Enabling Large-scale Blockchain Decision-making

Anonymous Author(s)

ABSTRACT

The blockchain community forms a decentralized collaborative society.

is a decentralized system, and

1. more token does not make a person to make better decision.

2. register tokens, NDSS2019 open to the public, but the adversary might attack the system by checking how many tokens have been registered in the voting period.

3. reputation should consider proposer.

A treasury system is a community controlled and decentralized collaborative decision-making mechanism for sustainable funding of the blockchain development and maintenance. During each treasury period, project proposals are submitted, discussed, and voted for; top-ranked projects are funded from the treasury. The Dash governance system is a real-world example of such kind of systems. In this work, we, for the first time, provide a rigorous study of the treasury system. We modelled, designed, and implemented a provably secure treasury system that is compatible with most existing blockchain infrastructures, such as Bitcoin, Ethereum, etc. More specifically, the proposed treasury system supports liquid democracy/delegative voting for better collaborative intelligence. Namely, the stake holders can either vote directly on the proposed projects or delegate their votes to experts. Its core component is a distributed universally composable secure end-to-end verifiable voting protocol. The integrity of the treasury voting decisions is guaranteed even when all the voting committee members are corrupted. To further improve efficiency, we proposed the world's first honest verifier zero-knowledge proof for unit vector encryption with logarithmic size communication. This partial result may be of independent interest to other cryptographic protocols. A pilot system is implemented in Scala over the Scorex 2.0 framework, and its benchmark results indicate that the proposed system can support tens of thousands of treasury participants with high efficiency.

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper;

KEYWORDS

template; formatting; pickling

1 INTRODUCTION

The blockchain technology enables an opportunity for an opening, a dispersion of power and information, and profound possibilities for large-scale collaboration in the modern digital society. By distributing power and value across the entire network, blockchain makes the exchange of information and value more efficient and equitable. As a decentralized system, ideally, the development and governance of a blockchain platform should resist control by a single or outside power. A democratic on-chain decision-making system is essential for sustainable and healthy blockchain platforms.

Although several simple blockchain decision-making systems have been used in practice, such as the Dash governance system and DaoStack [**?**], those systems fail to protect voter privacy. The only known provably secure blockchain decision making system in the literature is proposed by Zhang *et al.* [**?**]. The system supports so-called *liquid democracy* with privacy assurance, and its security is analyzed under the Universally Composable (UC) framework [**?**]. Roughly speaking, in the system, the voters can either make a decision by themselves or delegate their votes to an expert. However, the same as a traditional voting scheme, each execution can only handle one project decision making, i.e., we need to run multiple voting protocol instances simultaneously if several projects need to be decided at the same time. Therefore, the overall communication and computation scales up with respect to the number of projects.

For instance, in Zhang *et al.*'s scheme [**?**], a voter's choice vector is element-wisely encrypted by

In the blockchain context, it is often needed to make decisions for multiple projects, such as prediction market, decentralized oracle, funding allocation.

In this work, we propose the world's first voting scheme, that can be used to vote multiple projects at the same time.

There are a number of drawback

an on-chain decision-making system

it is important to have a scalable and resilient blockchain decision-making system that can support the processing of large number of crowd decisions effectively. Here, scalability encompasses two aspects: (a) voter number and (b) project s number

has two folder

A long-term decision making system needs a reputation management scheme

The kernel of a reputation management scheme is the predictive power of reputation scores, which means that past behaviours of a user can be regarded as a meaningful indicative factor of his future behaviours, capability, and reliability [**?**]. A user enjoying a highly valued reputation score would imply that this user has conducted right decisions (which is the same as the majority) in the system with a high enough rhythm in the past. Therefore, he can be trusted to perform honestly in future projects and to be qualified as an expert as well. To accumulate reputation scores, users are motivated to participate in the system and to utilise their expertise.

In this paper, we propose a new reputation management scheme for decentralized decision-making systems to optimize the benefit of users' expert knowledge. In contrast to our former setting [**?**], based on this reputation management scheme, users don't necessarily need to have huge amount of stakes to qualify as experts in the system. Unlike the subjective reputation in the sense of social networks, here in this work, we compute objective criteria of two-dimensional reputation to track every user's contribution in the system and to help estimate the quantitative expertise. Despite all interacting contributory factors for quantifying an individual's expertise, we tie an expert's reputation to his *Regularity of Work*

and *Quality of Total Productive Contributions*[**?**]. Specifically, it's computed based on both the frequency of the user's engagement quantified by the voting power ratio he deposits and the outcome of projects that he joins, over the entire period of time since the whole system has been triggered.

Considering that reputation is based on the user's past behaviours in the system, thus, when an attacker joins the system at time t, even if he has a huge amount of stakes, or instantaneous computational power, he would have no significant reputation (other than the initial reputation value set by the scheme) at updating epoch k. Even shortly after, as he did not contribute to the system before k. When a user deviates from the system specifications (his ballot/decesion failed to agree with the majority), we lower his reputation in consequence of this negative contribution. This prevents a powerful malicious user from attacking the system repeatedly without significant consequences.

1.1 Our Contributions

1.2 Organisation

2 PRELIMINARIES

2.1 Notations

Throughout this document, we will use the following notations. Denote a value U indexed by a label x as $U^{(x)}$, while U^x means the value of U power of x. In addition, we abbreviate $[a, b]$ as the set $\{a, a+1, \ldots, b\}$, let $[b]$ denote $[1, b]$. We use $\langle a_1, \ldots, a_n \rangle$ to represent a vector with fixed order. (a, b) means continuous rational numbers between a and b, such as $(1, 3) = \{1, 2, 3\}$, while $(a, \ldots, b)$ represent either continuous or discontinuous numbers between a and b, for example $(1, \ldots, 3)$ can be discontinuous numbers $\{1, 3\}$ or continuous numbers $\{1, 2, 3\}$.

2.2 Blockchain Abstraction

Without loss of generality, we abstract the underlying blockchain platform encompasses the following concepts.

∘ *Mainchain and sidechain.* Mainchain is a primary blockchain, where all entities can send requests and browse the ledger (*e.g.* Bitcoin, Ethereum). Sidechain is a separated system of pegged ledgers attached to mainchain, connected via a cross-chain transfer protocol. Sidechain offers ability to offload tasks from mainchain to sidechain, which brings interoperability, scalability, and upgradability for blockchain system.

∘ *Merkle Tree.* Merkle tree is a binary tree structure to check the integrity of data in a general blockchain environment. The input of Merkel Tree is list of hashed data records, every two records are grouped together and their hash value will be the value of an internal node. The Merkel root is the root of binary hash tree created out of all the transactions in a block.

∘ *Coin.* We assume the underlying blockchain platform has the notion of *Coins* or its equivalent. Each coin can be spent only once, and all the value of coin must be consumed. As depicted in Fig. **??**, each coin consists of the following 4 attributes:

- **Coin ID**: It is an implicit attribute, and every coin has a unique ID that can be used to identify the coin.
- **Value**: It contains the value of the coin.
- **Cond**: It contains the conditions under which the coin can be spent.
- **Payload**: It is used to store any non-transactional data.

∘ *Address.* We also generalize the concept of the *address*. Conventionally, an address is merely a public key, pk, or hash of a public key, hash(pk). To create coins associated with the address, the spending condition of the coin should be defined as a valid signature under the corresponding public key pk of the address. In this work, we define an address as a generic representation of some spending condition. Using the recipient's address, a sender is able to create a new coin whose spending condition is the one that the recipient intended; therefore, the recipient may spend the coin later.

∘ *Transaction.* Each transaction takes one or more (unspent) coins, denoted as $\{\mathsf{In}_i\}_{i\in[n]}$, as input, and it outputs one or more (new) coins, denoted as $\{\mathsf{Out}_j\}_{j\in[m]}$. Except special transactions, the following condition holds:

$$\sum_{i=1}^{n} \mathsf{In}_i.\mathsf{Value} \geq \sum_{j=1}^{m} \mathsf{Out}_j.\mathsf{Value}$$

and the difference is interpreted as transaction fee. As shown in Fig. **??**, the transaction has a *Verification data* field that contains the necessary verification data to satisfy all the spending conditions of the input coins $\{\mathsf{In}_i\}_{i\in[n]}$. In addition, each transaction also has a Payload field that can be used to store any non-transactional data. We denote a transaction as $\mathsf{Tx}(A; B; C)$, where A is the set of input coins, B is the set of output coins, and C is the Payload field. Note that the verification data is not explicitly described for simplicity.

2.3 Universal Composability

Following Canetti's framework [**?**], a protocol is represented as a set of interactive Turing machines (ITMs), each of which represents the program to be run by a participant. Protocols that securely carry out a given task are defined in three steps, as follows. First, the process of executing a protocol in an adversarial environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. The parties have access to an "ideal functionality" which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. Let $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathbb{Z}}$ denote the output of the environment $\mathbb{Z}$ when interacting with parties running the protocol Π and real-world adversary $\mathcal{A}$. Let $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathbb{Z}}$ denote output of $\mathbb{Z}$ when running protocol ϕ interacting with the ideal functionality $\mathcal{F}$ and the ideal adversary $\mathcal{S}$.

Definition 2.1. We say that a protocol Π UC-realizes $\mathcal{F}$ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for any environment $\mathbb{Z}$ that obeys the rules of interaction for UC security we have $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathbb{Z}} \approx \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathbb{Z}}$.

Definition 2.2 (DDH). We say that the decisional Diffie-Hellman (DDH) assumption is hard with respective to $\mathbb{G} = \{\mathbb{G}_n\}$ if for any PPT algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ in n such that $\|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] \leq \mathsf{negl}(n)$, where q is the order of $\mathbb{G}$ and the probabilities are taken over the choice of g and $x, y, z \in \mathbb{Z}_q$.

Definition 2.3 (DCR). We say that the decisional composite residuosity (DCR) assumption is hard if for any PPT algorithm $\mathcal{A}$ there exists a negligible function negl in n such that $\|\Pr[\mathcal{A}(N,z) = 1 \mid z = y^N \mod N^2] - \Pr[\mathcal{A}(N,z) = 1 \mid z = (N+1)^r \cdot y^N \mod N^2]\| \leq \text{negl}(n)$, where N is a random n-bit RSA composite, r is chosen at random in $\mathbb{Z}_N$, and the probability are taken over the choices N, y and r.

2.4 Homomorphic PKE

A public-key encryption scheme $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is homomorphic if for all security parameter $\lambda \in \mathbb{N}$ and all $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Gen}(1^\lambda)$, it is possible to define a message space $\mathcal{M}$ and a ciphertext space C such that for any $m_1, m_2 \in \mathcal{M}$ and $c_1, c_2 \in C$ with $m_1 = \mathsf{Dec}_{\mathsf{sk}}(c_1)$ and $m_2 = \mathsf{Dec}_{\mathsf{sk}}(c_2)$, it holds that

$$\{\mathsf{pk}, c_1, c_1 \odot c_2\} \equiv \{\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_1), \mathsf{Enc}_{\mathsf{pk}}(m_1 \circ m_2)\} \ .$$

Here $\odot$ and $\circ$ are the group operations over C and $\mathcal{M}$, respectively. When $\circ$ is addition, the public-key encryption scheme is called additively homomorphic; when $\circ$ is multiplication, the public-key encryption scheme is called multiplicative homomorphic. If a public-key encryption scheme can perform both addition and (several) multiplication homomorphicly, it is called (somewhat) fully homomorphic encryption.

3 THE MODEL

3.1 Entities

Let $p^{(i)}, v^{(i)}, e^{(i)}, s^{(i)}, m^{(i)}, o$ be integers in $poly(\lambda)$. In each treasury period i, the stake holder might play one or more the following entity roles. To facilitate a better understanding, the dependence of entities is shown in Fig.??. Since we offload mainchain tasks to sidechain, one subtle implication is that our users have accounts on both mainchain and sidechain. More specifically, they receive funding and stake reward on mainchian, but accomplish a series of treasury tasks on sidechain.

- The proposal owners $\mathcal{O}^{(i)} := \{\mathsf{O}_1^{(i)}, \ldots, \mathsf{O}_{p^{(i)}}^{(i)}\}$ are registered users, who have proposal(s) in order to get support. Proposal owners submit their proposals on sidechain, if funded, they will receive the fund on mainchain.
- The voters $\mathcal{VT}^{(i)} := \{\mathsf{VT}_1^{(i)}, \ldots, \mathsf{VT}_{v^{(i)}}^{(i)}\}$ is a subset of stake holders who lock amount of stakes on mainchain, then register as voters and participant voting on sidechain. The voting power of a voter $\mathsf{VT}_j^{(i)}$ is proportional to his deposited stakes in current treasury period, denoted as $\mathsf{VPR}^{(i)}(\mathsf{VT}_j^{(i)})$, where $j \in [v^{(i)}]$. After voting, voters will get their rewards on mainchain.
- The experts $\mathcal{E}_{\mathsf{Fld}_m}^{(i)} := \{\mathsf{E}_{\{\mathsf{Fld}_m,1\}}^{(i)}, \ldots, \mathsf{E}_{\{\mathsf{Fld}_m,e^{(i)}\}}^{(i)}\}$ are a subset of voters $\mathcal{VT}^{(i)}$, in a specific field Fld_m, where m is total number of field tags in treasury system. Experts are users whose reputation scores are higher than a threshold $\mathsf{TH}_{\mathsf{Fld}_m}$, or highly recognised experts from outside. Experts can *only* get voting power from delegation on sidechain, which means that they can only vote if some voters delegate their voting power to them.

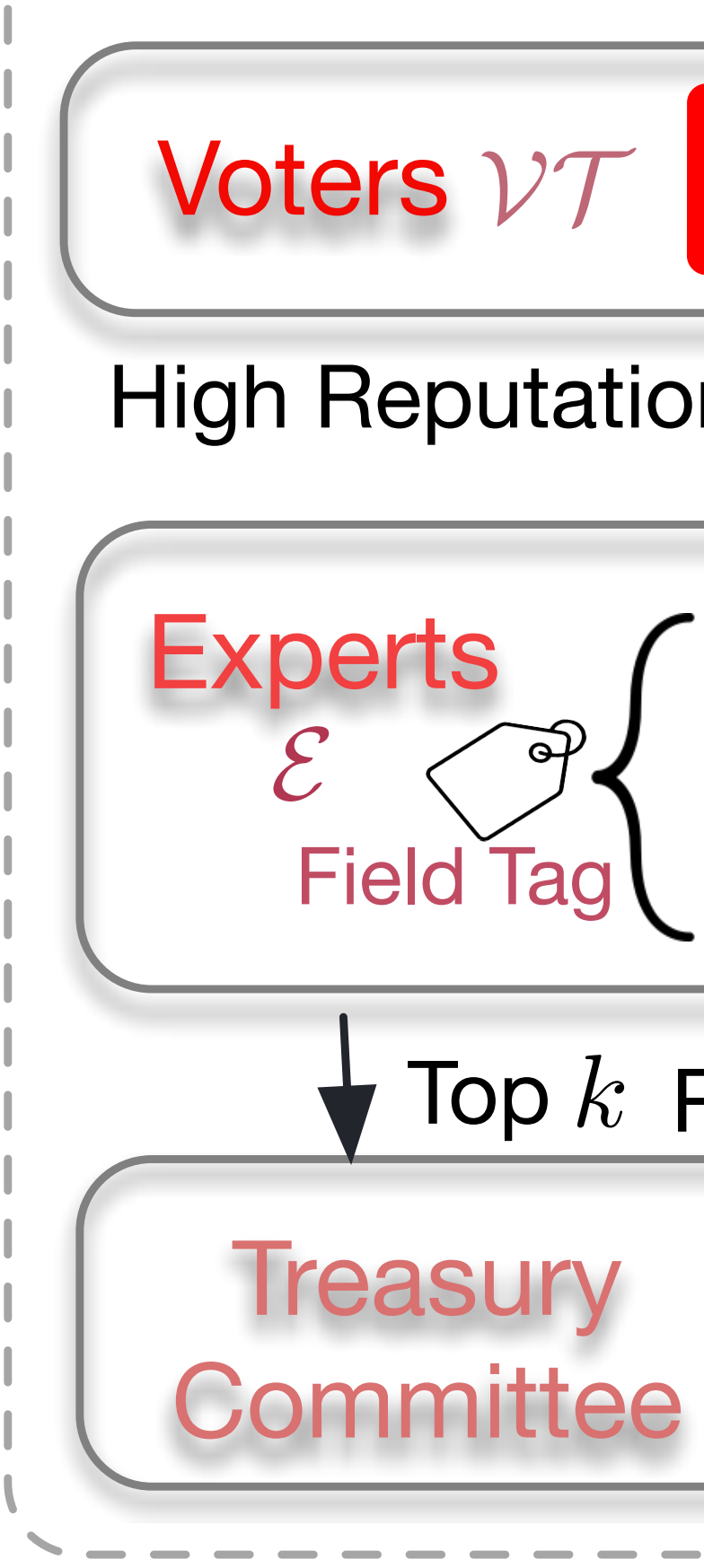

Figure 1: Entities involved in our system

3.2 System Overview

The whole system consists of iterative treasury periods. For each period, it has three main stages, pre-voting stage, voting stage, and post-voting stage, as shown in Fig.??.

Pre-voting stage is foundation of our system, which is composed of a two-stage project proposing, and related participants registration. In order to provide fairness and avoid plagiarism cheating (particularly for late submissions), we provide a two-stage project proposing model. In the first stage of project proposing, proposal owner will submit a commitment of their proposals (including their address on mainchain) assigned with field tags asking for fund on sidechain. In the second stage of project proposing, after every proposal owner submits their proposals within a predefined time frame (fixed size of blocks), proposal owners will de-commite the proposals on sidechain. Users/stake holders who are interested in voting can lock some amount of stakes on mainchain to register as voters, their voting power will be proportional to the stakes they locked in this stage. A group of voters with fixed size (*e.g.* 100) will be randomly selected as Organising Committee. Organising Committee will create a sidechain, and copy paste essential information about proposals, and voters to sidechain. public proposal, offchain

People who are well-known, highly regarded and reputable can be invited to register as experts on mainchain awarded an initial reputation score. We name this type of experts outside experts, they will register an account on mainchain for receiving rewards, and choose one field tag to show their specific expertise. Note that one expert account can only be assigned with one field tag, in other words, if a user is a cross-disciplinary/domain expert, he needs to register multiple accounts. For voters whose reputation scores (Cf. Sec **??**) exceed the threshold in a certain field, can also register as experts in pre-voting stage. We name them inside experts, to distinguish them from reputable outside experts. Similarly, if they are qualified to be experts in multiple fields, they can only choose one field from those when registering as experts. The voting power of experts (including outside and inside experts) only comes from the voting power delegated to him (we name this kind of voting power "Delegation Power", which will be explained later). Based on the proposal fields in the two-stage proposing phase, we select top k experts who own the highest reputation scores from the experts set in a certain field, and these experts will automatically sit on treasury committee. The scope of fields is the same as the fields of proposals in current treasury epoch.

From Voting stage to Post-voting stage, there is a distribute key generation protocol (Cf. Sec **??**) running concurrently. At the beginning, first generation of selection committee will be generated by cryptographic sortition, the portability of be chosen is proportional to their total locked stakes contribution on mainchain in the past, the size of selection committee is determined by running cryptographic sortition protocol each time. Then each selection committee member will choose a registered voter who are willing to chosen as a member of maintenance committee, based on their locked stakes on mainchain, as the first generation of maintenance committee. We name this phase Designating, which is executed receptively on sidechain until the last round. The first generation of maintenance committee will jointly setup key pairs, public key will be released on sidechain, secret keys will be handover to next maintaining committee which is nominated by next selection committee, until the last round.

Voting stage is composed of a two stage voting scheme: Nominating Stage and Endorsing Stage. In Nominating stage, treasury committee will nominate a number of proposals to generate a short-list for consideration in next voting stage on sidechain. They can undertake discussion either online or offline. The size of this short-list is determined by the total funds in current treasury period and funds asked by each proposal. To ensure integrity, authentication, and system reliability, this short-list is only valid if it's signed by a number of committee members whose total reputation scores are more than 50% of all committee members' reputation scores. In order to improve project owners' experience, treasury committee can provide feedback to project owners about their proposals.

In the endorsing stage, a subset of voters who are willing to vote/endorse can join and vote based on the short-list given by treasury committee. Then, they encrypt their ballots with the public key generated by the first generation of maintenance committee. After a fixed number of blocks, voting stage will be terminated. another way to prevent DoS in the sidechain, besides transaction fees, is just a limit of ballot submission attempts. I.e., each registered voter may submit his ballot, then change his opinion and resubmit ballot again, but up to 5 ballot submissions per registered voter per one treasury epoch. From Roman and Dmytro

Post-voting stage contains tally calculation stage and execution stage. The last generation of maintenance committee will decrypt tally together with their secret key shares, after that, organising committee will verify tally result. In the execution stage, the winning proposal will be funded, and the experts, voters will be rewarded accordingly on mainchain. Based on the final list of funded proposals, reputation scores of treasury committee, experts, voters will be updated accordingly (Cf. Sec. **??**). Lastly, organising committee will write essential information (tally results, reputation scores) back to mainchain and then destroy the sidechain. no one knows which project is in the finalist, they only receive the fund. If the funding is not used up, the remaining fund will be transferred to next treasury period.

3.3 Transaction Structure

Project proposal.

Project owner should register accounts on both mainchian and sidechain. In the first stage of project proposing, the project owners $\{O_1, \ldots, O_{p^{(i)}}\}$ will first commit their project proposals to sidechain within a pre-defined time frame. In the second stage, project owners need to de-commit these proposals. To commit a project, the project owner needs to submit a special *project proposal transaction* in form of

$$\mathsf{Tx}\Big(\{\mathsf{In}^{(i)}\}_{i=1}^{n}; \mathsf{TCoin}; \{\textsc{Project}, \mathsf{Fld}, \mathsf{TID}, \mathsf{PCOM}, \mathsf{M\_Addr}\}\Big) \ ,$$

where $\{\mathsf{In}^{(i)}\}_{i=1}^{n}$ are the input coins, and TCoin is a special output coin whose spending condition is defined as, the coin can only be spent according to the corresponding treasury decision (cf. Subsection "supplying the treasury", below). Moreover, the coin value $\mathsf{TCoin.Value} \geq \alpha_{\min}$, where $\alpha_{\min}$ is the minimum required fee for a project proposal to prevent *denial-of-service* attacks. In the Payload field, Project is a tag that indicates it is a special project proposal transaction; Fld is the field tag assigned with the proposal

Figure 2: System Overview

(the whole tag set is decided before); TID is the treasury ID that is used to uniquely identify a treasury period; PCOM is the committee of project proposal, and M-Addr is the return address for the project owner to receive money on mainchain if the project is funded.

