IET Research Journals

Journals
The Institution of
Engineering and Technology

PriVeto: A Fully Private Two Round Veto Protocol

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Samiran Bag, Muhammad Ajmal Azad, and Feng Hao

Department of Computer Science, University of Warwick, United Kingdom
** E-mail: {samiran.bag, muhammad.azad, feng.hao}@warwick.ac.uk*

Abstract: Veto is a prerogative to unilaterally overrule a decision. A private veto protocol consists of a number of participants who wish to decide whether or not to veto a particular motion without revealing the individual opinions. Essentially all participants jointly perform a multi-party computation (MPC) on a boolean-OR function where an input of "1" represents veto and "0" represents not veto. In 2006, Hao and Zieliński presented a two round veto protocol named Anonymous Veto network (AV-net), which is exceptionally efficient in terms of the number of rounds, computation and bandwidth usage. However, AV-net has two generic issues: 1) a participant who has submitted a veto can find out whether she is the only one who vetoed; 2) the last participant who submits her input can pre-compute the boolean-OR result before submission, and may amend her input based on that knowledge. These two issues generally apply to any multi-round veto protocol where participants commit their input in the last round. In this paper, we propose a novel solution to address both issues within two rounds, which are the best possible round efficiency for a veto protocol. Our new private veto protocol, called PriVeto, has similar system complexities to AV-net, but it binds participants to their inputs in the very first round, eliminating the possibility of runtime changes to any of the inputs. At the end of the protocol, participants are strictly limited to learning nothing more than the output of the boolean-OR function and their own inputs.

1 Introduction

In 1988, David Chaum first proposed a dining cryptographers (DC) problem [9]. Three cryptographers sit at a table to have dinner together. At the time of payment, they are informed by the waiter that someone, who may either be NSA or one of the cryptographers, has already paid for the dinner. The cryptographers respect each other's right in making a private payment but they want to find out if NSA paid. Essentially they want to securely compute a boolean-OR function with the input "1" representing 'I paid' and "0" representing 'I did not pay'. If the output of the boolean-OR function is "0", it means that NSA must have paid (since no cryptographer has paid); otherwise, NSA has not paid.

To securely find the result, the three cryptographers executed a two-stage protocol called Dining Cryptographers network (DC-net), proposed by Chaum [9]. In the first stage, the three cryptographers establish a pairwise shared secret bit between each other, e.g., by tossing a coin secretly behind a menu. In the second stage, each cryptographer announces a bit which is the XOR of his shared secret bits with the other cryptographers if he did not pay, or the opposite value if he paid. After the two stages, everyone, including any third party observers, can calculate the XOR of all announced bits: if the result is "0", it means NSA has paid for the dinner; and otherwise, NSA has not paid. While the original protocol is described for three participants, it can be easily generalized to accommodate any number of participants more than three.

The aim of the DC-net protocol is to allow multiple parties to securely compute a boolean-OR function, but it has a "collision" problem whereby when an even number of participants use "1" in their inputs, the protocol will falsely return "0". This problem is acknowledged in Chaum's paper [9] as one major limitation of DC-net. Chaum proposed to resolve this problem by "retransmission", but details of the retransmission mechanism are not specified in [9]. Follow-up works by other researchers propose to address this problem by setting up "traps" to probabilistically detect the collision and disruption [5, 15, 25], but they tend to make the DC-net system much more complex. There are also other limitations of DC-net, such as the requirement of pairwise shared keys between participants, which leads to a $O(n^2)$ system complexity for n participants.

To address the limitations of DC-net, Hao and Zieliński [19] proposed an alternative solution called Anonymous Veto network (AV-net). In some sense, AV-net can be seen as a translation of DC-net from a non-cryptographic setting to a cryptographic one. The construction of AV-net echoes a similar idea in DC-net by combining shared secrets in a particular way to achieve a cancellation effect. The major difference between the two is that DC-net only involves sending boolean values, while AV-net is built on public-key cryptography, and it utilizes established cryptographic primitives (e.g., zero-knowledge proofs) to enforce every participant to honestly follow the protocol specification. AV-net only requires 2 rounds of broadcast, which is the best possible round efficiency for a veto protocol. It is also exceptionally efficient in terms of the computation and the bandwidth usage. The system complexity is $O(n)$, as opposed to $O(n^2)$ in DC-net. Among the veto protocols proposed in the literature [7, 17, 21], AV-net is the most efficient in every of the following aspects: the number of rounds, computation, bandwidth and system complexity.

However, the AV-net protocol has two generic issues. First, a participant who has submitted "1" as the input can find the boolean-OR of other participants' inputs. In other words, if a participant vetoes, she can find out if she is the only one who has vetoed or not. Second, the last participant who submitted her input can pre-compute the result of the boolean-OR function, and hence may amend her input accordingly. The latter problem can be alleviated by introducing an intermediate round in which all participants first commit to their inputs, thus eliminating the scope of any runtime change to the value of the input in the subsequent round. However, this remedy increases the round complexity of the protocol to 3 rounds.

In this paper we address these issues by proposing a new private veto protocol called "PriVeto". Same as AV-net, the proposed protocol has only two rounds. It binds participants to their inputs in the very first round in a way that no participant will be able to find the boolean-OR of all the inputs until the second round ends. The ciphertext generated by a participant in the second round depends upon the ciphertexts generated by other participants in the first round. By then all the ciphertexts coming from the first round of the protocol have already been committed (e.g., published on a public bulletin board), which eliminates the possibility of amendment by any participant. Our protocol strictly limits each participant to learn nothing more than the output of the boolean-OR function and their own private inputs. The privacy of each individual input is guaranteed even under the extreme case that all other participants collude against a single victim (i.e., a full-collusion attack). By contract, in AV-net, in the extreme scenario of full-collusion, the privacy of the individual input can no longer be guaranteed.

The rest of the paper is organized as follows. Section 2 discusses approaches that have been proposed for securely computing the boolean-OR function. Section 3 describes our proposed approach for anonymous veto. Section 4 presents the security proofs of the proposed protocol. Section 5 presents the system complexity of the protocol. Finally, Section 6 concludes the paper.

2 Related Work

2.1 Multiparty computation

The DC-net problem can be seen as a secure multi-party computation (MPC) problem for computing a boolean-OR function while preserving the privacy of each individual input. A general MPC problem is to compute a function $f(x_1, x_2, \ldots, x_n)$, on secret inputs x_i provided by n participants respectively. At the end of the protocol, each participant would learn the output of the function f over all the secret inputs, but they do not learn anything more than the output and their own inputs.

The first general protocol for the secure two-party computation was proposed by Yao [26]. The Fairplay [22] system presents an implementation of Yaos protocol. This system generates the circuits by representing the computed functions of each participating party as high level languages (Secure Function Definition Language (SFDL) and Secure Hardware Definition Language (SHDL)). A number of methods have been proposed [3, 4, 10, 14] to generalize Yao's two-party computation method to multi-party computation (MPC). The MPC protocols of Goldreich et al. [14] and Assaf et al. [3] are based on boolean circuits that represent the function. In theory, the general MPC solutions can work with an arbitrary polynomial-time computable function assuming that the majority of the players are honest [10]. However, general MPC techniques are rather inefficient for computing specific functions such as boolean-OR [20]. Furthermore, they typically require pairwise secrete authenticated channels between players in addition to a public authenticated channel. The establishment of pairwise secure channels is complex to realize in practice [19]. For these reasons, researchers normally propose specific MPC solutions to efficiently solve specific problems, e.g., e-voting [18], secure network statistics aggregation [1, 2, 8, 16], secure function computation [12] and secure auctions [23] etc.

2.2 Review of AV-net

There have already been a number of protocols proposed to perform secure multi-party computation on a boolean-OR function [7, 9, 17, 19, 21]. Among these solutions, AV-net, due to Hao and Zieliński in 2006 [19], is known to be the most efficient in terms of the number of rounds, computation and bandwidth.

AV-net does not require any pairwise secret and authenticated channels. It only assumes an authenticated public channel, which is typically implemented by a public bulletin board [17] where participants publish data with a digital signature to prove authenticity. Let G denote a finite cyclic group of prime order p in which the Decision Diffie-Hellman (DDH) problem is intractable. Let g be a generator in G. Assume there are n participants. The protocol proceeds in two rounds. (All computations are modular, and we omit the modulus for simplicity.)

Round 1: Every participant P_i chooses a random secret $x_i \in_R \mathbb{Z}_p$ and publishes $A_i = g^{x_i}$ and a zero-knowledge proof (Schnorr signature [24]) for proving the knowledge of x_i. When this round finishes, each participant P_i computes

$$B_i = \prod_{j=1}^{i-1} A_j / \prod_{j=i+1}^{n} A_j$$

We can also represent the above as $B_i = g^{y_i}$ where $y_i = \sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j$.

Round 2: Every participant P_i publishes $C_i = B_i^{c_i}$, where c_i is either x_i for the input "1" or a random value $r_i \in_R \mathbb{Z}_p$ for the input "0", together with a zero-knowledge proof (Schnorr signature [24]) to prove the knowledge of c_i.

