On E-Vote Integrity in the Case of Malicious Voter Computers

Full Version, September 22, 2010

Sven Heiberg[1], Helger Lipmaa[1,2], and Filip van Laenen[3]

[1] Cybernetica AS, Estonia
[2] Tallinn University, Estonia
[3] Computas AS, Norway

Abstract. Norway has started to implement e-voting (over the Internet, and by using voters' own computers) within the next few years. The vulnerability of voter's computers was identified as a serious threat to e-voting. In this paper, we study the vote integrity of e-voting when the voter computers cannot be trusted. First, we make a number of assumptions about the available infrastructure. In particular, we assume the existence of two out-of-band channels that do not depend on the voter computers. The first channel is used to transmit integrity check codes to the voters prior the election, and the second channel is used to transmit a check code, that corresponds to her vote, back to a voter just after his or her e-vote vast cast. For this we also introduce a new cryptographic protocol. We present the new protocol with enough details to facilitate an implementation, and also present the timings of an actual implementation.

Keywords. Implementation, integrity, malicious voter computers, nationwide e-voting, proxy oblivious transfer, zero-knowledge proofs.

1 Introduction

The first e-voting pilot (that is, voting over the Internet by using voters' own computers) pilot in Norway is currently scheduled for 2011[1], with plans to have nation-wide e-voting by 2017. As it should be in all democratic countries, Norway aims the electronic elections to be both as accessible/usable and as secure as possible. It is not always easy to reach a sensible compromise. In this paper, we describe our e-voting solution that was proposed to the Norwegian election officials in Summer of 2009. The proposed e-voting protocol tries to find a good compromise between various security and usability.

A nationwide implementation of e-voting has to be secure against as many attacks as possible, and in presence of as many malicious parties as possible without seriously hurting usability or the ease of verifiably correct implementation. Abundant research has been done on the security of e-voting in the presence of malicious voting servers, see [CFSY96,CGS97,FS01,Nef01,Gro03,Fur04,GL07] or the overview [Lip05b]. Thus, this part of e-voting can be considered to be solved to at least certain degree, and thus

[1] See http://www.regjeringen.no/en/dep/krd/kampanjer/election_portal/electronic-voting.html?id=437385 for more information.

in this paper, we will not focus on this aspect of e-voting. (The real e-voting will implement additional means to guarantee security against malicious voting servers.)

On the other hand, it is even more difficult to guarantee security in the case when voter computers cannot be trusted. The seeming impossibility of guaranteeing vote privacy and integrity in the presence of malicious voter computers has been one of the main obstacles that has delayed the real-world implementation of e-voting.[2] Moreover, achieving vote privacy in the case of malicious voter computers seems to hurt usability [AHL^{+}09], since the value input by a voter to the computer should not reveal voter's preferred candidate. In practice, this amounts to inputing a pseudorandom code (or something similar), unknown to the voter computer but yet securely delivered to the voter himself or herself. Due to both the impossibility of implementing secure yet guaranteed code delivery and to the usability concerns, solutions where a voter is required to enter a random code to the computer are definitely out of the question[3]. In Norway, solutions where the voter could obtain the used random values, and then use her own program to verify the correctness of ciphertexts [Adi08] were not even considered.

Our Contributions. We show that it is possible to guarantee e-vote integrity in the presence of malicious voter computers without drastically changing the user experience, and without the necessity of 100% delivery of random codes (or say, secure hardware tokens). More precisely, we construct a cryptographic protocol at the end of which, after she has entered her vote to the computer, the voter obtains a relatively short integrity check code. Given this check code (and/or the absence or presence of the message itself), the voter can verify the integrity of her vote. This easy verification is the only change in her voting experience as compared to a similar non-secure system: she is not required to enter long codes, nor has the user interface to be particularly clunky. Moreover, in our case, the delivery of the check codes and the subsequent verification is not obligatory: voters who are paranoid enough or just have a reason not to trust either the idea of e-voting, or the security of their own computers, can take additional measures to first obtain the codes and then to perform verification.

We first introduce some organizational assumptions that seem to be necessary and yet practical enough to be implemented. We emphasize that these assumptions ("the necessary evil") have been approved by the Norwegian e-voting project officials. First, Norway has an ongoing parallel process to implement a national public-key infrastructure. This infrastructure will make it possible for the e-voting project to use eID-cards for the authentication of the voters, but not yet for signing the ballots digitally by 2011. This means that for authentication, the same scheme as the one used on the eID-card has to be used, but otherwise, the pilot project is free to use non-standard public-key cryptosystems. It has to be mentioned though that there are some commercial alternatives available that offer digital signature functionality, but it is unclear whether the public will be willing to trust commercial vendors to sign their ballots.

[2] We stress once more that we are interested in voting over the Internet by using voters' own computers. There are many solutions that involve e-voting in special voting booths, like [RS06], but this will not be the case in Norway.

[3] This is partially because in Norwegian elections, one has a wide choice of options, and can vote for a relatively large number of candidates. Entering many long pseudorandom codes is definitely not user-friendly.

Second, we require the existence of two secure and authenticated channels prechannel and postchannel. Briefly, before the elections, every voter v gets a list of candidates cnd together with integrity check codes $\mathsf{Code}_v[\mathsf{cnd}]$, where the voter-dependent codes are random and independent. The codes are transfered to all voters over a secure and authenticated prechannel that is unlikely to be controlled by the same attacker that controls her computer. This is not restrictive in Norway, where voter registration cards are mailed to all voters in advance (and people trust the postal system). Once more, the delivery of check codes to *all* voters is not necessary: we just assume that a large majority of voters have access to the prechannel by default, and other voters (who are still sufficiently interested in e-voting security) must take a special action to obtain the codes.[4] In principle, there are several alternative ways to build prechannel, but the important requirement is that the check codes should not be known by the voter's computer. Alternatives include using secure Web pages (available only when accessed by using say a smartphone for which the real e-voting client is not available), or SMSs from a fixed mobile number.

Moreover, a real-time channel postchannel (say, SMS, or a Web page that can be checked by using a smartphone) is used to inform the voter about the success of her actions. More precisely, every time she has voted, an integrity check code is sent to her by using postchannel. Note that in Norway, virtually every voter has a mobile phone with the mobile number known to the government—namely, they are extensively used for tax payment—, and thus there exists an efficient postchannel. Those voters whose mobiles have not been registered yet, but who are interested in e-voting security, have to take additional action. However, voters can choose not to do it. Also, a message from postchannel makes sense even if the voter has not received the original codes from the prechannel: in this case, she at least knows that her vote has been recorded.

In addition, the Norwegian e-voting procedure will allow the voters to revote either electronically—such that later e-vote takes precedence over an earlier e-vote—or by (later) participating in conventional paper voting (p-voting), which will take precedence over e-votes. This will provide at least some (though not complete) protection against vote buying and coercion: if either of these has happened, the voter can choose to revote later by using either an e-vote or a p-vote. (The p-voting period will start several days after the e-voting period has ended.) Clearly, if the voter can be both physically coerced (to the extent where she cannot go and participate in p-voting) and she cannot trust her computer, then she cannot be completely protected against all frauds. However, the revoting procedure, which is already implemented in Estonian national e-voting procedure, offers at least some protection against vote buying and coercion. Moreover, due to the existence of the postchannel, a voter will get a timely notification when her vote was altered by her computer. In this case, she can use a different computer to revote, or when necessary, participate in p-voting. Therefore, the combination of a quick-response postchannel and revoting not only guarantees fraud detection but also allows the voters to act on it.

On the flip side, every voter can legally use the same PC to vote many times for (not necessarily) different candidates. This limits the choice of postchannel in our case

[4] This can say organized by delivering voter cards 3 weeks in advance, and if an interested voter has not received it by then, she will contact corresponding authorities.

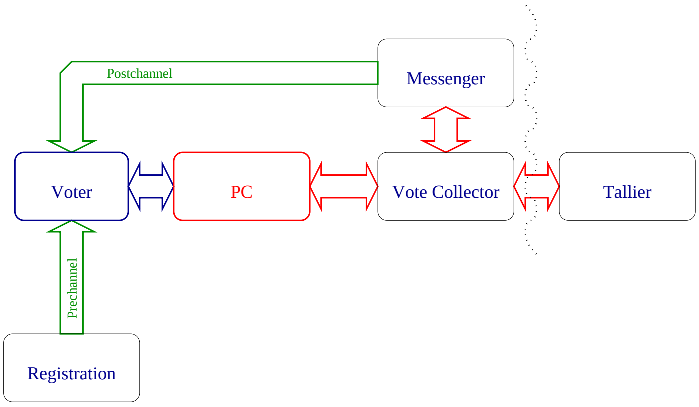

Fig. 1. Parties and channels in the proposed setting

significantly. In particular, it is not secure to use the (possibly malicious) PC itself as the $\mathsf{postchannel}$. Namely, assume that the voter votes for candidate A, then is coerced to vote for B, and then votes again for A. The PC, already knowing the integrity check codes of A and B, can submit a vote for B but display the integrity check code for A. [5]

Given those organizational assumptions, we consider the next setting, see Fig. 1. Voter's ballot (vote) is encrypted and signed (possibly by the attacker), and then sent to the vote collector. (Without loss of generality, in this paper we will assume that there is a single vote collector. In practice, there will be more, but all our protocols will naturally generalize. We will not mention this important point anymore.) The vote collector computes, given an encrypted and signed vote, a ciphertext of the integrity check code $\mathsf{Code}_v[\mathsf{cnd}]$ and sends it to another server (called the messenger). The messenger decrypts the code, and then sends an SMS alert of the type "You, [name], voted at [time], the check code is $\mathsf{Code}_v[\mathsf{cnd}]$ " to the voter over $\mathsf{postchannel}$. The voter verifies the correctness: she complains when she got a wrong message over $\mathsf{postchannel}$ (which say contains a wrong check code), or did not get it all when she voted (in particular when her computer tells her that the vote collector is unavailable), or gets a message when she did not vote. Here, we need that the messenger, who can be behind a firewall, is unaware of the correspondence between the candidates and the corresponding check codes. I.e., a malicious messenger should not collaborate with a malicious vote collector.

