Privacy-Preserving Decision-Making over Blockchain

Jiajie Zhang, Bingsheng Zhang, Andrii Nastenko, Hamed Balogun, Roman Oliynykov

Abstract—Many blockchain applications require democratic on-chain decision-making. In this work, we propose a community-inclusive decentralised collaborative decision-making system with privacy assurance. Its key component is a two-stage voting scheme inspired by choice architecture. Our decision-making system is compatible with most existing blockchain infrastructures. In addition, it supports liquid democracy/delegative voting for better collaborative intelligence. Namely, stake holders can either vote directly on proposals or delegate their voting power to experts. When majority of voting committee members are honest, no one can derive voters' voting preferences or delegations with non-negligible probability. To support concurrent multiple voting events, we design a distributed batch key generation protocol that can generate multiple keys simultaneously by voting committee members with amortised communication cost of $\mathcal{O}(n)$ per key, where n is the number of participants. Besides, our system supports "evolving committee", i.e., voting committee members can be changed during the voting period. We implemented a pilot system in Scala, benchmark results indicate that our system can support large number of participants with high efficiency.

Index Terms—Decision making, Blockchain, Distributed batch key generation, Two-stage voting.

1 Introduction

Blockchain technology enables an opportunity for a dispersion of power and information, as well as profound possibilities for large-scale collaboration in modern digital society. The primary advantage of blockchain is its decentralisation realised by distributing power and value across the entire network, which makes information and value exchange more efficient and equitable without a central third party. This decentralisation feature makes collaborative intelligence over blockchain promising.

Recently, many decentralised on-chain decision-making systems have been developed. For example, it solves the sustainable funding issue for blockchain development and maintenance in treasury system [1]; it attests to external data in order to bring it onto the blockchain in decentralised oracle with robust security guarantee [2], [3]; it enables decentralised crowd sourcing for forecasting in decentralised prediction market [4]; it resolves the software development problem in decentralised blockchain governance system [5], [6], [7].

In particular, the treasury system proposed by Zhang *et al*. [1] is one of the few provably secure on-chain decision-making systems. It manages funding and distributes power to edges to empower the community of individuals, as an enabling force for change and progress about the blockchain platform. However, their system has several downsides.

- It uses the Distributed Key Generation (DKG) protocol by Gennaro *et al*. [8] for each voting event. The overall communication of this DKG protocol is $\mathcal{O}(n^2)$ per key, where n is the number of participants; therefore, the committee size cannot scale;
- In their voting protocol, the voters and/or experts need to make decision on every submitted proposal; it requires too much voting effort when the number of proposed proposals becomes large;
- Voting committee needs to hold their key shares during the entire voting period, which could take 1-3 months in practice; such design increases the risk of system failure;
- This proposed treasury system does not explicitly support participatory budgeting (PB, [9], [10]).

In this paper, we preserve the liquid democracy feature of Zhang *et al*.'s work, while resolving all above downsides.

To address the first downside, we propose a new *Distributed Batch Key Generation (DBKG)* protocol based on Hyper Invertible Matrix (HIM) [11]. Our DBKG protocol can generate multiple keys simultaneously, achieving amortised complexity of $\mathcal{O}(n)$ per key. The generated public keys are used to encrypt the ballots of voters and experts and keep ballots private. With threshold encryption and non-interactive zero knowledge proofs, only the delegated voting power of experts and final tally results are revealed at the end of voting. Therefore, our system can assure ballots privacy [12] and guarantee end-to-end verifiability [13].

To address the second downside, we propose a new *Two Stage Voting (TSV)* scheme to reduce the community's voting effort, which is inspired from choice architecture [14]. In the first stage (preferential voting), voters and experts publicly announce their encrypted preferences. Depending on fund availability, a shortlist is produced. In the second stage (threshold voting), voters and experts vote Yes/No/Abstain on each shortlisted proposal; proposals

- *J. Zhang is with the School of Computing and Communication, Lancaster University, United Kingdom. E-mail: j.zhang41@lancaster.ac.uk*
- *B.Zhang as the corresponding author is with Input Output HK and College of Computer Science and Technology, Zhejiang University. E-mail: bingsheng@zju.edu.cn*
- *A. Nastenko is with Input Output HK. E-mail: andrii.nastenko@iohk.io*
- *H. Balogun is with School of Psychology and Computer Science, University of Central Lancashire. E-mail: hbalogun1@uclan.ac.uk*
- *R. Oliynykov is with Input Output HK and V. N. Karazin Kharkiv National University. E-mail: roman.oliynykov@iohk.io*

that receives more than threshold supports are funded.

To address the third downside, our system supports "evolving-committee" [15]. Periodically, a new committee is selected by cryptographic sortition [15], [16], and the secret keys are re-shared to the new committee while keeping the public keys unchanged.

To address the fourth downside, the proposed system supports participatory budgeting. Namely, proposals, voters and experts are associated with fields. Each field has its own fixed budget, the shortlist and winning proposals are tallied independently of the other fields.

We analyse the security of our proposed protocols under the Universal Composability (UC) framework [17]. In addition, we also provide prototype implementation of the proposed decision-making system in Scala programming language for benchmarking in the real world environment. All implemented protocols are fully decentralised and are resilient up to 50% of malicious participants. We launched a testnet comprising a dozen full nodes successfully operating hundreds of polling periods with different parameters.

2 PRELIMINARY

2.1 Universal Composability

We prove the security of our protocols under the Universal Composability (UC) framework [17], which is reflected in the indistinguishability of environment $\mathcal{Z}$'s views in real world execution and in ideal world execution. In the UC framework, we use ideal functionality to represent the desired security properties of a protocol, which can be considered as an absolutely honest third party carrying out the task by definition in ideal world. Ideal functionality explicitly captures adversary's influence and knowledge it can gain from the protocol execution in real world. We use $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ to represent environment $\mathcal{Z}$'s output in real world when it interacts with parties running protocol Π and real-world adversary $\mathcal{A}$, while $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is environment $\mathcal{Z}$'s output in ideal world, in which it interacts with ideal functionality $\mathcal{F}$ and ideal adversary $\mathcal{S}$.

Definition 1 (UC Realisation [17]). *We say that a protocol Π UC-realises $\mathcal{F}$ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ that obeys the rules of interaction for UC security we have $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.*

Theorem 1 (Composition Theorem [17]). *For any protocol Π, with the realised functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model, $\mathcal{F}$ may be used as a subroutine, and the composed protocol ρ^{π} replacing $\mathcal{F}$ with a secure protocol π also securely realises $\mathcal{G}$ in the real world.*

We follow the synchronous communication model defined in Canetti's work [17], where the computation is proceeded in rounds. In each round, parties receive the messages sent in last round and they send out messages for next round. We model parties in our system as Interactive Turing Machines (ITMs). Every party P_i has a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, all messages sent to blockchain by P_i are signed by sk_i. With a slight abuse of notation, P_i is used to represent both the machine itself and its identity. Additionally, we set sid as a unique session identifier to specify which session an ITM belongs to.

2.2 Homomorphic Public Key Encryption

To support efficient zero knowledge proofs and tally computation, we use homomorphic public key encryption to perform additively homomorphic computation on encrypted messages without decryption. We define group generator by $\mathsf{Gen}(1^{\kappa})$, which takes security parameter κ as input, and outputs group parameter param. $\mathsf{Gen}(1^{\kappa})$ defines a multiplicative cyclic group $\mathbb{G}$ with prime order q with generator g, where $|q| = \kappa$.

A public key encryption scheme $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is additively homomorphic if for all security parameters κ and all key pairs $(\mathsf{pk}, \mathsf{sk})$, it is possible to define a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$ such that for any $(m_1, m_2) \in \mathcal{M}$ and $(c_1, c_2) \in \mathcal{C}$ with $m_1 := \mathsf{Dec}_{\mathsf{sk}}(c_1)$ and $m_2 := \mathsf{Dec}_{\mathsf{sk}}(c_2)$, it holds that $\{\mathsf{pk}, c_1, c_2, c_1 \cdot c_2\} = \{\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_1), \mathsf{Enc}_{\mathsf{pk}}(m_2), \mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2)\}$.

We use Lifted ElGamal encryption scheme [18] as the candidate of additively homomorphic PKE, which is IND-CPA secure under the Decisional Diffie-Hellman (DDH) assumption. It consists of the following four algorithms:

- $\mathsf{Gen}(\mathsf{param})$: pick $\mathsf{sk} \leftarrow (\mathbb{Z}_q)^*$, set $\mathsf{pk} := g^{\mathsf{sk}}$, output $(\mathsf{pk}, \mathsf{sk})$;
- $\mathsf{Enc}_{\mathsf{pk}}(m; r)$: pick $r \leftarrow (\mathbb{Z}_q)^*$, output (c_1, c_2), where $c_1 := g^r$, $c_2 := g^m \cdot \mathsf{pk}^r$;
- $\mathsf{Add}((c_{1,1}, c_{1,2}), \ldots, (c_{\ell,1}, c_{\ell,2}))$: output (C_1, C_2), where $C_1 := \prod_{i=1}^{\ell} c_{i,1}$, $C_2 := \prod_{i=1}^{\ell} c_{i,2}$;
- $\mathsf{Dec}_{\mathsf{sk}}(c_1, c_2)$: output $\mathsf{Dlog}(c_2 \cdot c_1^{-\mathsf{sk}})$, where $\mathsf{Dlog}(x)$ is the discrete logarithm of x.

In the rest of the paper, for simplicity, given the key pair $(\mathsf{pk}, \mathsf{sk})$, we use $(\mathbf{A}, \mathbf{A}') = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}; \mathbf{r})$ and $\mathbf{m} = \mathsf{Dec}_{\mathsf{sk}}(\mathbf{A}, \mathbf{A}')$ for encryption and decryption operations of a vector. To support computing discrete logarithm for Lifted ElGamal decryption, we split message into a vector. Let p be a perfect power of 2, ℓ be an integer parameter, denote vector $\mathbf{d}^{[\ell]}$ by $\{d_1, \ldots, d_\ell\}$, and denote vector $\mathbf{p}^{[\ell]}$ by $\{p, \ldots, p^{\ell}\}$. Message m is mapped to smaller message space by $\mathbf{d}^{[\ell]} = \mathsf{encode}(m)$, where $\mathbf{d}^{[\ell]} \cdot \mathbf{p}^{[\ell]} = m$, corresponding decoding function is denoted by $m = \mathsf{decode}(\mathbf{d}^{[\ell]})$.

2.3 Pedersen Commitment

We use Pedersen Commitment [19] in DKG protocols and zero knowledge proofs to commit messages, which has the following four algorithms:

- $\mathsf{Setup}^{\mathsf{C}}(\mathsf{param})$: pick $s \leftarrow \mathbb{Z}_q$, set $\mathsf{ck} := g^s$, output ck;
- $\mathsf{Com}_{\mathsf{ck}}(m; r)$: pick $r \leftarrow \mathbb{Z}_q$, output $c := g^m \cdot \mathsf{ck}^r$;
- $\mathsf{Open}(c)$: output $d := (m, r)$;
- $\mathsf{Verify}^{\mathsf{C}}(c, d)$: return 1 if and only if $c = g^m \cdot \mathsf{ck}^r$.

Pedersen commitment is perfectly hiding and computationally binding. Moreover, it is additively homomorphic, where $\mathsf{Com}_{\mathsf{ck}}(m_1; r_1) \cdot \mathsf{Com}_{\mathsf{ck}}(m_2; r_2) = \mathsf{Com}_{\mathsf{ck}}(m_1 + m_2; r_1 + r_2)$.

2.4 Σ protocol

Σ protocol [20] is a 3-round interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. We use Σ protocol to construct non-interactive zero knowledge proofs, which are used in threshold encryption and ballots validation without conveying any additional information except the statements.

Let $\mathcal{R}$ be a polynomial time decidable binary relation, w is said to be a witness for a statement x, if $(x, w) \in \mathcal{R}$. We define the language $\mathcal{L} := \{x \mid \exists w : (x, w) \in \mathcal{R}_{\mathcal{L}}\}$ as the set of all statements x that have a witness w for the relation $\mathcal{R}$. In the protocol, $\mathcal{P}$ needs to convince $\mathcal{V}$ that it knows some value w related to a common input x by revealing nothing except only w. In the first round, $\mathcal{P}$ commits x to c and send it to $\mathcal{V}$; in the second round, $\mathcal{V}$ sends a random challenge e back; in the last round, $\mathcal{P}$ responds to $\mathcal{V}$ by sending back a response z related to e and c, $\mathcal{V}$ decides to either accept or reject. A Σ protocol is perfect complete, special sound, and special honest verifier zero-knowledge.

2.5 Hyper-Invertible Matrix

Hyper-Invertible Matrix (HIM) [11] is a matrix that any square submatrix formed by removing rows and columns of this matrix is invertible.

Definition 2 (Hyper-Invertible Matrix [11])**.** *An $m \times n$ matrix $\mathbf{M}$ is a hyper-invertible matrix if for any index sets $\mathbf{I} \subseteq \{1, \ldots, m\}$ and $\mathbf{O} \subseteq \{1, \ldots, n\}$ with $|\mathbf{I}| = |\mathbf{O}| > 0$, the matrix $\mathbf{M}_{\mathbf{O}}^{\mathbf{I}}$ is invertible of which the rows $i \in \mathbf{I}$ and columns $j \in \mathbf{O}$.*

Theorem 2 (Hyper-Invertible Matrix Mapping [11])**.** *Let $\mathbf{M}$ be a $n \times n$ hyper-invertible Matrix, given $(y_1, \ldots, y_n) = \mathbf{M} \cdot (x_1, \ldots, x_n)$, for any index sets $\mathbf{A}, \mathbf{B} \subseteq \{1, \ldots, n\}$ where $|\mathbf{A}| + |\mathbf{B}| = n$, there exists an invertible liner function which can map $(\{x_i\}_{i \in \mathbf{A}}, \{y_i\}_{i \in \mathbf{B}})$ to the values $(\{x_i\}_{i \neq \mathbf{A}}, \{y_i\}_{i \neq \mathbf{B}})$.*

HIM can be used to share randomness. In our DBKG protocol, we use HIM to replace the Shamir secret sharing used in Gennaro *et al.*'s work [8] to achieve lower complexity. Let t be the threshold, applying n sharings to a $n \times n$ hyper-invertible matrix results in n sharings with the following properties:

Property 1. *[11] If any (up to t) of the inputs sharings are broken, then this can be seen in every subset of t output sharings.*

Property 2. *[11] If any $n - t$ input sharings are uniformly random, then every subset of size $n - t$ of output sharings is uniformly random.*

Construction 1. *Given fixed elements $\{\alpha_i\}_{i=1}^n$ and $\{\beta_i\}_{i=1}^n$ from $\mathbb{Z}_q$, let $F(z)$ be a polynomial such that $F(\alpha_i) := x_i$ for $i \in [n]$, which maps $\{x_i\}_{i=1}^n$ to $\{y_i\}_{i=1}^n$ by $y_i = F(\beta_i)$ for $i \in [n]$. $\{y_i\}_{i=1}^n$ can be computed by*

$$y_i = F(\beta_i) = \sum_{j=1}^{n} \prod_{k=1, k \neq j}^{n} \frac{\beta_i - \alpha_k}{\alpha_j - \alpha_k} \cdot x_j = \sum_{j=1}^{n} \lambda_{i,j} \cdot x_j,$$

where $\{\lambda_{i,j}\}_{i=1,j=1}^{n,n}$ is a hyper-invertible matrix $\mathbf{HIM}$.

3 System Overview

The proposed decision-making system runs in iterative periods, and provides public deliberation and accountability by cryptographic protocols. As shown in Fig. 1, one period consists of three epochs: *Pre-Voting Epoch*, *Voting Epoch*, and *Post-Voting Epoch*. Each epoch takes multiple rounds, and one round may take several blocks.

3.1 Security Model and Design Goals

There are four types of parties in our system, including project proposers, voters, experts and voting committee members. Proposers, voters, and experts are not fully trusted, and their behaviours shall be verified in the system. In addition, at most $1/2 \cdot c - 1$ of the voting committee members can be compromised, where c is the number of voting committee members. In particular, w.r.t. the voting protocol, we consider static corruption, where adversary should decide which parties to corrupt before executing protocol. Security is modelled under the UC framework by proving indistinguishability between ideal world execution and real world execution. We assume the underlying blockchain infrastructure is trusted.

The goal of this work is to design a privacy-preserving decision-making system over blockchain. More specifically, we aim at satisfying the following objectives [12], [21], [22], [23], [24]

- **Privacy.** Voter's voting preference and delegation choice should be kept secret;
- **Fairness.** All the participants gain equal treatment regarding to the receipt of information and involvement in a fair and neutral voting/proposing process. Namely, no proposers, voters or experts have the advantage to revise their decisions based on peers' outputs and published results;
- **Flexibility and Efficiency.** As voting process takes relatively long time, voting committee members should be allowed to work flexibly. In addition, we aim to minimise the communication cost and improve overall efficiency as much as possible;
- **End-to-end Verifiability.**
  - **Individual Verifiability.** Voters and experts can verify if their ballots are published correctly and included in the final tally result;
  - **Universal Verifiability.** Anyone can verify the fairness of voting and correctness of final tally result based on submitted ballots;
  - **Eligibility Verifiability.** Only registered and valid parties can submit ballots/proposals, each voter/expert can only submit one valid ballot. Anyone can verify that if the finally tally result comes from the valid ballots and eligible voters.