After Round 2, every participant computes $\prod_i C_i$. If no one vetoes, we have $\prod_i C_i = 1$; otherwise we have $\prod_i C_i \neq 1$. Note that when no one vetoes, all the random factors x_i vanish because $\sum_i x_i y_i = 0$ [19].

Although AV-net is exceptionally efficient, it has two generic issues. The first issue, as highlighted in the original AV-net paper [19], is that a participant P_i who has sent the veto message ("1") in the second round will be able to find out if she is the only one who vetoed at the end of the second round. To learn this, she just needs to substitute her input in Round 2 with "0" and re-calculate the boolean-OR function. Note that she can do this non-interactively without the cooperation from other participants. Thus she can learn the boolean-OR of all other inputs, which is equivalent to knowing if she is the only one who has submitted a veto. As only the vetoer can learn this extra-bit information, we call this the *vetoer's advantage*. The second issue is that the participant who sends data at last in Round 2 can pre-compute the boolean-OR, thus knowing the outcome before others, and she might alter her input based on that knowledge. We call this the *last player's advantage*. It seems inevitable that the participant who sends data at last will learn the result before others, but the run-time change of the input should be prevented.

These two issues are generally applicable to other veto protocols [7, 17, 21] where participants send the input in the last round. The last player's advantage can be alleviated by requiring all participants to commit their input first, so they cannot change it later, however that will require an additional round of sending commitments.

3 The PriVeto Scheme

In this section, we present a novel solution to address both issues in AV-net without increasing the number of rounds. Our new protocol is called PriVeto. For any integer $n \geq 1$, we define $[n]$ to be the set $\{1, 2, \ldots, n\}$.

3.1 Setup

Let there be n participants identified as $V_1, V_2, \ldots, V_n$. Let id_i, $i \in [n]$, denote the unique identities of the participants. Each participant $V_i; i \in [n]$ has a secret input bit $v_i \in \{0, 1\}$. The n participants want to securely compute the value of $T = \bigvee_{i=1}^{n} v_i$. Our protocol uses a finite multiplicative group G of p elements, p being a prime number. The Decisional Diffie Hellman problem is assumed to be intractable in G. Let g and $\tilde{g}$ be two random generators of G, whose discrete logarithm relationship is unknown. Let, $Hash_1, Hash_2 : \{0,1\}^* \to \mathbb{Z}_p$ be two secure one-way hash functions. Hence, the setup function takes as input the security parameter λ and generates the following system parameters.

$$(G, p, g, \tilde{g}, Hash_1, Hash_2) \leftarrow Setup(1^\lambda)$$

We assume there is an authenticated public channel, which can be realized by a public bulletin board as in [7, 17, 19, 21]. The bulletin board is readable to all, and writable only to authenticated participants with a digital signature to prove the data authenticity.

3.2 Protocol

The PriVeto protocol works in two rounds as below:

Round 1: Each participant $V_i : i \in [n]$ chooses $a_i, z_i \in_R \mathbb{Z}_p$ and publishes on the bulletin board $Z_i = g^{z_i}, \phi_i = g^{a_i z_i}$. V_i also computes π_i and π_i^* where π_i is a non-interactive zero-knowledge (NIZK) proof [24] to prove the knowledge of $z_i = \log_g Z_i$, and π_i^* is a NIZK proof [11] to prove the knowledge of $a_i = \log_{Z_i} \phi_i$. That is

$$\pi_i = NIZK\left[z_i : g, Z_i = g^{z_i}\right]$$

and

$$\pi_i^* = NIZK\left[a_i : g, Z_i = g^{z_i}, \phi = g^{a_i z_i}\right]$$

Every participant $V_i : i \in [n]$ posts these 2 NIZK proofs on the bulletin board. V_i also posts an intermediate ballot b_i computed as follows:

$$b_i = \begin{cases} g^{a_i} \text{ if } v_i = 0 \\ g^{a_i} g_i \text{ if } v_i = 1 \end{cases}$$

Here, $g_i = \tilde{g}^{r_i}$ where $r_i = Hash_1(id_i||Z_i||\phi_i)$. Hence, both r_i and g_i can be computed by anyone after fetching all the three inputs (i.e. id_i, Z_i, ϕ_i) from the bulletin board. The participant V_i also computes NIZK proof $\tilde{\pi}_i$ of the fact that b_i is either g^{a_i} or $g^{a_i} g_i$. This is an OR proof of two statements with the first statement being $\log_g b_i = \log_{Z_i} \phi_i$ and the second statement being $\log_g b_i / g_i = \log_{Z_i} \phi_i$ (see Appendix for a detailed construction of this NIZK proof). We express $\tilde{\pi}_i$ as follows.

$$NIZK\left[a_i, z_i : g, g_i, Z_i = g^{z_i}, \phi_i = g^{a_i z_i}, b_i \in \{g^{a_i}, g^{a_i} g_i\}\right]$$

Each participant $V_i : i \in [n]$ posts b_i and $\tilde{\pi}_i$ on the bulletin board.

Round 2: Each participant $V_i : i \in [n]$ copies the intermediate ballot $Z_j, \phi_j, \pi_j, \pi_j^*, b_j$ for all $j \in [n]$ from the bulletin board. Then she computes

$$\tilde{a}_j = Hash_2(id_j||Z_j||\phi_j||\pi_j||\pi_j^*||b_j), \forall j \in [n]$$

Each participant $V_i : i \in [n]$ computes a final ballot B_i as follows:

$$B_i = \tilde{B}_i^{(a_i + \tilde{a}_i)}$$

Here $\tilde{B}_i = \prod_{j=1}^{i-1} g^{\tilde{a}_j} b_j / \prod_{j=i+1}^{n} g^{\tilde{a}_j} b_j$. V_i also computes a NIZK proof Π_i of wellformedness of B_i. This NIZK proof proves that B_i is indeed equal to $(\tilde{B}_i)^{a_i + \tilde{a}_i}$, given the value of $\tilde{B}_i$,g^{a_i} and $\tilde{a}_i$. Essentially, this is an equality proof that $\log_{\tilde{B}_i} B_i / \tilde{B}_i^{\tilde{a}_i} = \log_{Z_i} \phi_i$ (see Appendix for more details on the construction of this NIZK proof). V_i posts B_i and Π_i on the bulletin board.

Once, every participant has posted the ballot in the second round anyone can compute $\tilde{T} = \prod_{i=1}^{n} B_i$. Let us assume $\hat{a}_i = a_i + \tilde{a}_i$. It is easy to see that

$$\tilde{T} = g^{\sum_{i=1}^{n} \sum_{j=1}^{i-1} \hat{a}_i \hat{a}_j - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \hat{a}_i \hat{a}_j} \prod_{i=1}^{n} (\prod_{j=1}^{i-1} g_j^{v_j} / \prod_{j=i+1}^{n} g_j^{v_j})^{\hat{a}_i}$$

It can be proved that $\sum_{i=1}^{n} \sum_{j=1}^{i-1} \hat{a}_i \hat{a}_j - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \hat{a}_i \hat{a}_j = 0$. Readers are referred to [19] for a proof of the above fact. Now, $\tilde{T} = \prod_{i=1}^{n} (\prod_{j=1}^{i-1} g_j^{v_j} / \prod_{j=i+1}^{n} g_j^{v_j})^{\hat{a}_i}$. If $v_i = 0, \forall i \in [n]$, then $\tilde{T} = 1$. Let us assume that $v_k = 1$ for some $k \in [n]$. As such, $\tilde{T} = g_k^{\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j} * A$. Here

$$A = \prod_{i \in [n] \setminus \{k\}} (g_i^{v_i})^{\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j}$$

Note, that g_k is independent of A and $g_k^{\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j}$ is random. Hence, $\tilde{T}$ is a random element in G (and is not 1 with overwhelming probability). Thus we get:

$$\tilde{T} \begin{cases} = 1, \text{ with probability 1, if } T = \bigvee_{i=1}^{n} v_i = 0 \\ \neq 1, \text{ with overwhelming probability, if } T = \bigvee_{i=1}^{n} v_i = 1 \end{cases}$$

This way everyone can find T, the boolean-OR of all n input bits by computing $\tilde{T} = \prod_{i=1}^{n} B_i$.

