Enabling Large-scale Blockchain Decision-making

1st Jiajie Zhang
School of Computing and Communication
Lancaster University
Lancaster, United Kindom
j.zhang41@lancaster.ac.uk

2nd Bingsheng Zhang
College of Computer Science and Technology
Zhejiang University, IOHK Research
Hangzhou, China
bingsheng@zju.edu.cn

3rd Zhengpeng Li
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Hamed Balogun
School of Computing and Communication
Lancaster University
Lancaster, United Kindom
h.balogun@lancaster.ac.uk

5th Roman Oliynykov
Department of information technology security
V.N.Karazin Kharkiv National University, IOHK Research
Kharkiv, Ukraine
roman.oliynykov@iohk.io

***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.

***Index Terms*—component, formatting, style, styling, insert**

I. 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.

A. Our Contributions

B. Organisation

II. Preliminaries

A. 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 $< a_1, \ldots, a_n >$ 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 $\{1, 2, 3\}$.

B. 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, $\mathsf{hash}(\mathsf{pk})$. 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 $\mathsf{Payload}$ 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 $\mathsf{Payload}$ field. Note that the verification data is not explicitly described for simplicity.

C. 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 $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the output of the environment $\mathcal{Z}$ when interacting with parties running the protocol Π and real-world adversary $\mathcal{A}$. Let $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote output of $\mathcal{Z}$ when running protocol ϕ interacting with the ideal functionality $\mathcal{F}$ and the ideal adversary $\mathcal{S}$.

Definition 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 $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.*

Definition 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 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 $\|\Pr[\mathcal{A}(N, z) = 1 \mid z = y^N \mod N^2] - \Pr[\mathcal{A}(N, z) = 1 \mid z = (N+1)^r \cdot y^N \mod N^2]\| \leq \mathsf{negl}(n)$, 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.*

D. 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 $\mathcal{C}$ such that for any $m_1, m_2 \in \mathcal{M}$ and $c_1, c_2 \in \mathcal{C}$ with $m_1 = \mathsf{Dec}_{\mathsf{sk}}(c_1)$ and $m_2 = \mathsf{Dec}_{\mathsf{sk}}(c_2)$, it holds that

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

Here $\odot$ and $\circ$ are the group operations over $\mathcal{C}$ 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.

III. The Model

A. Entities

Let $p^{(i)}, v^{(i)}, e^{(i)}, s^{(i)}, m^{(i)}$ be integers in $poly(\lambda)$. In each treasury period i, the stake holder might have one or more the following entity roles. To facilitate a better understanding, the dependence of entities is shown in Fig.1. Since we offload mainchain tasks to sidechain, one subtle implication is that our users have accounts on both mainchain and sidechain. 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 a subset of stake holders, 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.
- The treasury committee is a subset of experts, currently holding the top k reputation scores in different fields, denoted by: $\mathcal{T}^{(i)} := \{\mathsf{T}_{\{\mathsf{Fld}_j,1\}}^{(i)}, \ldots, \mathsf{T}_{\{\mathsf{Fld}_j,k\}}^{(i)}\}, \ldots, \{\mathsf{T}_{\{\mathsf{Fld}_s,1\}}^{(i)}, \ldots, \mathsf{T}_{\{\mathsf{Fld}_s,k\}}^{(i)}\}$. Field domain $\{\mathsf{Fld}_j, \ldots, \mathsf{Fld}_s\}$ is decided by current proposals fields. Treasury committee's main responsibility is to generate a short-list of proposals in the first voting stage on sidechain.
- The selection committee is a subset of voters $\mathcal{VT}^{(i)}$ who nominate the maintenance committee $\mathcal{MC}^{(i)}$ and hide identities of $\mathcal{MC}^{(i)}$ from adversary. In addition, selection committee will be in charge of tally verification from sidechain and tally enrolment on mainchian. They are selected by cryptography sortition, denoted by $\mathcal{SC}^{(i)} := \{\mathsf{SC}_1^{(i)}, \ldots, \mathsf{SC}_{s^{(i)}}^{(i)}\}$.
- The maintenance committee is a subset of voters $\mathcal{VT}^{(i)}$, denoted by $\mathcal{MC}^{(i)} := \{\mathsf{MC}_1^{(i)}, \ldots, \mathsf{MC}_{m^{(i)}}^{(i)}\}$. They are nominated by selection committee, the probability of being selected is based on the stake they locked on mainchain. Maintenance committee is selected to generate and maintain the keys on sidechain during the whole treasury system.

B. System Overview

The whole system consists of iterative treasury periods. For each period, it has three main stages, pre-voting stage, voting

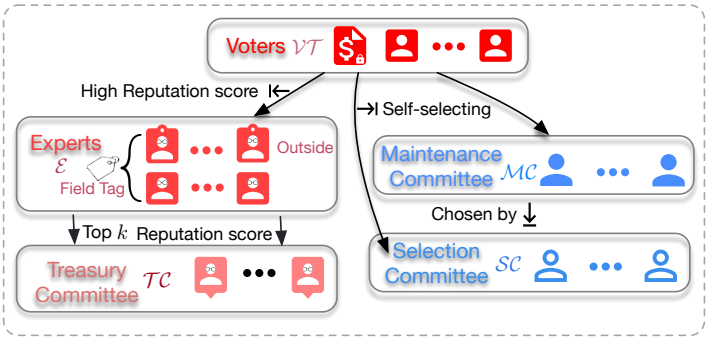

Fig. 1: Entities involved in our system

stage, and post-voting stage, as shown in Fig.2.

Pre-voting stage is foundation of our system, which encompasses a two-stage project proposing, and related participant registration on both mainchain and sidechain. 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 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, and register as voters on sidechain, their voting power will be proportional to the stakes they locked in this stage.

People who are well-known, highly regarded and reputable can also register as experts on sidechain and enrol on mainchain, we name them outside experts. They will choose one field tag to show their specific expertise, and register an account on mainchain for getting rewards. Outside experts will get initial reputation scores set by the system. 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 VII) exceed the threshold in a certain field, can also register as experts in pre-voting stage. We name them inside experts, in order to distinguish them from reputable outside experts. 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, 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 each field, and these experts will automatically sit on treasury committee.

Voting stage is composed of a two stage voting scheme: Nominating Stage and Endorsing Stage. Nominating stage involves part of a distribute key generation protocol (Cf. Sec V). In the designating phase of DKG, selection committee will be selected by cryptographic sortition, from the registered voters who are willing to be considered for selection to the committee. 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 cryptographic sortition 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. After been nominated by selection committee member, maintenance committee members should register on sidechain by providing proof of locked stakes on mainchain and proof of selected by electing committee member. Note that nothing in the information posted by selection committee on sidechain, or any actions made by maintenance committee should reveal maintenance committee's identities. In the KeyGen phase of DKG, which is the first treasury period, the first generation of maintenance committee will jointly setup key pairs. In the maintaining phase, maintenance committee will handover the keys to next generation of maintenance committee in next treasury period. Note that we only need to generate key once during operation of the entire system.