Voter registration. In order to register to be a voter, a stake holder (or a set of stake holders) need(s) to first lock stakes on mainchain and submit a special *voter registration transaction* on sidechain in form of

$$\mathsf{Tx}\Big(\{\mathsf{In}^{(i)}\}_{i=1}^{n}; \mathsf{TCoin}; \big\{\textsc{Voter-Reg}, \mathsf{TID}, \{\mathsf{Proof}(\mathsf{S}^{(i)})\}_{i=1}^{\ell}, \text{S-Cond}, \mathsf{vk}, \mathsf{VolM}, \text{M-Addr}\big\}\Big) ,$$

where $\{\mathsf{In}^{(i)}\}_{i=1}^{n}$ are the input coins, and TCoin is a special output coin whose spending condition is defined in Subsection "supplying the treasury", below. In the Payload field, Voter-Reg is a tag that indicates it is a special voter registration transaction; TID is the treasury ID that be used to uniquely identify a treasury period; $\{\mathsf{Proof}(\mathsf{S}^{(i)})\}_{i=1}^{\ell}$ are the proof of *freezed* unspent coins on mainchain that will be used to claim stake value, S-Cond is the required data that satisfies all the stake attributes of $\{\mathsf{S}^{(i)}\}_{i=1}^{\ell}$, vk is a freshly generated signature key. The voter's ID is defined as the hash of vk, denoted as $\mathsf{VT}^{(i)} := \mathsf{hash}(\mathsf{vk})$. $\mathsf{VolM} \in [0, 1]$ show if voters wants to volunteer as maintenance committee (1 for volunteering; otherwise 0). M-Addr is the return address for the voter to receive treasury reward on mainchain.

Expert registration. As mentioned before, experts is splitted into outside experts and inside experts. For outside experts, they are invited to register in our system. They will be assigned an initial reputation value, and choose their field tag based on their expertise. Inside experts are voters who are qualified to register only if they have a sufficient amount of reputation scores in certain field(s), and they can only choose a field tag from the field(s) as their expertise.

To register as an experts, outside experts need to register on mainchain and submit a special *expert registration transaction* to sidechain in the form of:

$$\mathsf{Tx}\Big(\{\mathsf{In}^{(i)}\}_{i=1}^{n}; \mathsf{TCoin}; \{\textsc{Expert-Reg}, \mathsf{Fld}, \mathsf{TID}, \mathsf{vk}, \text{M-Addr}\}\Big) ,$$

where $\{\mathsf{In}^{(i)}\}_{i=1}^{n}$ are the input coins, and TCoin is a special output coin whose spending condition is defined in section "supplying the treasury". In the Payload field, Expert-Reg is a tag that indicates it is a special expert registration transaction; TID is the treasury ID that be used to uniquely identify a treasury period; vk is a freshly generated signature key; and M-Addr is the return address for the expert to receive treasury reward on mainchain. The expert's ID is defined as the hash of vk, denoted as $\mathsf{E}_j := \mathsf{hash}(\mathsf{vk})$.

For inside experts registration, the only difference from outside expert registration is that inside experts need to prove their reputation scores for qualification. They need to submit a transaction in the form of:

$\mathsf{Tx}\Big(\{\mathsf{In}^{(i)}\}_{i=1}^{n}; \mathsf{TCoin}; \{\textsc{Expert-Reg}, \mathsf{Fld}, \mathsf{TID}, \mathsf{Proof(REP)}, \mathsf{vk}, \mathsf{M\text{-}Addr}\}\Big)$,

where Proof(REP) is the proof of reputation scores on sidechain.

3.4 selection committee selection

selection committee is selected by cryptography sortition. As mentioned before, selection committee's main responsibility is to nominate and hide maintenance committee from adversary. Thus, how to sort out appropriate selection committee is crucial for the whole system. Three requirements should meet when choosing selection committee, 1) Can defend against Sybil attacks by minimising the probability of selecting malicious nodes as committee members; 2) No centralisation should be introduced; 3) Low complexity with minimum message exchanges among nodes. To meet the aforementioned requirements, cryptographic sortition for choosing a set of voters as selection committee should according to voters' total amount of deposits in history.

In treasury period d, Let $\mathsf{st}^{(i)} = \sum_{k=1}^{d-1}\sum_{j=1}^{\ell} S_{\{k,j\}}$.Value for all the locked stake coins S_j in past $d-1$ periods claimed in the payload of the voter registration transaction of $\mathsf{VT}^{(i)}$. In other words, $\mathsf{st}^{(i)}$ is the total stake amount claimed by $\mathsf{VT}^{(i)}$ in history. $\mathsf{TS} := \sum_{k=1}^{d-1}\sum_{i=1}^{v_k}\sum_{j=1}^{\ell_{\{k,i\}}} S_{\{k,i,j\}}$.Value is the total stakes locked during the last $d-1$ periods, where v_k is the number of voters in the k period, $\ell_{\{k,i\}}$ is the total stake coins locked by $\mathsf{VT}^{(i)}$ in the k period. The probability that $\mathsf{VT}^{(i)}$ is selected is proportional to $\mathsf{st}^{(i)}$, which is called sortition weight. Cryptographic sortition based on sortition weight guarantee that the probability of being selection proportional to the past deposit stakes/contributions. So that malicious nodes who only joins in current period without having reasonable or sustainable portion contribution in the past, will have very low probability of being picked out as selection committee.

We employ verifiable random functions (VRFs) to implement our cryptographic sortition. Given any input string x, $VRF_{sk}(x)$ returns two values: a *hash-len* bit-long hash value uniquely determined by sk and x (indistinguishable from random value without knowledge of sk), and a proof π which enables public verification of hash value based on pk.

We give our cryptographic sortition functionality in Fig. **??**. Ideally, we would like to model cryptographic sortition as a âĂIJperfectâĂİ lottery functionality that select nodes based on their sortition weight. Considering that the adversary has some influence over the public input (even nodes' key pairs), and making it possible for it to try many inputs until it finds one that it likes, we model the adversary has the power to restart the whole sortition scheme and even change keys of voters. The second property is claimed by VRF.

Our cryptographic sortition protocol is demonstrated in Fig. **??**. We first invoke a VRF by giving current period id k, and random seed associated with k defined by selection committee in $k-1$ period. VRF returns a *hash-len* bit-long hash value $hash^{(i)}$ and its proof $\pi^{(i)}$. We consider the total sortition weight of selection committee as $\bar{st} := \sum_{d}^{(i)} \sigma^{(i)}\mathsf{st}^{(i)} \leq \mathsf{TS}$, as an admission control, where $\sigma^{(i)}$ is a threshold, which indicates the probability of selecting $\mathsf{VT}^{(i)}$ into the selection committee. In addition, $\sigma^{(i)}$ should be proportional to $\mathsf{st}^{(i)}$, namely $\sigma^{(i)}/\mathsf{st}^{(i)} = \sigma_j/\mathsf{st}_j$, where $i, j \in [v_d]$.

The ideal functionality $\mathcal{F}_{\textsc{Sortion}}$

The functionality $\mathcal{F}_{\textsc{Sortion}}$ interacts with a set of parties $\mathcal{P} := \{P_1, \ldots, P_n\}$ and adversary $\mathcal{S}$. It is parameterized with a variable $p \in [0,1]$, and tables Pub and T. Let $\mathcal{P}_c$ be the set of corrupted parties.
Initially, set $\mathsf{Pub} := \emptyset$, $T := \emptyset$, and $P_c := \emptyset$.

Initialization:

- Upon receiving (Init, sid, ssid) from the adversary $\mathcal{A}$, for each party P_i, pick a random bit b_{P_i} with $\Pr[b_{P_i}^{\mathsf{ssid}} = 1] = p$ and set $T[\mathsf{ssid}] := \{\langle P_i, b_{P_i}^{\mathsf{ssid}}\rangle\}_{P_i \in \mathcal{P}_c}$, and then it sends $T[\mathsf{ssid}]$ to the adversary $\mathcal{S}$.
- Upon receiving (Refresh, sid) from $\mathcal{S}$, last step will be repeated.

Query:

- Upon receiving (Query, sid, ssid) from $P_i \in \mathcal{P}$, if $T[\mathsf{ssid}]$ is initialized, send (Query, sid, ssid, $b_{P_i}^{\mathsf{ssid}}$) to the requestor.

Reveal:

- Upon receiving (Reveal, sid, ssid) from $P_i \in \mathcal{P}$, set $\mathsf{Pub}[\mathsf{ssid}] := \mathsf{Pub}[\mathsf{ssid}] \cup \{\langle P_i, b_{P_i}^{\mathsf{ssid}}\rangle\}$.
- Upon receiving (State, sid, ssid) from any party $P_i \in \mathcal{P}$, send (State, sid, ssid, $\mathsf{Pub}[\mathsf{ssid}]$) to the requestor.

Corruption handling:

- Upon receiving (Corrupt, sid, P_i) from $\mathcal{S}$, set $\mathcal{P}_c := \mathcal{P}_c \cup \{P_i\}$.
- Upon receiving (Uncorrupt, sid, P_i) from $\mathcal{S}$, set $\mathcal{P}_c := \mathcal{P}_c \setminus \{P_i\}$.

Figure 3: The multi-session cryptographic sortition functionality $\mathcal{F}_{\textsc{Sortion}}$

Combing these two setting together, we can get $\sigma^{(i)} = \frac{\mathsf{st}^{(i)}\bar{st}}{\sum_{j=1}^{v_d}(\mathsf{st}_j)^2}$. Then we check whether $hash^{(i)}/2^{hashlen}$ is no greater than the threshold $\sigma^{(i)}$.

The cryptographic sortition $\Pi_{Sortition}^{sk^{(i)},k,seed_k}$

- $(hash^{(i)}, \pi^{(i)}) = VRF(k, seed_k)$;
- $\sigma^{(i)} = w^{(i)}\bar{w}/\sum_{j,j\in N} w_j^2$;
- if $hash^{(i)}/2^{hashlen} \leq \sigma^{(i)}$,
  then
  return $(True, hash^{(i)}, \pi^{(i)})$
  else
  return $(False, hash^{(i)}, \pi^{(i)})$

Figure 4: The cryptographic sortition protocol $\Pi_{\textsc{Treasury}}^{sk^{(i)},k,seed_k}$

At the beginning of the selection committee selection step, the selection committee of the previous treasury period jointly reveal the committed seed, seed. Let $\mathsf{st}^{(i)} = \sum_{j=1}^{\ell} S_j$.Value for all the stake coins S_j claimed in the payload of the voter registration transaction of $\mathsf{vk}^{(i)}$, i.e. $\mathsf{st}^{(i)}$ is the total stake amount claimed by $\mathsf{vk}^{(i)}$. Once seed is announced, any registered voter, who have an address $\mathsf{vk}^{(i)}$ with claimed stake $\mathsf{st}^{(i)}$, can volunteer to participate in the voting committee if the following inequality holds:

$$\mathsf{hash}\Big(\mathsf{vk}^{(i)}, \mathsf{sign}_{\mathsf{sk}'^{(i)}}(\mathsf{seed})\Big) \leq \mathsf{st}^{(i)} \cdot T$$

where $\mathsf{sk}'^{(i)}$ is the corresponding signing key for $\mathsf{vk}^{(i)}$, and T is a pre-defined threshold. When the in-equation holds, he/she can submit a special *registration transaction* in forms of

$$\mathsf{Tx}\Big(\{\mathsf{In}^{(i)}\}_{i=1}^{n}; \mathsf{TCoin}; \big\{\text{VC-Reg}, \mathsf{TID}, \overline{\mathsf{vk}}, \tilde{\mathsf{pk}}, \mathsf{sign}_{\mathsf{sk}'^{(i)}}(\mathsf{seed}), \mathsf{Addr}\big\}\Big) ,$$

where $\{\mathsf{In}^{(i)}\}_{i=1}^{n}$ are the input coins, and TCoin is a special output coin whose spending condition is defined in Subsection "supplying the treasury", below. Moreover, the coin value $\mathsf{TCoin.Value} \geq \gamma_{\min}$. In the Payload field, VC-Reg is a tag that indicates it is a special voting committee registration transaction; TID is the treasury ID that be used to uniquely identify a treasury period; $\overline{\mathsf{vk}}$ is a freshly generated signature verification key; $\tilde{\mathsf{pk}}$ is a freshly generated public key for a pre-defined public key cryptosystem; $\mathsf{sign}_{\mathsf{sk}'^{(i)}}(\mathsf{seed})$ is the signature of seed under the signing key corresponding to $\mathsf{vk}^{(i)}$; and M-Addr is the return address for the committee member to receive treasury reward. The threshold T is properly defined to ensure that approximately $\lambda' = \omega(\log \lambda)$ (e.g., $\lambda' = \mathrm{polylog}(\lambda)$) committee members are selected, assuming constant fraction of them will be active. Note that, analogous to most *proof-of-stake* systems, T needs to be updated frequently. See [**?**] for a common threshold/difficulty T adjustment approach.

Remark. Jumping ahead, we will need honest majority of the voting committee to guarantee voter privacy and protocol termination. 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 are selected among all λ' committee members. Since $\lambda' = \omega(\log \lambda)$, by Chernoff bound, we have

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

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

3.5 Components Design

Enabling stake delegation. In our treasury system, the voting power of a voter is proportional to the corresponding locked stake value. We distinguish between the ownership of the stake and the ownership of the actual coin; namely, the stake of the coin can be "owned" by a user other than the coin owner. This feature allows us to delegate the stake of a coin to someone else without transferring the ownership of the coin. To achieve this, we introduce a stake attribute, denoted as S-Attr, that can be attached to the Payload of a coin. The user who can provide the required data that satisfies the condition(s) in the S-Attr is able to claim the stake of the coin. Of course, the stake of an unspent coin can only be claimed at most once at any moment. In practice, to ensure this, additional checks should be executed. If the user A wants to delegate the stake of a coin to the user B, he simply needs to put the user B's desired S-Attr in the Payload of the coin. Note that this type of delegation is persistent in the sense that if the coin is not consumed, the S-Attr of the coin remains the same. This feature allows users to stay offline while the stake of their coins can still be used in the treasury process by the delegates. However, this type of delegation only guarantees pseudonymity-based privacy level, as everyone can learn "who owns" the stake of the coin by checking the S-Attr of the coin.

Supplying the treasury. Treasury funds are accumulated via a collection of coins. For example, the taxation/haircut of the block reward can be collected through a special transaction at the beginning of each block. The output of this type of transactions are new coins, whose spending condition, Cond, specifies that the coin can only be spent according to the corresponding treasury decision. As will be mentioned in details later, the treasury funds will be distributed in forms of transactions jointly made by the corresponding voting committee; therefore, the coins dedicated to certain treasury period must allow the voting committee in that treasury period to jointly spend. More specifically, there are λ' committee members selected at the beginning of the voting period of each treasury period. Let $\mathsf{seed}_{\mathsf{TID}^{(i)}}$ denote the seed opened in the treasury period indexed by $\mathsf{TID}^{(i)}$. Let $\{\overline{\mathsf{vk}}_j\}_{j=1}^{\ell}$ be the set of signature verification keys in the valid committee registration transactions proposed by $\mathsf{vk}^{(i)}$ such that the condition $\mathsf{hash}\big(\mathsf{vk}^{(i)}, \mathsf{sign}_{\mathsf{sk}'^{(i)}}(\mathsf{seed})\big) \leq \mathsf{st}^{(i)} \cdot T$ holds. The treasury coin can be spent in a transaction if majority of the signatures w.r.t. $\{\overline{\mathsf{vk}}_j\}_{j=1}^{\ell}$ are present.

Handling the treasury specific data in the payload. Note that typically the underlying blockchain transaction validation rules do not take into account of the content stored in the payload of a transaction. Therefore, additional checks are needed for the treasury specific transactions. More specifically, we verify the payload data of those transactions with additional algorithms. In particular, a coin must be *frozen* during the entire treasury period in order to claim its stake. This can be done by, for example, adding extra constrain in spending condition, saying that the coin cannot be spent until the certain block height, which is no earlier than the end of the treasury period. Furthermore, the stake of one coin can only be claimed once during each treasury period.

Decision making. In the decision making phase, we employ two stage voting, where all the committee members (including treasury committee, maintenance committee, and selection committee), the voters, and the experts follow the protocol description in Sec. **??**. The whole episode covers distributed key generation (including key handover), and nominating shortlisted proposals in voting stage 1 (nominating stage), ballot casting, delegation, and tally in voting stage 2 (endorsing stage). In terms of security, as shown before, with overwhelming probability, the majority of the committee members are honest, which can guarantee voter privacy and protocol termination. In an unlikely extreme case, where all the voting committee members are corrupted, our voting scheme can still ensure the integrity of the voting result. If a cheating voting committee member is detected, she will lose all her deposit.

In the nominating stage, treasury committee in each field will jointly nominate shortlisted proposals (can either be online or offline discussing). In the endorsing stage, voters can either vote directly or delegate their vote to experts; voters can choose different delegates for different proposals. In both scenario, voters/experts are expected to submit an independent ballot for each proposal from the shortlisted proposals. In our prototype, we adopt

the "YES-NO-ABSTAIN" type of voting scheme. More specifically, after the voting in the second voting stage (endorsing stage), the project proposals are scored based on *the number of* yes *votes minus the number of* no *votes*. Proposals that got at least 10% (of all votes) of the positive difference will be enrolled in final wait list. Shortlisted proposals are ranked according to their score, and the top ranked proposals are funded in turns until the treasury fund is exhausted.[JJ: We discussed before one assumption here is that we got enough budget to fund all shortlisted proposals.] Each of the voting committee members will then sign the treasury decision and treasury transactions, and those transactions are valid if it is signed by more than t-out-of-k voting committee members. [JJ: in the endorsing phase, we should know how many voters will join, otherwise how can we have the threshold signature? Thus I think we should keep the endorsement committee and design related selection, such sortition based on past behaviours]

Post-voting execution. Certain proportion (e.g. 20%) of the treasury fund will be used to reward treasury committee, selection committee, maintenance committee, endorsement committee, and experts who got delegation from endorsement committee members in endorsing stage.

The voting committee members $\mathsf{C}_\ell \in C$ will receive a fix amount of reward, denoted as ζ_1. Note that as the voting committee members are required to perform more actions in the next treasury period, their reward will only be transferred after the completion of those actions at the end of pre-voting period in the next treasury period. The voter $\mathsf{VT}^{(i)} \in \mathcal{V}$ will receive reward that is proportional to his/her deposited amount, denoted as $\zeta_2 \cdot \mathsf{st}^{(i)}$, where $\mathsf{st}^{(i)}$ is the amount of the stake claimed by $\mathsf{VT}^{(i)}$. The expert $\mathsf{E}_j \in \mathcal{E}$ will receive reward that is proportional to his/her received delegations, denoted as $\zeta_3 \cdot D_j$, where D_j is the amount of delegations that E_j has received. Meanwhile, if a voting committee member cheats or an expert fails to submit a valid ballot, he/she will lose the deposited coin as a punishment. In addition, the voting committee members will jointly generate and commit to a random seed for the next treasury period, in a protocol depicted as follows. To generate and commit a random seed, voting committee members C_ℓ, $\ell \in [k]$ needs to invoke a coin flipping protocol. However, the cost of such a protocol is very small when they already jointly setup a public key pk. More specifically, each voting committee members $\mathsf{C}_\ell, \ell \in [k]$ will pick a random group element $R_\ell \leftarrow \mathbb{G}$ and post the encryption of it, $C_\ell \leftarrow \mathsf{Enc}_{\mathsf{pk}}(R_\ell)$ to the blockchain. $C := \prod_{\ell=1}^{k} C_\ell$ is defined as the committed/encrypted seed for the next treasury period. Note that C can be jointly decrypted as far as majority of the voting committee members are honest, and the malicious voting committee members cannot influence the distribution of the seed.

Partitionary budgeting. The main goal of treasury is decentralized community-driven self-sustainable cryptocurrency development through projects funding and adoption. The naive approach is to select projects for funding by ranking all submitted proposals according to the number of votes they get and take a number of projects whose total budget does not exceed the treasury budget. However, there exists a risk of underfunding vital areas due to numerous project submissions and inflated discussions on some other areas. We can categorize proposals and allocate a certain amount of treasury funding for each category to independently guarantee funds to every vital area.

Analysis of existing blockchain development funding [**?**] reveal marketing, PR, integration, software development and organisational costs are most prominent categories. Considering this and general business development rules, we propose to include (at least) the following categories.

- **Marketing.** This covers activities devoted to cryptocurrency market share growth; market analysis, advertisement, conferences, etc. The vastness of the area demands this category should take the biggest percent of the funding budget.
- **Technology adoption.** This includes costs needed for wider spreading of cryptocurrency; integration with various platforms, websites and applications, deployment of ATMs etc.
- **Development and security.** This includes costs allocated for funding core and non-core development, security incident response, patch management, running testnets, as well as similar critical technology areas.
- **Support.** This category includes user support, documentation, maintaining of web-infrastructure needed for the community and other similar areas.
- **Organization and management.** This category includes costs on team coordination and management, legal support, etc.
- **General.** This includes projects not covered by the earlier categories, e.g., research on prospective technologies for cryptocurrency application, external security audit, collaboration with other communities, charity and so on.

It should be noted that the given list of categories is not final, and treasury deployment in cryptocurrencies will take into account specific of a given solution based on its development effort.

Nevertheless, having such an approach guarantees that critical areas for cryptocurrency routine operation, support and development will always get funding via treasury, which in turn, guarantees cryptocurrency self-sustainability.

4 THE TWO STAGE VOTING SCHEME

4.1 Security modeling

The entities involved in the voting schemes are a set of voting committee members $C := \{\mathsf{C}_1, \ldots, \mathsf{C}_k\}$, a set of voters $\mathcal{V} := \{\mathsf{VT}_1, \ldots, \mathsf{VT}_n\}$, and a set of experts $\mathcal{E} := \{\mathsf{E}_1, \ldots, \mathsf{E}_m\}$. We consider the security of our treasury voting scheme in the UC framework with static corruption. The security is based on the indistinguishability between real/hybrid world executions and ideal world executions, i.e., for any PPT real/hybrid world adversary $\mathcal{A}$ we will construct an ideal world PPT simulator $\mathcal{S}$ that can present an indistinguishable view to the environment $\mathcal{Z}$ operating the protocol.

The Ideal world execution. In the ideal world, the voting committee C, the voters $\mathcal{V}$, and the experts $\mathcal{E}$ only communicate to an ideal functionality $\mathcal{F}_{\text{VOTE}}$ during the execution. The ideal functionality $\mathcal{F}_{\text{VOTE}}$ accepts a number of commands from $C, \mathcal{V}, \mathcal{E}$. At the same time it informs the adversary of certain actions that take

place and also is influenced by the adversary to elicit certain actions. The ideal functionality $\mathcal{F}_{\text{VOTE}}$ is depicted in Fig. **??**, and it consists of three phases: Preparation, Voting/Delegation, and Tally.

Preparation phase. During the preparation phase, the voting committees $\mathsf{C}_i \in \mathcal{C}$ need to initiate the voting process by sending (INIT, *sid*) to the ideal functionality $\mathcal{F}_{\text{VOTE}}^{t,k}$. The voting will not start until all the committees have participated the preparation phase.

