Efficient Receipt-Free Voting Based on Homomorphic Encryption

Martin Hirt[1]* and Kazue Sako[2]

[1] ETH Zürich, Switzerland
hirt@inf.ethz.ch
[2] NEC Corporation, Japan
sako@ccm.cl.nec.co.jp

Abstract. Voting schemes that provide *receipt-freeness* prevent voters from proving their cast vote, and hence thwart vote-buying and coercion. We analyze the security of the multi-authority voting protocol of Benaloh and Tuinstra and demonstrate that this protocol is *not* receipt-free, opposed to what was claimed in the paper and was believed before. Furthermore, we propose the first practicable receipt-free voting scheme. Its only physical assumption is the existence of secret *one-way* communication channels from the authorities to the voters, and due to the public verifiability of the tally, voters only join a single stage of the protocol, realizing the "vote-and-go" concept. The protocol combines the advantages of the receipt-free protocol of Sako and Kilian and of the very efficient protocol of Cramer, Gennaro, and Schoenmakers, with help of designated-verifier proofs of Jakobsson, Sako, and Impagliazzo.
Compared to the receipt-free protocol of Sako and Kilian for security parameter ℓ (the number of repetitions in the non-interactive cut-and-choose proofs), the protocol described in this paper realizes an improvement of the total bit complexity by a factor ℓ.

1 Introduction

1.1 Background

Secret-ballot voting protocols are one of the most significant application of cryptographic protocols. The most efficient secret-ballot voting protocols can be categorized by their approaches into three types: Schemes using mix-nets [Cha81,PIK93,SK95,OKST97,Jak98,Abe99], schemes using homomorphic encryption [CF85,CY86,Ben87,BT94,SK94,CFSY96,CGS97], and schemes using blind signatures [FOO92,Sak94,Oka97]. The suitability of each of these three types varies with the conditions under which it is to be applied.

In a model with vote-buyers (or coercers), a voting scheme must ensure not only that a voter *can* keep his vote private, but also that he *must* keep it private. In other words, the voter should not be able to prove to a third party that he has cast a particular vote. He must neither obtain nor be able to construct a receipt proving the content of his vote. This property is referred to as *receipt-freeness*.

* Supported by the Swiss National Science Foundation, project no. SPP 5003-045293.

The concept of receipt-freeness was first introduced by Benaloh and Tuinstra [BT94]. Based on the assumption of a voting booth that physically guarantees secret communication between the authorities and each voter, they proposed two voting protocols using homomorphic encryptions. The first one is a single-authority voting protocol which, while being receipt-free, fails to maintain vote secrecy. Then they extend this protocol to the second protocol, which is a multi-authority scheme achieving vote secrecy. However, we show that this scheme is *not* receipt-free, as opposed to what is claimed in the paper.

Another receipt-free voting protocol based on a mix-net channel was proposed by Sako and Kilian [SK95]. In contrast to [BT94], it assumes only *one-way* secret communication from the authorities to the voters. The heavy processing load required for tallying in mix-net schemes, however, is a significant disadvantage of this protocol.

Finally, a receipt-free voting scheme using blind signatures was given by Okamoto [Oka97]. Here, the assumption was of *anonymous* one-way secret communication from each voter to each authority. Achieving communication that is both secret *and* anonymous would, however, be extremely difficult. Also, this scheme requires each voter to be active in *three rounds* (authorization stage, voting stage, and claiming stage), which is not acceptable in practice.

Another stream of research which relates to receipt-freeness is incoercible multi-party computation. Without any physical assumption, deniable encryption [CDNO97] allows an entity to lie *later* how the ciphertext decrypts, and this technique is used to achieve incoercible multi-party computation [CG96]. However, the concept of incoercibility is weaker than receipt-freeness. It would allow a voter to lie about his vote, but it cannot help against a voter who *wants* to make his encryption undeniable, and hence cannot prevent vote-buying.

1.2 Contributions

In this paper, we first demonstrate that the multi-authority protocol of Benaloh and Tuinstra [BT94] is not receipt-free, opposed to what was claimed in the paper and was believed before. We then present a novel generic construction for introducing receipt-freeness into a voting scheme based on homomorphic encryption by assuming some additional properties of the encryption function. This construction also includes a solution for the case that an authority does not send correct information through the untappable channel.[1] Moreover, as opposed to previous receipt-free protocols, we disable vote-buying even in cases where some authorities are colluding with the voter-buyer. The security of these protocols is specified with respect to a threshold t, where the correctness of the tally is guaranteed as long as at least t authorities remain honest during the whole protocol execution, and privacy is guaranteed as long as no t or more curious authorities pool their information.

[1] Due to the untappability of the channel the voter cannot prove that the received information is incorrect. In previous protocols, this problem was ignored, and the situation of a voter complaining about an authority would have lead to a deadlock.

Our construction gives a receipt-free voting protocol which runs as follows: For each voter the authorities jointly generate a randomly ordered list with an encryption of each valid vote, along the lines of [SK95]. The ordering of the list is secretly conveyed and proven to the voter by deploying the technique of designated-verifier proofs [JSI96], and the voter points to the encryption of his choice. Tallying of votes is performed using the homomorphic property of the encryption function.

By applying this generic construction to the voting protocol of Cramer, Gennaro, and Schoenmakers [CGS97], we obtain an efficient receipt-free voting protocol based on homomorphic encryption.

The efficiency achieved by our protocol compared to the protocol of Sako and Kilian [SK95] with security parameter ℓ in the case of 1-out-of-2 voting is as follows: The communication through the untappable channels and through the public channels are reduced by a factor of $\ell/4$ and $3\ell/2$, respectively. Altogether, this results in a speedup by a factor of ℓ. As an example, for $M = 1,000,000$ voters, $N = 10$ authorities, a $K = 1024$ bit group, and security parameter $\ell = 80$, the protocol of [SK95] communicates 102 GB (gigabyte) over the untappable channels and 924 GB over the public channels, whereas the protocol of this paper communicates 5 GB over untappable channels and 8 GB over public channels.

