applied sciences

MDPI

Article

A P2P Scheme for Debating and Voting with Unconditional Flexibility

Diego Antonio López-García [1,*,†], Juan Pérez Torreglosa [2,†], David Vera [3,†] and Manuel Sánchez-Raya [1,†]

[1] Department of Electrical Engineering, Computing and Automatics, Superior Technical College of Engineering, University of Huelva, Campus El Carmen, Avda. de las Fuerzas Armadas, s/n, 21007 Huelva, Spain; msraya@uhu.es
[2] Department of Electrical Engineering, Superior Technical College of Engineering, University of Huelva, Campus El Carmen, Avda. de las Fuerzas Armadas, s/n, 21007 Huelva, Spain; juan.perez@die.uhu.es
[3] Department of Electrical Engineering, Superior Polythechnic College of Engineering of Linares, University of Jaén, Avda. de la Universidad s/n, 23700 Linares, Spain; dvera@ujaen.es
* Correspondence: diego.lopez@diesia.uhu.es
† These authors contributed equally to this work.

Abstract: Most e-voting schemes make use of central servers. Users are obliged to trust these servers, which represent a vulnerability of the scheme. In the last few years, a very small group of schemes has been published that overcomes this handicap by using a peer-to-peer (P2P) approach. These are known as boardroom e-voting schemes, whereby users take the role of the servers. They act as managers of the process: they cast votes, keep a record of them, and verify the cryptographic operations made by others. Nevertheless, ballots must fulfill certain constraints which conflict with the possibilities of recent debate tools. These tools allow users to decide what to vote on, thus enabling the ballot frame to remain unknown before the voting process. The scheme presented here is a new boardroom voting protocol. It provides privacy, eligibility, and verifiability among other relevant features. The key advantage of this system is its high degree of flexibility, due to the absence of a need to impose any constraint on the ballots. This paper includes experimental results with two debate groups.

Keywords: boardroom e-voting; mixnets; anonymous channel; blind signature; P2P; decentralized; flexibility

Citation: López-García, D.A.; Torreglosa, J.P.; Vera, D.; Sánchez-Raya, M. A P2P Scheme for Debating and Voting with Unconditional Flexibility. *Appl. Sci.* **2024**, *14*, 3502. https://doi.org/10.3390/app14083502

Academic Editors: Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 13 March 2024
Revised: 14 April 2024
Accepted: 16 April 2024
Published: 21 April 2024

1. Introduction

Many e-voting systems have been developed over the last few decades [1,2]. Most of these provide privacy, eligibility, verifiability, and other features needed in common elections, but they rely on the integrity and trustworthiness of single points of failure. This vulnerability derives from their centralized nature. If the servers are few, collusion or failure becomes feasible. If there are many, the process turns slow and inefficient (especially for small groups). A decentralized e-voting system may be the answer; nevertheless, this approach poses additional challenges to the common problems of centralized e-voting systems [2]. The role that central entities generally play (bulletin boards, votes validation, break traceability, and so on) becomes collective. Following this idea some traditional e-voting methods could be adapted to a P2P network. For example, a mixnet [3] can be composed of all the users, turning a centralized system into a fully decentralized one. Nevertheless, in a naive implementation, the failure of a single user disrupts the process, ballots must follow a fixed size and frame, and each voting session needs predefined periods in which all the ballots are collected before starting the decryption process. Some of these problems have been addressed in improved methods such as in [4] where users interact with mixnet nodes to verify the process, in [5] where a pseudorandom disclosure over the permutation is shown, or in [1,6], where the use of threshold secret-sharing schemes

improves robustness. Nevertheless, the solutions take a substantial computational effort for every user and predefined periods are compulsory for every round of votes. Protocols that do not need mixnets [7] tend to incorporate homomorphic encryption instead [8,9], but this property implies a predefined and established ballot, resulting in inflexible systems.

As has been recently claimed in [10], it is difficult to find research related to decentralized e-voting. As far as we are aware, no such system has been designed that fulfills a minimum set of security requirements (privacy, eligibility, verifiability) along with flexibility. Early attempts come from the area of distributed systems which require synchronization and mutual exclusion in order to access shared resources. Examples of this kind of voting scheme are proposed by [11,12]. Other voting systems based on P2P have been proposed as a solution to the problem of malicious peers [13]. Such systems, however, do not address the issues of privacy or eligibility.

A field of interest is Byzantine fault-tolerant algorithms (BFTs). Anonymity in BFTs has been studied from different points of view, which are not exactly the case for e-voting systems. For example, Bitcoin has been compared to a BFT problem where participants are anonymous and it is solved by a proof of work [14]. But in this case participants have no constraints, they do not have to be members of any census. Privacy is also used as a tool in BEAT to avoid certain attacks [15], but only on the client side. In other paper, identities are checked by zero-knowledge proofs and devices could remain anonymous [16], but there are not published results. Therefore, it seem not easy extrapolate BFT models to e-voting.

As a response to these issues, a set of e-voting schemes called "boardroom voting protocols" have been developed [17]. For example, Kiayias [18] proposed a scheme in which every user broadcasts a pair of keys for the rest of the users who can then build a ballot with these keys and broadcast them in such a way that the product of all the ballots contains the tally. This scheme was improved in [19–21], but cannot avoid the homomorphic nature of these methods, which results in inflexibility. A solution to this problem is provided by [22], which uses a mixnet approach to introduce more flexibility into the ballot design by allowing priority ranking and write-in ballots. Nevertheless, the ballot frame must be defined at the beginning of the voting session and cannot be altered during the process.

Another area of research for e-voting is block-chain [23]. All methods taking this approach achieve partial decentralization in comparison to classic methods. Specifically, the bulletin board is substituted by the block-chain, a sequence of blocks where all public data is stored [24]. Even the processes involved can be established in the block-chain by the use of smart contracts [25]. Nevertheless, there are always certain functions that must be carried out by a third party or in a centralized manner. In [26], registration servers, authentication servers and a web server are needed. In [27], voters use a network of "Bootnodes" and "District nodes" that a third party must set up. The work published in [28] would seem to be the most decentralized scheme, but it requires a census authority to assign a secret number to each voter, which implies that this entity can reveal the vote of any participant (limited privacy). In other block-chain methods the problem stems from the homomorphic encryption and its inherent inflexibility [29]. All of these methods improve decentralization, but none of them achieves the flexibility of enabling variable-length ballots and the submission of new proposals or candidates for voters during the election process.

The purpose of this work is to provide flexibility to decentralized e-voting systems. The outcome is a new boardroom e-voting scheme free of message constraints. It is based on anonymous keys, as in the centralized method [30], but is fully distributed and retains the advantage of flexibility. The method works as an encryption layer that guarantees anonymity while preventing double voting or access to unregistered users. Thus, this layer provides the common e-voting security features to the application layer on which any debate software can sit. Therefore, it is not only suitable for voting on a list of candidates, but also for negotiating the minutes of meetings, collaborating on budgets, sharing documents, and so on. The drawback, as occurs with other decentralized e-voting schemes, is the computational cost, which increases with the number of participants.

In summary, the main contributions of this paper are

1. A new boardroom e-voting scheme with unconditional flexibility, suitable for any P2P debate tool.
2. A security study that determines its strength.
3. A list of features with the degree of compliance in each of them.
4. Experiments that measure the computational cost for different scenarios.

The paper is organized as follows. Section 2 explains key aspects required to understand the new scheme. This is detailed in Section 3. Section 4 analyzes its main features. Section 5 presents and discuss experimental results. Section 6 closes with the conclusions.

2. Preliminaries

Before explaining the new scheme, it is necessary to comment on certain aspects that are basic to understanding it. This section is dedicated to clarifying them.

The key idea of this scheme is that of the blind signature by an "alias" in the form of a public key, whether it belongs to Rivest–Shamir–Adleman cryptosystem (RSA) or ElGamal. Other studies have exploited blind signatures in voting schemes applied to ballots or tokens [31,32], but not applied to another key. With this alias key, users can interact with others as no privacy was needed. Privacy is solved by the alias validated from a blind signature and an untraceable physical access, such as The Onion Router network (TOR), Freenet, virtual private networks (VPNs), public wireless access points (AP), etc. There are two required functions: alias blind signature of valid users and vote checking of valid aliases. In this scheme, these functions are replicated by all the users. With this general idea the next subsections detail the elements needed in the present scheme.

2.1. Blind Signature

A blind signature is similar to a digital signature except that it allows an entity (a user) to obtain another entity (usually the authority that checks the identity of the user) to sign a message without revealing its content. Blind signature [33] is used to authenticate the user without disclosing the content of a ballot. Hence, the authority, whose function is to verify the eligibility of a user, will not know who is voting. The security level of blind signature has been analyzed with satisfactory results [34].