Voting/Delegation phase. During the voting/delegation phase, the expert $\mathsf{E}_i \in \mathcal{E}$ can vote for his choice v_i by sending (VOTE, *sid*, v_i) to the ideal functionality $\mathcal{F}_{\text{VOTE}}^{t,k}$. Note that the voting choice v_i is leaked only when majority of the voting committees are corrupted. The voter $\mathsf{VT}_j \in \mathcal{V}$, who owns α_j stake, can either vote directly for his choice v_j or delegate his voting power to an expert $\mathsf{E}_i \in \mathcal{E}$. Similarly, when all the voting committees are corrupted, $\mathcal{F}_{\text{VOTE}}^{t,k}$ leaks the voters' ballots to the adversary $\mathcal{S}$.

Tally phase. During tally phase, the voting committee $\mathsf{C}_i \in \mathcal{C}$ sends (DELCAL, sid) to the ideal functionality $\mathcal{F}_{\text{VOTE}}^{t,k}$ to calculate and reveal the delegations received by each expert. After that, they then send (TALLY, *sid*) to the ideal functionality $\mathcal{F}_{\text{VOTE}}^{t,k}$ to open the tally. Once all the committees have opened the tally, any party can read the tally by sending (READTALLY, *sid*) to $\mathcal{F}_{\text{VOTE}}^{t,k}$. Note that due to the natural of threshold cryptography, the adversary $\mathcal{S}$ can see the voting tally result before all the honest parties. Hence, the adversary can refuse to open the tally depending on the tally result. The tally algorithm TallyAlg is described in Fig. **??**.

The real/hybrid world execution. In the real/hybrid world, the treasury voting scheme utilises a number of supporting components. Those supporting components are modelled as ideal functionalities. First of all, we need a blockchain functionality $\mathcal{F}_{\text{LEDGER}}$ [**?**] to model the underlying blockchain infrastructure that the treasury system is built on. We then use the key generation functionality $\mathcal{F}_{\text{KEY}}^{t,k}$ [**?**] for threshold key generation of the underlying public key crypto system. Finally, a global clock functionality $\mathcal{G}_{\text{CLOCK}}$ [**?**] is adopted to model the synchronised network environment. Let $\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the output of the environment $\mathcal{Z}$ when interacting with parties running the protocol Π and real-world adversary $\mathcal{A}$. Let $\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote output of $\mathcal{Z}$ when running protocol ϕ interacting with the ideal functionality $\mathcal{F}$ and the ideal adversary $\mathcal{S}$.

Algorithm DelCal

Input: a set of the expert labels $\mathcal{E}$, and a set of ballots ϕ_2
Output: the delegation result δ

Init:
- For $i \in [1, m]$, create and initiate $D_i = 0$.

Delegation interpretation:
- For each ballot $B \in \phi_2$: parse B in form of $(\mathsf{VT}_j, \text{CAST}, v_j, \alpha_j)$; if $v_j = (\text{Delegate}, \mathsf{E}_i)$ for some $\mathsf{E}_i \in \mathcal{E}$, then $D_i := D_i + \alpha_j$.

Output:
- Return $\delta := \{(\mathsf{E}_i, D_i)\}_{i\in[m]}$.

Figure 5: The delegation calculation algorithm DelCal

Definition 4.1. We say that a protocol Π UC-realizes $\mathcal{F}$ 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 $\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

4.2 The voting scheme

Let m be the number of experts and n be the number of voters. Let $\mathbf{e}_i^{(m)} \in \{0,1\}^m$ be the unit vector where its i-th coordinate is 1 and the rest coordinates are 0. We also abuse the notation to denote $\mathbf{e}_0^{(\ell)}$ as an ℓ-vector contains all 0's. We use $\text{Enc}_{\text{pk}}(\mathbf{e}_i^{(\ell)})$ to denote coordinate-wise encryption of $\mathbf{e}_i^{(\ell)}$, i.e. $\text{Enc}_{\text{pk}}(e_{i,1}^{(\ell)}), \ldots, \text{Enc}_{\text{pk}}(e_{i,\ell}^{(1)})$, where $\mathbf{e}_i^{(\ell)} = (e_{i,1}^{(\ell)}, \ldots, e_{i,\ell}^{(\ell)})$.

The tally algorithm TallyAlg

Input: a set of the voters $\mathcal{V}$, a set of the experts $\mathcal{E}$, two sets of ballots ϕ_1, ϕ_2 and the delegation δ.
Output: the tally result τ

Init:
- Create and initiate $\tau_{\text{yes}} = 0$, $\tau_{\text{no}} = 0$ and $\tau_{\text{abstain}} = 0$.
- Parse δ as $\{(\mathsf{E}_i, D_i)\}_{i\in[m]}$.

Tally Computation:
- For each ballot $B \in \phi_2$: parse B in form of $(\mathsf{VT}_j, \text{CAST}, v_j, \alpha_j)$; if $v_j = (\text{Vote}, a_j)$ for some $a_j \in \{\text{yes}, \text{no}, \text{abstain}\}$, then $\tau_{a_j} := \tau_{a_j} + \alpha_j$.
- For each ballot $B \in \phi_1$: parse B in form of $(\mathsf{E}_i, \text{VOTE}, b_i)$ for some $b_i \in \{\text{yes}, \text{no}, \text{abstain}\}$, then $\tau_{b_i} := \tau_{b_i} + D_i$.

Output:
- Return $\tau := (\tau_{\text{yes}}, \tau_{\text{no}}, \tau_{\text{abstain}})$.

Figure 6: The tally algorithm

4.2.1 Vote encoding. In our scheme, we encode the vote into a (unit) vector. Let $\text{encode}^{\mathsf{E}}$ and $\text{encode}^{\mathsf{VT}}$ be the vote encoding algorithm for the expert and voter, respectively. For an expert, upon receiving input $x \in \{\text{YES}, \text{NO}, \text{ABSTAIN}\}$, the $\text{encode}^{\mathsf{E}}$ returns 100, 010, 001 for YES, NO, ABSTAIN, respectively. For a voter, the input is $y \in \{\mathsf{E}_1, \ldots, \mathsf{E}_m\} \cup \{\text{YES}, \text{NO}, \text{ABSTAIN}\}$. When $y = \mathsf{E}_i$, $i \in [m]$, it means that the voter delegate his/her delegate his voting power to the expert E_i. When $y \in \{\text{YES}, \text{NO}, \text{ABSTAIN}\}$, it means that the voter directly vote to the project. The $\text{encode}^{\mathsf{VT}}$ returns a unit vector of length $(m+3)$, denoted as v, such that $v = e_i^{(m+3)}$ if $y = \mathsf{E}_i$, for $i \in [m]$; and v is set to $e_{m+1}^{(m+3)}$, $e_{m+2}^{(m+3)}$, and $e_{m+3}^{(m+3)}$ if y is YES, NO, ABSTAIN, respectively.

Since sending data to the blockchain consumes coins, we implicitly assume all the experts $\mathcal{E}$ and voters $\mathcal{V}$ have spare coins to pay the transaction fees that occurred during the protocol execution. More specifically, we let each party prepare $\{\mathsf{In}_i\}_{i=1}^{\ell_1}, \{\mathsf{Out}_j\}_{j=1}^{\ell_2}$ s.t.

$$\sum_{i=1}^{\ell_1} \mathsf{In}_i.\mathsf{Value} \geq \sum_{j=1}^{\ell_2} \mathsf{Out}_j.\mathsf{Value} \ .$$

Denote the corresponding coins owned by a voter $\mathsf{VT}_i \in \mathcal{V}$, an expert $\mathsf{E}_j \in \mathcal{E}$, and a voting committee member $\mathsf{C}_t \in \mathcal{C}$ as $(\{\mathsf{In}_\eta^{(\mathsf{VT}_i)}\}_{\eta=1}^{\ell_1}, \{\mathsf{Out}_\eta^{(\mathsf{VT}_i)}\}$

$(\{\mathsf{In}_\eta^{(\mathsf{E}_j)}\}_{\eta=1}^{\ell_1}, \{\mathsf{Out}_\eta^{(\mathsf{E}_j)}\}_{\eta=1}^{\ell_2})$, and $(\{\mathsf{In}_\eta^{(\mathsf{C}_t)}\}_{\eta=1}^{\ell_1}, \{\mathsf{Out}_\eta^{(\mathsf{C}_t)}\}_{\eta=1}^{\ell_2})$, respectively. The protocol is depicted in Fig. **??**. It consists of preparation phase, voting/delegation phase, and tally phase.

Sending/Reading data to/from $\mathcal{F}_{\text{Ledger}}$. Fig. **??** describes the macro for a party to send and read data to/from the blockchain $\mathcal{F}_{\text{Ledger}}$. According the blockchain model proposed by [**?**], three types of delays need to be considered. First, we have a bounded network delay, and it is assumed that all messages can be delivered within Δ_1 rounds, which is $2\Delta_1$ clock-ticks in [**?**]. Subsequently, a desynchronised user can get up-to-date within $2\Delta_1$ rounds (i.e. $4\Delta_1$ clock-ticks) after registration. The second type of delay is the fact that the adversary can hold a valid transaction up to certain blocks, but she cannot permanently denial-of-service such a transaction. This is modeled by the ExtendPolicy in $\mathcal{F}_{\text{Ledger}}$, where if a transaction is more than Δ_2 rounds (i.e. $2\Delta_2$ clock-ticks) old, and still valid with respect to the current state, then it will be included into the state. Finally, we have a so-called windowsize. Namely, the adversary can set state-slackness of all the honest parties up to the windowsize, which is consistent with the *common prefix* property in [**?**]. Hence, all the honest parties can have a common state of any blocks that have been proposed more than windowsize. Denote Δ_3 rounds (i.e. $2\Delta_3$ clock-ticks) as the windowsize.

To send a message x to $\mathcal{F}_{\text{Ledger}}$, we need to first check if this party has deregistered and desynchronized. If so, the party needs to first send (Register, sid) to $\mathcal{F}_{\text{Ledger}}$. Note that the registered but desynchronized party can still send a transaction before it is fully updated. We simply make a 'dummy' transaction whose input coins and output coins share the same owner (spending condition), and the message x is stored in the payload of the transaction. To read a message (stored in the payload of some transaction) from $\mathcal{F}_{\text{Ledger}}$, analogously a deregistered party needs to first send (Register, sid) to $\mathcal{F}_{\text{Ledger}}$. After $4\delta_1$ clock-ticks, the party can get synchronised. In order to receive the latest message, the party needs to wait a maximum of $2(\Delta_2 + \Delta_3)$ clock-ticks until the transaction that carries the intended message to be included in the state of the party.

5 RECEIVER NON-COMMITTING ENCRYPTION

To achieve proactive security, receiver non-committing encryption (RNCE) is a key building block of our system. As shown in [**?**], a (polynomial-size message space) RNCE scheme can be instantiated from the DDH assumption using the idea of the Cramer-Shoup cryptosystem [**?**]. In this work, since we will use RNCE together with some (non-interactive) zero-knowledge proofs, we can further simplify the core part of the RNCE. As depicted in Fig. **??**, an RNCE scheme consists of five PPT algorithms RNCE := (KeyGen, Enc, Dec, SimEnc, Reveal).

5.1 UC ideal functionalities

In this work, we adopt the following UC ideal functionalities as building blocks.

5.1.1 The distributed key generation functionality. We use the key generation functionality $\mathcal{F}_{\text{KEY}}^{t,k}$ [**?**] for threshold key generation of the underlying public key crypto system. The functionality depicted in Fig. **??**, interacts with a set of committees $\mathcal{C} := \{\mathsf{C}_1, \ldots, \mathsf{C}_k\}$ to generate a public key pk and deal the corresponding secret key sk among the committees. In Appendix **??**, we give a detailed description of our threshold distributed key generation protocol adopted from Gennaro *et al.* [**?**].

Let $\mathbb{G} = \{\mathbb{G}_k\}_{k\in\mathbb{N}}$ be a family of finite cyclic groups, where each group $\mathbb{G}_k$ has known prime order q_k and $|q_k| = k$. Secure RNCE scheme is shown in Fig. **??**.

Observe that $\mu^{m'}\rho^{n'} = \mu^{m+s\cdot c}\cdot\rho^n\cdot\mu^{-c\cdot s} = \mu^m\rho^n = \sigma$, then we can show the fake ciphertext $(\ddot{v}, \ddot{\omega}, \ddot{\epsilon}, \ddot{\kappa})$ with fake secret key $\ddot{sk} := (m', n', x, y)$ can decrypt the original message s:

$$\ddot{\epsilon}/(\ddot{v}^{m'}\cdot\ddot{w}^{n'}) = \mu^n\sigma^r/((\mu^r)^{m'}\cdot(\mu\rho^r)^{n'}) = \mu^n\sigma^r/(\sigma^r\cdot\mu^{n'}) = \mu^s$$

5.2 RNCE based DKG protocol

6 UNIT VECTOR ZK PROOF WITH LOGARITHMIC SIZE COMMUNICATION

Let $\mathbf{e}_i^{(n)} = (e_{i,0}, \ldots, e_{i,n-1})$ denote a unit vector where its i-th coordinate is 1 and the rest coordinates are 0. Conventionally, to show a vector of ElGamal ciphertexts element-wise encrypt a unit vector, Chaum-Pederson proofs are used to show each of the ciphertexts encrypts either 0 or 1 (via Sigma OR composition) and the product of all the ciphertexts encrypts 1. Such kind of proof is used in many well-known voting schemes, e.g., Helios. However, the proof size is linear to the length of the unit vector, and thus the communication overhead is quit significant when the unit vector length becomes larger.

In this section, we propose a novel special honest verifier ZK (SHVZK) proof for unit vector that allows the prover to convince the verifier that a vector of ciphertexts $(C_0, \ldots, C_{n-1})$ encrypts a unit vector $\mathbf{e}_i^{(n)}$, $i \in [0, n-1]$ with $O(\log n)$ proof size. Without loss of generality, assume n is a perfect power of 2. If not, we append $\mathsf{Enc}_{\mathsf{pk}}(0;0)$ (i.e., trivial ciphertexts) to make the total number of ciphertexts to be the next power of 2. The proposed SHVZK protocol can also be Fiat-Shamir transformed to a non-interactive ZK (NIZK) proof in the random oracle model. The key idea behind our construction is that there exists a data-oblivious algorithm that can take input as $i \in \{0,1\}^{\log n}$ and output the unit vector $\mathbf{e}_i^{(n)}$. Let $i_1, \ldots, i_{\log n}$ be the binary representation of i. The algorithm is depicted in Fig. **??**.

Intuitively, we let the prover first bit-wisely commit the binary presentation of $i \in [0, n-1]$ for the unit vector $\mathbf{e}_i^{(n)}$. The prover then shows that each of the commitments of $(i_1, \ldots, i_{\log n})$ indeed contain 0 or 1, using the Sigma protocols. Note that in the 3rd move of such a Sigma protocol, the prover reveals a degree-1 polynomial of the committed message. Denote $z_{\ell,1} := i_\ell x + r_\ell$, $\ell \in [\log n]$ as the corresponding degree-1 polynomials, where r_ℓ are chosen by the prover and x is chosen by the verifier. By linearity, we can also define $z_{\ell,0} := x - z_{\ell,1} = (1 - i_\ell)x - r_\ell$, $\ell \in [\log n]$. According to the algorithm described in Fig.**??**, for $j \in [0, n-1]$, let $j_1, \ldots, j_{\log n}$ be the binary representation of j, and the product $\prod_{\ell=1}^{\log n} z_{\ell,j_\ell}$ can

The voting protocol $\Pi^{t,k,m,n}_{\text{TREASURY}}$

The protocol interacts with a set of committee members $C := \{\mathsf{C}_1, \ldots, \mathsf{C}_k\}$, a set of experts $\mathcal{E} := \{\mathsf{E}_1, \ldots, \mathsf{E}_m\}$, and a set of voters $\mathcal{V} := \{\mathsf{VT}_1, \ldots, \mathsf{VT}_n\}$.

Initialization phase:

- Upon receiving (INIT, sid) from the environment $\mathcal{Z}$, the committee member $\mathsf{C}_o \in C$, $o \in [k]$ does the following:
  - For $i \in [m+3]$, send (KEYGEN, sid, ssid := (sid, i)) to $\mathcal{F}^{t}_{\text{KEY}}$[RNCE];

Voting/Delegation phase:

- Upon receiving (VOTE, sid, u_j) from the environment $\mathcal{Z}$, the expert E_j, $j \in [m]$ does the following:
  - Send (READPK, sid) to $\mathcal{F}^{t,\ell}_{\text{KEY}}$[RNCE], and receive (PUBLICKEY, sid, $\{\mathsf{pk}_\eta\}_{\eta\in[\ell]}$) from $\mathcal{F}^{t,\ell}_{\text{KEY}}$[RNCE];
  - Set the unit vector $(e_1, e_2, e_3) \leftarrow \mathsf{encode}^{\mathsf{E}}(u_j)$;
  - For $\eta \in [3]$, encrypt $U_{j,\eta} \leftarrow \mathsf{RNCE.Enc}_{\mathsf{pk}_\eta}(e_\eta)$;
  - Generator the NIZK proof π_j (Cf. Sec. **??**);
  - Post $(\mathsf{E}_j, \{U_{j,\eta}\}_{\eta\in[3]}, \pi_j)$ to $\mathcal{F}_{\text{LEDGER}}$;
- Upon receiving (CAST, sid, v_i, α_i) from the environment $\mathcal{Z}$, the voter VT_i, $i \in [n]$ does the following:
  - Send (READPK, sid) to $\mathcal{F}^{t,\ell}_{\text{KEY}}$[RNCE], and receive (PUBLICKEY, sid, $\{\mathsf{pk}_\eta\}_{\eta\in[\ell]}$) from $\mathcal{F}^{t,\ell}_{\text{KEY}}$[RNCE];
  - Set the unit vector $(e_1, \ldots, e_{m+3}) \leftarrow \mathsf{encode}^{\mathsf{VT}}(v_i)$.
  - For $\eta \in [m+3]$, encrypt $V_{i,\eta} \leftarrow \mathsf{RNCE.Enc}_{\mathsf{pk}_\eta}(e_\eta)$;
  - Generator the NIZK proof σ_i (Cf. Sec. **??**);
  - Post $(\mathsf{VT}_i, \{V_{i,\eta}\}_{\eta\in[m+3]}, \sigma_i, \alpha_i)$ to $\mathcal{F}_{\text{LEDGER}}$;

Tally phase:

- Upon receiving (DELCAL, sid) from the environment $\mathcal{Z}$, the committee C_o, $o \in [k]$ does:
  - Fetch the ballots $(\mathsf{E}_j, \{U_{j,\eta}\}_{\eta\in[3]}, \pi_j)$ and $(\mathsf{VT}_i, \{V_{i,\eta}\}_{\eta\in[m+3]}, \sigma_i, \alpha_i)$ from $\mathcal{F}_{\text{LEDGER}}$.
  - For $i \in [m]$, check $\mathsf{Verify}(\{U_{j,\eta}\}_{\eta\in[3]}, \pi_j) = 1$; for $j \in [n]$, $\mathsf{Verify}(\{V_{i,\eta}\}_{\eta\in[m+3]}, \sigma_i) = 1$. Remove all invalid ballots.
  - For a valid ballot $\{V_{i,\eta}\}_{\eta\in[m+3], i\in[n']}$:
    - For $\eta \in [m+3]$, compute $\hat{V}_{i,\eta} := (V_{i,\eta})^{\alpha_i}$.
  - For $\eta \in [4, m+3]$, compute $D_{\eta-3} := \prod_{i\in[n']} \hat{V}_{i,\eta}$ and send (DECRYPT, sid, ssid := (sid, η), $D_{\eta-3}$) to $\mathcal{F}^{t}_{\text{KEY}}$[RNCE], obtaining $d_{\eta-3}$;
  - Post $(\mathsf{C}_o, \{d_j\}_{j\in[m]})$ to $\mathcal{F}_{\text{LEDGER}}$;
- Upon receiving (TALLY, sid) from the environment $\mathcal{Z}$, the committee C_o, $o \in [k]$ does:
  - For $\eta \in [3]$, compute $S_\eta := \prod_{j=1}^{m} (U_{j,\eta})^{d_j} \cdot \prod_{i=1}^{n} \hat{V}_{i,\eta}$ and send (DECRYPT, sid, ssid := (sid, η), S_η) to $\mathcal{F}^{t}_{\text{KEY}}$[RNCE], obtaining y_η;
  - Post $(\mathsf{C}_o, \{d_\eta\}_{\eta\in[3]})$ to $\mathcal{F}_{\text{LEDGER}}$;
- Upon receiving (READTALLY, sid) from the environment $\mathcal{Z}$, the party P does the following:
  - Fetch $\{d_\eta\}_{\eta\in[3]}$ from $\mathcal{F}_{\text{LEDGER}}$.

Figure 7: The voting protocol $\Pi^{t,k,m,n}_{\text{TREASURY}}$ in $\{\mathcal{F}_{\text{LEDGER}}, \mathcal{F}^{t}_{\text{KEY}}[\text{RNCE}]\}$-hybrid model

be viewed as a degree-($\log n$) polynomial of the form

$$p_j(x) = e_{i,j} x^{\log n} + \sum_{k=0}^{\log n - 1} p_{j,k} x^k$$

for some $p_{j,k}$, $k \in [0, \log n - 1]$. We then use batch verification to show that each of C_j indeed encrypts $e_{i,j}$. More specifically, for a randomly chosen $y \leftarrow \mathbb{Z}_q$, let $E_j := (C_j)^{x^{\log n}} \cdot \mathsf{Enc}(-p_j(x); 0)$; the prover needs to show that $E := \sum_{j=0}^{n-1} (E_j)^{y^j} \cdot \prod_{k=0}^{\log n - 1} (D_k)^{x^k}$ encrypts 0, where $D_\ell := \mathsf{Enc}_{\mathsf{pk}}(\sum_{j=0}^{n-1} (p_{j,k} \cdot y^j); R_\ell)$, $\ell \in [0, \log n - 1]$ with fresh randomness $R_\ell \in \mathbb{Z}_q$. The construction is depicted in Fig. **??**, and it consists of 5 moves. Both the prover and the verifier shares a common reference string (CRS), which is a Pedersen commitment key that can be generated using random oracle. The prover first commits to each bits of the binary representation of i, and the commitments are denoted as I_ℓ, $\ell \in [\log n]$. Subsequently, it produces B_ℓ, A_ℓ as the first move of the Sigma protocol showing I_ℓ commits to 0 or 1. Jumping ahead, later the prover will receive a challenge $x \leftarrow \{0,1\}^\lambda$, and it then computes the third move of the Sigma protocols by producing $\{z_\ell, w_\ell, v_\ell\}_{\ell=1}^{\log n}$. To enable batch verification, before that, the prover is given another challenge $y \leftarrow \{0,1\}^\lambda$ in the second move. The prover the computes and sends the aforementioned $\{D_\ell\}_{\ell=0}^{\log n - 1}$. The verification consists of two parts. In the first part, the verifier checks the following equations to ensure that I_ℓ commits to 0 or 1.