1.3 Organization of the Paper

The paper is organized as follows: In Sect. 2, we analyze the receipt-freeness of the protocol with multiple voting authorities of Benaloh and Tuinstra [BT94] and demonstrate its non receipt-freeness by showing how a voter can construct a receipt for the vote he casts. In Sect. 3, we present a generic receipt-free protocol for 1-out-of-L voting based on homomorphic encryptions, and in Sect. 4 we apply these techniques to the protocol of Cramer, Gennaro, and Schoenmakers [CGS97] and obtain an efficient receipt-free voting scheme. Finally, in Sect. 5, we even improve the efficiency of our protocol by tailoring it to 1-out-of-2 voting.

2 Analysis of the Benaloh-Tuinstra Protocol

The notion of receipt-freeness was first introduced by Benaloh and Tuinstra in [BT94]. They present two protocols that are claimed to be receipt-free. In the single-authority protocol, the authority learns how each vote was cast. This is of course far from satisfactory. In this section, we analyze the receipt-freeness of their protocol with multiple voting authorities and show how a voter can construct a receipt for the vote he casts.

2.1 Key Ideas of the Protocol

The basic idea of the multiple-authority protocol [BT94] is to have every voter secret-share his vote among the authorities (using Shamir's secret-sharing scheme [Sha79]), who then add up the shares and interpolate the tally. This idea works due to the linearity of the secret-sharing scheme. There are two major tasks to solve: First, the voter must send one share to each authority in a receipt-free manner, and second, the voter must prove that the secret (the vote) is valid.

We concentrate on the second task: In order to secret-share the vote, the voter selects a random polynomial P of appropriate degree, such that $P(0) \in \{0,1\}$ is his vote. The share for the j-th authority is hence $P(j)$. Clearly, it is inherently important that the vote is valid, i.e. $P(0) \in \{0,1\}$, since otherwise the tally will be incorrect. Hence, the voter must provide a proof of validity for the cast vote.

For the sake of the proof of validity, the voter wishing to cast a vote v_0 submits a bunch of $n+1$ vote pairs, where n is a security parameter. That is, the voter submits the votes $(v_0, v_0'), \ldots, (v_n, v_n')$, and each pair (v_i, v_i') of votes must contain one 0-vote and one 1-vote in random order. For each pair (v_i, v_i') but the first, a coin is tossed and the voter is either asked to open the pair and show that indeed there is a 0-vote and a 1-vote, or he is asked to prove that either $v_i = v_0$ and $v_i' = v_0'$ is satisfied, or that $v_i = v_0'$ and $v_i' = v_0$ is satisfied. If the voter passes these tests, then with probability at least $1 - 2^{-n}$, v_0 is valid and is accepted as the voters vote.

2.2 How to Construct a Receipt

This cut-and-choose proof of validity offers an easy ability to prove a particular vote: In advance, the voter commits to the ordering of each pair of votes (i.e. he commits to the bit string $v_0, \ldots, v_n$). In each round of the cut-and-choose proof, one can verify whether the revealed data is consistent with this commitment. If no inconsistencies are detected while proving the validity of the vote, then with probability at least $1-2^{-n}$ the voter has chosen the ordering as committed, and also v_0 is as announced.

In order to obtain a receipt, the voter could select an arbitrary string s, and set the string $(v_0, \ldots, v_n)$ as the bitwise output of a known cryptographic hash function (e.g. MD5 or SHA) for that string s. Then, s is a receipt of the vote v_0.

3 Generic Receipt-Free Protocol

In this section, we present a novel and general construction for converting a voting protocol based on homomorphic encryption (with additional properties of the encryption function) into a receipt-free voting protocol. Receipt-freeness means that the voter cannot prove to a third party that he has cast a particular vote. The reason why most classical voting schemes are not receipt-free is simple: Each encrypted vote is published, and the voter himself can prove the content of his vote by revealing the randomness he used for encrypting. When a scheme requires the voter to choose randomness, then often the voter can exploit this to construct a receipt, for example by using the hash of a predetermined value (cf. Sect. 2). Therefore, in the protocol of this paper, the authorities jointly generate an encryption of each valid vote in random order, and each voter only *points* to the encrypted vote of his choice. The ordering of the encrypted valid votes is proven to the voter in designated verifier manner through the untappable channel, so that the voter cannot transfer this proof to a vote-buyer.

3.1 Model and Definitions

Entities. We consider a model with N authorities $A_1, \ldots, A_N$ and M voters. A threshold t denotes the lower bound on the number of authorities that is guaranteed to remain honest during the complete protocol execution.

Communication. Communication takes place by means of a bulletin board which is publicly readable, and which every participant can write to (into his own section), but nobody can delete from. The bulletin board can be considered as public channels with memory. Furthermore, we assume the existence of untappable one-way channels from the authorities to the voters. The security of these channels must be physical, in such a way that even the voter cannot demonstrate what was sent over the channel (of course, the voter can record all received data, but he must not be able to *prove* to a third party that he received a particular string). Even a coercer who is physically present at the voter's place must not be able to eavesdrop the untappable channels. Note that some physical assumption seems to be inevitable for achieving receipt-freeness.[2] Indeed, untappable one-way channels from the authorities to the voters (as assumed in this paper and in [SK95]) are the weakest physical assumption for which receipt-free voting protocols are known to exist.