Simultaneously, 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, a number of committee members whose total reputation scores are more than 50% of all committee members' reputation scores are required to sign the short-list. 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 should end, the current generation of maintenance committee will decrypt the ballots and save ballots on sidechain.

Post-voting stage contains tally calculation stage and execution stage. After calculating tally automatically, selection committee will verify tally result, then write/enrol and sign the result on mainchian. All selection committee members should sign the result, the result will only be valid based on threshold signature. 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. VII).

C. Transaction Structure

Project proposal.

Project owner should register accounts on both mainchian and sidechain. In the first stage of project proposing, the project

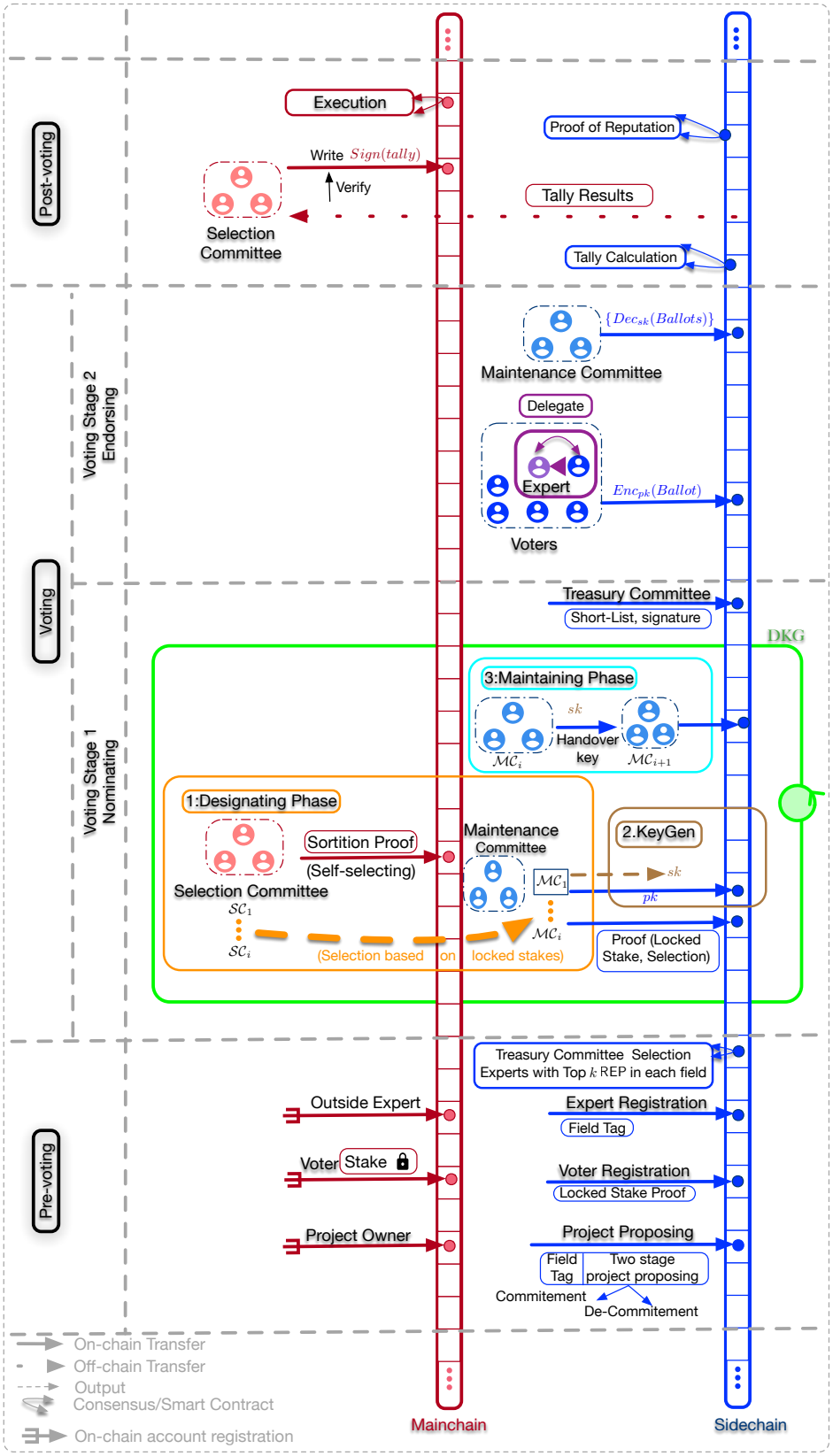

Fig. 2: System Overview

owners $\{\mathsf{O}_1, \ldots, \mathsf{O}_{p^{(i)}}\}$ 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 (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

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

where $\{\mathsf{In}^{(i)}\}_{i=1}^{n}$ are the input coins, and TCoin is a special output coin whose spending condition is defined in 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:

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

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 $\mathsf{Proof(REP)}$ is the proof of reputation scores on sidechain.

D. 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 $\mathsf{st}^{(i)} = \sum_{k=1}^{d-1}\sum_{j=1}^{\ell} S_{\{k,j\}}.\mathsf{Value}$ 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. $\mathsf{TS} := \sum_{k=1}^{d-1}\sum_{i=1}^{v_k}\sum_{j=1}^{\ell_{\{k,i\}}} S_{\{k,i,j\}}.\mathsf{Value}$ 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 $hash\text{-}len$ 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. 3. Ideally, we would like to model cryptographic sortition as a "perfect" 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.

The ideal functionality $\mathcal{F}_{Sortion}^{n}$

The functionality $\mathcal{F}_{Sortion}^{n}$ interacts with a set of voters $\{\mathsf{VT}_1, \ldots, \mathsf{VT}_n\}$ and adversary $\mathcal{S}$, sortition weight $w^{(i)} \in (0, 1)$.

Preparation:

- Upon receiving $(\textsc{Init}, \mathsf{sid})$ from voters $\mathsf{VT}^{(i)} \in \mathcal{VT}$, choose a random independent bit $b^{(i)}$ with $Pr[b^{(i)} = 1] = w^{(i)}$, and send message $(b^{(i)}, \mathsf{VT}^{(i)})$ to the adversary $\mathcal{S}$;
- Upon receiving $(\textsc{Refresh}, \mathsf{sid})$ from $\mathcal{S}$, last step will be repeated.
- Upon receiving $(\textsc{ChangeKey}, \mathsf{VT}^{(i)}, sk^{(i)}, pk^{(i)}\mathsf{sid})$ from $\mathcal{S}$, send $sk^{(i)}, pk^{(i)}\mathsf{sid})$ to $\mathsf{VT}^{(i)}$ and repeat the first step.