3.2 Participatory Budgeting

We categorise proposals, voters and experts into different fields. Each field has its own budget, and is voted individually from other fields. Taking blockchain development funding as an example [25], budget could be divided into the following fields:

- *Marketing*. Activities devoted to cryptocurrency market share growth. *e.g.*, market analysis, advertisement, conferences, etc;
- *Technology adoption*. Costs needed for wider spreading of cryptocurrency. *e.g.*, integration with various platforms, websites and applications, etc;

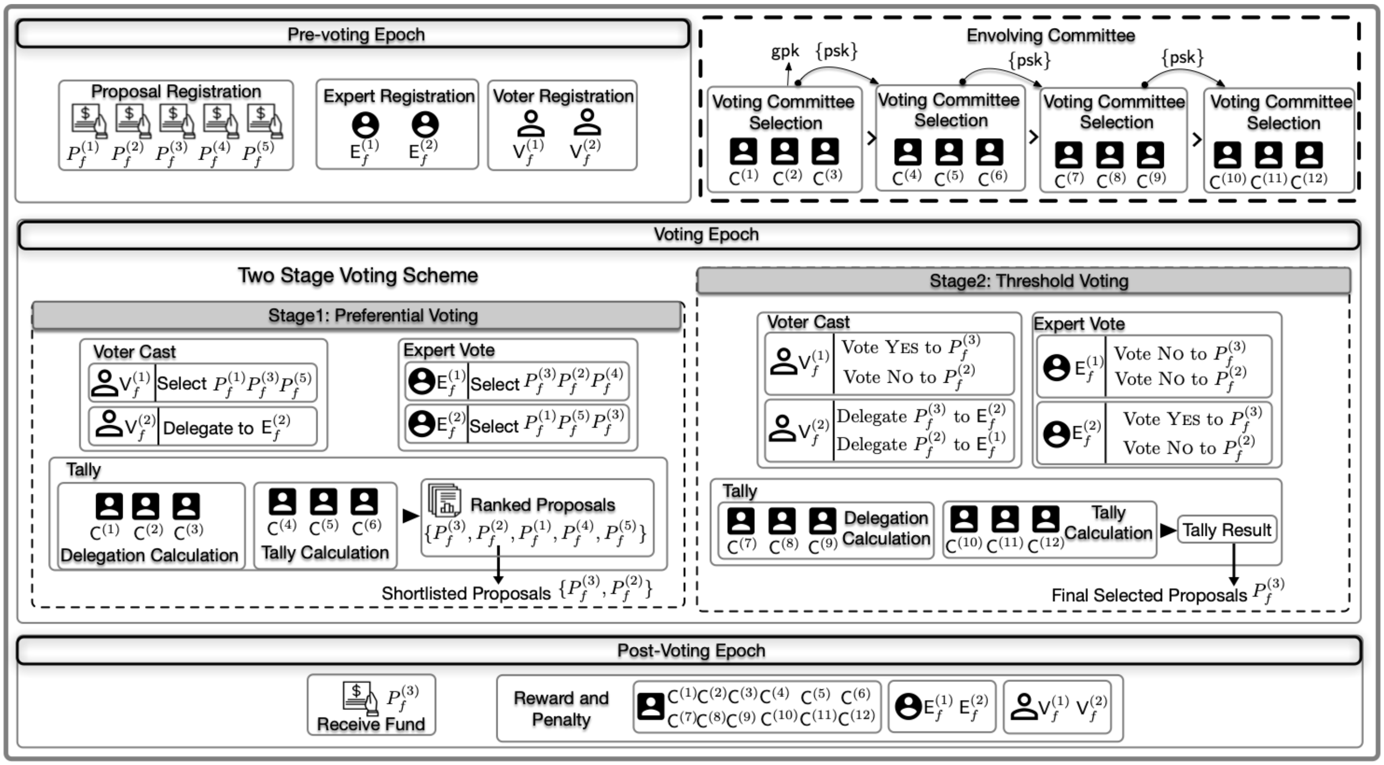

Pre-voting Epoch
Proposal Registration
$P_f^{(1)}$ $P_f^{(2)}$ $P_f^{(3)}$ $P_f^{(4)}$ $P_f^{(5)}$
Expert Registration
$E_f^{(1)}$ $E_f^{(2)}$
Voter Registration
$V_f^{(1)}$ $V_f^{(2)}$
Envolving Committee
gpk {psk}
{psk}
{psk}
Voting Committee Selection
$C^{(1)}$ $C^{(2)}$ $C^{(3)}$
Voting Committee Selection
$C^{(4)}$ $C^{(5)}$ $C^{(6)}$
Voting Committee Selection
$C^{(7)}$ $C^{(8)}$ $C^{(9)}$
Voting Committee Selection
$C^{(10)}$ $C^{(11)}$ $C^{(12)}$
Voting Epoch
Two Stage Voting Scheme
Stage1: Preferential Voting
Voter Cast
$V_f^{(1)}$ Select $P_f^{(1)}P_f^{(3)}P_f^{(5)}$
$V_f^{(2)}$ Delegate to $E_f^{(2)}$
Expert Vote
$E_f^{(1)}$ Select $P_f^{(3)}P_f^{(2)}P_f^{(4)}$
$E_f^{(2)}$ Select $P_f^{(1)}P_f^{(5)}P_f^{(3)}$
Tally
$C^{(1)}$ $C^{(2)}$ $C^{(3)}$
Delegation Calculation
$C^{(4)}$ $C^{(5)}$ $C^{(6)}$
Tally Calculation
Ranked Proposals
$\{P_f^{(3)}, P_f^{(2)}, P_f^{(1)}, P_f^{(4)}, P_f^{(5)}\}$
Shortlisted Proposals $\{P_f^{(3)}, P_f^{(2)}\}$
Stage2: Threshold Voting
Voter Cast
$V_f^{(1)}$ Vote YES to $P_f^{(3)}$
Vote NO to $P_f^{(2)}$
$V_f^{(2)}$ Delegate $P_f^{(3)}$ to $E_f^{(2)}$
Delegate $P_f^{(2)}$ to $E_f^{(1)}$
Expert Vote
$E_f^{(1)}$ Vote NO to $P_f^{(3)}$
Vote NO to $P_f^{(2)}$
$E_f^{(2)}$ Vote YES to $P_f^{(3)}$
Vote NO to $P_f^{(2)}$
Tally
$C^{(7)}$ $C^{(8)}$ $C^{(9)}$ Delegation Calculation
$C^{(10)}$ $C^{(11)}$ $C^{(12)}$ Tally Calculation
Tally Result
Final Selected Proposals $P_f^{(3)}$
Post-Voting Epoch
$P_f^{(3)}$
Receive Fund
Reward and Penalty
$C^{(1)}C^{(2)}C^{(3)}C^{(4)}$ $C^{(5)}$ $C^{(6)}$
$C^{(7)}C^{(8)}C^{(9)}$ $C^{(10)}C^{(11)}C^{(12)}$
$E_f^{(1)}$ $E_f^{(2)}$
$V_f^{(1)}$ $V_f^{(2)}$

Fig. 1: Systematic Design: A decision-making period w.r.t. field f.

- *Development and security*. Costs allocated for funding core and non-core development. *e.g.*, security incident response, patch management, running testnets, similar critical technology areas;
- *Organisation and management*. Costs on team coordination and management, legal support, etc;
- *Support*. Costs on user support, documentation, maintaining of web-infrastructure needed for the community and other similar areas;
- *General*. Proposals not covered by the aforementioned categories. *e.g.*, research on prospective technologies for cryptocurrency application, external security audit, collaboration with other communities, charity.

3.3 Pre-Voting Epoch

Pre-voting epoch consists of proposal registration, expert registration, and voter registration.

Proposal Registration: A series of proposal templates are released, each template shows the scope and requirement of relative proposal field, proposal owners can choose related proposal template to submit their proposals using *Two-Stage Project Proposing* procedure to ensure fairness:

- Stage 1: Commit. Proposal owners submit the commitment of their proposals by the deadline.
- Stage 2: Reveal. After the deadline, proposal owners open the commitment and reveal proposals. All proposals are checked by automated criteria to prevent spam and DDOS attacks. Only the valid proposals can be proceed in the next epoch.

Voter Registration: Users/stake holders who are interested in voting can lock stakes on blockchain to register as voters. The voting power is proportional to the amount of locked stakes.

Expert Registration: Stake holders who are highly regarded and reputable can register as experts by depositing a fixed amount of stakes on blockchain. The voting power of experts only comes from the voting power delegated to them by voters, which is called "Delegated Voting Power".

3.4 Voting Epoch

Our system uses two-stage voting to reduce the overall voting effort. The first stage is preferential voting: voters and experts announce their preferences, at the end of this stage, a shortlist is generated according to the budget in each field [§]. The second stage is threshold voting: voters and experts vote on the shortlist and generate a final winning list. We give an example to show the plaintext version of their ballots in Fig. 2. We allow voters and experts to submit multiple ballots, on the condition that only the newest and valid ballot of voter/expert is considered into the final tally.

Stage 1: Preferential Voting. In this stage, voters and experts vote on proposals after pre-filter in the pre-voting epoch using Borda Count Voting [26], [27]. A proposal shortlist is generated at the end of this stage based on voters' and experts' ballots, fund asked by each proposal, and the total available fund in the corresponding field.

- *Voter Cast*. Voter can either select its own shortlist or delegate its voting power to an expert;
- *Expert Vote*. Expert selects its own shortlist;

[§] Budget is enough to fund every project in the shortlist.

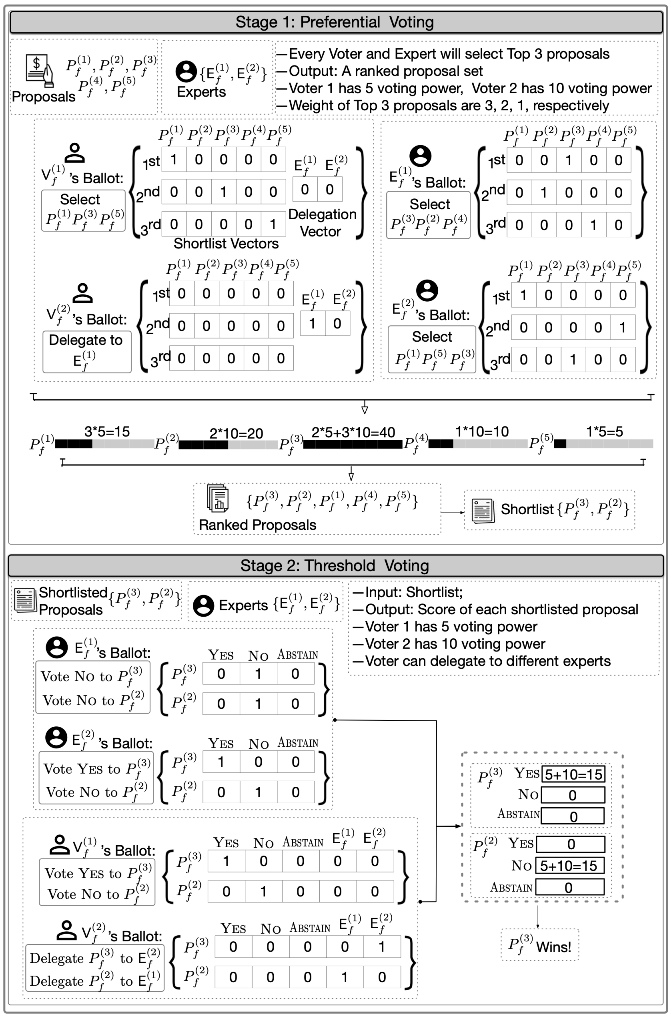

Stage 1: Preferential Voting
Proposals $P_f^{(1)}, P_f^{(2)}, P_f^{(3)}, P_f^{(4)}, P_f^{(5)}$
Experts $\{E_f^{(1)}, E_f^{(2)}\}$
—Every Voter and Expert will select Top 3 proposals
—Output: A ranked proposal set
—Voter 1 has 5 voting power, Voter 2 has 10 voting power
—Weight of Top 3 proposals are 3, 2, 1, respectively
$V_f^{(1)}$'s Ballot: Select $P_f^{(1)} P_f^{(3)} P_f^{(5)}$
1st 1 0 0 0 0
2nd 0 0 1 0 0
3rd 0 0 0 0 1
Shortlist Vectors
$E_f^{(1)}$ $E_f^{(2)}$ 0 0
Delegation Vector
$E_f^{(1)}$'s Ballot: Select $P_f^{(3)} P_f^{(2)} P_f^{(4)}$
1st 0 0 1 0 0
2nd 0 1 0 0 0
3rd 0 0 0 1 0
$V_f^{(2)}$'s Ballot: Delegate to $E_f^{(1)}$
1st 0 0 0 0 0
2nd 0 0 0 0 0
3rd 0 0 0 0 0
$E_f^{(1)}$ $E_f^{(2)}$ 1 0
$E_f^{(2)}$'s Ballot: Select $P_f^{(1)} P_f^{(5)} P_f^{(3)}$
1st 1 0 0 0 0
2nd 0 0 0 0 1
3rd 0 0 1 0 0
$P_f^{(1)}$ 3*5=15
$P_f^{(2)}$ 2*10=20
$P_f^{(3)}$ 2*5+3*10=40
$P_f^{(4)}$ 1*10=10
$P_f^{(5)}$ 1*5=5
Ranked Proposals $\{P_f^{(3)}, P_f^{(2)}, P_f^{(1)}, P_f^{(4)}, P_f^{(5)}\}$
Shortlist $\{P_f^{(3)}, P_f^{(2)}\}$
Stage 2: Threshold Voting
Shortlisted Proposals $\{P_f^{(3)}, P_f^{(2)}\}$
Experts $\{E_f^{(1)}, E_f^{(2)}\}$
—Input: Shortlist;
—Output: Score of each shortlisted proposal
—Voter 1 has 5 voting power
—Voter 2 has 10 voting power
—Voter can delegate to different experts
$E_f^{(1)}$'s Ballot: Vote No to $P_f^{(3)}$ Vote No to $P_f^{(2)}$
Yes No Abstain
$P_f^{(3)}$ 0 1 0
$P_f^{(2)}$ 0 1 0
$E_f^{(2)}$'s Ballot: Vote Yes to $P_f^{(3)}$ Vote No to $P_f^{(2)}$
Yes No Abstain
$P_f^{(3)}$ 1 0 0
$P_f^{(2)}$ 0 1 0
$V_f^{(1)}$'s Ballot: Vote Yes to $P_f^{(3)}$ Vote No to $P_f^{(2)}$
Yes No Abstain $E_f^{(1)}$ $E_f^{(2)}$
$P_f^{(3)}$ 1 0 0 0 0
$P_f^{(2)}$ 0 1 0 0 0
$V_f^{(2)}$'s Ballot: Delegate $P_f^{(3)}$ to $E_f^{(2)}$ Delegate $P_f^{(2)}$ to $E_f^{(1)}$
Yes No Abstain $E_f^{(1)}$ $E_f^{(2)}$
$P_f^{(3)}$ 0 0 0 0 1
$P_f^{(2)}$ 0 0 0 1 0
$P_f^{(3)}$ Yes 5+10=15 No 0 Abstain 0
$P_f^{(2)}$ Yes 0 No 5+10=15 Abstain 0
$P_f^{(3)}$ Wins!

Fig. 2: Ballots examples in two stage voting scheme (Assume all voters and experts are honest).

- *Tally*. Voting committee jointly computes and reveals the tally results; Proposals are ranked according to the scores. Top ranked proposals are shortlisted until the total fund is exhausted.

Stage 2: Threshold Voting. In this stage, voters and experts vote YES, NO, and ABSTAIN to each proposal in the shortlist. At the end of this stage, the final selected proposals which can be funded are generated automatically.

- *Voter Cast*. For each proposal in the shortlist, voter can vote directly, or delegate its voting power to an expert †.
- *Expert Vote*. Expert votes on the shortlist, its ballot is weighted w.r.t. its delegated voting power.
- *Tally*. Voting committee jointly computes experts' delegated voting power and tally results, reveals tally results (the number of YES votes, NO votes, and ABSTAIN votes) of each proposal in the shortlist. Denote the score of each proposal by the number of YES votes minus the number of NO votes. Proposals that got at least 10% (of all votes) of the positive difference are ranked according to the score, top ranked proposals are funded.

3.5 Post-Voting Epoch

Winning proposal(s) can be funded in this phase. Simultaneously, certain proportion of the fund is used to reward the parties who made correct decisions including voters and experts who have delegated voting power. Honest voting committee members receive a fix amount of reward. Meanwhile, if a voting committee member cheats or a voter/expert fails to submit a valid ballot, it loses deposited stakes as a punishment.

3.6 Evolving Committee

As the whole voting phase may take relatively long time (*e.g.*, 1 month), to prevent single voting committee from holding secret key shares (partial secret keys) for a long time, evolving committee is introduced to replace voting committee periodically.

In each round, stake holders who want to join voting committee lock certain amount of stakes on blockchain and privately perform cryptography sortition [15], [16], the probability of being selected is proportional to the number of locked stakes. After winning cryptography sortition, stake holder posts sortition proof on blockchain to announce its identity as the voting committee member. The genesis voting committee in the first round jointly runs DKG protocol to generate global key pairs (global public key and global secret key). Afterwards, voting committee in each round re-shares the global secret key to next voting committee until the end of voting epoch. At the end of voting epoch, the voting committee performs the Tally tasks (Delegation Calculation and Tally Calculation).

4 DISTRIBUTED BATCH KEY GENERATION

As will be mentioned later in the voting scheme (Sec. 5.2 and Sec. 5.3), we need multiple keys to encrypt ballots from voters and experts. In this section, we show how to efficiently generate multiple global key pairs with Distributed Batch Key Generation (DBKG) protocol based on Hyper Invertible Matrix (HIM).