4 Analysis

4.1 Completeness of the Scheme

We show that our 2-round PriVeto protocol is correct, that is it correctly computes the boolean-OR of all input bits. It is worth noticing that when $v_i = 0$, for all $i \in [n]$, then $\tilde{T} = \prod_{i=1}^{n} B_i = g^{\sum_{i=1}^{n} \hat{a}_i \sum_{j=1}^{i-1} \hat{a}_j - \sum_{i=1}^{n} \hat{a}_i \sum_{j=i+1}^{n} \hat{a}_j} = 1$. Now, we need to show that if there exists at least one $k \in [n]$, such that $v_k = 1$, then $\hat{T}$ will be random. For this purpose, let us assume $v_k = 1$ for some $k \in [n]$. Hence, $\hat{T} = \prod_{i=1}^{n} B_i = g_k^{\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j} * \prod_{i \in [n] \setminus \{k\}} (g_i^{v_i})^{\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j}$. Our algorithm will output $\bigvee_{i=1}^{n} v_i = 1$ if and only if $\tilde{T} \neq 1$. So, for the scheme to be correct, we need to show that $\tilde{T} \neq 1$ with overwhelming probability. We know that, $g_i = \tilde{g}^{r_i}, \forall i \in [n]$. Therefore, $\tilde{T} = \tilde{g}^{r_k (\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j)} * \prod_{i \in [n] \setminus \{k\}} \tilde{g}^{r_i v_i (\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j)}$. If $\tilde{T} = 1$, then $r_k (\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j) + \sum_{i \in [n] \setminus \{k\}} r_i v_i (\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j) = 0$. So, $r_k = -\frac{\sum_{i \in [n] \setminus \{k\}} r_i v_i (\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j)}{(\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j)}$. That is,

$$Hash_1(id_k||Z_k||\phi_k) = -\frac{\sum_{i \in [n] \setminus \{k\}} r_i v_i (\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j)}{(\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j)}$$

Here, $\hat{a}_i = a_i + \tilde{a}_i = a_i + Hash_2(id_i||Z_i||\phi_i||\pi_i||\pi_i^*||b_i)$.

Since, $Hash_1(\cdot)$ is a secure hash function, $Pr\left[Hash_1(id_k||Z_k||\phi_k) = -\frac{\sum_{i \in [n] \setminus \{k\}} r_i v_i (\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j)}{(\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j)}\right] \leq negl(\lambda)$. That is, $Pr\left[Hash_1(id_k||Z_k||\phi_k) \neq -\frac{\sum_{i \in [n] \setminus \{k\}} r_i v_i (\sum_{j=i+1}^{n} \hat{a}_j - \sum_{j=1}^{i-1} \hat{a}_j)}{(\sum_{j=k+1}^{n} \hat{a}_j - \sum_{j=1}^{k-1} \hat{a}_j)}\right] \geq 1 - negl(\lambda)$.

Hence, $Pr\left[\tilde{T} \neq 1\right] \geq 1 - negl(\lambda)$. Hence, we get:

$$Pr\left[\tilde{T} = 1 \middle| \bigvee_{i=1}^{n} v_i = 0\right] = 1$$

$$Pr\left[\tilde{T} \neq 1 \middle| \bigvee_{i=1}^{n} v_i = 1\right] \geq 1 - negl(\lambda)$$

Thus, the scheme is correct.

4.2 Security assumptions

Assumption 1. *Given g, g^a, g^b and a challenge $\Omega \in \{g^{ab}, R\}$, where $R \xleftarrow{\$} G$, it is hard to find whether $\Omega = g^{ab}$ or $\Omega = R$. This assumption is known as the Decisional Diffie-Hellman (DDH) assumption.*

Assumption 2. *Given g, g^b, g^{ab} and a challenge $\Omega \in \{g^a, R\}$, where $R \xleftarrow{\$} G$, it is hard to find whether $\Omega = g^a$ or $\Omega = R$.*

Lemma 1. *Assumption 1 implies assumption 2.*

Proof: We shall show that if there exists an adversary $\mathcal{A}$ against assumption 2, then the same can be used to construct another adversary $\mathcal{A}'$ against assumption 1. The adversary $\mathcal{A}'$ works as follows: It receives as input g, g^a, g^b and a challenge $\Omega \in \{g^{ab}, R\}$. It sends (g, g^b, Ω, g^a) to $\mathcal{A}$. Note that, if $\Omega = g^{ab}$, then the tuple will be of the form (g, g^b, g^{ab}, g^a). Else the tuple will be (g, g^b, g^{cb}, R) where $c = \log_g(\Omega)/b$. Now, if $\mathcal{A}$ can distinguish between them, $\mathcal{A}'$ will be able to identify Ω. □

Assumption 3. *Given $g, g^b, g^{ab}, g^c, g^{ca}$ and a challenge $\Omega \in \{g^a, R\}$, it is hard to find whether $\Omega = g^a$ or $\Omega = R$.*

Lemma 2. *Assumption* 2 *implies assumption* 3.

Proof: We show that if there exists an adversary $\mathcal{A}$ against assumption 3, it could be used to construct another adversary $\mathcal{A}'$ against assumption 2. $\mathcal{A}'$ receives as input g, g^b, g^{ab} and a challenge $\Omega \in \{g^a, R\}$. It chooses random $x \in \mathbb{Z}_p$ and computes $g^c = (g^b)^x$ and $g^{ca} = (g^{ab})^x$. Then $\mathcal{A}'$ sends $g, g^b, g^{ab}, g^c, g^{ca}$ and Ω to $\mathcal{A}$. If $\mathcal{A}$ can identify the correct value of Ω, then so can $\mathcal{A}'$. □

Assumption 4. *Given $m, d \in \mathbb{Z}_p, g, g^b, g^{ab}, g^c, g^{ac}$ and a challenge $\Omega \in \{g^a, g^a(g^c/g^{1/d})^m\}$, it is hard to find whether $\Omega = g^a$ or $\Omega = g^a(g^c/g^{1/d})^m$.*

Lemma 3. *Assumption* 3 *implies assumption* 4.

Proof: According to assumption 3, $(m, d, g, g^b, g^{ab}, g^c, g^{ac}, g^a) \overset{c}{\approx} (m, d, g, g^b, g^{ab}, g^c, g^{ac}, R) \overset{c}{\approx} (m, d, g, g^b, g^{ab}, g^c, g^{ac}, R * (g^c/g^{1/d})^m) \overset{c}{\approx} (m, d, g, g^b, g^{ab}, g^c, g^{ac}, g^a(g^c/g^{1/d})^m)$ □

4.3 Security analysis of the PriVeto scheme

4.3.1 The security model: Now, we show that our scheme is secure against a PPT adversary who can compromise all but one participant. If the scheme can protect the privacy of the sole uncompromised participant under full collusion scenario, then the scheme could be called secure. We show that if the Decisional Diffie Hellman assumption holds in the group G, the scheme will be secure. In order for proving our security claim, we choose the following security model. In this model, there are two entities: the challenger and the adversary who together enact the full collusion scenario. The setup function generates the public parameters $(G, p, g, \tilde{g}, n)$. The challenger creates one participant with a random id $k \in [n]$. The adversary creates $n-1$ participants with ids in the set $[n] \setminus \{k\}$. The adversary controls these $n-1$ colluding participants. The challenger chooses random $Z_k, \phi_k \in G$, and generates(or simulates) the NIZK proofs $\pi_k = NIZK\left[z_k : g, Z_k = g^{z_k}\right]$ and $\pi_k^* = NIZK\left[a_k : g, Z_k, \phi_i = Z_k^{a_k}\right]$. The challenger sends $G, p, g, \tilde{g}, n, Z_k, \phi_k, \pi_k, \pi_k^*$ to the adversary. The challenger replies to all queries to the hash oracles $\tilde{H}$ and H made by the adversary. The adversary outputs a_i, z_i, π_i, π_i^*. Since, π_i and π_i^* are proofs of knowledge of z_i, and a_i, the adversary has to invoke the prover algorithm with these two witnesses (a_i, z_i) in order to generate π_i and π_i^*. The challenger can get to learn these values when the prover algorithm is called by the adversary, and can match them with the values of Z_i and ϕ_i sent to the challenger by the adversary. If any of the NIZK proofs returned by the adversary is not properly generated, the challenger aborts, and the adversary fails. The adversary also returns a set of inputs $\{v_i : i \in [n] \setminus \{k\}\}$ which represent the inputs of the $n-1$ colluding participants. It can be observed that if all the $n-1$ inputs of the colluding participants are 0, the output of our protocol (boolean OR of all inputs) will be equal to the input of the honest participant controlled by the challenger. Thus, in this model we add a condition to make it mandatory for at least one colluding participant to have a 1-bit as input. Note that, the adversary may choose the inputs of other colluding participants as she wishes. Thus, there is no upper bound to the value of $\kappa = |\{i : i \in [n] \setminus \{k\}, v_i = 1\}|$. So, $\kappa \in [n-1]$. Since, $\kappa \geq 1$, there must be at least one $i \in [n] \setminus \{k\}$, satisfying $v_i = 1$. We assume that it is α. If there is no such $\alpha \in [n] \setminus \{k\}$, such that $v_\alpha = 1$, then the challenger aborts and the adversary fails. Else, the challenger computes c_0 and c_1 which are the intermediate ballots corresponding to the two cases $v_k = 0$, and $v_k = 1$ respectively. Then the challenger randomly selects a bit $e \in \{0, 1\}$, and sets the intermediate ballot $b_k = c_e$. The challenger computes the final ballot B_k, and generates (or simulates) the NIZK proofs of well-formednesses of the intermediate and the final ballot, and sends the intermediate and the final ballot along with the NIZK proofs to the adversary. The adversary wins if she can distinguish between the two cases: $e = 0$ and $e = 1$.

