Anonymous Voting by 2-Round Public Discussion

Feng Hao[1] and Peter Ryan[2] Piotr Zieliński[3]

[1] Thales E-Security, Cambridge, UK
[2] Faculty of Science, University of Luxembourg,
[3] Google Inc.
`feng.hao@thales-esecurity.com`, `peter.ryan@uni.lu`,
`piotrzielinski@google.com`

Abstract. In 2006, Hao and Zielínski proposed a 2-round anonymous veto protocol (called AV-net), which provides exceptional efficiency compared to related techniques. In this paper, we add a self-tallying function to the AV-net, making it a general-purpose voting protocol. The new protocol works in the same setting as the AV-net - it requires no trusted third parties or private channels, and participants execute the protocol by sending 2-round public messages. Compared with related voting protocols in past work, ours is significantly more efficient in terms of the number of rounds, computational cost and bandwidth usage.

1 Introduction

Electronic voting is one of the most intriguing cryptographic problems. Depending on whether trusted third parties are involved, it can be divided into two classes: 1) decentralized elections where the protocol is essentially run by the voters themselves; 2) centralized elections where trusted authorities are employed to administer the process [10].

In general, the first case allows for better voter privacy, and is thus most suitable for small-scale (boardroom) elections. The second case has the advantage of being more robust, in the sense of being more resilient in the face of voters attempting to disrupt the protocols, and hence often finds its applications in large-scale (countrywide) elections [10, 11].

In this paper, we focus on the first class. In particular, we aim to explore, with strong voter privacy as the primary objective, what is the best efficiency achievable through cryptographic means. First, let us look at the following example of the boardroom voting.

In a boardroom, corporate directors are discussing who is to become the new chairman. They decide to hold an election. All communication is public; any "private" talk between directors can be heard by all. And no trusted third parties exist. So how to arrange a voting protocol such that the voters' privacy will be preserved?

There are two challenges here. First, no trusted third parties exist. Many security problems could be easily solved if we assume a trusted third party, but then the "trusted" third party may become the one who breaks the security policy totally [1]. A standard approach used in many voting protocols is to distribute trust among several third parties by using a threshold scheme, so that the protocol does not rely on a single trusted entity [4, 5, 7]. However, our goal here is to eliminate the use of trusted third parties altogether.

The second challenge is that there are no voter-to-voter private channels. This is to ensure dispute-freeness – everybody can check whether all voters have faithfully followed the protocol [10,11]. This also has the advantage of minimizing the assumptions required for the protocol to be secure. Hence, protocols that depend on pairwise private channels are not suitable for our purpose [16].

Kiayias and Yung first studied this problem in detail and proposed a concrete voting protocol that fulfills these challenging requirements [10]. Their protocol has the following attractive features: it is self-tallying; it provides the maximum protection of the voters' privacy; and is dispute-free. The round efficiency is reasonably good – only 3 rounds. The downside of their protocol, however, is the heavy computational load for each voter which increases linearly with the number of voters [11].

Groth investigated the efficiency limitations of the Kiayias-Yung protocol in [11], and proposed a new protocol with the improved computational complexity. The computational load for each voter is relatively light and remains constant even with more voters. Unfortunately, Groth's protocol design trades off round efficiency for less computation. As a result, it requires $n + 1$ rounds, where n is the total number of voters. This is worse than the constant 3 rounds in the Kiayias-Yung protocol.

In this paper, we describe a new solution to this problem. Our solution is inspired by the Anonymous Veto network (AV-net) protocol [12]. In fact, the protocol in this paper can be seen as a generalization of the AV-net protocol with the added self-tallying function. We will show that our scheme is as secure as the Kiayias-Yung and Groth's [10, 11], but significantly more efficient than both.

2 The Protocol

We assume an authenticated public channel available for every participant. This assumption is also made in the Kiayias-Yung and Groth's protocols [10, 11]. (In fact, an authenticated public channel is a very basic requirement for not only voting [4] but also general multi-party secure computations [9, 16].) There are several ways to realize such a public channel: by using physical means or, more commonly, a public bulletin board where authentication can be achieved by using, for example, digital signatures [10, 11]. Apart from this basic requirement, we assume no trusted third parties or private channels.