Without losing generality, we denote the number of generated global key pairs by m, the DBKG functionality is captured by $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ in Fig. 3 to generate m global key pairs, and we also present protocol $\Pi_{\text{DBKG}}^{n,\mu,m}$ that *UC-realises* $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ in Fig. 4. Functionality $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ interacts with n parties $\mathsf{P} := \{P_1, \ldots, P_n\}$ of which the threshold is μ ($\mu \geq 1/2 \cdot n$, a.k.a, the number of honest parties), and an ideal adversary $\mathcal{S}$. Each P_i sends $(\mathsf{KeyGen}, \mathsf{sid}, P_i)$ to $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ to start keys generation, $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ begins to generate global key pairs until all parties send this message.

To guarantee the uniformly distribution of global public keys and secret keys, $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ firstly picks random global secret keys $\{\mathsf{gsk}_v\}_{v=1}^{m}$, and then computes global public keys $\mathsf{gpk}_v := g^{\mathsf{gsk}_v}$ for $v \in [m]$.

Denote $\mathsf{P_c}$ as the set of corrupted parties, $\mathsf{P_h}$ as the set of honest parties, $|\mathsf{P_c}| + |\mathsf{P_h}| = n$, and $|\mathsf{P_c}| \leq n - \mu - 1$.

† In threshold voting, voter can delegate each proposal to a different expert; whereas in preferential voting, voter can only delegate to one expert. Delegation ballots can be found in Fig. 2 submitted by $\mathsf{V}_f^{(2)}$.

Functionality $\mathcal{F}_{\textbf{DBKG}}^{n,\mu,m}$

$\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ interacts with parties $\mathsf{P} := \{P_1, \ldots, P_n\}$, and ideal adversary $\mathcal{S}$. It's parameterised with threshold μ, and the number of generated global key pairs m. Denote $\mathsf{P_c}$ as the set of corrupted parties, and $\mathsf{P_h}$ as the set of honest parties, $|\mathsf{P_c}| + |\mathsf{P_h}| = n$, and $|\mathsf{P_c}| \leq n - \mu - 1$. $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ maintains a set $\mathbf{N}$ (initially set to $\emptyset$).

$\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ does the following:

- Upon receiving $(\mathsf{KeyGen}, \mathsf{sid}, P_i)$ from $P_i \in \mathsf{P}$, set $\mathbf{N} = \mathbf{N} \cup \{P_i\}$, send $(\textsc{KeyGenNotify}, \mathsf{sid}, P_i)$ to $\mathcal{S}$, continue to next step until $|\mathbf{N}| = n$;
- Upon receiving message $(\textsc{CorruptShares}, \mathsf{sid}, \{j, \{\mathsf{psk}_{j,v}\}_{v=1}^{m}\}_{P_j \in \mathsf{P_c}})$ from $\mathcal{S}$:
  - Pick $\{\mathsf{gsk}_v\}_{v=1}^{m} \leftarrow (\mathbb{Z}_q)^{[m]}$, and compute $\mathsf{gpk}_v := g^{\mathsf{gsk}_v}$ for $v \in [m]$;
  - Set $a := \mu - |\mathsf{P_c}| - 1$, $\mathsf{P_h}' \subset \mathsf{P_h}$, $|\mathsf{P_h}'| = a$, select $\{\mathsf{psk}_{i,v}\}_{P_i \in \mathsf{P_h}', v \in [m]} \leftarrow (\mathbb{Z}_q)^{[m \cdot a]}$;
  - For $v \in [m]$, construct random polynomial $F_v(x) := \sum_{b=0}^{\mu-1} a_b \cdot x^b$ under the restriction $F_v(j) = \mathsf{psk}_{j,v}$ for $P_j \in \{\mathsf{P_c} \cup \mathsf{P_h}'\}$, and $F_v(0) = \mathsf{gsk}_v$;
  - Compute $\mathsf{psk}_{v,i} := F_v(i)$ and $\mathsf{ppk}_{v,i} := g^{\mathsf{psk}_{v,i}}$ for $P_i \in \mathsf{P_h}$, $v \in [m]$;
- Upon receiving $(\textsc{ReadKeyShare}, \mathsf{sid}, P_i)$ from $\mathcal{S}$, send $(\textsc{ReadKeyShareReturn}, \mathsf{sid}, \{\mathsf{psk}_{v,i}\}_{v=1}^{m})$ to P_i;
- Upon receiving $(\textsc{ReadPK}, \mathsf{sid})$ from any party, return the corresponding party the message $(\textsc{ReadPKReturn}, \mathsf{sid}, \{\mathsf{gpk}_v\}_{v=1}^{m}, \{\mathsf{ppk}_{v,i}\}_{v=1,i=1}^{m,n})$.

Fig. 3: DBKG Ideal Functionality $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$.

To guarantee adversary's power of controlling global secret keys shares (partial secret keys) of corrupted parties, we model that ideal adversary $\mathcal{S}$ can send corrupted parties' partial secret keys $\{\mathsf{psk}_{j,v}\}_{v \in [m], P_j \in \mathsf{P_c}}$ to $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$, $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ then constructs m degree-$(\mu - 1)$ polynomials based on m random global secret keys, $m \cdot |\mathsf{P_c}|$ corrupted partial secret keys and $m \cdot a$ random partial secret keys of a honest parties, where $a := \mu - |\mathsf{P_c}| - 1$.

4.1 HIM based DBKG Protocol $\Pi_{\textbf{DBKG}}^{n,\mu,m}$

Naively, generating multiple keys can be achieved by repeatedly executing DKG protocols (such as Gennaro etal.'s version [8]) with linear complexity. A more efficient way is to replace the single secret sharing scheme used in DKG (such as Shamir Secret Sharing) with HIM based random secret sharing [11], which shares n randomnesses with only n^2 sharings and achieves amortised $\mathcal{O}(n)$ communication cost. However, due to feasible mapping from inputs to outputs in HIM (Cf. Theorem 2 and Property 2), to ensure the uniformly distribution of global key pairs, we can only generate $n - \mu$ pairs, where μ is the threshold ($|\mathsf{P_c}| \leq \mu - 1$).

We give the protocol $\Pi_{\text{DBKG}}^{n,\mu,m}$ (Fig. 4) that *UC-realises* $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ in $\{\mathcal{F}_{\text{LEDGER}}\}$-*hybrid world*, where $\mathcal{F}_{\text{LEDGER}}$ is a blockchain functionality taken from Cheng *et al.*'s work [28].

Initially, we assume that every party holds a same **HIM** denoted by $\{\lambda_{v,w}\}_{v=1,w=1}^{n,n}$. Once a party P_i gets message $(\mathsf{KeyGen}, \mathsf{sid}, P_i)$ from $\mathcal{Z}$, it selects a random value x_i from $\mathbb{Z}_q$ and share x_i to other parties with polynomial $F_i(\cdot)$, all the shares are encrypted by receiver's public key using Lifted Elgamal encryption. In addition, P_i needs to commit the polynomial and generate Correct Sharing NIZK σ_i (Cf. Sec. 1.1 in supporting document) to show that each share is correctly computed from the polynomial based on the commitment and ciphertexts. Later, P_i posts the ciphertexts together with commitment and σ_i to $\mathcal{F}_{\text{LEDGER}}$[‡].

In next round, P_i gets its shares $\{s_{j,i}\}_{j=1}^{n}$ from other parties and constructs qualified party set QUAL by verifying Correct Sharing NIZK proofs. P_i's m partial secret keys can be locally computed based on the shares from qualified parties and corresponding **HIM** parameters, with HIM based random secret sharing.

For threshold decryption (Cf. Sec. 2 in supporting document), P_i commits its polynomial and posts new commitments $\{D_{i,k}\}_{k=0}^{\mu-1}$ to $\mathcal{F}_{\text{LEDGER}}$. In next round, P_i gets other parties' commitments $\{D_{j,k}\}_{j \in \mathsf{QUAL}, k \in [0,\mu-1]}$ and checks if they are consistent with $\{s_{j,i}\}_{j \in \mathsf{QUAL}}$ by $\prod_{k=0}^{\mu-1} (D_{j,k})^{i^k} = g^{s_{j,i}}$. If verification about a party P_j fails, P_i needs to reveal $s_{j,i}$ and clarify a valid complain about P_j, by generating a Correct Decryption NIZK σ_i' (Cf. Sec. 1.2 in supporting document) to prove that it indeed gets $s_{j,i}$ from decrypting the ciphertexts encrypted by its public key sent by P_j. The rest parties jointly interpolate P_j's polynomial if the complain is valid, then post correct $\{D_{j,k}\}_{k=0}^{\mu-1}$ to $\mathcal{F}_{\text{LEDGER}}$. P_i can compute the global public keys by $\mathsf{gpk}_v := \prod_{j \in \mathsf{QUAL}} (D_{j,0})^{\lambda_{v,j}}$ for $v \in \mathbf{N}$ and post them to $\mathcal{F}_{\text{LEDGER}}$, where $\mathbf{N} \subset [n]$ is a predefined set and $|\mathbf{N}| = m$.

4.2 Security Analysis

Theorem 3 (Distributed Batch Key Generation). *Let $\mu \geq n/2$, assume* Com *is perfect hiding and computational binding with adversary advantage of* $\mathsf{Adv}_{\mathsf{Com}}^{\mathsf{Bind}}(1^\kappa, \mathcal{A})$. *Assume Correct Sharing* NIZK *and Correct Decryption* NIZK *are perfect complete, perfect special honest verifier zero knowledge, and computational sound with adversary advantage of* $\mathsf{Adv}_{\mathsf{NIZK},Sharing}^{\mathsf{Sound}}(1^\kappa, \mathcal{A})$ *and* $\mathsf{Adv}_{\mathsf{NIZK},Dec}^{\mathsf{Sound}}(1^\kappa, \mathcal{A})$. *Assume* Enc *is IND-CPA secure with adversary advantage of* $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A})$. *The protocol* $\Pi_{\text{DBKG}}^{n,\mu,m}$ *in Fig.4 UC-realises* $\mathcal{F}_{\text{DBKG}}^{n,\mu,m}$ *in Fig.3 in* $\{\mathcal{F}_{\text{LEDGER}}\}$*-hybrid world against static corruption up to $n - \mu - 1$ parties with distinguishing advantage upper bounded by*

$$(n - \mu - 1) \cdot \Big(\mathsf{Adv}_{\mathsf{Com}}^{\mathsf{Bind}}(1^\kappa, \mathcal{A}) + \mathsf{Adv}_{\mathsf{NIZK},Sharing}^{\mathsf{Sound}}(1^\kappa, \mathcal{A}) + \mathsf{Adv}_{\mathsf{NIZK},Dec}^{\mathsf{Sound}}(1^\kappa, \mathcal{A})\Big) + (n - 1) \cdot \mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A}).$$

Its proof can be found in the supporting document (Sec. 3.1).

5 Two Stage Voting Scheme

In this section, we give the security modelling of the proposed Two Stage Voting (TSV) Scheme against static corruption, and prove the security of each protocol (Preferential Voting and Threshold Voting) under the UC framework based on the indistinguishability of ideal world execution and real world execution.

[‡] Sender like P_i does not need to post message for itself, we use $j \in [n]$ instead of $j \in [n] \setminus \{i\}$ in Fig. 4 to make protocol neat.

The HIM based DBKG Protocol $\Pi^{n,\mu,m}_{\mathbf{DBKG}}$

Denote parties set by $\mathsf{P} := \{P_1, \dots, P_n\}$, of which the threshold is μ, each party has a predefined HIM set $\mathbf{HIM} := \{\lambda_{v,w}\}^{n,n}_{v=1,w=1}$and a predefined set $\mathbf{N} \subset [n]$, where $|\mathbf{N}| = m$. Denote the Elgamal encryption scheme parameters by ℓ, p.

- Upon receiving $(\mathsf{KeyGen}, \mathsf{sid}, P_i)$ from $\mathcal{Z}$, $P_i \in \mathsf{P}$ does the following:
  - Select $x_i \leftarrow \mathbb{Z}_q$, $\{a_{i,k}\}^{\mu-1}_{k=1} \leftarrow (\mathbb{Z}_q)^{[\mu-1]}$, construct a polynomial $F_i(z) := \sum^{\mu-1}_{k=0} a_{i,k} \cdot z^k$, where $a_{i,0} := x_i$;
  - Select $\{a'_{i,k}\}^{\mu-1}_{k=0} \leftarrow (\mathbb{Z}_q)^{[\mu]}$, compute $C_{i,k} := \mathsf{Com}_{\mathsf{ck}}(a_{i,k}; a'_{i,k})$ for $k \in [0, \mu - 1]$;
  - Compute $s_{i,j} := F_i(j)$, and $(d_{i,j,1}, \dots, d_{i,j,\ell}) := \textsc{Encode}(s_{i,j})$ for $j \in [n]$;
  - Select $r_{i,j,b} \leftarrow \mathbb{Z}_q$, and compute $(A_{i,j,b}, B_{i,j,b}) := \mathsf{Enc}_{\mathsf{pk}_j}(d_{i,j,b}; r_{i,j,b})$ for $j \in [n]$ and $b \in [\ell]$;
  - Generate Correct Sharing NIZK σ_i:

$$\sigma_i \leftarrow \mathsf{NIZK} \left\{ \begin{array}{l} (g, \mathsf{ck}, \{\mathsf{pk}_j\}^n_{j=1}, \{A_{i,j,b}, B_{i,j,b}\}^{n,\ell}_{j=1,b=1}, \{C_{i,k}\}^{\mu-1}_{k=0}), \\ (\{s_{i,j}\}^n_{j=1}, \{r_{i,j,b}\}^{n,\ell}_{j=1,b=1}, \{a_{i,k}, a'_{i,k}\}^{\mu-1}_{k=0}) : \\ \{\prod^{\ell}_{b=1}(A_{i,j,b})^{p^b} = g^{\sum^{\ell}_{b=1} r_{i,j,b} \cdot p^b}\}^n_{j=1} \wedge \{\prod^{\ell}_{b=1}(B_{i,j,b})^{p^b} = g^{s_{i,j}} \cdot \mathsf{pk}_j^{\sum^{\ell}_{b=1} r_{i,j,b} \cdot p^b}\}^n_{j=1} \\ \wedge\{\prod^{\mu-1}_{k=0}(C_{i,k})^{j^k} = g^{s_{i,j}} \cdot \mathsf{ck}^{\sum^{\mu-1}_{k=0}(a'_{i,k})^{j^k}}\}^n_{j=1} \end{array} \right\}$$

  - Post $(\{A_{i,j,b}, B_{i,j,b}\}^{n,\ell}_{j=1,b=1}, \sigma_i, \{C_{i,k}\}^{\mu-1}_{k=0})$ to $\mathcal{F}_{\textsc{Ledger}}$;
  - Fetch $\{\{A_{j,i,b}, B_{j,i,b}\}^{\ell}_{b=1}, \sigma_j, \{C_{j,k}\}^{\mu-1}_{k=0}\}\}^n_{j=1}$, compute $\{s_{j,i} = \textsc{Decode}(\{\mathsf{Dec}_{\mathsf{sk}_i}(A_{j,i,b}, B_{j,i,b})\}^{\ell}_{b=1})\}^n_{j=1}$;
  - For $j \in [n]$, if $\mathsf{Verify}(\sigma_j, \{A_{i,j,b}, B_{i,j,b}\}^{n,\ell}_{j=1,b=1}, \{C_{i,k}\}^{\mu-1}_{k=0}) = 0$, set $\mathsf{QUAL} = [n] \setminus \{j\}$;
  - Compute $D_{i,k} := \mathsf{Com}_{\mathsf{ck}}(a_{i,k}; 0)$ for $k \in [0, \mu - 1]$;
  - Compute its own partial secret keys as $\mathsf{psk}_{v,i} := \sum_{j \in \mathsf{QUAL}} \lambda_{v,j} \cdot s_{j,i}$ for $v \in \mathbf{N}$;
  - Post $(\{D_{i,k}\}^{\mu-1}_{k=0})$ to $\mathcal{F}_{\textsc{Ledger}}$, fetch $\{\{D_{j,k}\}^{\mu-1}_{k=0}\}_{j \in \mathsf{QUAL}\setminus\{i\}}$, check if $\prod^{\mu-1}_{k=0}(D_{j,k})^{i^k} = g^{s_{j,i}}$ holds. If verification fails for P_j, post $(\textsc{Complaint}, s_{j,i}, \sigma'_i)$ to $\mathcal{F}_{\textsc{Ledger}}$, where σ'_i is a Correct Decryption NIZK:

$$\sigma'_i \leftarrow \mathsf{NIZK} \left\{ \begin{array}{l} (g, \mathsf{pk}_i, \{A_{j,i,b}, B_{j,i,b}\}^{\ell}_{b=1}, s_{j,i}), (\mathsf{sk}_i) : \\ \{d_{j,i,b} = \mathsf{Dec}_{\mathsf{sk}_i}(A_{j,i,b}, B_{j,i,b})\}^{\ell}_{b=1} \wedge s_{j,i} = \textsc{Decode}(\{d_{j,i,b}\}^{\ell}_{b=1}) \wedge \mathsf{pk}_i = g^{\mathsf{sk}_i} \end{array} \right\}$$

  - If any party $\{P_w\}_{w \in \mathsf{QUAL}}$ posted $(\textsc{Complaint}, s_{v,w}, \sigma'_w)$ about P_v to $\mathcal{F}_{\textsc{Ledger}}$, do the following if $\mathsf{Verify}(\sigma'_k, \{(A_{v,w,b}, B_{v,w,b})\}^{\ell}_{b=1}) = 1$:
    - * Fetch $\{s_{v,j}, \sigma'_j\}_{j \in \mathsf{QUAL}}$, select μ values from $\{s_{v,j}\}_{j \in \mathsf{QUAL}}$, where $\prod^{\mu-1}_{k=0}(D_{v,k})^{j^k} = g^{s_{a,j}}$;
    - * Interpolate $F_v(x)$, compute $\{D_{v,k}\}^{\mu-1}_{k=0}$, post $(\{D_{v,k}\}^{\mu-1}_{k=0}, P_w)$ to $\mathcal{F}_{\textsc{Ledger}}$.
  - Compute global public keys as $\mathsf{gpk}_v := \prod_{j \in \mathsf{QUAL}}(D_{j,0})^{\lambda_{v,j}}$ for $v \in \mathbf{N}$, post $(\{\mathsf{gpk}_v\}_{v \in \mathbf{N}})$ to $\mathcal{F}_{\textsc{Ledger}}$.
- Upon receiving $(\textsc{ReadKeyShare}, \mathsf{sid}, P_i)$ from $\mathcal{Z}$, send $(\textsc{ReadKeyShareReturn}, \mathsf{sid}, \{\mathsf{psk}_{v,i}\}_{v \in \mathbf{N}})$ to $\mathcal{Z}$;
- Upon receiving $(\textsc{ReadPK}, \mathsf{sid})$ from $\mathcal{Z}$, the party P fetches $(\{\mathsf{gpk}_v\}_{v \in \mathbf{N}}, \{D_{j,k}\}^{\mu-1,n}_{k=0,j=1})$, computes $\mathsf{ppk}_{v,i} = \prod_{j \in \mathsf{QUAL}}(\prod^{\mu-1}_{k=0}(D_{j,k})^{i^k})^{\lambda_{v,j}}$ for $v \in \mathbf{N}$ and $i \in [n]$, returns the environment $\mathcal{Z}$ $(\textsc{ReadPKReturn}, \mathsf{sid}, (\{\mathsf{gpk}_v\}_{v \in \mathbf{N}}, \{\mathsf{ppk}_{v,i}\}_{v \in \mathbf{N}, i \in [n]}))$.