- $I_\ell^x \cdot B_\ell = \mathsf{Com}_{\mathsf{ck}}(z_\ell; w_\ell) = g^{z_\ell} h^{w_\ell}$

Sending and reading messages

Macro Send-Msg(x, $\{\mathsf{In}_i\}_{i=1}^{\ell_1}$, $\{\mathsf{Out}_j\}_{j=1}^{\ell_2}$):

- If the party has deregistered and desynchronized:
  - Send (Register, sid) to $\mathcal{F}_{\text{Ledger}}$.
  - Send $\left(\text{Submit}, sid, \mathsf{Tx}(\{\mathsf{In}_i\}_{i=1}^{\ell_1}; \{\mathsf{Out}_j\}_{j=1}^{\ell_2}; x)\right)$ to $\mathcal{F}_{\text{Ledger}}$.
  - Send (De-Register, sid) to $\mathcal{F}_{\text{Ledger}}$.
- If the party is already synchronized:
  - Send $\left(\text{Submit}, sid, \mathsf{Tx}(\{\mathsf{In}_i\}_{i=1}^{\ell_1}; \{\mathsf{Out}_j\}_{j=1}^{\ell_2}; x)\right)$ to $\mathcal{F}_{\text{Ledger}}$.

Macro Read-Msg:

- If the party has deregistered and desynchronized:
  - Send (Register, sid) to $\mathcal{F}_{\text{Ledger}}$.
  - Wait for $\max\{4\Delta_1, 2(\Delta_2+\Delta_3)\}$ clock-ticks by keeping sending (Tick, sid) to the $\mathcal{G}_{\text{Clock}}$.
  - Send (Read, sid) to $\mathcal{F}_{\text{Ledger}}$ and receive (Read, sid, data) from $\mathcal{F}_{\text{Ledger}}$.
  - Send (De-Register, sid) to $\mathcal{F}_{\text{Ledger}}$.
- If the party is already synchronized:
  - Wait for $\max\{4\Delta_1, 2(\Delta_2+\Delta_3)\}$ clock-ticks by keeping sending (Tick, sid) to the $\mathcal{G}_{\text{Clock}}$.
  - Send (Read, sid) to $\mathcal{F}_{\text{Ledger}}$ and receive (Read, sid, data) from $\mathcal{F}_{\text{Ledger}}$.
- Return data.

Figure 8: Macro for sending and receiving message via $\mathcal{F}_{\text{Ledger}}$

Functionality $\mathcal{F}_{\text{KEY}}^{t}$[RNCE]

The functionality $\mathcal{F}_{\text{KEY}}^{t}$ interacts with a set of committee $C := \{C_1, \ldots, C_n\}$, a set of parties $\mathcal{P}$, and the adversary $\mathcal{A}$. It is parameterised with the RNCE scheme RNCE.

- Upon receiving (KeyGen, sid, ssid) from at least t committee members in C, generate a public key pair $(\mathsf{pk}_{\mathsf{ssid}}, \mathsf{sk}_{\mathsf{ssid}}) \leftarrow \mathsf{RNCE.KeyGen}(\mathsf{gk})$;
- Upon receiving (ReadPK, sid, ssid) from any party $p \in \mathcal{P} \cup \mathcal{A}$, send (PublicKey, sid, $\mathsf{pk}_{\mathsf{ssid}}$) to the requestor.
- Upon receiving (Decrypt, sid, ssid, c) from at least t committee members in C, decrypts $m \leftarrow \mathsf{RNCE.Dec}_{\mathsf{sk}_{\mathsf{ssid}}}(c)$ and sends (Decrypt, sid, m) to the requestors.

Figure 9: Functionality $\mathcal{F}_{\text{KEY}}^{t}$[RNCE]

- $I_\ell^{x-z_\ell} \cdot A_\ell = \mathsf{Com}_{\mathsf{ck}}(0; v_\ell) = h^{v_\ell}$

In the second part, the verifier checks if

$$\prod_{j=0}^{n-1} \left((C_j)^{x^{\log n}} \cdot \mathsf{Enc}_{\mathsf{pk}}(-\prod_{\ell=1}^{\log n} z_{\ell,j_\ell}; 0)\right)^{y^j} \cdot \prod_{\ell=0}^{\log n-1} (D_\ell)^{x^\ell}$$

is encryption of 0 by asking the prover to reveal the randomness.

Theorem 6.1. *The protocol described in Fig.* **??** *is a* 5*-move public coin special honest verifier zero-knowledge arguement of knowledge of* $\mathbf{e}_i^{(n)}$ *and* $(r_0, \ldots, r_{n-1}) \in (\mathbb{Z}_p)^n$ *such that* $C_j = \mathsf{Enc}_{\mathsf{pk}}(e_{i,j}; r_j)$, $j \in [0, n-1]$.

Proof. For perfect completeness, we first observe that the verification equations $(I_\ell)^x \cdot B_\ell = \mathsf{Com}_{\mathsf{ck}}(z_\ell; w_\ell)$ and $(I_\ell)^{x-z_\ell} \cdot A_\ell = \mathsf{Com}_{\mathsf{ck}}(0; v_\ell)$ holds. Indeed, by additively homomorphic property of the commitment scheme, $(I_\ell)^x \cdot B_\ell = \mathsf{Com}_{\mathsf{ck}}(i_\ell \cdot x + \beta_\ell; \alpha_\ell \cdot x + \gamma_\ell)$ and $(I_\ell)^{x-z_\ell} \cdot A_\ell = \mathsf{Com}_{\mathsf{ck}}(i_\ell \cdot (x - z_\ell) + i_\ell \cdot \beta_\ell; \alpha_\ell \cdot (x - z_\ell) + \delta_\ell) = \mathsf{Com}_{\mathsf{ck}}(i_\ell(1 - i_\ell) \cdot x; v_\ell)$. Since $i_\ell(1 - i_\ell) = 0$ when $i_\ell \in \{0, 1\}$, we have $(I_\ell)^{x-z_\ell} \cdot A_\ell = \mathsf{Com}_{\mathsf{ck}}(0; v_\ell)$. Moreover, for each $j \in [0, n-1]$, $\prod_{\ell=1}^{\log n} z_{\ell,j_\ell}$ is a polynomial in the form of

$$p_j(x) = e_{i,j} x^{\log n} + \sum_{k=0}^{\log n-1} p_{j,k} x^k$$

where x is the verifier's challenge. Therefore, it is easy to see that

$$\prod_{j=0}^{n-1} \left((C_j)^{x^{\log n}} \cdot \mathsf{Enc}_{\mathsf{pk}}(-\prod_{\ell=1}^{\log n} z_{\ell,j_\ell}; 0)\right)^{y^j} \cdot \prod_{\ell=0}^{\log n-1} \mathsf{Enc}_{\mathsf{pk}}(\sum_{j=0}^{n-1} (p_{j,\ell} \cdot y^j); R_\ell)^{x^\ell}$$

$$= \mathsf{Enc}_{\mathsf{pk}}\Big(\sum_{j=0}^{n-1} \left(e_{i,j} \cdot x^{\log n} - p_j(x) + \sum_{\ell=0}^{\log n-1} p_{j,\ell} \cdot x^\ell\right) \cdot y^j; R\Big) = \mathsf{Enc}_{\mathsf{pk}}(0; R) \ .$$

For soundness, first of all, the Sigma protocols for commitments of $i_\ell, \ell \in [\log n]$ is specially sound, i.e., given two transactions with the same $\{I_\ell, B_\ell, A_\ell\}_{\ell=1}^{\log n}$ and two different x and $\{z_\ell, w_\ell, v_\ell\}_{\ell=1}^{\log n}$, there exists a PPT extractor that can output the corresponding witness $i_\ell \in \{0, 1\}$. The protocol builds and evaluates a degree-$\log n$ polynomial $p_j(x)$. By Schwartz-Zippel lemma, we have $e_{i,j} = \prod_{\ell=1}^{\log n} i_{\ell,j_\ell}$ with overwhelming probability. This concludes the proof.

Key maintenance protocol $\Pi_{\text{KEY}}^{t,\ell}[\text{RNCE}]$

CRS: $\mathbb{G}$ and $g_1, g_2 \in \mathbb{G} \setminus \{1\}$.
Init Setup: $\mathcal{K} := \{\mathsf{PK}_i\}_{i\in[n]}$.

In the δ-round ($\delta \geq 1$), $P_i \in \mathcal{P}$ sends $(\text{QUERY}, \text{sid}, \text{ssid} := \delta)$ to $\mathcal{F}_{\text{SORTION}}$ and obtains $b_{P_i}^{\text{ssid}}$. If $b_{P_i}^{\text{ssid}} = 1$, do the following:

- Randomly select a party $\hat{P}_j \leftarrow \mathcal{P}$.
- For $j \in [\ell]$, generate $(\hat{\mathsf{pk}}_{i,j}^{(\delta)}, \hat{\mathsf{sk}}_{i,j}^{(\delta)}) \leftarrow \text{RNCE.KeyGen}(\text{gk})$.
- Encrypt $\mathsf{CT}_i^{(\delta)} \leftarrow \text{AnonE.Enc}_{\mathsf{PK}_j}(\hat{\mathsf{sk}}_{i,1}^{(\delta)}, \ldots, \hat{\mathsf{sk}}_{i,\ell}^{(\delta)})$.
- Send $(\text{Reveal}, \text{sid}, \text{ssid})$ to $\mathcal{F}_{\text{SORTION}}$ and Post $(\delta, P_i, \{\hat{\mathsf{pk}}_{i,j}^{(\delta)}\}_{j\in[\ell]}, \mathsf{CT}_i^{(\delta)})$ to $\mathcal{F}_{\text{LEDGER}}$.

In the δ-round ($\delta \geq 3$), $P_i \in \mathcal{P}$ create $K^{(\delta)} := \emptyset$:

- Fetch $\{(\delta-2, P_j, \{\hat{\mathsf{pk}}_{j,k}^{(\delta-2)}\}_{k\in[\ell]}, \mathsf{CT}_j^{(\delta-2)})\}_{j\in N_{\delta-2}}$ and $\{(\delta-1, P_j, \{\hat{\mathsf{pk}}_{j,k}^{(\delta-1)}\}_{k\in[\ell]}, \mathsf{CT}_j^{(\delta-1)})\}_{j\in N_{\delta-1}}$ to $\mathcal{F}_{\text{LEDGER}}$;
- Send $(\text{STATE}, \text{sid}, \text{ssid} := \delta-2)$ to $\mathcal{F}_{\text{SORTION}}$ and obtain Pub[ssid];
- For $j \in [N_{\delta-1}]$:
  - Assert $\langle P_j, b_{P_j}^{\delta-2} = 1\rangle$ is in record Pub[ssid];
  - Decrypt $(\hat{\mathsf{sk}}_{j,1}^{(\delta-2)}, \ldots, \hat{\mathsf{sk}}_{j,\ell}^{(\delta-2)}) \leftarrow \text{AnonE.Dec}_{\mathsf{SK}_i}(\mathsf{CT}_j^{(\delta-2)})$;
- Set $K^{(\delta-2)} := \{(\hat{\mathsf{pk}}_{j,k}^{(\delta-2)}, \hat{\mathsf{sk}}_{j,k}^{(\delta-2)})\}_{j\in[N_{\delta-1}],k\in[\ell]}$;
- If $K^{(\delta-2)} \neq \emptyset$ and $\delta = 3$, do the following:
  - Randomly choose $s_i, s_i' \leftarrow \mathbb{Z}_q$;
- If $K^{(\delta-2)} \neq \emptyset$ and $\delta > 3$, do the following:
  - Fetch $M_1^{(\delta)}, \ldots, M_{N_{\delta-2}}^{(\delta)}$ from $\mathcal{F}_{\text{LEDGER}}$ and verify all the NIZK proofs. Remove invalid tuples.
  - Let QUAL_δ be the qualified set with size N_{QUAL_δ}. Set corresponding ciphertexts as $\{ct_{i',i,k}\}_{i'=1,k=1}^{N_{\text{QUAL}_\delta},\ell}$; /*Global Public Key can be extracted by $\prod_{i'=1}^{N_{\text{QUAL}_\delta}} \sigma_{i'}$*/
  - Compute $s_{i',i,k} := \text{Dlog}_{g_1}(\epsilon_{i',i,k}/(v_{i',i,k}^{m_{f,k}} \cdot \omega_{i',i,k}^{n_{f,k}}))$, $s'_{i',i,k} := \text{Dlog}_{g_1}(\epsilon'_{i',i,k}/(v'^{m_{f,k}}_{i',i,k} \cdot \omega'^{n_{f,k}}_{i',i,k}))$;
  - Compute $s_i := \begin{cases} \sum_{i'=1}^{N_{\text{QUAL}_\delta}} \sum_{k=1}^{\ell} s_{i',i,k} \cdot p^k & \delta = 3 \\ \sum_{i'=1}^{t_{\text{QUAL}_\delta}} (\sum_{k=1}^{\ell} s_{i',i,k} \cdot p^k)\gamma_{i',i} & \delta > 3 \end{cases}$, $s_i' := \begin{cases} \sum_{i'=1}^{N_{\text{QUAL}_\delta}} \sum_{k=1}^{\ell} s'_{i',i,k} \cdot p^k & \delta = 3 \\ \sum_{i'=1}^{t_{\text{QUAL}_\delta}} (\sum_{k=1}^{\ell} s'_{i',i,k} \cdot p^k)\gamma'_{i',i} & \delta > 3 \end{cases}$
  - Generate two degree-$t_{\delta-1}$ random polynomials over $\mathbb{Z}_q$, where $t_{\delta-1} := \lfloor \frac{N_{\delta-1}}{2} \rfloor + 1$, $a_{i,0} := s_i, a'_{i,0} := s'_i$: $F_i(x) = a_{i,0} + a_{i,1}x + \cdots + a_{i,t_{\delta-1}}x^{t_{\delta-1}}$, $F'_i(x) = a'_{i,0} + a'_{i,1}x + \cdots + a'_{i,t_{\delta-1}}x^{t_{\delta-1}}$;
  - For $\eta \in [0, t_{\delta-1}]$ commit $\sigma_\eta := g_1^{a_{i,\eta}} \cdot g_2^{a'_{i,\eta}}$;
  - For $j \in [N_{\delta-1}]$:
    - Compute $\{s_{i,j} := F_i(j), s'_{i,j} := F'_i(j)\}_{j=1}^{N_{\delta-2}}$;
    - Truncate $s_{i,j}, s'_{i,j}$ into ℓ elements: $\{s_{i,j,0}, \cdots, s_{i,j,\ell}\}$ and $\{s'_{i,j,0}, \cdots, s'_{i,j,\ell}\}$, where $\sum_{k=1}^{\ell} s_{i,j,k}p^k = s_{i,j}$, $\sum_{k=1}^{\ell} s'_{i,j,k}p^k = s'_{i,j}$, p is a perfect power of 2.
    - For $k \in [\ell]$, parse $\hat{\mathsf{pk}}_{j,k}^{(\delta-2)} = h_{j,k}$:
      - Choose $r_{i,j,k}, r'_{i,j,k} \leftarrow \mathbb{Z}_q$, compute $v_{i,j,k} := g_1^{r_{i,j,k}}$, $\omega_{i,j,k} := g_2^{r_{i,j,k}}$, $\epsilon_{i,j,k} := g_1^{s_{i,j,k}} h_{j,k}^{r_{i,j,k}}$, $v'_{i,j,k} := g_1^{r'_{i,j,k}}$, $\omega'_{i,j,k} := g_2^{r'_{i,j,k}}$, $\epsilon'_{i,j,k} := g_1^{s'_{i,j,k}} h_{j,k}^{r'_{i,j,k}}$, set $ct_{i,j,k} := (v_{i,j,k}, \omega_{i,j,k}, \epsilon_{i,j,k}, v'_{i,j,k}, \omega'_{i,j,k}, \epsilon'_{i,j,k})$;
- Generate NIZK proof π_i. (Cf. Fig.??);
- Generate a new pair of long-term key $(\mathsf{PK}_i^*, \mathsf{SK}_i^*) \leftarrow \text{AnonE.KeyGen}(1^\lambda)$;
- Erase the state and post $M_i^{(\delta)} := (\{\sigma_\eta\}_{\eta\in[0,t_{\delta-1}]}, \pi_i, \mathsf{PK}_i^*, \{ct_{i,j,k}\}_{j=1,k=1}^{N_{\delta-1},\ell})$ to $\mathcal{F}_{\text{LEDGER}}$;

Figure 10: Key maintenance protocol $\Pi_{\text{KEY}}^{t,\ell}[\text{RNCE}]$

Reveiver Non-committing Encryption

Key Generation $\mathsf{KeyGen}(\mathsf{gk} := (q, g_1, g_2, \mathbb{G}))$:

- Pick random $x, y \leftarrow \mathbb{Z}_q$;
- Compute $h := g_1^x g_2^y$;
- Output $(\mathsf{pk} := (\mathsf{gk}, h), \mathsf{sk} := (\mathsf{pk}, x, y))$;

Encryption $\mathsf{Enc}_{\mathsf{pk}}(m; r)$:

- Compute $c_1 := g_1^r$, $c_2 := g_2^r$ and $c_3 := g_1^m h^r$;
- Compute $\pi \leftarrow \mathsf{PoK.Prove}((\langle \mathsf{PK}, c_1, c_2, c_3\rangle, \langle m, r\rangle) : c_1 = g_1^r \ \wedge \ c_2 := g_2^r \ \wedge \ c_3 := g_1^m h^r)$;
- Output $c := (c_1, c_2, c_3, \pi)$;

Decryption $\mathsf{Dec}_{\mathsf{sk}}(c)$:

- Assert $\mathsf{PoK.Verify}(\mathsf{PK}, c_1, c_2, c_3, \pi) = 1$, otherwise, output $\perp$;
- Compute $D := c_1^x c_2^y$ and $M := c_3/D$;
- Output $m := \mathsf{Dlog}_{g_1}(M)$;

Sim Encryption $\mathsf{SimEnc}_{\mathsf{sk}}(s)$:

- Compute $c_1' := g_1^s$, $c_2' := g_1 g_2^s$ and $c_3' := g_1^y h^s$;
- Compute $\pi \leftarrow \mathsf{PoK.Sim}(\mathsf{pk}, c_1', c_2', c_3')$;
- Output $c' := (c_1', c_2', c_3', \pi)$;

Reveal Algorithm $\mathsf{Reveal}_{\mathsf{SK}}(c', m, s, \tau)$: (where $g_2 = g_1^\tau$)

- Compute $x' := x + m\tau$ and $y' := y - m$;
- Output $\mathsf{sk}' := (x', y')$;

Figure 11: Reveiver Non-committing Encryption

7 ZERO KNOWLEDGE FOR DKG PROTOCOL

7.1 Zero Knowledge for DKG Key Generation

Theorem 7.1. *Assume the DDH problem is hard. Let* $\sum_{k=1}^{\ell} s_{i,j,k} p^k = F_i(j) = s_{i,j}$, $\sum_{k=1}^{\ell} s'_{i,j,k} p^k = F'_i(j) = s'_{i,j}$, *with CRS* $\mathbb{G}$, μ, ρ, η, *the protocol described in Fig.*?? *is a honest verifier zero-knowledge argument of knowledge of* $\{a_{i,t}, a'_{i,t}\}_{t=0}^{t_2}$, $\{r_{i,j,k}, r'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell}$ *such that there exists two degree-*t_2 *polynomials:* $F_i(z) = \sum_{t=0}^{t_2} a_{i,t} z^t$, $F'_i(z) = \sum_{t=0}^{t_2} a'_{i,t} z^t$ *that:*

- $F_i(0) := z_i$, $F'_i(0) := z'_i$
- $\{\upsilon_{i,j,k} := \mu^{r_{i,j,k}}, \omega_{i,j,k} := \rho^{r_{i,j,k}}, \epsilon_{i,j,k} := \mu^{s_{i,j,k}} \sigma_{j,k}^{r_{i,j,k}}, \upsilon'_{i,j,k} := \mu^{r_{i,j,k}}, \omega'_{i,j,k} := \rho^{r_{i,j,k}}, \epsilon'_{i,j,k} := \mu^{s_{i,j,k}} \sigma_{j,k}^{r_{i,j,k}}\}_{j=1,k=1}^{M_2,\ell}$

see the following holds:

$$(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \upsilon_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot A = \mu^{e\cdot(\sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} r_{i,j,k}\cdot p^k)\cdot\lambda^{j-1})} \cdot \mu^f = \mu^{z_4}$$

$$(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \upsilon' {}_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot A' = \mu^{e\cdot(\sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} r'_{i,j,k}\cdot p^k)\cdot\lambda^{j-1})} \cdot \mu^{f'} = \mu^{z'_4}$$

$$(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot B = \rho^{e\cdot(\sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} r_{i,j,k}\cdot p^k)\cdot\lambda^{j-1})} \cdot \rho^f = \rho^{z_4}$$

$$(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega' {}_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot B' = \rho^{e\cdot(\sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} r'_{i,j,k}\cdot p^k)\cdot\lambda^{j-1})} \cdot \rho^{f'} = \rho^{z'_4}$$

Proof. **Completeness.** Following the protocol, we first observe $\Delta = \mu^{\sum_{j=1}^{M_2} s_{i,j}\cdot\lambda^{j-1}} \rho^{\sum_{j=1}^{M_2} s'_{i,j}\cdot\lambda^{j-1}} \cdot \eta^{\sum_{j=1}^{M_2}(\sum_{t=0}^{t_2} o_{i,t}\cdot j^t\cdot\lambda^{j-1})}$, we have $\mu^{z_1}\cdot\rho^{z_2}\cdot\eta^{z_3} = \mu^{b+(\sum_{j=1}^{M_2} s_{i,j}\cdot\lambda^{j-1})\cdot e}\cdot\rho^{c+(\sum_{j=1}^{M_2} s'_{i,j}\cdot\lambda^{j-1})\cdot e}\cdot\eta^{d+e\cdot(\sum_{j=1}^{M_2}(\sum_{t=0}^{t_2} o_{i,t}\cdot j^t)\cdot\lambda^{j-1})} = \Delta^e\cdot\mu^b\cdot\rho^c\cdot\eta^d = \Delta^e\cdot T$. Moreover, for $j \in [M_2], k \in [\ell]$, based on the ciphertexts $\upsilon_{i,j,k} := \mu^{r_{i,j,k}}$, $\omega_{i,j,k} := \rho^{r_{i,j,k}}$, $\epsilon_{i,j,k} := \mu^{s_{i,j,k}}\sigma_{j,k}^{r_{i,j,k}}$, $\upsilon'_{i,j,k} := \mu^{r'_{i,j,k}}$, $\omega'_{i,j,k} := \rho^{r'_{i,j,k}}$, $\epsilon'_{i,j,k} := \mu^{s'_{i,j,k}}\sigma_{j,k}^{r'_{i,j,k}}$, it is easy to

RNCE DKG

CRS: $\mathbb{G}$ and $\mu, \rho \in \mathbb{G} \setminus \{1\}$.