Key Infrastructure. To each voter, a secret key and a public key is associated, where it must be ensured that each voter knows the secret key according to his public key. This assumption is very natural and typically used for voter identification in any voting protocol. For the purpose of receipt-freeness, the knowledge of his own key is *essential*. If a voter can prove that he does not know his own secret key, then he can obtain a receipt (this holds for this protocol as well as for the protocol in [SK95]). We assume that the underlying public-key infrastructure guarantees that each voter knows his own key, but nevertheless we present a verification protocol (see Appendix A). Note that the receipt-free property is still achieved even if a voter discloses his secret key to the vote-buyer.

Generality. The protocol is a 1-out-of-L voting scheme, where each entitled voter may submit one vote from the set $\mathcal{V}$ of valid votes, $|\mathcal{V}| = L$ (e.g. $L = 2$ and $\mathcal{V} = \{-1, 1\}$). The goal of the protocol is to securely compute the tally as the sum of the cast votes. Note that the restriction on a discrete set of valid votes is necessary in any receipt-free voting scheme. Voting schemes that allow the voter to cast an arbitrary string as his vote cannot ensure receipt-freeness, because they allow the voter to tag his vote.

[2] If the coercer is able to tap *all* communication channels between the voter and the authorities, then apparently the voter's private information (including secret key and randomness) is a receipt of the vote he has cast. The model for incoercible multi-party computation [CG96] does not assume physically secure channels, but participants who *want* to prove a certain behavior can do so in this setting, thus receipt-freeness is not achieved.

Security. The security of the protocol comprises that the correctness of the computed tally is guaranteed as long as at least t authorities remain honest during the whole protocol execution (correctness); that any set of less than t authorities cannot decrypt any cast vote (privacy); and that a voter cannot prove to a third party which particular vote he has cast (receipt-freeness). In general, we assume that authorities do not collude with the vote-buyer, respectively the coercer (as assumed in all previous papers considering receipt-freeness). However, under certain circumstances some colluding can be tolerated. This is discussed in detail in Section 3.5.

3.2 Protocol Overview

The basic idea of the voting phase is illustrated in Fig. 1: First, each valid vote is encrypted in some deterministic way (e.g., by using the encryption function with "randomness" 0). This list of encrypted votes is publicly known (on the very left in the figure). Then, the first authority picks this list, shuffles it, and hands it to the next authority. To shuffle the list means to re-randomize each entry and to permute the order of the entries. In the figure, encryption is illustrated in terms of drawing a circle around the secret, and re-randomization is illustrated by rotating the secret. Then, the next authority picks the list, shuffles it, and so on. In addition to this shuffling, each authority must secretly reveal to the voter how the list was reordered yet in a *privately verifiable* manner through a secure untappable channel. This allows the voter to keep track of the ordering of the encrypted entries, and once each authority has shuffled the list, he can point to the encrypted vote of his choice. In order to prevent a voter who is colluding with an authority from casting an invalid vote, each authority must publicly prove that she shuffled correctly (without revealing the reordering, not shown in the figure). Votes cast this way are receipt-free: due to the *private* verifiability of how shuffling was performed, the voter has no way to convince a third party of the content of his vote. This property will be achieved by using designated-verifier proof technique [JSI96].

3.3 Requirements for the Basic Protocol

We assume a basic (non receipt-free) voting protocol based on homomorphic encryption for the stated model, and we require some extra properties of its encryption function. Let E be the (probabilistic) encryption function, and let $E(v)$ denote the set of encryptions for a vote v. An encryption e of vote v is one particular encryption of v, i.e. $e \in E(v)$. We require the following properties to be satisfied. The properties 1–3 are straightforward requirements for the encryption function of any voting scheme. The properties 4–6 are required exclusively in order to introduce the receipt-free property.

1. **Encryption Secrecy**
   For any group of less than t authorities it must be infeasible to decrypt any encryption e.

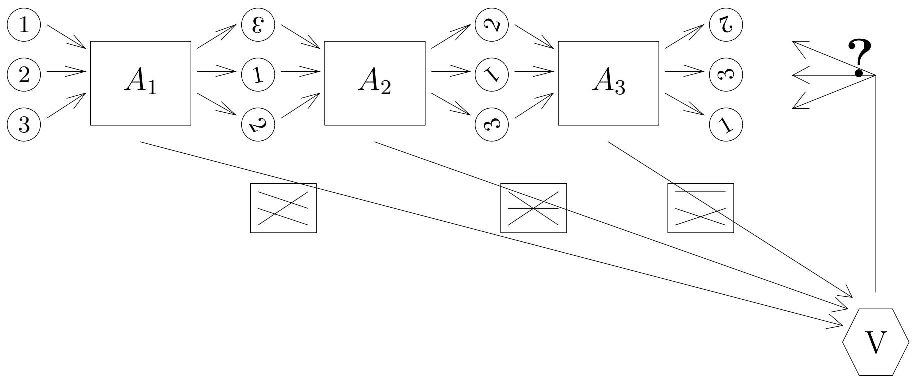

Fig. 1. Constructing a vote in $\mathcal{V} = \{1, 2, 3\}$ with 3 authorities.

2. **Homomorphic Property**
   We assume that the encryption function is homomorphic, that is, given the encryptions $e_1 \in E(v_1)$ and $e_2 \in E(v_2)$, the *addition* of these encryptions yields an encryption $e = e_1 \oplus e_2$ that encrypts the sum vote, i.e. $e \in E(v_1 + v_2)$. We require that this addition can be computed efficiently without any secrets.
3. **Verifiable Decryption**
   We require an efficient protocol for verifiably decrypting an encrypted sum vote, that is given any encryption $e \in E(T)$ the authorities can provide the sum of votes T and a proof that e indeed decrypts to T. This decryption and the proof must also work if up to $N - t$ authorities refuse cooperation or even misbehave maliciously. This protocol must not reveal any information that could weaken Property 1 (secrecy) of other encryptions.
4. **Random Re-encryptability**
   We require an algorithm for random re-encryption of any encryption e. Given $e \in E(v)$ (where typically v is unknown), there is a probabilistic re-encryption algorithm R that outputs $e' \in E(v)$, where e' is uniformly distributed over $E(v)$. We call the randomness used for generating e' the *witness*.
5. **Existence of a 1-out-of-L Re-encryption Proof**
   Based on the random re-encryptability property, we assume the existence of an efficient protocol that given an encryption e, a list $e_1, \ldots, e_L$ of encryptions, and a witness that e_i is a re-encryption of e (for a given i), proves that indeed e_i is a re-encryption of e, without revealing i. This proof is called 1-out-of-L re-encryption proof.
6. **Existence of a Designated-Verifier Re-encryption Proof**
   We assume the existence of an efficient protocol that given encryptions e and e' and a witness for e' being a re-encryption of e, proves the existence of such a witness in a manner that only the designated verifier can verify