2.1 Two-round Referenda

Let G denote a finite cyclic group of prime order q in which the Decision Diffie-Hellman (DDH) problem is intractable [3]. Let g be a generator in G. There are n participants, and they all agree on (G, g). Each participant P_i selects a random value as the secret: $x_i \in_R \mathbb{Z}_q$. Let us consider the single-candidate case first. Suppose a vote is either "yes" or "no". Then participants execute the following 2-round protocol:

Round 1. Every participant P_i publishes g^{x_i} and a zero knowledge proof for x_i. When this round finishes, each participant P_i checks the validity of the ZK proofs and computes

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} / \prod_{j=i+1}^{n} g^{x_j}$$

Round 2. Every participant publishes $g^{x_i y_i} g^{v_i}$ and a zero knowledge proof showing that v_i is one of $\{1, 0\}$.

$$v_i = \begin{cases} 1 & \text{if } P_i \text{ votes "yes"} \\ 0 & \text{if } P_i \text{ votes "no"} \end{cases} \tag{1}$$

To tally the "yes" votes, each participant, or indeed anyone observing the protocol, can compute $\prod_i g^{x_i y_i} g^{v_i} = g^{\sum_i v_i}$. The term $\sum_i v_i$ is the number of "yes" votes, which we denote γ. All zero knowledge proofs should be checked to ensure that no badly formed ballots have been cast that could distort the tally. The equality holds because $\prod_i g^{x_i y_i} = 1$ (Proposition 1, see also [12]). Since γ is normally a small number, it is not difficult to compute the discrete logarithm of g^γ, for example, by using exhaustive search or Shanks' baby-step giant-step algorithm. [17]

Proposition 1 *For the x_i and y_i as defined in the protocol, $\sum_i x_i y_i = 0$.*

Proof. By definition $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j$, hence

$$\begin{aligned}
\sum_i x_i y_i &= \sum_i \sum_{j<i} x_i x_j - \sum_i \sum_{j>i} x_i x_j \\
&= \sum_{j<i} \sum x_i x_j - \sum_{i<j} \sum x_i x_j \\
&= \sum_{j<i} \sum x_i x_j - \sum_{j<i} \sum x_j x_i \\
&= 0.
\end{aligned}$$

Table 1 illustrates this equality in a more intuitive way.

	x_1	x_2	x_3	x_4	x_5
x_1		−	−	−	−
x_2	+		−	−	−
x_3	+	+		−	−
x_4	+	+	+		−
x_5	+	+	+	+	

Table 1. A simple illustration of $\sum_{i=1}^{n} x_i y_i = 0$ for $n = 5$. The sum $\sum_{i=1}^{n} x_i \left(\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j\right)$ is the addition of all the cells, where $+$, $-$ represent the sign. They cancel each other out.

In the protocol, we use Zero Knowledge Proofs to ensure participants follow the protocol faithfully. The same technique is also used in [10, 11]. In the first round, each participant needs to demonstrate his knowledge of the exponent without revealing it. We can use Schnorr's signature [15], which is a well-established technique. Let H be a publicly agreed, secure hash function. To prove the knowledge of the exponent for g^{x_i} , one sends $(g^v, r = v - x_i z)$ where $v \in_R \mathbb{Z}_q$ and $z = H(g, g^v, g^{x_i}, i)$. This signature can be verified by anyone through checking whether g^v and $g^r g^{x_i z}$ are equal.

In the second round, each participant needs to demonstrate that the encrypted vote is one of $\{1, 0\}$ without revealing which one. For this we can adapt an efficient technique proposed by Cramer, Damgård and Schoenmakers in [6] (also see [4]). Firstly we need to convert the terms of our protocol into the form of ElGamal encryptions. This we can readily do, in a universally verifiable fashion, by treating the g^{y_i} terms as public keys and using the previously published terms of the protocol. We thus form:

$$(g^{x_i}, (g^{y_i})^{x_i} \cdot g^{v_i})$$

If participant i is playing by the rules, this will be an ElGamal encryption of g or 1 with public key g^{y_i} and randomisation x_i. More generally, given an ElGamal encryption $(x, y) = (g^{x_i}, h^{x_i} m)$, the CDS protocol demonstrates that m is either m_0 or m_1 without revealing which. This is achieved by proving the following OR statement:

$$\log_g x = \log_h (y/m_0) \quad \vee \quad \log_g x = \log_h (y/m_1)$$

Figure 1 shows a 3-move interactive protocol using the CDS technique, with $m_0 = g^0$ and $m_1 = g^1$ and $h = g^{y_i}$. Applying the Fiat-Shamir's heuristics makes the protocol non-interactive [8], by letting $c = H(i, x, y, a_1, b_1, a_2, b_2)$ where H is a publicly secure hash function. In summary, the 1-out-of-2 Knowledge Proof produced in Round 2 contains: $(w, a_1, b_1, a_2, b_2, d_1, d_{2,})$. More details on the 1-out-of-n Knowledge Proof can be found in [4, 6].

Note that we form these ElGamal ciphertexts in order to be able to apply the CDS protocol. We do not need to decrypt these ciphertexts in order to extract the tally.

Voter			Verifier
$v = 1$	$v = 0$		
$w, r_1, d_1 \in_R \mathbb{Z}_q$	$w, r_2, d_2 \in_R \mathbb{Z}_q$		
$x \leftarrow g^{x_j}$	$x \leftarrow g^{x_j}$		
$y \leftarrow h^{x_j} \cdot g$	$y \leftarrow h^{x_j}$		
$a_1 \leftarrow g^{r_1} x^{d_1}$	$a_1 \leftarrow g^{w}$		
$b_1 \leftarrow h^{r_1} y^{d_1}$	$b_1 \leftarrow h^{w}$		
$a_2 \leftarrow g^{w}$	$a_2 \leftarrow g^{r_2} x^{d_2}$		
$b_2 \leftarrow h^{w}$	$b_2 \leftarrow h^{r_2} (y/g)^{d_2}$	$\xrightarrow{x, y, a_1, b_1, a_2, b_2}$	
$d_2 \leftarrow c - d_1$	$d_1 \leftarrow c - d_2$	$\xleftarrow{\quad c \quad}$	$c \in_R \mathbb{Z}_q$
$r_2 \leftarrow w - x_j d_2$	$r_1 \leftarrow w - x_j d_1$	$\xrightarrow{d_1, d_2, r_1, r_2}$	$c \overset{?}{=} d_1 + d_2$
			$a_1 \overset{?}{=} g^{r_1} x^{d_1}$
			$b_1 \overset{?}{=} h^{r_1} y^{d_1}$
			$a_2 \overset{?}{=} g^{r_2} x^{d_2}$
			$b_2 \overset{?}{=} h^{r_2} (y/g)^{d_2}$

Fig. 1. 1-out-of-2 Proof of Knowledge: the ballot (x, y) is either $(g^{x_j}, h^{x_j} \cdot g)$ or (g^{x_j}, h^{x_j}) where $h = g^{y_j}$.

2.2 Extension to multiple candidates

We can extend the above single-candidate protocol to cater for multiple candidates[4]. This is useful as many practical elections involve more than one candidate. Depending on how the election is arranged, there can be several methods. A straightforward way is to run the single-candidate protocol in parallel for k candidates. Each voter casts a "yes/no" vote to each of the candidates. The tallying for each candidate is done independently, so the maximum tries for the exhaustive search is $k \times n$. The voting still executes in 2 rounds, but the computational load per participant will increase k times.

A more elegant (and more efficient in some aspects) method was described in [4], and subsequently adopted in [10,11]. In this case, each voter is only permitted to choose one candidate. The basic idea is to obtain k independent generators $g_1, g_2, \ldots, g_k$ (one for each candidate). The first round remains the same. In the second round, each participant sends $g^{x_i y_i} \cdot \varrho_i$ with a Zero Knowledge proof showing that ϱ_i is one of $\{g_1, g_2, \ldots, g_k\}$ (using the same CDS technique [6]). For tallying, one computes $\prod_i g^{x_i y_i} \cdot \varrho_i = g_1^{c_1} \cdot g_2^{c_2} \cdots g_k^{c_k}$ where c_1 to c_k are the counts of votes for the k candidates correspondingly.