Designating Phase
In the δ-th round:

- Get $\mathcal{SC}_\delta := \{\mathsf{SC}_{\{\delta,1\}}, \ldots, \mathsf{SC}_{\{\delta,s_\delta\}}\}$ from $\mathcal{F}^n_{Sortion}$.
- For $\mathsf{SC}_{\{\delta,j\}} \in \mathcal{SC}_\delta$, where $j \in [s_\delta]$, do the followings:
  - Select $\mathsf{MC}_{\{\delta,j\}}$ based on locked stakes.
  - for $k = 1, \cdots, \ell$
    - Choose $m_{j,k}, n_{j,k} \leftarrow \mathbb{Z}_q$, compute $\sigma_{j,k} := \mu^{m_{j,k}} \rho^{n_{j,k}}$;
    - Set $epk_{j,k} := \sigma_{j,k}, esk_{j,k} := (m_{j,k}, n_{j,k})$, Compute $ct_{j,k} := Enc_{pk_j}(esk_{j,k})$
  - Post $\{epk_{j,k}, ct_j\}_{k=1}^{\ell}$ on sidechain.
- For $\mathsf{MC}_{\{\delta,j\}} \in \mathcal{MC}_\delta$, do the following:
  - Get $\{epk_{j,k}, ct_{j,k}\}_{k=1}^{\ell}$ from sidechain, compute $\{esk_{j,k}\}_{k=1}^{\ell}$ by decrypting $ct_{j,k}$ with secret key sk_j.

Key Generation Phase
In the *first* round, for all committee members $\{\mathsf{MC}_{\{1,i\}}\}_{i=1}^{M_1}$ in $\mathcal{MC}_1$, do the following:

- Randomly choose $z_i, x_i \leftarrow \mathbb{Z}_q$. Generate two degree-$t_{\delta-1}$ random polynomials over $\mathbb{Z}_q$, where $t_{\delta-1} := \lfloor \frac{N_{\delta-1}}{2} \rfloor + 1$, $a_{i,0} := z_i, a'_{i,0} := x_i$: $F_i(z) = a_{i,0} + a_{i,1}z + \cdots + a_{i,t_{\delta-1}} z^{t_{\delta-1}}$, $F'_i(z) = a'_{i,0} + a'_{i,1}z + \cdots + a'_{i,t_{\delta-1}} z^{t_{\delta-1}}$.
- for $j = 1, \cdots, N_{\delta-1}$, do the following:
  - Compute $\{s_{i,j} := F_i(j), s'_{i,j} := F'_i(j)\}_{j=1}^{M_1}$. Slice $s_{i,j}, s'_{i,j}$ into ℓ elements: $\{s_{i,j,0}, \cdots, s_{i,j,\ell}\}$ and $\{s'_{i,j,0}, \cdots, s'_{i,j,\ell}\}$, where $\sum_{k=1}^{\ell} s_{i,j,k} p^k = s_{i,j}$, $\sum_{k=1}^{\ell} s'_{i,j,k} p^k = s'_{i,j}$, p is a perfect power of 2.
  - for $k = 1, \cdots, \ell$
    - Choose $r_{i,j,k}, r'_{i,j,k} \leftarrow \mathbb{Z}_q$, compute $v_{i,j,k} := \mu^{r_{i,j,k}}$, $\omega_{i,j,k} := \rho^{r_{i,j,k}}$, $\epsilon_{i,j,k} := \mu^{s_{i,j,k}} \sigma_{j,k}^{r_{i,j,k}}$, $v'_{i,j,k} := \mu^{r'_{i,j,k}}$, $\omega'_{i,j,k} := \rho^{r'_{i,j,k}}$, $\epsilon'_{i,j,k} := \mu^{s'_{i,j,k}} \sigma_{j,k}^{r'_{i,j,k}}$, set $ct_{i,j,k} := (v_{i,j,k}, \omega_{i,j,k}, \epsilon_{i,j,k}, v'_{i,j,k}, \omega'_{i,j,k}, \epsilon'_{i,j,k})$;
- Compute $\sigma_i := \mu^{z_i} \cdot \rho^{x_i}$;. Generate NIZK proof π_i. (Cf. Fig.??). Choose a new pair of long-term keys (pk'_i, sk'_i), post $\{\sigma_i, \pi_i, pk'_i, \{ct_{i,j,k}\}_{j=1,k=1}^{N_{\delta-1},\ell}\}$ on sidechain, then erase all the protocol secrets he holds.

Maintenance Phase
For all committee members $\{\mathsf{MC}_{\{\alpha+1,f\}}\}_{f=1}^{M_{\alpha+1}}$ in $\mathcal{MC}_{\alpha+1}$, where $\alpha \geq 1$ do the following:

- Collect ciphertexts $\{ct_{j',f,k}\}_{j'=1,k=1}^{M_\alpha,\ell}$ and proofs $\{\pi_{j'}\}_{j'=1}^{M_\alpha}$ from $\mathcal{MC}_\alpha$. Construct a set QUAL_δ with size N_{QUAL_δ} which consists the seat number of $\mathcal{MC}_\alpha$ who provided valid proofs, ordered lexicographically by the public key values of the corresponding members, set corresponding ciphertexts as $\{ct_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\delta},\ell}$; /*Global Public Key can be extracted by $\prod_{i'=1}^{N_{\mathsf{QUAL}_\delta}} \sigma_{i'}$*/
- Compute $\eta_{i',f,k} := \epsilon_{i',f,k} / (v_{i',f,k}^{m_{f,k}} \cdot \omega_{i',f,k}^{n_{f,k}})$, $\eta'_{i',f,k} := \epsilon'_{i',f,k} / (v'^{m_{f,k}}_{i',f,k} \cdot \omega'^{n_{f,k}}_{i',f,k})$ and get $s_{i',f,k}, s'_{i',f,k}$ by an exhaustive search through $\mathbb{Z}_q$ to determine $\mu^{s_{i',f,k}} \stackrel{\text{def}}{==} \eta_{i',f,k}, \mu^{s'_{i',f,k}} \stackrel{\text{def}}{==} \eta'_{i',f,k}$.
- Compute $s_f := \begin{cases} \sum_{i'=1}^{N_{\mathsf{QUAL}_\delta}} \sum_{k=1}^{\ell} s_{i',f,k} \cdot p^k & \alpha = 1 \\ \sum_{i'=1}^{t_{\mathsf{QUAL}_\delta}} (\sum_{k=1}^{\ell} s_{i',f,k} \cdot p^k) \gamma_{i',f} & \alpha > 1 \end{cases}$, $s'_f := \begin{cases} \sum_{i'=1}^{N_{\mathsf{QUAL}_\delta}} \sum_{k=1}^{\ell} s'_{i',f,k} \cdot p^k & \alpha = 1 \\ \sum_{i'=1}^{t_{\mathsf{QUAL}_\delta}} (\sum_{k=1}^{\ell} s'_{i',f,k} \cdot p^k) \gamma'_{i',f} & \alpha > 1 \end{cases}$
- Choose two $t_{\alpha+2}$ degree random polynomials, where $t_{\alpha+2} := \lfloor \frac{M_{\alpha+2}}{2} \rfloor + 1$, $a_{f,0} = s_f, a'_{f,0} = s'_f$: $G_f(z) = a_{f,0} + a_{f,1}z + \cdots + a_{f,t_{\alpha+2}} z^{t_{\alpha+2}}$, $G'_f(z) = a'_{f,0} + a'_{f,1}z + \cdots + a'_{f,t_{\alpha+2}} z^{t_{\alpha+2}}$. Compute $\{g_{f,m'} := G_f(m'), g'_{f,m'} := G'_f(m')\}_{m'=1}^{M_{\alpha+2}}$.
- for $m' = 1, \cdots, M_{\alpha+2}$, do the following:
  - Divide $g_{f,m'}$ and $g'_{f,m'}$ into ℓ smaller elements individually: $\{g_{f,m',k}, g'_{f,m',k}\}_{k=1}^{\ell}$, where $\sum_{k=1}^{\ell} g_{f,m',k} p^k = g_{f,m'}$, $\sum_{k=1}^{\ell} g_{f,m',k} p^k = g'_{f,m'}$.
  - for $k = 1, \cdots, \ell$
    - Choose $r_{f,m',k}, r'_{f,m',k} \leftarrow \mathbb{Z}_q$, compute $v_{f,m',k} := \mu^{r_{f,m',k}}$, $\omega_{f,m',k} := \rho^{r_{f,m',k}}$, $\epsilon_{f,m',k} := \mu^{g_{f,m',k}} \sigma_{m',k}^{r_{f,m',k}}$, $v'_{f,m',k} := \mu^{r'_{f,m',k}}$, $\omega'_{f,m',k} := \rho^{r'_{f,m',k}}$, $\epsilon'_{f,m',k} := \mu^{g'_{f,m',k}} \sigma_{m',k}^{r'_{f,m',k}}$, set $ct_{f,m',k} := (v_{f,m',k}, \omega_{f,m',k}, \epsilon_{f,m',k}, v'_{f,m',k}, \omega'_{f,m',k}, \epsilon'_{f,m',k})$
- Generate NIZK proof π_f. Cf. Fig.??. Choose a new pair of long-term keys (pk'_f, sk'_f) for himself, post $\{\pi_f, pk'_f, \{ct_{f,m',k}\}_{m'=1,k=1}^{M_{\alpha+2},\ell}\}$ on sidechain, then erase all the protocol secrets he holds.

Figure 12: RNCE DKG

The algorithm that maps $i \in [0, n-1]$ to $\mathbf{e}_i^{(n)}$

Input: index $i = (i_1, \ldots, i_{\log n}) \in \{0,1\}^{\log n}$
Output: unit vector $\mathbf{e}_i^{(n)} = (e_{i,0}, \ldots, e_{i,n-1}) \in \{0,1\}^n$
1. For $\ell \in [\log n]$, set $b_{\ell,0} := 1 - i_\ell$ and $b_{\ell,1} := i_\ell$;
2. For $j \in [0, n-1]$, set $e_{i,j} := \prod_{\ell=1}^{\log n} b_{\ell, j_\ell}$, where $j_1, \ldots, j_{\log n}$ is binary representation of j;
3. Return $\mathbf{e}_i^{(n)} = (e_{i,0}, \ldots, e_{i,n-1})$;

Figure 13: The algorithm that maps $i \in [0, n-1]$ to $\mathbf{e}_i^{(n)}$

Unit vector ZK argument

CRS: the commitment key ck
Statement: A set of public keys $\{\mathsf{pk}_i\}_{i\in[0,n-1]}$ and the ciphertexts $C_0 := \mathsf{Enc}_{\mathsf{pk}_0}(e_{i,0}; r_0), \ldots, C_{n-1} := \mathsf{Enc}_{\mathsf{pk}_{n-1}}(e_{i,n-1}; r_{n-1})$
Witness: the unit vector $\mathbf{e}_i^{(n)} \in \{0,1\}^n$ and the randomness $r_0, \ldots, r_{n-1} \in \mathbb{Z}_p$
Protocol:

- The prover P, for $\ell = 1, \ldots, \log n$, do:
  - Pick random $\alpha_\ell, \beta_\ell, \gamma_\ell, \delta_\ell \leftarrow \mathbb{Z}_p$;
  - Compute $I_\ell := \mathsf{Com}_{\mathsf{ck}}(i_\ell; \alpha_\ell)$, $B_\ell := \mathsf{Com}_{\mathsf{ck}}(\beta_\ell; \gamma_\ell)$ and $A_\ell := \mathsf{Com}_{\mathsf{ck}}(i_\ell \cdot \beta_\ell; \delta_\ell)$;
- $P \to V$: $\{I_\ell, B_\ell, A_\ell\}_{\ell=1}^{\log n}$;
- $V \to P$: Random $y \leftarrow \{0,1\}^\lambda$;
- The prover P for $\ell = 0, \ldots, \log n - 1$, do:
  - Pick random $R_\ell \leftarrow \mathbb{Z}_p$ and compute $D_\ell := \mathsf{Enc}_{\mathsf{pk}}\Big(\sum_{j=0}^{n-1}(p_{j,\ell} \cdot y^j); R_\ell\Big)$
- $P \to V$: $\{D_\ell\}_{\ell=0}^{\log n-1}$;
- $V \to P$: Random $x \leftarrow \{0,1\}^\lambda$;
- The prover P does the following:
  - Compute $R := \sum_{j=0}^{n-1}(r_j \cdot x^{\log n} \cdot y^j) + \sum_{\ell=0}^{\log n-1}(R_\ell \cdot x^\ell)$;
  - For $\ell = 1, \ldots, \log n$, compute $z_\ell := i_\ell \cdot x + \beta_\ell$, $w_\ell := \alpha_\ell \cdot x + \gamma_\ell$, and $v_\ell := \alpha_\ell(x - z_\ell) + \delta_\ell$;
- $P \to V$: R and $\{z_\ell, w_\ell, v_\ell\}_{\ell=1}^{\log n}$

Verification:

- Check the followings:
- For $\ell = 1, \ldots, \log n$, do:
  - $(I_\ell)^x \cdot B_\ell = \mathsf{Com}_{\mathsf{ck}}(z_\ell; w_\ell)$
  - $(I_\ell)^{x - z_\ell} \cdot A_\ell = \mathsf{Com}_{\mathsf{ck}}(0; v_\ell)$
- $\prod_{j=0}^{n-1}\Big((C_j)^{x^{\log n}} \cdot \mathsf{Enc}_{\mathsf{pk}}(-\prod_{\ell=1}^{\log n} z_{\ell,j_\ell}; 0)\Big)^{y^j} \cdot \prod_{\ell=0}^{\log n-1}(D_\ell)^{x^\ell} = \mathsf{Enc}_{\mathsf{pk}}(0; R)$, where $z_{j,1} = z_j$ and $z_{j,0} = x - z_j$.

Figure 14: Unit vector ZK argument

$$(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot C$$

$$= \mu^{e\cdot(\sum_{j=1}^{M_2} s_{i,j,}\cdot\lambda^{j-1})} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{r_{i,j,k}\cdot p^k})^{\lambda^{j-1}})^e \cdot \mu^b \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell}(\sigma_{j,k})^{f_{j,k}})^{\lambda^{j-1}})$$

$$= \mu^{z_1} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{z_{i,j,k}})^{\lambda^{j-1}})$$

$$(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^e \cdot C'$$

$$= \mu^{e\cdot(\sum_{j=1}^{M_2} s'_{i,j,}\cdot\lambda^{j-1})} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{r'_{i,j,k}\cdot p^k})^{\lambda^{j-1}})^e \cdot \mu^c \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell}(\sigma'_{j,k})^{f'_{j,k}})^{\lambda^{j-1}})$$

$$= \mu^{z'_1} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{z'_{i,j,k}})^{\lambda^{j-1}})$$

Soundness. We assume that there exists a PPT extractor Ext which rewinds the protocol to the first challenge phase (λ) and runs it with fresh challenges until it has M_2 acceptable arguments. More specifically, each time $\alpha \in [M_2]$, Ext gives new challenge λ_α and the same challenge e_1, then Ext can get:

$$z_1^{(\alpha,1)} = b + (\sum_{j=1}^{M_2} s_{i,j} \cdot \lambda_\alpha^{j-1}) \cdot e_1, z_2^{(\alpha,1)} = c + (\sum_{j=1}^{M_2} s'_{i,j} \cdot \lambda_\alpha^{j-1}) \cdot e_1$$

$$\Big[z_{i,j,k}^{(\alpha,1)} = f_{j,k} + r_{i,j,k} \cdot p^k \cdot e_1, z'_{i,j,k}{}^{(\alpha,1)} = f'_{j,k} + r'_{i,j,k} \cdot p^k \cdot e_1\Big]_{j=1,k=1}^{M_2,\ell}$$

RNCE DKG Key Generation ZK argument

CRS: $\mathbb{G}$, μ, ρ, $\eta \in \mathbb{G} \setminus \{1\}$.
Statement: $\{ct_{i,j,k} = (v_{i,j,k}, \omega_{i,j,k}, \epsilon_{i,j,k}, v'_{i,j,k}, \omega'_{i,j,k}, \epsilon'_{i,j,k}), \sigma_{j,k}\}_{j=1,k=1}^{M_2,\ell}$.
Witness: $\{a_{i,t}, a'_{i,t}\}_{t=0}^{t_2}$, $\{r_{i,j,k}, r'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell}$.
Protocol:

- The prover $\mathcal{P}$ randomly selects $\{o_{i,t}\}_{t=0}^{t_2} \leftarrow \mathbb{Z}_q$, and computes $\Delta_{i,t} := \mu^{a_{i,t}} \cdot \rho^{a'_{i,t}} \cdot \eta^{o_{i,t}}$ for $t = 0, \cdots, t_2$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle \Delta_{i,t} \rangle_{t=0}^{t_2}$;
- $\mathcal{V} \rightarrow \mathcal{P} : \lambda \leftarrow \mathbb{Z}_q$; /* For NIZK, set $\lambda \leftarrow \mathsf{hash}(\langle \Delta_{i,t} \rangle_{t=0}^{t_2})$ */
- $\mathcal{P}$ does the followings:
  - Choose $b, c, d, \{f_{j,k}, f'_{j,k}\}_{j=1,k=1}^{M_2,\ell} \leftarrow \mathbb{Z}_q$;
  - Compute $f := \sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} f_{j,k}) \cdot \lambda^{j-1}$, $f' := \sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} f'_{j,k}) \cdot \lambda^{j-1}$;
  - Compute $A := \mu^f$, $B := \rho^f$, $C := \mu^b \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell}(\sigma_{j,k})^{f_{j,k}})^{\lambda^{j-1}})$, $A' := \mu^{f'}$, $B' := \rho^{f'}$, $C' := \mu^c \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell}(\sigma_{j,k})^{f'_{j,k}})^{\lambda^{j-1}})$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle A, B, C, A', B', C' \rangle$;
- $\mathcal{V} \rightarrow \mathcal{P} : e \leftarrow \mathbb{Z}_q$; /* For NIZK, set $e \leftarrow \mathsf{hash}(\langle A, B, C, A', B', C', \lambda \rangle)$ */
- $\mathcal{P}$ computes the following:
  - $z_1 := b + (\sum_{j=1}^{M_2} s_{i,j} \cdot \lambda^{j-1}) \cdot e$, $z'_1 := c + (\sum_{j=1}^{M_2} s'_{i,j} \cdot \lambda^{j-1}) \cdot e$, $z_2 := d + e \cdot (\sum_{j=1}^{M_2}(\sum_{t=0}^{t_2} o_{i,t} \cdot j^t) \cdot \lambda^{j-1})$;
  - $T := \mu^b \cdot \rho^c \cdot \eta^d$;
  - $z_{i,j,k} := f_{j,k} + r_{i,j,k} \cdot p^k \cdot e$, $z'_{i,j,k} := f'_{j,k} + r'_{i,j,k} \cdot p^k \cdot e$ for $j \in [M_2]$, $k \in [\ell]$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle T, z_1, z'_1, z_2, \langle z_{i,j,k}, z'_{i,j,k} \rangle_{j=1,k=1}^{M_2,\ell} \rangle$

Verification:
$\mathcal{V}$ computes the followings:

- /* For NIZK, set $\lambda \leftarrow \mathsf{hash}(\langle \Delta_{i,t} \rangle_{t=0}^{t_2})$ */
- /* For NIZK, set $e \leftarrow \mathsf{hash}(\langle A, B, C, A', B', C', \lambda \rangle)$ */
- $\Delta := \prod_{j=1}^{M_2}(\prod_{t=0}^{t_2}(\Delta_{i,t})^{j^t})^{\lambda^{j-1}})$;
- $z_3 := \sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} z_{i,j,k}) \cdot \lambda^{j-1}$, $z'_3 := \sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} z'_{i,j,k}) \cdot \lambda^{j-1}$

$\mathcal{V}$ check the followings:

- $\mu^{z_1} \cdot \rho^{z'_1} \cdot \eta^{z_2} = \Delta^e \cdot T$;
- $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} v_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot A = \mu^{z_3}$, $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} v'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^e \cdot A' = \mu^{z'_3}$;
- $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot B = \rho^{z_3}$, $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^e \cdot B' = \rho^{z'_3}$;
- $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^e \cdot C = \mu^{z_1} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{z_{i,j,k}})^{\lambda^{j-1}})$, $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^e \cdot C' = \mu^{z'_1} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{z'_{i,j,k}})^{\lambda^{j-1}})$;

Figure 15: RNCE DKG Key Generation ZK argument

Then, Ext rewinds the protocol feeding new challenge e_2 and gets:

$$z_1^{(\alpha,2)} = b + (\sum_{j=1}^{M_2} s_{i,j} \cdot \lambda_\alpha^{j-1}) \cdot e_2, z_2^{(\alpha,2)} = c + (\sum_{j=1}^{M_2} s'_{i,j} \cdot \lambda_\alpha^{j-1}) \cdot e_2$$

$$\left[z_{i,j,k}^{(\alpha,2)} = f_{j,k} + r_{i,j,k} \cdot p^k \cdot e_2, {z'_{i,j,k}}^{(\alpha,2)} = f'_{j,k} + r'_{i,j,k} \cdot p^k \cdot e_2\right]_{j=1,k=1}^{M_2,\ell}$$

By computing $(z_1^{(\alpha,1)} - z_1^{(\alpha,2)})/(e_1 - e_2)$, $(z_2^{(\alpha,1)} - z_2^{(\alpha,2)})/(e_1 - e_2)$, Ext will get

$$\Gamma_{\alpha,1} = \left[s_{i,1} \cdots s_{i,M_2}\right] \cdot \begin{bmatrix} 1 \\ \lambda_\alpha \\ \cdots \\ \lambda_\alpha^{(M_2-1)} \end{bmatrix} \quad \Gamma_{\alpha,2} = \left[s'_{i,1} \cdots s'_{i,M_2}\right] \cdot \begin{bmatrix} 1 \\ \lambda_\alpha \\ \cdots \\ \lambda_\alpha^{(M_2-1)} \end{bmatrix}$$

There is overwhelming probability that we have transcripts with M_2 different challenges, these challenges give us a $(M_2) \times (M_2)$ invertible transposed Vandermonde polynomial matrix:

$$\Lambda = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_{M_2} \\ \vdots & \vdots & \ddots & \vdots \\ (\lambda_1)^{(M_2-1)} & (\lambda_2)^{(M_2-1)} & \cdots & \lambda_{M_2}^{(M_2-1)} \end{pmatrix}$$

We denote Γ_1 as the $1 \times (M_2)$ matrixes which consists of columns $\Gamma_{1,1}, \cdots, \Gamma_{(M_2),1}, \Gamma_2$ as $\Gamma_{1,2}, \cdots, \Gamma_{(M_2),2}$. Then we have $\left[s_{i,1} \cdots s_{i,M_2}\right] \cdot \Lambda = \Gamma_1$, $\left[s'_{i,1} \cdots s'_{i,M_2}\right] \cdot \Lambda = \Gamma_2$, Ext can compute $\left[s_{i,1} \cdots s_{i,M_2}\right] = \Gamma_1 \cdot \Lambda^{-1}$, $\left[s'_{i,1} \cdots s'_{i,M_2}\right] = \Gamma_2 \cdot \Lambda^{-1}$ as Λ is invertible and $\{\lambda_\alpha\}_{\alpha \in [M_2]}$ are distinct. Next, taking t_2 numbers from $\{s_{i,j}, s'_{i,j}\}_{j=1}^{M_2}$ individually, Ext can reconstruct polynomials $F_{i,j}$ and $F'_{i,j}$ for $j = 1, \cdots, M_2$. Hence witness $\{a_{i,t}, a'_{i,t}\}_{t=0}^{t_2}$ can be extracted.