To ensure the secrecy of their message, users generate a random number r in a blinding function and send it to the authority. For example, if the authority A has a pair of RSA private and public keys denoted by $\{K_A^{-1}, K_A\}$, the blind function for m (where m is the hash of a message M), $Blind(m, r, K_A)$, can be defined as:

$$Blind(m, r, K_A) \equiv m \cdot r^{K_A} \equiv m' \pmod{n_A} \tag{1}$$

This (1) and the following equations represent operations in modular arithmetic. The authority receives this blinded message and cannot disclose it without r. Next, the authority signs the blinded message with its private key K_A^{-1} and retrieves it to the user:

$$Sign(m', K_A^{-1}) \equiv (m')^{K_A^{-1}} \equiv (m \cdot r^{K_A})^{K_A^{-1}} \equiv$$

$$m^{K_A^{-1}} \cdot r \pmod{n_A} \tag{2}$$

Finally, the user removes the blinding factor r by multiplying all by r^{-1} and obtains the message signed by the authority:

$$Sign(m', K_A^{-1}) \cdot r^{-1} \equiv m^{K_A^{-1}} \cdot r \cdot r^{-1} \equiv m^{K_A^{-1}} \equiv$$

$$Sign(m, K_A^{-1}) \pmod{n_A} \tag{3}$$

In the method presented here, every user contacts with the other users, revealing his/her identity by means of a digital certificate. The other users check that user against a

list and make just a blind signature per user ($n-1$ signatures where n is the total number of users), playing the role of authorities. Blinded messages can be stored but signers will not be able to disclose them. Therefore, privacy is guaranteed. Although RSA cryptography has been used in this example, other cryptographic systems capable of implementing blinding and signing functions may also be valid.

2.2. Physical Traceability

In the first stage users must be identified to allow blind signatures. Users then participate in the second stage using an alias to protect their identity. Although blind signatures provide privacy at application layer, users can nevertheless be identified by correlating the transmission of the aliases with the IP addresses of the packets. This means that another tool for privacy is required at the network layer. This problem is solved by anonymous channels, that is, channels in which the sender of a message cannot be identified. Various proposals have been made for anonymizing channels:

1. Network Address Translation (NAT) [35]: One defense consists of using the same IP address for multiple users, which is private and dynamic. A border router maintains the NAT tables, which must therefore be protected in order to maintain privacy.
2. Mixnets [6,7]: Mixnets are devices that receive a list of encrypted messages which are then shuffled and decrypted. Here, security depends on trusting in at least one node of the mixnet. Their shortcoming is that they cannot work until all the messages have been received, making it difficult to adapt to dynamic online debates.
3. DC-nets [36]: In a DC-net, users cooperate to behave as a virtual broadcast network. Every message is processed by all the users in the network. The problems in this case are twofold: scalability and the fact that one single user can block communications.
4. Public places: Public places, such as libraries, hotels, universities, Internet cafes, venues, etc., are points from which users can safely connect. Should an attacker manage to trace the IP address of an alias, this will show the public place, which cannot easily be related to a particular user, more so in the case of broadcast communications like Wi-Fi networks, for which authorities could offer access points.
5. Private proxies/VPN [37]: A proxy is an intermediary between users and the Internet. The connection with users can be protected by a VPN (encrypted channel), such that an attacker can follow the alias submissions to the proxy, but not beyond. This solution depends on the trustworthiness of the proxy. Even with this protection, however, statistical analysis over the network can correlate the times of outgoing alias submissions with user submissions entering the proxy. Nevertheless, this kind of attack is extremely difficult because the attacker must be able to manage most of the routers along the path (usually owned by different companies), and take advantage of low traffic conditions.
6. TOR network [38]: The idea behind TOR is to connect to a web server by tracing a route through a series of nodes. The user chooses the sequence of TOR nodes to follow from a list. Each node receives a request to send encrypted data to the following node, such that each one knows only the immediately previous and subsequent nodes, but cannot reveal the sequence, the sender or the destination. The higher the number of nodes the greater the latency, but also the better the security. TOR is constantly scrutinized and is vulnerable to statistical analysis, but is widely used as a secure privacy provider.
7. Invisible Internet Project (I2P) [39]: I2P provides a similar solution to TOR, but is more resistant to traffic analysis and incorporates different node sequences for incoming and outgoing packets, thus increasing security.

Other solutions are available (e.g., Freenet, GNUnet, RetroShare), which can be combined (proxy with TOR) to increase security. Nevertheless, this question can be freely decided by the users as long as the e-voting scheme is properly designed.

3. Scheme

This scheme considers a set of n users. Each one has a digital certificate DC_i whose public key K_i is known to the others, or signed by a third party trusted by them, and whose private key K_i^{-1} is secret. The process starts when a user decides to propose a debate. This user then sends a signed message to the rest. This message contains: the list of users (list of public keys of their digital certificates), the periods for each stage (blind signature, vote, checking, etc.) and other data related to the question to be debated. If the receiving users accept these conditions, then the first phase of the scheme starts. If any of them does not, they can notify this to the rest, who will decide whether to continue (a message with a new list must be sent) or not. At this point users can discuss openly and warn others of any misbehavior on the part of the initiator. They will only accept if they all agree to the conditions, which must be the same for all users (this is checked in the following steps).

3.1. Phase 1: Obtaining an Alias

The objective in this stage is to provide users with an alias key signed by themselves. All the steps are depicted in Figure 1. For each user the steps are the same:

1. Each user establishes a private channel with each of the other users. Transport Layer Security protocol (TLS) can provide it, but in the future there could be better solutions [40,41]. In general, user U_i builds a channel with user U_j where $j \in [1, n]$ and $j \neq i$. This makes a total of $n-1$ channels for user U_i. Basically, they use their digital certificates DC_i and DC_j to prove their identity by signature checking and negotiate a symmetric key.
2. They compare the message received from the initiator. If there is a mismatch, they check the initiator signature, determining who is being deceptive, whether the user (only one coherent signature) or the initiator (two different and coherent signatures). If initiator is being deceptive or the number of users trying to deceive is greater than $\sqrt{n+1}-1$ the process ends, namely this user will warn the rest and will close connections. Otherwise the process continues.
3. Each user generates a pair of RSA keys. For example, user U_i generates the pair $\{A_i^{-1}, A_i\}$, from now on 'alias' keys. The hash of the public alias key, $a_i = hash(A_i)$, is then computed.
4. Each user U_i blinds their public alias key hash a_i with a random number r_{ij} and encrypts it with the public key of the other users. Therefore, user U_i computes a maximum of $n-1$ blind messages, $m_{ij} = Blind(a_i, r_{ij}, K_j)$ according to Equation (1). These messages are then sent through the private channels to the corresponding peer (the user who owns the private key used in blinding in each message, that is, U_j).
5. Each user U_j receives these messages: blinded alias of the other users. Then each one shows others a table with just one received message per user. If there are any mistakes it can be solved publicly with a signature by the sender (any user can check what is sent by the signature and what is stored by the published table). Finally, everyone knows what messages have been sent to the others and that their own have been logged correctly.
6. Each user U_j signs with her/his private key K_j^{-1} the received messages and gives back the results. That is, U_j sends $Sign(m_{ij}, K_j^{-1})$ back to U_i. If any user U_j refuses to sign or sends a wrong signature, the affected U_i can claim publicly and force U_j to compute a right signature, because it can be checked by the other users.
7. Each user U_i removes the blinding factor r of the message and obtains his/her alias key signed by U_j. Therefore, at the end of the process each user U_i has the hash of their alias public key signed by the other $n-1$ users, which allows building $n-1$ alias certificates AC_{ij}.

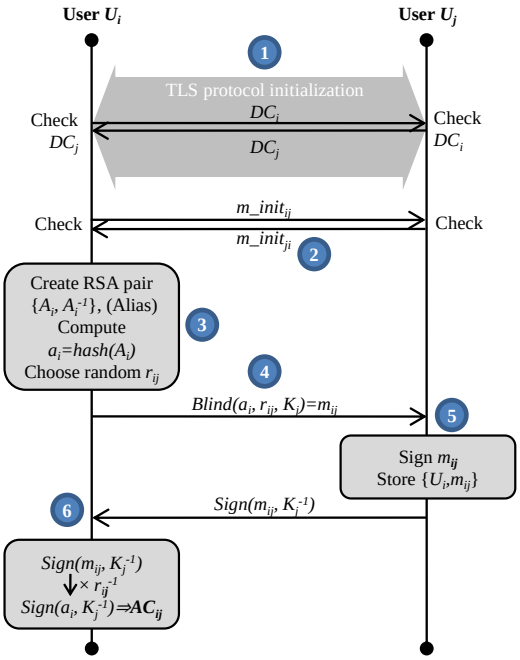


Figure 1. Phase 1: Each pair of users (U_i, U_j) builds a secure communication channel (1,2) and generates their alias keys (3). Then they transmit their hidden alias thanks to a random number. For example, U_i sends a_i hidden by r_{ij} (4). The receiving user (U_j) signs and returns it (5). Removing the r_{ij} factor gives the signed alias AC_{ij} (6).