$Exp_{\mathcal{D}}^{DDHV}(\lambda)$
$(G, p, g) \leftarrow Setup(\lambda)$ $\hat{g} \in_R G$ $d \leftarrow_R \mathbb{Z}_p$ $a, z \leftarrow_R \mathbb{Z}_p$ $m \leftarrow \mathbb{Z}_p$ $Z \leftarrow g^z, \phi \leftarrow g^{az}$ $\pi \leftarrow NIZK[z : g, Z = g^z]$ $\pi^* \leftarrow NIZK[a : g, Z, \phi = Z^a]$ $B \leftarrow \hat{g}^a$ $c_0 \leftarrow g^a$ $c_1 \leftarrow g^a(\hat{g}/g^{1/d})^m$ $e \overset{\$}{\leftarrow} \{0, 1\}$ $e' \leftarrow \mathcal{D}(G, p, q, g, Z, \phi, c_e, \hat{g}, B, d, m, \pi, \pi^*)$ return$(e = e')$

Fig. 1: Description of the security experiment $Exp_{\mathcal{D}}^{DDHV}(\lambda)$.

4.3.2 Security experiments: Figure 2 describe the security experiment $Exp_{\mathcal{B}}^{POBA}(\lambda)$. This security experiment characterize the full-collusion scenario, where the adversary has colluded with $n-1$ participants with the intention to learn the private input of the sole honest participant V_k for some $k \in [n]$. $b_i : i \in [n] \setminus \{k\}$ represent the intermediate ballots of each of the compromised participant. c_e is the intermediate ballot of the honest participant V_k. B_k is the final ballot of V_k. The adversary's job is to identify e. We define the advantage of the adversary $\mathcal{B}$ as below;

$$Adv_{\mathcal{B}}^{POBA}(\lambda) = 2 * P[Exp_{\mathcal{B}}^{POBA}(\lambda) = 1] - 1$$

In Figure 1, we present $Exp_{\mathcal{D}}^{DDHV}(\lambda)$. This is the security game that emulates an attack against the Decisional Diffie Hellman assumption. The adversary $\mathcal{D}$ receives as input $d, m \in \mathbb{Z}_p, \hat{g}, Z = g^z, \phi = g^{az}, B = \hat{g}^a, \pi, \pi^*$ a challenge $c_e, e \in \{0, 1\}$, where $c_0 = g^a$ and $c_1 = g^a(\hat{g}/g^{1/d})^m$. π and π^* are two NIZK proofs that prove that the prover knows the values of $\log_g Z$ and $\log_Z \phi$ respectively. The NIZK proofs do not provide any useful information other than what they are intended to prove. The DDH adversary $\mathcal{D}$ has to find e. Since the NIZK proof systems used in the experiment are secure, we may define the advantage of the adversary $\mathcal{D}$ as below:

$$Adv_{\mathcal{D}}^{DDHV}(\lambda) = 2 * P[Exp_{\mathcal{D}}^{DDHV}(\lambda) = 1] - 1$$

From the discussion in section 4.2, we can say that if the DDH assumption holds in G, then

$$Adv_{\mathcal{D}}^{DDHV}(\lambda) \leq negl(\lambda)$$

We now present a series of games below.

Game 1: This game is the experiment $Exp_{\mathcal{B}}^{POBA}(\lambda)$ of Figure 2.

Game 2: This game same is the experiment $EXp_{\mathcal{B}'}^{POBB}(\lambda)$ of Figure 3. The advantage of the adversary $\mathcal{B}'$ is defined as below:

$$Adv_{\mathcal{B}'}^{POBB}(\lambda) = 2 * P[Exp_{\mathcal{B}'}^{POBB}(\lambda) = 1] - 1$$

The difference between Game 1 and Game 2 is that in Game 2, we substitute the two hash functions viz. $Hash_1$ and $Hash_2$ by two random functions $\mathcal{RO}_1$ and $\mathcal{RO}_2$ respectively. The difference between the advantages of the adversaries in game 1 and 2 is negligible if the two hash functions are secure. That is if $Hash_1$ and $Hash_2$ are secure,

$$\left|Adv_{\mathcal{B}}^{POBA}(\lambda) - Adv_{\mathcal{B}'}^{POBB}(\lambda)\right| \leq negl(\lambda)$$

Game 3: This game same as the experiment $EXp_{\mathcal{A}}^{POBC}(\lambda)$ of Figure 4. The advantage of the adversary $\mathcal{A}$ is defined as below:

$$Adv_{\mathcal{A}}^{POBC}(\lambda) = 2 * P[Exp_{\mathcal{A}}^{POBC}(\lambda) = 1] - 1$$

The difference between game 3 and game 2 is that in game 3, the query-format to the second oracle H is of the form $id_i, a_i, z_i, \pi_i, \pi_i^*$ as opposed to its format game 2 in which the query-input is of the form $id_i, g^{z_i}, g^{a_i z_i}, \pi_i, \pi_i^*$. Note that when the adversary of game 2 makes a query of the form $id_i, g^{z_i}, g^{a_i z_i}, \pi_i, \pi_i^*$, she has to know the values of a_i and z_i, as otherwise computing π_i and π_i^* would amount to breaking the security of the non-interactive zero knowledge proof system used to compute π_i or π_i^* which are proofs of knowledge [11, 24]. Hence, the difference between the distinguishing advantages of an adversary in game 2 and game 3 is negligible, that is,

$$\left|Adv_{\mathcal{B}'}^{POBB}(\lambda) - Adv_{\mathcal{A}}^{POBC}(\lambda)\right| \leq negl(\lambda)$$

From the above discussion, we can infer that

$$\left|Adv_{\mathcal{B}}^{POBA}(\lambda) - Adv_{\mathcal{A}}^{POBC}(\lambda)\right| \leq negl(\lambda)$$

That is:

$$Adv_{\mathcal{B}}^{POBA}(\lambda) \leq Adv_{\mathcal{A}}^{POBC}(\lambda) + negl(\lambda)$$

4.3.3 Security proof: Now, we shall show that our PriVeto protocol is secure against full collusion. For this purpose, we first state and prove the following theorem.

Theorem 1. $Adv_{\mathcal{A}}^{POBC}(\lambda) \leq poly(\lambda) * Adv_{\mathcal{D}}^{DDHV}(\lambda)$.

Proof: We show that if there exists an adversary $\mathcal{A}$ against the $POBC$ experiment, then it could be used to construct the $DDHV$ adversary $\mathcal{D}$. $\mathcal{D}$ receives as input $d, m, g, \hat{g}, Z = g^z, \phi = g^{az}, \pi, \pi^*, c_b \in \{g^a, g^a(\hat{g}/g^{1/d})^m\}$ and $B = \hat{g}^a$. π and π^* are the two NIZK proofs that prove that the values of a and z are known to the prover. $\mathcal{D}$ now simulates $EXP_{\mathcal{A}}^{POBC}(\lambda)$ with the help of $\mathcal{A}$. $\mathcal{D}$ answers $\mathcal{A}$'s queries to the oracles $\tilde{H}$ and $\mathcal{H}$. Let Q be the maximum number of queries $\mathcal{A}$ makes to the oracle $\tilde{H}, \mathcal{H}$. Obviously, $Q \in poly(\lambda)$.

Step 1: $\mathcal{D}$ computes $\tilde{g} = \hat{g}/g^{1/d}$. $\mathcal{D}$ selects $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_{k-1}, \hat{v}_{k+1}, \ldots, \hat{v}_n$ randomly such that $\hat{v}_i = 1$ holds at least for one value of $i \in [n] \setminus \{k\}$. $\mathcal{D}$ implicitly sets $Z_k = g^{z_k} = Z = g^z$ and $\phi_k = g^{a_k z_k} = \phi = g^{az}$. Then $\mathcal{D}$ sends $G, p, g, \tilde{g}, Z_k, \phi_k, \pi, \pi^*$ to $\mathcal{A}$. $\mathcal{A}$ outputs the values of $\alpha \in [n] \setminus \{k\}$ $(a_i, z_i) : i \in [n] \setminus \{k\}$ and $v_i : i \in [n] \setminus \{k, \alpha\}$. It should be noted that $v_\alpha = 1$. During this process, $\mathcal{A}$ may also make queries to both the oracles $\tilde{H}$ and $\mathcal{H}$. These queries are answered by $\mathcal{D}$ as follows:

- If the query is to the oracle $\tilde{H}$ and the query input can be parsed as $id_i, \hat{Z}_i, \hat{\phi}_i$, for some $i \in [n] \setminus \{k\}$, then sample a random $m_i \in \mathbb{Z}_p$ and return it. Register $(id_i, \hat{Z}_i, \hat{\phi}_i, m_i)$ in the log $\hat{L}_i$.
- If the query is to the oracle $\tilde{H}$ and input can be parsed as id_k, Z_k, ϕ_k, and return m.
- Else if the query is to the oracle $\tilde{H}$ return a random element from $\mathbb{Z}_p$.