In addition, for $j \in [M_2]$, $k \in [\ell]$, Ext computes $(z_{i,j,k}^{(\alpha,1)} - z_{i,j,k}^{(\alpha,2)})/(e_1 - e_2)$, $({z'_{i,j,k}}^{(\alpha,1)} - {z'_{i,j,k}}^{(\alpha,2)})/(e_1 - e_2)$, and get $r_{i,j,k} \cdot p^k$, $r'_{i,j,k} \cdot$

RNCE DKG Key Generation ZK argument

CRS: $\mathbb{G}$, $\langle \mu, \rho, \eta, \{g_v\}_{v=1}^{\sqrt{M_2 \cdot \ell}} \rangle \in \mathbb{G} \setminus \{1\}$.

Statement: $\{ct_{i,j,k} = (v_{i,j,k}, \omega_{i,j,k}, \epsilon_{i,j,k}, v'_{i,j,k}, \omega'_{i,j,k}, \epsilon'_{i,j,k}), \sigma_{j,k}\}_{j=1,k=1}^{M_2,\ell}$.

Witness: $\{a_{i,t}, a'_{i,t}\}_{t=0}^{t_2}$, $\{r_{i,j,k}, r'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell}$.

Protocol:

- The prover $\mathcal{P}$ randomly selects $\{o_{i,t}\}_{t=0}^{t_2} \leftarrow \mathbb{Z}_q$, and computes $\Delta_{i,t} := \mu^{a_{i,t}} \cdot \rho^{a'_{i,t}} \cdot \eta^{o_{i,t}}$ for $t = 0, \cdots, t_2$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle \Delta_{i,t} \rangle_{t=0}^{t_2}$;
- $\mathcal{V} \rightarrow \mathcal{P} : \lambda \leftarrow \mathbb{Z}_q$; /* For NIZK, set $\lambda \leftarrow \mathsf{hash}(\langle \Delta_{i,t} \rangle_{t=0}^{t_2})$ */
- $\mathcal{P}$ does the followings:
  - Choose $b, c, d, \{f_{j,k}, f'_{j,k}\}_{j=1,k=1}^{M_2,\ell} \leftarrow \mathbb{Z}_q$;
  - Compute $f := \sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} f_{j,k})$, $f' := \sum_{j=1}^{M_2}(\sum_{k=1}^{\ell} f'_{j,k})$;
  - Compute $A := \mu^f$, $B := \rho^f$, $C := \mu^b \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell}(\sigma_{j,k})^{f_{j,k}}))$, $A' := \mu^{f'}$, $B' := \rho^{f'}$, $C' := \mu^c \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell}(\sigma_{j,k})^{f'_{j,k}}))$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle A, B, C, A', B', C' \rangle$;
- $\mathcal{V} \rightarrow \mathcal{P} : e \leftarrow \mathbb{Z}_q$; /* For NIZK, set $e \leftarrow \mathsf{hash}(\langle A, B, C, A', B', C', \lambda \rangle)$ */
- $\mathcal{P}$ computes the following:
  - $z_1 := b + (\sum_{j=1}^{M_2} s_{i,j} \cdot \lambda^{j-1}) \cdot e$, $z'_1 := c + (\sum_{j=1}^{M_2} s'_{i,j} \cdot \lambda^{j-1}) \cdot e$, $z_2 := d + e \cdot (\sum_{j=1}^{M_2}(\sum_{t=0}^{t_2} o_{i,t} \cdot j^t \cdot \lambda^{j-1}))$;
  - $T := \mu^b \cdot \rho^c \cdot \eta^d$;
  - $z_{i,j,k} := f_{j,k} + r_{i,j,k} \cdot p^k \cdot e$, $z'_{i,j,k} := f'_{j,k} + r'_{i,j,k} \cdot p^k \cdot e$ for $j \in [M_2]$, $k \in [\ell]$;
  - Get $\{z_{i,v,w}, z'_{i,v,w}\}_{v=1,w=1}^{\sqrt{M_2 \cdot \ell}, \sqrt{M_2 \cdot \ell}}$ from $\textsc{ConvMat}(\{z_{i,j,k}\}_{j=1,k=1}^{M_2,\ell})$ and $\textsc{ConvMat}(\{z'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell})$, get $\{\sigma_{v,w}\}_{v=1,w=1}^{\sqrt{M_2 \cdot \ell}, \sqrt{M_2 \cdot \ell}}$ from $\textsc{ConvMat}(\{\sigma_{j,k}\}_{j=1,k=1}^{M_2,\ell})$;
  - $Q_v := \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g_v^{z_{i,v,w}}$, $Q'_v := \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g_v^{z'_{i,v,w}}$ for $v \in [\sqrt{M_2 \cdot \ell}]$;
  - Select $\{y_{v,w}, y'_{v,w}, t_{v,w}, t'_{v,w}, x_{v,w}, x'_{v,w}\}_{v=1,w=1}^{\sqrt{M_2 \cdot \ell}, \sqrt{M_2 \cdot \ell}} \leftarrow \mathbb{Z}_q$;
  - $G_v := \sum_{w=1}^{\sqrt{M_2 \cdot \ell}} t_{v,w}$, $G'_v := \sum_{w=1}^{\sqrt{M_2 \cdot \ell}} t'_{v,w}$ for $v \in [\sqrt{M_2 \cdot \ell}]$;
  - $D := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g_v^{y_{v,w}})$, $D' := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g_v^{y'_{v,w}})$;
  - $E := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g^{y_{v,w}} h^{t_{v,w}})$, $E' := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g^{y'_{v,w}} h^{t'_{v,w}})$;
  - $F := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g^{x_{v,w}})$, $F' := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g^{x'_{v,w}})$;
  - $H := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} \sigma_{v,w}^{x_{v,w}})$, $H' := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(\prod_{w=1}^{\sqrt{M_2 \cdot \ell}} \sigma_{v,w}^{x'_{v,w}})$;
  - $K_v := \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} \sigma_{v,w}^{z_{i,v,w}}$, $K'_v := \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} \sigma_{v,w}^{z'_{i,v,w}}$ for $v \in [\sqrt{M_2 \cdot \ell}]$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle T, z_1, z'_1, z_2, \langle Q_v, G_v, G'_v, K_v, K'_v \rangle_{v=1}^{\sqrt{M_2 \cdot \ell}}, D, D', E, E', F, F', H, H' \rangle$
- $\mathcal{V} \rightarrow \mathcal{P} : e' \leftarrow \mathbb{Z}_q$; /* For NIZK, set $e' \leftarrow \mathsf{hash}(\langle T, z_1, z'_1, z_2, \langle Q_v, G_v, G'_v, K_v, K'_v \rangle_{v=1}^{\sqrt{M_2 \cdot \ell}}, D, D', E, E', F, F', H, H' \rangle)$ */
- $\mathcal{P}$ computes the following:
  - $Y_v := \sum_{w=1}^{\sqrt{M_2 \cdot \ell}}(y_{v,w} + z_{i,v,w} \cdot e')$, $Y'_v := \sum_{w=1}^{\sqrt{M_2 \cdot \ell}}(y'_{v,w} + z'_{i,v,w} \cdot e')$ for $v \in [\sqrt{M_2 \cdot \ell}]$;
  - $X_{v,w} := x_{v,w} + z_{i,v,w} \cdot e'$, $X'_{v,w} := x'_{v,w} + z'_{i,v,w} \cdot e'$ for $w \in [\sqrt{M_2 \cdot \ell}]$, $v \in [\sqrt{M_2 \cdot \ell}]$;
  - $Z_1 := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}} \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g_v^{X_{v,w}}$, $Z'_1 := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}} \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g_v^{X'_{v,w}}$
  - $z_3 := \sum_{v=1}^{\sqrt{M_2 \cdot \ell}}(\sum_{w=1}^{\sqrt{M_2 \cdot \ell}} z_{i,v,w})$, $z'_3 := \sum_{v=1}^{\sqrt{M_2 \cdot \ell}}(\sum_{w=1}^{\sqrt{M_2 \cdot \ell}} z'_{i,v,w})$
  - $z_4 := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}} \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} \sigma_{v,w}^{X_{v,w}}$, $z'_3 := \prod_{v=1}^{\sqrt{M_2 \cdot \ell}} \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} \sigma_{v,w}^{X'_{v,w}}$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle \langle Y_v, Y'_v \rangle_{v=1}^{\sqrt{M_2 \cdot \ell}}, Z_1, Z'_1, z_3, z'_3, z_4, z'_3, \rangle$

Verification:

$\mathcal{V}$ computes the followings:

- /* For NIZK, set $\lambda \leftarrow \mathsf{hash}(\langle \Delta_{i,t} \rangle_{t=0}^{t_2})$ */
- /* For NIZK, set $e \leftarrow \mathsf{hash}(\langle A, B, C, A', B', C', \lambda \rangle)$ */
- /* For NIZK, set $e' \leftarrow \mathsf{hash}(\langle T, z_1, z'_1, z_2, \langle Q_v, G_v, G'_v, K_v, K'_v \rangle_{v=1}^{\sqrt{M_2 \cdot \ell}}, D, D', E, E', F, F', H, H' \rangle)$ */
- $\Delta := \prod_{j=1}^{M_2}(\prod_{t=0}^{t_2}(\Delta_{i,t})^{j^t})^{\lambda^{j-1}})$;

$\mathcal{V}$ check the followings:

- $\prod_{v=1}^{\sqrt{M_2 \cdot \ell}} g_v^{Y_v} = D \cdot \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(Q_v^{e'})$, $\prod_{v=1}^{\sqrt{M_2 \cdot \ell}} g_v^{Y'_v} = D' \cdot \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(Q'^{e'}_v)$;
- $E \cdot (\prod_{v=1}^{\sqrt{M_2 \cdot \ell}} \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g^{z_{i,v,w}})^{e'} = \prod_{v=1}^{\sqrt{M_2 \cdot \ell}} g^{Y_v} h^{G_v}$, $E' \cdot (\prod_{v=1}^{\sqrt{M_2 \cdot \ell}} \prod_{w=1}^{\sqrt{M_2 \cdot \ell}} g^{z'_{i,v,w}})^{e'} = \prod_{v=1}^{\sqrt{M_2 \cdot \ell}} g^{Y'_v} h^{G'_v}$;
- $Z_1 = \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(Q_v^{e'}) \cdot F$, $Z'_1 = \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}((Q'_v)^{e'}) \cdot F'$;
- $z_4 = \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}(K_v^{e'}) \cdot H$, $z'_3 = \prod_{v=1}^{\sqrt{M_2 \cdot \ell}}((K'v)^{e'}) \cdot H'$;
- $\mu^{z_1} \cdot \rho^{z'_1} \cdot \eta^{z_2} = \Delta^e \cdot T$;
- $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} v_{i,j,k}^{p^k})^e \cdot A = \mu^{z_3}$, $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} v'^{p^k}_{i,j,k})^e \cdot A' = \mu^{z'_3}$;
- $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega_{i,j,k}^{p^k})^e \cdot B = \rho^{z_3}$, $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega'^{p^k}_{i,j,k})^e \cdot B' = \rho^{z'_3}$;
- $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon_{i,j,k}^{p^k})^{\lambda^{j-1}})^e \cdot C = \mu^{z_1} \cdot (\prod_{v=1}^{\sqrt{M_2 \cdot \ell}} K_v)$, $(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon'^{p^k}_{i,j,k})^{\lambda^{j-1}})^e \cdot C' = \mu^{z'_1} \cdot (\prod_{v=1}^{\sqrt{M_2 \cdot \ell}} K'_v)$;

Figure 16: RNCE DKG Key Generation ZK argument

p^k. Then Ext can extract $\{r_{i,j,k}, r'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell}$ by slicing $r_{i,j,k} \cdot p^k, r'_{i,j,k} \cdot p^k$ individually into ℓ pieces.

Zero Knowledge. In terms of special honest verifier zero-knowledge, we construct a simulator Sim that takes the challenges $\{\lambda, e\} \leftarrow \mathbb{Z}_q$ and statement $\{ct_{i,j,k}, \sigma_{j,k}\}_{j=1,k=1}^{M_2,\ell}$ as inputs, it outputs a simulated transcript the distribution of which is indistinguishable from the real one.

More specifically, Sim first picks $z_1, z_2, z_3, \{a_{i,t}, a'_{i,t}, o_{i,t}\}_{t=0}^{t_2} \leftarrow \mathbb{Z}_q$, computes Δ and $\{\Delta_{i,t}\}_{t=0}^{t_2}$ according to the protocol description, and computes $T := \mu^{z_1} \cdot \rho^{z_2} \cdot \eta^{z_3} \cdot \Delta^{-e}$. Next, Sim picks $\{z_{i,j,k}, z'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell} \leftarrow \mathbb{Z}_q$, computes z_4 as described in protocol, and computes $A := \mu^{z_4} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} v_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^{-e}$, $A' := \mu^{z'_4} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} v'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^{-e}$, $B = \rho^{z_4} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^{-e}$, $B' = \rho^{z'_4} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \omega'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^{-e}$, $C = \mu^{z_1} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{z_{i,j,k}}))\cdot(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon_{i,j,k}^{p^k})^{\cdot\lambda^{j-1}})^{-e}$, $C' = \mu^{z_2} \cdot (\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \sigma_{j,k}^{z'_{i,j,k}}))\cdot(\prod_{j=1}^{M_2}(\prod_{k=1}^{\ell} \epsilon'^{p^k}_{i,j,k})^{\cdot\lambda^{j-1}})^{-e}$.

After that, Sim outputs the simulated transcripts:

$$\langle T, z_1, z_2, z_3, \langle z_{i,j,k}, z'_{i,j,k}\rangle_{j=1,k=1}^{M_2,\ell}, \langle \Delta_{i,t}\rangle_{t=0}^{t_2}, A, B, C, A', B', C'\rangle$$

Since $b, c, d, \{f_{i,j,k}, f'_{i,j,k}\}_{j=1,k=1}^{M_2,\ell}$ are uniformly random in a real argument, the distribution of simulated $z_1, z_2, z_3, \langle z_{i,j,k}, z'_{i,j,k}\rangle_{j=1,k=1}^{M_2,\ell}$ is identical to the distribution of them in a real argument. Similarly, T are uniquely determined for fixed elements from group G. Furthermore, $A, B, C, A', B', C', \{\Delta_{i,t}\}_{t=0}^{t_2}$ are all random group elements. Therefore, simulated transcripts has the same distribution as real transcripts in a real argument. □

7.2 Zero Knowledge for DKG Maintaining Key

Theorem 7.2. *Assume the DDH problem is hard. When $\alpha = 1$, let $s_f = \sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} \sum_{k=1}^{\ell} s_{i',f,k} \cdot p^k$, $s'_f = \sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} \sum_{k=1}^{\ell} s'_{i',f,k} \cdot p^k$; when $\alpha > 1$, let $s_f = \sum_{i'=1}^{t_{\mathsf{QUAL}_\alpha}} (\sum_{k=1}^{\ell} s_{i',f,k} \cdot p^k)\gamma_{i',f}$, $s'_f = \sum_{i'=1}^{t_{\mathsf{QUAL}_\alpha}} (\sum_{k=1}^{\ell} s'_{i',f,k} \cdot p^k)\gamma'_{i',f}$, with CRS $\mathbb{G}, \mu, \rho, \eta$, the protocol described in Fig.?? is a honest verifier zero-knowledge argument of knowledge of $\{c_{f,t}, c'_{f,t}\}_{t=0}^{t_{\alpha+2}}, \{r_{f,m',k}, r'_{f,m',k}\}_{m'=1,k=1}^{M_{\alpha+2},\ell}, \{m_{f,k}, n_{f,k}\}_{k=1}^{\ell}$ such that there exists two degree-t_2 polynomials: $G_f(z) = \sum_{t=0}^{t_{\alpha+2}} c_{i,t} z^t$, $G'_f(z) = \sum_{t=0}^{t_{\alpha+2}} c'_{i,t} z^t$ that:*

- $G_i(0) := s_f$, $G'_i(0) := s'_f$
- $v_{f,m',k} := \mu^{r_{f,m',k}}, \omega_{f,m',k} := \rho^{r_{f,m',k}}, \epsilon_{f,m',k} := \mu^{g_{f,m',k}} \sigma_{m',k}^{r_{f,m',k}}$, $v'_{f,m',k} := \mu^{r'_{f,m',k}}, \omega'_{f,m',k} := \rho^{r'_{f,m',k}}, \epsilon'_{f,m',k} := \mu^{g'_{f,m',k}} \sigma_{m',k}^{r'_{f,m',k}}$ *for $m' \in [M_{\alpha+2}], k \in [\ell]$*
- $Dec(\{ct_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell}) = s_f$

Proof. For the proof, we only show the correctness of encryption. As for the proof of resharing secret key, it's similar to the poof in ??.

Completeness.

Completeness.

Completeness.

Completeness.

Completeness.

For $i' = 1, \cdots, N_{\mathsf{QUAL}_\alpha}$, we first have $D_{i'} = \prod_{k=1}^{\ell}((\epsilon_{i',f,k}/(v_{i',f,k}^{m_{f,k}} \cdot \omega_{i',f,k}^{n_{f,k}}))\cdot h^{d_{i',f,k}})^{p^k} = (\prod_{k=1}^{\ell}((\epsilon_{i',f,k}/(v_{i',f,k}^{m_{f,k}} \cdot \omega_{i',f,k}^{n_{f,k}}))^{p^k}) \cdot h^{\sum_{k=1}^{\ell} d_{i',f,k}^{p^k}}$, $P_{i'} = \prod_{k=1}^{\ell}((\epsilon_{i',f,k}/(v_{i',f,k}^{m'_{f,k}} \cdot \omega_{i',f,k}^{n'_{f,k}}))\cdot h^{d'_{i',f,k}})^{p^k} = (\prod_{k=1}^{\ell}((\epsilon_{i',f,k}/(v_{i',f,k}^{m'_{f,k}} \cdot \omega_{i',f,k}^{n'_{f,k}}))^{p^k}) \cdot h^{\sum_{k=1}^{\ell} d'^{p^k}_{i',f,k}}$. Then we can get

$$\begin{aligned} P_{i'} \cdot (D_{i'})^e &= \prod_{k=1}^{\ell}(((\epsilon_{i',f,k})^{e+1}/(v_{i',f,k}^{m'_{f,k}+e\cdot m_{f,k}} \cdot \omega_{i',f,k}^{n'_{f,k}+e\cdot n_{f,k}}))^{p^k} \\ &\quad \cdot (h^{\sum_{k=1}^{\ell}(d'_{i',f,k}+e\cdot d_{i',f,k})\cdot p^k}) \\ &= \prod_{k=1}^{\ell}(\epsilon_{i',f,k})^{e+1}/(v_{i',f,k}^{z_{4,f,k}} \cdot \omega_{i',f,k}^{z_{5,f,k}})^{p^k}) \cdot h^{z_{6,i',f}} \end{aligned}$$

Similarly, we can see that $P'_{i'} \cdot (D'_{i'})^e = \prod_{k=1}^{\ell}(\epsilon'_{i',f,k})^{e+1}/(v'^{z'_{4,f,k}}_{i',f,k} \cdot \omega'^{z'_{5,f,k}}_{i',f,k})^{p^k}) \cdot h^{z^*_{6,i',f}}$ also holds.

Next, the ephemeral public key of $\mathsf{MC}_{\{\alpha+1,f\}}$ is $\sigma_{f,k} := \mu^{m_{f,k}} \rho^{n_{f,k}}$, as we have $Q := \prod_{k=1}^{\ell}(\mu^{m'_{f,k}} \rho^{n'_{f,k}})^{\lambda^{k-1}}$, we can see the following holds:

$$\begin{aligned} Q \cdot (\prod_{k=1}^{\ell}(\sigma_{f,k})^{\lambda^{k-1}})^e &= \prod_{k=1}^{\ell}(\mu^{m'_{f,k}} \rho^{n'_{f,k}})^{\lambda^{k-1}} \cdot (\prod_{k=1}^{\ell}(\mu^{m_{f,k}} \rho^{n_{f,k}})^{\lambda^{k-1}})^e \\ &= \mu^{\sum_{k=1}^{\ell}(m'_{f,k}+e\cdot m_{f,k})\cdot\lambda^{k-1}} \cdot \rho^{\sum_{k=1}^{\ell}(n'_{f,k}+e\cdot n_{f,k})\cdot\lambda^{k-1}} \\ &= \mu^{z_4}\rho^{z_5} \end{aligned}$$

Similarly, $Q' \cdot (\prod_{k=1}^{\ell}(\sigma_{f,k})^{\lambda^{k-1}})^e = \mu^{z'_4}\rho^{z'_5}$ also holds.

In addition, we have $R = \mu^{s_f} \cdot h^{(\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}(\sum_{k=1}^{\ell} d_{i',f,k}\cdot p^k))}$ when $\alpha = 1$, and $R = \mu^{s_f} \cdot h^{(\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} \sum_{k=1}^{\ell} d_{i',f,k}\cdot p^k)\cdot\gamma_{i',f})}$ when $\alpha > 1$, $R' = \mu^{s'_f} \cdot h^{(\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}(\sum_{k=1}^{\ell} \beta_{i',f,k}\cdot p^k))}$ when $\alpha = 1$, and $R' = \mu^{s'_f} \cdot h^{(\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}(\sum_{k=1}^{\ell} \beta_{i',f,k}\cdot p^k)\cdot\gamma_{i',f})}$ when $\alpha > 1$. We can prove that the following hold, when $\alpha = 1$:

$$\begin{aligned} S \cdot R^e &= \mu^{w_1} h^{w_2} \cdot \mu^{s_f e} \cdot h^{(\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}(\sum_{k=1}^{\ell} d_{i',f,k}\cdot p^k))\cdot e} \\ &= \mu^{w_1+s_f\cdot e} \cdot h^{w_2+(\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}(\sum_{k=1}^{\ell} d_{i',f,k}\cdot p^k))\cdot e} \\ &= \mu^{z_7} \cdot h^{z_9} \end{aligned}$$

We can also prove that the aforementioned equation also hold when $\alpha > 1$, and $S' \cdot R'^e = \mu^{z'_7} \cdot h^{z'_9}$ also holds in the same way.

Lastly, we have $T := \mu^{w_1}\rho^{w_3}\eta^{w_4}$, $\Delta_{f,0} = \mu^{s_f}\rho^{s'_f}\eta^{o_{i,0}}$. Then we can show that

$$\begin{aligned} T \cdot \Delta_{f,0}^e &= \mu^{w_1}\rho^{w_3}\eta^{w_4} \cdot (\mu^{s_f}\rho^{s'_f}\eta^{o_{i,0}})^e \\ &= \mu^{w_1+s_f\cdot e}\rho^{w_3+s'_f\cdot e}\eta^{w_4+o_{i,0}\cdot e} \\ &= \mu^{z_7} \cdot \rho^{z'_7} \cdot \eta^{z_8} \end{aligned}$$

Soundness. We assume that there exists a PPT extractor Ext which rewinds the protocol with two challenge e_1 and e_2, for $k \in [\ell]$, Ext can get:

RNCE DKG Maintaining ZK argument

CRS: $\mathbb{G}$, $\langle \mu, \rho, \eta, h \rangle \in \mathbb{G} \setminus \{1\}$.