In Sect. 4, we propose a cryptographic protocol by which the messenger obtains $\mathsf{Code}_v[\mathsf{cnd}]$. The basic idea of the protocol is as follows. Voter's computer sends to the vote collector two ciphertexts that "encrypt" cnd, one with tallier's public key, and another one with messenger's public key. This is accompanied by a non-interactive zero-knowledge (NIZK) proof of knowledge that the two encrypted values are equal and belong to the correct range (i.e., correspond to a valid candidate). The correspond-

[5] There are other, rather theoretical, ways of achieving coercion-resistance, like the use secure hardware or anonymous channels [JCJ05]. All such means were seen to limit accessibility seriously, and therefore discarded, by the Norwegian government officials.

ing full NIZK proof of knowledge is presented in Sect. 3.2, and its full security proof is given in an appendix. When the NIZK proof of knowledge is correct, the vote collector cryptocomputes, based on the second ciphertext, a ciphertext of $\mathsf{Code}_v[\mathsf{cnd}]$ that is encrypted by messenger's public key. This is done by using a "proxy oblivious transfer" protocol [NPS99] with the additional requirement that the proxy should not get to know the index used by the chooser even when he knows the whole unordered database. The vote collector then sends an encryption of cnd (under tallier's public key) to the tallier, and an encryption of $\mathsf{Code}_v[\mathsf{cnd}]$ (under messenger's public key) to the messenger. In Sect. 4, the new protocol is presented in sufficient details to facilitate an implementation.

We then give an informal security assessment of the full integrity check protocol, and explain our choice of underlying cryptographic primitives and protocols. In this paper, we are *not* going to discuss the operation of tallier since there is a decent amount of literature on this part of the e-voting process. However, we stress that the full e-voting solution in Norway must use additional cryptographic protocols to guarantee better security against malicious voting servers (i.e., vote collectors, talliers, and messengers).

We finish the paper by describing an implementation of the new integrity check protocol, and by giving the timings in the case where there is both a small and a large number of voters and candidates. For example, if there are 80 candidates, the vote collector's throughput is around 2 000 votes per hour on our test machine. The throughput can be increased dramatically by using several vote collectors, better (faster and multicore) CPUs, or even hardware acceleration. In particular, our next task consists of implementing the described protocol in a commercial Hardware Security Module.

Risk Assessment: Avoided Attacks Versus New Attacks. Without the use of the new protocol (or something similar), the voters will not be informed at all whether their e-votes reached the voting servers. Thus, a malicious entity (say some foreign government, or a terrorist organization) can mount a full-scale attack (by writing malicious software that covertly takes over many of voter computers) on the e-voting process and stay undetected. Alternatively, they may reveal themselves after the end of the elections and prove that they in fact manipulated the elections — even that case would be quite devastating. If the integrity protocol of this paper is implemented, such attacks will all be at least detected—given that sufficiently many voters verify the codes—, and the voters can also react by revoting on paper if necessary.

The new protocol also creates some genuinely new attacks. For example, an attacker can take over the prechannel (for example, by distributing fake voter registration cards) or the postchannel (by massively distributing fake SMSs). Both attacks are arguably much more difficult to perform without detection than the takeover of voter computers, since they at least require some physical presence. Attacks on only the postchannel basically amount to the voters receiving bogus messages with (very high probability) wrong check codes. In this case the voters will be alerted, and can revote. Even if both channels are successfully attacked (which is quite difficult by an outsider in the case the $\mathsf{prechannel}$ is implemented by using "snail mail" and the $\mathsf{postchannel}$ is implemented by using SMSs), there is no more harm done than by attacking voter computers: the attacker can then both break correctness (by just reordering codes sent by the $\mathsf{prechannel}$) and anonymity, but both can done trivially by just a malicious computer.

Finally, there are some genuinely new attacks which more hinge on human psychology than cryptography or computer security in general. As an example, voters can falsely claim that they received wrong codes, and thus cause alarm and distrust in elections. Here we emphasize, that the new protocol makes it possible for voters to detect attacks (so that they can revote) but in most of the cases, not to prove their presence. (With some exceptions, such as when they receve incorrectly formatted SMSs from the correct mobile number.) In our own opinion, due to this attack, voter complaints should thus always taken with a grain of salt: if such a complaint occurs, then clearly either there was an attack by an outsider or the voter herself. This should be explained to the voters before the e-voting. Moreover, without such a protocol, any voter can (legitimately) claim that she does not trust e-voting since she may have a virus — and that the government has done nothing to protect her in such a case. We think that the latter complaint is much more valid.[6]

Real-World Status of Proposed Protocol. The new protocol was proposed primarily to answer to the concerns of the relevant government institutions—together with voiced criticism in academia—that Norway is not ready to implement e-voting, in particular because of the seeming threat posed by potentially malicious voter computers. Our work in this direction has appeased our client, who in particular has verified (with the help of cryptographers from Norwegian universities) the correctness of the proposed protocol, though there is still work to be done to make the protocol more efficient. The infrastructural assumptions of this paper (like the existence of two out-of-the-band channels) have been accepted by the government institutions and will be a part of the real e-voting procedure. In fact, the government institutions use the same assumptions as the ones in this paper in their own presentations about the progress of the e-voting project in Norway [Bul09], and have cited our work to show that they know how to protect against malicious voter computers. Criticism from the local academia has also decreased considerably [Gjø10]. However, we (together with other researchers) are still working on constructing more efficient underlying cryptographic protocols.

Due to the lack of space, many details have been omitted. They can be found in the full version [HLV10].

2 Cryptographic Preliminaries

Notation. All logarithms are on basis 2. k is the security parameter, we assume that $k = 80$. $x \leftarrow X$ denotes assignment; if X is a set or a randomized algorithm, then $x \leftarrow X$ denotes a random selection of x from the set or from the possible outputs of X as specified by the algorithm. In the case of integer operations, we will explicitly mention the modulus, like in $z \leftarrow a + b \mod q$. On the other hand, we will omit modular reduction in the case of group operations (like $h \leftarrow g^r$), since in this case depending on the group, reduction may or may not make sense.

Hash Functions, Random Oracle Model and Key Derivation Functions. A function $H : A \to B$ is a hash function if $|B| < |A|$. Within this paper, we usually need

[6] As a side note, due to this issue the voters should not get a possibility to obtain codes from a prechannel after the voting: if they do, they will not be able to revote but can only complain. As we just explained, this would not be a desirable situation.

to assume that H is a random oracle [BR93]. I.e., the value of $H(x)$ is completely unpredictable if one has not seen $H(x)$ before. Random oracles are useful in many cryptographic applications, by making it possible to design efficient cryptographic protocols. In practice, one would instantiate H with a strong cryptographic hash function like SHA2 or the future winner of the SHA3 competition. While there exist schemes which are secure in the random oracle model but which are insecure given any "real" function [CGH98], all known examples are quite contrived. A *key derivation function* $\mathsf{Kdf} : A \rightarrow B$ takes a random element from set A and outputs a pseudorandom element in set B. If $|B| < |A|$ then Kdf is a pseudorandom function, but if $|B| \geq |A|$ then Kdf can be constructed without any cryptographic assumptions. See, e.g., [CFGP06]. For the sake of simplicity, we think of Kdf as a random oracle.

Signature Schemes. A signature scheme $\mathsf{SC} = (\mathsf{Gen}^{\mathsf{sc}}, \mathsf{Sign}, \mathsf{Ver})$ is a triple of efficient algorithms, where $\mathsf{Gen}^{\mathsf{sc}}$ is a randomized key generation function, Sign is a (possibly randomized) signing algorithm and Ver is a verification algorithm. A signature scheme is EUF-CMA (existentially unforgeable against chosen message attacks) secure, if it is computationally infeasible to generate a new signature (i.e., a signature to a message that was not queried from the oracle), given an access to an oracle who signs messages chosen by the adversary. For the purpose of this paper, any of the well-known EUF-CMA secure signature schemes can be used. However, since e-voting is most probably going to use the existing PKI infrastructure of the relevant country, the most prudent approach is to rely on whatever signature scheme has been implemented in the corresponding ID-cards. As an example, one can use either the NIST standard (DSA, [FIP09]) or the PKCS $\sharp 2.1$ standard (RSA-PSS [BR96], also specified in RFC 3447) signature scheme, depending on the country.

Public-Key Cryptosystems. Let $\mathsf{PKC} = (\mathsf{Gen}^{\mathsf{pkc}}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key cryptosystem, where $\mathsf{Gen}^{\mathsf{pkc}}$ is a randomized key generation algorithm that on input $(1^k; r)$, for some random string r, outputs a new secret/public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}^{\mathsf{pkc}}(1^k; r)$, Enc is a randomized encryption algorithm with $c = \mathsf{Enc}_{\mathsf{pk}}(m; r')$, and Dec is a decryption algorithm with $\mathsf{Dec}_{\mathsf{sk}}(c) = m'$. It is required that if $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}^{\mathsf{pkc}}(1^k; r)$ then $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; r')) = m$ for all valid m, r and r'. We denote $\mathsf{Enc}_{\mathsf{pk}}(m; r)$ (resp., $\mathsf{Gen}^{\mathsf{pkc}}(1^k; r)$) for a randomly chosen r also just as $\mathsf{Enc}_{\mathsf{pk}}(m)$ (resp., $\mathsf{Gen}^{\mathsf{pkc}}(1^k)$).

In the case of the Elgamal cryptosystem [Elg85], one fixes a cyclic group $\mathbb{G}$ of a prime order $2^{2k+1} > q > 2^{2k}$, together with a generator g of $\mathbb{G}$. Then, $\mathsf{Gen}^{\mathsf{pkc}}(1^k)$ generates a random $\mathsf{sk} \leftarrow \mathbb{Z}_q$, and sets $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$. On input $m \in \mathbb{G}$, the encryption algorithm generates a new random $r \leftarrow \mathbb{Z}_q$, and sets $\mathsf{Enc}_{\mathsf{pk}}(m; r) := (m \cdot \mathsf{pk}^r, g^r)$. On input $c = (c_1, c_2) \in \mathbb{G}^2$, the decryption algorithm outputs $m' \leftarrow c_1/c_2^{\mathsf{sk}}$. Elgamal is multiplicatively homomorphic. I.e., $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m_1; r_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2; r_2)) = m_1 \cdot m_2$ for $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{Gen}^{\mathsf{pkc}}(1^k)$. Here, the product of ciphertexts is defined coordinate-wise, $(m_1 \cdot \mathsf{pk}^{r_1}, g^{r_1}) \cdot (m_2 \cdot \mathsf{pk}^{r_2}, g^{r_2}) = (m_1 m_2 \cdot \mathsf{pk}^{r_1+r_2}, g^{r_1+r_2})$.

One can implement Elgamal on top of several different group families, like order-q subgroups of $\mathbb{Z}_p$ where p is a large prime (say $p > 2^{1024}$). How to choose p's and q's in this case is probably the best explained in the standard FIPS 186-3 [FIP09]. In this case, one generates g by first choosing a random element $h \leftarrow \mathbb{Z}_p^* \setminus \{1\}$, and then sets $g \leftarrow h^{(p-1)/q}$. Moreover, all group operations are done modulo p, while operations in exponents are done modulo q (like $c \leftarrow g^{r+r' \bmod q} \bmod p$ though in this case the reduction modulo q is implicitly done for us). However, we will not mention reduction modulo p explicitly since one could use other groups. In particular, the most efficient known implementation of Elgamal cryptosystem uses elliptic-curve groups over finite fields that are recommended in [FIP09]. In this case, the group elements can be represented by ≤ 256 bits and thus Elgamal ciphertext can be represented by ≤ 512 bits. In any case, we assume that the underlying group $\mathbb{G}$ is uniquely described by some short string $descr(\mathbb{G})$. For example, in the first case, $descr(\mathbb{G}) = (p, q)$.