Lottery:

- Upon receiving $(\textsc{Reveal}, \mathsf{sid})$ from $\mathcal{S}$, send $b^{(i)}$ to $\mathsf{VT}^{(i)} \in \mathcal{VT}$.

Verification:

- Upon receiving $(\mathsf{Verify}, \mathsf{sid})$ from $\mathsf{U}_j \in \mathsf{U}$, send $\pi_{\mathsf{VT}^{(i)}}$ to U_j.

Fig. 3: The cryptographic sortition functionality

Our cryptographic sortition protocol is demonstrated in Fig. 4. 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 $i, j \in [v_d]$. 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 $(False, hash^{(i)}, \pi^{(i)})$

Fig. 4: The cryptographic sortition protocol $\Pi_{\text{TREASURY}}^{sk^{(i)},k,seed_k}$

At the beginning of the selection committee selection step, the selection committee of the previous treasury period jointly reveal the committed seed, seed. Let $\mathsf{st}^{(i)} = \sum_{j=1}^{\ell} S_j.\mathsf{Value}$ 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:

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

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

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

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., $\lambda' = \text{polylog}(\lambda)$) 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

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

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

E. 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. IV. 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 $\mathsf{C}_\ell \in \mathcal{C}$ 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 C_ℓ, $\ell \in [k]$ 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.

IV. The Two Stage Voting Scheme

V. Our DKG Protocol

Distributed Key Generation (DKG) is one of key components of our proposed system, where we authorise some voters as a series of maintaining committees for generating and maintaining the keys. Specifically, the first generation of maintaining committee in the first treasury period is in charge of generating keys, then handover the keys to next generation of maintaining committee. By doing so, we only need to run DKG once during the execution of the whole system, and each generation of maintaining committee only need to be online for once.

In our setting, identities of maintaining committee should remain anonymous. To achieve this goal, we introduce another committee named selection committee. In each treasury period d, we mainly have two groups of voters online in each period for DKG, Selection Committee $\mathcal{SC}_d$ and Maintenance Committee $\mathcal{MC}_d$. The main responsibility of $\mathcal{SC}_d$ is to nominate Maintenance Committee $\mathcal{MC}_d$ and hide their identities from the adversary by giving them temporary new identities (key pairs). Maintenance Committee $\mathcal{MC}_d$ will handle public and secret key pair generation, and key handover.

The whole DKG scheme can be divided into three phases as shown in Fig. 5, Designating Phase to select $\mathcal{SC}_d$ and $\mathcal{MC}_d$, Key Generation Phase to generate secret key shares and public key, Handover Phase to keep consistency of key and handover keys to next maintenance committee.

Designating Phase. In treasury period d, registered users who want to be selection committee will firstly execute $\mathcal{F}^n_{Sortion}$. After long enough blocks, the d-th selection committee $\mathcal{SC}_d$ is generated. Given the total locked stake coins S_d in period d, each member $\mathsf{SC}_{\{d,j\}}$ in $\mathcal{SC}_d$, will randomly select one coin from S_d coins. Note that if the owner of this coin set his VolM tag as 0, $\mathsf{SC}_{\{d,j\}}$ needs to select the coin again; otherwise the owner of the coin, named as $\mathsf{MC}_{\{d,j\}}$, will be one of the maintenance committee members $\mathcal{MC}_d$. The probability of being selected to sit in maintenance committee is proportional to the locked tokens on mainchain. Next, $\mathsf{SC}_{\{d,j\}}$, as the nominator of maintenance committee member $\mathcal{MC}_d$, will prepare ephemeral public and secret key pair for $\mathsf{MC}_{\{d,j\}}$. Given a cyclic group G of order q with a random generator g, $\mathsf{SC}_{\{d,j\}}$ will choose an integer $x_j \leftarrow \mathcal{Z}_q$ as the ephemeral secret key for $\mathsf{MC}_{\{d,j\}}$, and set ephemeral public key as $h_j := g^{x_j}$. Then $\mathsf{SC}_{\{d,j\}}$ post (h_j, ct_j) on sidechain, where $ct_j := Enc_{pk_j}(x_j)$, pk_j is the public key of $\mathsf{MC}_{\{d,j\}}$. Simultaneously, all users will monitor and try to decrypt messages on blockchain, if a user can decrypt the message posted by $\mathcal{SC}_d$, it means that this user is chosen as a member of $\mathcal{MC}_d$. Lastly, he can get his ephemeral secret key x_j by decrypting ct_j with his own long-term secret key sk_j.

Key Generation Phase. In the *first* period, the first generation of maintenance committee $\mathcal{MC}_1$ will jointly generate the secret key x and public key $y = g^x$. Specifically, each member $\mathsf{MC}_{\{1,i\}}$ in $\mathcal{MC}_1$ will choose a random z_i, and generate two polynomial functions over Z_q of degree $t_1 := \lfloor \frac{n_1}{2} \rfloor + 1$, where n_1 is the size of $\mathbf{V_1}$: $F_i(z) = a_{i,0} + a_{i,1}z + \cdots + a_{i,t_1}z^{t_1}$, $F'_i(z) = b_{i,0} + b_{i,1}z + \cdots + b_{i,t_1}z^{t_1}$. The free term $a_{i,0}$ of first polynomial function $F_i()$ is z_i.

Next, $\mathsf{MC}_{\{1,i\}}$ commits coefficients $\{a_{i,0}, \cdots, a_{i,t_1}\}$ of $F_i()$

by posting $\Delta_{i,k'} = g^{a_{i,k'}} h^{b_{i,k'}}$ on sidechain for $k' \in [0, t_1]$, h is the commitment key. Then $\mathsf{MC}_{\{1,i\}}$ computes $s_{i,j} := F_i(j)$ and $s'_{i,j} := F'_i(j)$ for $j \in [M_1]$. Afterwards, $\mathsf{MC}_{\{1,i\}}$ shares $s_{i,j}$ for $j \in [M_1]$ among M_2 maintenance committee members $\mathsf{MC}_{\{2,i\}}$, with t_2-out-of-M_2 secret sharing, where $t_2 = \lfloor \frac{M_2}{2} \rfloor + 1$. Later, $\mathsf{MC}_{\{1,i\}}$ chooses j polynomial functions $G_{i,j}(z) = s_{i,j} + c_{i,j,1} z + \cdots + c_{i,j,t_2} z^{t_2}$ for $j \in [M_1]$.