However, one (slight) disadvantage of this approach lies in the complexity of the exhaustive search. Given n votes, k candidates and that each vote is cast to one of the k candidates, the number of possible voting results is $\binom{n+k-1}{k-1}$ $= O(n^{k-1})$ (see the Combinations with Repetitions problem [18]). This is less scalable than the previous $k \times n$, but should still be within the realm of feasible

[4] Obviously, if there are only two candidates, the same protocol can be used – instead of sending "Yes/No", one simply sends "A/B". Therefore, by "multiple", we really mean more than two.

computation for most practical elections where the number of candidates k is normally small [4]. Another disadvantage of this approach is that we need to show that the g_is are appropriately independent, i.e. that distinct values of the c_is do not give rise to the same product. Let N be the number of voters, then we require:

$$\forall c_i, c_i' \in N, \ \ \prod g_i^{c_i} = \prod g_i^{c_i'}, \ \ \Rightarrow \ \ \forall i \in \{1, ..., k\}, \ \ c_i = c_i'$$

A preferred way to deal with multiple candidates is to use the method due to Cramer et al [7] (also see [19]): suppose that we have n voters, choose m so that m is the smallest integer such that $2^m > n$. Now a vote for candidate 1 is encoded as 2^0, for candidate 2 as 2^m, for candidate 3 is 2^{2m}, and so on. In other words, redefine Eq. 1 as:

$$v_i = \begin{cases} 2^0 & \text{if } P_i \text{ votes candidate 1} \\ 2^m & \text{if } P_i \text{ votes candidate 2} \\ \cdots & \cdots \\ 2^{(k-1)m} & \text{if } P_i \text{ votes candidate } k \end{cases}$$

Tabulation is much as before: $\prod_i g^{x_i y_i} g^{v_i} = g^{\sum_i v_i}$. The votes are summed and the super-increasing nature of the encoding ensures that the total can unambiguously be resolved into the totals for the candidates. Hence, $\sum_i v_i = 2^0 \cdot c_1 + 2^m \cdot c_2 + \ldots + 2^{(k-1)m} \cdot c_k$, where c_1 to c_k are the counts of votes for the k candidates correspondingly. As before, this resolution requires searching over possible combinations, but of course pre-computation over (the more likely) combinations could speed this up.

3 Security Analysis

To analyse the security of the protocol, we consider two types of attackers: a passive one who merely eavesdrops on the communication, and an active one who takes part in the voting. Active attackers may collude in an effort to breach other voters' privacy or manipulate the voting outcome. The *full collusion* against a voter involves all the other voters in the election. Any decentralized voting protocol, by nature, cannot preserve the voter's privacy under that circumstance; attackers simply need to subtract their own votes from the final tally. Therefore, in this paper we only consider *partial collusion*, which involves some voters, but not all.

Under the threat model of partial collusion, an anonymous voting protocol should fulfill the following requirements (also see [10, 11]):

Maximum ballot secrecy: Each cast ballot is a ciphertext that is indistinguishable from random, and hence does not reveal anything about the voter's choice.

Self-tallying: After all ballots have been cast, anyone can compute the result without external help. This is a natural requirement for a distributed voting scheme.

Dispute-freeness: A scheme is dispute-free if everybody can check whether all voters act according to the protocol. In particular, this means that the result is publicly verifiable.

Notice that the Maximum ballot secrecy property is necessary but not sufficient to ensure a voter's privacy against a large collusion. Suppose that all the voters not in a collusion set all votes the same way then their ballot privacy will be violated. For example, if the final tally turns out to be 0, it will be clear to everyone that all voters had voted "no". This is because the tally – not the encrypted ballots – reveals information. Therefore, the best that is achievable to limit each voter to learn nothing more than his own vote and the final tally.

It has been shown in [10, 11] that the Kiayias-Yung and Groth's protocols satisfy the above requirements. We now show our protocol fulfills those requirements too.

3.1 Maximum Ballot Secrecy

Let us first look at the maximum ballot secrecy. In the protocol, each voter sends an ephemeral public key g^{x_i} in the first round and sends an encrypted ballot $g^{x_i y_i} g^{v_i}$, $v_i \in \{0, 1\}$, in the second round. (For simplicity of illustration, we omit the mention of Zero Knowledge Proofs, which will be addressed later.)