Non-Interactive Zero-Knowledge Proof of Knowledge. Let L be an arbitrary NP-language, and let $R = \{(x, y)\}$ where $x \in L$ and y is the corresponding NP-witness. A Σ-protocol (P_1, V_1, P_2, V_2) for a relation R is a three-message protocol between a prover and a verifier (both stateful), such that (1) the prover and verifier have a common input x, and the prover has a private input y, (2) the prover sends the first (P_1) and the third (P_2) message, and the verifier sends the second message V_1, after which the verifier either rejects or accepts (by using V_2), (3) the protocol is public-coin: i.e., the verifier chooses her response V_1 completely randomly from some predefined set, (4) the protocol satisfies the security properties of correctness, special honest-verifier zero-knowledge (SHVZK), and special soundness. We identify a protocol run with the tuple $(x; \mathfrak{i}, \mathfrak{c}, \mathfrak{r})$ where $(\mathfrak{i}, \mathfrak{c}, \mathfrak{r})$ are the three messages of this protocol. A protocol run is *accepting*, if an honest verifier accepts this run, i.e., on having input x and seeing the messages $\mathfrak{i}$, $\mathfrak{c}$, and $\mathfrak{r}$. See App. A for security definitions.

Based on an arbitrary Σ-protocol, one can build a non-interactive zero-knowledge (NIZK) proof of knowledge in the random oracle model, by using the Fiat-Shamir heuristic [FS86]. I.e., given $(x, y) \in R$ and a random oracle H [BR93], the corresponding NIZK proof of knowledge π consists of $(\mathfrak{i}, \mathfrak{c}, \mathfrak{r})$, where $\mathfrak{i} \leftarrow P_1(x, y)$, $\mathfrak{c} \leftarrow H(param, x, \mathfrak{i})$, and $\mathfrak{r} \leftarrow P_2(x, y, \mathfrak{c})$, where $param$ is the set of public parameters (like the description of the underlying group, etc). See [CDS94].

We use the next common notation. A NIZK proof of knowledge $\mathrm{PK}(R(\dots))$ is for relation R, where the prover has to prove the knowledge of variables denoted by Greek letters. All other variables are known to both the prover and the verifier. For example, $\mathrm{PK}(y = \mathsf{Enc}_{\mathsf{pk}}(\mu; \rho) \wedge \mu \in \{0, 1\})$ denotes a NIZK proof of knowledge that the prover knows a Boolean μ and some ρ such that $y = \mathsf{Enc}_{\mathsf{pk}}(\mu; \rho)$.

NIZK Proof of Equality of Plaintexts. Let $\mathsf{PKC} = (\mathsf{Gen}^{\mathsf{pkc}}, \mathsf{Enc}, \mathsf{Dec})$ be the Elgamal cryptosystem. Fix $\mathbb{G}$, g, and two key pairs $(\mathsf{sk}_1, \mathsf{pk}_1) \in \mathsf{Gen}^{\mathsf{pkc}}(1^k)$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \in \mathsf{Gen}^{\mathsf{pkc}}(1^k)$. Let H be a random oracle. The NIZK proof of equality of plaintext is a NIZK proof of knowledge $\mathrm{PK}(e_1 = \mathsf{Enc}_{\mathsf{pk}_1}(g^{\mu}; \rho_1) \wedge e_2 = \mathsf{Enc}_{\mathsf{pk}_2}(g^{\mu}; \rho_2))$, that e_1 and e_2 encrypt the same plaintext under a different key. The next NIZK proof of equality of plaintexts is standard (with history going back at least to [Sch91]): generate random $\mu', \rho'_1, \rho'_2 \leftarrow \mathbb{Z}_q$. Set $\mathfrak{i}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}_1}(g^{\mu'}; \rho'_1)$, $\mathfrak{i}_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}_2}(g^{\mu'}; \rho'_2)$, $\mathfrak{c} \leftarrow H(descr(\mathbb{G}), g, e_1, e_2, \mathfrak{i}_1, \mathfrak{i}_2)$, $\mu'' \leftarrow \mu' + \mu \cdot \mathfrak{c} \mod q$, $\rho''_1 \leftarrow \rho'_1 + \rho_1 \cdot \mathfrak{c} \mod q$, $\rho''_2 \leftarrow \rho'_2 + \rho_2 \cdot \mathfrak{c} \mod q$. The NIZK proof of knowledge is equal to the tuple $(\mathfrak{i}_1, \mathfrak{i}_2; \mathfrak{c}; \mu'', \rho''_1, \rho''_2)$. The verifier accepts its correctness iff $\mathsf{Enc}_{\mathsf{pk}_1}(g^{\mu''}; \rho''_1) = \mathfrak{i}_1 \cdot e_1^{\mathfrak{c}}$ and $\mathsf{Enc}_{\mathsf{pk}_2}(g^{\mu''}; \rho''_2) = \mathfrak{i}_2 \cdot e_2^{\mathfrak{c}}$. Alternatively, one can define the NIZK proof of knowledge just to be the tuple $(\mathfrak{c}; \mu'', \rho''_1, \rho''_2)$, where $\mathfrak{i}_1, \mathfrak{i}_2, \mathfrak{c}, \mu'', \rho''_1, \rho''_2$ are computed as earlier. In this case, one just verifies that $\mathfrak{c} = H(descr(\mathbb{G}), g, e_1, e_2, \mathsf{Enc}_{\mathsf{pk}_1}(g^{\mu''}; \rho''_1) \cdot e_1^{-\mathfrak{c}}, \mathsf{Enc}_{\mathsf{pk}_2}(g^{\mu''}; \rho''_2) \cdot e_2^{-\mathfrak{c}})$.

Range Proof in Exponents. In the following we need a range proof in exponents, i.e., a NIZK proof of knowledge $\mathrm{PK}(e = \mathsf{Enc}_{\mathsf{pk}}(g^{\mu}; \rho) \wedge \mu \in [0, \mathsf{CC}])$ for some positive integer CC. In the discrete logarithm setting the most efficient known range proof in exponents was proposed in [LAN02]. (Another range proof in exponents that is comparably communication-efficient, was recently proposed in [CCS08]. However, the latter proof uses pairings and is thus computationally less efficient.) The communication complexity of the range proof in exponents from [LAN02] is logarithmic in CC. In the general case (when assuming stronger assumptions), there exist range proofs in exponents with communication that is essentially independent of CC [Bou00,Lip03]. However, if the value of CC is relatively small, the latter proofs actually are less efficient than the proof of [LAN02]. See Sect. B for a full description of the range proof of [LAN02].

We specify this proof fully in Sect. 3.1, where we present a NIZK proof of knowledge that uses this range proof in exponents as a subproof.

3 Cryptographic Tools

3.1 Strong Proxy Oblivious Transfer

In a 1-out-of-n proxy oblivious transfer protocol, $(n, 1)$-POT [NPS99], for ℓ-bit strings, the chooser has an index $x \in \{0, \dots, n-1\}$ and a public key pk, the sender has pk and a database $f = (f_0, \dots, f_{n-1})$ with $f_i \in \{0, 1\}^{\ell}$, and the proxy has a decryption key. At the end of the protocol, the proxy obtains f_x. A two-message $(n, 1)$-POT protocol $\Gamma = (\mathsf{Gcpir}, \mathsf{Query}, \mathsf{Reply}, \mathsf{Answer})$ is a quadruple of polynomial-time algorithms, with Gcpir and Query being randomized, such that for any r, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gcpir}(1^k; r)$, x, f and r', $\mathsf{Answer}_{\mathsf{sk}}(x, \mathsf{Reply}_{\mathsf{pk}}(f, \mathsf{Query}_{\mathsf{pk}}(x; r'))) = f_x$. As before, we denote $\mathsf{Gcpir}(1^k) := \mathsf{Gcpir}(1^k; r)$ and $\mathsf{Query}_{\mathsf{pk}}(x) := \mathsf{Query}_{\mathsf{pk}}(x; r')$ for randomly chosen r and r'. Here, the proxy generates the key pair $(\mathsf{sk}, \mathsf{pk})$ and sends pk to the chooser and to the sender. The chooser then sends $\mathsf{Query}_{\mathsf{pk}}(x)$ to the sender, who sends $\mathsf{Reply}_{\mathsf{pk}}(f, \mathsf{Query}_{\mathsf{pk}}(x))$ to the proxy. The proxy obtains f_x by applying $\mathsf{Answer}_{\mathsf{sk}}$.

Semisimulatable Privacy for Strong Proxy Oblivious Transfer. Let $\Gamma = (\mathsf{Gcpir}, \mathsf{Query}, \mathsf{Reply}, \mathsf{Answer})$ be a 2-message $(n, 1)$-POT protocol. Within this work we use the convention of many previous papers [NP99] on oblivious transfer protocols to only require (semisimulatable) privacy in the malicious model. I.e., chooser's privacy is guaranteed in the sense of indistinguishability (CPA-security), while sender's privacy is guaranteed in the sense of simulatability. We note that POT's privacy definition is a simple modification of the standard OT's semisimulatable privacy definition.

We give an informal definition of semisimulatable privacy. For the *CPA-security* (i.e., the privacy) *of the chooser*, (1) no malicious nonuniform probabilistic polynomial-time sender should be able to distinguish, with non-negligible probability, between the distributions $(\mathsf{pk}, \mathsf{Query}_{\mathsf{pk}}(x_0))$ and $(\mathsf{pk}, \mathsf{Query}_{\mathsf{pk}}(x_1))$ that correspond to any two of chooser's inputs x_0 and x_1 that are chosen by the sender, and (2) no malicious nonuniform probabilistic polynomial-time proxy should be able to distinguish, with non-negligible probability, between the distributions $(\{f\}, \mathsf{sk}, \mathsf{pk}, \mathsf{Reply}_{\mathsf{pk}}(f, \mathsf{Query}_{\mathsf{pk}}(x_0)))$ and $(\{f\}, \mathsf{sk}, \mathsf{pk}, \mathsf{Reply}_{\mathsf{pk}}(f, \mathsf{Query}_{\mathsf{pk}}(x_1)))$ that correspond to any two of chooser's inputs x_0 and x_1 that are chosen by the sender. (Here, $\{f\}$ denotes an unordered version of f.) For *sender-privacy*, we require the existence of an *unbounded* simulator that, given pk, chooser's message Q^*_{pk} and proxy's legitimate output corresponding to this message, generates sender's message that is *statistically* indistinguishable from honest sender's message $\mathsf{Reply}_{\mathsf{pk}}$ in the real protocol; here Q^*_{pk} does not have to be correctly computed. As in earlier papers that use semisimulatable privacy [NP99], unboundedness is required mostly so that the simulator could "decrypt" chooser's first message. A protocol is *private* if it is both chooser-private and sender-private.

