Published in IET Information Security
Received on 13th December 2008
Revised on 25th April 2009
doi: 10.1049/iet-ifs.2008.0127

ISSN 1751-8709

# Anonymous voting by two-round public discussion

*F. Hao*[1] *P.Y.A. Ryan*[2] *P. Zieliński*[3]
[1]*Thales E-Security, Jupiter House, Station Road, Cambridge, UK*
[2]*Faculty of Science, Technology and Communication, University of Luxembourg, 6 Rue Coudenhove-Kalergi, L-1359, Luxembourg-Kirchberg*
[3]*Google Inc., Belgrave House, 76 Buckingham Palace Road, London, SWIW 9TQ, UK*
*E-mail: haofeng66@gmail.com*

**Abstract:** In 2006, Hao and Zieliński proposed a two-round anonymous veto protocol (called AV-net), which provided exceptional efficiency compared to related techniques. In this study, the authors 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 two-round public messages. Compared with related voting protocols in past work, this 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: (i) decentralised elections where the protocol is essentially run by the voters themselves; (ii) centralised elections where trusted authorities are employed to administer the process [1].

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 [1, 2].

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 [3]. 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 [4–6]. 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 [1, 2]. This also has the advantage of minimising the assumptions required for the protocol to be secure. Hence, protocols that depend on pairwise private channels are not suitable for our purpose [7].

Kiayias and Yung [1] first studied this problem in detail and proposed a concrete voting protocol that fulfils these challenging requirements. 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 three rounds. The downside of their protocol, however, is the heavy computational load for each voter which increases linearly with the number of voters [2].

Groth [2] investigated the efficiency limitations of the Kiayias–Yung protocol in, 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 three 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 [8]. In fact, the protocol in this paper can be seen as a generalisation of the AV-net protocol with the added self-tallying function. We will show that our scheme is as secure as the Kiayias–Yung [1] and Groth's [2], but significantly more efficient than both.

## 2 Protocol

We assume an authenticated public channel available for every participant. This assumption is also made in the Kiayias–Yung [1] and Groth's protocols [2]. (In fact, an authenticated public channel is a very basic requirement for not only voting [4] but also general multi-party secure computations [7, 9].) There are several ways to realise 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 [1, 2]. 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 problem is intractable [10]. 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 two-round protocol:

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

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

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

$$v_i = \begin{cases} 1 & \text{if } P_i \text{ votes 'yes'} \\ 0 & \text{if } P_i \text{ votes 'no'} \end{cases} \qquad (1)$$

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 $\gamma$. All ZKPs 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 [8]). Since $\gamma$ is normally a small number, it is not difficult to compute the discrete logarithm of $g^{\gamma}$, for example, by using exhaustive search or Shanks' baby-step giant-step algorithm [11].

*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

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

Table 1 illustrates this equality in a more intuitive way.

In the protocol, we use ZKPs to ensure participants follow the protocol faithfully. The same technique is also used in [1, 2]. In the first round, each participant needs to demonstrate his knowledge of the exponent without revealing it. We can use Schnorr's signature [12], 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.

**Table 1** Simple illustration of $\sum_{i=1}^{n} x_i y_i = 0$ for $n = 5$

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $x_1$ | | $-$ | $-$ | $-$ | $-$ |
| $x_2$ | $+$ | | $-$ | $-$ | $-$ |
| $x_3$ | $+$ | $+$ | | $-$ | $-$ |
| $x_4$ | $+$ | $+$ | $+$ | | $-$ |
| $x_5$ | $+$ | $+$ | $+$ | $+$ | |

The sum $\sum_{i=1}^{n} x_i(\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j)$ is the addition of all the cells, where $+$, $-$ represent the sign. They cancel each other out

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 (CDS) in [13] (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