All queries to the oracle $\mathcal{H}$ can be answered by $\mathcal{D}$ as follows:

- If the query is to the oracle $\mathcal{H}$, and the input can be parsed as $id_i, a_i, z_i, \pi_i, \pi_i^*, b_i$ for some $i \in [n] \setminus \{k\}$, $b_i \in \{g^{a_i}, g^{a_i} \tilde{g}^{\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})}\}$(where $\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$ has already been queried by $\mathcal{A}$) and if there exists a $j \in [n] \setminus \{k\}$ such that no query of the form $\mathcal{H}(id_j, a_j, z_j, \pi_j, \pi_j^*, b_j)$ has ever been made, then return a random $\tilde{a}_i \in \mathbb{Z}_p$. Register $id_i, a_i, z_i, \pi_i, \pi_i^*, b_i, \tilde{a}_i$ on the log L_i.
- Else, if the query is to the oracle $\mathcal{H}$, and the input can be parsed as $id_i, a_i, z_i, \pi_i, \pi_i^*, b_i$ for some $i \in [n] \setminus \{k\}$, $b_i \in \{g^{a_i}, g^{a_i} \tilde{g}^{\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})}\}$(where $\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$ has already been queried by $\mathcal{A}$) and for all $j \in [n] \setminus \{k\}$, the log L_j contains at least one entry of the form $id_j, a_j, z_j, \pi_j, \pi_j^*, b_j, \hat{a}_j$, then for all $j \in [n] \setminus \{k, i\}$ select an entry $id_j, a_j, z_j, \pi_j, \pi_j^*, b_j, \tilde{a}_j$ from each log L_j and compute

$$X = \begin{cases} -U_1 - a_i + \frac{F}{d} \text{ if } i < k \\ U_2 - a_i - \frac{F}{d} \text{ if } i > k \end{cases}$$

Here,

$$U_1 = \sum_{j=1}^{i-1}(a_j + \tilde{a}_j) + \sum_{j=i+1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{n}(a_j + \tilde{a}_j)$$

$$U_2 = \sum_{j=1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{i-1}(a_j + \tilde{a}_j) - \sum_{j=i+1}^{n}(a_j + \tilde{a}_j)$$

and

$$F = \sum_{j=1}^{k-1} \tilde{H}(id_j, g^{z_j}, g^{a_j z_j}) * \hat{v}_j - \sum_{j=k+1}^{n} \tilde{H}(id_j, g^{z_j}, g^{a_j z_j}) * \hat{v}_j$$

Sample a random $\mathfrak{R} \in_R \mathbb{Z}_p$. Return $\tilde{a}_i$, where:

$$\tilde{a}_i = \begin{cases} X \text{ with probability } 1/Q \\ \mathfrak{R} \text{ with probability } 1 - 1/Q \end{cases}$$

Enter $id_i, a_i, z_i, \pi_i, \pi_i^*, b_i, \hat{a}_i$ on the log L_i.

- Else return a random element from $\mathbb{Z}_p$.

Step 2: Now, $\mathcal{A}$ outputs the index α and (a_i, z_i, v_i) for all $i \in [n] \setminus \{k\}$, where $v_\alpha = 1$. Now, if there exists $j \in [n] \setminus \{k\}$ such that $\hat{v}_j \neq v_j$, $\mathcal{D}$ aborts.

Step 3: Else $\mathcal{D}$ computes the intermediate ballot b_i for all $i \in [n] \setminus \{k\}$ by invoking its internal oracle $\mathcal{O}$ with inputs id_i, a_i, z_i, v_i. If for all $i \in [n] \setminus \{k\}$, satisfying $v_i = 1$ there is an entry $(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i, \tilde{a}_i)$ in L_i such that $m_i = \tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$ has been queried, then if $\sum_{j=1}^{k-1} m_j v_j - \sum_{j=k+1}^{n} m_j v_j \neq d * (\sum_{j=1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{n}(a_j + \tilde{a}_j))$ abort.

Step 4: Else if for all $i \in [n] \setminus \{k\}$, satisfying $v_i = 1$, there is an entry $(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i, \tilde{a}_i)$ in L_i, and there exists at least one

$Exp_{\mathcal{B}}^{POBA}(\lambda)$

$(G, p, g, \tilde{g}, n) \leftarrow Setup(\lambda)$
$a_k, z_k \in \mathbb{Z}_p$
$(\{(a_i, z_i, \pi_i, \pi_i^*) : i \in [n] \setminus \{k\}\}, \alpha, \{v_i : i \in [n] \setminus \{k, \alpha\}\}, v_\alpha = 1, aux) \leftarrow$
$\mathcal{B}^{\tilde{H},H}(G, p, g, \tilde{g}, g^{z_k}, g^{a_k z_k}, \pi_k, \pi_k^*)$
$b_i \leftarrow \mathcal{O}^{\tilde{H}}(id_i, a_i, z_i, v_i) : i \in [n] \setminus \{k\}$
$c_0 \leftarrow \mathcal{O}^{\tilde{H}}(id_k, a_k, z_k, 0)$
$c_1 \leftarrow \mathcal{O}^{\tilde{H}}(id_k, a_k, z_k, 1)$
$\tilde{a}_i \leftarrow \mathcal{H}(id_i, Z_i, \phi_i, \pi_i, \pi_i^*, b_i) : \forall i \in [n] \setminus \{k\}$
$e \xleftarrow{\$} \{0, 1\}$
$\tilde{a}_k \leftarrow \mathcal{H}(id_k, Z_k, \phi_k, \pi_k, \pi_k^*, c_e)$
$B_k \leftarrow \left(\prod_{j=1}^{k-1} g^{\tilde{a}_j} b_j / \prod_{j=k+1}^{n} g^{\tilde{a}_j} b_j\right)^{a_k + \tilde{a}_k}$
$e' \leftarrow \mathcal{B}^{\tilde{H},\mathcal{H}}(aux, c_e, B_k, \tilde{\pi}_k, \Pi_k)$
return$(e = e')$

$\mathcal{O}^{\tilde{H}}(id_i, a_i, z_i, v_i)$	$\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$	$\mathcal{H}(id_i, g^{z_i}, g^{a_i z_i}, \pi_i, \pi_i^*, b_i)$
$m_i \leftarrow \tilde{H}(id_i \vert\vert g^{z_i} \vert\vert g^{a_i z_i})$ return $g^{a_i} \tilde{g}^{m_i v_i}$	return $Hash_1(id_i \vert\vert g^{z_i} \vert\vert g^{a_i z_i})$	return $Hash_2(id_i \vert\vert g^{z_i} \vert\vert g^{a_i z_i} \vert\vert \pi_i \vert\vert \pi_i^* \vert\vert b_i)$

e

Fig. 2: Description of the security experiment $Exp_{\mathcal{B}}^{POBA}(\lambda)$.

$j \in [n] \setminus \{k\}$, satisfying $v_j = 1$ and $\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$ has not yet been queried, then let

$$\mathbb{S} = \{j : v_j = 1, \tilde{H}(id_j, g^{z_j}, g^{a_j z_j}) \text{ has not been queried } \}$$

Choose a random $t \in \mathbb{S}$ For all $i \in \mathbb{S} \setminus \{t\}$, choose random $m_i \in \mathbb{Z}_p$ and set $m_i = \tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$. Let $u = \sum_{i<k}(a_i + \tilde{a}_i) - \sum_{i>k}(a_i + \tilde{a}_i)$ and $\alpha = \sum_{i<k, i \neq \max} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i - \sum_{i>k, i \neq \max} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i$. Set,

$$m_t = \begin{cases} -\alpha + du \text{ if } t < k \\ \alpha - du \text{ if } t > k \end{cases}$$

Return $m_t = \tilde{H}(id_t, g^{z_t}, g^{a_t z_t})$.

Step 5: Else if there exists $i \in [n] \setminus \{k\}$ such that $\mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)$ has not been queried, let

$$\mathbb{S} = \{j : j \in [n] \setminus \{k\}; \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i) \text{ has not been queried}\}$$

Select $l \in_R \mathbb{S}$. For all $i \in \mathbb{S} \setminus \{l\}$, choose random $\tilde{a}_i \in \mathbb{Z}_p$ and set $\tilde{a}_i = \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)$. Now, if $l < k$ compute

$$U_1 = \sum_{j=1}^{i-1}(a_j + \tilde{a}_j) + \sum_{j=l+1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{n}(a_j + \tilde{a}_j)$$

and if $l > k$, compute

$$U_2 = \sum_{j=1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{l-1}(a_j + \tilde{a}_j) - \sum_{j=l+1}^{n}(a_j + \tilde{a}_j)$$

Also let,

$$F = \sum_{j=1}^{k-1} \tilde{H}(id_j, g^{z_j}, g^{a_j z_j}) * v_j - \sum_{j=k+1}^{n} \tilde{H}(id_j, g^{z_j}, g^{a_j z_j}) * v_j$$

Compute $\tilde{a}_l$ as;

$$\tilde{a}_l = \begin{cases} -U_1 - a_l + \frac{F}{d} \text{ if } i < k \\ U_2 - a_l - \frac{F}{d} \text{ if } i > k \end{cases}$$

Set $\tilde{a}_l = \mathcal{H}(id_l, a_l, z_l, \pi_l, \pi_l^*, b_l)$.