Instantiation. In the proposed e-voting protocol, the database size n corresponds to the number of candidates, and therefore it is usually small (say $n \leq 64$). This means that it is sufficient to use a POT protocol with linear-in-n communication. (In the case when n is larger, one could consider relying on an underlying oblivious transfer protocol with small polylogarithmic communication like those of [Lip05a,GR05].) On the other hand, it is important to minimize sender's computation. Given those considerations, we base the new POT protocol on the AIR oblivious transfer protocol [AIR01]. The result has (in the case of a small n) good communication and computation, is based on a well-known security assumption (Decisional Diffie-Hellman), and allows one to construct efficient NIZK proofs of knowledge.

Let $\mathsf{PKC} = (\mathsf{Gen}^{\mathsf{pkc}}, \mathsf{Enc}, \mathsf{Dec})$ be the Elgamal cryptosystem, and let $g \in \mathbb{G}$ be a fixed generator of the plaintext group. Chooser's private input is $x \in \{0, \dots, n-1\}$, and sender's private input is $f = (f_0, \dots, f_{n-1})$ for $f_i \in \{0,1\}^\ell$ with (relatively) small ℓ. The new $(n,1)$-strong POT protocol consists of the next steps:

1. The proxy sets $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}^{\mathsf{pkc}}(1^k)$, and sends pk to the chooser and the sender.
2. For $\rho \leftarrow \mathbb{Z}_q$, the chooser sets $e \leftarrow \mathsf{Enc}_{\mathsf{pk}}(g^x; \rho)$, and sends $\mathsf{Query}_{\mathsf{pk}}(x) \leftarrow e$ to the sender.
3. The sender does on input pk and $\mathsf{Query}_{\mathsf{pk}}(x) = e$:
   (a) For every $i \in \{0, \dots, n-1\}$: generate new random values $r_i, r'_i \leftarrow \mathbb{Z}_q$, set $e_i \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(g^i; 1)/e)^{r_i} \cdot \mathsf{Enc}_{\mathsf{pk}}(g^{f_i}; r'_i)$.
   (b) Send $\mathsf{Reply} = \mathsf{Reply}_{\mathsf{pk}}(f, (\mathsf{pk}, e)) \leftarrow \{e_0, \dots, e_{n-1}\}$ to the proxy, where the set elements in Reply are given in a random order.
4. For all elements e' in the set Reply, the proxy computes $y \leftarrow \mathsf{Dec}_{\mathsf{sk}}(e')$. He finds an y, such that the discrete logarithm z of y on basis g is small. He outputs z as $\mathsf{Answer}_{\mathsf{sk}}(x, \mathsf{Reply})$.

Note that the sender can precompute the values $\mathsf{Enc}_{\mathsf{pk}}(g^i; 1)$ and $\mathsf{Enc}_{\mathsf{pk}}(g^{f_i}; 1)$, and therefore her online computation is dominated by $2n$ exponentiations in $\mathbb{G}$. (Note that in the actual implementation, this protocol will also be accompanied with a NIZK proof that x is in the correct range.)

Computing discrete logarithm is efficient when all database elements are small, say $\ell \leq 5$, and can be just done by table-lookup by comparing all values y with values g^i for small i. (Discrete logarithm step could be avoided by using an additively homomorphic cryptosystem like [Pai99]. However, known additively homomorphic cryptosystems are otherwise considerably less efficient than Elgamal.) Moreover, with an overwhelming probability, there is exactly one element e_j such that the discrete logarithm of $\mathsf{Dec}_{\mathsf{sk}}(e_j)$ is small. Thus, the proxy can just decrypt all values e', and then check them against a precomputed table lookup of g^i for small values of i; the comparison step will take $\Theta(n \cdot \log n)$ elementary operations. Since n is very small, this part is considerably faster than decrypting n different ciphertexts. When using say Lipmaa's [Lip05a] oblivious transfer-protocol based POT, the messenger will only have to decrypt a single element and then make $\Theta(\log n)$ comparisons by using binary search. However, the cost of computing Answer will be higher. Our choice is supported by implementation timings (Sect. 7) that show that proxy's time load is much smaller than that of sender. Finally, note that the messenger has to decrypt in average 50% of the elements, and thus his online cost is dominated by $\approx n/2$ exponentiations.

This protocol is clearly both correct and private, given that Elgamal is CPA-secure [AIR01]. See App. C for more.

Weak POT for Large Database Elements. We also need to use proxy oblivious transfer in a situation, where the database elements are significantly longer, such that computing discrete logarithm (as in the proposed strong POT protocol) will not anymore possible. However, in our application, the proxy is allowed to know an *unordered* version $\{f\}$ of the database f. More precisely, the proxy knows an unordered tuple $F := \{g^{f_0}, \dots, g^{f_{n-1}}\}$, and for efficiency reasons, we assume that this tuple is sorted. After the end of the POT protocol, he obtains g^{f_x} for some unknown x, and he can verify whether g^{f_x} is equal to some element of F by using binary search, in time $\Theta(\log n)$. However, that does not help him in determining x since F does not contain any information about indexes. We call this protocol a weak oblivious transfer protocol.

3.2 New NIZK Proof of Knowledge

We need a NIZK proof of knowledge $\mathrm{PK}(e = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho) \wedge e' = \mathsf{Query}_{\mathsf{pkm}}(\mu; \rho') \wedge \mu \in [0, \mathsf{CC}])$, where we use the Elgamal cryptosystem and the new proxy oblivious transfer protocol. Since in the new POT protocol, the first message is just $\mathsf{Enc}_{\mathsf{pkt}}(g^{\mu})$, we need to prove an AND of two statements, that e and e' "encrypt" the same value g^{μ} (under different keys), and that e' encrypts a value g^{μ} where $\mu \in [0, \mathsf{CC}]$. We already presented both proofs separately. To "and" the two NIZK POKs, one uses the standard technique from [CDS94]. For the sake of completeness, the full interactive version of this zero-knowledge proof is given in Prot. 1. We need actually a NIZK proof of knowledge version of it, which is presented later as Prot. 2.

Complexity. In Prot. 1, prover's computation is dominated by (at most) $3\lambda + 4$ public-key encryptions and λ exponentiations. Since Elgamal is used, if necessary most of the prover's computation can be done beforehand. However, this should not be necessary in our application, where it is perfectly fine that it takes a minute for the voter's computer to finish computation. Verifier's computation is dominated by $2\lambda + 3$ encryptions, λ of which can be precomputed, and $2\lambda + 2$ exponentiations. In real-world voting, we can in most cases assume that $\lambda \leq 6$, thus verifier's computation is dominated by ≤ 15 encryptions and ≤ 14 exponentiations.

Security. The security of Prot. 1 is a straightforward corollary of known results. However, for the sake of completeness we provide a complete proof.

Theorem 1. *Prot. 1 is a correct, specially sound and SHVZK proof of knowledge for* $\mathrm{PK}(e = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho) \wedge e' = \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu}; \rho') \wedge \mu \in [0, \mathsf{CC}])$.

Full proof of this theorem is given in App. D.

System parameters: $\mathbb{G}$, q, g.
Common inputs: CC and $\lambda := \lfloor \log_2 \mathsf{CC} \rfloor$, pkt, pkm, e'.
Prover's input: μ, ρ'.

1. Prover does:
   (a) Compute the values $\mu_j \in \{0, 1\}$ such that $\mu = \sum_{j=0}^{\lambda} \mu_j \mathsf{CC}_j$ with $\mathsf{CC}_j \leftarrow \lfloor (\mathsf{CC} + 2^j)/2^{j+1} \rfloor$.
   (b) For $j \in \{0, \dots, \lambda\}$ do:
      i. Generate random $\rho_j, \rho'_j \leftarrow \mathbb{Z}_q$, set $e_j \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_j}; \rho_j)$.
      ii. If $\mu_j = 0$ then: Set $\mathfrak{i}_{0,j} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(1; \rho'_j)$, $\mathfrak{c}_{1,j} \leftarrow \mathbb{Z}_{2^k}$, $\mathfrak{r}_{1,j} \leftarrow \mathbb{Z}_q$, $\mathfrak{i}_{1,j} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j}) \cdot (\mathsf{Enc}_{\mathsf{pkt}}(g; 0)/e_j)^{\mathfrak{c}_{1,j}}$.
      iii. Else if $\mu_j = 1$ then: Set $\mathfrak{i}_{1,j} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(1; \rho'_j)$, $\mathfrak{c}_{0,j} \leftarrow \mathbb{Z}_{2^k}$, $\mathfrak{r}_{0,j} \leftarrow \mathbb{Z}_q$, $\mathfrak{i}_{0,j} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j})/e_j^{\mathfrak{c}_{0,j}}$.
   (c) Generate random $\mu_{\mathsf{and}}, \rho_{\mathsf{and},1}, \rho_{\mathsf{and},2} \leftarrow \mathbb{Z}_q$. Set $\mathfrak{i}_{2,1} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_{\mathsf{and}}}; \rho_{\mathsf{and},1})$, $\mathfrak{i}_{2,2} \leftarrow \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu_{\mathsf{and}}}; \rho_{\mathsf{and},2})$.

   Send $\mathfrak{i} \leftarrow (e_0, \dots, e_\lambda, (\mathfrak{i}_{0,0}, \mathfrak{i}_{1,0}), \dots, (\mathfrak{i}_{0,\lambda}, \mathfrak{i}_{1,\lambda}), \mathfrak{i}_{2,1}, \mathfrak{i}_{2,2})$ to the verifier.
2. Verifier does: Set $\mathfrak{c} \leftarrow \mathbb{Z}_{2^k}$, send $\mathfrak{c}$ to the prover.
3. Prover does for $j \in \{0, \dots, \lambda\}$:
   (a) If $\mu_j = 0$ then: Set $\mathfrak{c}_{0,j} \leftarrow \mathfrak{c} - \mathfrak{c}_{1,j} \mod 2^k$, $\mathfrak{r}_{0,j} \leftarrow \rho'_j + \mathfrak{c}_{0,j} \cdot \rho_j \mod q$.
   (b) Else if $\mu_j = 1$ then: Set $\mathfrak{c}_{1,j} \leftarrow \mathfrak{c} - \mathfrak{c}_{0,j} \mod 2^k$, $\mathfrak{r}_{1,j} \leftarrow \rho'_j + \mathfrak{c}_{1,j} \cdot \rho_j \mod q$.