3.2. Phase 2: Alias Checking and Debating

The second phase groups the operations required to accept alias keys from the other users (Figure 2). This phase makes use of anonymous channels that prevent physical traceability (see Section 2). Users connect to the others through these channels, identifying themselves with their alias key only, and proving their eligibility with the corresponding signatures obtained in the previous phase. The steps are threefold:

1. User U_i runs the TLS protocol over an anonymous channel using the alias certificate AC_{ij} with each user U_j. In this link U_i acts as a client and U_j as a server. User U_i must check the digital certificate DC_j of user U_j and user U_j must check his/her own signature on alias hash, namely $Sign(a_i, K_j^{-1})$ shown in alias certificate AC_{ij}. If it passes the test, the alias is accepted. At the end of this process each user has two TLS channels with each of the other users: one as an alias client to send messages and the other as a public server to listen to other aliases.
2. Now users can interact with others to argue, propose, and vote. This can be conducted in many ways, depending on the software that manages the debate. Any of them will allow users to make contributions (opinions, proposals, arguments, candidates, votes, etc.), which can vary in types, length and quantity. However, regardless of their features (unconditional flexibility), these contributions are messages that can be sent by a secure TLS channel which uses symmetric encryption or, when available, a broadcast anonymous channel. Thus, using these channels, any debate software can be used. Notice that this software works in a centralized way with respect to the rest of aliases.
3. The end of the debate and voting is determined by the time defined at the beginning with the initiator. At this point, all the contributions of an alias A_i can be compiled in a list. Users can check whether all their contributions (as aliases) have been registered in this list, and provide anything missing if necessary. When the list is complete, its

hash signed with A_i^{-1} is sent by each alias and checked by the receiver. This ends alias interaction.

4. Each user gathers all data sent by aliases, signs it with his/her digital certificate, and sends it to the other users. This signature prevents users from sending different data to different users without being detected. Then they compare what they have received. If there is a consensus the process ends. If there is a mismatch, signatures of users are compared to detect inconsistencies, and the aliases are revised. Aliases which are present more than $n + 1 - \sqrt{n+1}$ times are accepted and the tally is computed with them.

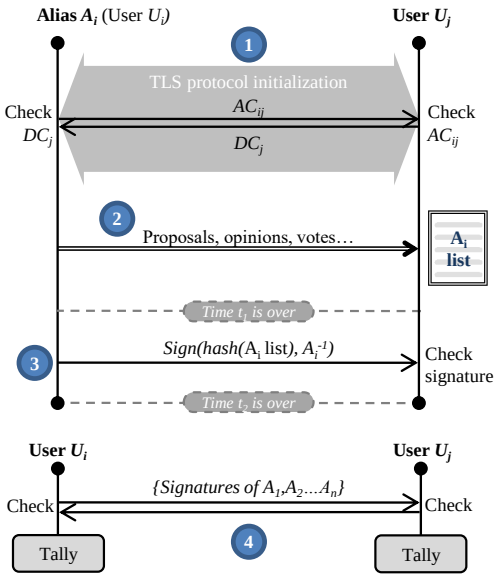


Figure 2. Phase 2: Users show their alias certificate AC_i to establish anonymous TLS channels (1). Then they can debate (2). When the time is over they sign their list of contributions (3). Signatures are checked to solve inconsistencies (4) and the tally is computed.

4. Analysis

In this section, the security of this scheme is analyzed in the light of critical assumptions. The first of these is that the time required for adversaries to break asymmetric keys is longer than the time required to carry out the poll, and that in which any consequences to the vote owners might accrue. The assumption is based on using adequate key lengths regarding the computational capability of any adversary. The second assumption is that the voter group has fewer adversaries than honest members. The limit is explained below. The third is that devices are free of any malware. This can be achieved by following the appropriate security policies, such as using live CDs recorded with verified software, so that users start with a secure operative system (OS) and debate application. Hackers should be deterred by having to find vulnerabilities in a system tested by experts, and dedicated solely to the process of voting (with no additional services or ports open). Should vulnerabilities exist, the opportunity to exploit them should be limited to the short interval in which voting is carried out. The fourth assumption is that network services guarantee communications among users. Problems related to service failures are not considered in the protocol. Lastly, the selected anonymous channel does not provide any information about senders. Features of messages received (time, frequency, size, sequence, etc.) cannot attribute more probability to one user than another. As mentioned above (Section 2), there are different tools to avoid physical traceability whose strength is not considered in this paper. Users can choose these anonymous channels according to security level and speed.

4.1. Threat Model

From these assumptions, we can study a specific threat model in which the capabilities of adversaries are:

1. They can participate as any other user, but they cannot exceed $\sqrt{n+1}-1$, where n is the total number of users.
2. They can send any message, without constraints about the content, or choose not to send anything, to any user.
3. They cannot alter the hardware or software of any user.
4. They cannot compute the inverse hash of any number. They could choose any value, compute the hash and compare it with a given hash, but the odds of matching are negligible.
5. They cannot deduce the private key of any user, nor compute the encryption of a given value with the private key. Therefore they cannot compute honest users' signatures.
6. They cannot stop communications between honest users.
7. They cannot obtain any information from network traffic when using anonymous channels.

From now on, for the following lemmas and theorems, these assumptions and models will be considered.

4.2. Preliminaries

The process starts with a user who proposes the debate to a group. If this user is an adversary, he/she could send any message with any parameter set to the others. However, in the second step of phase 1 users will look for differences in the received message, which has been signed. Thus, any mismatch is detected and attributed to its author. Users cannot deceive either in this checking process either, because with any fake message (attempting to inform against the initiator) they must show a coherent signature that only initiator can compute. Participants must connect to each other in this phase, and hence the absence of any of them is detected. The initiator can deliver different participant lists to users, but coherence among the members of the same list must be observed in order to pass onto the second step. Even so, users can refuse to participate if they judge that the list or any other parameter is not right.

At the beginning of each phase, each user contacts the others establishing secure channels, following the TLS protocol. This secure channel is based in asymmetric keys and collision free hash functions that provide authentication, confidentiality, and integrity. Hence, under the threat model assumptions, adversaries outside the user list cannot usurp legitimate user identities (man-in-the-middle attacks), or alter information without being detected or revealing the content of what is sent once the channel has been established. This applies not only to the first phase, which uses digital certificates, but also to the second phase, where alias keys are used instead.

The next part of the paper considers attacks from adversaries that are group members. As valid users they can achieve signatures from the others, but only one from each honest user. They can sign multiple aliases among themselves, but the rest of the users will only admit their own signature on each alias in phase 2. Therefore, as it is established in lemma 1 below, adversaries cannot login with additional aliases.

Lemma 1. *Adversaries cannot obtain more signed alias keys from any honest user than there are eligible users among said adversaries.*

Proof. To obtain an extra alias key means achieving the following set $\{A_e, A_e^{-1}, sign(a_e, K_i^{-1})\}$ where $\{A_e, A_e^{-1}\}$ must be a pair of asymmetric keys, $a_e = hash(A_e)$ and K_i^{-1} is the private key of any honest user. From a_e it is not possible to obtain A_e (condition 4 of Section 4.1), nor is it possible from A_e to deduce A_e^{-1} (condition 5). Therefore, adversaries obtain an extra alias key, the hash a_e must be computed from it, and the reverse path is not possible. Thus, adversaries must find a way to obtain the signature of a given number a_e, that could be any

value in the interval defined by the hash function. Nevertheless, were they to achieve this, there would be a method to obtain the signature of any number, which violates condition 5. Thus, for any given hash, the only way to obtain a signature is that the legitimate user signs it, and an honest user will sign just one per participant. □

During the debate, adversaries can show only the same number of alias keys as there are per user (Lemma 1). Any contribution (votes) is associated with one alias key and cannot be duplicated (uniqueness). The TLS channel and the signature of the contributions at the end prevents users from impersonating other aliases. Nevertheless, adversaries can use different aliases with each user. This way, they can vote differently in each user bulletin board, looking for inconsistency among the group. Therefore a threshold of alias appearances in bulletin boards must be established in order to admit them, and it is not as straightforward as choosing $n/2$. The next attack example shows the problem.

Attack Example

Let there be a group of six members: four honest and two adversaries. At the end of the first stage each one shows his/her own list of accepted aliases to the rest, resulting in the set of published lists in Figure 3.

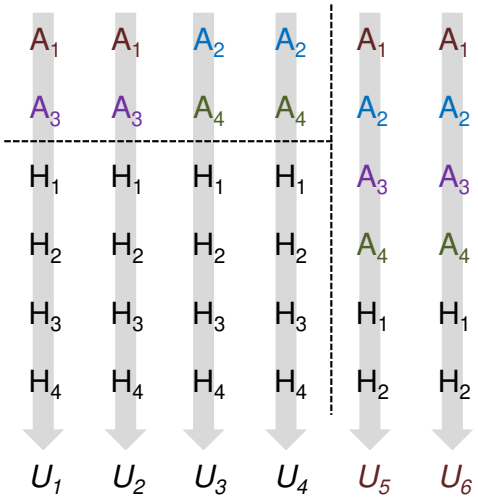


Figure 3. Attack example: Bulletin boards of users U_i shown as columns. H_i are alias of honest users; A_i are alias of adversaries (U_5, U_6). If the acceptance criterion is to be in at least 4 columns, the adversaries may vote double.

