the smart card (and revoked, if necessary). One can assume that a secure smart card offers such functionality, but it will depend heavily on the concrete smart card.[12]

[12] See again `http://www.nidsecurity.com/products/details-306.html`, that claims to have such a functionality, and footnote 9

Acknowledgments. We are grateful to Sven Heiberg, Arnis Paršovs and Jan Willemson for comments. The author was supported by the Estonian Research Council, STACC, and European Union through the European Regional Development Fund.

References

AHL+09. Arne Ansper, Sven Heiberg, Helger Lipmaa, Tom André Øverland, and Filip Van Laenen. Security and Trust for the Norwegian E-voting Pilot Project E-valg 2011. In Audun Jøsang, Torleiv Maseng, and Svein J. Knapskog, editors, *NordSec 2009*, volume 5838 of *LNCS*, pages 207–222, Oslo, Norway, 2009. Springer, Heidelberg. 1, 1

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, Santa Barbara, USA, August 15–19, 2004. Springer, Heidelberg. 2.1

BFK+12. Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient Padding Oracle Attacks on Cryptographic Hardware. In Rei Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 608–625, Santa Barbara, California, USA, August 19–23, 2012. Springer, Heidelberg. 4

BG12. Stephanie Bayer and Jens Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg. 1, 2.2

BR94. Mihir Bellare and Phil Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT 1994*, volume 950 of *LNCS*, pages 92–111, Perugia, Italy, 9–12 May 1994. Springer, Heidelberg. 2.1

BR96. Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures — How to Sign with RSA and Rabin. In Ueli Maurer, editor, *EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg. 2.1

CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Walter Fumy, editor, *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 103–118, Konstanz, Germany, 11–15 May 1997. Springer, Heidelberg. 1

Cha01. David Chaum. SureVote: Technical Overview. In *WOTE 2001*, 2001. Available from `http://www.vote.caltech.edu/wote01/pdfs/surevote.pdf`, as of August, 2010. 1

CS98. Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In Hugo Krawczyk, editor, *CRYPTO 1998*, volume 1462 of *LNCS*, pages 13–25, Santa Barbara, USA, 23–27 August 1998. Springer, Heidelberg. 2.1

DJ01. Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136, Cheju Island, Korea, February 13–15, 2001. Springer, Heidelberg. 1

ECR11. ECRYPT II Project. ECRYPT II Yearly Report on Algorithms and Keysizes (2010–2011). Technical Report ICT-2007-216676, June 30, 2011. Available at `http://www.ecrypt.eu.org/documents/D.SPA.17.pdf`. 2.1

Elg85. Taher Elgamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 2.1

Gjø11. Kristian Gjøsteen. The Norwegian Internet Voting Protocol. In Aggelos Kiayias and Helger Lipmaa, editors, *VoteID 2011*, volume 7187 of *LNCS*, pages 1–18, Tallinn, Estonia, September 29–30, 2011. Springer, Heidelberg. 1, 1, 2.3

GL07a. Jens Groth and Steve Lu. A Non-interactive Shuffle with Pairing Based Verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg. 1, 2.2

GL07b. Jens Groth and Steve Lu. Verifiable Shuffle of Large Size Ciphertexts. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 377–392, Beijing, China, April 16–20, 2007. Springer, Heidelberg. 2.2

GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17:281–308, 1988. 2.1

Gro03. Jens Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 145–160, Miami, Florida, USA, January 6–8, 2003. Springer, Heidelberg. 1, 2.2

GS08. Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In Nigel Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg. 3

HLV10. Sven Heiberg, Helger Lipmaa, and Filip Van Laenen. On E-Vote Integrity in the Case of Malicious Voter Computers. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *ESORICS 2010*, volume 6345 of *LNCS*, pages 373–388, Athens, Greece, September 20–22, 2010. Springer, Heidelberg. 1, 1, 2.3

HMVW12. Sven Heiberg, Tarvi Martens, Priit Vinkel, and Jan Willemson. Individuaalselt kontrollitav elektrooniline hääletamine: võrdlev analüüs. Versioon 1.3, April 5, 2012. 1, 2, 2.3, A

Lip11. Helger Lipmaa. Two Simple Code-Verification Voting Protocols. Technical Report 2011/317, International Association for Cryptologic Research, June 15, 2011. Available at http://eprint.iacr.org/2011/317. 1

LMS. Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. A Survey on Security for Mobile Devices. *IEEE Communications Surveys and Tutorials*, ?:?–?, ? Accepted for publication. 1

LZ12. Helger Lipmaa and Bingsheng Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In Ivan Visconti and Roberto De Prisco, editors, *SCN 2012*, volume 7485 of *LNCS*, pages 477–502, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg. 1, 2.2

Nef01. C. Andrew Neff. A Verifiable Secret Shuffle and Its Application to E-Voting. In *ACM CCS 2001*, pages 116–125, Philadelphia, Pennsylvania, USA, November 6–8 2001. ACM Press. 1, 2.2

Pai99. Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg. 2.1

RS91. Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In Joan Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *LNCS*, pages 433–444, Santa Barbara, California, USA, August 11–15, 1991. Springer, Heidelberg, 1992. 2.1

WH13. Jan Willemson and Sven Heiberg. Individually Verifiable Protocol for Estonian Internet Voting. Available from the authors, 2013. 1

A Proposed Mobile-Device Based Verification Protocol

Since [HMVW12] is currently only available in Estonian, we will provide here a very short description of their protocol. The voter computer encrypts vote and sends it, together with a signature to the voting server. At the same time, the voter computer will display a QR code that encapsulates the randomness that was used while forming the ciphertext. The voter then uses a special application on the mobile device (that has to be preinstalled). He or she first uses the mobile device camera to take a picture of the QR code, that will be decoded in the mobile device to the original randomness. The voting server sends to the mobile device voter's (signed) encrypted ballot and the candidate list. After verifying the signature, the mobile device checks that the ciphertext corresponds to an encryption of *some* valid candidate (possibly using exhaustive search over all candidates) with the seen randomness. If yes, it displays that candidate number/name. The voter then verifies that this is the correct candidate.