   Let $\rho' \leftarrow \sum \rho_j \mathsf{CC}_j \mod q$ (i.e., $e \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho')$). Set $\mathfrak{r}_3 \leftarrow \mu_{\mathsf{and}} + \mathfrak{c} \cdot \mu \mod q$, $\mathfrak{r}_{4,1} \leftarrow \rho_{\mathsf{and},1} + \mathfrak{c} \cdot \rho \mod q$, $\mathfrak{r}_{4,2} \leftarrow \rho_{\mathsf{and},2} + \mathfrak{c} \cdot \rho' \mod q$. Send $\mathfrak{r} \leftarrow (\mathfrak{c}_{0,0}, \dots, \mathfrak{c}_{0,\lambda}, (\mathfrak{r}_{0,0}, \mathfrak{r}_{1,0}), \dots, (\mathfrak{r}_{0,\lambda}, \mathfrak{r}_{1,\lambda}), \mathfrak{r}_3, \mathfrak{r}_{4,1}, \mathfrak{r}_{4,2})$ to the verifier.
4. Verifier does:
   (a) Let $e \leftarrow \prod_{j=0}^{\lambda} e_j^{\mathsf{CC}_j}$.
   (b) For $j \in \{0, \dots, \lambda\}$:
      i. Set $\mathfrak{c}_{1,j} \leftarrow \mathfrak{c} - \mathfrak{c}_{0,j} \pmod{2^k}$.
      ii. If $\mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j}) \neq \mathfrak{i}_{0,j} \cdot e_j^{\mathfrak{c}_{0,j}}$ or $\mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j}) \neq \mathfrak{i}_{1,j} \cdot (e_j/\mathsf{Enc}_{\mathsf{pkt}}(g; 0))^{\mathfrak{c}_{1,j}}$ then: reject.
   (c) If $\mathsf{Enc}_{\mathsf{pkt}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,1}) \neq \mathfrak{i}_{2,1} \cdot e^{\mathfrak{c}}$ or $\mathsf{Enc}_{\mathsf{pkm}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,2}) \neq \mathfrak{i}_{2,2} \cdot (e')^{\mathfrak{c}}$ then: reject.

   Otherwise: accept.

Protocol 1: Interactive version of the required zero-knowledge proof

1. Prover has inputs $(descr(\mathbb{G}), g, \mathsf{CC}, \mathsf{pkt}, \mathsf{pkm}, e')$. He computes $\mathfrak{i}$ as in Prot. 1, but then he sets $\mathfrak{c} \leftarrow H(descr(\mathbb{G}), g, \mathsf{CC}, \mathsf{pkt}, \mathsf{pkm}, e', \mathfrak{i})$, and computes $\mathfrak{r}$ that corresponds to this value of $\mathfrak{c}$. The NIZK proof of knowledge is equal to $\pi \leftarrow (e_0, \dots, e_\lambda, \mathfrak{c}, \mathfrak{r})$.
2. Verifier has inputs $(descr(\mathbb{G}), g, \mathsf{CC}, \mathsf{pkt}, \mathsf{pkm}, e', \pi)$. On input π, she computes the missing elements of $\mathfrak{i}$ exactly as in the proof of the SHVZK property of Prot. 1. Verifier accepts if and only if $\mathfrak{c} = H(descr(\mathbb{G}), g, \mathsf{CC}, \mathsf{pkt}, \mathsf{pkm}, e', \mathfrak{i})$.

Protocol 2: NIZK proof of knowledge version of Prot. 1

NIZK Proof of Knowledge Version. Since Prot. 1 is correct, specially sound and SHVZK, we can now use the Fiat-Shamir heuristic to construct a secure NIZK proof of knowledge. This version is depicted by Prot. 2. Note that when Elgamal in the subgroups of $\mathbb{Z}_p$ is used then $descr(\mathbb{G}) = (p, q)$ and thus $\mathfrak{c} \leftarrow H(p, q, g, \dots)$.

4 Cryptographic Protocol for E-Vote Integrity

The voting process consists of a number of voters $\mathcal{V}$, their PCs, one or more messengers (Messenger), one or more vote collectors (VC) and one or more talliers (Tallier). A voter enters her preferred candidate number—by using a user-friendly GUI—to her PC, that then runs a vote registration protocol with the vote collectors. Vote collectors collect the votes, and send their collection to the talliers after the voting period has finished. Within this paper, we are not going to specify most of the internal working of the vote collectors or the vote talliers since there exists already an extensive literature on that.

In this paper, we focus on the case when the voter's PC is dishonest. Clearly, if voters would only have access to their PCs, no security could be achieved at all. Therefore, in addition we need the presence of some independent channels accessible by the voters. As an example, in many countries, before any elections the voters will anyway receive a paper voter registration card. We can make use of this channel (prechannel), by adding extra information on this acknowledgment. In addition, most of the voters have access to more than one connected device. The second device (postchannel) may be something simple, like a mobile phone, even if it cannot perform any complex cryptographic operations, but can still guarantee real-time reception of messages.

Description of Protocol. Assume that we have $\mathsf{CC} + 1 > 0$ candidates, and every candidate has been assigned a number $\mathsf{cnd} \in \{0, \dots, \mathsf{CC}\}$. Since CC is small, we are going to use the AIR-based proxy oblivious transfer protocol (Gcpir, Query, Reply, Answer) and the Elgamal cryptosystem ($\mathsf{Gen}^{\mathsf{pkc}}$, Enc, Dec). In particular since Elgamal is multiplicatively homomorphic, instead of the candidate cnd we encrypt g^{cnd}, where g is a fixed generator of Elgamal's plaintext group. (If an additively homomorphic cryptosystem were used, one could instead just encrypt cnd. However, such cryptosystems tend to be less efficient in practice.) The protocol is depicted by Prot. 3.

Complexity. Vote collector's computation is dominated by the verification of the NIZK proof of knowledge (which takes at most $2\lambda + 3$ encryptions and $2\lambda + 2$ exponentiations), and by the execution of the sender's part in the POT protocol that is dominated by $2(\mathsf{CC}+1)$ encryptions ($\mathsf{CC}+1$ of which can be precomputed) and $\mathsf{CC}+1$ exponentiations. On top of that, the vote collector has to verify a signature, and sign her message

System parameters: $\mathbb{G}, g, q, H$.
Voter's inputs: encryption keys of tallier, messenger, her own private signature key, voter collector's signature verification key.
Vote collector's inputs: encryption key of messenger, his own private signature key, voters' signature verification keys.
Tallier's inputs: his own private decryption key, vote collector's signature verification key.
Common inputs: $\mathsf{CC} + 1$ candidates $c \in [0, \mathsf{CC}]$, $\lambda := \lfloor \log_2 \mathsf{CC} \rfloor$.

1. Before elections:
   (a) $(\mathbb{G}, q, g)$ and H are fixed and published by a trusted server.
   (b) Some server (be it vote collector or a separate server) generates for every voter-candidate pair (v, cnd) a uniformly random string $R_v[\mathsf{cnd}] \leftarrow \mathbb{Z}_q$, and sets $\mathsf{Code}_v[\mathsf{cnd}] \leftarrow \mathsf{Kdf}(g^{R_v[\mathsf{cnd}]})$ where Kdf is a key derivation function. It sends signed codes $\mathsf{Code}_v[\mathsf{cnd}]$ to corresponding voters (by using $\mathsf{prechannel}$) and to the messengers (in numerically sorted order), and signed values $R_v[\mathsf{cnd}]$ to the vote collectors. // In practice, only the first few, say 25 bits of $\mathsf{Code}_v[\mathsf{cnd}]$ are sent.
2. When voter v enters a candidate number cnd (by using favorite UI) to voter's PC:
   (a) Voter's PC does:
      i. He generates the first message $e' \leftarrow \mathsf{Query}_{\mathsf{pkm}}(\mathsf{cnd})$ of the new weak proxy oblivious transfer protocol.
      ii. He generates a non-interactive zero-knowledge proof $\pi = \mathrm{PK}(e = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho) \wedge e' = \mathsf{Query}_{\mathsf{pkm}}(\mu; \rho') \wedge \mu \in [0, \mathsf{CC}])$ that both e and e' correspond to the same *valid* candidate (see Prot. 2).
      iii. He signs (e', π) by using his secret signing key sk_v, $s \leftarrow \mathsf{Sign}_{\mathsf{sk}_v}(e, e', \pi)$.
      iv. He then sends (e', π, s) to the vote collector. (Note that π contains the list $(e_0, \ldots, e_\lambda)$ with $e_j = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_j})$ and $\mu_j \in \{0, 1\}$.)
   (b) After receiving a ballot from the PC, the vote collector does:
      i. He verifies both the signature and the zero-knowledge proof (as specified in Prot. 2). If both verifications are fine, it computes the second message $r \leftarrow \mathsf{Reply}_{\mathsf{pkm}}(e', \mathsf{Code}_v)$ of the POT protocol. Recall here that r consists of a number of randomly-reordered ciphertexts.
      ii. He sends to the voter's PC a signed message accept or reject.
      iii. He signs r and sends it to the messenger.
   (c) After receiving a message from the VC, the messenger does:
      – She verifies the signature on r. She complains when it does not verify.
      – Otherwise, she "decrypts" $g^{R_v[\mathsf{cnd}]} \leftarrow \mathsf{Answer}_{\mathsf{skm}}(\mathsf{cnd}, \mathsf{Reply})$, where skm is messenger's secret key, and obtains $\mathsf{Code}_v[\mathsf{cnd}] \leftarrow \mathsf{Kdf}(g^{R_v[\mathsf{cnd}]})$. (The procedure for this is specified in Sect. 3.1.) It also alerts the voter by using $\mathsf{postchannel}$ with the value of $\mathsf{Code}_v[\mathsf{cnd}]$.
   (d) When receiving a message from $\mathsf{postchannel}$, the voter checks that $\mathsf{Code}_v[\mathsf{cnd}]$ is correct, as in Step 5 if the ideal-world vote registration protocol. The voter also checks that her legitimate voting acts are accompanied by a $\mathsf{postchannel}$ message, and that she receives no spurious messages.
3. After the election period has ended, the vote collector sends all values $e = \prod e_j^{\mathsf{CC}_j}$, signed with his own private key, to the tallier. The tallier operates by using a suitable e-voting procedure to deduce the winner.

Protocol 3: The new protocol between a voter, her computer, vote collector, and messenger to the messenger. Given say $\mathsf{CC} + 1 = 63$ candidates (then $\lambda = 5$), her computation is thus dominated by $2\lambda + 3 + 2(\mathsf{CC} + 1) = 139$ encryptions and $2\lambda + 2 + \mathsf{CC} + 1 = 75$ exponentiations, some of which can be precomputed. Note that the bulk of vote collector's computation goes to computing her part of the POT protocol. This seems to be inevitable since most of the known oblivious transfer protocols (the only exception is [Lip09]) requite linear computation. On the other hand, while the description of the NIZK proof of knowledge is seemingly more complex, it is considerably more efficient than the POT protocol.