its correctness [JSI96]. This proof is called designated-verifier re-encryption proof.

3.4 Introducing Receipt-Freeness

Given a voting protocol for the stated model which satisfies the requirements of the previous section, we can construct a receipt-free voting protocol. We first show how votes are generated (by the authorities) and how the voter casts his vote, then how tallying is performed.

Vote Generation. Without loss of generality, assume that for each valid vote $v_i \in \mathcal{V}$, there exists a *standard encryption* $e_i^{(0)}$, where it is clear which v_i a given encryption $e_i^{(0)}$ belongs to.[3] Hence, $e_1^{(0)}, \dots, e_L^{(0)}$ is a public list of all standard-encrypted valid votes.

In turn, for each authority A_k (where $k = 1, \dots, N$):

1. A_k picks the list $e_1^{(k-1)}, \dots, e_L^{(k-1)}$ of encrypted valid votes (for the first authority A_1, this is the public list of standard-encrypted valid votes, and for all succeeding authorities, this is the list of the previous authority). Then the authority shuffles this list randomly, and hands it to the next authority. To shuffle the list means to re-encrypt each encrypted vote (Property 4) and to permute the order of the list. More precisely, the authority randomly selects a permutation $\pi_k : \{1, \dots, L\} \rightarrow \{1, \dots, L\}$, computes a random re-encryption of $e_i^{(k-1)}$ and assigns it to $e_{\pi_k(i)}^{(k)}$ (for all $i = 1, \dots, L$).
2. A_k publicly proves that she honestly shuffled, namely by proving for each i, there exists a re-encryption of $e_i^{(k-1)}$ in the list $e_1^{(k)}, \dots, e_L^{(k)}$ without revealing which (1-out-of-L re-encryption proof, Property 5).
3. A_k secretly conveys to the voter the permutation π_k she used for reordering the encrypted votes and proves privately to him its correctness. More precisely, the permutation π_k and a designed-verifier proof for each $i = 1, \dots, L$, that $e_{\pi_k(i)}^{(k)}$ is a re-encryption of $e_i^{(k-1)}$ (Property 6), is sent through the untappable channel to the voter.
4. If the voter does not accept the proof, he publicly complains about the authority. If the voter does so, then we set $e_1^{(k)} = e_1^{(k-1)}, \dots, e_L^{(k)} = e_L^{(k-1)}$, i.e. the shuffling of this authority is ignored. The voter may complain against at most $N - t$ authorities.

Casting a Vote. The voter derives the position i of the encrypted vote $e_i^{(N)}$ of his choice, and publicly announces it.

Tallying. The chosen encrypted votes of all voters are then summed for tallying. More precisely, they are added (using homomorphic addition $\oplus$, Property 2) to achieve an encryption $E(T)$ of the sum T of the votes. The authorities decrypt and output T and prove its correctness (Property 3).

[3] One technique to generate such encrypted votes is to use the probabilistic encryption algorithm E, and give as randomness the all-0 string. Such an encrypted vote $e_i^{(0)}$ can be decrypted by trying all valid votes $v \in \mathcal{V}$.

3.5 Security

Correctness. The correctness of the tally is guaranteed if all voters can cast the vote they wish (i.e. can trace the permutations of the authorities, Property 6), if they cannot cast invalid votes (Property 5), if the correct encrypted sum can be publicly computed (Property 2), and if the decryption of the sum is verifiable (Property 3).

Privacy. The privacy of each voter is guaranteed if an encrypted vote cannot be decrypted by an outstanding person or by any group of less than t authorities (Property 1). Also, given a list of encrypted votes and a shuffled list, it must be infeasible to find out which vote in the original list was permuted to which vote in the shuffled list (Property 4). Since at least t shufflings are performed correctly (at most $N-t$ shufflings can be skipped by a complaining voter), $t-1$ colluding authorities cannot find out the reordering of the list.

Receipt-Freeness. The voter actively interacts at two points: First (in vote generation), the voter can disable the shuffling of up to $N-t$ authorities, and second (in vote casting), the voter points to the encrypted vote of his choice. Through the untappable channels, the voter receives the permutations π_k and the designated-verifier proofs for the correctness of each π_k. Due to the non-transferability of designated-verifier proofs (Property 6) and the untappability of the channels used he can lie for any of these permutations π_k, and this is sufficient for not being able to prove the cast vote. Note that although the proposed scheme is receipt-free, a coercer still can coerce a voter not to vote, or can coerce a voter to vote randomly.

In case that authorities collude with a vote-buyer or a coercer, then apparently receipt-freeness is still ensured as long as each voter knows at least one authority not colluding with the vote-buyer (then the voter can lie for the permutation π_k of this authority A_k). If a voter does not know such an authority, he can select one authority at random and lie for this permutation. In the context of vote-buying this means that the voter can forge a receipt for a vote he did not cast, and the vote-buyer accepts such a forged receipt with probability linear in the number of authorities not colluding with him, which seems to be unacceptable for the vote-buyer. However, in the context of coercion, this means that the probability of a lying voter to be caught is linear in the number of authorities colluding with the coercer, and this seems to be unacceptable for the voter.