Fig. 4: HIM based DBKG Protocol $\Pi^{n,\mu,m}_{\mathrm{DBKG}}$ in $\{\mathcal{F}_{\textsc{Ledger}}\}$-*hybrid world*.

Denote voting committee size by c of which the threshold is μ ($\mu \geq 1/2 \cdot c$, a.k.a, the number of honest voting committee members), the number of candidate proposals by n, the number of selected proposals by s ($s \leq n$ in preferential voting, $s = n$ in threshold voting). To start with, we design a voting ideal functionality $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ to cover security properties in two voting stages. Then we propose preferential voting protocol $\Pi^{c,\mu,s,n}_{\textsc{Vote1}}$ and threshold voting protocol $\Pi^{c,\mu,s,n}_{\textsc{Vote2}}$ to realise $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ in the corresponding stage of TSV scheme.

5.1 Voting Ideal Functionality $\mathcal{F}^{c,\mu,s,n}_{\textsf{VOTE}}$

In the ideal world, voting ideal functionality $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ interacts with voters, experts, voting committee, and adversary $\mathcal{S}$. It's parameterised with a voting stage index $\gamma \in \{1, 2\}$, a delegation calculation algorithm DelAlg (Fig. 6) and a tally algorithm $\mathsf{TallyAlg}$ (Fig. 7) . There are four phases in $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$, Initialisation, Voter Cast, Expert Vote and Tally.

Initialisation Phase. Voting committee member $\mathcal{C}^{[c]}$ sends message $(\textsc{Init}, \mathsf{sid})$ to $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$, then $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ notifies $\mathcal{S}$ by $(\textsc{InitNotify}, \mathsf{sid}, \mathsf{C}^{(t)})$.

Voter Cast Phase. Voter $\mathsf{V}^{(i)}_{\mathsf{fld}} \in \mathcal{V}^{[v]}_{\mathsf{fld}}$ sends its ballots $\mathbf{a_i}$ and voting power η_i to $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ by $(\textsc{Cast}, \mathsf{sid}, \mathbf{a_i}, \eta_i)$. $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ notifies $\mathcal{S}$ by sending $(\textsc{CastNotify}, \mathsf{V}^{(i)}_{\mathsf{fld}}, \mathsf{sid}, \eta_i)$. If more than $v - \mu - 1$ voting committee members are corrupted, $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ additionally leaks voter's inputs to $\mathcal{S}$ by sending $(\textsc{Leak}, \mathsf{V}^{(i)}_{\mathsf{fld}}, \textsc{Cast}, \mathsf{sid}, \mathbf{a}_i, \eta_i)$.

Expert Vote Phase. Expert $\mathsf{E}^{(j)}_{\mathsf{fld}} \in \mathcal{E}^{[e]}_{\mathsf{fld}}$ sends its ballots $\mathbf{b_j}$ to $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ by $(\textsc{Vote}, \mathsf{sid}, \mathbf{b_j})$. $\mathcal{S}$ is aware of this action when receiving $(\textsc{VoteNotify}, \mathsf{E}^{(j)}_{\mathsf{fld}}, \mathsf{sid})$ from $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$. If more than $v - \mu - 1$ voting committee member are corrupted, $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ additionally leaks expert's inputs to $\mathcal{S}$.

Tally Phase. $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ firstly computes delegated voting power of experts by getting $(\textsc{CalDel}, \mathsf{sid})$ from voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$, and sends $(\textsc{CalDelNotify}, \mathsf{sid}, \mathsf{C}^{(t)})$ to $\mathcal{S}$. If there are more than μ voting committee members, $\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ can compute delegated voting power by $\{\mathbf{D_j}\}^e_{j=1} \leftarrow \mathsf{DelAlg}(n, s, e, \gamma, \{\mathbf{a_i}, \eta_i\}^v_{i=1})$, which is revealed to $\mathcal{S}$ if corruption exceeds μ.

$\mathcal{F}^{c,\mu,s,n}_{\textsc{Vote}}$ begins to compute the tally results of each proposal and notify $\mathcal{S}$ by $(\textsc{TallyNotify}, \mathsf{sid}, \mathsf{C}^{(t)})$, once it gets

Voting Ideal Functionality $\mathcal{F}_{\textbf{VOTE}}^{c,\mu,s,n}$

$\mathcal{F}_{\text{VOTE}}^{c,\mu,s,n}$ interacts with voters $\mathcal{V}_{\mathsf{fld}}^{[v]} := \{\mathsf{V}_{\mathsf{fld}}^{(i)}\}_{i=1}^{v}$, experts $\mathcal{E}_{\mathsf{fld}}^{[e]} := \{\mathsf{E}_{\mathsf{fld}}^{(j)}\}_{j=1}^{e}$, voting committee $\mathcal{C}^{[c]} := \{\mathsf{C}^{(t)}\}_{t=1}^{c}$ of which the threshold is μ, and adversary $\mathcal{S}$. $\mathcal{F}_{\text{VOTE}}^{c,\mu,s,n}$ is parameterised with three committee flag sets $\mathbf{C}_{key}$, $\mathbf{C}_{del}$, $\mathbf{C}_{tally}$, and a voting stage index $\gamma \in \{1,2\}$, a delegation calculation algorithm DelAlg, a tally algorithm $\mathsf{TallyAlg}$, corrupted voting committee $\mathcal{C}_{\mathsf{cor}}$, honest voting committee $\mathcal{C}_{\mathsf{hon}}$, the number of candidate proposals n, and the number of selected proposals s ($s \leq n$ in preferential voting, $s = n$ in threshold voting).

$\mathcal{F}_{\text{VOTE}}^{c,\mu,s,n}$ does the following:

Initialisation Phase:

- Upon receiving $(\textsc{Init}, \mathsf{sid})$ from a voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$, send $(\textsc{InitNotify}, \mathsf{sid}, \mathsf{C}^{(t)})$ to $\mathcal{S}$, and set $\mathbf{C}_{key} := \mathbf{C}_{key} \cup \{\mathsf{C}^{(t)}\}$, continue to next step until $|\mathbf{C}_{key}| = c$.

Voter Cast Phase:

- Upon receiving $(\textsc{Cast}, \mathsf{sid}, \mathbf{a_i}, \eta_i)$ from a voter $\mathsf{V}_{\mathsf{fld}}^{(i)} \in \mathcal{V}_{\mathsf{fld}}^{[v]}$, send $(\textsc{CastNotify}, \mathsf{V}_{\mathsf{fld}}^{(i)}, \mathsf{sid}, \eta_i)$ to $\mathcal{S}$. Send $(\textsc{Leak}, \mathsf{V}_{\mathsf{fld}}^{(i)}, \textsc{Cast}, \mathsf{sid}, \mathbf{a}_i, \eta_i)$ to $\mathcal{S}$ if $|\mathcal{C}_{\mathsf{cor}}| \geq v - \mu - 1$.

Expert Vote Phase:

- Upon receiving $(\textsc{Vote}, \mathsf{sid}, \mathbf{b_j})$ from an expert $\mathsf{E}_{\mathsf{fld}}^{(j)} \in \mathcal{E}_{\mathsf{fld}}^{[e]}$, send $(\textsc{VoteNotify}, \mathsf{E}_{\mathsf{fld}}^{(j)}, \mathsf{sid})$ to $\mathcal{S}$. Send $(\textsc{Leak}, \mathsf{E}_{\mathsf{fld}}^{(j)}, \textsc{Vote}, \mathsf{sid}, \mathbf{b_j})$ to $\mathcal{S}$ if $|\mathcal{C}_{\mathsf{cor}}| \geq v - \mu - 1$.

Tally Phase:

- Upon receiving $(\textsc{CalDel}, \mathsf{sid})$ from a voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$, does the following:
  - Set $\mathbf{C}_{del} := \mathbf{C}_{del} \cup \{\mathsf{C}^{(t)}\}$, send $(\textsc{CalDelNotify}, \mathsf{sid}, \mathsf{C}^{(t)})$ to $\mathcal{S}$;
  - If $|\mathbf{C}_{del}| \geq \mu$, compute $\{\mathbf{D_j}\}_{j=1}^{e} \leftarrow \mathsf{DelAlg}(n, s, e, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v})$, Cf. Fig.6;
  - Send $(\textsc{LeakDel}, \mathsf{sid}, \mathsf{DelAlg}(n, s, e, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v}))$ to $\mathcal{S}$ if $|\mathbf{C}_{del} \cap \mathcal{C}_{\mathsf{cor}}| \geq \mu$.
- Upon receiving $(\textsc{Tally}, \mathsf{sid}, \mathbf{T})$ from a voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$, does the following:
  - Set $\mathbf{C}_{tally} := \mathbf{C}_{tally} \cup \{\mathsf{C}^{(t)}\}$, send $(\textsc{TallyNotify}, \mathsf{sid}, \mathsf{C}^{(t)})$ to $\mathcal{S}$;
  - If $|\mathbf{C}_{tally}| \geq \mu$, compute $\{\mathbf{f_l}\}_{l=1}^{n} \leftarrow \mathsf{TallyAlg}(n, s, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v}, \{\mathbf{b_j}, \mathbf{D_j}\}_{j=1}^{e}, \mathbf{T})$, Cf. Fig.7;
  - Send $(\textsc{LeakCasting}, \mathsf{sid}, \mathsf{TallyAlg}(n, s, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v}, \{\mathbf{b_j}, \mathbf{D_j}\}_{j=1}^{e}, \mathbf{T}))$ to $\mathcal{S}$ if $|C_{tally} \cap \mathcal{C}_{\mathsf{cor}}| \geq \mu$;
- Upon receiving $(\textsc{ReadTally}, \mathsf{sid})$ from any party, return $(\textsc{ReadTallyReturn}, \mathsf{sid}, \{\mathbf{f_l}\}_{l=1}^{n})$ to the requester;
- Upon receiving $(\textsc{Reveal}, \mathsf{sid}, \mathsf{E}_{\mathsf{fld}}^{(j)})$ from any party, return $(\textsc{RevealExpert}, \mathsf{sid}, \mathbf{b_j})$ to the requester.

Fig. 5: The ideal functionality $\mathcal{F}_{\text{VOTE}}^{c,\mu,s,n}$.

$(\textsc{Tally}, \mathsf{sid}, \mathbf{T})$ from a voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$. When more than μ voting committee members send this command, $\mathcal{F}_{\text{VOTE}}^{c,\mu,s,n}$ calculates tally results of each proposal by $\{\mathbf{f_l}\}_{l=1}^{n} \leftarrow \mathsf{TallyAlg}(n, s, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v}, \{\mathbf{b_j}, \mathbf{D_j}\}_{j=1}^{e}, \mathbf{T})$, which is leaked to $\mathcal{S}$ if more than μ members are corrupted.

Algorithm $\mathsf{DelAlg}(n, s, e, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v})$

Input: The number of candidate proposals n, the number of selected proposals s, the number of experts e, the voting stage index γ, voters' ballots and voting power $\{\mathbf{a_i}, \eta_i\}_{i=1}^{v}$.

Delegation Calculation:

- If $\gamma = 1$: For $i \in [v]$, parse $\mathbf{a}_i$ to $(\{v_{i,l,k}\}_{l=1,k=1}^{s,n}, \{v'_{i,j}\}_{j=1}^{e})$. For $j \in [e]$, compute $D_j := \sum_{i=1}^{v} v'_{i,j} \cdot \eta_i$. Set $\mathbf{D_j} := D_j$.
- If $\gamma = 2$: For $i \in [v]$, parse $\mathbf{a}_i$ to $(\{w_{i,l,k}\}_{l=1,k=1}^{s,3+e})$. For $j \in [e]$, $l \in [s]$, compute $D_{j,l} := \sum_{i=1}^{v} w_{i,l,j+2} \cdot \eta_i$. Set $\mathbf{D_j} := \{D_{j,l}\}_{l=1}^{s}$;

Output: Experts' delegated voting power $\{\mathbf{D_j}\}_{j=1}^{e}$.

Fig. 6: Delegation Calculation Algorithm.

5.2 Stage 1: Preferential Voting protocol $\Pi_{\textbf{VOTE}1}^{c,\mu,s,n}$

Fig. 8 presents preferential voting protocol $\Pi_{\text{VOTE}1}^{c,\mu,s,n}$ to realise $\mathcal{F}_{\text{VOTE}}^{c,\mu,s,n}$ in $\{\mathcal{F}_{\text{LEDGER}}, \mathcal{F}_{\text{DBKG}}^{c,\mu,1}\}$-*hybrid world.*

Voter Cast Phase. Once a voter $\mathsf{V}_{\mathsf{fld}}^{(i)} \in \mathcal{V}_{\mathsf{fld}}^{[v]}$ gets $(\textsc{Cast}, \mathsf{sid}, \mathbf{a}_i, \eta_i)$ from $\mathcal{Z}$, it casts ballots based on $\mathbf{a}_i$ [‡]. Firstly, $\mathsf{V}^{(i)}$ parses $\mathbf{a}_i$ to ballots $(\{\mathbf{v_{i,k}}^{[n]}\}_{k=1}^{s}, \mathbf{v'_i}^{[e]})$ (see examples in Fig. 2), where $\{\mathbf{v_{i,k}}^{[n]}\}_{k=1}^{s}$ are shortlist vectors and $\mathbf{v'_i}^{[e]}$ is a delegation vector. $\mathsf{V}_{\mathsf{fld}}^{(i)}$ encrypts all ballots with a global public key generated by voting committee, and outputs ciphertexts $\{(\mathbf{A_{i,k}}^{[n]}, \mathbf{A'_{i,k}}^{[n]})\}_{k=1}^{s}$ and $(\mathbf{B_i}^{[e]}, \mathbf{B'_i}^{[e]})$.

$\mathsf{V}_{\mathsf{fld}}^{(i)}$ generates a Valid Ballot NIZK proof σ_i (Cf. Sec. 1.6 in supporting document) to show its ballot is valid:

- For $k \in [s]$, $(\mathbf{A_{i,k}}^{[n]} || \mathbf{B_i}^{[e]}, \mathbf{A'_{i,k}}^{[n]} || \mathbf{B'_i}^{[e]})$ encrypts a unit vector [1], of which only one element is 1, the rest are 0;
- For $l \in [n]$, $(\prod_{k=1}^{s} A_{i,k,l}, \prod_{k=1}^{s} A'_{i,k,l})$ encrypts either 0 or 1, where $\{(A_{i,k,l}, A'_{i,k,l})\}_{k=1,l=1}^{s,n} := (\mathbf{A_{i,k}}^{[n]}, \mathbf{A'_{i,k}}^{[n]})$.

Afterwards, $\mathsf{V}^{(i)}$ posts ciphertexts and σ_i to $\mathcal{F}_{\text{LEDGER}}$.