Discussion. If $R_v[\mathsf{cnd}]$ is long (say ≥ 20 bits) then computing Answer requires the computation of discrete logarithm with time complexity of $\geq 2^{10}$ steps by using Pollard's ρ algorithm. Our solution to this is that instead of $R_v[\mathsf{cnd}]$, the check code is $\mathsf{Code}_v[\mathsf{cnd}] = \mathsf{Kdf}(g^{R_v[\mathsf{cnd}]})$. This means that the values $\mathsf{Code}_v[\mathsf{cnd}]$ will be sent over $\mathsf{prechannel}$, too. On the other hand, this step is done by client's computer only once in a while and thus is not a bottleneck, and it may even be desirable to prevent DDoS attacks, by forcing client's computer to perform some work per every cast vote. Also, note that the tallier obtains a ciphertext of g^{cnd}. Here, computing of discrete logarithm is again simple since cnd is small (it can be done by using table-lookup).

5 Security of Integrity Protocol

We now state the security of the e-voting process, given the new integrity protocol. We will give informal security arguments, leaving formal proofs for further work. In all following paragraphs, we consider the case when one party is dishonest, but all other parties are honest. This assumption is not necessary when one additionally implements protocols that guarantee security against malicious servers. For example, one can use standard mixnets, but as said, this is not the topic of the current paper. Note that all parties can blindly refuse accept votes, claiming to have troubles with connection, but this is unavoidable.

Security against Voter Computer. There are no privacy guarantees against malicious voter's PC. However, by doing proper checks, a voter can clearly verify that the voter's PC has voted for a wrong candidate, or did not vote at all. In the case the verification fails, voters can participate in later paper voting that overrides the results of the e-voting.

Security against Vote Collector. Vote collector only sees encrypted data, and thus here privacy is guaranteed. She cannot change votes (since they are signed).

Security against Messenger. Messenger only sees the codes, and which code the voter is voting for right now, but nothing else. Thus, privacy is covered except in the next sense: the messenger can test, in the case of a revote, whether this time the voter is voting for a new or an old candidate. The messenger can also not send a $\mathsf{postchannel}$ message based on such tests. The messenger can also send back a message that corresponds to an earlier vote by the same candidate, but this will be detected by the voter.

Security against Tallier. Tallier only obtains a list of all encrypted ballots, signed by the vote collector. The tallier cannot thus breach the privacy. To guarantee some robustness/integrity while tallying, one can use some well-known cryptographic protocols (for example, mixnets).

6 Discussion

While choosing the underlying primitives and protocols, we considered efficiency to be the most important factor, closely followed by the simplicity of implementation and standardness of security assumptions. Next we will try to motivate our choices.

Public-key Cryptosystem. While Elgamal is only multiplicatively homomorphic, it is several times more efficient than the known additively homomorphic cryptosystems like [Pai99], especially in decryption. In addition, NIZK proofs of knowledge based on known additively homomorphic cryptosystems tend to be less efficient. Slower encryption, decryption and NIZK verifications would make vote collector's computations much more costly. On the other hand, by using standard tricks, we were able to minimize the drawbacks of Elgamal public-key cryptosystem, i.e., the need to compute discrete logarithms. Moreover, Elgamal encryption (and in particular, Elgamal encryption based on elliptic curves) is implemented by several commercially available Hardware Security Modules, which cannot be said about the known additively homomorphic cryptosystems.

Proxy Oblivious Transfer. Due to the lack of space, this discussion can be found in App. E.

Voter Education. For the added two channels and the new protocol to be useful in practice, the voters must be educated. They must be told that they should never enter the check codes to their computer, and that they should actively react to the messages (or their absence) on the postchannel. This will add extra costs, but the costs will be hopefully amortized over several elections. Moreover, the Internet and computers are ubiquitous in the developed world already now, with average people performing much more complex operations in a daily basis. Thus, after some years we can reasonably expect the voters to know how to guarantee their own vote privacy (and security in general case).

7 Implementation Data

We implemented a (slightly optimized) sandbox version of the new e-voting protocol. We tested it thoroughly, and measured its efficiency by using a personal computer that runs `Linux 2.6.18-6-686`, has a Pentium 4 CPU that runs at 2.80GHz and has 512 KB of cache, and has 2 GB of main memory. The code was compiled by using `gcc 4.1.2` with the option `-O2`. For generating the Elgamal parameters, we used the `openssl 0.9.8c` library, while other number-theoretic operations were implemented by using Victor Shoup's `NTL 5.5.1` library.

We measured the time that was spent during the election setup, and during the election itself. In the tallying, one can use any of the standard mixnet-based solutions, and thus we did not measure this part. For the time measurement, we used the standard `Unix` command `time`, and took the average over 100 different runs. The results are summarized in the next two tables, for $v = \{100, 1000, 10\,000\}$ voters, and $c \in \{8, 32, 80\}$ candidates. In all cases, $|p| = 1024$, $|q| = 160$, and $k = 80$. We used SHA2-256 as the hash function. The first table contains the one-time election setup cost (codecard generation and Elgamal system parameter value generation) which depends linearly on the product $v \cdot c$. More precisely, it is dominated by $v \cdot c$ random number generations and exponentiations modulo p.

	$v = 100$			$v = 1\,000$			$v = 10\,000$		
	$c = 8$	$c = 32$	$c = 80$	$c = 8$	$c = 32$	$c = 80$	$c = 8$	$c = 32$	$c = 80$
Setup	3.875s	15.40s	38.48s	38.58s	2m 34s	6m 25s	6m 25s	25m 38s	1h 4m 20s

The next table summarizes the online computation time of voter's PC, vote collector and messenger, both with and without the zero-knowledge proofs. The costs are given per one vote, and do not significantly depend on the number of the voters. The total row is the sum of the time spent by voter's PC, vote collector and messenger, and gives a (loose) lower bound on time that must elapse before the voter receives back a message on the postchannel.

	With ZK			Without ZK		
	$c = 8$	$c = 32$	$c = 80$	$c = 8$	$c = 32$	$c = 80$
Voter's PC	0.21s	0.30s	0.34s	0.02s	0.02s	0.02s
Vote collector	0.40s	1.07s	2.27s	0.20s	0.78s	1.95s
Messenger	0.02s	0.08s	0.22s	0.02s	0.08s	0.20s
Total	0.63s	1.45s	2.83s	0.24s	0.88s	2.17s

We also note that a single exponentiation on this machine took about 0.0048s. Moreover, the timings of the parties include also the precomputation time. In particular, vote collector's online computation in the POT protocol requires twice less time than her total computation in POT.

As seen from these tables, the computation time of the voter's PC and messenger is quite insignificant even in the case of 80 candidates. On the other hand, if there are 80 candidates, then the vote collector spends (on average) 2.27 seconds per vote and cannot process more than about 1 500 votes per hour even under ideal conditions. Assuming that the vote collector precomputes in the POT protocol, the throughput increases to 3 000 votes per hour. In the case of real e-voting, the cryptographic protocol is obviously only a part of what the vote collector is busy with, and thus the maximum throughput is probably around 2 000 votes per hour. In smaller countries, this is sufficient under normal conditions, but not during the first or the last few hours of the e-voting.[7] However, this can be alleviated by using either fast (and multicore) processors, parallel processing by many vote collectors, or even by using hardware acceleration. (In particular, we are currently considering a Hardware Security Module implementation based on elliptic curves.) The use of such (more expensive) alternatives is reasonable, given the importance of elections in a democratic society. Moreover, in the case of most elections, the number of candidates is not larger than 10.

Acknowledgments. We would like to thank Kristian Gjøsteen for useful comments. The second author was supported by Estonian Science Foundation, grant #8058, and European Union through the European Regional Development Fund.

[7] As an example, in the most recent Estonian e-voting in October 2009, more than 104 thousand e-votes were given in total, peaking with 4500 e-votes during the last hour. See http://vvv.vvk.ee/public/pics/EP09kokkuakt.jpg for the distribution of voters per hour during the voting period.

References

[Adi08] Ben Adida. Web-based Open-Audit Voting. In Paul C. Oorschot, editor, *USENIX 2008*, pages 335–348, San Jose, CA, USA, July 28–August 1, 2008. USENIX Association.

[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. volume 5838 of *LNCS*, pages 207–222, Oslo, Norway, 2009. Springer-Verlag.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag.

[Bou00] Fabrice Boudot. Efficient Proofs That a Committed Number Lies in an Interval. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444, Bruges, Belgium, 14–18 May 2000. Springer-Verlag.

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Victoria Ashby, editor, *ACM CCS 1993*, pages 62–73, Fairfax, Virginia, 3–5 November 1993. ACM Press.

[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-Verlag.

[Bul09] Christian Bull. E-Voting at Norwegian Municipal Elections. volume 5838 of *LNCS*, Oslo, Norway, 2009. Springer-Verlag. Invited talk.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient Protocols for Set Membership and Range Proofs. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 234–252, Melbourne, Australia, December 7–11, 2008. Springer-Verlag.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo G. Desmedt, editor, *CRYPTO 1994*, volume 839 of *LNCS*, pages 174–187, Santa Barbara, USA, August 21–25 1994. Springer-Verlag.

[CFGP06] Olivier Chevassut, Pierre-Alain Fouque, Pierrick Gaudry, and David Pointcheval. The Twist-AUgmented Technique for Key Exchange. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 410–426, New York, NY, USA, April 24–26, 2006. Springer-Verlag.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-Authority Secret Ballot Elections with Linear Work. In Ueli Maurer, editor, *EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 72–83, Saragossa, Spain, May 12–16, 1996. Springer-Verlag.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. In *STOC 1998*, pages 209–218, New York, May 23–26, 1998.

[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-Verlag.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computational Private Information Retrieval with Polylogarithmic Communication. In Jacques Stern, editor, *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 402–414, Prague, Czech Republic, May 2–6, 1999. Springer-Verlag.

[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.

[FIP09] FIPS. Digital Signature Standard. Technical report, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, USA, June 2009. FIPS 186-3.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, California, USA, 11–15 August 1986. Springer-Verlag, 1987.

[FS01] Jun Furukawa and Kazue Sako. An Efficient Scheme for Proving a Shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, USA, 19–23 August 2001. Springer-Verlag.

[Fur04] Jun Furukawa. Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 319–332, Singapore, March 1–4, 2004. Springer-Verlag.

[Gjø10] Kristian Gjøsteen. Private communication. February 2010.

[GL07] 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-Verlag.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In Luis Caires, Guiseppe F. Italiano, Luis Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 803–815, Lisboa, Portugal, 2005. Springer-Verlag.