Each column of Figure 3 corresponds to the list shown by each user. For example, the first column corresponds to the aliases accepted by the first user, which in this case belongs to the honest group $\{U_1, U_2, U_3, U_4\}$. The bottom part of this column contains the aliases sent by honest users $\{H_1, H_2, H_3, H_4\}$. As they are honest, they have used the same alias in all the processes. The upper part shows the aliases that adversaries have sent. With this user in particular, U_1, they have sent the aliases A_1 and A_3.

Adversaries can send different aliases to different users, for example U_1 and U_2 receive A_1 and A_3, while U_3 and U_4 receive A_2 and A_4. Also, they can show whichever set of aliases in their own list. In this case they choose to show A_3 and A_4 instead of H_3 and H_4. At the end, honest users realize that there are adversaries because there are more aliases than users, but cannot deduce who are them. A_i appears the same number of times as H_3 or H_4, which means that most users (more than $n/2 = 3$ in this case) accept these aliases. Therefore there must be a limit for adversaries and a threshold for accepting aliases higher than $n/2$, so that extra aliases are banned. With no extra aliases adversaries can neither vote twice nor lead to inconsistencies. The next lemma establishes the minimum amount of

adversaries to achieve this kind of attack and explains the threshold used in the last step of the protocol:

Lemma 2. *Adversaries are not able to force different results (inconsistency) among honest users if they number less than* $\sqrt{n+1}-1$ *where n is the total number of users.*

Proof. If half the honest users vote exactly the opposite to the other half, one adversary vote is able to decide the result. If adversaries manage to register this vote with a different value in different users, the result will differ among users. It would therefore appear that inconsistency can be achieved by changing just one vote. However, adversaries cannot vote differently with the same alias because at the end users will compare the contributions of each alias (Section 3.2 4.) and detect discrepancies. Registering different votes thus implies using different aliases. Furthermore, each honest user will accept just one alias per adversary, but they cannot detect that these accepted aliases are the same for every honest user. In other words, adversaries can use different alias for each honest user. Let us consider that there is the minimum number of adversaries m that achieves inconsistency. The rest $h = n - m$ are honest. Then, aliases from honest users will be present in at least h user bulletin boards. Let us choose this value as a minimal presence for any alias in order to be accepted (this is just the threshold to be calculated). For this purpose, adversaries can use the room in honest users' bulletin boards assigned to them (let us call this area 1, which corresponds to the upper left-hand corner of Figure 4), who can accept $m \cdot h$ blind signatures, and the room in their own bulletin boards as valid users (let us call this area 2, which corresponds to the right-hand side of Figure 4), $n \cdot m$, where they can show any alias list. Aliases cannot be repeated in the same user bulletin board (same column in Figure 4). With this constraint each adversary alias will collect no more than m signatures in area 2. If the objective of adversaries is to establish as many aliases as possible, they will fill area 2, which is larger than area 1, with different aliases (the maximum would be n). The rest of the signatures for each alias must be present in area 1 at least $h - m = n - 2m$ times. Dividing area 1 ($m(n-m)$) between $n - 2m$ the maximum number of aliases defined by this method without exceeding n is obtained. If adversaries achieve more aliases than there are eligible users among said adversaries, that is, at least $m + 1$, inconsistency is possible, which can be expressed as:

$$\frac{m(n-m)}{n-2m} \geq m+1 \equiv m \geq \sqrt{n+1}-1 \tag{4}$$

Therefore, if $m < \sqrt{n+1}-1$ adversaries cannot register different aliases, and inconsistency is not possible. □

In the previous example $n = 6$, therefore the threshold is $m < \sqrt{6+1}-1 = 1.6$. In other words, when there are six users the maximum number of adversaries is one and a minimum of five signatures are required for any alias. A single adversary would need four signatures for each alias from honest users besides his/her own. For two aliases it would be eight, and he/she can achieve only five. On the other hand, the example shows that a successful attack can be carried out with two adversaries, just one over the threshold.

As can be seen, phase one and the beginning of the next are devoted to obtain valid aliases in the sense that they must accomplish three conditions: representing an eligible user, just one per user, and keeping privacy.

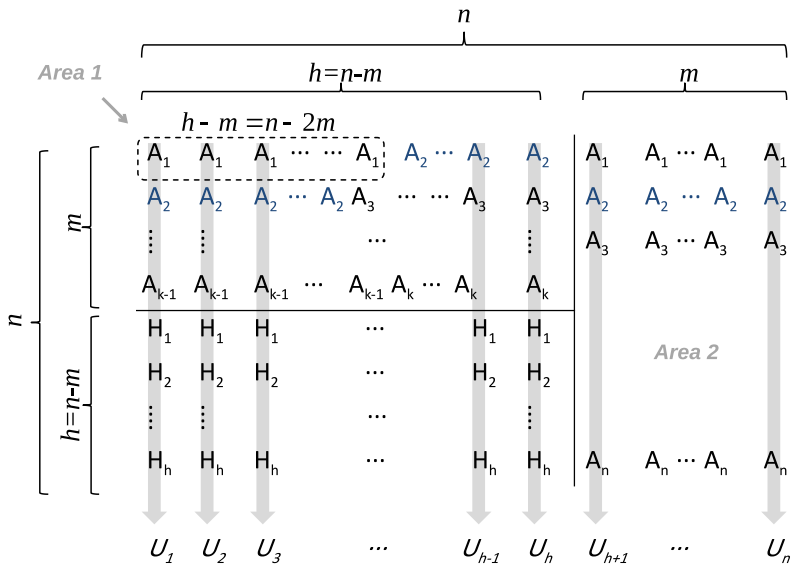


Figure 4. Aliases $\{A_i, H_i\}$ registered by each user U_i depicted in columns. There are n users where m are adversaries and $h = n - m$ are honest. Each adversary alias A_i needs to be present in at least h different columns in order to be accepted, just the number of times any honest alias will appear.

Definition 1. *A digital certificate is a valid alias if the next conditions are achieved: (1) The private key belongs to an eligible user. (2) The public key has been sent to the rest of users by a method that prevents them of knowing the sender (privacy). (3) The public key has been checked by the rest of users by a method that avoids multiple aliases for the same user (uniqueness).*

Theorem 1. *Users will obtain a valid alias per honest user and no more aliases than there are adversaries.*

Proof. As mentioned above, TLS channels and digital certificates prevent adversaries out of the user list from participating in this scheme; therefore, only eligible users can send their aliases to the rest. In other words, any alias received in this scheme belongs to an eligible user, so first property of valid aliases is accomplished. As aliases reach users by anonymous channels in phase 2, there is no way to relate user with alias by means of how the data is sent (physical traceability). Considering the data there are two exchanges: a blinded message b from a known sender in phase 1 and a public key, the alias a, signed by the receiver from an unknown sender in phase 2. Individually, none of them is enough to reveal the author: b is computed with a random and unknown number, a has nothing linked to the author except an unknown private key, and the signature on a only throws that he/she is one of the eligible users who were signed in phase 1. (An adversary could make only one signature in this phase to reveal one alias, but he/she will be detected in step 6 and banned.) However, taken together, there is a mathematical relation between a and b: $b = a \cdot r^{K_i} (mod\ \ n_i)$. But r is a random and unknown number. Let x be any other alias public key and let s be $(x^{-1} \cdot a)^{K_i^{-1}} \cdot r (mod\ \ n_i)$. Then, if s is chosen to build the blind signature with x it follows:

$$Blind(x, s, K_i) = x \cdot s^{K_i} = x \cdot ((x^{-1} \cdot a)^{K_i^{-1}} \cdot r)^{K_i} (mod\ \ n_i) = a \cdot r^{K_i} (mod\ \ n_i) = Blind(a, r, K_i)$$

In other words, for any alias x there is a number s such that the blind signature $Blind(x, s, K_i)$ matches b. Therefore, exchanged data cannot be used to deduce the owner of a. As the receiver cannot link alias with user, the second property of valid aliases is accomplished too. The last property (uniqueness) is satisfied in honest users because they will send

only one alias to the others. Taking into account that adversaries are less than $\sqrt{n+1}-1$ (security model) and Lemma 2, inconsistency is not possible. As in the proof of Lemma 2 is described, this means that honest users cannot accept more aliases from adversaries than they are. □

The consequence of this theorem is that either adversaries send a valid alias to honest users behaving like them, or they achieve less aliases accepted than they are, which is against adversary interest.

Definition 2. *An authenticated anonymous channel is a secure and anonymous channel built with a valid alias on one side and with the public digital certificate of a known user on the other.*

The term "secure" in this context must be identified with the three services provided by TLS protocol: confidentiality, integrity, and authentication. Therefore, this definition matches the channels described in phase 2.