[‡] We remark that in the case where participants' voting devices are not trusted, we can use cast-or-audit technique such as Benaloh challenge [29], [30] or Véronique etal.'s work [31] to support audit for every participant without voting again.

```
Algorithm
$\mathsf{TallyAlg}(n, s, \gamma, \{\mathbf{a_i}, \eta_i\}_{i=1}^{v}, \{\mathbf{b_j}, \mathbf{D_j}\}_{j=1}^{e}, \mathbf{T})$

Input: The number of candidate proposals $n$, the number of selected proposals $s$, the voting stage index $\gamma$, voters' ballots and voting power $\{\mathbf{a_i}, \eta_i\}_{i=1}^{v}$, experts' ballots and voting power $\{\mathbf{b_j}, \mathbf{D_j}\}_{j=1}^{e}$, the weight of each selected proposals $\mathbf{T} := \{\tau_l\}_{l=1}^{s}$.
Tally Calculation:
  • If $\gamma = 1$: For $i \in [v]$,
      – Parse $\mathbf{a}_i$ to $(\{v_{i,l,k}\}_{l=1,k=1}^{s,n}, \{v'_{i,j}\}_{j=1}^{e})$ ;
      – Parse $\mathbf{b_j}$ to $(\{p_{j,l,k}\}_{l=1,k=1}^{s,n})$ for $j \in [e]$;
      – For $k \in [n]$, compute $f_k := (\sum_{i=1}^{v}(\sum_{l=1}^{s} v_{i,l,k} \cdot \tau_l) \cdot \eta_i) + (\sum_{j=1}^{e}(\sum_{l=1}^{s} p_{j,l,k} \cdot \tau_l) \cdot D_j))$, set $\mathbf{f_k} := f_k$.
  • If $\gamma = 2$:
      – Parse $\mathbf{a}_i$ to $(\{w_{i,k,l}\}_{k=1,l=1}^{s,3+e})$ for $i \in [v]$;
      – Parse $\mathbf{b_j}$ to $(\{q_{j,k,l}\}_{k=1,l=1}^{s,3})$ for $j \in [e]$;
      – For $k \in [n]$, compute the following:
          * $f_{k,1} := (\sum_{i=1}^{v} w_{i,k,1} \cdot \eta_i) + (\sum_{j=1}^{e} q_{j,k,1} \cdot D_{j,k})$;
          * $f_{k,2} := (\sum_{i=1}^{v} w_{i,k,2} \cdot \eta_i) + (\sum_{j=1}^{e} q_{j,k,2} \cdot D_{j,k})$;
          * $f_{k,3} := (\sum_{i=1}^{v} w_{i,k,3} \cdot \eta_i) + (\sum_{j=1}^{e} q_{j,k,3} \cdot D_{j,k})$.
      – For $k \in [n]$, set $\mathbf{f_k} := \{f_{k,1}, f_{k,2}, f_{k,3}\}$.

Output: Tally results of each proposal $\{\mathbf{f_k}\}_{k=1}^{n}$.
```

Fig. 7: Tally Calculation Algorithm.

Expert Vote Phase. $\mathsf{E}_{\mathsf{fld}}^{(j)} \in \mathcal{E}_{\mathsf{fld}}^{[e]}$ begins to vote when it gets $(\textsc{Vote}, \mathsf{sid}, \mathbf{b_j})$ from $\mathcal{Z}$. $\mathsf{E}_{\mathsf{fld}}^{(j)}$ parses $\mathbf{b_j}$ to $\{\mathbf{p_{j,k}}^{[n]}\}_{k=1}^{s}$ (see examples in Fig. 2) and encrypts the ballots to ciphertexts $\{(\mathbf{K_{j,k}}^{[n]}, \mathbf{K'_{j,k}}^{[n]})\}_{k=1}^{s}$. $\mathsf{E}_{\mathsf{fld}}^{(j)}$ generates a Valid Ballot NIZK proof δ_j to show its ballot is valid:

- For $k \in [s]$, $(\mathbf{K_{j,k}}^{[n]}, \mathbf{K'_{j,k}}^{[n]})$ encrypts a unit vector;
- For $l \in [n]$, $(\prod_{k=1}^{s} K_{j,k,l}, \prod_{k=1}^{s} K'_{j,k,l})$ encrypts either 0 or 1, where $\{(K_{j,k,l}, K'_{j,k,l})\}_{l=1}^{n} := (\mathbf{K_{j,k}}^{[n]}, \mathbf{K'_{j,k}}^{[n]})$.

Lastly, $\mathsf{E}_{\mathsf{fld}}^{(j)}$ posts the encrypted ballot and δ_j to $\mathcal{F}_{\textsc{Ledger}}$.

Tally Phase. Voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ computes delegated voting power of each expert and tally result of each proposal once it gets $(\textsc{CalDel}, \mathsf{sid})$ and $(\textsc{Tally}, \mathsf{sid})$ from $\mathcal{Z}$. $\mathsf{C}^{(t)}$ removes repeated ballots and invalid ballots. Voting committee computes delegated voting power and tally results based on ciphertexts with additively homomorphic property, and jointly decrypt final voting result [*].

Theorem 4 (Preferential Voting)**.** *Assume that Valid ballot* NIZK *is perfect complete, perfect special honest verifier zero knowledge, and computational sound with adversary advantage of* $\mathsf{Adv}_{\mathsf{NIZK},Ballot}^{\mathsf{Sound}}(1^\kappa, \mathcal{A})$*. Assume Lifted Elgamal encryption* Enc *is IND-CPA secure with adversary advantage of* $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A})$*. The protocol* $\Pi_{\mathrm{VOTE1}}^{c,\mu,s,n}$ *in Fig. 8 UC-realises* $\mathcal{F}_{\mathrm{VOTE}}^{c,\mu,s,n}$ *in Fig. 5 in* $\{\mathcal{F}_{\textsc{Ledger}}, \mathcal{F}_{\mathrm{DBKG}}^{c,\mu,1}\}$*-hybrid world against a static adversary with distinguishing advantage*

$$(2ns + e) \cdot \mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{NIZK},Ballot}^{\mathsf{Sound}}(1^\kappa, \mathcal{A}).$$

Its proof can be found in the supporting document (Sec. 3.2).

5.3 Stage 2: Threshold Voting protocol $\Pi_{\mathsf{VOTE2}}^{c,\mu,s,n}$

Assume that disqualified voters and experts who do not follow $\Pi_{\mathrm{VOTE1}}^{c,\mu,s,n}$ have been banned from participating the second stage, we give threshold voting protocol $\Pi_{\mathrm{VOTE2}}^{c,\mu,s,n}$ in Fig. 9 to realise $\mathcal{F}_{\mathrm{VOTE}}^{c,\mu,s,n}$ in $\{\mathcal{F}_{\textsc{Ledger}}, \mathcal{F}_{\mathrm{DBKG}}^{c,\mu,s}\}$*-hybrid world.*

In voter cast phase and expert vote phase, experts and voters need generate Unit Vector NIZK to prove the ballot for each proposal encrypted a unit vector (Cf. Sec. 2.5 in supporting document). In tally phase, voting committee computes the number of $\textsc{Yes}$ votes, $\textsc{No}$ votes and $\textsc{Abstain}$ votes for each shortlisted proposal.

Theorem 5 (Threshold Voting)**.** *Assume Unit Vector* NIZK *is perfect complete, perfect special honest verifier zero knowledge, and computational sound with adversary advantage of* $\mathsf{Adv}_{\mathsf{NIZK},Unit}^{\mathsf{Sound}}(1^\kappa, \mathcal{A})$*. Assume Lifted Elgamal encryption* Enc *is IND-CPA secure with adversary advantage of* $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A})$*. The protocol* $\Pi_{\mathrm{VOTE2}}^{c,\mu,s,n}$ *in Fig. 9 UC-realises* $\mathcal{F}_{\mathrm{VOTE}}^{c,\mu,s,n}$ *in Fig. 5 in* $\{\mathcal{F}_{\mathrm{DBKG}}^{c,\mu,s}, \mathcal{F}_{\textsc{Ledger}}\}$*-hybrid world against a static adversary with distinguishing advantage* $((3 + e) \cdot s + 3s) \cdot \mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{NIZK},Unit}^{\mathsf{Sound}}(1^\kappa, \mathcal{A})$.

Its proof can be found in the supporting document (Sec. 3.3).

6 Evolving Committee

As mentioned before, voting committee need to generate global key pair(s) and reveal the final tally results. For the reason that the whole voting epoch might take relatively long time (*e.g.*, 1-3 month), to avoid voting committee to be online and hold global key pair(s) shares for a long time, our proposed system support evolving committee to replace voting committee periodically.

In each round, stake holders who want to join voting committee need to lock some stakes on blockchain to show their honesty and play cryptographic sortition privately. Once a party wins sortition, it posts sortition proof on blockchain and announces its identity as voting committee member in current round. We give cryptographic sortition ideal functionality $\mathcal{F}_{Sortition}^{n}$ in Sec. 5 of the supporting document to show how a voting committee is generated.

In the first round, the first voting committee is online to generate global key pairs with $\Pi_{\mathrm{DBKG}}^{n,t,m}$. Starting from the second round to the last round, we have two voting committees online, previous committee gets identities of the next committee and re-shares the global secret keys to the next committee. At the end of voting epoch, the voting committee performs the Tally tasks (Cf. Fig. 8 and Fig. 9).

Honest Majority. In terms of security, with overwhelming probability, the majority of the voting committee members in every round are honest, which can guarantee the privacy of ballots and protocol termination. If a cheating voting

[*] Note that expert's ballot is not considered in final tally, if it does not have delegated voting power.

Stage 1: Preferential Voting Protocol $\Pi^{c,\mu,s,n}_{\textsc{Vote1}}$

Initialisation Phase:

- Upon receiving $(\textsc{Init}, \mathsf{sid})$ from $\mathcal{Z}$, the voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ sends $(\textsc{KeyGen}, \mathsf{sid}, \mathsf{C}^{(t)})$ and $(\textsc{ReadKeyShare}, \mathsf{sid}, \mathsf{C}^{(t)})$ to $\mathcal{F}^{c,\mu,1}_{\mathrm{DBKG}}$, then receives $(\textsc{ReadKeyShareReturn}, \mathsf{sid}, \mathsf{psk}_i)$.

Voter Cast Phase:

- Upon receiving $(\textsc{Cast}, \mathsf{sid}, \mathbf{a}_i, \eta_i)$ from $\mathcal{Z}$, the voter $\mathsf{V}^{(i)}_{\mathsf{fld}} \in \mathcal{V}^{[v]}_{\mathsf{fld}}$ does the following:
  - Send $(\textsc{ReadPK}, \mathsf{sid})$ to $\mathcal{F}^{c,\mu,1}_{\mathrm{DBKG}}$ and receive $(\textsc{ReadPKReturn}, \mathsf{gpk}, \{\mathsf{ppk}_a\}_{a=1}^{c})$;
  - Parse $\mathbf{a}_i$ to $(\{\mathbf{v_{i,k}}^{[n]}\}_{k=1}^{s}, \mathbf{v}'_{\mathbf{i}}{}^{[e]})$, select $\{\mathbf{r_{i,k}}^{[n]}\}_{k=1}^{s} \leftarrow (\mathbb{Z}_q)^{[n*s]}$, and $\mathbf{r}'_{\mathbf{i}}{}^{[e]} \leftarrow (\mathbb{Z}_q)^{[e]}$;
  - For $k \in [s]$, compute $(\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]}) := \mathsf{Enc}_{\mathsf{gpk}}(\mathbf{v_{i,k}}^{[n]}; \mathbf{r_{i,k}}^{[n]})$;
  - Compute $(\mathbf{B_i}^{[e]}, \mathbf{B}'_{\mathbf{i}}{}^{[e]}) := \mathsf{Enc}_{\mathsf{gpk}}(\mathbf{v}'_{\mathbf{i}}{}^{[e]}; \mathbf{r}'_{\mathbf{i}}{}^{[e]})$;
  - Generate Valid Casting NIZK proof σ_i:
    - $*$ For $k \in [s]$, $(\mathbf{A_{i,k}}^{[n]} || \mathbf{B_i}^{[e]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]} || \mathbf{B}'_{\mathbf{i}}{}^{[e]})$ encrypts a unit vector;
    - $*$ For $l \in [n]$, $(\prod_{k=1}^{s} A_{i,k,l}, \prod_{k=1}^{s} A'_{i,k,l})$ encrypts either 0 or 1, where $\{(A_{i,k,l}, A'_{i,k,l})\}_{k=1,l=1}^{s,n} := (\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]})$.
  - Post $(\{(\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]})\}_{k=1}^{s}, (\mathbf{B_i}^{[e]}, \mathbf{B}'_{\mathbf{i}}{}^{[e]}), \sigma_i, \eta_i)$ to $\mathcal{F}_{\textsc{Ledger}}$.

Expert Vote Phase:

- Upon receiving $(\textsc{Vote}, \mathsf{sid}, \mathbf{b_j})$ from $\mathcal{Z}$, the expert $\mathsf{E}^{(j)}_{\mathsf{fld}} \in \mathcal{E}^{[e]}_{\mathsf{fld}}$ does the following:
  - Send $(\textsc{ReadPK}, \mathsf{sid})$ to $\mathcal{F}^{c,\mu,1}_{\mathrm{DBKG}}$, and receive $(\textsc{ReadPKReturn}, \mathsf{gpk}, \{\mathsf{ppk}_a\}_{a=1}^{c})$ from $\mathcal{F}^{c,\mu,1}_{\mathrm{DBKG}}$;
  - Parse $\mathbf{b_j}$ to $(\{\mathbf{p_{j,k}}^{[n]}\}_{k=1}^{s})$, select $\{\mathbf{r_{j,k}}^{[n]}\}_{k=1}^{s} \leftarrow (\mathbb{Z}_q)^{[n*s]}$;
  - For $k \in [s]$, compute $(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]}) := \mathsf{Enc}_{\mathsf{gpk}}(\mathbf{p_{j,k}}^{[n]}; \mathbf{r_{j,k}}^{[n]})$;
  - Generate Valid Voting NIZK proof δ_j:
    - $*$ For $k \in [s]$, $(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})$ encrypts a unit vector;
    - $*$ For $l \in [n]$, $(\prod_{k=1}^{s} K_{j,k,l}, \prod_{k=1}^{s} K'_{j,k,l})$ encrypts either 0 or 1, where $\{(K_{j,k,l}, K'_{j,k,l})\}_{l=1}^{n} := (\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})$.
  - Post $(\{(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})\}_{k=1}^{s}, \delta_j)$ to $\mathcal{F}_{\textsc{Ledger}}$.

Tally Phase:

- Upon receiving $(\textsc{CalDel}, \mathsf{sid})$ from $\mathcal{Z}$, the voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ does the following:
  - Fetch $\{(\{(\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]})\}_{k=1}^{s}, (\mathbf{B_i}^{[e]}, \mathbf{B}'_{\mathbf{i}}{}^{[e]}), \sigma_i, \eta_i)\}_{i=1}^{v}$ and $\{(\{(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})\}_{k=1}^{s}, \delta_j)\}_{j=1}^{e}$;
  - Check if $\mathsf{Verify}(\{(\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]})\}_{k=1}^{s}, (\mathbf{B_i}^{[e]}, \mathbf{B}'_{\mathbf{i}}{}^{[e]}), \sigma_i) = 1$ for $i \in [v]$, remove all the invalid casting ballots. If there are repeated ciphertexts in $\{(\{(\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]})\}_{k=1}^{s}, (\mathbf{B_i}^{[e]}, \mathbf{B}'_{\mathbf{i}}{}^{[e]}))\}_{i=1}^{v}$, remove all the repeated casting ballots except the first one sent to $\mathcal{F}_{\textsc{Ledger}}$. Set $\mathsf{VI}^{[\alpha]}_{\mathsf{fld}}$ as a set of voter index in new ascending order who provides valid ballots;
  - Check if $\mathsf{Verify}(\{(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})\}_{k=1}^{s}, \delta_j) = 1$ for $j \in [e]$, remove all the invalid voting ballots. If there are repeated ciphertexts in $\{(\{(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})\}_{k=1}^{s})\}_{j=1}^{e}$, remove all the repeated voting ballots except the first one sent to $\mathcal{F}_{\textsc{Ledger}}$. Set $\mathsf{EI}^{[\beta]}_{\mathsf{fld}}$ as a set of voter index in new ascending order who provides valid ballots;
  - Remove the ciphertexts sent by experts/voters, and sent to invalid experts, denote the rest ciphertexts by $\{(\{(\mathbf{A_{i,k}}^{[n]}, \mathbf{A}'_{\mathbf{i,k}}{}^{[n]})\}_{k=1}^{s}, (\mathbf{B_i}^{[\beta]}, \mathbf{B}'_{\mathbf{i}}{}^{[\beta]}), \sigma_i, \eta_i)\}_{i=1}^{\alpha}$ and $\{(\{(\mathbf{K_{j,k}}^{[n]}, \mathbf{K}'_{\mathbf{j,k}}{}^{[n]})\}_{k=1}^{s}, \delta_j)\}_{j=1}^{\beta}$;
  - Compute $\mathbf{I}^{[\beta]} := \prod_{i=1}^{\alpha} (\mathbf{B_i}^{[\beta]})^{\eta_i}$, $\mathbf{I'}^{[\beta]} := \prod_{i=1}^{\alpha} (\mathbf{B}'_{\mathbf{i}}{}^{[\beta]})^{\eta_i}$;
  - $\mathcal{C}^{[c]}$ jointly decrypt $(\mathbf{I}^{[\beta]}, \mathbf{I'}^{[\beta]})$ to $\{m_j\}_{j=1}^{\beta}$;
  - Post $\{m_j\}_{j=1}^{\beta}$ to $\mathcal{F}_{\textsc{Ledger}}$.
- Upon receiving $(\textsc{Tally}, \mathsf{sid})$ from $\mathcal{Z}$, the voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ does the following:
  - Denote the Borda Count Voting parameters by $\{\tau_k\}_{k=1}^{s}$, for $l \in [n]$, compute the following:
    - $*$ $S_l := \prod_{i=1}^{\alpha} \prod_{k=1}^{s} (\mathbf{A_{i,k}})^{\tau_k \cdot \eta_i} \cdot \prod_{j=1}^{\beta} \prod_{k=1}^{s} (\mathbf{K_{j,k}})^{\tau_k \cdot m_j}$;
    - $*$ $S'_l := \prod_{i=1}^{\alpha} \prod_{k=1}^{s} (\mathbf{A}'_{\mathbf{i,k}})^{\tau_k \cdot \eta_i} \cdot \prod_{j=1}^{\beta} \prod_{k=1}^{s} (\mathbf{K}'_{\mathbf{j,k}})^{\tau_k \cdot m_j}$;
  - $\mathcal{C}^{[c]}$ jointly decrypt (S_l, S'_l) to f_l for $l \in [n]$;
  - Post $(\{f_l\}_{l=1}^{n})$ to $\mathcal{F}_{\textsc{Ledger}}$.
- Upon receiving $(\textsc{Reveal}, \mathsf{sid}, \mathsf{E}^{(j)}_{\mathsf{fld}})$ from $\mathcal{Z}$, the expert $\mathsf{E}^{(j)}_{\mathsf{fld}} \in \mathcal{E}^{[e]}_{\mathsf{fld}}$ posts $\{\mathbf{r_{j,k}}^{[n]}, \mathbf{p_{j,k}}^{[n]}\}_{k=1}^{s}$ to $\mathcal{F}_{\textsc{Ledger}}$, returns $(\textsc{RevealExpert}, \mathsf{sid}, \{\mathbf{r_{j,k}}^{[n]}, \mathbf{p_{j,k}}^{[n]}\}_{k=1}^{s})$ to $\mathcal{Z}$;
- Upon receiving $(\textsc{ReadTally}, \mathsf{sid})$ from $\mathcal{Z}$, the party P returns $(\textsc{ReadTallyReturn}, \mathsf{sid}, (\{f_l\}_{l=1}^{n})$ to $\mathcal{Z}$ after fetching $(\{f_l\}_{l=1}^{n})$.