[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, January6–8, 2003. Springer-Verlag.

[Gro06] Jens Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, Shanghai, China, December 3–7, 2006. Springer-Verlag.

[HLV10] Sven Heiberg, Helger Lipmaa, and Filip Van Laenen. On E-Vote Integrity in the Case of Malicious Voter Computers. Technical Report 2010/195, International Association for Cryptologic Research, April 8, 2010. Available at http://eprint.iacr.org/2010/195.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In Vijay Atluri, Sabrina De Capitani di Vimercati, and Roger Dingledine, editors, *WPES 2005*, pages 61–70, Alexandria, VA, USA, November 7 2005. ACM.

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 87–101, Southhampton Beach, Bermuda, March 11–14, 2002. Springer-Verlag.

[Lip03] Helger Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In Chi Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 398–415, Taipei, Taiwan, November 30–December 4, 2003. Springer-Verlag.

[Lip05a] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In Jianying Zhou and Javier Lopez, editors, *ISC 2005*, volume 3650 of *LNCS*, pages 314–328, Singapore, September 20–23, 2005. Springer-Verlag.

[Lip05b] Helger Lipmaa. *Secure Electronic Voting Protocols*. John Wiley & Sons, Inc., 2005.

[Lip09] Helger Lipmaa. First CPIR Protocol with Data-Dependent Computation. In Donghoon Lee and Seokhie Hong, editors, *ICISC 2009*, volume 5984 of *LNCS*, pages 193–210, Seoul, Korea, December 2–4, 2009. Springer-Verlag.

[LL07] Sven Laur and Helger Lipmaa. A New Protocol for Conditional Disclosure of Secrets And Its Applications. In Jonathan Katz and Moti Yung, editors, *ACNS 2007*, volume 4521 of *LNCS*, pages 207–225, Zhuhai, China, June 5–8, 2007. Springer-Verlag.

[Nef01] C. Andrew Neff. A Verifiable Secret Shuffle and Its Application to E-Voting. In *8th ACM Conference on Computer and Communications Security*, pages 116–125, Philadelphia, Pennsylvania, USA, November 6–8 2001. ACM Press.

[NP99] Moni Naor and Benny Pinkas. Oblivious Transfer And Polynomial Evaluation. In *STOC 1999*, pages 245–254, Atlanta, Georgia, USA, May 1–4, 1999. ACM Press.

[NP01] Moni Naor and Benny Pinkas. Efficient Oblivious Transfer Protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, Washington, DC, USA, January 7–9, 2001. ACM Press.

[NPS99] Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy Preserving Auctions and Mechanism Design. In *ACM EC 1999*, Denver, Colorado, November 1999.

[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-Verlag.

[RS06] Peter Y. A. Ryan and Steve A. Schneider. Prêt à Voter with Re-encryption Mixes. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS 2006*, volume 4189 of *LNCS*, pages 313–326, Hamburg, Germany, September 18–20, 2006. Springer-Verlag.

[Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

A Security Definitions of Σ-Protocols

A Σ-protocol is *correct* if for any $(x, y) \in R$, an honest verifier accepts all runs with an honest prover. A Σ-protocol has the property of *special soundness*, if one can construct an efficient simulator that, given any two accepting runs $(x; \mathfrak{i}, c_1, \mathfrak{r}_1)$ and $(x; \mathfrak{i}, c_2, \mathfrak{r}_2)$ with $c_1 \neq c_2$, outputs a y such that $(x, y) \in R$. A Σ-protocol has the property of SHVZK if there exists an efficient simulator that, given as input an arbitrary $x \in L$ (without corresponding y), can construct accepting runs $(x; \mathfrak{i}^*, \mathfrak{c}^*, \mathfrak{r}^*)$ such that (a) the simulator starts by choosing uniformly random $\mathfrak{c}^*$ and $\mathfrak{r}^*$, and only then computes $\mathfrak{i}^*$, and (b) the distribution $(x; \mathfrak{i}^*, \mathfrak{c}^*, \mathfrak{r}^*)$ is computationally indistinguishable from the distribution $(x; \mathfrak{i}, \mathfrak{c}, \mathfrak{r})$ of runs between an honest prover and an honest verifier.

B Range Proof from [LAN02]

We now briefly recall the range proof in exponents from [LAN02]. (Note that the original description in [LAN02] was given for a different cryptosystem, but it clearly works also in the case of Elgamal.) For any $\mathsf{CC} > 0$, denote $\lambda := \lfloor \log_2 \mathsf{CC} \rfloor$ and $\mathsf{CC}_j := \lfloor (\mathsf{CC} + 2^j)/2^{j+1} \rfloor$ for $j \in \{0, \dots, \lambda\}$. As stated in [LAN02], $\mu \in [0, \mathsf{CC}]$ if and only if there exist $\mu_j \in \{0, 1\}$, such that $\mu = \sum_{j=0}^{\lambda} \mu_j \mathsf{CC}_j$. Here, the values μ_j can be computed by using a simple greedy algorithm. As an example, when $\mathsf{CC} = 12$, then $\mathsf{CC}_0 = 6$, $\mathsf{CC}_1 = 3$, $\mathsf{CC}_2 = 2$, $\mathsf{CC}_3 = 1$, and thus $\mu \in [0, 12]$ iff for some $\mu_j \in \{0, 1\}$, $\mu = 6\mu_0 + 3\mu_1 + 2\mu_2 + \mu_3$. E.g., $2 = 0 + 0 + 2 + 0$, $9 = 6 + 3 + 0 + 0$, and $12 = 6 + 3 + 2 + 1$. Thus, to prove that $\mu \in [0, \mathsf{CC}]$, one can separately encrypt (by using a multiplicatively homomorphic cryptosystem like Elgamal) all values g^{μ_j}, $e_j \leftarrow \mathsf{Enc}_{\mathsf{pk}}(g^{\mu_j})$, and then prove that each value e_j encrypts either 0 or 1 (by using standard techniques [CDS94]). Finally, one obtains an encryption of g^μ by computing $e \leftarrow \prod e_j^{\mathsf{CC}_j} = \mathsf{Enc}_{\mathsf{pk}}(\prod g^{\mu_j \mathsf{CC}_j}) = \mathsf{Enc}_{\mathsf{pk}}(g^{\sum \mu_j \mathsf{CC}_j}) = \mathsf{Enc}_{\mathsf{pk}}(g^\mu)$.

C Security of the New POT Protocol

For a short proof of correctness and sender's privacy, note that $e_i = \mathsf{Enc}_{\mathsf{pk}}(g^{r_i(i-x)}; r_i(1-\rho)) \cdot \mathsf{Enc}_{\mathsf{pk}}(g^{f_i}; r'_i) = \mathsf{Enc}_{\mathsf{pk}}(g^{r_i(i-x)+f_i}; r_i(1-\rho)+r'_i)$. Denote $r''_i \leftarrow r_i(1-\rho) + r'_i$, note that r''_i is uniformly random in $\mathbb{Z}_q$. Clearly, if $i = x$ then $e_i = \mathsf{Enc}_{\mathsf{pk}}(g^{f_i}; r''_i)$, and thus proxy obtains the correct value, with $y = g^{f_x}$. However, if $i \neq x$, then $r_i(i-x) + f_i$ is a completely random element of $\mathbb{Z}_q$ and thus the proxy obtains a random encryption of a random group element. For a short proof of chooser's privacy, note that the sender only sees an Elgamal encryption of client's index. On the other hand, even if the proxy gets back the value f_x, and has an unordered copy of f, he does not know the value x. Moreover, sender's computation is dominated by $2n$ encryptions (n of which can be precomputed) and n exponentiations. Proxy's computation is dominated by n decryption operations and n discrete logarithm computations.

D Proof of Theorem 1

Proof. **Completeness:** assume that both prover and verifier are honest, and in particular that $e' = \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu}; \rho_2)$ and $\mu \in [0, \mathsf{CC}]$. Then $\mu = \sum_{j=0}^{\lambda} \mu_j \mathsf{CC}_j$ for some $\mu_j \in \{0,1\}$ [LAN02]. We check that verification (on Step 4(b)ii) succeeds for every j and for both $\mu_j = 0$ and $\mu_j = 1$, and then that the verifications on Step 4c succeed. First, fix some j. If $\mu_j = 0$, then

$$\begin{aligned}
\mathfrak{i}_{0,j} \cdot e_j^{\mathfrak{c}_{0,j}} &= \mathsf{Enc}_{\mathsf{pkt}}(1; \rho'_j) \cdot \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_j}; \rho_j)^{\mathfrak{c}_{0,j}} = \mathsf{Enc}_{\mathsf{pkt}}(1; \rho'_j + \mathfrak{c}_{0,j} \cdot \rho_j) \\
&= \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j}), \\
\mathfrak{i}_{1,j} \cdot (e_j / \mathsf{Enc}_{\mathsf{pkt}}(g; 0))^{\mathfrak{c}_{1,j}} &= \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j}) \cdot (\mathsf{Enc}_{\mathsf{pkt}}(g; 0)/e_j)^{\mathfrak{c}_{1,j}} \cdot (e_j / \mathsf{Enc}_{\mathsf{pkt}}(g; 0))^{\mathfrak{c}_{1,j}} \\
&= \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j}) \enspace ,
\end{aligned}$$

as needed. If $\mu_j = 1$, then

$$\begin{aligned}
\mathfrak{i}_{0,j} \cdot e_j^{\mathfrak{c}_{0,j}} &= \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j}) / e_j^{\mathfrak{c}_{0,j}} \cdot e_j^{\mathfrak{c}_{0,j}} = \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j}), \\
\mathfrak{i}_{1,j} \cdot (e_j / \mathsf{Enc}_{\mathsf{pkt}}(g; 0))^{\mathfrak{c}_{1,j}} &= \mathsf{Enc}_{\mathsf{pkt}}(1; \rho'_j) \cdot \mathsf{Enc}_{\mathsf{pkt}}(1; \rho_j)^{\mathfrak{c}_{1,j}} \\
&= \mathsf{Enc}_{\mathsf{pkt}}(1; \rho'_j + \mathfrak{c}_{1,j} \cdot \rho_j) = \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j})
\end{aligned}$$

as needed. Second,

$$\begin{aligned}
\mathfrak{i}_{2,1} \cdot e^{\mathfrak{c}} &= \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_{\mathsf{and}}}; \rho_{\mathsf{and},1}) \cdot \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho')^{\mathfrak{c}} = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_{\mathsf{and}} + \mathfrak{c} \cdot \mu}; \rho_{\mathsf{and},1} + \mathfrak{c} \cdot \rho') \\
&= \mathsf{Enc}_{\mathsf{pkt}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,1}) \enspace , \\
\mathfrak{i}_{2,2} \cdot (e')^{\mathfrak{c}} &= \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu_{\mathsf{and}}}; \rho_{\mathsf{and},2}) \cdot \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu}; \rho')^{\mathfrak{c}} = \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu_{\mathsf{and}} + \mathfrak{c} \cdot \mu}; \rho_{\mathsf{and},2} + \mathfrak{c} \cdot \rho') \\
&= \mathsf{Enc}_{\mathsf{pkm}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,2}) \enspace .
\end{aligned}$$