Theorem 2. *Authenticated anonymous channels allows for (1) authenticity for a known user on one side, (2) authenticity of membership for the other, and (3) anonymity for any data sent from the last one.*

Proof. The scheme describes that these channels are built with TLS protocol over anonymous channels. As TLS protocol provides authentication based on digital certificates, the user that shows the known one is identified, satisfying the first property. The other user shows a valid alias. The first requirement of a valid alias is belonging to an eligible user. Therefore, a valid alias proves the membership to the group of eligible users, satisfying property 2. The last one is accomplished if nothing in the channel itself or the data exchanged identifies the sender. As it was explained in the proof of theorem 1, anonymous channels break physical traceability and data exchanged until alias acceptance cannot reveal the sender. Next, the TLS protocol needs only the alias certificate and random numbers to build the channel, nothing else from the sender is used. Therefore, the channel is built without revealing the sender, and any data can be sent through it. □

4.3. Security Requirements

Some of the security features usually analyzed in other voting systems [2,34] are uniqueness (mentioned above), privacy, eligibility, verifiability, dispute-freeness, accuracy, fairness, mobility, incoercibility, robustness, and scalability. These requirements and consistency are analyzed in the next subsections and summarized in Table 1.

Table 1. Summary of requirements.

Property	Degree of Compliance
Uniqueness	Satisfied.
Privacy	Maximum privacy at logical level.
Eligibility	Satisfied.
Verifiability	Individual and universal verifiability.
Accuracy	Satisfied.
Dispute-freeness	Satisfied.
Fairness	Requires vote encryption and disclosing at the end.
Mobility	Satisfied.
Robustness	Adversaries $< \sqrt{n+1}-1$.
Consistency	Adversaries $< \sqrt{n+1}-1$.
Incoercibility	Unsatisfied.
Flexibility	Unconditional flexibility.
Scalability	Unsatisfied.

4.3.1. Privacy

In e-voting systems, any traceability between the user and their vote must be removed. Clearly, if the rest of the users reveal their vote, the latter would be exposed. When this is the only case in which a vote can be associated with its user, the highest level of privacy is achieved, called 'maximum privacy'. However, this scheme protects messages without constraints until they are published. Messages could provide information about users in different ways (style of text, content of messages, alias profile, Italian attacks, etc.). Depending on the context, this data leakage is possible or not. In a open debate users can include revealing details in their opinion or proposals. Nevertheless, a simple yes/no vote cannot reveal anything about its author except the case of maximum privacy. Therefore, the following property is stated.

Proposition 1. *Under the conditions of the security model and as long as the content of the messages does not reveal anything about the users, any two of them are indistinguishable in phase 2.*

Proof. In this context, all the information that adversaries can accumulate is the following: a list of blind messages ($m_{ij} = Blind(a_i, r_{ij}, K_j)$) that are linked with their users, a list of blind signatures, a list of signed aliases (a_i), and the contributions (proposals, votes) signed by each alias. Since the premise of the proposition discards the information obtained from the contributions, only the information related to the alias itself remains. Distinguishing one user from another in phase 2 means distinguishing their aliases. However, considering all the information available to adversaries, there is only one element in which the user and alias are involved, which is the blinded message. If adversaries manage to distinguish two users, U_1 and U_2, then they are able to guess the relationship between user and alias, which implies that they know two sets of values, r_{1j} and r_{2j}, such that:

$$m_{1j} = Blind(a_1, r_{1j}, K_j) = a_1 \cdot r_{1j}^{K_j} (mod\ \ n_j)$$
$$m_{2j} = Blind(a_2, r_{2j}, K_j) = a_2 \cdot r_{2j}^{K_j} (mod\ \ n_j)$$

The other option is that they are able to show that there is no set of values s_{1j} capable of satisfying $m_{1j} = Blind(a_2, s_{1j}, K_j)$. However, given r_{ij}, users could have chosen s_{ij}, defined as $s_{ij} = \left(a_h^{-1} \cdot a_i\right)^{K_j^{-1}} \cdot r_{ij}$, for any other alias h. Specifically, if the values 1 and 2 are chosen for i and h:

$$Blind(a_2, s_{1j}, K_j) = a_2 \cdot s_{1j}^{K_j} = a_2 \cdot \left((a_2^{-1} \cdot a_1)^{K_j^{-1}} \cdot r_{1j}\right)^{K_j} (mod\ \ n_j) = Blind(a_1, r_{1j}, K_j) = m_{1j}$$
$$Blind(a_1, s_{2j}, K_j) = a_1 \cdot s_{2j}^{K_j} = a_1 \cdot \left((a_1^{-1} \cdot a_2)^{K_j^{-1}} \cdot r_{2j}\right)^{K_j} (mod\ \ n_j) = Blind(a_2, r_{2j}, K_j) = m_{2j}$$

In other words, depending on what values users U_1 and U_2 would have chosen, they could very well be the authors of either alias a_1 or alias a_2. There is no unique mathematical link between blinded messages and aliases. Consequently, adversaries have no way to distinguish one user from another with the information they have. □

As a result of this proposition, if content of messages do not reveal its authorship (e.g., yes/no votes) maximum privacy is preserved.

4.3.2. Eligibility

This refers to the ability of a system to determine whether users have the right to participate or not. Usually, this means that the user belongs to the list of registered users (census). To this end, somebody would have to impersonate a valid user at the starting phase or afterwards:

1. In the first case any adversary must show a digital certificate of a user featured on the list. This list is composed of the user who proposes the debate to the others, and in this phase users can check the list and choose to reject the debate. Thus, including a fake

user in the list requires deceiving all other users. The alternative is to use the digital certificate of a valid user. This requires cracking the RSA key of the certificate, to steal the digital certificate from the user, or to make a new certificate by using the private key of the third party. The first two options violate the assumptions of the threat model (keys are considered safe and user devices are free of malware). The third option deals with the trust on a third party. Any centralized voting system requires trust in at least one entity. Nevertheless, even this requirement can be avoided in this scheme. The solution is that each user makes their own key pair (as in a digital certificate), and then communicates the public one to the others by channels considered safe (phone calls, physical meetings, encrypted e-mails, etc.).

2. In the second case the level of security depends again on the strength of the RSA keys. Contributions (votes, proposals, and so on) are accepted when they came from a TLS channel built with a signed alias key. Therefore, any adversary who wants to impersonate another user must know the private alias key, which goes against the assumptions of the threat model. The alternative is to achieve acceptance for a fake alias, and this is not possible according to Lemma 1.

4.3.3. Verifiability

This is the user's ability to verify that their vote has been correctly recorded and accounted for in the final tally. There are two definitions [4]. One is individual verifiability where only the user can verify their vote in the tally. The second is universal verifiability where, after the tally has been published, anyone can verify that all valid votes were included and the tally process was accurate.

In this scheme every user has a file which fulfills the role of bulletin board with all the contributions. Users can compare their files in order to detect and add missed contributions (Section 3, phase 2, item 3). Further, in the final step everybody can add a signature when they have checked that all their contributions have been included. Thus, at the end of the process, everyone must have the same file, and users can check their contributions (individual verifiability) in it. On the other hand, fake contributions are avoided by the TLS channel security and the alias signature. It is not possible to sign a fake contribution of another user without knowing their private alias key. The owner of an alias is the only one capable of inserting contributions, but they will be associated with the same alias and will not affect the others. Thus, all the users can check every contribution and its signature, relate these contributions with its alias, and check if any alias has submitted more than one vote per proposal or any other irregularity (universal verifiability). The tally is computed by each user. Thus, there is individual and universal verifiability.

Verifiability is concerned with **accuracy**, which is mentioned in other papers as an additional feature. An e-voting system must be error free, that is, votes must be registered correctly and votes of invalid users must not be counted. The way to achieve this accuracy is by verifiability, such that if an error happens, it can be detected and corrected.

4.3.4. Dispute-Freeness

Voting methods must provide a mechanism to resolve disputes. Nevertheless this method avoids them because every user is in some respect the absolute manager of the whole process. In fact, every user can reject any received message. They have tools for verifying that every step by the other users is performed correctly, and for rejecting misbehavior. Ultimately, those who follow the rules have the capacity to learn the will of the other users who comply with them too.

4.3.5. Fairness

In order to conduct fair voting, no one should be able to compute a partial tally as the election progresses. This scheme is designed for debating tools, where knowing the most voted proposals (by showing a partial tally) could help the rest of the users to focus on the most important decisions for the group and be shielded from a flood of futile proposals.

Thus, fairness might not always be a desirable feature. However, an easy improvement can be made so that fairness is achieved. This consists of encrypting each vote with a symmetric alias key. When the period for contributions has finished (no more votes are accepted and the hash of the contributions list is signed by each alias) these symmetric keys are sent in a predefined interval, and votes can be decoded. Adversaries could wait to the end of the process to send their keys in order to have those of the rest. This would allow them to decrypt and know the tally before the rest, although they would not be able to change their vote in consequence. They could refuse to send their keys, but this would only exclude their votes from the tally. Therefore, with this improvement fairness is achieved.

4.3.6. Mobility

This scheme is designed to operate on the Internet. For this purpose, a digital certificate is needed. This can be signed by a trusted third party (a prestigious firm, a government office) or even among the users themselves. Once this certificate is signed, all security requirements can be achieved from any place in the world and multiple debates and decisions can be carried out with the same digital certificate. However, there are alternative means of managing identification, such as those based on blockchain [42].