Without loss of generality, we assume that p is a perfect power of 2 (*e.g.* $p = 2^{30}$). $\mathsf{MC}_{\{1,i\}}$ computes $G_{i,j}(m)$ for evaluation points $m = 1, \cdots, M_2$, and encrypts $G_{i,j}(m)$ by ElGamal encryption. Since the message space in ElGamal encryption system lies in the group G, $\mathsf{MC}_{\{1,i\}}$ needs to map $G_{i,j}(m)$ to elements of G. First, $\mathsf{MC}_{\{1,i\}}$ needs to divide $G_{i,j}(m)$ into ℓ smaller elements $\{d^{(0)}_{i,j,m}, \cdots, d^{(\ell)}_{i,j,m}\}$, where $\sum_{k=1}^{\ell} d^{(k)}_{i,j,m} p^k = G_{i,j}(m)$. Then $\mathsf{MC}_{\{1,i\}}$ encrypts $g^{d^{(k)}_{i,j,m}}$ with the ephemeral public key h_m of the party who owns the m-th seat in MC_2. Specifically, $\mathsf{MC}_{\{1,i\}}$ chooses ℓ integers $\{y^{(1)}_m, \cdots, y^{(\ell)}_m\}$ from $\{1, \cdots, q-1\}$ and computes $c1^{(k)}_{i,j,m} := g^{y^{(k)}_m}$, $c2^{(k)}_{i,j,m} := g^{d^{(k)}_{i,j,m}} \cdot h_m^{y^{(k)}_m}$. Then $\mathsf{MC}_{\{1,i\}}$ posts $\{c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}\}$ on sidechain.

$\mathsf{MC}_{\{1,i\}}$ generates the NIZK proof $\pi_{i,j}$ (Cf. Sec.VI-A) with witness $(s_{i,j}, s'_{i,j})$ to show that 1) $g^{s_{i,j}} h^{s'_{i,j}} = \prod_{k=0}^{t_1} (\Delta_{i,k})$; This is easy to understand since $g^{s_{i,j}} h^{s'_{i,j}} = \prod_{k=0}^{t_1} (\Delta_{i,k}) = g^{\sum_{k=0}^{t_1} a_{i,k}} h^{\sum_{k=0}^{t_1} b_{i,k}}$. 2) $\mathsf{MC}_{\{1,i\}}$ uses same $s_{i,j}$ to compute $G_{i,j}(0)$ as $F_i(j)$, which shows that $\mathsf{MC}_{\{1,i\}}$ has reshared its shares correctly; 3) $\{c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}\}$ indeed encrypts $g^{d^{(k)}_{i,j,m}}$ For $k = 1, \cdots, \ell$. $\mathsf{MC}_{\{1,i\}}$ chooses a new pair of long-term keys (pk'_i, sk'_i) for himself, posts $\{\pi_{i,j}, pk'_i, i\}$ on the blockchain, then erases all the protocol secrets he holds.

Maintaining Phase. Starting from the *second* treasury period, the value of key generated in the first period should be maintained and securely handed to next maintenance committee.

In period $d+1$, $\mathsf{MC}_{\{d+1, f \in [M_{d+1}]\}}$ in $\mathcal{MC}_{d+1}$ collects all the messages $\{c1^{(k)}_{i,f}, c2^{(k)}_{i,f}\}^{k \in [\ell]}_{i \in [M_d]}$ encrypted by his ephemeral public key h_f posted on the blockchain by $\mathcal{MC}_d$, and decrypts them with x_f to get $\{g^{d^{(k)}_{i,f}}\}^{k \in [\ell]}$. Since $\{d^{(k)}_{i,f}\}^{k \in [\ell]}$ are small elements, $\mathsf{MC}_{\{d+1,f\}}$ is able to get $\{d^{(k)}_{i,f}\}^{k \in [\ell]}$ from $\{g^{d^{(k)}_{i,f}}\}^{k \in [\ell]}$. Afterwards, $\mathsf{MC}_{\{d+1,f\}}$ can compute $s_{i,f} = \sum_{k=1}^{\ell} d^{(k)}_{i,f} p^k$.

$\mathsf{MC}_{\{d+1,f\}}$ then checks the proofs π_i for $i = \{1, \cdots, M_d\}$ provided by $\mathcal{MC}_d$ on sidechain, and constructs a set QUAL_d which is composed of the seat number of all qualified parties who provides right proof. Note that for all honest parties in $\mathcal{MC}_{d+1}$, set QUAL_d will be the same. $\mathsf{MC}_{\{d+1,f\}}$ computes $m_f := \sum_{i \in \mathsf{QUAL}_d} s_{i,f}$. Currently, the global secret key $x := \sum \lambda_f m_f$ can be computed by interpolation, where λ_f are appropriate Lagrange interpolation coefficients. Next, $\mathsf{MC}_{\{d+1,f\}}$ re-shares m_f to next maintenance committee by t_{d+2}-out-of-M_{d+2} secret sharing, where $t_{d+2} := \lfloor \frac{M_{d+2}}{2} \rfloor + 1$. $\mathsf{MC}_{\{d+1,f\}}$ chooses two t_{d+2} degree polynomial functions $\hat{F}_f(z) = c'_0 + c'_1 z + \cdots + c'_{t_{d+2}} z^{t_{d+2}}$ and $\hat{G}_f(z) = c''_0 + c''_1 z + \cdots + c''_{t_{d+2}} z^{t_{d+2}}$, the free term of $\hat{F}_f(z)$ is m_f. $\mathsf{MC}_{\{d+1,f\}}$ computes $\hat{F}_f(i)$ and $\hat{G}_f(i)$ for all parties in next maintenance committee, where $i \in [M_{d+2}]$, and commits $\{c'_k\}_{k \in [0, t_{d+2}]}$ on sidechain by posting $\{\Delta_{f,k} := g^{c'_k} \cdot h^{c''_k}\}_{k \in [0, t_{d+2}]}$.

Subsequently, $\mathsf{MC}_{\{d+1,f\}}$ divide $\hat{F}_f(t)$ into ℓ smaller elements $\{m^{(1)}_{f,t}, \cdots, m^{(\ell)}_{f,t}\}$, where $\sum_{k=1}^{\ell} m^{(k)}_{f,t} p^k = \hat{F}_f(t)$. Then $\mathsf{MC}_{\{d+1,f\}}$ encrypts $g^{m^{(k)}_{f,t}}$ with the ephemeral public key h_t of the party who owns the t-th seat in $\mathcal{MC}_{d+2}$. Firstly, $\mathsf{MC}_{\{d+1,f\}}$ chooses ℓ integers $\{y^{(1)}_t, \cdots, y^{(\ell)}_t\}$ from $\{1, \cdots, q-1\}$ and computes $c1^{(k)}_t := g^{y^{(k)}_t}$, $c2^{(k)}_t := g^{m^{(k)}_{f,t}} \cdot h_t^{y^{(k)}_t}$. Then P_i posts $\{c1^{(k)}_t, c2^{(k)}_t\}$ on the sidechain.