Special Soundness. Assume that $\mathfrak{i}$ is defined as in Prot. 1, $c \neq \hat{c}$, and $\mathfrak{r}$ and $\widehat{\mathfrak{r}}$ are two different third round messages that an honest verifier accepts. Then, according to the verification equations on Step 4c,

$$\mathsf{Enc}_{\mathsf{pkt}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,1}) = \mathfrak{i}_{2,1} \cdot e^{\mathfrak{c}} \enspace , \quad \mathsf{Enc}_{\mathsf{pkt}}(g^{\widehat{\mathfrak{r}_3}}; \widehat{\mathfrak{r}_{4,1}}) = \mathfrak{i}_{2,1} \cdot e^{\widehat{\mathfrak{c}}} \enspace .$$

Dividing the first equality with the second, we get $e^{\mathfrak{c}-\widehat{\mathfrak{c}}} = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mathfrak{r}_3-\widehat{\mathfrak{r}_3}}; \mathfrak{r}_{4,1} - \widehat{\mathfrak{r}_{4,1}})$, or equivalently, $e = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho)$ for $\mu \leftarrow (\mathfrak{r}_3 - \widehat{\mathfrak{r}_3})/(\mathfrak{c} - \widehat{\mathfrak{c}})$, and $\rho \leftarrow (\mathfrak{r}_{4,1} - \widehat{\mathfrak{r}_{4,1}})/(\mathfrak{c} - \widehat{\mathfrak{c}})$. Analogously, $(e')^{\mathfrak{c}-\widehat{\mathfrak{c}}} = \mathsf{Enc}_{\mathsf{pkm}}(g^{\mathfrak{r}_3-\widehat{\mathfrak{r}_3}}; \mathfrak{r}_{4,2} - \widehat{\mathfrak{r}_{4,2}})$, or equivalently, $e' = \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu}; \rho')$ for $\rho' \leftarrow (\mathfrak{r}_{4,2} - \widehat{\mathfrak{r}_{4,2}})/(\mathfrak{c} - \widehat{\mathfrak{c}})$. Thus, from here the simulator can extract μ, ρ, ρ' such that $e = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu}; \rho)$ and $e' = \mathsf{Enc}_{\mathsf{pkm}}(g^{\mu}; \rho')$.

Now, let us consider the verifications on Step 4(b)ii. Fix some j. Note that since $\mathfrak{c} \neq \widehat{\mathfrak{c}}$, and we fix the first step, then $\mathfrak{c}_{1-\mu_j,j} \neq \widehat{\mathfrak{c}_{1-\mu_j,j}}$ while $\mathfrak{c}_{\mu_j,j} \neq \widehat{\mathfrak{c}_{\mu_j,j}}$.

Thus, if $\mathfrak{c}_{0j} \neq \widehat{\mathfrak{c}_{0j}}$, then

$$\mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j}) = \mathfrak{i}_{0,j} \cdot e_j^{\mathfrak{c}_{0,j}} \quad , \quad \mathsf{Enc}_{\mathsf{pkt}}(1; \hat{z}_{0,j}) = \mathfrak{i}_{0,j} \cdot e_j^{\widehat{\mathfrak{c}_{0,j}}} \quad .$$

Thus, $e_j^{\mathfrak{c}_{0,j}-\widehat{\mathfrak{c}_{0,j}}} = \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j} - \hat{z}_{0,j})$, or equivalently, $e_j = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mu_j}; \rho_j)$ for $\mu_j = 0$ and $\rho_j = (\mathfrak{r}_{0,j} - \hat{z}_{0,j})/(\mathfrak{r}_{0,j} - \hat{z}_{0,j})$.

On the other hand, if $\mathfrak{c}_{1j} \neq \widehat{\mathfrak{c}_{1j}}$, then

$$\mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j}) = \mathfrak{i}_{1,j} \cdot (e_j/\mathsf{Enc}_{\mathsf{pkt}}(g; 0))^{\mathfrak{c}_{1,j}} \quad , \quad \mathsf{Enc}_{\mathsf{pkt}}(1; \hat{z}_{1,j}) = \mathfrak{i}_{1,j} \cdot (e_j/\mathsf{Enc}_{\mathsf{pkt}}(g; 0))^{\widehat{\mathfrak{c}_{1,j}}} \quad .$$

Thus, $e_j^{\mathfrak{c}_{1j}-\widehat{\mathfrak{c}_{1,j}}}\mathsf{Enc}_{\mathsf{pkt}}(g^{\widehat{\mathfrak{c}_{1,j}}-\mathfrak{c}_{1j}}; 0) = \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j} - \hat{z}_{1,j})$, or $e_j^{\mathfrak{c}_{1j}-\widehat{\mathfrak{c}_{1,j}}} = \mathsf{Enc}_{\mathsf{pkt}}(g^{\mathfrak{c}_{1j}-\widehat{\mathfrak{c}_{1,j}}}; \mathfrak{r}_{1,j} - \hat{z}_{1,j})$, or equivalently, $e_j = \mathsf{Enc}_{\mathsf{pkt}}(g; \rho_j)$ for $\mu_j = 1$ and $\rho_j = (\mathfrak{r}_{1,j} - \hat{z}_{1,j})/(\mathfrak{c}_{1j} - \widehat{\mathfrak{c}_{1,j}})$. Thus, the simulator can extract values μ_k, ρ_j, such that $e_j = \mathsf{Enc}_{\mathsf{pkt}}(\mu_j; \rho_j)$.

The rest is now easy. The simulator knows that both e and e' encrypt $\mu = \sum \mu_j \cdot \mathsf{CC}_j$, and since $\mu_j \in \{0, 1\}$ then $\mu \in [0, \mathsf{CC}]$.

SHVZK. A simulator can work as follows. Assume that the simulator gets

$$(G, q, g, \mathsf{pkt}, \mathsf{pkm}, e_0, \dots, e_\lambda, e')$$

as input. First, generate random $\mathfrak{c} \leftarrow \mathbb{Z}_{2^k}$, and compute random $c_{i,j} \leftarrow \mathbb{Z}_{2^k}$ such that $c = c_{0,j} + c_{1,j} \mod 2^k$. Generate random $\mathfrak{r}_{0,j}, \mathfrak{r}_{2,j}, \mathfrak{r}_3, \mathfrak{r}_{4,1}, \mathfrak{r}_{4,2} \leftarrow \mathbb{Z}_q$. Now define $\mathfrak{i}_{0,j}, \mathfrak{i}_{1,j}, \mathfrak{i}_{2,1}, \mathfrak{i}_{2,2}$ such that the verifications on Steps 4(b)ii and 4c would succeed, i.e., $\mathfrak{i}_{0,j} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{0,j})/e_j^{\mathfrak{c}_{0,j}}$, $\mathfrak{i}_{1,j} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(1; \mathfrak{r}_{1,j}) \cdot (\mathsf{Enc}_{\mathsf{pkt}}(g; 0)/e_j)^{\mathfrak{c}_{1j}}$, $\mathfrak{i}_{2,1} \leftarrow \mathsf{Enc}_{\mathsf{pkt}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,1})/e^{\mathfrak{c}}$ and $\mathfrak{i}_{2,2} \leftarrow \mathsf{Enc}_{\mathsf{pkm}}(g^{\mathfrak{r}_3}; \mathfrak{r}_{4,2})/(e')^{\mathfrak{c}}$. Clearly the resulting view $(\mathfrak{i}, \mathfrak{c}, \mathfrak{r})$, where $\mathfrak{i}$ and $\mathfrak{r}$ are defined as in Prot. 1, is both accepting and has exactly the same distribution as accepting views between honest prover and verifier.

This completes the proof. □

E Discussion: Proxy Oblivious Transfer

In the case of oblivious transfer protocols, it is usually easy to start with protocols with linear communication, but the usual goal is to decrease the communication. Since in our case, the databases are small (say ≤ 64 elements), linear communication of the used POT protocol does not matter, while its computation complexity is the bottleneck. Another bottleneck is the need for the messenger to perform $\approx n \cdot \log_2 n$ comparisons, but as mentioned before, this is strongly dominated by the cost of doing $\Theta(n)$ public-key operations. Thus, even in this case, it is more efficient to use AIR based on Elgamal than the version of AIR based on the Paillier cryptosystem [LL07]: while in the latter case, the number of comparisons would drop, the cost of public-key operations will increase.

In most of the existing OT protocols, sender's online computational complexity is dominated by $\Theta(n)$ public-key operations. The only exception is the OT protocol proposed in [Lip09] which reduces this computation to $\Theta(n/\log n)$, but the cost of public-key operations in that protocol is increased significantly, which is important when n is so small. Moreover, the underlying assumption of [Lip09], the Decisional Composite Residuosity assumption (first proposed in [Pai99]) is much less known than the Decisional Diffie-Hellman assumption and thus arguably not yet ready to be used in an application of such an importance.

In [NP01, Sect. 3.1], Naor and Pinkas proposed an OT protocol where the sender's online computational complexity is dominated by 1 exponentiation and n multiplications/hash function computations. However, their protocol is secure only in the random oracle model. While we use random oracles in our protocol, it is a good design principle to limit their usage as much as possible. (Removing random oracles from any NIZK protocols requires currently the use of pairing-based NIZK protocols [Gro06], which are considerably slower than the presented protocols. Moreover, there does not seem to be yet enough confidence in pairing assumptions.) Moreover, it is not immediately clear how to modify their OT protocol to a strong POT protocol since there the proxy needs to use the correct public key to decrypt, and thus will obtain information about the client's input.

Another interesting case is the Gentry-Ramzan oblivious transfer protocol [GR05] that has server's online computational complexity dominated by $\Theta(n)$ multiplications. While the Gentry-Ramzan oblivious transfer protocol is very efficient by itself, it is not clear how to base a POT on it. Briefly, in the Gentry-Ramzan protocol, the client generates a modulus N such that for some $x \in \{0, \dots, n-1\}$, a previously fixed prime power p_x divides $\varphi(N)$. While decoding the answer, the client who knows the factorization of N can efficiently obtain the value of f_x. In the case of the POT, the proxy (who is doing the decoding) does not and should not know the value of x. Construction of an efficient strong POT protocol based on the Gentry-Ramzan oblivious transfer protocol is thus an interesting open question. Note also that the underlying security assumption behind the Gentry-Ramzan protocol, the Phi-Hiding assumption [CMS99], is even less studied than the Decisional Composite Residuosity assumption.