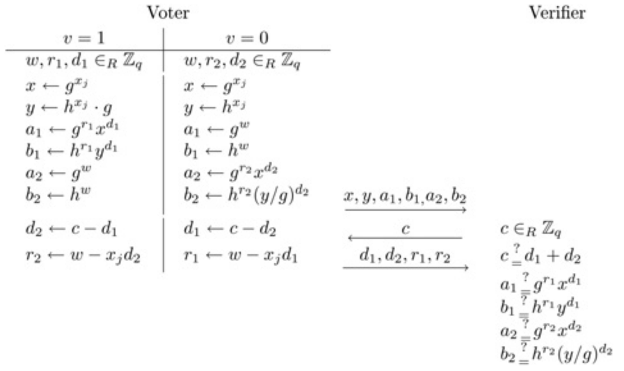
$$\log_g x = \log_h (y/m_0) \vee \log_g x = \log_h (y/m_1)$$

Fig. 1 shows a three-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 [14], 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 one-out-of-two knowledge proof produced in round 2 contains: $(w, a_1, b_1, a_2, b_2, d_1, d_2)$. More details on the one-out-of-$n$ knowledge proof can be found in [4, 13].

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.

## 2.2 Extension to multiple candidates

We can extend the above single-candidate protocol to cater for multiple candidates 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. This is useful as many practical elections involve more than one candidate.



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

**Figure 1** *One-out-of-two 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 $h = g^{y_j}$*

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 two 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 [1, 2]. 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 ZKP showing that $\varrho_i$ is one of $\{g_1, g_2, \ldots, g_k\}$ (using the same CDS technique [13]). 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 $\begin{pmatrix} n+k-1 \\ k-1 \end{pmatrix} = O(n^{k-1})$ (see the combinations with repetitions problem [15]). This is less scalable than the previous $k \times n$, but should still be within the realm of feasible 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 $g_i$'s are appropriately independent, that is, that distinct values of the $c_i$'s do not give rise to the same product. Let $N$ be the number of voters, then we require

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

A preferred way to deal with multiple candidates is to use the method because of Baudron *et al.* [16]: 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 (1) as

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

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, $\sum_i v_i = 2^0 \cdot c_1 + 2^m \cdot c_2 + \cdots + 2^{(k-1)m} \cdot c_k$ 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 decentralised 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 fulfil the following requirements (also see [1, 2]):

*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 proto col. 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 [1, 2] that the Kiayias–Yung and Groth's protocols satisfy the above requirements. We now show our protocol fulfils 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 ZKPs, 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 $P$ $(k \neq i)$ 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 ZKP for the proof of the exponent $x_i$, and a ZKP 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. We refer readers to [10, 12, 13] for security proofs on Schnorr's signature and Cramer *et al.*'s one-out-of-$n$ technique. However, we should note that if these techniques are chosen to realise 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 [12, 13].

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 [17]. 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 fulfils self-tallying. In the protocol any interested party can compute the final tally without external help. This feature is realised by making use of the vanishing property of Proposition 1 (also see [8]). 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 ZKPs in the protocol is to prevent active attacks, ensuring that voters faithfully follow the protocol [1, 2].

## 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 than one ballot will be detected by the other participants. Furthermore, the use of the Cramer, Damgård and Schoenmakers ZKPs 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 decentralised 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 [18]. However in a decentralised 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 [1, 2]. To a large extent, they reflect the decentralised nature of the protocol rather than any weakness in the protocol itself. A centralised voting scheme that employs tallying authorities is certainly more robust and scalable than a decentralised 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 centralised or decentralised depends on the application. For small-scale elections where the DoS attack and voter coercion are usually not of great concern, a decentralised 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 DOS 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 [1, 2]. Therefore in this section, we mainly compare with the Kiayias–Yung and Groth's schemes [1, 2]. Table 2 presents a summary of the comparison results.

The Kiayias–Yung's [1] voting protocol executes in three rounds. 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 ZKP to prevent cheating (see Table 2). More details can be found in [1].

**Table 2** Comparison with past work

| 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** |

Groth [2] aims to improve the system complexity in the Kiayias–Yung's protocol. 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 (the first voter performs one less exponentiation than the rest [1]) – needs to perform four exponentiations in total, and produces four 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 [7]. 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 ZKP. Overall, this kind of efficiency compares very favourably to both the Kiayias–Yung and Groth's protocols (see Table 2).