4 [CGS97] Made Receipt-Free

In this section, we construct a receipt-free 1-out-of-L voting scheme based on the construction of Sect. 3 and on the protocol of Cramer, Gennaro, and Schoenmakers [CGS97].

4.1 Homomorphic ElGamal Encryption

The encryption scheme is exactly the same as used in [CGS97]. Here a very brief summary: The scheme is based on the ElGamal cryptosystem [E84]. Let

G be a commutative group of order $|G| = q$, where q is a large prime. G can be constructed as a subgroup of Z_p^*, where p is a large prime, but can also be obtained from elliptic curves. In the sequel, all operations are meant to be performed in G.

Let g be a generator of G, i.e. $G = \langle g \rangle$. The secret key z is chosen uniformly from Z_q, and the public key is $h = g^z$. The key pair (z, h) is constructed in a way that each authority receives a share z_i of z in a (t, N)-threshold secret-sharing scheme and is publicly committed to this share by $h_i = g^{z_i}$ [Ped91,CGS97]. Also, γ is another (independent) generator of G. The set $\mathcal{V}$ of valid votes contains L values in Z_q. An encryption of a vote $v \in \mathcal{V}$ is given by

$$E(v) = (g^\alpha, \gamma^v h^\alpha),$$

where $\alpha \in_R Z_q$ is a random number and γ^v is the "message" in the context of ElGamal.[4] We further let $e_v^{(0)} = (1, \gamma^v)$ be the standard encryption of v.

4.2 Encoding of Votes

There are several ways of encoding L votes in Z_q, such that the sum of several votes yields the sum of each type of vote. If for example $L = 2$, then one could set $\mathcal{V} = \{+1, -1\}$ and can derive how many 1-votes and how many (-1)-votes were cast from the sum and the number of cast votes.

For particular cases with $L > 2$, one can still use a similar approach. For example, if voters are allowed to cast "yes", "no", or "empty", and we are only interested in whether there are more "yes" or more "no" votes (disregarding the number of "empty" votes), one can use the encoding 1 for "yes", -1 for "no", and 0 for "empty".

However, if it must be possible to derive the exact number of cast votes for each choice, then more involved approaches are necessary. Along the ideas of [CFSY96], one can set $\mathcal{V} = \{1, M, M^2, \ldots, M^{L-1}\}$, where M denotes the number of voters. One can easily compute the number of cast votes for each choice, once the sum of the votes is computed.

We note that in any examples given in this subsection, decryption of the tally requires computing the discrete logarithm of γ^T, where T is the sum of all cast votes (as in [CGS97]). This can be done with complexity $O(\sqrt{M}^{L-1})$, see [CGS97] for more details.

4.3 Main Protocol

The main protocol is according to the generic protocol of Sect. 3. All we have to show is that the above encryption scheme satisfies the required properties of Sect. 3:

[4] The original ElGamal scheme is homomorphic with respect to multiplication. In order to achieve it to be homomorphic with respect to addition (Property 2), the message is chosen as γ^v. Multiplication of two messages corresponds to addition of the votes.

1. (Secrecy) The secret key z is shared among the authorities such that any $t-1$ authorities cannot compute z. Violating the secrecy of the scheme would mean to either break ElGamal [E84] or the secret-sharing scheme [Ped91].
2. (Homomorphic Property) Addition of two encryptions $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ is defined as
$$e_1 \oplus e_2 = (x_1 x_2, y_1 y_2) \ .$$
It is obvious that if $e_1 \in E(v_1)$ and $e_2 \in E(v_2)$, then $(e_1 \oplus e_2) \in E(v_1 + v_2)$.
3. (Verifiable Decryption) In order to decrypt T from $e = (x, y)$ the authorities first jointly compute, reveal and prove $\hat{x} = x^z$. This can be achieved by having every authority A_i compute $\hat{x}_i = x^{z_i}$, where z_i is A_i's share of the secret key z, and then compute $\hat{x}$ from $\hat{x}_i$. This is possible if at least t authorities reveal and prove $\hat{x}_i$. More details can be found in [Ped91,CGS97]. Once $\hat{x}$ is known, one can compute
$$\frac{y}{\hat{x}} = \frac{\gamma^T \cdot h^\alpha}{(g^\alpha)^z} = \gamma^T \ .$$
Then, the authorities must find T. The computation complexity of this task is discussed in Sect. 4.2
4. (Re-encryptability) The re-encryption $e' = (x', y')$ of an encrypted vote $e = (x, y)$ is given by
$$(x', y') = (g^\xi x, h^\xi y)$$
for a random integer $\xi \in_R Z_q$. Clearly, if ξ is chosen uniformly in Z_q, then (x', y') is uniformly distributed. This ξ serves as a witness of re-encryption.
5. (1-out-of-L Re-encryption Proof) An efficient witness indistinguishable protocol with which an authority can prove that a re-encryption of a given encrypted vote e is contained in the list $e_1, \ldots, e_L$ will be given in Sect. 4.4.
6. (Designated-Verifier Re-encryption Proof) An efficient witness indistinguishable protocol with which an authority can prove privately that an encrypted vote e' is a re-encryption of e will be given in Sect. 4.5.

4.4 1-out-of-L Re-encryption Proof

We present a witness indistinguishable protocol with which a prover can prove that for an encrypted vote (x, y), there is a re-encryption in the L encrypted votes $(x_1, y_1), \ldots, (x_L, y_L)$ (1-out-of-L re-encryption proof). The protocol is based on techniques presented in [CDS94,CFSY96,CGS97]. For this protocol, assume that (x_t, y_t) is a re-encryption of (x, y), and the re-encryption randomness (the witness) is ξ, i.e. $(x_t, y_t) = (g^\xi x, h^\xi y)$.