After that, $\mathsf{MC}_{\{d+1,f\}}$ generates the NIZK proof π_f (Cf. Sec.VI-B) to show that: 1) he indeed gets $d^{(k)}_{i,f}$ from decrypting $\{c1^{(k)}_{i,f} := g^{y^{(k)}_f}, c2^{(k)}_{i,f} := g^{d^{(k)}_{i,f}} \cdot h_f^{y^{(k)}_f}\}_{k \in [\ell]}$; 2) he uses same m_f computed from the qualified parties to compute $\hat{F}_f(0)$. Lastly, $\mathsf{MC}_{\{d+1,f\}}$ chooses a new pair of long-term keys (pk'_f, sk'_f) for himself, posts $\{\pi_f, pk'_f, f\}$ on the blockchain, then erases all the protocol secrets he holds.

VI. Zero Knowledge Proof

A. DKG ZK proof

In this step, we propose a honest verifier ZK (HVZK) proof for DKG phase that allows the prover P to show the verifier V that $\{c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}\}$ indeed encrypts $g^{d^{(k)}_{i,j,m}}$ For $k = 1, \cdots, \ell$, and he uses the same $s_{i,j}$ got from $F_i(j)$ to compute $G_{i,j}(0)$.

Lemma 1. *Assume the DDH problem is hard. The protocol described in Fig.7 is a honest verifier zero-knowledge argument of knowledge of* $g^{d^{(k)}_{i,j,m}}$, $s_{i,j}$, $s'_{i,j}$, $\{y^{(1)}_m, \cdots y^{(\ell)}_m\}$ *such that* $c1^{(k)}_{i,j,m} := g^{y^{(k)}_m}$, $c2^{(k)}_{i,j,m} := g^{d^{(k)}_{i,j,m}} \cdot h_m^{y^{(k)}_m}$, $s_{i,j} = F_i(j) = G_{i,j}(0)$ *where* $k \in \{1, \cdots, \ell\}$.

Proof. **Perfect completeness.** Firstly, as shown in Fig. 7, Prover P transforms the ciphertexts $\{c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}\}^{\ell, M_2}_{k=1,m=1}$ encrypted by the public key $\{h_m\}^{M_2}_{m=1}$ to $\{r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\}^{\ell, M_2}_{k=1,m=1}$ encrypted by his own key pair $\{\tau, F\}$. Our first step is to verify the correctness of this transformation; namely, $(c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m})$ and $r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}$ encrypts the same message, respectively. For efficiency, we use (powers of) λ_m to combine them for batching verification. For the first verification equation, it is easy to see that

$$A1 \cdot (C1/R1)^e = \Pi_{m=1}^{M_2} g^{\omega_m - v_m} \cdot g^{e \cdot (\sum_{m=1}^{M_2} (Y_m - X_m))} = g^{v1 - v2}.$$

Next, for each $m \in M_2$, verifier V can compute $C2_m, R2_m$ by concatenating ℓ ciphertexts together. Hence, we have:

$$A2_m \cdot (C2_m / R2_m)^e = h_m^{\omega_m + Y_m \cdot e} / F^{v_m + X_m \cdot e} = h_m^{v1_m} / F^{v2_m} \quad .$$

To prove that $\{c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}\}^{M_2}_{m=1}$ correctly encrypted $g^{d^{(k)}_{i,j,m}}$, we can reduce this problem to prove that the prod-

Distributed Key Generation Π_{DKG}

Designating Phase

In the d-th treasury period, do the followings:

- Get $\mathcal{SC}_d := \{\mathsf{SC}_{\{d,1\}}, \ldots, \mathsf{SC}_{\{d,s_i\}}\}$ from $\mathcal{F}^n_{Sortion}$.
- For $\mathsf{SC}_{\{d,j\}} \in \mathcal{SC}_d$, do the followings:
  - Select $\mathsf{MC}_{\{d,j\}}$ based on locked stakes.
  - Randomly select $x_j \leftarrow \mathcal{Z}_q$, set $h_j := g^{x_j}$.
  - Set $ct_j := Enc_{pk_j}(x_j)$.
  - Post (h_j, ct_j, j) on sidechain.
- For $\mathsf{MC}_{\{d,j\}} \in \mathcal{MC}_d$, do the following:
  - Get h_j from sidechain.
  - Get $x_j = Enc_{sk_j}(ct_j)$ by decrypting.

Key Generation Phase

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

- Randomly choose $z_i \leftarrow \mathcal{Z}_q$.
- Generate two t_1 degree polynomial functions over $\mathcal{Z}_q$, where $t_1 := \lfloor \frac{M_1}{2} \rfloor + 1$, $a_{i,0} := z_i$:
  $F_i(z) = a_{i,0} + a_{i,1}z + \cdots + a_{i,t_1}z^{t_1}$, $F'_i(z) = b_{i,0} + b_{i,1}z + \cdots + b_{i,t_1}z^{t_1}$.
- Post $\Delta_{i,k'} = g^{a_{i,k'}} h^{b_{i,k'}}$ on sidechain for $k' \in [0, t_1]$.
- Compute $s_{i,j} := F_i(j)$ and $s'_{i,j} := F'_i(j)$ for $j \in [M_1]$.
- Choose j-th t_2 degree polynomial function $G_{i,j}(z) = s_{i,j} + c_{i,j,1}z + \cdots + c_{i,j,t_2}z^{t_2}$, where $t_2 := \lfloor \frac{M_2}{2} \rfloor + 1$.
- For $m \in [M_2]$, do the following.
  - Compute $G_{i,j}(m)$, and slice $G_{i,j}(m)$ into ℓ smaller elements $\{d^{(0)}_{i,j,m}, \cdots, d^{(\ell)}_{i,j,m}\}$, where $\sum_{k=1}^{\ell} d^{(k)}_{i,j,m} p^k = G_{i,j}(m)$, p is a perfect power of 2.
  - Choose ℓ integers $\{y^{(1)}_m, \cdots, y^{(\ell)}_m\}$ from $\{1, \cdots, q-1\}$.
  - Compute $c1^{(k)}_{i,j,m} := g^{y^{(k)}_m}$, $c2^{(k)}_{i,j,m} := g^{d^{(k)}_{i,j,m}} \cdot h^{y^{(k)}_m}_m$.
  - Post $\{c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}\}$ on sidechain.
- Generate NIZK proof π_i.
- Choose a new pair of long-term keys (pk'_i, sk'_i) for himself, post $\{\pi_{i,j}, pk'_i, i\}$ on sidechain, then erases all the protocol secrets he holds.