4.3.7. Robustness

This parameter measures the strength against passive attacks, when corrupt users skip one or more steps of the protocol. At the first stage, if the alias owner does not follow the protocol, this user will lose the blind signature and the access to the signer in the second stage. If it is the signer who fails, then the affected user will publicly claim in step 6 and if the signer does not send a right signature he/she will be banned. At the second stage, if users acting as aliases fail, they will not be able to send their votes. If it is the known user who fails, he/she will not be able to know the votes of an eligible user. In any case, lack of action or misbehavior from any user does not stop the process for the rest and only affects the liable user. Nevertheless, a minimum number of signatures per alias is required, as shown in Lemma 2, namely there must be a minimum number of honest users who follow the protocol. Therefore robustness is limited to the maximum number of tolerable adversaries which is under $\sqrt{n+1}-1$ (Lemma 2).

4.3.8. Consistency

This is defined as the ability of honest users to obtain the same results. Due to the decentralized nature of this method, it could be possible that some users have different data than others and compute different tallies. However, this is not possible if adversaries do not reach $\sqrt{n+1}-1$ (Lemma 2). Therefore, consistency is conditional on this value.

4.3.9. Incoercibility

Incoercibility refers to the difficulties that an adversary finds when trying to buy users. Some researchers prefer the term "coercion-resistance" that measures coercer opportunities in comparison with a traditional system with no corrupt authorities [43]. Voting via the Internet means that any adversary can position themselves close to another user to observe what that user sees, and consequently check whether he/she follows their will. Therefore, there are no voting systems able to prevent coercion when mobility is imposed (no booths are used). However, some papers claim a degree of coercion resistance under certain constraints, such as untappable channels [4] or voters' refusal to share their private keys [44]. In this sense, the scheme presented here can show certain coercion resistance under certain conditions, such as inviolability of the user's equipment, restricted messages (yes/no votes), absence of coercers at the end of the process, and simultaneous sending of votes. Since these conditions are far from those that can be assumed in real environments, this scheme does not exhibit significant coercion resistance (as is the case in other boardroom e-voting schemes).

4.3.10. Flexibility

This parameter indicates how many conditions votes must fulfill. Many voting schemes need a fixed format for the ballots or at least a predefined list of candidates before the poll. This scheme does not require any of these. Rather, it provides a security layer over which any software can exchange data. All this data (proposals, candidates, votes) is protected without any constraint by an alias key in the same way as TLS channels provide security without restrictions on the application data by means of digital certificates. Therefore, this scheme achieves unconditional flexibility.

4.3.11. Scalability

Like other boardroom e-voting schemes, this method does not provide scalability. The reason is the number of exponentiations that users must compute for every member of the group. This is detailed below.

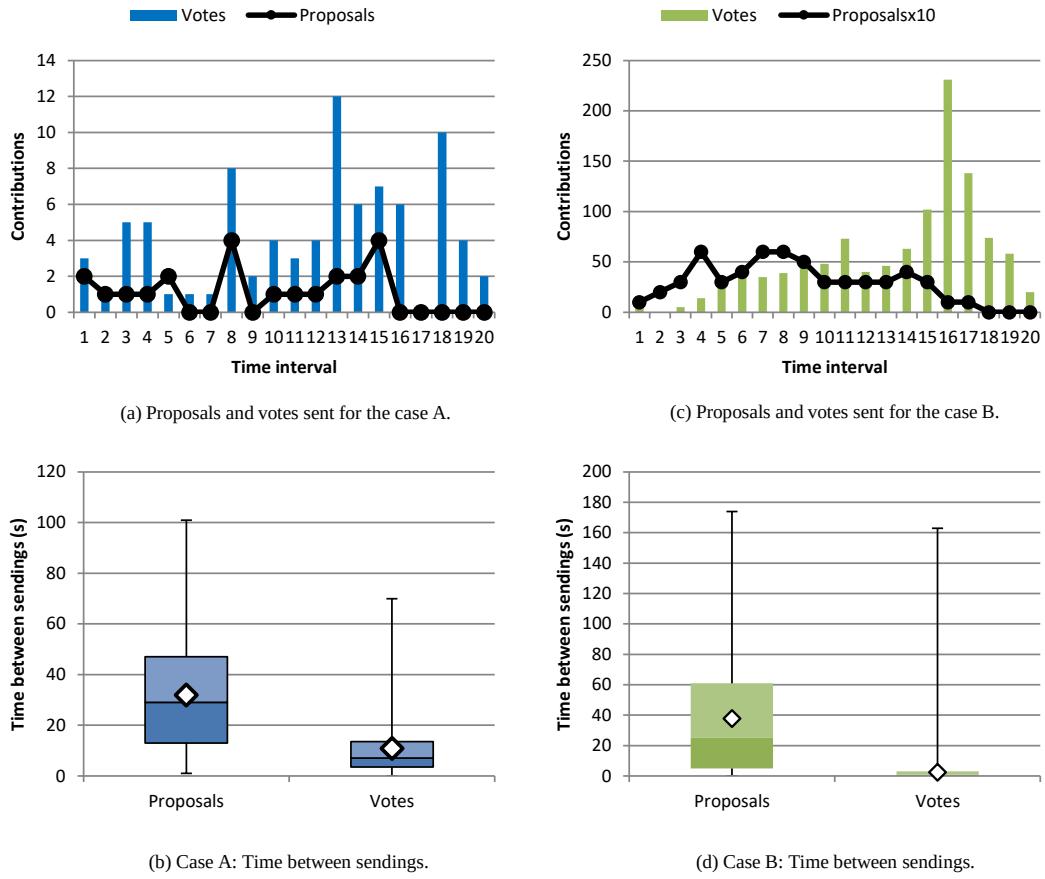
4.4. Time Complexity

Establishing a TLS channel requires one private and one public key encryption per user, which means two exponentiations per user and one more for verifying the digital certificate. Each pair of users will require one channel in phase 1 and two channels in phase 2 (as alias and as server). Blinding and checking implies two exponentiations on one side, while the signature on the other side consumes one more. These operations must be conducted with each user. Adding the signature and check per alias at the end, each user must compute $14(n-1)+1$ exponentiations, where n is the number of users. In addition to this, users must generate an alias key, but no knowledge proofs are required.

When comparing with previous boardroom e-voting methods, the fastest [21] requires only $3+N_c$ exponentiations [10], where N_c is the number of adversaries. However, in these schemes users must compute different types of knowledge proofs and verify them for every user. For example, ref. [21] requires N_c discrete logarithm equality proofs, which means $n-1+n-2+\ldots+n-N_c = N_c(n-(N_c+1)/2)$ verifications per user. Each check of this type implies ten exponentiations (Section 2.3 in [21]). Additionally users must compute a discrete logarithm for the tally. As claimed in [10] there are algorithms that can reduce this time, but at the cost of weaker checks [44]. Moreover, adversaries increase the number of rounds. By contrast, the method presented here does not depend on this number, because adversaries are automatically excluded when they do not follow the protocol.

5. Experiments and Discussion

The following experiments are oriented towards measuring the suitability of this method to carry out boardroom debates and voting. The debate tool has been selected on the basis of its features which include flexibility and the lack of a moderator [45]. In this debate tool, participants can send proposals, opinions, and votes. In this approach, only proposals and votes must be protected by the system. Figure 5 shows these kinds of contributions in relation to two different experiences. The first, case A, (charts (a) and (b) in Figure 5) corresponds to a group of 18 students who debated for 20 min. The second, case B, (charts (c) and (d) in Figure 5) was another debate carried out by 23 students in a period of 44 min. Charts (a) and (c) in Figure 5 show the activity in each interval related to sending proposals (black line) or sending votes (columns). Box plots (b) and (d) in Figure 5 show the average interval and its distribution between deliveries (votes and proposals) measured in each experience.



(a) Proposals and votes sent for the case A.

(c) Proposals and votes sent for the case B.