Fig. 8: Stage 1: Preferential Voting protocol $\Pi^{c,\mu,s,n}_{\textsc{Vote1}}$ in $\{\mathcal{F}_{\textsc{Ledger}}, \mathcal{F}^{c,\mu,1}_{\mathrm{DBKG}}\}$-*hybrid world.*

Stage 2: Threshold Voting protocol $\Pi_{\textsc{Vote2}}^{c,\mu,s,n}$

Initialisation Phase:

- Upon receiving $(\textsc{Init}, \mathsf{sid})$ from $\mathcal{Z}$, the voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ sends $(\textsc{KeyGen}, \mathsf{sid}, \mathsf{C}^{(t)})$ and $(\textsc{ReadKeyShare}, \mathsf{sid}, \mathsf{C}^{(t)})$ to $\mathcal{F}_{\text{DBKG}}^{c,\mu,s}$, receives $(\textsc{ReadKeyShareReturn}, \mathsf{sid}, \{\mathsf{psk}_{l,i}\}_{l=1}^{s})$ from $\mathcal{F}_{\text{DBKG}}^{c,\mu,s}$.

Voter Cast Phase:

- Upon receiving $(\textsc{Cast}, \mathsf{sid}, \mathbf{v_i}, \eta_i)$ from $\mathcal{Z}$, the voter $\mathsf{V}_{\mathsf{fld}}^{(i)} \in \mathcal{V}_{\mathsf{fld}}^{[v]}$ does the following:
  - Send $(\textsc{ReadPK}, \mathsf{sid})$ to $\mathcal{F}_{\text{DBKG}}^{c,\mu,s}$, and receive $(\textsc{ReadPKReturn}, \mathsf{sid}, \{\mathsf{gpk}_l\}_{l=1}^{s}, \{\mathsf{ppk}_{l,a}\}_{l=1,a=1}^{s,c})$;
  - Parse $\mathbf{v_i}$ to $(\{\mathbf{w_{i,l}}^{[3+e]}\}_{l=1}^{s})$, select $\{\mathbf{r_{i,l}}^{[3+e]}\}_{l=1}^{s} \leftarrow (\mathcal{Z}_p)^{[(3+e)*s]}$;
  - For $l \in [s]$, compute $(\mathbf{X_{i,l}}^{[3+e]}, \mathbf{X'_{i,l}}^{[3+e]}) := \mathsf{Enc}_{\mathsf{gpk}_l}(\mathbf{w_{i,l}}^{[3+e]}, \mathbf{r_{i,l}}^{[3+e]})$;
  - Generate Unit Vector Encryption NIZK proof Δ_i to prove $(\mathbf{X_{i,l}}^{[3+e]}, \mathbf{X'_{i,l}}^{[3+e]})$ encrypts a unit vector for $l \in [s]$;
  - Post $(\{(\mathbf{X_{i,l}}^{[3+e]}, \mathbf{X'_{i,l}}^{[3+e]})\}_{l=1}^{s}, \Delta_i, \eta_i)$ to $\mathcal{F}_{\textsc{Ledger}}$.

Expert Vote Phase:

- Upon receiving $(\textsc{Vote}, \mathsf{sid}, \mathbf{b_j})$ from $\mathcal{Z}$, the expert $\mathsf{E}_{\mathsf{fld}}^{(j)} \in \mathcal{E}_{\mathsf{fld}}^{[e]}$ does the following:
  - Send $(\textsc{ReadPK}, \mathsf{sid})$ to $\mathcal{F}_{\text{DBKG}}^{c,\mu,s}$, and receive $(\textsc{ReadPKReturn}, \mathsf{sid}, \{\mathsf{gpk}_l\}_{l=1}^{s}, \{\mathsf{ppk}_{l,a}\}_{l=1,a=1}^{s,c})$;
  - Parse $\mathbf{v_j}^{[s]}$ to $(\{\mathbf{q_{j,l}}^{[3]}\}_{l=1}^{s}$, select $\{\mathbf{r'_{j,l}}^{[3]}\}_{l=1}^{s} \leftarrow (\mathcal{Z}_p)^{[s*3]}$;
  - For $l \in [s]$, compute $(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]}) = \mathsf{Enc}_{\mathsf{gpk}_l}(\mathbf{q'_{j,l}}^{[3]}; \mathbf{r'_{j,l}}^{[3]})$;
  - Generate Unit Vector Encryption NIZK proof γ_j to prove $(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]})$ encrypts a unit vector for $l \in [s]$;
  - Post $(\{(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]})\}_{l=1}^{s}, \gamma_j)$ to $\mathcal{F}_{\textsc{Ledger}}$.

Tally Phase:

- Upon receiving $(\textsc{CalDel}, \mathsf{sid})$ from $\mathcal{Z}$, the voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ does the following:
  - Fetch $\{(\{(\mathbf{X_{i,l}}^{[3+e]}, \mathbf{X'_{i,l}}^{[3+e]})\}_{l=1}^{s}, \Delta_i, \eta_i)\}_{i=1}^{v}$ and $\{(\{(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]})\}_{l=1}^{s}, \gamma_j)\}_{j=1}^{e}$;
  - Check if $\mathsf{Verify}(\{(\mathbf{X_{i,l}}^{[3+e]}, \mathbf{X'_{i,l}}^{[3+e]})\}_{l=1}^{s}, \Delta_i, \eta_i) = 1$ for $i \in [v]$, remove all the invalid casting ballots. If there are repeated ciphertexts in $\{(\{(\mathbf{X_{i,l}}^{[3+e]}, \mathbf{X'_{i,l}}^{[3+e]})\}_{l=1}^{s}, \Delta_i, \eta_i)\}_{i=1}^{v}$, remove all the repeated casting ballots except the first one sent to $\mathcal{F}_{\textsc{Ledger}}$. Set $\mathsf{VI}_{\mathsf{fld}}^{[v']}$ as a set of voter index in new ascending order who provided valid ballots;
  - Check if $\mathsf{Verify}(\{(\{(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]})\}_{l=1}^{s}, \gamma_j) = 1$ for $j \in [e]$, remove all the invalid voting ballots. If there are repeated ciphertexts in $\{(\{(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]})\}_{l=1}^{s})\}_{j=1}^{e}$, remove all the repeated voting ballots except the first one sent to $\mathcal{F}_{\textsc{Ledger}}$. Set $\mathsf{EI}_{\mathsf{fld}}^{[e]}$ as a set of voter index in new ascending order who provided valid ballots;
  - Remove the ciphertexts sent by experts/voters, and sent to invalid experts, denote the rest ciphertexts by $\{(\{(\mathbf{X_{i,l}}^{[3+e']}, \mathbf{X'_{i,l}}^{[3+e']})\}_{l=1}^{s}, \Delta_i, \eta_i)\}_{i=1}^{v'}$ and $\{(\{(\mathbf{Y_{j,l}}^{[3]}, \mathbf{Y'_{j,l}}^{[3]})\}_{j=1}^{e'}$;
  - For $j \in [e']$, for $l \in [s]$, compute $I_{j,l} := \prod_{i=1}^{v'}(X_{i,l,3+j})^{\eta_i}$, $I'_{j,l} := \prod_{i=1}^{v'}(X'_{i,l,3+j})^{\eta_i}$;
  - $\mathcal{C}^{[c]}$ jointly decrypt $(I_{j,l}, I'_{j,l})$ to $m_{j,l}$ for $j \in [e']$, for $l \in [s]$.
  - Post $(\{m_{j,l}\}_{l=1}^{s})$ to $\mathcal{F}_{\textsc{Ledger}}$.
- Upon receiving $(\textsc{Tally}, \mathsf{sid})$ from $\mathcal{Z}$, the voting committee member $\mathsf{C}^{(t)} \in \mathcal{C}^{[c]}$ does the following:
  - For $l \in [s]$, compute the following:
    - $*$ $S_{l,1} := (\prod_{i=1}^{v'}(X_{i,l,1})^{\eta_i}) \cdot (\prod_{j=1}^{e'}(Y_{i,l,1})^{m_{j,l}})$, $S'_{l,1} := (\prod_{i=1}^{v'}(X'_{i,l,1})^{\eta_i}) \cdot (\prod_{j=1}^{e'}(Y'_{i,l,1})^{m_{j,l}})$;
    - $*$ $S_{l,2} := (\prod_{i=1}^{v'}(X_{i,l,2})^{\eta_i}) \cdot (\prod_{j=1}^{e'}(Y_{i,l,2})^{m_{j,l}})$, $S'_{l,2} := (\prod_{i=1}^{v'}(X'_{i,l,2})^{\eta_i}) \cdot (\prod_{j=1}^{e'}(Y'_{i,l,2})^{m_{j,l}})$;
    - $*$ $S_{l,3} := (\prod_{i=1}^{v'}(X_{i,l,3})^{\eta_i}) \cdot (\prod_{j=1}^{e'}(Y_{i,l,3})^{m_{j,l}})$, $S'_{l,3} := (\prod_{i=1}^{v'}(X'_{i,l,3})^{\eta_i}) \cdot (\prod_{j=1}^{e'}(Y'_{i,l,3})^{m_{j,l}})$;
  - For $l \in [s]$, $\mathcal{C}^{[c]}$ jointly decrypt the following:
    - $*$ $(S_{l,1}, S'_{l,1})$ to $f_{l,1}$;
    - $*$ $(S_{l,2}, S'_{l,2})$ to $f_{l,2}$;
    - $*$ $(S_{l,3}, S'_{l,3})$ to $f_{l,3}$.
  - Post $(\{f_{l,1}, f_{l,2}, f_{l,3}\}_{l=1}^{s})$ to $\mathcal{F}_{\textsc{Ledger}}$.
- Upon receiving $(\textsc{Reveal}, \mathsf{sid}, \mathsf{E}_{\mathsf{fld}}^{(j)})$ from $\mathcal{Z}$, the expert $\mathsf{E}_{\mathsf{fld}}^{(j)} \in \mathcal{E}_{\mathsf{fld}}^{[e]}$ posts $\{\mathbf{r_{j,k}}^{[s]}, \mathbf{p_{j,k}}^{[s]}\}_{k=1}^{s}$ to $\mathcal{F}_{\textsc{Ledger}}$, returns $(\textsc{RevealExpert}, \mathsf{sid}, \{\mathbf{r_{j,k}}^{[s]}, \mathbf{p_{j,k}}^{[s]}\}_{k=1}^{s})$ to $\mathcal{Z}$;
- Upon receiving $(\textsc{ReadTally}, \mathsf{sid})$ from $\mathcal{Z}$, the party P fetches $(\{f_{l,1}, f_{l,2}, f_{l,3}\}_{l=1}^{s})$, and return $\mathcal{Z}$ the message $(\textsc{ReadTallyReturn}, \mathsf{sid}, (\{f_{l,1}, f_{l,2}, f_{l,3}\}_{l=1}^{s})$.

Fig. 9: Stage 2: Threshold Voting protocol $\Pi_{\textsc{Vote2}}^{c,\mu,s,n}$ $\{\mathcal{F}_{\textsc{Ledger}}, \mathcal{F}_{\text{DBKG}}^{c,\mu,s}\}$-*hybrid world.*

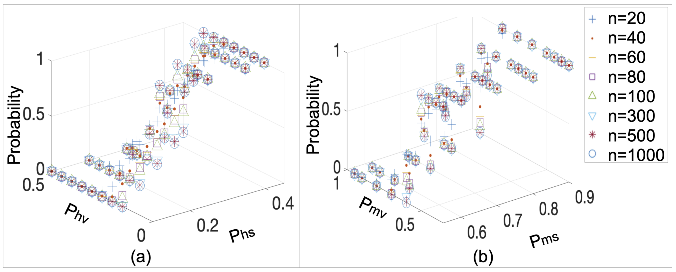


Fig. 10: (a) The probability that at least R_{hv} of the n voting committee members are honest if R_{hs} of the whole stakes are honest; (b) The probability that adversary corrupts at least R_{mv} of the n voting committee members if it takes over R_{ms} of the whole stakes.

committee member is detected, it loses all the deposit, and get banned by decision-making system forever.

Assume the majority of the stake of all the registered voters is honest; therefore, the probability that a selected committee member is honest is $p = 1/2 + \varepsilon$ for any $\varepsilon \in (0, 1/2]$. Let X be the number of malicious committee members selected among all λ' committee members. Since $\lambda' = \omega(\log \lambda)$, by Chernoff bound, we have

$$\begin{aligned}\Pr[X \geq \lambda'/2] &= \Pr[X \geq (1+\delta)(1/2-\varepsilon)\lambda'] \\ &< \exp(-\delta^2(1/2-\varepsilon)\lambda'/4) \\ &= \frac{1}{\exp(\omega(\log\lambda))} = \mathsf{negl}(\lambda)\end{aligned}$$

for $\delta = 2\varepsilon/(1-2\varepsilon)$.

We discuss the honest of voting majority in Fig. 10. Combinatorial analysis of voting committee's honesty can be found in Sec. 4 of supporting document. Based on the analysis, we suggest that committee size should be at least 70 members, and at least 60% of the committee members should participant for threshold signature. Within the assumptions and recommended parameters, the probability of adversary preventing honest participants to put the corresponding transaction on blockchain is less than 0.0001 for a single decision-making period. The probability of adversarial control over the decision-making fund (within the same assumptions and parameters) is negligibly small.

Handover. Handover ideal functionality $\mathcal{F}_{Handover}[\mathbb{G}]$ is presented in Fig. 11 to support Evolving Committee. It interacts with two continuous voting committees: previous committee $\mathcal{C}_r^{[n_r]}$, and new committee $\mathcal{C}_{r+1}^{[n_{r+1}]}$. Upon receiving $(\textsc{Handover}, sid, \{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}, \{\mathsf{psk}_{v,i}\}_{v=1}^{m})$ from previous committee $\mathsf{C}_r^{(i)} \in \mathcal{C}_r^{[n_r]}$, $\mathcal{F}_{Handover}[\mathbb{G}]$ reconstructs the global secret keys $\{\mathsf{gsk}_v\}_{v=1}^{m}$ and notifies $\mathcal{S}$ by $(\textsc{HandoverNotify}, \mathsf{sid}, \mathsf{C}_r^{(i)}, \{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r})$.

We model that adversary can choose shares for the corrupted new committee members by letting $\mathcal{S}$ send $(\textsc{CorruptShares}, \mathsf{sid}, \{i, \{\mathsf{psk}_{v,i}\}_{v=1}^{m}\}_{\mathsf{C}_{r+1}^{(i)} \in \mathcal{C}_{\mathsf{c},r+1}^{[n_{r+1}]}}))$ to $\mathcal{F}_{Handover}[\mathbb{G}]$. For $v \in [m]$, $\mathcal{F}_{Handover}[\mathbb{G}]$ constructs random polynomial based on all the shares and gsk_v. Then every new committee member can send $(\textsc{ReadNewShare}, \mathsf{sid})$ to $\mathcal{F}_{Handover}[\mathbb{G}]$ asking for their shares. $\mathcal{F}_{Handover}[\mathbb{G}]$ computes partial secret key and partial global key for $\mathsf{C}_{r+1}^{(j)}$ based on the polynomials generated in last step, and returns $(\textsc{ReadNewShareReturn}, \mathsf{sid}, \{\mathsf{psk}'_{v,j}\}_{v=1}^{m}, \{\mathsf{ppk}'_{v,j}\}_{v=1,j=1}^{m,n_{r+1}})$.

$\mathcal{F}_{Handover}[\mathbb{G}]$

$\mathcal{F}_{Handover}[\mathbb{G}]$ interacts $\mathcal{C}_r^{[n_r]}, \mathcal{C}_{r+1}^{[n_{r+1}]}$ of which the thresholds are t_r and t_{r+1} respectively, and maintains a set $\mathbf{O}$ (Initially set to $\emptyset$). Denote $\mathcal{C}_{\mathsf{c},r+1}$ as corrupted voting committee members in $\mathcal{C}_{r+1}^{[n_{r+1}]}$, and $\mathcal{C}_{\mathsf{h},r+1}$ as honest voting committee members in $\mathcal{C}_{r+1}^{[n_{r+1}]}$, $|\mathcal{C}_{\mathsf{c},r+1}^{[n_{r+1}]}| + |\mathcal{C}_r^{[n_{r+1}]}| = n_{r+1}$, and $|\mathcal{C}_{\mathsf{h},r+1}| \geq t_{r+1}$.