1. The prover selects $d_1, \ldots, d_L$ and $r_1, \ldots, r_L$ at random, and computes
$$a_i = \left(\frac{x_i}{x}\right)^{d_i} \cdot g^{r_i} \text{ and } b_i = \left(\frac{y_i}{y}\right)^{d_i} \cdot h^{r_i} \quad (\text{for } i = 1, \ldots, L)$$
and sends it to the verifier. Note that these values commit the prover to d_i and r_i for all $i = 1, \ldots, L$ except for $i = t$. a_t and b_t only commit the prover to a value $w = \xi d_t + r_t$, since $a_t = g^{\xi d_t + r_t}$ and $b_t = h^{\xi d_t + r_t}$. This means that the prover still can change d_t and r_t *after* this round.
2. The verifier picks a random challenge $c \in_R Z_q$ and sends it to the prover.
3. The prover modifies d_t such that $c = d_1 + \ldots + d_L$, modifies r_t such that $w = \xi d_t + r_t$ (both mod q) and sends $d_1, \ldots, d_L$ and $r_1, \ldots, r_L$ (with d_t and r_t modified) to the verifier.
4. The verifier tests whether
$$c \stackrel{?}{=} d_1 + \ldots + d_L \pmod q$$
$$a_i \stackrel{?}{=} \left(\frac{x_i}{x}\right)^{d_i} \cdot g^{r_i} \quad (\text{for } i = 1, \ldots, L)$$
$$b_i \stackrel{?}{=} \left(\frac{y_i}{y}\right)^{d_i} \cdot h^{r_i} \quad (\text{for } i = 1, \ldots, L)$$

The proposed protocol is a 3-move witness-indistinguishable proof. Using the Fiat-Shamir-heuristic [FS86] the proof can be converted to be non-interactive. Using a technique of [CFSY96], we can even achieve a proof that only requires the prover to send $2L$ elements of G. Let $\mathcal{H}$ denote a cryptographic hash function, then

1. The prover computes a_i and b_i (for $i = 1, \ldots, L$) as in the interactive proof.
2. Then the prover computes the challenge $c = \mathcal{H}(E \| a_1 \| \ldots \| a_L \| b_1 \| \ldots \| b_L)$, where $a \| b$ is the concatenation of a and b, and $E = (x \| y \| x_1 \| x_2 \| \ldots \| x_L \| y_L)$ is the environment.
3. For this challenge, the prover computes d_i and r_i (for $i = 1, \ldots, L$). The proof is the $2L$-vector $(d_1, \ldots, d_L, r_1, \ldots, r_L)$.
4. A verifier examines whether
$$d_1 + \ldots + d_L \stackrel{?}{=} \mathcal{H}\left(E \middle\| (\tfrac{x_1}{x})^{d_1} g^{r_1} \middle\| \ldots \middle\| (\tfrac{x_L}{x})^{d_L} g^{r_L} \middle\| \left(\tfrac{y_1}{y}\right)^{d_1} h^{r_1} \middle\| \ldots \middle\| \left(\tfrac{y_L}{y}\right)^{d_L} h^{r_L}\right).$$

4.5 Designated-Verifier Re-encryption Proof

Each authority secretly conveys and proves to the voter how she reordered the list of encrypted votes. Therefore, for each $i = 1, \ldots, L$, the authority proves that $e^{(k)}_{\pi(i)}$ is a re-encryption of $e^{(k-1)}_i$. In the sequel, based on techniques from [JSI96], we show how the authority can privately prove that (x', y') is a re-encryption of (x, y), where ξ is the witness, i.e. $(x', y') = (g^\xi x, h^\xi y)$. The voter's secret key is denoted as z_v and the corresponding public key is given by $h_v = g^{z_v}$. This protocol relies on the voter's knowledge of his secret-key. If this property is not ensured by the underlying public-key infrastructure, a protocol for guaranteeing it must be employed (see Appendix A).

1. The prover selects d, w and r at random, computes
$$a = g^d \ , \ \ b = h^d \ , \text{and} \ \ s = g^w h_v^r \ ,$$
and sends it to the verifier. These values commit the prover to d, w and r. However, s is a chameleon commitment for w and r, and the verifier can use his knowledge of z_v to open s to arbitrary values w' and r' satisfying $w' + z_v r' = w + z_v r$.
2. The verifier picks a random challenge $c \in_R Z_q$ and sends it to the prover.
3. The prover computes $u = d + \xi(c + w)$ and sends w, r, u to the verifier.
4. The verifier tests whether
$$s \stackrel{?}{=} g^w h_v^r$$
$$g^u \stackrel{?}{=} \left(\frac{x'}{x}\right)^{c+w} \cdot a$$
$$h^u \stackrel{?}{=} \left(\frac{y'}{y}\right)^{c+w} \cdot b$$

This protocol can be made non-interactive using Fiat-Shamir-heuristic [FS86]:

1. The prover computes a, b and s as in the interactive proof.
2. Then the prover computes the challenge $c = \mathcal{H}(E\|a\|b\|s)$, where $a\|b\|s$ means the concatenation of a, b and s, and $E = (x\|y\|x'\|y')$ is the environment.
3. For this challenge, the prover computes u. The proof is the vector (c, w, r, u).[5]
4. A verifier tests whether
$$c \stackrel{?}{=} \mathcal{H}\left(E \left\| \frac{g^u}{\left(\frac{x'}{x}\right)^{c+w}} \right\| \frac{h^u}{\left(\frac{y'}{y}\right)^{c+w}} \left\| g^w h_v^r \right.\right) \ .$$

Now we show how that the verifier who knows the secret z_v such that $g^{z_v} = h_v$ can generate the above proof for any (x, y) and $(\tilde{x}, \tilde{y})$. The key is that the value s does *not* stick the verifier to w and r. The verifier selects α, β and $\tilde{u}$ at random, and computes
$$\tilde{c} = \mathcal{H}\left(\tilde{E} \left\| \frac{g^{\tilde{u}}}{\left(\frac{\tilde{x}}{x}\right)^{\alpha}} \right\| \frac{h^{\tilde{u}}}{\left(\frac{\tilde{y}}{y}\right)^{\alpha}} \left\| g^{\beta} \right.\right)$$
$$\tilde{w} = \alpha - \tilde{c} \pmod q$$
$$\tilde{r} = \frac{\beta - \tilde{w}}{z_v} \pmod q$$

and sets $(\tilde{c}, \tilde{w}, \tilde{r}, \tilde{u})$ as the proof. It is easy to see that this proof passes the above verification, i.e. for any $(\tilde{x}, \tilde{y})$, the voter can "prove" that it is a re-encryption of (x, y).