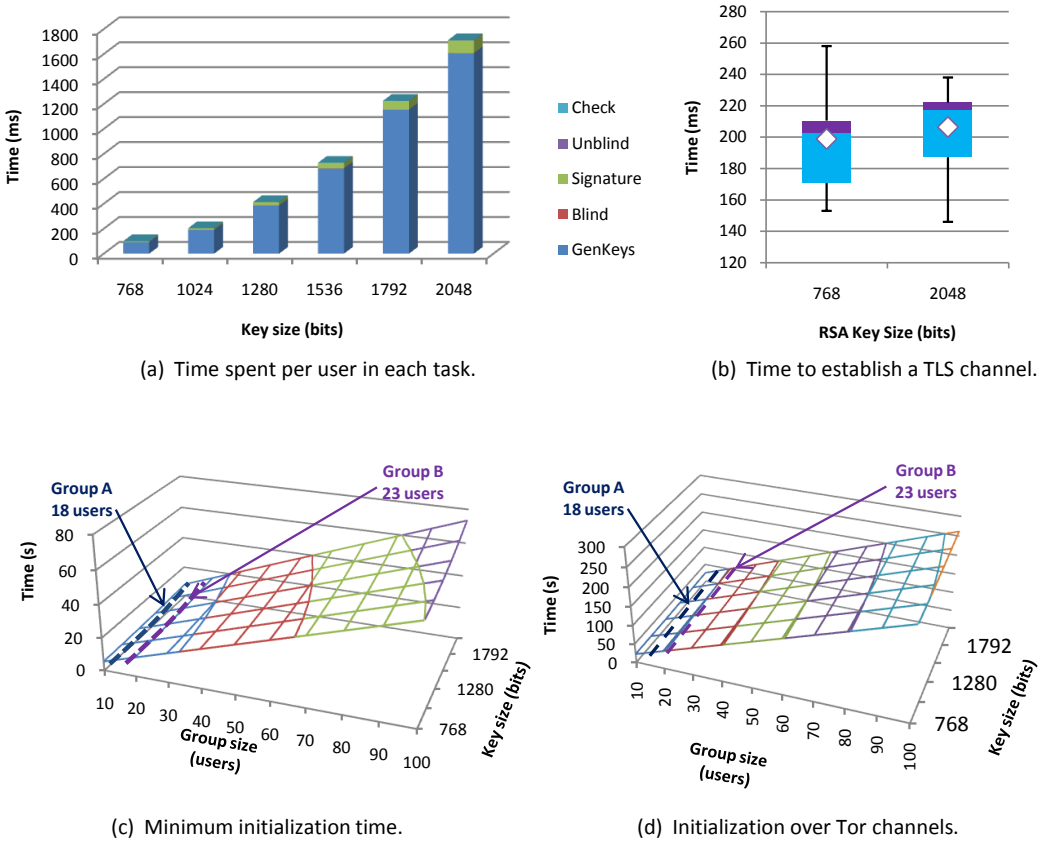
(b) Case A: Time between sendings.

(d) Case B: Time between sendings.

Figure 5. Debate experiences. Case A: 18 students. Case B: 23 students.

These two experiences offer an estimation of the activity in a debate group. Specifically, the average number of proposals from each user which could be voted on was 2.47. The boardroom voting system has to deal with this level of activity, so the process time will be analyzed in these scenarios. The first phase is to obtain the signing of the aliases, which implies generating the alias keys, blinding, signature, removing the blind factor, and checking. All of these steps were measured for only two users using different key sizes (from 768 bits to 2048 bits), and the result presented in chart (a) in Figure 6. These times were measured in computers with Intel Core i-3 processors, 3.1 GHz, and 4 GB of RAM. The chart shows that the heaviest processes were key generation and signature. Nevertheless, the slowest case takes only 1.71 s.

Additionally, all the traffic crosses a TLS channel between each pair of users. Chart (b) in Figure 6 presents the time needed to establish a TLS channel for two RSA key sizes. The dispersion of time intervals is caused by the different keys and the operating system (Windows 10) which had to manage many threads. The highlight of this chart is the obtention of a reference for two magnitudes: the actual duration of setting a TLS channel and the expected delay added by other common processes. Chart (c) reveals the time in seconds that each user took to initialize when there were more than two users (case of chart (a)). This increased following a curve (it is non-linear) as the key size increases, and following a straight line when group size increased. For instance, the group of 18 users (case A) generated a starting time from 10.38 s (key size of 768 bits) to 13.94 s (key size of 2048 bits). Case B varied from 13.4 s to 17.57 s. This increased with the number of users to frustratingly lengthy times (73.45 s for 100 users and 2048 bits).



(a) Time spent per user in each task.

(b) Time to establish a TLS channel.

(c) Minimum initialization time.

(d) Initialization over Tor channels.

Figure 6. Length of the cryptographic processes to be performed by each user (**a**), where some are so short that their color cannot be appreciated. Duration of setting a secure channel (**b**). Times required for phase 1 without considering anonymous channels (**c**) and considering TOR (**d**).

Chart (c) shows only the minimum time required in the protocol for initialization, without considering anonymous channels. These channels can add time or not (Section 2), and are selected according to the system features (delay, security, mobility, etc.). However, as an example, chart (d) shows the complete initialization time including the anonymous channels when they are based on the TOR network. Assuming three hops per circuit, the average establishing time is 1.9 s. This raises the initialization to a value of 261.6 s. in the worst case scenario (100 users and 2048 bits for RSA keys).

Nevertheless, this happens only at the beginning of the process. Once the debate is initialized, the messages are encrypted by symmetric keys (TLS channels), which means a negligible computing cost compared with usual network delays. The question now is how much traffic can be managed by common users. On the one hand the experimental results produced an average number of proposals per user (2.49), which provided the number of votes cast in a limited time (1920s). On the other hand users have a maximum bit rate to access the Internet. Assuming a limit of 50 Mbps and frames of 1500 bytes, the maximum number of users is 1796. That is, more users would increase the average time of a debate.

At the end of the process a signature is required. Each user must compute one and check $n - 1$ (faster in RSA). In total this took 200 ms in the worst case (100 users and 2048 key bits).

Summarizing these results for the two discussion groups mentioned above in the case of the most secure keys (2048 bits) and use of the TOR network, it is shown that group A would take 46.24 s to initialize, and group B 59.37 s. After that, the interaction of users in the debate would be fast (only 1.9 s delay in sending messages due to the TOR network). At the end, the signatures will take just 119 ms for group A and 124 ms for group B. It can be seen that these initialization times of less than one minute are acceptable and that the scheme is therefore suitable for groups of this size.

6. Conclusions

The following is a summary of the most relevant achievements and key aspects of this document. In order to clarify them, it is necessary to explain the starting point. In the situation prior to this work, flexible and secure debating in a decentralized manner was not possible. In fact, for simple decentralized voting there were very few published systems. The proposed scheme broadens the range of boardroom e-voting systems. It provides unconditional flexibility because no constraints are imposed on the vote format; as a consequence this scheme is independent of any debate software. Moreover, other interactions, such as proposals and opinions, can also be protected, maintaining privacy, eligibility, verifiability, and the rest of features. This makes it suitable for other uses like debates or gathering criticism from a group anonymously.

Regarding BFTs systems, in the case of our scheme, it is feasible to use them in phase 2, when anonymity problem is solved. Thus, any of the leaderless (P2P) BFT systems like DBFT [46], Honey Badger [47], or BEAT [15] could be selected instead of the alias signature used to achieve consensus. However, it implies the loss of verifiability, which is a desirable feature in e-voting systems.

The scheme does not include any step that requires the involvement of all the participants, so it is robust against adversaries inside the group, and they cannot block the process.

A flexible debate can be compared to previous voting systems from two points of view:

1. Each proposal is an independent voting. Then, the user who sends the proposal is the one who defines the ballot frame. In a debate there are many proposals; therefore, there will be many polls. Thus, a comparison between our voting system and others results in the advantage of making multiple polls with the computational cost of just one.
2. All the proposals are in the same voting. Then, we are considering a unique ballot that contains the votes for each proposal. This ballot can be huge and extremely complex, a problem for some schemes as was explained in Section 1. Even more, for other e-voting systems, this ballot must be defined by an initiator. In other words, users have not a secure way to choose what to vote.

The experiments and analysis show the ratio between group size, key size, and process times. These results are based on two specific study cases; therefore, they cannot be generalized. However, they do illustrate an actual example of debates.

The main constraint of this scheme, as in any boardroom system, is the group size. However, this alias approach suggests that a hierarchical structure of nets is possible. Thus, future work will be based on this idea to improve scalability.