$\mathcal{F}_{Handover}[\mathbb{G}]$ does the following:

- Upon receiving $(\textsc{Handover}, sid, \{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}, \{\mathsf{psk}_{v,i}\}_{v=1}^{m})$ from $\mathsf{C}_r^{(i)} \in \mathcal{C}_r^{[n_r]}$:
  - Set $\mathbf{O} := \mathbf{O} \cup \{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}$;
  - If there are more than t_r values of $\{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}$ in $\mathbf{O}$ are the same, assert all $\{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}$ to this value, and continue to next step;
  - For $v \in [m]$, compute $\mathsf{gsk}_v := \prod_{j \in \mathbf{R}} \lambda_{v,j} \cdot \mathsf{psk}_{v,j}$, where $\mathbf{R}$ is the set of honest parties' indexes in $\mathcal{C}_r^{[n_r]}$, $|\mathbf{R}| = t_r$, $\{\lambda_{v,j}\}_{v \in [m], j \in \mathbf{R}}$ are Lagrange Interpolation coefficients;
  - Send $(\textsc{HandoverNotify}, \mathsf{sid}, \mathsf{C}_r^{(i)}, \{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r})$ to $\mathcal{S}$.
- Upon receiving $(\textsc{CorruptShares}, \mathsf{sid}, \{i, \{\mathsf{psk}_{v,i}\}_{v=1}^{m}\}_{\mathsf{C}_{r+1}^{(i)} \in \mathcal{C}_{\mathsf{c},r+1}^{[n_{r+1}]}}))$ from $\mathcal{S}$:
  - Set $a := t_{r+1} - |\mathcal{C}_{\mathsf{c},r+1}| - 1$, $\mathcal{C}'_{\mathsf{h},r+1} \subset \mathcal{C}_{\mathsf{h},r+1}$, $|\mathcal{C}'_{\mathsf{h},r+1}| = a$, select $\{\mathsf{psk}_{v,i}\}_{\mathsf{C}_{r+1}^{(i)} \in \mathcal{C}'_{\mathsf{h},r+1}, v \in [m]} \leftarrow (\mathbb{Z}_q)^{[m \cdot a]}$;
  - For $v \in [m]$, construct random polynomial $F_v(z) := \sum_{t=0}^{t_{r+1}-1} a_{v,t} \cdot z^t$ under the restriction $F_v(j) = \mathsf{psk}_{j,v}$ for $\mathsf{C}_{r+1}^{(i)} \in \{\mathcal{C}'_{\mathsf{h},r+1} \cup \mathcal{C}_{\mathsf{c},r+1}\}$, and $F_v(0) = \mathsf{gsk}_v$.
- Upon receiving $(\textsc{ReadNewShare}, \mathsf{sid})$ from $\mathsf{C}_{r+1}^{(j)} \in \mathcal{C}_{\mathsf{c},r+1}^{[n_{r+1}]}$:
  - For $v \in [m]$, $j \in [n_{r+1}]$, compute $\mathsf{psk}'_{v,j} := F_v(j)$, $\mathsf{ppk}'_{v,j} := g^{s_{v,j}}$;
  - Send $(\textsc{ReadNewShareReturn}, \mathsf{sid}, \{\mathsf{psk}'_{v,j}\}_{v=1}^{m}, \{\mathsf{ppk}'_{v,j}\}_{v=1,j=1}^{m,n_{r+1}})$ to $\mathsf{C}_{r+1}^{(j)}$;
- Upon receiving $(\textsc{ReadPK}, \mathsf{sid})$ from any party, compute $\mathsf{gpk}_v := \prod_{j=1}^{t_{r+1}} (\mathsf{ppk'}_{v,j})^{\gamma_{v,j}}$ for $v \in [m]$ and return $(\textsc{ReadPKReturn}, \mathsf{sid}, \{\mathsf{gpk}_v\}_{v=1}^{m}, \{\mathsf{ppk}'_{v,i}\}_{v=1,i=1}^{m,n_{r+1}})$ to the requester.

Fig. 11: The Key Handover functionality, $\mathcal{F}_{Handover}[\mathbb{G}]$.

We give $\Pi_{Handover}[\mathbb{G}]$ in $\{\mathcal{F}_{\textsc{Ledger}}\}$ *-hybrid world* to realise $\mathcal{F}_{Handover}[\mathbb{G}]$ in Fig. 12. The previous committee share their partial secret keys $\{\mathsf{psk}_{v,i}\}_{v=1}^{m}$ to new committee, and encrypt shares with recipients' public keys. New committee can verify if the shares are valid through verifiable secret sharing, and compute their own partial secret keys and partial public keys.

Theorem 6 (Handover). *Let* $t_{r+1} \geq 1/2 \cdot \mathcal{C}_{r+1}^{[n_{r+1}]}$, *assume Lifted Elgamal encryption* Enc *is IND-CPA secure with adversary advantage of* $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND\text{-}CPA}}(1^\kappa, \mathcal{A})$. *Assume Correct Decryption* NIZK *is perfect complete, perfect special*

Handover Protocol $\Pi_{Handover}[\mathbb{G}]$

Assume that the party have its own key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$.

- Upon receiving $(\text{HANDOVER}, sid, \{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}, \{\mathsf{psk}_{v,i}\}_{v=1}^{m})$ from $\mathcal{Z}$, $\mathsf{C}_r^{(i)} \in \mathcal{C}_r^{[n_r]}$ does the following:
  - Select random polynomial $F_{v,i}(z) := \sum_{t=0}^{t_{r+1}-1} a_{v,i,t} \cdot z^t$, where $a_{v,i,0} = \mathsf{psk}_{v,i}$, $\{a_{v,i,t}\}_{t=1}^{t_{r+1}-1} \leftarrow (\mathbb{Z}_q)^{[t_{r+1}-1]}$;
  - Compute $s_{v,i,j} := F_{v,i}(j)$ for $j \in [n_{r+1}]$;
  - Choose $\{r_{v,i,j}\}_{j=1}^{n_{r+1}} \leftarrow (\mathbb{Z}_q)^{[n_{r+1}]}$, compute $(A_{v,i,j}, B_{v,i,j}) := \mathsf{Enc}_{\mathsf{pk}_j}(s_{v,i,j}; r_{v,i,j})$ for $j \in [n_{r+1}]$, $v \in [m]$;
  - Compute $H_{v,i,t} := \mathsf{Com}_{\mathsf{ck}}(a_{v,i,t}; 0)$ for $v \in [m]$, $t \in [1, t_{r+1} - 1]$;
  - Post $(\{H_{v,i,t}\}_{v=1,t=1}^{m,t_{r+1}-1}, \{A_{v,i,j}, B_{v,i,j}\}_{v=1,j=1}^{m,n_{r+1}}, \{\mathsf{ppk}_{v,j}\}_{j=1,v=1}^{n_r,m})$ to $\mathcal{F}_{\text{LEDGER}}$.
- Upon receiving $(\text{READNEWSHARE}, \mathsf{sid})$ from $\mathcal{Z}$, $\mathsf{C}_{r+1}^{(i)} \in \mathcal{C}_{r+1}^{[n_{r+1}]}$ does the following:
  - Fetch $(\{\{H_{v,j,t}\}_{t=1}^{t_{r+1}-1}, A_{v,j,i}, B_{v,j,i}, \mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r})$;
  - Assert $\{\mathsf{ppk}_{v,j}\}_{v=1,j=1}^{m,n_r}$ following majority rule;
  - Compute $s_{v,j,i} = \mathsf{Dec}_{\mathsf{sk}_j}(A_{v,j,i}, B_{v,j,i})$ for $v \in [m]$, $j \in [n_r]$;
  - For $j \in [n_r]$, for $v \in [m]$, set $H_{v,j,0} = \mathsf{ppk}_{v,j}$, check if $g^{s_{v,j,i}} = \prod_{t=0}^{t_{r+1}-1} (H_{v,j,t})^{i^t}$. If this verification fails for $\mathsf{C}_r^{(j)}$, post $(\text{COMPLAINT}, s_{v,j,i}, \sigma_i)$ to $\mathcal{F}_{\text{LEDGER}}$, where σ_i is Correct Decryption NIZK:

$$\sigma_i \leftarrow \mathsf{NIZK}\left\{ \begin{array}{l} (g, \mathsf{pk}_i, \{A_{v,j,i}, B_{v,j,i}\}_{j=1}^{n_r}), \\ (\mathsf{sk}_i) : \mathsf{pk}_i = g^{\mathsf{sk}_i} \\ \wedge s_{v,j,i} = \mathsf{Dec}_{\mathsf{sk}_i}(A_{v,j,i}, B_{v,j,i}) \end{array} \right\}$$

  - If there is a valid complain to $\mathcal{F}_{\text{LEDGER}}$ about $\mathsf{C}_r^{(j)} \in \mathcal{C}_r^{[n_r]}$, set $\mathbf{V} := [n_r] \setminus \{j\}$, select any t_{r+1} values from $\mathbf{V}$ as $\mathbf{V}'$;
  - For $v \in [m]$, compute $\mathsf{psk}'_{v,i} := \sum_{j \in \mathbf{V}'} s_{v,j,i} \cdot \gamma_{v,j}$ by interpolation;
  - Compute $\mathsf{ppk}'_{v,k} := \prod_{j \in \mathbf{V}'} (\prod_{t=0}^{t_{r+1}-1} (H_{v,j,t})^{k^t})^{\gamma_{v,j}}$ for $v \in [m]$, $k \in [n_{r+1}]$, post $(\{\{\mathsf{ppk}'_{v,k}\}_{v=1,k=1}^{m,n_{r+1}}\})$ to $\mathcal{F}_{\text{LEDGER}}$.
- Upon receiving $(\text{READPK}, \mathsf{sid})$ from any party P, P fetches $\{\mathsf{ppk}'_{v,j}\}_{v=1,j=1}^{m,n_{r+1}}$, computes $\mathsf{gpk}_v := \prod_{j=1}^{t_{r+1}} (\mathsf{ppk}'_{v,j})^{\gamma_{v,j}}$, and return $(\text{READPKRETURN}, \mathsf{sid}, \{\mathsf{gpk}_v\}_{v \in [m]}, \{\mathsf{ppk}'_{v,j}\}_{v=1,j=1}^{m,n_{r+1}})$ to $\mathcal{Z}$.

Fig. 12: Handover Protocol, $\Pi_{Handover}[\mathbb{G}]$ in $\{\mathcal{F}_{\text{LEDGER}}\}$ *-hybrid world.*

honest verifier zero knowledge, and computational sound with adversary advantage of $\mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{NIZK},Dec}(1^\kappa, \mathcal{A})$*. Assume* Enc *is IND-CPA secure with adversary advantage of* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{Enc}}(1^\kappa, \mathcal{A})$*. The protocol* $\Pi_{Handover}[\mathbb{G}]$ *in Fig.12 UC-realises* $\mathcal{F}_{Handover}[\mathbb{G}]$ *in Fig.11 in* $\{\mathcal{F}_{\text{LEDGER}}\}$ *-hybrid world against static corruption up to* $n_{r+1} - t_{r+1} - 1$ *parties with distinguishing advantage upper bounded by* $(m \cdot n_{r+1}) \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{Enc}}(1^\kappa, \mathcal{A}) + (n_{r+1} - t_{r+1} - 1) \cdot \mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{NIZK},Dec}(1^\kappa, \mathcal{A})$.

The proof can be found in supporting document (Sec. 3.4).

7 Security and Performance

7.1 Secuirty

We first examine how our decision-making system satisfies the design properties:

- **Privacy:** Voters' ballots are encrypted by Lift Elgamal encryption with public keys generated by voting committee. Based on DDH assumption, it is infeasible to infer the original message from the ciphertexts. Moreover, during the tally phase, voting committee members compute tally based on additively homomorphic property without decrypting ballots. If majority of the voting committee members are uncorrupted, ballots privacy is guaranteed;
- **Fairness:** In the pre-voting epoch, the final proposals are made public by deadline based on a two-stage project proposing procedure, which separates submission of proposal commitment from revealing the proposals on blockchain. Therefore, proposers cannot know other proposals in advance based on the hiding property of commitment. In the voting epoch, voters and experts should submit encrypted ballots together with zero-knowledge proofs. Because of DDH assumption, no one can infer original messages from ciphertexts, therefore voters and experts cannot change their ballots by counting on others' outputs. Additionally, each ballot contains a NIZK proof, even if some party directly copy-pastes and randomises others' ballots to avoid duplication, it cannot provide valid NIZK proof. Moreover, the finally tally results are only revealed at the end of voting epoch, voters and experts cannot change their decisions after seeing the final tally results;
- **Efficiency and Flexibility:** In the pre-voting epoch, we use a commitment based two-stage project proposing procedure to avoid advantage over late submission and guarantee fairness, which improves the proposal submission efficiency. In the voting epoch, we introduce a new two-stage voting to save voters and experts' voting effort, which improves overall voting efficiency. Besides, we propose a new DBKG protocol to generate distributed keys with amortised communication cost of $\mathcal{O}(n)$. Moreover, our Handover protocol supports changing voting committee members flexibly in each round;
- **End-to-end verifiability:**
  Individual Verifiability. Voters and experts can verify if their ballots are recorded on blockchain, which is guaranteed by the immutability, traceability and auditability of blockchain. As mentioned before, the honesty of an untrusted voting device can be assured by cryptographic techniques such as Benaloh challenge [29], [30] and the protocol proposed in [31]. In addition, voters and experts can validate the correctness of tally results and get decrypted delegated voting power of all experts by checking NIZK proofs. Therefore, they can check if the correct encrypted tally results contain their ballots by additively homomorphically computing based on all the encrypted ballots, voting power of voters and delegated voting power.

Universal Verifiability. Everyone can check the messages posted on blockchain to verify fairness of proposal submission and voting, including proposal commitment, encrypted ballots, and final tally results. Based on the encrypted ballots, voters' voting power, experts' delegated voting power, public keys, decryption shares and final tally results on blockchain, everyone can verify correctness of final tally result.

Eligibility Verifiability. To participant voting, voters and experts are required to lock stakes on blockchain and submit encrypted ballots to blockchain. As all transactions on blockchain are signed by the sender's secret key, everyone can check if the final tally contains ballots from valid parties together with universal verifiability.

7.2 Performance

We compared our voting scheme with the existing voting schemes, in terms of basic security requirements including privacy, fairness, end-to-end (E2E) verifiability, and new properties including universal composability (UC) security, flexibility and 2-stage voting. All the voting schemes guarantee ballots privacy and end-to-end verifiability, some of the schemes cannot guarantee fairness which gives voters additional advantage. For example, Yu etal. [32] introduced a single voting administrator to trigger and reveal tally, which breaks fairness if it reveals partial tally to some voters. We check if the schemes are proved under the UC framework and find that only [1], [33] are universal composable. The comparison results in in Table 1 shows that our voting scheme is the only one that provides UC security and flexible 2-stage voting to save voting efforts and improve voting efficiency besides satisfying all the security properties.

We now present the theoretical analysis of our decision-making system. In the DBKG protocol $\Pi_{\text{DBKG}}^{n,\mu,m}$ when generating $\mathcal{O}(n)$ keys, every party needs to compute and send $\mathcal{O}(n)$ cipher-texts and commitments. The Correct Sharing NIZK has $\mathcal{O}(n)$ computation and communication cost for proving and verification. Therefore, the overall cost for DBKG protocol is amortised $\mathcal{O}(n)$. In the Preferential Voting stage, every voter computes its ballots with $\mathcal{O}(n \cdot s + e)$ cost. The cost of Valid Casting NIZK proof for voter's ballot validation is $\mathcal{O}(s \cdot \log(n+e) + n + s)$ for proving, verifier's computation cost is $\mathcal{O}(s \cdot \log(n+e))$. An expert costs $\mathcal{O}(n \cdot s)$ to generate ballots, the cost of Valid Voting NIZK proof for expert's ballot validation is $\mathcal{O}(s \cdot \log n + n + s)$ for proving, $\mathcal{O}(s \cdot \log n)$ for computation on verifier's side. In the Tally phase, the computation cost is $\mathcal{O}(e \cdot s)$, communication cost is $\mathcal{O}(s)$. Hence, overall cost of $\Pi_{\text{VOTE1}}^{c,\mu,s,n}$ is $\mathcal{O}(v \cdot s \cdot n)$. Similarly, the overall cost of $\Pi_{\text{VOTE2}}^{c,\mu,s,n}$ is $\mathcal{O}(v \cdot s \cdot e)$. In the Handover protocol, for a previous committee member, computation cost is $\mathcal{O}(n_{r+1})$, communication cost is $\mathcal{O}(\max(n_{r+1}, n_r))$. For a new committee member, the computation cost is $\mathcal{O}(\max(n_{r+1}, n_r))$, the communication cost is $\mathcal{O}(n_{r+1})$. The overall communication cost of $\Pi_{Handover}[\mathbb{G}]$ is $\mathcal{O}(\max(n_{r+1}, n_r) \cdot n_{r+1})$, and computation cost is $\mathcal{O}(\max(n_{r+1}, n_r)^2)$ for per key.

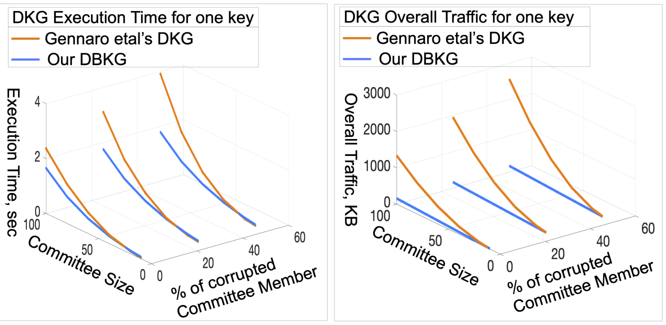


Fig. 13: DKG Execution Time and Overall Traffic.