[5] We note that this construction is slightly more efficient than the one presented in [JSI96], where they require a 5-vector as proof.

4.6 Communication-Complexity Analysis

In this section we analyze the communication complexity of the 1-out-of-L voting scheme. We assume that there are N authorities and M active (participating) voters. Let K denote the number of bits that are used to store an element of the group G.

Both initialization of the ElGamal keys and revealment of the final result use constant (in M) many messages and are thus ignored. The relevant costs are related to shuffling and to the designated-verifier proofs. For each active voter, in turn every authority shuffles the list of encrypted votes, posts the new list to the bulletin board ($2L$ group elements), posts a proof for honest shuffling ($L \cdot 2L$ group elements), and secretly conveys and proves the reordering to the voter ($L \log_2 L$ bits for the permutation and $L \cdot 4$ group elements for the proofs). Finally, the voter posts the index of the encrypted vote of his choice to the bulletin board ($\log_2 L$ bits). In total, there are $2KLMN(L+1) + M \log_2 L$ bits posted to the bulletin board, and $LMN(4K + \log_2 L)$ bits transfered through the untappable channels.

When the protocol for ensuring that each voter knows his own secret key (cf. Appendix A) is considered as part of the protocol (and not as part of the public-key infrastructure), then the number of bits posted to the bulletin board is increased by $MK(N + t)$, and the number of bits transfered through the untappable channels is increased by MNK.

5 Efficient Receipt-Free 1-out-of-2 Voting

In this section we give a more efficient receipt-free protocol for 1-out-of-2 voting based on the scheme of [CGS97]. We take advantage of the encryption scheme that enables *flipping* of votes easily. That is, one can generate the opposite of an encrypted vote e without knowing the vote.

We define the set of valid votes $\mathcal{V} = \{-1, +1\}$ and we use the same encryption scheme as in Sect. 4. We define $e^{(0)} = (1, \gamma)$ be the standard encryption for the vote 1.

Consider an encrypted vote $(x, y) \in E(v)$ (where $v \in \mathcal{V}$ is unknown). One can easily *flip* the vote by $\overline{e} = (x^{-1}, y^{-1})$ which yields $\overline{e} \in E(-v)$. In other words, for every encrypted vote, the encrypted opposite vote is implicitly defined. Following we give an efficient receipt-free protocol for 1-out-of-2 elections that makes extensive use of this flipping property.

In turn, for each authority A_k (where $k = 1, \ldots, N$):

1. A_k picks $e^{(k-1)}$, an encrypted 1-vote or (-1)-vote (for the first authority A_1, this is the standard encryption of the 1-vote, and for all succeeding authorities, this is the encrypted vote of the previous authority). Then the authority computes a random re-encryption of $e^{(k-1)}$ and either flips it or not, and assigns the result to $e^{(k)}$.

2. A_k publicly proves that she honestly re-encrypted (and optionally flipped), namely by proving that either $e^{(k)}$ or $\overline{e^{(k)}}$ is a re-encryption of $e^{(k-1)}$. Therefore, the proof of Sect. 4.4 is used, where $L = 2$.
3. A_k secretly conveys and proves privately to the voter whether she flipped or not. This proof will be the same designated-verifier proof as given in Sect. 4.5, where $L = 2$.
4. At most $N-t$ times the voter may not accept the proof and publicly complain about the authority. Then $e^{(k)} = e^{(k-1)}$.

Finally the voter casts his vote by announcing whether his vote is $e^{(N)}$ or $\overline{e^{(N)}}$. This encrypted vote is then summed for tallying.

The analysis of this scheme gives that totally $6KMN + M$ bits are posted to the bulletin board, and $MN(4K + 1)$ bits are sent over the untappable channels (both quantities are almost half of the costs with the protocol of Sect. 4, where $L = 2$).

A careful analysis of the receipt-free voting scheme of Sako and Kilian [SK95] for security parameter ℓ (the number of rounds in the non-interactive cut-and-choose proofs) reveals the complexity of that scheme: There are in total $(9K + \log_2 M)MN\ell$ bits sent over the public channels and $KMN\ell$ bits over the untappable channels. This is more than $3\ell/2$ times more on the public channels and $\ell/4$ times more on the untappable channels than the scheme of this paper. The costs of the protocol from Appendix A must be added to all quantities if required.

6 Concluding Remarks

We have presented a generic construction of a receipt-free protocol from a given basic voting scheme. By applying this generic construction to the voting protocol of [CGS97] we obtain an efficient receipt-free 1-out-of-L voting protocol, and by tailoring it to 1-out-of-2 voting this results in a protocol which is ℓ times more efficient than the protocol of [SK95] with security parameter ℓ. Due to the protocol failure in [BT94], the constructions in this paper give the first receipt-free voting scheme based on homomorphic encryptions.