Author Contributions: Conceptualization, D.A.L.-G.; Methodology, D.A.L.-G.; Literature review, D.A.L.-G., J.P.T., D.V. and M.S.-R.; Software, D.A.L.-G., J.P.T., D.V. and M.S.-R.; Validation, J.P.T., D.V. and M.S.-R.; Writing—original draft preparation, D.A.L.-G.; writing—review and editing, D.A.L.-G., J.P.T., D.V. and M.S.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets at https://www.uhu.es/diego.lopez/research/dataP2Pevoting1.xlsx (accessed on 13 March 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Neji, W.; Blibech, K.; Rajeb, N.B. A Survey on e-voting protocols based on secret sharing techniques. In Proceedings of the CARI 2018, Stellenbosch, South Africa, 14–16 October 2018; p. 142.
2. Sampigethaya, K.; Poovendran, R. A framework and taxonomy for comparison of electronic voting schemes. *Comput. Secur.* **2006**, *25*, 137–153. [CrossRef]
3. Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **1981**, *24*, 84–90. [CrossRef]
4. Sako, K.; Kilian, J. Receipt-free mix-type voting scheme. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Saint-Malo, France, 21–25 May 1995; pp. 393–403.
5. Jakobsson, M.; Juels, A.; Rivest, R.L. Making mix nets robust for electronic voting by randomized partial checking. In Proceedings of the 11th USENIX Security Symposium (USENIX Security 02), San Francisco, CA, USA, 5–9 August 2002.
6. Sampigethaya, K.; Poovendran, R. A survey on mix networks and their secure applications. *Proc. IEEE* **2006**, *94*, 2142–2181. [CrossRef]
7. Stathakidis, E. Formal Modeling and Analysis of Mix Net Implementations. Ph.D. Dissertation, University of Surrey, Guildford, UK, 2015.
8. Acquisti, A. Receipt-Free Homomorphic Elections and Write-in Ballots. IACR Cryptology ePrint Archive. 2004; p. 105. Available online: https://ia.cr/2004/105 (accessed on 18 April 2024).
9. Zahhafi, L.; Khadir, O. A Fast Cryptographic Protocol for Anonymous Voting. *MathLAB J.* **2018**, *1*, 89–99.
10. Khazaei, S.; Rezaei-Aliabadi, M. A rigorous security analysis of a decentralized electronic voting protocol in the universal composability framework. *J. Inf. Secur. Appl.* **2018**, *43*, 99–109. [CrossRef]
11. Nakajima, A. Decentralized voting protocols. In Proceedings of the ISADS 93: International Symposium on Autonomous Decentralized Systems, Kawasaki, Japan, 30 March–1 April 1993; pp. 247–254.
12. Hardekopf, B.; Kwiat, K.; Upadhyaya, S. A decentralized voting algorithm for increasing dependability in distributed systems. In Proceedings of the 5th World Multi-Conference on Systemic, Cybernetics and Informatics (SCI2001), Orlando, FL, USA, 22–25 July 2001.
13. Bocek, T.; Peric, D.; Hecht, F.; Hausheer, D.; Stiller, B. PeerVote: A Decentralized Voting Mechanism for P2P Collaboration Systems. In Proceedings of the Scalability of Networks and Services: Third International Conference on Autonomous Infrastructure, Management and Security, AIMS 2009, Enschede, The Netherlands, 30 June–2 July 2009; Proceedings 3, pp. 56–69.
14. Tseng, L. Recent Results on Fault-Tolerant Consensus in Message-Passing Networks. In Proceedings of the Structural Information and Communication Complexity: 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, 19–21 July 2016; Revised Selected Papers 23; Springer International Publishing: Berlin/Heidelberg, Germany; pp. 92–108.
15. Duan, S.; Reiter, M.K.; Zhang, H. BEAT: Asynchronous BFT made practical. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 2028–2041.
16. Chen, J.H.; Chen, M.R.; Zeng, G.Q.; Weng, J.S. BDFL: A byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle. *IEEE Trans. Veh. Technol.* **2021**, *70*, 8639–8652. [CrossRef]
17. Kulyk, O.; Neumann, S.; Budurushi, J.; Volkamer, M.; Haenni, R.; Koenig, R.; von Bergen, P. Efficiency Evaluation of Cryptographic Protocols for Boardroom Voting. In Proceedings of the 10th International Conference on Availability, Reliability and Security (ARES), Toulouse, France, 24–28 August 2015; pp. 224–229.
18. Kiayias, M.; Yung, M. Self-tallying elections and perfect ballot secrecy. In Proceedings of the International Workshop on Public Key Cryptography, Paris, France, 12–14 February 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 141–158.
19. Groth, J. Efficient maximal privacy in boardroom voting and anonymous broadcast. In Proceedings of the International Conference of Financial Cryptography, Key West, FL, USA, 9–12 February 2004; pp. 90–104.
20. Hao, F.; Ryan, P.Y.; Zielinski, P. Anonymous voting by two-round public discussion. *IET Inf. Secur.* **2010**, *4*, 62–67. [CrossRef]
21. Khader, D.; Smyth, B.; Ryan, P.; Hao, F. A fair and robust voting system by broadcast. In Proceedings of the Lecture Notes in Informatics (LNI), Proceedings-Series of the Gesellschaft fur Informatik (GI), Bregenz, Austria, 11–14 July 2012; pp. 285–299.
22. Kulyk, O.; Neumann, S.; Volkamer, M.; Feier, C.; Koster, T. Electronic voting with fully distributed trust and maximized flexibility regarding ballot design. In Proceedings of the 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), Bregenz, Austria, 29–31 October 2014; pp. 1–10.
23. Abuidris, Y.; Kumar, R.; Wenyong, W. A Survey of Blockchain Based on E-voting Systems. In Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications, Xi'an, China, 9–11 December 2019; pp. 99–104.
24. Lee, K.; James, J.I.; Ejeta, T.T.; Kim, H.J. Electronic voting service using block-chain. *J. Digit. Forensics Secur. Law* **2016**, *11*, 8. [CrossRef]
25. Yavuz, E.; Koç, A.K.; Çabuk, U.C.; Dalkılıç, G. Towards secure evoting using ethereum blockchain. In Proceedings of the 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–7.
26. Hsiao, J.H.; Tso, R.; Chen, C.M.; Wu, M.E. Decentralized e-voting systems based on the blockchain technology. In Proceedings of the Advances in Computer Science and Ubiquitous Computing: CSA-CUTE 17, Kuala Lumpur, Malaysia, 17–19 December 2018; pp. 305–309.
27. Hjálmarsson, F.Þ.; Hreiðarsson, G.K.; Hamdaqa, M.; Hjálmtýsson, G. Blockchain-based e-voting system. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 983–986.

28. Yi, H. Securing e-voting based on blockchain in p2p network. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 137. [CrossRef]
29. Khan, K.M.; Arshad, J.; Khan, M.M. Secure digital voting system based on blockchain technology. *Int. J. Electron. Gov. Res. (IJEGR)* **2018**, *14*, 53–62. [CrossRef]
30. López-García, D.A. A flexible e-voting scheme for debate tools. *Comput. Secur.* **2016**, *56*, 50–62. [CrossRef]
31. Boyd, C. A new multiple key cipher and an improved voting scheme. In Proceedings of the Advances in Cryptology, EUROCRYPT'89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, 10–13 April 1989; pp. 617–625.
32. Chaum, D. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In Proceedings of the Advances in Cryptology, EUROCRYPT'88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, 25–27 May 1988; Volume 330, pp. 177–182.
33. Chaum, D. Blind signature system. In Proceedings of the Advances in Cryptology: Proceedings of Crypto 83, Boston, MA, USA, 21–24 August 1983; p. 153.
34. Bellare, M.; Namprempre, C.; Pointcheval, D.; Semanko, M. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *J. Cryptol.* **2003**, *16*, 185–215. [CrossRef]
35. Ford, B.; Srisuresh, P.; Kegel, D. Peer-to-Peer Communication Across Network Address Translators. In Proceedings of the USENIX Annual Technical Conference, General Track, Anaheim, CA, USA, 10–15 April 2005; pp. 179–192.
36. Chaum, D. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.* **1988**, *1*, 65–75. [CrossRef]
37. Knight, P.; Lewis, C. Layer 2 and 3 virtual private networks: Taxonomy, technology, and standardization efforts. *IEEE Commun. Mag.* **2004**, *42*, 124–131. [CrossRef]
38. Dingledine, R.; Mathewson, N.; Syverson, P.F. Tor: The Second-Generation Onion Router. In Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA, 9–13 August 2004; Volume 4, pp. 303–320.
39. Herrmann, M.; Grothoff, C. Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using I2P. In Proceedings of the Privacy Enhancing Technologies: 11th International Symposium, PETS 2011, Waterloo, ON, Canada, 27–29 July 2011; Proceedings 11, pp. 155–174.
40. Rozenman, G.G.; Kundu, N.K.; Liu, R.; Zhang, L.; Maslennikov, A.; Reches, Y.; Youm, H.Y. The quantum internet: A synergy of quantum information technologies and 6G networks. *IET Quantum Commun.* **2023**, *4*, 147–166. [CrossRef]
41. Perepechaenko, M.; Kuang, R. Quantum encryption of superposition states with quantum permutation pad in IBM quantum computers. *EPJ Quantum Technol.* **2023**, *10*, 7. [CrossRef]
42. Liu, Y.; He, D.; Obaidat, M.S.; Kumar, N.; Khan, M.K.; Choo, K.K.R. Blockchain-based identity management systems: A review. *J. Netw. Comput. Appl.* **2020**, *166*, 102731. [CrossRef]
43. Wu, Z.Y.; Wu, J.C.; Lin, S.C.; Wang, C. An electronic voting mechanism for fighting bribery and coercion. *J. Netw. Comput. Appl.* **2014**, *40*, 139–150. [CrossRef]
44. Bellare, M.; Garay, J.A.; Rabin, T. Fast batch verification for modular exponentiation and digital signatures. In Proceedings of the EUROCRYPT'98, Helsinki, Finland, 31 May–4 June 1998; pp. 236–250.
45. López-García, D.A.; Sanguino, T.J.M.; Ancos, E.C.; de Viana González, I.F. A debate and decision-making tool for enhanced learning. *IEEE Trans. Learn. Technol.* **2016**, *9*, 205–216. [CrossRef]
46. Crain, T.; Gramoli, V.; Larrea, M.; Raynal, M. DBFT: Efficient leaderless Byzantine consensus and its application to blockchains. In Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA USA, 1–3 November 2018; pp. 1–8.
47. Miller, A.; Xia, Y.; Croman, K.; Shi, E.; Song, D. The honey badger of BFT protocols. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 31–42.