In the protocol, the value of y_i is determined by the private keys of all participants except P_i. The following lemma shows the security property of y_i.

Lemma 1 *The y_i is a secret random value to attackers in partial collusion against the participant P_i.*

Proof. Consider the worst case where only P_k $(k \neq i)$ is not involved in the collusion. Hence x_k is uniformly distributed over $\mathbb{Z}_q$ and unknown to colluders. The knowledge proofs required in the protocol show that all participants know their private keys x_i. Since y_i is computed from x_j $(j \neq i, k)$ known to colluders plus (or minus) a random number x_k, y_i must be uniformly distributed over $\mathbb{Z}_q$. Colluders cannot learn y_i even in this worst case.

Theorem 1 *Under the Decision Diffie-Hellman assumption, attackers in partial collusion against P_i cannot distinguish the ballot $g^{x_i y_i} \cdot g^{v_i}$, $v_i \in \{0, 1\}$, from a random group element.*

Proof. Besides the ballot, the data available to attackers concerning P_i include: g^{x_i} and a Zero-Knowledge Proof for the proof of the exponent x_i, and a Zero-Knowledge Proof for the proof of $v_i \in \{0, 1\}$. The first ZKP reveals one bit: whether the sender knows the discrete logarithm x_i of g^{x_i}. Also, the second ZKP only reveals one bit: whether the second round message is well-formed (that is whether it is the ElGamal encryption of $v_i \in \{0,1\}$). The ZKPs do not reveal any more information than what is intended.[5]

The secret x_i is chosen randomly by P_i. Lemma 1 shows that y_i is a random value, unknown to the attacker. Therefore, according to the Decision Diffie-Hellman assumption, one cannot distinguish between $g^{x_i y_i}$ (the *no*-vote) and a random element in the group [2]. Obviously, one cannot distinguish $g^{x_i y_i} \cdot g$ (the *yes*-vote) from random either. To sum up, one cannot distinguish $g^{x_i y_i} \cdot g^{v_i}$, $v_i \in \{0,1\}$, from random.

The above theorem states that the individual ballot does not leak any useful information about the voter's choice. It is the multiplication of all ballots that tells the tally. For each participant, what he learns from the election is strictly confined to the final tally and his own input.

Informally we can argue as follows: our protocol has the form suited to the game style definitions of security: the adversary is trying to distinguish with better than negligible advantage between the two ciphertexts. The ZK proofs of well-formedness simply serve to ensure that that we are indeed in the context of such a game. The ZK property of the proofs ensure that they do not help the adversary in making the distinction. As has been shown earlier, we can transform the terms of the protocol into ElGamal encryptions of the 0 or 1 votes. Therefore, an attack on the protocol could be used to violate the semantic security of the ElGamal encryption.

The above proof relates to the referendum case but is readily extensible to the multiple candidates case.

3.2 Self Tallying

Our protocol also fulfills self-tallying. In the protocol any interested party can compute the final tally without external help. This feature is realized by making use of the vanishing property of the proposition 1 (also see [12]). We let voters choose random private keys in the first round, and in the second round we combine the public keys in such a structured way that the random factors immediately vanish after Round 2, thus revealing the tally. The use of zero knowledge proofs in the protocol is to prevent active attacks, ensuring that voters faithfully follow the protocol [10, 11].

3.3 Dispute Freeness

In addition, our protocol satisfies dispute-freeness. First, we observe that the use of authenticated public channels ensures that any attempt to try to cast more

[5] We refer readers to [3, 6, 15] for security proofs on Schnorr's signature and Cramer et al's 1-out-of-n technique. However, we should note that if these techniques are chosen to realize ZKPs, then the proof of the protocol implicitly assumes a random oracle (i.e., a secure one-way hash function), since both techniques are secure under the random oracle model [6, 15].

than one *ballot* will be detected by the other participants. Furthermore, the use of the Cramer, Damgård and Schoenmakers zero-knowledge proofs serves to ensure that each ballot will encode exactly one candidate. Hence the *one-man-one-vote* requirement is enforced. This, along with the fact that the protocol is effectively self-tallying, ensures the overall accuracy of the count.