Acknowledgments

Special thanks go to Herbert Schnider for his cooperation on analyzing the security of [BT94]. Furthermore, we would like to thank Daniel Bleichenbacher, Ronald Cramer, Rosario Gennaro, Markus Jakobsson, Ueli Maurer, Kouichi Sakurai, Berry Schoenmakers, and Stefan Wolf for many interesting discussions. Also, we would like to thank the anonymous referees for their useful comments on the paper.

References

Abe99. Masayuki Abe. Mix-networks on permutation networks In *Advances in Cryptology — ASIACRYPT '99*, vol. 1716 of *LNCS*, pp. 258–273. Springer-Verlag, 1999.

Ben87. Josh Cohen Benaloh: *Verifiable Secret-Ballot Elections.* Yale University PhD thesis, YALEU/DCS/TR-561, 1987.

BT94. Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pp. 544–553. ACM, 1994.

CDNO97. Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *Advances in Cryptology — CRYPTO '97*, vol. 1294 of *LNCS*, pp. 90–104. Springer-Verlag, 1997.

CG96. Ran Canetti and Rosario Gennaro. Incoercible multiparty computation. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1996.

Cha81. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94*, vol. 839 of *LNCS*. Springer-Verlag, 1994.

CF85. Josh D. Cohen (Benaloh) and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 372–382. IEEE, 1985.

CY86. Josh Cohen (Benaloh) and Moti Yung: Distributing the Power of a Government to Enhance the Privacy of Voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 52–62. ACM, 1986.

CFSY96. Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology — EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 72–83. Springer-Verlag, May 1996.

CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8:481–489, 1997. Preliminary version in *Advances in Cryptology — EUROCRYPT '97.*

E84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, vol. 196 of *LNCS*, pp. 10–18. Springer-Verlag, 1984.

Fel87. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. 28th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 427–437, 1987.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194. Springer-Verlag, 1986.

FOO92. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology — AUSCRYPT '92*, pp. 244–251, 1992.

JSI96. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated-verifier proofs and their applications. In *Advances in Cryptology — EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 143–154. Springer-Verlag, 1996.

Jak98. Markus Jakobsson. A Practical Mix. In *Advances in Cryptology — EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 448–461, Springer-Verlag, 1998.

MH96. Markus Michels and Patrick Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *Advances in Cryptology — ASIACRYPT '96*, vol. 1163 of *LNCS*, pp. 125–132. Springer-Verlag, 1996.

Oka97. Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. of Workshop on Security Protocols '97*, vol. 1361 of *LNCS*, pp. 25–35. Springer-Verlag, 1997.

OKST97. Wakaha Ogata, Kaoru Kurosawa, Kazue Sako and Kazunori Takatani Fault Tolerant Anonymous Channel In *Information and Communications Security ICICS '97*, vol. 1334 of *LNCS*, pp. 440–444. Springer-Verlag, 1997.

PIK93. Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology — EUROCRYPT '93*, vol. 765 of *LNCS*, pp. 248–259. Springer-Verlag, 1993.

Ped91. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology — EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 522–526. Springer-Verlag, 1991.

Sak94. Kazue Sako. Electronic voting schemes allowing open objection to the tally. In *Transactions of IEICE,* vol. E77-A No.1, Jan. 1994.

SK94. Kazue Sako and Joe Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptology — CRYPTO '94*, vol. 839 of *LNCS*, pp. 411–424. Springer-Verlag, 1994.

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme – A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 393–403. Springer-Verlag, 1995.

Sha79. Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

A Ensuring Knowledge of the Secret-Key

In a model providing receipt-freeness, it is essential that each voter knows his own secret-key. We assume that this verification is part of the underlying public-key infrastructure, but nevertheless we provide a protocol that ensures a voter's knowledge of his secret-key. This protocol may be performed as part of the key registration (in the public-key infrastructure), or as part of the voting protocol if the key infrastructure does not provide this property. This protocol requires a secure one-way untappable channel as used in the vote generation phase.

The following protocol is based on Feldman's secret-sharing scheme [Fel87]. It establishes that a voter v knows the secret key z_v corresponding to his public key h_v (where $g^{z_v} = h_v$):

– The voter shares his secret key z_v among the authorities by using Feldman's secret-sharing scheme [Fel87]: The voter v chooses a uniformly distributed random polynomial $f_v(x) = z_v + a_1 x + \ldots + a_{t-1}x^{t-1}$ of degree $t-1$, and secretly sends[6] the share $s_i = f_v(i)$ to authority A_i (for $i = 1, \ldots, N$). Fur-

[6] Either the voter encrypts the share with the authority's public-key, or alternatively the authority first sends a one-time pad through the untappable channel, and the voter then encrypts with this pad.

thermore, the voter commits to the coefficient of the polynomial by sending $c_i = g^{a_i}$ for $i = 1, \ldots, t-1$ to the bulletin board.
- Each authority A_i verifies with the following equation whether the received share s_i indeed lies on the committed polynomial $f_v(\cdot)$:

$$g^{s_i} \overset{?}{=} h_v \cdot c_1^{i} \cdot \ldots \cdot c_{t-1}^{i^{t-1}} \qquad \left(= g^{z_v} \cdot g^{a_1 i} \cdot \ldots \cdot \ldots \cdot g^{a_{t-1} i^{t-1}} = g^{f_v(i)}\right) .$$

  If an authority detects an error, she complains and the voter is requested to post her share to the bulletin board. If the posted share does not correspond to the commitments, the voter is disqualified.
- Finally, every authority (which did not complain in the previous stage) sends her share through the untappable channel to the voter.

In the above protocol, clearly after the second step, either the (honest) authorities will have consistent shares of the voter's secret key z_v, or the voter will be disqualified. However, so far it is not ensured that the voter indeed knows the secret key, as the shares could have been provided by the coercer. In any case, in the final step the voter learns z_v. There are at least t honest authority who either complained (and thus their share is published), or who sent their share to the voter, and hence the voter can interpolate the secret key z_v.