Step 6: : If $\mathcal{D}$ has not already aborted, the following relation holds:

$d\left(\sum_{i=1}^{k-1}(a_i + \tilde{a}_i) - \sum_{i=k+1}^{n}(a_i + \tilde{a}_i)\right) = \sum_{i=1}^{k-1} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i - \sum_{i=k+1}^{n} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i$.

That is

$\prod_{i=1}^{k-1} g^{\tilde{a}_i} * b_i / \prod_{i=k+1}^{n} g^{\tilde{a}_i} * b_i = (g^{1/d}\tilde{g})^{d\left(\sum_{i=1}^{k-1}(a_i+\tilde{a}_i) - \sum_{i=k+1}^{n}(a_i+\tilde{a}_i)\right)} = (\hat{g})^{d\left(\sum_{i=1}^{k-1}(a_i+\tilde{a}_i) - \sum_{i=k+1}^{n}(a_i+\tilde{a}_i)\right)} = (B^{1/a})^{d\left(\sum_{i=1}^{k-1}(a_i+\tilde{a}_i) - \sum_{i=k+1}^{n}(a_i+\tilde{a}_i)\right)}$.

Here, $\tilde{a}_i = \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)$. $\mathcal{D}$ sets

$B_k = B^{d\left(\sum_{i=1}^{k-1}(a_i+\tilde{a}_i) - \sum_{i=k+1}^{n}(a_i+\tilde{a}_i)\right)} * \hat{g}^{\mu} = (\hat{g}^a)^{d\left(\sum_{i=1}^{k-1}(a_i+\tilde{a}_i) - \sum_{i=k+1}^{n}(a_i+\tilde{a}_i)\right)} * \hat{g}^{\mu}$.
Here, $\mu = d\left(\sum_{i=1}^{k-1}(a_i + \tilde{a}_i) - \sum_{i=k+1}^{n}(a_i + \tilde{a}_i)\right)\tilde{a}_k$.

$Exp^{POBB}_{\mathcal{B}'}(\lambda)$

$(G, p, g, \tilde{g}, n) \leftarrow Setup(\lambda)$
$a_k, z_k \in \mathbb{Z}_p$
$(\{(a_i, z_i) : i \in [n] \setminus \{k\}\}, \alpha, \{v_i : i \in [n] \setminus \{k, \alpha\}\}, v_\alpha = 1, aux) \leftarrow$
$\mathcal{B}'^{\tilde{H},H}(G, p, g, \tilde{g}, g^{z_k}, g^{a_k z_k}, \pi_k, \pi_k^*)$
$b_i \leftarrow \mathcal{O}^{\tilde{H}}\ (id_i, a_i, z_i, v_i) : i \in [n] \setminus \{k\}$
$c_0 \leftarrow \mathcal{O}^{\tilde{H}}\ (id_k, a_k, z_k, 0)$
$c_1 \leftarrow \mathcal{O}^{\tilde{H}}\ (id_k, a_k, z_k, 1)$
$\tilde{a}_i \leftarrow \mathcal{H}(id_i, Z_i, \phi_i, \pi_i, \pi_i^*, b_i) : \forall i \in [n] \setminus \{k\}$
$e \xleftarrow{\$} \{0, 1\}$
$\tilde{a}_k \leftarrow \mathcal{H}(id_k, Z_k, \phi_k, \pi_k, \pi_k^*, c_e)$
$B_k \leftarrow \left(\prod_{j=1}^{k-1} g^{\tilde{a}_j} b_j / \prod_{j=k+1}^{n} g^{\tilde{a}_j} b_j\right)^{a_k + \tilde{a}_k}$
$e' \leftarrow \mathcal{B}'^{\tilde{H},\mathcal{H}}(aux, c_e, B_k, \tilde{\pi}_k, \Pi_k)$
return$(e = e')$

$\mathcal{O}^{\tilde{H}}(id_i, a_i, z_i, \pi_i, \pi_i^*, v_i)$

$m_i \leftarrow \tilde{H}(id_i || g^{z_i} || g^{a_i z_i})$

return $g^{a_i} \tilde{g}^{m_i v_i}$

$\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$

return $\mathcal{RO}_1(id_i || g^{z_i} || g^{a_i z_i})$

$\mathcal{H}(id_i, g^{z_i}, g^{a_i z_i}, \pi_i, \pi_i^*, b_i)$

return $\mathcal{RO}_2(id_i || g^{z_i} || g^{a_i z_i} || \pi_i || \pi_i^* || b_i)$

Fig. 3: Description of the security experiment $Exp^{POBB}_{\mathcal{B}'}(\lambda)$.

Now, invoke $\mathcal{A}$ again with all these parameters. It will return a bit b. If $b = 0$ then $\mathcal{D}$ outputs $e = 0$, else $\mathcal{D}$ outputs $e = 1$. Alternatively, if $\mathcal{D}$ aborts, return a random bit.

Let us calculate the success probability of $\mathcal{D}$.
$P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1] = P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1 | \mathcal{D}$ aborts during the game $] * P[\mathcal{D}$ aborts during the game $] +$
$P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1 | \mathcal{D}$ does not abort during the game $] *$
$P[\mathcal{D}$ does not abort during the game $]$. Let,

$$\Psi = Pr[\mathcal{D} \text{ aborts during the simulation of } Exp^{POBC}_{\mathcal{A}}(\lambda)]$$

Thus, $P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1] = P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1 | \mathcal{D}$ aborts during the game $] * \Psi + P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1 | \mathcal{D}$ does not abort during the game $](1 - \Psi)$. If $\mathcal{D}$ does not abort during the game, then it returns whatever $\mathcal{A}$ returns, hence, $P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1 | \mathcal{D}$ does not abort during the game $] \geq P[Exp^{POBC}_{\mathcal{A}}(\lambda) = 1] = \frac{Adv^{POBC}_{\mathcal{A}}(\lambda)+1}{2}$. If $\mathcal{D}$ aborts in the game, $\mathcal{D}$ returns a random bit. Therefore, $P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1 | \mathcal{D}$ aborts during the game $] = \frac{1}{2}$. So, $P[Exp^{DDHV}_{\mathcal{D}}(\lambda) = 1] \geq \frac{Adv^{POBC}_{\mathcal{A}}(\lambda)+1}{2}(1 - \Psi) + \frac{\Psi}{2}$. From this we get, $\frac{Adv^{DDHV}_{\mathcal{D}}(\lambda)+1}{2} \geq \frac{Adv^{POBC}_{\mathcal{A}}(\lambda)+1}{2}(1 - \Psi) + \frac{\Psi}{2}$. Thus, we can write:

$$Adv^{DDHV}_{\mathcal{D}}(\lambda) \geq Adv^{POBC}_{\mathcal{A}}(\lambda)(1 - \Psi)$$

Let, A denote the event that $\mathcal{D}$ does not abort in step 2 and B denote the event that $\mathcal{D}$ does not abort in step 3. Hence, $\Psi = 1 - P[A \cap B]$. So, $1 - \Psi = P[A \cap B] = P[A]P[B|A]$. In step 2, $\mathcal{D}$ aborts only if $\hat{v}_i \neq v_i$ for any $i \in [n] \setminus \{k\}$. Since, $\hat{v}_i$s are chosen randomly by $\mathcal{D}$ with the condition that $\hat{v}_i = 1$ for at least one $i \in [n] \setminus \{k\}$, the probability $P[A] = P[\hat{v}_i = v_i : \forall i \in [n] \setminus \{k\}] = \frac{1}{2^{n-1}-1}$.