3.4 Limitations

There are however some limitations with our protocol. One is the potential subjection to the Denial of Service (DoS) attack. For a fully decentralized voting scheme, it naturally requires collaborative efforts from all voters otherwise it will not work. For example, if some voters refuse to send data in round 2, the tallying process will fail. This kind of attack is overt; everyone will know who the attackers are. To rectify this, voters need to expel the disrupters and restart the protocol; their privacy remains intact. However the voting process would be delayed, which may prove costly for large-scale (countrywide) elections.

Another limitation is the lack of coercion-resistance. If the voter is coerced to vote for a particular candidate, he could be forced to reveal his secret values and so prove how he voted. Normally, the coercion resistance is achieved by involving election authorities who provide trusted source of entropy [20]. But in a decentralized environment where no such trusted authorities exist, ensuring coercion-resistance seems very difficult.

Note that the above limitations also apply to the Kiayias-Yung and Groth's schemes [10, 11]. To a large extent, they reflect the decentralized nature of the protocol rather than any weakness in the protocol itself. A centralized voting scheme that employs tallying authorities is certainly more robust and scalable than a decentralized approach, but the trustworthiness of the authorities is often called into question. (If the people who administer the election receive a subpoena, they really have no interest in defying the order and going to jail in order to protect voters' privacy.)

So, whether an election should be centralized or decentralized depends on the application. For small-scale elections where the DoS attack and voter coercion are usually not of great concern, a decentralized protocol like ours can prove useful and efficient. For larger elections, the environment will be different and so will be the requirements. For example, the resistance to Denial of Service will be not only necessary but essential. In that case, it seems unavoidable that some form of trusted third parties will be introduced. Obviously, the trust must be threshold-controlled but still we need to trust the authorities do not collude altogether.

4 Comparison

In this section, we evaluate the efficiency of the protocol. There are many voting protocols in the past literature, however most of them involve using tallying authorities as trusted third parties [10,11]. Therefore, in this section, we mainly

Protocols	**Year**	**Round**	**Exp**	**KP for exponent**	**KP for equality**	**KP for 1-of-k**
Kiayias-Yung	2002	3	$2n+2$	$n+1$	n	1
Groth	2004	$n+1$	4	2	1	1
—	**2009**	**2**	**2**	**1**	**0**	**1**

Table 2. Comparison with past work

compare with the Kiayias-Yung and Groth's schemes [10,11]. Table 2 presents a summary of the comparison results.

The Kiayias-Yung's voting protocol executes in 3 rounds [10]. In round 1, each voter i sends out a public key $g_i = g^{\alpha_i}$, where $\alpha_i \in_R \mathbb{Z}_q$. This public key will be used as the voter's personal generator. In round 2, each voter defines additional n ephemeral private keys $(s_1, \ldots, s_n)$ and sends out the corresponding public keys $(g^{s_1}, \ldots, g^{s_n})$. This requires n exponentiations. In addition, he raises each of the n personal generators g_j to the power of s_j, leading to another n exponentiations. In round 3, everyone performs one more exponentiation before the tally can be universally computed. In total, there are $2n+2$ exponentiations; for each exponentiation, there is a corresponding zero knowledge proof to prevent cheating (see Table 2). More details can be found in [10].

Groth aims to improve the system complexity in the Kiayias-Yung's protocol [11]. His approach is to trade-off round efficiency for less computation. In the first round, everyone sends out an ephemeral public key. Then, each voter publishes data sequentially, encrypting the vote based on the previous voter's publication. Three exponentiations are need to encrypt the vote. The protocol finishes in $n+1$ rounds, where n is the total number of voters. Everyone – except the first voter[6] – needs to perform 4 exponentiations in total, and produces 4 knowledge proofs correspondingly.

Compared with the Kiayias-Yung and Groth's protocol, ours is a lot more efficient. The protocol has only two rounds, which is the best round efficiency for multi-party secure computation protocols [16]. For each voter, it requires one exponentiation to generate an ephemeral public key in round 1 and another exponentiation to encrypt the vote in round 2. In addition to each exponentiation, the voter needs to produce a corresponding Zero Knowledge proof. Overall, this kind of efficiency compares very favorably to both the Kiayias-Yung and Groth's protocols (see Table 2).