## 5 Conclusion

In this paper, we demonstrate how to arrange a two-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 show that the protocol is exceptionally efficient. It has only two rounds; in each round, a voter performs constant one exponentiation and produces one ZKP accordingly. This kind of efficiency compares very favourably to past decentralised voting protocols, and is close to the best possible.

## 6 References

[1] KIAYIAS A., YUNG M.: 'Self-tallying elections and perfect ballot secrecy'. Public Key Cryptography'02, 2002 (*LNCS*, **2274**), pp. 141–158

[2] GROTH J: 'Efficient maximal privacy in boardroom voting and anonymous broadcast'. Financial Cryptography'04, 2004 (*LNCS*, **3110**), pp. 90–104

[3] ANDERSON R.J.: 'Security engineering : a guide to building dependable distributed systems' (Wiley, New York, 2008, 2nd edn.)

[4] CRAMER R., GENNARO R., SCHOENMAKERS B.: 'A secure and optimally efficient multi-authority election scheme'. EUROCRYPT'97, May 1997 (*LNCS*, **1233**), pp. 103–118

[5] BENALOH J.C., YUNG M.: 'Distributing the power of a government to enhance the privacy of voters'. Proc. Fifth Annual ACM Symp. on Principles of Distributed Computing, 1986, pp. 52–62

[6] CRAMER R., FRANKLIN M., SCHOENMAKERS B., YUNG M.: 'Multi-authority secret-ballot elections with linear work'. EUROCRYPT'96, 1996 (*LNCS*, **1070**), pp. 72–83

[7] GENNARO R., ISHAI Y., KUSHILEVITZ E., RABIN T.: 'On 2-round secure multiparty computation'. Crypto'02, 2002 (*LNCS*, **2442**), pp. 178–193

[8] HAO F., ZIELIŃSKI P.: 'A 2-round anonymous veto protocol'. Proc. 14th Int. Workshop on Security Protocols, Cambridge, UK, 2006

[9] GOLDREICH O., MICALI S., WIGDERSON A.: 'How to play any mental game or a completeness theorem for protocols with honest majority'. Proc. Nineteenth Annual ACM Conf. on Theory of Computing, 1987, pp. 218–229

[10] STINSON D.: 'Cryptography: theory and practice' (Chapman & Hall/CRC, 2006, 3rd edn.)

[11] LENSTRA A.K., LENSTRA H.W.: 'Algorithms in number theory'. Handbook of Theoretical Computer Science, 1991, pp. 673–715

[12] SCHNORR C.P.: 'Efficient signature generation by smart cards', *J. Cryptol.*, 1991, **4**, (3), pp. 161–174

[13] CRAMER R., DAMGÅRD I., SCHOENMAKERS B.: 'Proofs of partial knowledge and simplified design of witness hiding protocols'. Proc. 14th Annual Int. Cryptology Conf. on Advances in Cryptology, 1994 (*LNCS*, **839**), pp. 174–187

[14] FIAT A., SHAMIR A.: 'How to prove yourself: practical solutions to identification and signature problems'. Crypto'86, 1987 (*LNCS*, **263**), pp. 186–194

[15] STANLEY R.P.: 'Enumerative combinatorics' (Cambridge University Press, 1997)

[16] BAUDRON O., FOUQUE P., POINTCHEVAL D., POUPARD G., STERN J.: 'Practical multi-candidate election system'. Proc. 20th ACM Symp. on Principles of Distributed Computing, 2001, pp. 274–283

[17] BONEH D.: 'The decision Diffie–Hellman problem'. Proc. Third Int. Symp. on Algorithmic Number Theory, 1998 (*LNCS*, **1423**), pp. 48–63

[18] JUELS A., CATALANO D., JAKOBSSON M.: 'Coercion-resistant electronic voting'. WPES'05, 2005, pp. 61–70