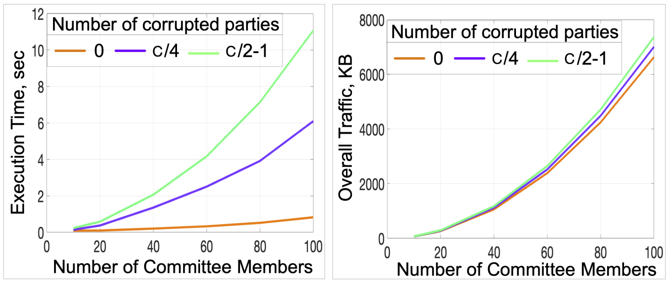


Fig. 14: Handover Execution Time and Overall Traffic.

We developed a special set of tests as a part of the cryptographic library to evaluate performance of the cryptographic protocols in the proposed decision-making system. The configuration of the work station is : Intel Core i7-6500U CPU @ 2.50GHz, 16GB RAM, running Linux Ubuntu 16.04 64 bit, Scala Version "2.12.3", OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-2ubuntu1.16.04.3-b11), and org.bouncycastle "1.58" as the Elliptic Curve Math Library. The parameter of the elliptic curve is Secp256k1.

To evaluate the performance of our proposed DBKG protocol, we benchmarked the execution time and overall traffic for generating one key comparing with Gennaro *et al.*'s DKG in Fig. 13, regarding to different committee size (from 10 to 100) and different percentages of corrupted committee members (from 0 to $c/2-1$, where c is the number of voting committee members). For Lift Elgamal encryption, segments are taken of minimal size (*e.g.*, 8-bit) to minimise overall time of protocol execution as DLOG bruteforcing during cyphertexts decryption takes significant time for bigger segments. For example, the overall time with 5 members is 29 sec for 16-bit segments, while 0.5 s for 8-bit segments. The larger segments decrease overall traffic about linearly (for the case with 5 members the overall traffic is 32 KB for 16-bit segments instead of 61 KB for 8-bit segments).

In addition, we tested the execution time and overall traffic about Handover protocol in Fig. 14, regarding to different numbers of committee members c (from 10 to 100), and different numbers of corrupted parties (from 0 to $c/2-1$), the segment size is 32.

Fig. 15 demonstrates the performance evaluation of Cor-

TABLE 1: Comparison with other voting schemes.

Schemes	[34]	[35]	[36]	[37]	[38]	[32]	[39]	[1]	[40]	[41]	[42]	[43]	[44]	[33]	[45]	[46]	[47]	[48]	Ours
Privacy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fairness	✓	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
E2E Verifiability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
UC Security	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Flexibility	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
2-stage Voting	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

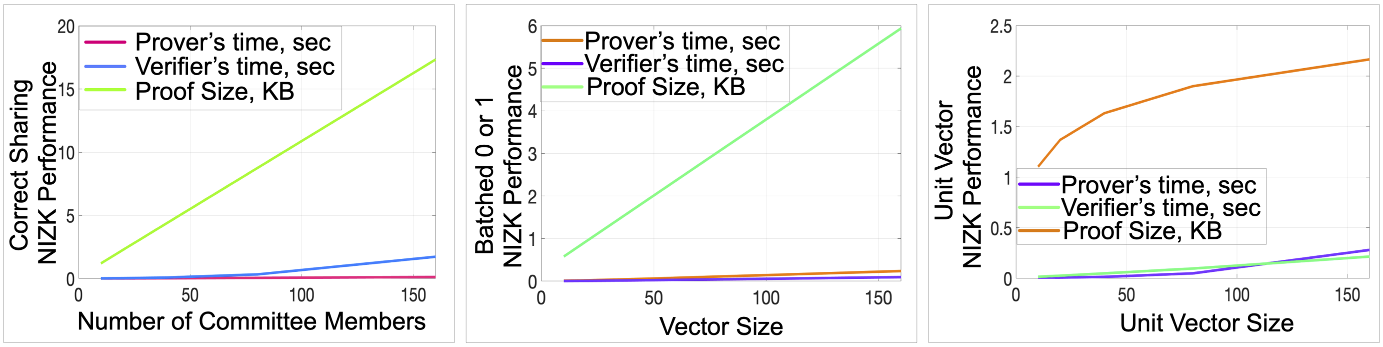


Fig. 15: The prover's running time, verifier's running time and the proof size for NIZK proofs.

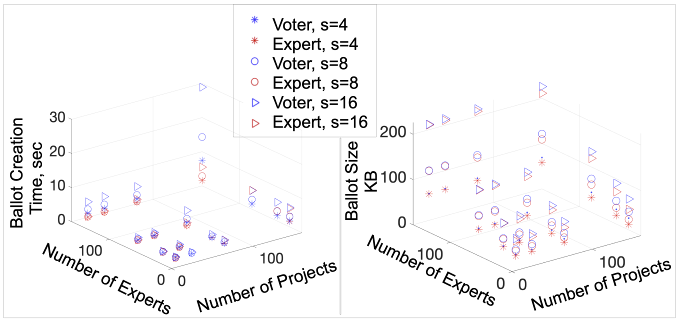


Fig. 16: Ballot size and creation time for each voter and expert in Preferential Voting Stage.

rect Sharing NIZK proof used in DBKG protocol, Bactched 0 or 1 NIZK proof and Unit Vector NIZK proof used in ballot casting in two stage voting. For Correct Decryption NIZK proof used in DBKG protocol and Handover protocol, the prover's running time is 0.889 ms, verifier's running time is 0.924 ms, and the proof size is $102B$.

We tested ballot creation time and size for a voter and an expert in Preferential Voting Stage in Fig. 16. In the Threshold Voting Stage, ballot is done once by a voter and takes less than 1 second for several hundreds of experts, so it has very small influence on the voting protocol performance. With 5000 voters and 50 experts, the overall communication is approximately 20 MB per project in the Threshold Voting Stage.

8 Conclusion

In this work, we initiated the study of privacy-preserving decision-making system over blockchain by leveraging the techniques of [1]. Firstly, we proposed a low complexity Distributed Batch Key Generation (DBKG) protocol to generate multiple keys simultaneously. Secondly, we designed a new Two-Stage Voting scheme to reduce voting effort. Thirdly, we extend the previous work to enable participatory budgeting. Fourthly, we support Evolving Committee to replace voting committee during the voting epoch. Moreover, we proved all our protocols secure under the UC framework, and analysed how the proposed system guarantees ballots privacy and end-to-end verifiability. Tests from our prototype implementation show that the proposed system is practical.

Acknowledgements

Bingsheng Zhang is supported by the National Key R&D Program of China (No. 2021YFB3101601) and the National Natural Science Foundation of China (Grant No. 62072401 and No. 62232002). This project is also supported by Input Output (iohk.io).

References

[1] B. Zhang, R. Oliynykov, and H. Balogun, "A treasury system for cryptocurrencies: Enabling better collaborative intelligence," in *The Network and Distributed System Security Symposium (NDSS '19)*, 2019.

[2] J. Adler, R. Berryhill, A. G. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018.* IEEE, 2018, pp. 1145–1152. [Online]. Available: https://doi.org/10.1109/Cybermatics_2018.2018.00207

[3] K. Nelaturu, J. Adler, M. Merlini, R. Berryhill, N. Veira, Z. Poulos, and A. G. Veneris, "On public crowdsource-based mechanisms for a decentralized blockchain oracle," *IEEE Trans. Engineering Management*, vol. 67, no. 4, pp. 1444–1458, 2020. [Online]. Available: https://doi.org/10.1109/TEM.2020.2993673

[4] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform," *arXiv preprint arXiv:1501.01042*, 2015.

[5] E. Duffield and D. Diaz, "Dash: A payments-focused cryptocurrency," *Whitepaper, https://github. com/dashpay/dash/wiki/Whitepaper*, 2018.

[6] Decred, "Decred governance," online; date last accessed: 2021-12-27. [Online]. Available: https://docs.decred.org/governance/overview

[7] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv preprint arXiv:1805.04548*, 2018.

[8] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *J. Cryptol.*, vol. 20, no. 1, pp. 51–83, 2007. [Online]. Available: https://doi.org/10.1007/s00145-006-0347-3

[9] A. Shah, *Participatory budgeting*. World Bank Publications, 2007.

[10] Y. Cabannes, "Participatory budgeting: a significant contribution to participatory democracy," *Environment and urbanization*, vol. 16, no. 1, pp. 27–46, 2004.

[11] Z. Beerliová-Trubíniová and M. Hirt, "Perfectly-secure MPC with linear communication complexity," in *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, ser. Lecture Notes in Computer Science, R. Canetti, Ed., vol. 4948. Springer, 2008, pp. 213–230. [Online]. Available: https://doi.org/10.1007/978-3-540-78524-8_13

[12] J. C. P. Carcía, A. Benslimane, and S. Boutalbi, "Blockchain-based system for e-voting using blind signature protocol," in *IEEE Global Communications Conference, GLOBECOM 2021, Madrid, Spain, December 7-11, 2021*. IEEE, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOBECOM46510.2021.9685189

[13] S. Kremer, M. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *European Symposium on Research in Computer Security*. Springer, 2010, pp. 389–404.

[14] R. H. Thaler, C. R. Sunstein, and J. P. Balz, "Choice architecture," in *The behavioral foundations of public policy*. Princeton University Press, 2013, pp. 428–439.

[15] F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin, "Can a public blockchain keep a secret?" in *Theory of Cryptography Conference*. Springer, 2020, pp. 260–290.

[16] H. U. Kumar and R. P. SG, "Algorand: A better distributed ledger," in *2019 1st International Conference on Advances in Information Technology (ICAIT)*. IEEE, 2019, pp. 496–499.

[17] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145. [Online]. Available: https://doi.org/10.1109/SFCS.2001.959888

[18] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[19] T. Pedersen and B. Petersen, "Explaining gradually increasing resource commitment to a foreign market," *International business review*, vol. 7, no. 5, pp. 483–501, 1998.

[20] I. Damgård. On Σ-Protocols, year = 2010, url = https://cs.au.dk/ ivan/Sigma.pdf, urldate = 2021-09-27.

[21] P. G. Neumann, "Security criteria for electronic voting," in *16th National Computer Security Conference*, vol. 29, 1993, pp. 478–481.

[22] S. Kremer, M. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, ser. Lecture Notes in Computer Science, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds., vol. 6345. Springer, 2010, pp. 389–404. [Online]. Available: https://doi.org/10.1007/978-3-642-15497-3_24

[23] Y. Yang, Z. Guan, Z. Wan, J. Weng, H. Pang, and R. H. Deng, "Priscore: Blockchain-based self-tallying election system supporting score voting," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4705–4720, 2021. [Online]. Available: https://doi.org/10.1109/TIFS.2021.3108494

[24] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, X. Du, and M. Guizani, "A blockchain-based self-tallying voting protocol in decentralized iot," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 1, pp. 119–130, 2022. [Online]. Available: https://doi.org/10.1109/TDSC.2020.2979856

[25] D. Kaidalov, A. Nastenko *et al.*, "Dash governance system: Analysis and suggestions for improvements." [Online]. Available: https://iohk.io/research/papers/#NSJ554WR

[26] J. Fraenkel and B. Grofman, "The borda count and its real-world alternatives: Comparing scoring rules in nauru and slovenia," *Australian Journal of Political Science*, vol. 49, no. 2, pp. 186–205, 2014.

[27] B. Reilly, "Social choice in the south seas: Electoral innovation and the borda count in the pacific island countries," *International Political Science Review*, vol. 23, no. 4, pp. 355–372, 2002.

[28] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. M. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 2019, pp. 185–200. [Online]. Available: https://doi.org/10.1109/EuroSP.2019.00023

[29] O. d. M. Ben Adida and O. Pereira, "Helios voting system," online; date last accessed: 2022-8-21. [Online]. Available: https://vote.heliosvoting.org

[30] B. Adida, O. De Marneffe, O. Pereira, J.-J. Quisquater *et al.*, "Electing a university president using open-audit voting: Analysis of real-world use of helios," *EVT/WOTE*, vol. 9, no. 10, 2009.

[31] V. Cortier, J. Dreier, P. Gaudry, and M. Turuani, "A simple alternative to benaloh challenge for the cast-as-intended property in helios/belenios," 2019.

[32] B. Yu, J. K. Liu, A. Sakzad, S. Nepal, R. Steinfeld, P. Rimba, and M. H. Au, "Platform-independent secure blockchain-based voting system," in *International Conference on Information Security*. Springer, 2018, pp. 369–386.

[33] R. Küsters, J. Liedtke, J. Müller, D. Rausch, and A. Vogt, "Ordinos: A verifiable tally-hiding e-voting system," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 216–235.

[34] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *International Workshop on Public Key Cryptography*. Springer, 2002, pp. 141–158.

[35] F. Hao, P. Y. Ryan, and P. Zieliński, "Anonymous voting by two-round public discussion," *IET Information Security*, vol. 4, no. 2, pp. 62–67, 2010.

[36] Z. Zhao and T.-H. H. Chan, "How to vote privately using bitcoin," in *International Conference on Information and Communications Security*. Springer, 2015, pp. 82–96.

[37] S. Bistarelli, M. Mantilacci, P. Santancini, and F. Santini, "An end-to-end voting-system based on bitcoin," in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 1836–1841.

[38] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *International conference on financial cryptography and data security*. Springer, 2017, pp. 357–375.

[39] W.-J. Lai, Y.-C. Hsieh, C.-W. Hsueh, and J.-L. Wu, "Date: A decentralized, anonymous, and transparent e-voting system," in *2018 1st IEEE international conference on hot information-centric networking (HotICN)*. IEEE, 2018, pp. 24–29.

[40] K. M. R. Alam, S. Tamura, S. S. Rahman, and Y. Morimoto, "An electronic voting scheme based on revised-svrm and confirmation numbers," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 400–410, 2019.

[41] R. Krishnamurthy, G. Rathee, and N. Jaglan, "An enhanced security mechanism through blockchain for e-polling/counting process using iot devices," *Wireless Networks*, vol. 26, no. 4, pp. 2391–2402, 2020.

[42] H. Li, Y. Li, Y. Yu, B. Wang, and K. Chen, "A blockchain-based traceable self-tallying e-voting protocol in ai era," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1019–1032, 2020.

[43] C. Angsuchotmetee, P. Setthawong, and S. Udomviriyalanon, "Blockvote: An architecture of a blockchain-based electronic voting system," in *2019 23rd International Computer Science and Engineering Conference (ICSEC)*. IEEE, 2019, pp. 110–116.

[44] S. T. Alvi, M. N. Uddin, and L. Islam, "Digital voting: A blockchain-based e-voting system using biohash and smart contract," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020, pp. 228–233.

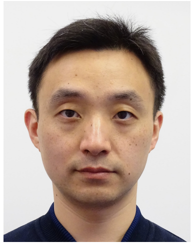
[45] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, X. Du, and M. Guizani, "A blockchain-based self-tallying voting protocol in decentralized iot," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[46] X. Zhang, B. Zhang, A. Kiayias, T. Zacharias, and K. Ren, "An efficient e2e crowd verifiable e-voting system," *IEEE Transactions on Dependable and Secure Computing*, 2021.

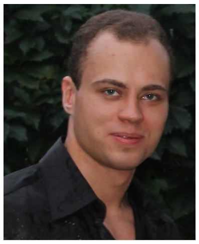
[47] J. C. P. Carcia, A. Benslimane, and S. Boutalbi, “Blockchain-based system for e-voting using blind signature protocol,” in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 01–06.

[48] Y. Yang, Z. Guan, Z. Wan, J. Weng, H. H. Pang, and R. H. Deng, “Priscore: blockchain-based self-tallying election system supporting score voting,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4705–4720, 2021.

JIAJIE ZHANG received the B.Sc. (Hons) in Computer Science from Jinan University, China. She is currently pursuing the Ph.D. degree at Lancaster University, U.K. Her research interests include Applied Cryptography, Privacy Preserving Computation, and Blockchain.



Bingsheng Zhang received the B.E. degree from the Zhejiang University of Technology, Hangzhou, China, in 2007, the M.S. degree from University College London, UK, in 2008, and the Ph.D. degree from the University of Tartu, Estonia, in 2011. He is currently a Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. Before that, he was program director of Lancaster University’s master’s degree in cybersecurity and leader of the university’s security research group. He specialises in cryptography, verifiable electronic voting (e-voting), and zero-knowledge proofs. In recent years, his research interests include secure computing, collaborative decision-making, and blockchain security.



Andrii Nastenko received a Ph.D. degree in symmetric cryptography at Kharkiv National University of Radioelectronics, Ukraine. Currently he is IOG Junior Research Fellow. His research interests are: modern cryptographic algorithms, symmetric block and stream ciphers and blockchain technologies.

Hamed Balogun lectures at the University of Central Lancashire, United Kingdom. He obtained his PhD in computer science from Lancaster University, UK, where his research focused on secure blockchain development. His research efforts and interests span privacy (in online social networks), security, and applied cryptography.

Roman Oliynykov received a Dr.Habil degree in symmetric cryptology at Kharkiv National University of Radio Electronics, Ukraine. Currently he is a professor at Information Systems and Technologies Security Department at V.N. Karazin Kharkiv National University, Ukraine; visiting professor at Information Technologies Security Department at Kharkiv National University of Radio Electronics, Ukraine; IOG Research Fellow. He also lectured as an invited professor in South Korea and Norway and was involved in training of Ukrainian cyberpolice officers. His professional experience includes the development of Ukrainian cryptographic standards: the Kalyna block cipher and the Kupyna hash function. His research interests are: blockchain technologies, design and analysis of cryptographic primitives, software security, computer networks security. Biography text here.