Now, let us calculate the probability $P[B|A]$ that $\mathcal{D}$ does not abort in step 3, given $\mathcal{D}$ does not abort in step 2. This happens only If for all $i \in [n] \setminus \{k\}$, satisfying $v_i = 1$, there is an entry $(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i, \tilde{a}_i)$ in L_i such that $r_i = \tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$ and $\prod_{j=1}^{k-1} g^{a_j + \tilde{a}_j} \tilde{g}^{r_j v_j} / \prod_{j=k+1}^{n} g^{a_j + \tilde{a}_j} \tilde{g}^{r_j v_j} = (g^{1/d} * \tilde{g})^{d(\sum_{j=1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{n}(a_j + \tilde{a}_j))}$. We try to find the probability that $\mathcal{A}$ queries $\mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)$ for all $i \in [n] \setminus \{k\}$, and $\prod_{j=1}^{k-1} g^{a_j + \tilde{a}_j} \tilde{g}^{r_j v_j} / \prod_{j=k+1}^{n} g^{a_j + \tilde{a}_j} \tilde{g}^{r_j v_j} = (g^{1/d} * \tilde{g})^{d(\sum_{j=1}^{k-1}(a_j + \tilde{a}_j) - \sum_{j=k+1}^{n}(a_j + \tilde{a}_j))}$ holds. The event $B|A$ happens when there exists $j \in [n] \setminus \{k\}$ such that in step 1, $\mathcal{D}$ replies to the query $\mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)$ with $\tilde{a}_i$, for all $i \in [n] \setminus \{k, j\}$ and for the query $\mathcal{H}(id_j, a_j, z_j, \pi_j, \pi_j^*, b_j)$ it returns $\tilde{a}_j$, where

$$\tilde{a}_j = \begin{cases} A_1 \text{ if } j < k \\ A_2 \text{ if } j > k \end{cases}$$

Here, $A_1 = -a_j - (\sum_{i<k, i \neq j}(a_i + \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)) - \sum_{i>k, i \neq j}(a_i + \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i))) + \frac{\sum_{i=1}^{k-1} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i - \sum_{i=k+1}^{n} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i}{d}$ and $A_2 = -a_j + (\sum_{i<k, i \neq j}(a_i + \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)) - \sum_{i>k, i \neq j}(a_i + \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i))) - \frac{\sum_{i=1}^{k-1} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i - \sum_{i=k+1}^{n} \tilde{H}(id_i, g^{z_i}, g^{a_i z_i}) * v_i}{d}$

The probability $Pr[B|A] \geq \frac{\prod_{i \in [n] \setminus \{k,j\}} \frac{1}{l_i}}{Q} = \frac{1}{Q} \prod_{i \in [n] \setminus \{k,j\}} \frac{1}{l_i}$, where l_i is the number of entries of the form $(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i, \tilde{a}_i)$ in log L_i when $\mathcal{H}(id_j, a_j, z_j, \pi_j, \pi_j^*, b_j)$ was queried.

Now, $1 - \Psi = P[A]P[B|A] \geq \frac{1}{2^{n-1}-1} \frac{1}{Q} \prod_{i \in [n] \setminus \{k,j\}} \frac{1}{l_i}$.

Thus,

$$Adv^{POBC}_{\mathcal{A}}(\lambda) \leq (2^{n-1} - 1)Q * \prod_{i \in [n] \setminus \{k,j\}} l_i * Adv^{DDHV}_{\mathcal{D}}(\lambda)$$

$Exp_{\mathcal{A}}^{POBC}(\lambda)$

$(G, p, q, g, \tilde{g}, n) \leftarrow Setup(\lambda)$
$a_k, z_k \in \mathbb{Z}_p$
$(\{(a_i, z_i) : i \in [n] \setminus \{k\}\}, \alpha, \{v_i : i \in [n] \setminus \{k, \alpha\}\}, v_\alpha = 1, aux) \leftarrow$
$\mathcal{B}^{\tilde{H}, \mathcal{H}}(G, p, g, \tilde{g}, g^{z_k}, g^{a_k z_k}, \pi_k, \pi_k^*)$
$b_i \leftarrow \mathcal{O}^{\tilde{H}}(id_i, a_i, z_i, v_i) : i \in [n] \setminus \{k\}$
$c_0 \leftarrow \mathcal{O}^{\tilde{H}}(id_k, a_k, z_k, 0)$
$c_1 \leftarrow \mathcal{O}^{\tilde{H}}(id_k, a_k, z_k, 1)$
$\tilde{a}_i \leftarrow \mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i) : \forall i \in [n] \setminus \{k\}$
$e \xleftarrow{\$} \{0, 1\}$
$\tilde{a}_k \leftarrow \mathcal{H}(id_k, Z_k, \phi_k, \pi_k, \pi_k^*, c_e)$
$B_k \leftarrow \left(\prod_{j=1}^{k-1} g^{\tilde{a}_j} b_j / \prod_{j=k+1}^{n} g^{\tilde{a}_j} b_j\right)^{a_k + \tilde{a}_k}$
$e' \leftarrow \mathcal{B}^{\tilde{H}, \mathcal{H}}(aux, c_e, B_k, \tilde{\pi}_k, \Pi_k)$
return$(e = e')$

$\mathcal{O}^{\tilde{H}}(id_i, a_i, z_i, \pi_i, \pi_i^*, v_i)$

$m_i \leftarrow \tilde{H}(id_i || g^{z_i} || g^{a_i z_i})$

return $g^{a_i} \tilde{g}^{m_i v_i}$

$\tilde{H}(id_i, g^{z_i}, g^{a_i z_i})$

return $\mathcal{RO}_1(id_i || g^{z_i} || g^{a_i z_i})$

$\mathcal{H}(id_i, a_i, z_i, \pi_i, \pi_i^*, b_i)$

return $\mathcal{RO}_2(id_i || g^{z_i} || g^{a_i z_i} || \pi_i || \pi_i^* || b_i)$

Fig. 4: Description of the security experiment $Exp_{\mathcal{A}}^{POBC}(\lambda)$.

Now, $\sqrt[n-2]{\prod_{i \in [n] \setminus \{k,j\}} l_i} \leq \frac{\sum_{i \in [n] \setminus \{k,j\}} l_i}{n-2}$. Since, the maximum number of oracle queries made by the adversary is Q, $\sum_{i \in [n] \setminus \{k,j\}} l_i \leq Q$. So, $\prod_{i \in [n] \setminus \{k,j\}} l_i \leq \left(\frac{Q}{n-2}\right)^{n-2}$. Hence,

$$Adv_{\mathcal{A}}^{POBC}(\lambda) \leq \left(2^{n-1} - 1\right) Q \left(\frac{Q}{n-2}\right)^{n-2} * Adv_{\mathcal{D}}^{DDHV}(\lambda)$$

That is

$$Adv_{\mathcal{A}}^{POBC}(\lambda) \leq \frac{2^{n-1} - 1}{(n-2)^{n-2}} Q^{n-1} * Adv_{\mathcal{D}}^{DDHV}(\lambda)$$

Since, $Q \in poly(\lambda)$, and n is constant,

$$Adv_{\mathcal{A}}^{POBC}(\lambda) \leq poly(\lambda) * Adv_{\mathcal{D}}^{DDHV}(\lambda)$$

Thus, the theorem holds. □

Note that, since, $Adv_{\mathcal{B}}^{POBA}(\lambda) \leq Adv_{\mathcal{A}}^{POBC}(\lambda) + negl(\lambda)$, we have,

$$Adv_{\mathcal{B}}^{POBA}(\lambda) \leq poly(\lambda) * Adv_{\mathcal{D}}^{DDHV}(\lambda) + negl(\lambda)$$

We have shown in section 4.3.2 that, if the DDH assumption holds in the algebraic group G, then $Adv_{\mathcal{D}}^{DDHV}(\lambda) \leq negl(\lambda)$ holds. Therefore, our two round PriVeto protocol will be secure against full collusion, if the DDH assumption holds in G.

5 Efficiency Analysis

In this section, we analyze the computation and communication efficiency of our scheme. Since, exponentiation is the costliest operation, we measure the computation cost in terms of the number of exponentiations done by an entity. In Round 1, each participant needs to do just 3 exponentiation operations to compute the Z_i, ϕ_i and the intermediate ballot. The NIZK proofs π_i and π_i^* need one exponentiation each. The NIZK proof $\tilde{\pi}_i$ requires 6 exponentiations, making the total exponentiations needed by a single participant in Round 1 equal to 11. In Round 2, the computation of the final ballot would require 2 exponentiations and the associated NIZK proof Π_i would require 2 exponentiations. Thus, in Round 2, each participant needs to do 4 exponentiations. Hence, the total number of exponentiations that a participant is required to perform during the entire protocol is 15. On the other hand, a verifier who wants to check all the NIZK proofs would need to perform $17n$ exponentiations. Table 1 shows a break-down of the computation overhead on a participant and a verifier.

Now, we calculate the communication overhead of the protocol. Each participant V_i needs to post Z_i, ϕ_i and b_i in Round 1. The participant also needs to post the two NIZK proofs π_i and π_i^* of size 6 in total. Thus, in Round 1, each participant needs to transfer 9 items. In Round 2, each participant needs to post only the final ballot B_i and the NIZK proof Π_i. The participant also needs to download all arguments uploaded by all other participants in Round 1. Together, they add up to $9n + 5$. The size of all NIZK proofs of all the n participants is 18. Table 2 shows a break-down of the communication overhead on a participant and a verifier.

	Round I			Round II			
Entity	Public Parameters	Intermediate Ballot	NIZKP	Public Parameters	Final Ballot	NIZKP	Total
Participant	2	1	8	–	2	2	15
Verifier	-	-	$12n$	-	-	$5n$	$17n$

Table 1 The number of exponentiations in computational load. Here, n is the total number of participants.