Maintaining Phase

In the $d+1$ treasury period, for all committee members $\{\mathsf{MC}_{\{d+1,f\}}\}_{f=1}^{M_{d+1}}$ in $\mathcal{MC}_{d+1}$, do the following:

- Collect all the messages $\{c1^{(k)}_{i,f}, c2^{(k)}_{i,f}\}^{\ell, M_d}_{k=1,i=1}$ encrypted by h_f posted on sidechain by $\mathcal{MC}_d$.
- Decrypt them with x_f to get $\{g^{d^{(k)}_{i,f}}\}^{\ell}_{k=1}$, and compute $\{d^{(k)}_{i,f}\}^{\ell}_{k=1}$ from $\{g^{d^{(k)}_{i,f}}\}^{\ell}_{k=1}$.
- Compute $s_{i,f} := \sum_{k=1}^{\ell} d^{(k)}_{i,f} p^k$.
- Check the proofs $\{\pi_i\}_{i=1}^{M_d}$ provided by $\mathcal{MC}_d$, and construct a set QUAL_d with size of N_{QUAL_d}.
- Compute $m_f := \sum_{i=1}^{N_{\mathsf{QUAL}_d}} s_{i,f}$.
- Choose two t_{d+2} degree polynomial functions, where $t_{d+2} := \lfloor \frac{M_{d+2}}{2} \rfloor + 1$, $c'_0 = m_f$:
  $\hat{F}_f(z) = c'_0 + c'_1 z + \cdots + c'_{t_{d+2}} z^{t_{d+2}}$
  $\hat{G}_f(z) = c''_0 + c''_1 z + \cdots + c''_{t_{d+2}} z^{t_{d+2}}$
- Compute $\hat{F}_f(i), \hat{G}_f(i)$ for all $i \in [n_{d+2}]$.
- Post $\{\Delta_{f,k} := g^{c'_k} \cdot h^{c''_k}\}_{k=0}^{t_{d+2}}$ on sidechain.
- Divide $\hat{F}_f(t)$ into ℓ smaller elements $\{d^{(1)}_{f,t}, \cdots, d^{(\ell)}_{f,t}\}$, where $\sum_{k=1}^{\ell} d^{(k)}_{f,t} p^k = \hat{F}_f(t)$.
- Choose ℓ integers $\{y^{(1)}_t, \cdots, y^{(\ell)}_t\}$ from $\{1, \cdots, q-1\}$ and compute $c1^{(k)}_{f,t} := g^{y^{(k)}_t}$, $c2^{(k)}_{f,t} := g^{d^{(k)}_{f,t}} \cdot h^{y^{(k)}_t}_t$.
- Post $\{c1^{(k)}_{f,t}, c2^{(k)}_{f,t}\}$ on sidechain.
- Generates NIZK proof π_f.
- Choose a new pair of long-term keys (pk'_f, sk'_f) for himself, posts $\{\pi_f, pk'_f, f\}$ on sidechain, then erases all the protocol secrets he holds.

Fig. 5: Distributed Key Generation

DKG ZK argument

Common Knowledge: $u_{i,j,1}, \cdots, u_{i,j,t_2}$ are distinct points in $\mathcal{Z}_q$ defining Lagrange polynomial $G_{i,j}(z)$ such that $G_{i,j}(0) = s_{i,j}$, $\Delta_{i,k'} = g^{a_{ik'}} h^{b_{ik'}}$ for $k' \in [0, t_1]$.
Common Reference String: Commitment key h.

Statement: Public key $h_m = g^{x_m}$ and $c1^{(k)}_{i,j,m} := g^{y^{(k)}_m}$, $c2^{(k)}_{i,j,m} := g^{d^{(k)}_{i,j,m}} \cdot (h_m)^{y^{(k)}_m}$ for $k \in [\ell]$, $m \in [M_2]$,

Witness: $\{g^{d^{(k)}_{i,j,m}}\}^{\ell,M_2}_{k=1,m=1}$, $s_{i,j}$, $s'_{i,j}$, $\{y^{(1)}_m, \cdots y^{(\ell)}_m\}$ from $\mathcal{Z}_q$.
Protocol:

- The prover P does the followings:
  - Select $\tau \leftarrow \mathcal{Z}_q$;
  - Compute $F := g^\tau$; %Prover sets his pk as F, and sk as τ.
  - For $m = 1, \cdots, M_2$
    - For $k = 1, \cdots, \ell$
      - Select $\delta^{(k)}_m \leftarrow \mathcal{Z}_q$;
      - Compute $r1^{(k)}_{i,j,m} := g^{\delta^{(k)}_m}$, $r2^{(k)}_{i,j,m} := g^{d^{(k)}_{i,j,m}} \cdot (F)^{\delta^{(k)}_m}$.
  - Set ~~$\lambda \leftarrow \mathsf{hash}(\langle c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}, r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\rangle^{\ell,M_2}_{k=1,m=1})$~~ $\{\lambda_m\}^{M_2}_{m=1} \leftarrow \mathsf{hash}(\langle c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}, r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\rangle^{\ell,M_2}_{k=1,m=1})$
  - For $m = 1, \cdots, M_2$
    - Select $\omega_m, \upsilon_m \leftarrow \mathcal{Z}_q$;
    - Compute ~~$Y_m := y^{(1)}_m \lambda + y^{(2)}_m \lambda^2 + \cdots + y^{(\ell)}_m \lambda^\ell, X_m := \delta^{(1)}_m \lambda + \delta^{(2)}_m \lambda^2 + \cdots + \delta^{(\ell)}_m \lambda^\ell$~~
      $Y_m := y^{(1)}_m \lambda_m + y^{(2)}_m \lambda^2_m + \cdots + y^{(\ell)}_m \lambda^\ell_m, X_m := \delta^{(1)}_m \lambda_m + \delta^{(2)}_m \lambda^2_m + \cdots + \delta^{(\ell)}_m \lambda^\ell_m$;
    - Compute ~~$A1_m := g^{\omega_m - \upsilon_m}$~~, $A2_m := h^{\omega_m}_m / F^{\upsilon_m}$;
    - ~~Select $s_m \leftarrow G, t_m \leftarrow \mathcal{Z}_q$;~~
    - ~~Compute $E1_m := g^{t_m}$, $E2_m := s_m F^{t_m}$;~~
  - $A1 := \Pi^{M_2}_{m=1} g^{\omega_m - \upsilon_m}$
  - Select $s \leftarrow G, t, a, b, c \leftarrow \mathcal{Z}_q$;
  - Compute $E1 := g^t, E2 := sF^t$;
  - Compute $T_1 := g^a \cdot h^b$, $T_2 := g^c$, $T_3 := g^a \cdot F^c$;
- Set $e \leftarrow \mathsf{hash}(\langle c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}, r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\rangle^{\ell,M_2}_{k=1,m=1}, E1, E2, A1, T_1, T_2, T_3, \langle A2_m\rangle^{M_2}_{m=1})$;
- For $m = 1, \cdots, M_2$
  - Compute $v1_m := \omega_m + Y_m \cdot e$, $v2_m := \upsilon_m + X_m \cdot e$;
  - ~~Compute $\alpha_m := s_m \cdot ((g^{d^{(1)}_{i,j,m}})^\lambda \cdots (g^{d^{(\ell)}_{i,j,m}})^{\lambda^\ell})^e$, $\beta_m := t_m + X_m \cdot e$;~~
  - Compute $z_m := \sum^\ell_{k=1} \delta^{(k)}_m p^k$;
- Compute $v1 := \sum^{M_2}_{m=1} v1_m, v2 := \sum^{M_2}_{m=1} v2_m$;
- Compute $X := \sum^{M_2}_{m=1} X_m$;
- Compute $\alpha := s \cdot (\Pi^{M_2}_{m=1} \Pi^\ell_{k=1} (g^{d^{(k)}_{i,j,m}})^{\lambda^k_m})^e, \beta := t + X \cdot e$;
- Compute $z := \sum^{t_2}_{m'=1} z_{m'} u_{i,j,m'}$;
- Compute $y_1 := a + s_{i,j} \cdot e$, $y_2 := b + s'_{i,j} \cdot e$, $y_3 := c + z \cdot e$;
- $P \rightarrow V : (T_1, T_2, T_3, \langle E1_m, E2_m, A1_m, A2_m, v1_m, v2_m, \alpha_m, \beta_m\rangle^{M_2}_{m=1}, \{r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\}^{\ell,M_2}_{k=1,m=1}, y_1, y_2, y_3)$.