Statement: $\{ct_{f,m',k} = (v_{f,m',k}, \omega_{f,m',k}, \epsilon_{f,m',k}, v'_{f,m',k}, \omega'_{f,m',k}, \epsilon'_{f,m',k}), \sigma_{m',k}\}_{m'=1,k=1}^{M_{\alpha+2},\ell}$, $\{ct_{i',f,k} = (v_{i',f,k}, \omega_{i',f,k}, \epsilon_{i',f,k}, v'_{i',f,k}, \omega'_{i',f,k}, \epsilon'_{i',f,k})\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell}$.

Witness: $\{a_{f,t}, a'_{f,t}\}_{t=0}^{t_{\alpha+2}}$, $\{r_{f,m',k}, r'_{f,m',k}\}_{m'=1,k=1}^{M_{\alpha+2},\ell}$, $\{m_{f,k}, n_{f,k}\}_{k=1}^{\ell}$.

Protocol:

- The prover $\mathcal{P}$ randomly selects $\{o_{f,t}\}_{t=0}^{t_{\alpha+2}} \leftarrow \mathbb{Z}_q$, and computes $\Delta_{f,t} := \mu^{a_{f,t}} \cdot \rho^{a'_{f,t}} \cdot \eta^{o_{f,t}}$ for $t = 0, \cdots, t_{\alpha+2}$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle \Delta_{f,t} \rangle_{t=0}^{t_{\alpha+2}}$;
- $\mathcal{V} \rightarrow \mathcal{P} : \lambda \leftarrow \mathbb{Z}_q$; /* For NIZK, set $\lambda \leftarrow \mathsf{hash}(\langle \Delta_{f,t} \rangle_{t=0}^{t_{\alpha+2}})$ */
- $\mathcal{P}$ does the followings:
  - Choose $a_1, a_2, a_3, \{b_{m',k}, b'_{m',k}\}_{m'=1,k=1}^{M_{\alpha+2},\ell} \leftarrow \mathbb{Z}_q$;
  - Compute $b := \sum_{m'=1}^{M_{\alpha+2}} (\sum_{k=1}^{\ell} b_{m',k}) \cdot \lambda^{m'-1}$, $b' := \sum_{m'=1}^{M_{\alpha+2}} (\sum_{k=1}^{\ell} b'_{m',k}) \cdot \lambda^{m'-1}$;
  - Compute $A := \mu^b$, $B := \rho^b$, $C := \mu^{a_1} \cdot (\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} (\sigma_{m',k})^{b_{m',k}})^{\lambda^{m'-1}})$;
  - Compute $A' := \mu^{b'}$, $B' := \rho^{b'}$, $C' := \mu^{a_2} \cdot (\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} (\sigma_{m',k})^{b'_{m',k}})^{\lambda^{m'-1}})$;
  - Select $\{d_{i',f,k}, \beta_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell} \leftarrow \mathbb{Z}_q$
  - Compute $D_{i'} := \prod_{k=1}^{\ell} ((\epsilon_{i',f,k} / (v_{i',f,k}^{m_{f,k}} \cdot \omega_{i',f,k}^{n_{f,k}})) \cdot h^{d_{i',f,k}})^{p^k}$, $D'_{i'} := \prod_{k=1}^{\ell} ((\epsilon'_{i',f,k} / (v'^{m_{f,k}}_{i',f,k} \cdot \omega'^{n_{f,k}}_{i',f,k})) \cdot h^{\beta_{i',f,k}})^{p^k}$ for $i' \in [N_{\mathsf{QUAL}_\alpha}]$;
  - Select $\{m'_{f,k}, n'_{f,k}\}_{k=1}^{\ell}, \{d'_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell}, \{m''_{f,k}, n''_{f,k}\}_{k=1}^{\ell}, \{\beta'_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell} \leftarrow \mathbb{Z}_q$
  - Compute $P_{i'} := \prod_{k=1}^{\ell} ((\epsilon_{i',f,k} / (v_{i',f,k}^{m'_{f,k}} \cdot \omega_{i',f,k}^{n'_{f,k}})) \cdot h^{d'_{i',f,k}})^{p^k}$, $P'_{i'} := \prod_{k=1}^{\ell} ((\epsilon'_{i',f,k} / (v'^{m''_{f,k}}_{i',f,k} \cdot \omega'^{n''_{f,k}}_{i',f,k})) \cdot h^{\beta'_{i',f,k}})^{p^k}$ for $i' \in [N_{\mathsf{QUAL}_\alpha}]$;
  - Compute $Q := \prod_{k=1}^{\ell} (\mu^{m'_{f,k}} \rho^{n'_{f,k}})^{\lambda^{k-1}}$, $Q' := \prod_{k=1}^{\ell} (\mu^{m''_{f,k}} \rho^{n''_{f,k}})^{\lambda^{k-1}}$; /*user's public key $\sigma_{f,k} := \mu^{m_{f,k}} \rho^{n_{f,k}}$*/
  - Select $w_1, w_2, w_3, w_4, w_5 \leftarrow \mathbb{Z}_p$;
  - Compute $S := \mu^{w_1} h^{w_2}$, $S' := \mu^{w_3} h^{w_5}$, $T := \mu^{w_1} \rho^{w_3} \eta^{w_4}$;
- $\mathcal{P} \rightarrow \mathcal{V} : \langle A, B, C, A', B', C', \langle D_{i'}, D'_{i'}, P_{i'}, P'_{i'} \rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, Q, Q', R, S, S', T \rangle$;
- $\mathcal{V} \rightarrow \mathcal{P} : e \leftarrow \mathbb{Z}_q$; /* For NIZK, set $e \leftarrow \mathsf{hash}(\langle \lambda, A, B, C, A', B', C', \langle D_{i'}, D'_{i'}, P_{i'}, P'_{i'} \rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, Q, Q', R, S, S', T \rangle)$ */
- $\mathcal{P}$ computes the following:
  - $z_1 := a_1 + (\sum_{m'=1}^{M_{\alpha+2}} (g_{f,m'}) \cdot \lambda^{m'-1}) \cdot e$, $z'_1 := a_2 + (\sum_{m'=1}^{M_{\alpha+2}} (g'_{f,m'}) \cdot \lambda^{m'-1}) \cdot e$, $z_2 := a_3 + e \cdot (\sum_{m'=1}^{M_{\alpha+2}} (\sum_{t=0}^{t_{\alpha+2}} o_{f,t} \cdot m'^t) \cdot \lambda^{m'-1})$;
  - $T := \mu^{a_1} \cdot \rho^{a_2} \cdot \eta^{a_3}$;
  - $z_{f,m',k} := b_{m',k} + r_{f,m',k} \cdot p^k \cdot e$, $z'_{f,m',k} := b'_{m',k} + r'_{f,m',k} \cdot p^k \cdot e$ for $m \in [M_{\alpha+2}]$, $k \in [\ell]$;
  - $z_{4,f,k} := m'_{f,k} + e \cdot m_{f,k}$, $z_{5,f,k} := n'_{f,k} + e \cdot n_{f,k}$, $z'_{4,f,k} := m''_{f,k} + e \cdot m_{f,k}$, $z'_{5,f,k} := n''_{f,k} + e \cdot n_{f,k}$ for $k \in [\ell]$;
  - $z_{6,i',f} := \sum_{k=1}^{\ell} (d'_{i',f,k} + e \cdot d_{i',f,k}) \cdot p^k$, $z'_{6,i',f} := \sum_{k=1}^{\ell} (\beta'_{i',f,k} + e \cdot \beta_{i',f,k}) \cdot p^k$ for $i \in [N_{\mathsf{QUAL}_\alpha}]$;
  - $z_7 := w_1 + e \cdot s_f$, $z'_7 := w_3 + e \cdot s'_f$, $z_8 := w_4 + e \cdot o_{f,0}$
  - $z_9 := \begin{cases} w_2 + e \cdot (\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} (\sum_{k=1}^{\ell} d_{i',f,k} \cdot p^k)) & \alpha = 1 \\ w_2 + e \cdot (\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} (\sum_{k=1}^{\ell} d_{i',f,k} \cdot p^k) \cdot \gamma_{i',f}) & \alpha > 1 \end{cases}$, $z'_9 := \begin{cases} w_5 + e \cdot (\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} (\sum_{k=1}^{\ell} \beta_{i',f,k} \cdot p^k)) & \alpha = 1 \\ w_5 + e \cdot (\sum_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} (\sum_{k=1}^{\ell} \beta_{i',f,k} \cdot p^k) \cdot \gamma_{i',f}) & \alpha > 1 \end{cases}$
- $\mathcal{P} \rightarrow \mathcal{V} : \langle T, z_1, z'_1, z_2, \langle z_{f,m',k}, z'_{f,m',k} \rangle_{m'=1,k=1}^{M_{\alpha+2},\ell}, \langle z_{4,f,k}, z_{5,f,k} \rangle_{k=1}^{\ell}, \langle z_{6,i',f}, z'_{6,i',f} \rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, z_7, z'_7, z_8, z_9, z'_9 \rangle$

Verification:

$\mathcal{V}$ computes the followings:

- /* For NIZK, set $\lambda \leftarrow \mathsf{hash}(\langle \Delta_{f,t} \rangle_{t=0}^{t_{\alpha+2}})$ */ , /* For NIZK, set $e \leftarrow \mathsf{hash}(\langle \lambda, A, B, C, A', B', C', \langle D_{i'}, D'_{i'}, P_{i'}, P'_{i'} \rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, Q, Q', R, S, S', T \rangle)$ */
- $\Delta := \prod_{m'=1}^{M_{\alpha+2}} (\prod_{t=0}^{t_{\alpha+2}} (\Delta_{f,t})^{m'^t})^{\lambda^{m'-1}})$, $z_3 := \sum_{m'=1}^{M_{\alpha+2}} (\sum_{k=1}^{\ell} z_{f,m',k}) \cdot \lambda^{m'-1}$, $z'_3 := \sum_{m'=1}^{M_{\alpha+2}} (\sum_{k=1}^{\ell} z'_{f,m',k}) \cdot \lambda^{m'-1}$
- $D := \prod_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} D_{i'}$, $D' := \prod_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} D'_{i'}$;
- $z_4 := \sum_{k=1}^{\ell} z_{4,f,k} \cdot \lambda^{k-1}$, $z_5 := \sum_{k=1}^{\ell} z_{5,f,k} \cdot \lambda^{k-1}$, $z'_4 := \sum_{k=1}^{\ell} z'_{4,f,k} \cdot \lambda^{k-1}$, $z'_5 := \sum_{k=1}^{\ell} z'_{5,f,k} \cdot \lambda^{k-1}$
- Compute $R := \begin{cases} \prod_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} D_{i'} & \alpha = 1 \\ \prod_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} (D_{i'})^{\gamma_{i',f}} & \alpha > 1 \end{cases}$, Compute $R' := \begin{cases} (\prod_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} D'_{i'}) & \alpha = 1 \\ \prod_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} (D_{i'})^{\gamma_{i',f}} & \alpha > 1 \end{cases}$

$\mathcal{V}$ check the followings:

- $\mu^{z_1} \cdot \rho^{z'_1} \cdot \eta^{z_2} = \Delta^e \cdot T$;
- $(\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} v_{f,m',k}^{p^k})^{\cdot \lambda^{m'-1}})^e \cdot A = \mu^{z_3}$, $(\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} v'^{p^k}_{f,m',k})^{\cdot \lambda^{m'-1}})^e \cdot A' = \mu^{z'_3}$;
- $(\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} \omega_{f,m',k}^{p^k})^{\cdot \lambda^{m'-1}})^e \cdot B = \rho^{z_3}$, $(\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} \omega'^{p^k}_{f,m',k})^{\cdot \lambda^{m'-1}})^e \cdot B' = \rho^{z'_3}$;
- $(\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} \epsilon_{f,m',k}^{p^k})^{\cdot \lambda^{m'-1}})^e \cdot C = \mu^{z_1} \cdot (\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} \sigma_{m',k}^{z_{f,m',k}})^{\lambda^{m'-1}})$,
  $(\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} \epsilon'^{p^k}_{f,m',k})^{\cdot \lambda^{m'-1}})^e \cdot C' = \mu^{z'_1} \cdot (\prod_{m'=1}^{M_{\alpha+2}} (\prod_{k=1}^{\ell} \sigma_{m',k}^{z'_{f,m',k}})^{\lambda^{m'-1}})$;
- for $i' = 1, \cdots, N_{\mathsf{QUAL}_\alpha}$
  - $P_{i'} \cdot (D_{i'})^e = (\prod_{k=1}^{\ell} (\epsilon_{i',f,k})^{e+1} / (v_{i',f,k}^{z_{4,f,k}} \cdot \omega_{i',f,k}^{z_{5,f,k}})^{p^k}) \cdot h^{z_{6,i',f}}$; The correctness of decryption using sk: $\{m_{f,k}, n_{f,k}\}_{k=1}^{\ell}$
  - $P'_{i'} \cdot (D'_{i'})^e = (\prod_{k=1}^{\ell} (\epsilon'_{i',f,k})^{e+1} / (v'^{z'_{4,f,k}}_{i',f,k} \cdot \omega'^{z'_{5,f,k}}_{i',f,k})^{p^k}) \cdot h^{z'_{6,i',f}}$; The correctness of decryption using sk: $\{m_{f,k}, n_{f,k}\}_{k=1}^{\ell}$
- $Q \cdot (\prod_{k=1}^{\ell} (\sigma_{f,k})^{\lambda^{k-1}})^e = \mu^{z_4} \rho^{z_5}$; /* Using right secret key to decrypt, which is tied with the public key*/
- $Q' \cdot (\prod_{k=1}^{\ell} (\sigma_{f,k})^{\lambda^{k-1}})^e = \mu^{z'_4} \rho^{z'_5}$; /* Using right secret key to decrypt, which is tied with the public key*/
- $S \cdot R^e = \mu^{z_7} \cdot h^{z_9}$, $S' \cdot R'^e = \mu^{z'_7} \cdot h^{z'_9}$, $T \cdot \Delta_{f,0}^e = \mu^{z_7} \cdot \rho^{z'_7} \cdot \eta^{z_8}$; /* Decryption results are consistent with $\Delta_{f,0}$*/

Figure 17: RNCE DKG Maintaining ZK argument

$$z^{(1)}_{4,f,k} := m'_{f,k} + e_1 \cdot m_{f,k}, z_{5,f,k} := n'_{f,k} + e_1 \cdot n_{f,k}$$
$$z^{(2)}_{4,f,k} := m'_{f,k} + e_2 \cdot m_{f,k}, z_{5,f,k} := n'_{f,k} + e_2 \cdot n_{f,k}$$

Thus, Ext will get $\{m_{f,k}, n_{f,k}\}_{k=1}^{\ell}$ by computing $(z^{(1)}_{4,f,k} - z^{(2)}_{4,f,k})/(e_1 - e_2)$ and $(z^{(1)}_{5,f,k} - z^{(2)}_{5,f,k})/(e_1 - e_2)$ for $k \in [\ell]$.

Zero Knowledge. In terms of special honest verifier zero-knowledge, we construct a simulator Sim that takes the challenge $e \leftarrow \mathbb{Z}_q$ and statement $\{ct_{f,m',k}, \sigma_{m',k}\}_{m'=1,k=1}^{M_{\alpha+2},\ell}, \{ct_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell}$ as inputs, it outputs a simulated transcript the distribution of which is indistinguishable from the real one.

Sim first picks $\{\langle z_{4,f,k}, z_{5,f,k}\rangle_{k=1}^{\ell}, \langle z_{6,i',f}, z'_{6,i',f}\rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}\} \leftarrow \mathbb{Z}_q$, then picks $\langle D_{i'}, D'_{i'}\rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}} \leftarrow \mathbb{G}$. For $i' = 1, \cdots, N_{\mathsf{QUAL}_\alpha}$, Sim computes $P_{i'} = ((\prod_{k=1}^{\ell}(\epsilon_{i',f,k})^{e+1}/(v_{i',f,k}^{z_{4,f,k}} \cdot \omega_{i',f,k}^{z_{5,f,k}})^{p^k}) \cdot h^{z_{6,i',f}}) \cdot (D_{i'})^{-e}$, and $P'_{i'} = ((\prod_{k=1}^{\ell}(\epsilon'_{i',f,k})^{e+1}/(v'^{z'_{4,f,k}}_{i',f,k} \cdot \omega'^{z'_{5,f,k}}_{i',f,k})^{p^k}) \cdot h^{z'_{6,i',f}}) \cdot (D'_{i'})^{-e}$.

Next, Sim computes $z_4 = \sum_{k=1}^{\ell} z_{4,f,k} \cdot \lambda^{k-1}, z_5 = \sum_{k=1}^{\ell} z_{5,f,k} \cdot \lambda^{k-1}$, $z'_4 = \sum_{k=1}^{\ell} z'_{4,f,k} \cdot \lambda^{k-1}, z'_5 = \sum_{k=1}^{\ell} z'_{5,f,k} \cdot \lambda^{k-1}$, using the valus he picked in last step. Then computes $Q = \mu^{z_4}\rho^{z_5} \cdot (\prod_{k=1}^{\ell}(\sigma_{f,k})^{\lambda^{k-1}})^{-e}$ and $Q' = \mu^{z'_4}\rho^{z'_5} \cdot (\prod_{k=1}^{\ell}(\sigma_{f,k})^{\lambda^{k-1}})^{-e}$.

Lastly, Sim selects $z_7, z'_7, z_8, z_9, z'_9 \leftarrow \mathbb{Z}_q$, and computes R following the protocol. Then Sim computes $S = (\mu^{z_7} \cdot h^{z_9}) \cdot R^{-e}$, $S = (\mu^{z'_7} \cdot h^{z'_9})' \cdot R'^{-e}$, $T = (\mu^{z_7} \cdot \rho^{z'_7} \cdot \eta^{z_8}) \cdot \Delta_{f,0}^{-e}$.

Sim outputs the simulated transcripts:

$$\langle\langle z_{4,f,k}, z_{5,f,k}\rangle_{k=1}^{\ell}, \langle z_{6,i',f}, z'_{6,i',f}\rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, z_7, z'_7, z_8, z_9, z'_9, \langle D_{i'}, D'_{i'}, P_{i'}, P'_{i'}\rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, Q, Q', R, S, S', T\rangle$$

.

As $\{m'_{f,k}, n'_{f,k}, m''_{f,k}, n''_{f,k}\}_{k=1}^{\ell}, \{d'_{i',f,k}, \beta'_{i',f,k}\}_{i'=1,k=1}^{N_{\mathsf{QUAL}_\alpha},\ell}, w_1, w_2, w_3, w_4, w_5$ are all random values, $\langle z_{4,f,k}, z_{5,f,k}\rangle_{k=1}^{\ell}, \langle z_{6,i',f}, z'_{6,i',f}\rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}, z_7, z'_7, z_8, z_9, z'_9$ are also uniformly random. Moreover, Q, Q', S, S', T are random group element. In addition, $\langle D_{i'}, D'_{i'}, P_{i'}, P'_{i'}\rangle_{i'=1}^{N_{\mathsf{QUAL}_\alpha}}$ are in the same distribution as real transcripts since they follow the same relation based on the verification equation. Therefore, simulated transcripts has the same distribution as real transcripts in a real argument. □

8 REPUTATION MANAGEMENT SCHEME

8.1 Reputation Score Calculation

The proposed reputation management scheme objectively maps each user's activity in treasury system to a dynamically updated reputation score in specific field. To diversify treasury system in practical, we encourage users to participant proposals in different fields, and calculate/compare reputation scores in different fields individually. In our scheme, reputation score $\mathsf{Rep}^{(k)}_{\mathsf{Fld}}(u_i)$ of user u_i in field Fld in the k-th treasury period is aggregated by four reputation factors, $\mathsf{Rep}^{(k)}_{\mathsf{Fld}}(u_i) := (\text{Rep-RW}^{(k)}(u_i), \text{Rep-PC}_{\mathsf{Fld}}{}^{(k)}(u_i), \text{Rep-PP}_{\mathsf{Fld}}{}^{(k)}(u_i), \text{Rep-TS}^{(k)}(u_i))$. $\text{Rep-RW}^{(k)}(u_i)$ represents partial reputation score gained from user's **regularity of work** during user's involvement. Users are expected to regularly join the system and contribute their knowledge in a certain field, the usability of the system is directly dependent on users' engagement. $\text{Rep-PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$ stands for partial reputation score based on the **quality of total productive contributions**. Users are motivated to provide the system constructive decisions which reflect their own areas of expertise, the outcome of these decisions correspondingly influence users' reputation scores. $\text{Rep-PP}_{\mathsf{Fld}}{}^{(k)}(u_i)$ is partial reputation score related to the **funded percentage** of user's proposed proposals in a field, when user serve as a project proposer during his all involvement. We believe that a reputable user should have experience in submitting valuable and winning proposals in a certain field. The more winning proposals a user has, the more he is likely a down-to-earth user beyond just empty talk, hence, the more reputation scores should the user get. Different from other three reputation factors, $\text{Rep-TS}^{(k)}(u_i)$ as partial reputation score of **treasury short-list contribution** is particularly designed for treasury committee. As mentioned before, treasury committee manages to nominate short-listed proposals, which will be endorsed by voters in the second voting stage. To prevent tyranny of treasury committee, if short-listed proposals are different from voters' final decision, reputation scores of treasury committee members who signed the short-list should be negative values; otherwise, the reputation scores will be positive values. From another perspective, by doing so, we can incentivise treasury committee to make proper decision by increasing or decreasing their reputation scores. For those treasury committee members who didn't sign the short-list, their $\text{Rep-TS}^{(k)}(u_i)$ value is zero as they didn't contribute in short-list genration.

To assist understanding of the following reputation algorithm RepCal in Fig. **??**, we summarised the frequently used notations in Table **??**. In every updating treasury period k, we calculate reputation score $\mathsf{Rep}_{\mathsf{Fld}}(u_i)^{(k)} \in [0, 1]$ of user u_i, in different fields indexed by Fld. Considering that users might not join the system continuously, we use $[k_0, \ldots, k]$ for the discontinuous/continuous periods when user participated, and use T for user's total participated periods. As mentioned before, each user might have one or more entity roles in our system. Thus we introduce the role concept $\text{Role} \in \{\mathcal{O}^{(k)} \cup \mathcal{VT}^{(k)} \cup \mathcal{E}^{(k)} \cup \mathcal{T}_k\}$ when computing reputation score, user's role in current treasury period will influence how to compute the four reputation factors as follows.