5 Conclusion

In this paper, we demonstrated how to arrange a 2-round anonymous election that requires no trusted third parties nor private voter-to-voter interactions. The voter's anonymity is preserved unless all of the other voters have been compromised. In addition, we showed that the protocol is exceptionally efficient. It has

[6] The first voter performs one less exponentiation than the rest [11].

only two rounds; in each round, a voter performs constant one exponentiation and produces one zero knowledge proof accordingly. This kind of efficiency compares very favorably to past decentralized voting protocols, and is close to the best possible.

References

1. R.J. Anderson, *Security Engineering : A Guide to Building Dependable Distributed Systems*, 2nd Edition, New York, Wiley, 2008.
2. D. Boneh, “The decision Diffie-Hellman problem,” Proceedings of the Third International Symposium on Algorithmic Number Theory, LNCS 1423, pp. 48–63, 1998.
3. D. Stinson, *Cryptography: Theory and Practice*, Third Edition, Chapman & Hall/CRC, 2006.
4. R. Cramer, R. Gennaro, and B. Schoenmakers, “A secure and optimally efficient multi-authority election scheme,” EUROCRYPT ’97, LNCS, vol. 1233, pp. 103-118, May 1997.
5. J.C. Benaloh, M. Yung, “Distributing the power of a government to enhance the privacy of voters,” Proceedings of the fifth annual ACM symposium on Principles of distributed computing, pp. 52-62, 1986.
6. R. Cramer, I. Damgård, B. Schoenmakers, “Proofs of partial knowledge and simplified design of witness hiding protocols,” Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, LNCS, vol. 839, pp. 174-187, 1994.
7. R. Cramer, M. Franklin, B. Schoenmakers and Moti Yung, “Multi-authority secret-ballot elections with linear work,” EUROCRYPT ’96, LNCS, vol. 1070, pp. 72-83, 1996.
8. A. Fiat, A. Shamir, “How to prove yourself: practical solutions to identification and signature problems,” Crypto ’86, LNCS, vol. 263, pp. 186-194, 1987.
9. O. Goldreich, S. Micali and A. Wigderson, “How to play any mental game or a completeness theorem for protocols with honest majority,” Proceedings of the nineteenth annual ACM Conference on Theory of Computing, pp. 218-229, 1987.
10. A. Kiayias, M. Yung, “Self-tallying elections and perfect ballot secrecy,” Public Key Cryptography ’02, LNCS, vol. 2274, pp. 141-158, 2002.
11. Jens Groth, “Efficient maximal privacy in boardroom votisng and anonymous broadcast,” Financial Cryptography ’04, LNCS, vol. 3110, pp. 90-104, 2004.
12. F. Hao, P. Zielínski, “A 2-round anonymous veto protocol,” Proceedings of the 14th International Workshop on Security Protocols, Cambridge, UK, 2006.
13. F. Hao, P. Y. A. Ryan, “Password Authenticated Key Exchange by Juggling,” Proceedings of the 16th International Workshop on Security Protocols, Cambridge, UK, April 2008.
14. D. Chaum, “The dining cryptographers problem: unconditional sender and recipient untraceability,” *Journal of Cryptology*, vol. 1, no. 1, pp. 65-67, 1988.
15. C.P. Schnorr, “Efficient signature generation by smart cards,” *Journal of Cryptology*, vol. 4, no. 3, pp. 161-174, 1991.
16. R. Gennaro, Y. Ishai, E. Kushilevitz and T. Rabin. “On 2-round secure multiparty computation,” Crypto ’02, LNCS, vol. 2442, pp. 178-193, 2002.
17. A.K. Lenstra and H.W. Lenstra, “Algorithms in number theory,” *Handbook of Theoretical Computer Science*, pp. 673-715, 1991.

18. R.P. Stanley, *Enumerative Combinatorics*, Cambridge University Press, 1997.
19. O. Baudron, P. Fouque, D. Pointcheval, G. Poupard, J. Stern, " Practical Multi-Candidate Election System,", Proceedings of the 20th ACM symposium on Principles of distributed computing, pp. 274-283, 2001.
20. A. Juels, D. Catalano, M. Jakobsson, "Coercion-Resistant Electronic Voting," WPES '05, pp. 61-70, 2005.