	Round I			Round II			
Entity	Public Parameters	Intermediate Ballot	NIZKP	Downloaded Parameters	Final Ballot	NIZKP	Total
Participant	2	1	14	$17(n-1)$	1	4	$17n+5$
Verifier	$2n$	n	$14n$	-	n	$4n$	$22n$

Table 2 The number of group elements in the usage of communication bandwidth. Here, n is the total number of participants.

6 Conclusion

In this paper, we have proposed a novel two round veto protocol that addresses the two major limitations of the original AV-net protocol. Our scheme achieves the best possible round efficiency for a veto protocol. In addition to that, it has the same asymptotic system complexities as the AV-net protocol. We have proved that our scheme is secure against full collusion in a random oracle model, assuming that the Decisional Diffie-Hellman problem is intractable in the algebraic group used in the scheme.

Acknowledgement

This work is supported by the ERC Starting Grant, No. 306994.

7 References

1 M Ajmal Azad, S. Bag, S. Tabassum, and F. Hao. privy: Privacy preserving collaboration across multiple service providers to combat telecoms spam. *IEEE Transactions on Emerging Topics in Computing*, PP(99):1–1, 2017.
2 Samiran Bag, Muhammad Ajmal Azad, and Feng Hao. A privacy-aware decentralized and personalized reputation system. *Computers & Security*, 77:514 – 530, 2018.
3 Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: A system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 257–266, New York, NY, USA, 2008. ACM.
4 Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security*, pages 325–343, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
5 Jurjen Bos and Bert den Boer. Detection of disrupters in the dc protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 320–327, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.
6 S.A. Brands. *Rethinking public key infrastructures and digital certificates : building in privacy*. MIT Press, 2000.
7 Felix Brandt. Efficient cryptographic protocol design based on distributed el gamal encryption. In *International Conference on Information Security and Cryptology*, pages 32–47. Springer, 2005.
8 Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 15–15, Berkeley, CA, USA, 2010. USENIX Association.
9 David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, Jan 1988.
10 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 462–462, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
11 David Chaum and Torben Pedersen. Transferred cash grows in size; eurocrypt'92, lncs 658, 1993.
12 Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *NSDI*, pages 259–282, 2017.
13 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in CryptologyCRYPTO86*, pages 186–194. Springer, 1986.
14 O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.
15 Philippe Golle and Ari Juels. Dining cryptographers revisited. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 456–473, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
16 S. Goryczka, L. Xiong, and B. C. M. Fung. m -privacy for collaborative data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2520–2533, Oct 2014.
17 Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *International Conference on Financial Cryptography*, pages 90–104. Springer, 2004.
18 F. Hao, P. Y. A. Ryan, and P. Zielinski. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, June 2010.
19 Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *International Workshop on Security Protocols*, pages 202–211. Springer, 2006.
20 Feng Hao and Piotr Zieliński. The power of anonymous veto in public discussion. In *Transactions on Computational Science IV*, pages 41–52. Springer, 2009.
21 Aggelos Kiayias and Moti Yung. Non-interactive zero-sharing with applications to private distributed decision making. In *International Conference on Financial Cryptography*, pages 303–320. Springer, 2003.
22 Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.
23 Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 129–139, New York, NY, USA, 1999. ACM.
24 Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.
25 Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '89, pages 690–, New York, NY, USA, 1990. Springer-Verlag New York, Inc.
26 A. C. C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, Oct 1986.

Appendix

Non-Interactive Zero-Knowledge Proofs

Here we discuss the construction of the non-interactive proofs used in our 2-round anonymous veto protocol. We shall use Fiat-Shamir heuristic [13] for converting standard interactive zero knowledge proofs into non-interactive ones. Such conversion requires a secure cryptographic hash function, hence our proof is constructed in a random oracle model. We tie every NIZK proof argument to the identity of the prover by making it mandatory to include the identity of the prover into the set of parameters that are fed as input to the hash function in order to generate the random challenge. This would effectively eliminate replay attacks [19]. The constructions of these kinds of NIZK proofs can be found in [6, 11, 24].

$\pi_i = NIZK[z_i : g, Z_i = g^{z_i}]$ **:** This NIZK proof proves the fact that given g and Z_i, the prover knows the value of $z_i = \log_g Z_i$. The construction of this NIZK proof can be found in [24] and is described in the following:
The prover selects $r \in_R \mathbb{Z}_p$ and generates a commitment $\epsilon = g^r$. Let the random challenge be ch. The prover generates a response $\rho = r - z_i * ch$. The verification equation is as follows:

$$g^\rho \stackrel{?}{=} \frac{\epsilon}{Z_i^{ch}}$$

The above NIZK proof requires one exponentiation for computation and two exponentiations for verification. The arguments include a commitment, a challenge and a response, making the total size equal to 3.

$\pi_i^* = NIZK[a_i : g, Z_i = g^{z_i}, \phi_i = g^{a_i z_i}]$ **:** This NIZK proof proves the fact that given Z_i and ϕ_i, the prover knows the value of $a_i = \log_{Z_i} \phi_i$. The construction of the NIZK proof is same as the one above.

$\tilde{\pi}_i = NIZK\left[a_i, z_i : g, g_i, Z_i = g^{z_i}, \phi_i = g^{a_i z_i}, b_i \in \{g^{a_i}, g^{a_i} g_i\}\right]$ **:** This NIZK proof proves that given g, g_i, Z_i, ϕ_i, b_i, either $\phi = DH_g(b_i, Z_i)$ or $\phi = DH_g(b_i/g_i, Z_i)$. The construction of this NIZK proof can be found in [6, 11] and is described as follows:
Let us assume that $b_i = g^{a_i}$. The prover selects random $r_1, \rho_2, ch_2 \in_R \mathbb{Z}_p$ and computes 4 commitments

$$\epsilon_{11} = g^{r_1}, \epsilon_{12} = Z_i^{r_1}$$

and

$$\epsilon_{21} = g^{\rho_2}(b_i/g_i)^{ch_2}, \epsilon_{22} = Z_i^{\rho_2}(\phi_i)^{ch_2}$$

Let, the grand challenge of the NIZK proof be Ch. The prover computes $ch_1 = ch - ch_2$. Also, the prover generates a response $\rho_1 = r_1 - a_i * ch_1$. The verification equations are as below.

- $g^{\rho_1} \stackrel{?}{=} \frac{\epsilon_{11}}{b_i^{ch_1}}$
- $Z_i^{\rho_1} \stackrel{?}{=} \frac{\epsilon_{12}}{\phi_i^{ch_1}}$
- $g^{\rho_2} \stackrel{?}{=} \frac{\epsilon_{21}}{(b_i/g_i)^{ch_2}}$
- $Z_i^{\rho_2} \stackrel{?}{=} \frac{\epsilon_{22}}{\phi_i^{ch_2}}$

Now, we show how a similar NIZK proof can be generated when $b_i = g^{a_i} g_i$. This time the prover selects random $r_2, \rho_1, ch_1 \in_R \mathbb{Z}_p$ and computes these 4 commitments

$$\epsilon_{11} = g^{\rho_1} b_i^{ch_1}, \epsilon_{12} = Z_i^{\rho_1} \phi^{ch_1}$$

and

$$\epsilon_{21} = g^{r_2}, \epsilon_{22} = Z_i^{r_2}$$

Let, the grand challenge be ch. The prover computes $ch_2 = ch - ch_1$. Also, the prover generates a response $\rho_2 = r_2 - a_i * ch_2$. The verification equations are as above.

This NIZK proof requires 6 exponentiations for computation and 8 exponentiations for verification. The arguments consist of 4 commitments, 2 challenges and 2 responses, hence the size of the proof is 8.

$\Pi_i = NIZK\left[a_i, z_i : g, \tilde{a}_i, Z_i = g^{z_i}, \phi_i = g^{a_i z_i}, \tilde{B}_i, B_i = \tilde{B}_i^{a_i + \tilde{a}_i}\right]$ **:** This NIZK proof proves that given $g, \tilde{a}_i, Z_i = g^{z_i}, \phi_i = g^{a_i z_i}, \tilde{B}_i, B_i = \tilde{B}_i^{a_i+\tilde{a}_i}$. The construction of the proof is as below: The prover selects random $r \in \mathbb{Z}_p$ and generates two commitments:

$$\epsilon_1 = Z_i^r, \epsilon_2 = \tilde{B}_i^r$$

Let, the random challenge be ch. The prover generates a response $\rho = r - ch * a_i$. The verification is given by

- $Z_i^\rho \stackrel{?}{=} \frac{\epsilon_1}{\phi_i^{ch}}$
- $\tilde{B}_i^\rho \stackrel{?}{=} \frac{\epsilon_2}{(B_i/\tilde{B}_i^{\tilde{a}_i})^{ch}}$

This NIZK proof requires two exponentiations for computation and 5 exponentiations for verifications. The size of the proof is 4.