Regularity of work. Voters who want to participant in the system are required to freeze a number of stakes on mainchain, this part of stakes will become their voting power. Thus, we define their regularity of work as voting power ratio $\{\mathsf{VPR}^{(t)}(u_i)\}_{t\in[k_0,\ldots,k]}$ (personal voting power divided by the total voting power in each period), illustrated in Algorithm **??** as $\text{Rep-RW}^{(k)}(u_i)$. We first compute user's average voting power ratio $\overline{\mathsf{VPR}}$ during the whole T periods. Then, we measure the regularity with which a user contributes voting power to the whole system, which is indicated as standard deviation of the voting power ratio SD_{VPR}. By combining $\overline{\mathsf{VPR}}$ with SD_{VPR}, we can ensure user's partial reputation score $\text{Rep-RW}^{(k)}(u_i)$ is computed based on the user's total regularity of voting power contributions. There is an important point that requires attention regarding treasury committee's voting power ratio in this reputation factor. As mentioned earlier, only voters need to cast stakes as voting power to show their honesty and loyalty, while experts (including treasury committee) are trusted

acquiescently because of high reputation scores. Therefore, if a user has been served as a treasury committee member in some certain treasury period $h \in \{[k_0, \ldots, k]\}$, then his voting power ratio $\{\mathsf{VPR}^{(h)}(u_i)\}$ should be zero.

Quality of total productive contributions. Ideally, users are always encouraged to make right decisions in treasury system. In one treasury period, proposals are in more than 1 fields. Hence, we measure this part of reputation score based on fields that users participated during his involvement, which is different from $\text{Rep-RW}^{(k)}(u_i)$. To diversity opinions in the system, we incentive users to express their own opinions instead of delegating voting power to experts consistently. Thus, we define the partial reputation score of total productive contributions quality (denoted by $\text{Rep-PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$) as the outcome of decisions made by the user himself (not the decisions made through delegation). Specifically, we introduce the number of projects u_i voted Yes by himself (denoted by $Y_{\mathsf{Fld}}^{[k_0,\ldots,k]}(u_i)$) and the number of projects u_i voted No by himself (denoted by $N_{\mathsf{Fld}}^{[k_0,\ldots,k]}(u_i)$) to define users' personal contributions sign Δ. If he hasn't voted for any projects (either Yes or No) by himself during periods $[k_0, \ldots, k]$, Δ will be 0, and this user will get zero score in this reputation factor; otherwise Δ will be 1.

We define $\{\mathsf{ACC}_{\mathsf{Fld}}{}^{(t)}(u_i)\}_{t\in[T]}$ as accuracy rate, which means the percentage of proposals supported by u_i entered the list of winning projects when he joined the t-th time. Namely, period k_0 is the first time, while period k is the T-th time. Note that if a user has been a member of treasury committee in $o \in [T]$ period, then $\mathsf{ACC}_{\mathsf{Fld}}{}^{(o)}(u_i)$ should be zero. We define $\mu^{(k)}$ of $\text{Rep-PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$ as the average accuracy rate since the user joins the system (during periods $[k_0, \ldots, k]$), which represents the fraction of positive work that a user has contributed to the whole system. Even reputation is calculated on user's previous behaviours, as mentioned by Josang *et al.* [**?**], a user's behaviour in the last few days is a more accurate factor of the user's future behaviour than analysing all previous behaviour on the network. Thus, instead of computing the normal average, we calculate the exponential moving average (EMA) $\mu^{(t)}$ of accuracy rate $\mathsf{ACC}_{\mathsf{Fld}}{}^{(t)}(u_i)$. Consequently, $\mu^{(t)}$ provides multiplying factors to give different weights to accuracy rate at different time during periods $[k_0, \ldots, k]$. We set α as a smoothing factor that can be adjusted to make the reputation factor $\text{Rep-RW}^{(k)}(u_i)$ more or less progressive. The closer that α gets to 0, the more weight will be assigned to the initial accuracy rate in earlier projects. On the contrary, the closer that α gets to 1, the recent accuracy rate that user gets is more important. Combining EMA of accuracy rate $\mu^{(k)}$ with standard deviation of accuracy rate $SD_{\mathsf{ACC}_{\mathsf{Fld}}}$ and personal contribution sign Δ, we can get $\text{Rep-PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$ as user's total productive personal contribution to the system in field Fld.

Funded percentage. We believe that a reputable user should have past involvement in proposing proposals in our system. We name this type of reputation factor $\text{Rep-PP}_{\mathsf{Fld}}{}^{(k)}(u_i)$ as the funded percentage (number of funded projects divided by number of total proposed projects) when he served as project owner. We define $\mathsf{PWR}_{\mathsf{Fld}}{}^{[k_0,\ldots,k]}$ winning rate of proposals proposed by user u_i as proposer during periods $[k_0, \ldots, k]$, then $\text{Rep-PP}_{\mathsf{Fld}}{}^{(k)}(u_i)$ will simply be user's winning rate $\mathsf{PWR}_{\mathsf{Fld}}{}^{[k_0,\ldots,k]}$.

Treasury short-list contribution. When a user served as treasury committee, he will be asked to jointly generate short list of proposals with other committee members. Since the final winning projects list is heavily dependent on this short-list, we need to regulate treasury committee's decision by adjusting their reputation scores based on the decision made in endorsement stage. As mentioned before, a group of treasury committee members should sign the short-list, and the short-list will only be valid if the total reputation scores of this group are more than 50% of the treasury committee's reputation scores. If a user (treasury committee member) didn't sign the short-list, we think he might not agree with this decision, thus his partial reputation score based on his treasury short-list contribution $\text{Rep-TS}^{(k)}(u_i)$ should be 0. When a user sign the short-list, we compare this short-list with the decision in the second voting stage, and count percentage of differences in these two sheets, $\mathsf{ConsR}(u_i)^{(k)}$ as the consistent rate of short-listed proposals and decision made by voters. Then this part of reputation score will be $\text{Rep-TS} := \omega \cdot log_{10}(\mathsf{ConsR}(u_i)^{(k)}) + 1$, where ω is a treasury committee smoothing parameter to control Rep-TS punishment.

After computing the aforementioned four reputation factors, we aggregate them based on difference weights associated with different reputation factors, which is $w_1 \cdot \text{Rep-RW}^{(k)}(u_i) + w_2 \cdot \text{Rep-PC}_{\mathsf{Fld}}{}^{(k)}(u_i) + w_3 \cdot \text{Rep-PP}_{\mathsf{Fld}}{}^{(k)}(u_i) + w_4 \cdot \text{Rep-TS}^{(k)}(u_i)$. However, when weighting Rep-RW and $\text{Rep-PC}_{\mathsf{Fld}}$, we set that w_1 should be smaller than w_2. Otherwise, users with more voting power can gain more reputation scores than experts, it breaks the fairness of our reputation management scheme. Then, this weighted aggregated result multiplied by the current treasury period number k, forms the final basis of user's reputation score x. The design of multiplying k meets our goal where we think the recent contribution values more, the reputation scores gained from recent activities should be higher.

At last, we adopt a sigmoid function $f(x)$ to compute the progression of reputation score and restrain the overall reputation score between $[0, 1]$. This sigmoid function is paramterized by (a, λ), which can dynamically adjust reputation score growth rate and slope. With this sigmoid function, we can ensure that users can only increase their reputation scores very slowly at first. Thus, new comer in this system won't be trusted that much. After being honest and loyalty in the system for a long enough periods, we will increase their reputation score grow rate. When $x = a$, $f(x)$ reaches the inflection point, after that reputation growth rate will start to decline.

In addition to the behaviours calculated inside of treasury system, our scheme also consider public reputation of the user from outside sources, denoted by $\text{Rep-Pub}_{\mathsf{Fld}} \in [0, 1]$. Public reputation scores are user facing, it implies the fundamental credibility of users. When a knowledgeable and renowned expert joins the voting system, it logically follows that he has higher initial reputation than other ordinary new users. $\mathsf{CR}(u_i) \in \{0, 1\}$ is the credibility of the user based on binary rating scheme, $\mathsf{CR}(u_i)$ is set to 1 for each honest user, and is set to 0 if a user has misbehaved. Note that if $\mathsf{CR}(u_i) = 0$, his reputation score will forever be 0. The corresponding user will be blacklisted by the reputation management scheme, since we consider that this entity is not trustworthy any more.

Table 1: Notations

k_0	the first treasury period when user joined
k	current treasury period
T	total treasury periods u_i joined including the k-th period
Fld	field of reputation score
$\text{Rep}_{\text{Fld}}(u_i)^{(k)}$	reputation score of user u_i in field Fld in k-th treasury period
Role $\in \{\mathcal{O}^{(k)} \cup \mathcal{VT}^{(k)} \cup \mathcal{E}^{(k)} \cup \mathcal{T}_k\}$	The role of user in current treasury period
$\{\text{VPR}^{(t)}(u_i)\}_{t\in[k_0,\ldots,k]}$	the voting power ratio casted by u_i in t-th period
$\{\text{ACC}_{\text{Fld}}{}^{(t)}(u_i)\}_{t\in[T]}$	the percentage of proposals supported by u_i in filed Fld entered the list of winning projects when he joined the t-th time
$Y_{\text{Fld}}^{[k_0,\ldots,k]}(u_i)$	the number of projects u_i voted Yes by himself in filed Fld from k_0-th period to k-th period
$N_{\text{Fld}}^{[k_0,\ldots,k]}(u_i)$	the number of projects u_i voted No by himself in filed Fld from k_0-th period to k-th period
$\text{Sign}^{(k)}(u_i) \in \{0,1\}$	if u_i is a treasury committee member in period k and he signed the short-list, then $\text{Sign}^{(k)}(u_i) = 1$; otherwise, $\text{Sign}^{(k)}(u_i) = 0$.
$\text{ConsR}(u_i)^{(k)}$	consistent rate of short-listed proposals and decision made by voters
ω	treasury committee smoothing parameter
$\text{PWR}_{\text{Fld}}{}^{[k_0,\ldots,k]}$	winning rate of proposals proposed by u_i as a proposer from k_0-th period to k-th period in field Fld
$\alpha \in (0,1)$	a constant smoothing factor
(w_1, w_2, w_3, w_4)	reputation weighting parameters where w_2 should be superior than w_1
(a, λ)	reputation system parameters
$\text{CR}(u_i)$	User's credibility, if u_i is honest, $\text{CR}(u_i) = 1$; otherwise $\text{CR}(u_i) = 0$
$\text{Rep-Pub}_{\text{Fld}}$	the optional external source of reputation for u_i

Algorithm RepCal

Input: Fld; k_0; k; $\text{Sign}^{(k)}(u_i)$; $\text{ConsR}(u_i)^{(k)}$; α; (w_1, w_2, w_3, w_4); (a, λ); $\text{CR}(u_i)$; $\text{Rep-Pub}_{\text{Fld}}$. $\{\text{VPR}^{(t)}(u_i)\}_{t\in[k_0,\ldots,k]}$; $\{\text{ACC}_{\text{Fld}}{}^{(t)}(u_i)\}_{t\in[k_0,\ldots,k]}$; $Y_{\text{Fld}}^{[k_0,\ldots,k]}(u_i)$; $N_{\text{Fld}}^{[k_0,\ldots,k]}(u_i)$; $\text{PWR}_{\text{Fld}}{}^{[k_0,\ldots,k]}$; T; Role; ω;

Output: $\text{Rep}_{\text{Fld}}(u_i)^{(k)} \in [0,1]$

Regularity of Work:

- $\overline{\text{VPR}} := \frac{1}{T}\sum_{t\in[k_0,\ldots,k]} \text{VPR}^{(t)}(u_i)$
- $SD_{\text{VPR}} := \sqrt{\frac{\sum_{t\in[k_0,\ldots,k]}(\text{VPR}^{(t)}(u_i)-\overline{\text{VPR}})^2}{T}}$
- $\text{Rep-RW}^{(k)}(u_i) := \frac{\overline{\text{VPR}}}{1+SD_{\text{VPR}}}$;

Quality of total productive contributions:

- If $Y_{\text{Fld}}^{[k_0,\ldots,k]}(u_i) + N_{\text{Fld}}^{[k_0,\ldots,k]}(u_i) \geq 1$
  - $\Delta_{\text{Fld}} := 1$;
  - else $\Delta_{\text{Fld}} := 0$.
- $\mu^{(1)} := \text{ACC}_{\text{Fld}1}(u_i)$;
- $S^{(1)} := 0$;
- for $t = [2, T]$
  - $\mu^{(t)} := (1-\alpha)\mu^{(t-1)} + \alpha \cdot \text{ACC}_{\text{Fld}}{}^{(t)}(u_i)$;
  - $S^{(t)} := (1-\alpha)S^{(t-1)} + \alpha(\text{ACC}_{\text{Fld}}{}^{(t)}(u_i) - \mu^{(t-1)})^2)$;
- $SD_{\text{ACC}_{\text{Fld}}} := \sqrt{S^{(t)}}$;
- $\text{Rep-PC}_{\text{Fld}}{}^{(k)}(u_i) := \Delta\frac{\mu^{(k)}}{1+SD_{\text{ACC}_{\text{Fld}}}}$;

Funded Percentage:

- $\text{Rep-PP}_{\text{Fld}}{}^{(k)}(u_i) := \text{PWR}_{\text{Fld}}{}^{[k_0,\ldots,k]}$;

Treasury Short-List Contribution:

- If Role $\in \mathcal{T}_k$:
  - if $\text{Sign}^{(k)}(u_i) = 1$, $\text{Rep-TS}^{(k)}(u_i) = \omega \cdot log_{10}(\text{ConsR}(u_i)^{(k)}) + 1$;
  - else, $\text{Rep-TS}^{(k)}(u_i) = 0$.
- else, $\text{Rep-TS}^{(k)}(u_i) = 0$;

Reputation Factors Aggregation:

- $x = (w_1 \cdot \text{Rep-RW}^{(k)}(u_i) + w_2 \cdot \text{Rep-PC}_{\text{Fld}}{}^{(k)}(u_i) + w_3 \cdot \text{Rep-PP}_{\text{Fld}}{}^{(k)}(u_i) + w_4 \cdot \text{Rep-TS}^{(k)}(u_i)) \cdot k$;
- $f(x) = \frac{1}{2} \cdot (1 + \frac{x-a}{\lambda+|x-a|})$;

Output:

- $\text{Rep}_{\text{Fld}}(u_i)^{(k)} = min(1, \text{CR}(u_i) \cdot (\text{Rep-Pub}_{\text{Fld}}{}^{(k_0)}(u_i) + f(x)))$;

Figure 18: Reputation Computation Algorithm RepCal

8.2 Proof of Reputation

In our system, reputation score of each user is required to be recorded and agreed by all the other users. To meet this goal, we design special data-structure to store reputation scores for each node/user, and redesign block structure as shown in Fig. ??.

A block is divided into three parts: block header, transaction block contents, and reputation block contents. In block header, we store the Height as the index of current block, Time-stamp as a non-repeated random nonce, Version as the serial number when generating the block based consensus protocol, Signature as the encryption or hash value generated by block creator, and Hash value of previous block (including Transaction Block Contents and Reputation Block Contents). Transaction Block Contents and Reputation Block Contents have a list of transactions and users' reputation scores in order, which are organised in the form of Merkel tree.

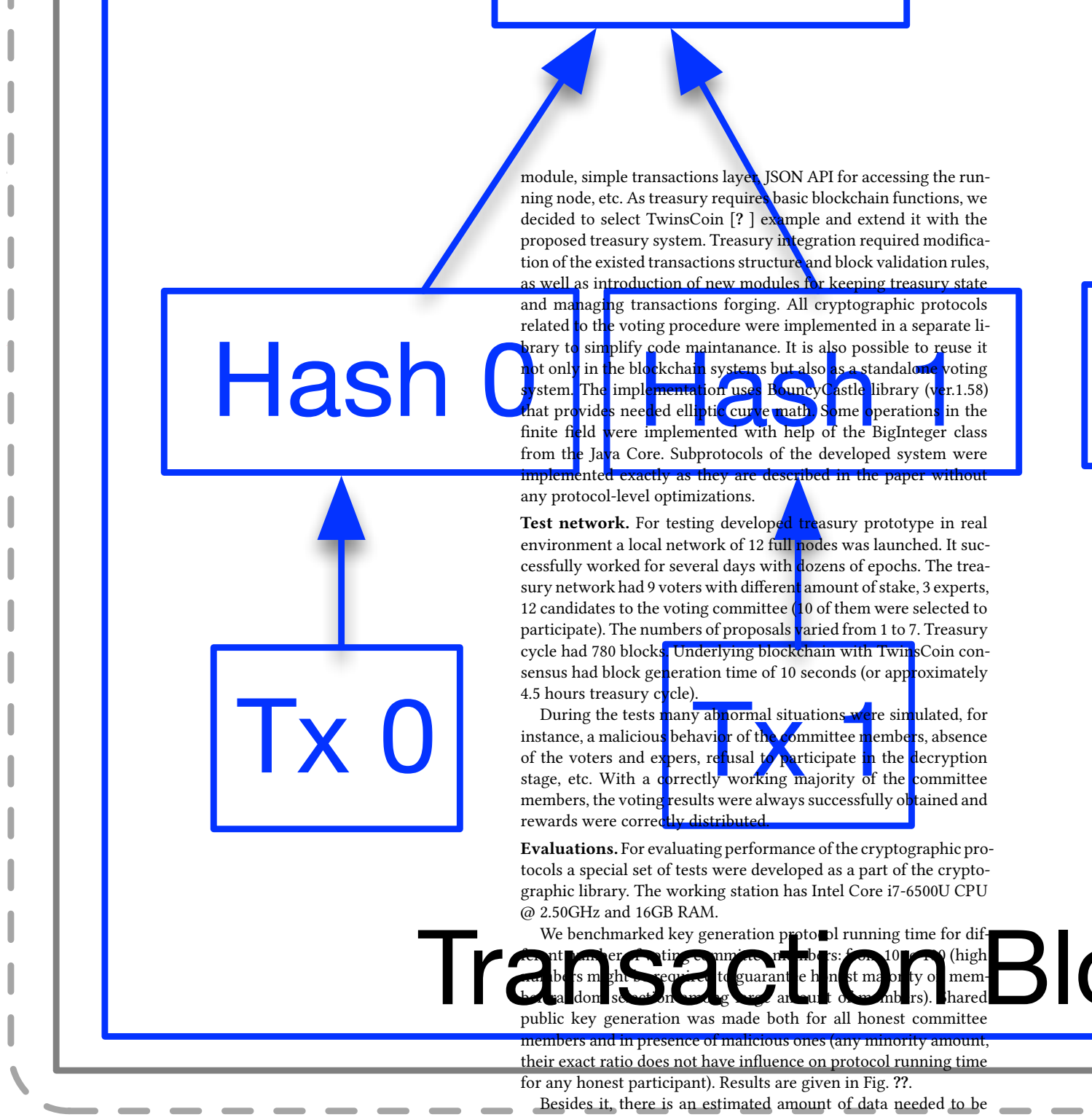

Figure 19: Block Structure

9 IMPLEMENTATION AND PERFORMANCE

Prototyping. The proposed treasury system was implemented as a fully functional cryptocurrency prototype. As an underlying framework we used Scorex 2.0 [**?**] that provides basic blockchain functionality. It is a flexible modular framework designed particularly for fast prototyping with a rich set of already implemented functionalities such as asynchronous peer-to-peer network layer, built-in blockchain support with pluggable and extendable consensus module, simple transactions layer, JSON API for accessing the running node, etc. As treasury requires basic blockchain functions, we decided to select TwinsCoin [**?**] example and extend it with the proposed treasury system. Treasury integration required modification of the existed transactions structure and block validation rules, as well as introduction of new modules for keeping treasury state and managing transactions forging. All cryptographic protocols related to the voting procedure were implemented in a separate library to simplify code maintanance. It is also possible to reuse it not only in the blockchain systems but also as a standalone voting system. The implementation uses BouncyCastle library (ver.1.58) that provides needed elliptic curve math. Some operations in the finite field were implemented with help of the BigInteger class from the Java Core. Subprotocols of the developed system were implemented exactly as they are described in the paper without any protocol-level optimizations.

Test network. For testing developed treasury prototype in real environment a local network of 12 full nodes was launched. It successfully worked for several days with dozens of epochs. The treasury network had 9 voters with different amount of stake, 3 experts, 12 candidates to the voting committee (10 of them were selected to participate). The numbers of proposals varied from 1 to 7. Treasury cycle had 780 blocks. Underlying blockchain with TwinsCoin consensus had block generation time of 10 seconds (or approximately 4.5 hours treasury cycle).

During the tests many abnormal situations were simulated, for instance, a malicious behavior of the committee members, absence of the voters and expers, refusal to participate in the decryption stage, etc. With a correctly working majority of the committee members, the voting results were always successfully obtained and rewards were correctly distributed.

Evaluations. For evaluating performance of the cryptographic protocols a special set of tests were developed as a part of the cryptographic library. The working station has Intel Core i7-6500U CPU @ 2.50GHz and 16GB RAM.

We benchmarked key generation protocol running time for different number of voting committee members: from 10 to 50 (high numbers might be required to guarantee honest majority of members in random selection among large amount of members). Shared public key generation was made both for all honest committee members and in presence of malicious ones (any minority amount, their exact ratio does not have influence on protocol running time for any honest participant). Results are given in Fig. **??**.

Besides it, there is an estimated amount of data needed to be transmitted over a peer-to-peer network to complete the protocol, in dependence of committee size and malicious members ratio. Results are given in Fig. **??** (recall that even controlling 50% of the committee, an attacker can break confidentiality of voters' ballots, but not their integrity or tally result).

Ballot generation 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. To get tally results, it is needed to collect all ballots from participating voters, validate

their correctness (via attached NIZK) and then do tally for all correct ballots. Figure **??** shows the prover's running time, the verifier's running time and the size of the unit vector ZK proof that has been used in the ballot casting.

Finally, the overall communication cost for all the voting ballots per project during the entire treasury period is depicted in Fig. **??**. In particular, for a treasury period with 5000 voters and 50 experts, the overall communication is approximately 20 MB per project.

10 RELATED WORK

10.1 Blockchain Governance

[JJ: This section is about blockchain based voting scheme, instead of voting in blockchain consensus. More likely about blockchain governance part.]

Khan *et al.*[**?**] analysed Blockchain governance using Nash equilibrium to predict the occurrence of a hard fork and indication of community's choice in terms of decision-making process in IT governance, they showed that Online Governance can achieve much better governance proposition than Off-chain governance such as Bitcoin.

10.2 Reputation in blockchain

[JJ: This section is about reputation in blockchain, including Proof of Reputation (PoF) or PoX+PoF.]

Reputation management scheme has been applied in peer-to-peer (P2P) networks to establish the trusted interaction among users in both centralised and decentralised ways. For example, ebay gathers reputation from sellers and buyers base on transactions in online commerce [**?**]. Reputation scores are also used to quantify users' active participation in file sharing according to other nodes [**?**]. In decentralised systems, many reputation-based consensus protocols are proposed to enable good nodes to create blocks and gain rewards, as a supplement to classic proof-of-work consensus. In these Proof of Reputation mining algorithms, miners are selected based on his reputation values [**? ? ?**]. Yu *et al.*[**?**] proposed RepCoin to define the mining power based on reputation scores in PoW, which can tolerate attacks compromising 51% attacks. RepuChain [**?**] adopts reputation to decide on the leader and validators in leader-based mining algorithm. In addition, reputation scores are also used in decentralised oracle network and prediction markets [**? ? ?**]. Basically, in these systems, the more reputation a user has, the more likely he will be assigned a task and the more rewards he can be awarded. If the user's outcome fails to match the consensus, his reputation scores will be redistributed to other users who math in accordance with the consensus.

11 CONCLUSION