Verification:

- Verifier V computes the followings:
  - $\{\lambda_m\}^{M_2}_{m=1} \leftarrow \mathsf{hash}(\langle c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}, r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\rangle^{\ell,M_2}_{k=1,m=1})$;
  - $e \leftarrow \mathsf{hash}(\langle c1^{(k)}_{i,j,m}, c2^{(k)}_{i,j,m}, r1^{(k)}_{i,j,m}, r2^{(k)}_{i,j,m}\rangle^{\ell,M_2}_{k=1,m=1}, E1, E2, A1, T_1, T_2, T_3, \langle A2_m\rangle^{M_2}_{m=1})$;
  - For $m = 1, \cdots, M_2$
    - ~~$C1_m = (c1^{(1)}_{i,j,m})^\lambda \cdot (c1^{(2)}_{i,j,m})^{\lambda^2} \cdots (c1^{(\ell)}_{i,j,m})^{\lambda^\ell}$, $R1_m = (r1^{(1)}_{i,j,m})^\lambda \cdot (r1^{(2)}_{i,j,m})^{\lambda^2} \cdots (r1^{(\ell)}_{i,j,m})^{\lambda^\ell}$;~~
    - ~~$C2_m = (c2^{(1)}_{i,j,m})^\lambda \cdot (c2^{(2)}_{i,j,m})^{\lambda^2} \cdots (c2^{(\ell)}_{i,j,m})^{\lambda^\ell}$, $R2_m = (r2^{(1)}_{i,j,m})^\lambda \cdot (r2^{(2)}_{i,j,m})^{\lambda^2} \cdots (r2^{(\ell)}_{i,j,m})^{\lambda^\ell}$;~~
      $C2_m = \Pi^\ell_{k=1} (c2^{(k)}_{i,j,m})^{\lambda^k_m}, R2_m = \Pi^\ell_{k=1} (r2^{(k)}_{i,j,m})^{\lambda^k_m}$
    - $H1_m := \Pi^\ell_{k=1} ((r1_{i,j,m})^{(k)})^{p^k}$, $H2_m := \Pi^\ell_{k=1} ((r2_{i,j,m})^{(k)})^{p^k}$;
    - ~~$z_m := \sum^\ell_{k=1} \delta^{(k)}_m p^k$.~~[JJ: V doesn't need to compute.]
  - $C1 = \Pi^{M_2}_{m=1} \Pi^\ell_{k=1} (c1^{(k)}_{i,j,m})^{\lambda^k_m}$;
  - $R1 := \Pi^{M_2}_{m=1} \Pi^\ell_{k=1} (r1^{(k)}_{i,j,m})^{\lambda^k_m}$, $R2 := \Pi^{M_2}_{m=1} R2_m$;
  - ~~$z := \sum^{t_2}_{m'=1} z_{m'} u_{i,j,m'}$~~[JJ: V doesn't need to compute.]
  - $W_1 := \Pi^{t_2}_{m'=1} (H1_{m'})^{u_{i,j,m'}}$, $W_2 := \Pi^{t_2}_{m'=1} (H2_{m'})^{u_{i,j,m'}}$;
  - $D := g^{s_{i,j}} h^{s_{i,j}} = \prod^{t_1}_{k=0} (\Delta_{i,k})^{j^k}$;[JJ: no typo here]
  - $v1 := \sum^{M_2}_{m=1} v1_m, v2 := \sum^{M_2}_{m=1} v2_m$;
- Verifier V check the followings:
  - For $m = 1, \cdots, M_2$
    - ~~$A1_m \cdot (C1_m / R1_m)^e = g^{v1_m - v2_m}$~~, $A2_m \cdot (C2_m / R2_m)^e = h^{v1_m}_m / F^{v2_m}$. %Correctness of Ciphertext Conversion.
    - ~~$E1_m \cdot (R1_m)^e = g^{\beta_m}$; $E2_m \cdot (R2_m)^e = \alpha_m \cdot (F)^{\beta_m}$.~~
  - $A1 \cdot (C1/R1)^e = g^{v1 - v2}$.%Correctness of Ciphertext Conversion.
  - $E1 \cdot (R1)^e = g^\beta$; $E2 \cdot (R2)^e = \alpha \cdot (F)^\beta$;%Correctness of Decryption.
  - $g^{y_1} \cdot h^{y_2} = D^e \cdot T_1$, $g^{y_3} = W^e_1 \cdot T_2$, $g^{y_1} \cdot (F)^{y_3} = W^e_2 \cdot T_3$;%Correctness of Key Handover.

Fig. 6: DKG ZK argument

DKG ZK argument

Common Knowledge: $u_{i,j,1}, \cdots, u_{i,j,t_2}$ are distinct points in $\mathcal{Z}_q$ defining Lagrange polynomial $G_{i,j}(z)$ such that $G_{i,j}(0) = s_{i,j}$, $\Delta_{i,k'} = g^{a_{ik'}} h^{b_{ik'}}$ for $k' \in [0, t_1]$.
Common Reference String: Commitment key h.
Statement: Public key $h_m = g^{x_m}$ and $c1_{i,j,m}^{(k)} := g^{y_m^{(k)}}$, $c2_{i,j,m}^{(k)} := g^{d_{i,j,m}^{(k)}} \cdot (h_m)^{y_m^{(k)}}$ for $k \in [\ell]$, $m \in [M_2]$,
Witness: $\{g^{d_{i,j,m}^{(k)}}\}_{k=1,m=1}^{\ell,M_2}$, $s_{i,j}$, $s'_{i,j}$, $\{y_m^{(1)}, \cdots y_m^{(\ell)}\}$ from $\mathcal{Z}_q$.
Protocol:

- The prover P does the followings:
  - Select $\tau \leftarrow \mathcal{Z}_q$;
  - Compute $F := g^{\tau}$; %Prover sets his pk as F, and sk as τ.
  - For $m = 1, \cdots, M_2$
    - For $k = 1, \cdots, \ell$
      - Select $\delta_m^{(k)} \leftarrow \mathcal{Z}_q$;
      - Compute $r1_{i,j,m}^{(k)} := g^{\delta_m^{(k)}}$, $r2_{i,j,m}^{(k)} := g^{d_{i,j,m}^{(k)}} \cdot (F)^{\delta_m^{(k)}}$.
  - Set $\{\lambda_m\}_{m=1}^{M_2} \leftarrow \mathsf{hash}(\langle c1_{i,j,m}^{(k)}, c2_{i,j,m}^{(k)}, r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)} \rangle_{k=1,m=1}^{\ell,M_2})$
  - For $m = 1, \cdots, M_2$
    - Select $\omega_m, \upsilon_m \leftarrow \mathcal{Z}_q$;
    - Compute $Y_m := y_m^{(1)}\lambda_m + y_m^{(2)}\lambda_m^2 + \cdots + y_m^{(\ell)}\lambda_m^{\ell}, X_m := \delta_m^{(1)}\lambda_m + \delta_m^{(2)}\lambda_m^2 + \cdots + \delta_m^{(\ell)}\lambda_m^{\ell}$;
    - Compute $A2_m := h_m^{\omega_m} / F^{\upsilon_m}$;
  - $A1 := \Pi_{m=1}^{M_2} g^{\omega_m - \upsilon_m}$;
  - Select $s \leftarrow G, t, a, b, c \leftarrow \mathcal{Z}_q$;
  - Compute $E1 := g^t, E2 := sF^t$;
  - Compute $T_1 := g^a \cdot h^b$, $T_2 := g^c$, $T_3 := g^a \cdot F^c$;
- Set $e \leftarrow \mathsf{hash}(\langle c1_{i,j,m}^{(k)}, c2_{i,j,m}^{(k)}, r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)} \rangle_{k=1,m=1}^{\ell,M_2}, E1, E2, A1, T_1, T_2, T_3, \langle A2_m \rangle_{m=1}^{M_2})$;
- For $m = 1, \cdots, M_2$
  - Compute $v1_m := \omega_m + Y_m \cdot e$, $v2_m := \upsilon_m + X_m \cdot e$;
  - Compute $z_m := \sum_{k=1}^{\ell} \delta_m^{(k)} p^k$;
- Compute $v1 := \sum_{m=1}^{M_2} v1_m, v2 := \sum_{m=1}^{M_2} v2_m$, $X := \sum_{m=1}^{M_2} X_m$;
- Compute $\alpha := s \cdot (\Pi_{m=1}^{M_2} \Pi_{k=1}^{\ell} (g^{d_{i,j,m}^{(k)}})^{\lambda_m^k})^e, \beta := t + X \cdot e$;
- Compute $z := \sum_{m'=1}^{t_2} z_{m'} u_{i,j,m'}$;
- Compute $y_1 := a + s_{i,j} \cdot e$, $y_2 := b + s'_{i,j} \cdot e$, $y_3 := c + z \cdot e$;
- $P \rightarrow V : (E1, E2, A1, T_1, T_2, T_3, \alpha, \beta, \langle A2_m, v1_m, v2_m, \rangle_{m=1}^{M_2}, \{r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)}\}_{k=1,m=1}^{\ell,M_2}, y_1, y_2, y_3)$.

Verification:

- Verifier V computes the followings:
  - $\{\lambda_m\}_{m=1}^{M_2} \leftarrow \mathsf{hash}(\langle c1_{i,j,m}^{(k)}, c2_{i,j,m}^{(k)}, r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)} \rangle_{k=1,m=1}^{\ell,M_2})$
  - $e \leftarrow \mathsf{hash}(\langle c1_{i,j,m}^{(k)}, c2_{i,j,m}^{(k)}, r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)} \rangle_{k=1,m=1}^{\ell,M_2}, E1, E2, A1, T_1, T_2, T_3, \langle A2_m \rangle_{m=1}^{M_2})$;
  - For $m = 1, \cdots, M_2$
    - $C2_m = \Pi_{k=1}^{\ell} (c2_{i,j,m}^{(k)})^{\lambda_m^k}, R2_m = \Pi_{k=1}^{\ell} (r2_{i,j,m}^{(k)})^{\lambda_m^k}$
    - $H1_m := \Pi_{k=1}^{\ell} ((r1_{i,j,m})^{(k)})^{p^k}$, $H2_m := \Pi_{k=1}^{\ell} ((r2_{i,j,m})^{(k)})^{p^k}$;
  - $C1 = \Pi_{m=1}^{M_2} \Pi_{k=1}^{\ell} (c1_{i,j,m}^{(k)})^{\lambda_m^k}$;
  - $R1 := \Pi_{m=1}^{M_2} \Pi_{k=1}^{\ell} (r1_{i,j,m}^{(k)})^{\lambda_m^k}$, $R2 := \Pi_{m=1}^{M_2} R2_m$;
  - $W_1 := \Pi_{m'=1}^{t_2} (H1_{m'})^{u_{i,j,m'}}$, $W_2 := \Pi_{m'=1}^{t_2} (H2_{m'})^{u_{i,j,m'}}$;
  - $D := g^{s_{i,j}} h^{s'_{i,j}} = \prod_{k=0}^{t_2} (\Delta_{i,k})$.
  - $v1 := \sum_{m=1}^{M_2} v1_m, v2 := \sum_{m=1}^{M_2} v2_m$;
- Verifier V check the followings:
  - $A1 \cdot (C1/R1)^e = g^{v1-v2}$.%Correctness of Ciphertext Conversion.
  - For $m = 1, \cdots, M_2$
    - $A2_m \cdot (C2_m / R2_m)^e = h_m^{v1_m} / F^{v2_m}$. %Correctness of Ciphertext Conversion.
  - $E1 \cdot (R1)^e = g^{\beta}$, $E2 \cdot (R2)^e = \alpha \cdot (F)^{\beta}$;%Correctness of Decryption.
  - $g^{y_1} \cdot h^{y_2} = D^e \cdot T_1$, $g^{y_3} = W_1^e \cdot T_2$, $g^{y_1} \cdot (F)^{y_3} = W_2^e \cdot T_3$;%Correctness of Key Handover.

Fig. 7: DKG ZK argument

uct of $\{r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)}\}_{k=1}^{\ell}$ correctly encrypted the product of $\{g^{d_{i,j,m}^{(k)}}\}_{k=1}^{\ell}$, with homomorphic property. Since all $\{r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)}\}_{k=1}^{\ell}$ are encrypted by the same pk provided by Prover P as the first verification, we can concatenate all $\{r1_{i,j,m}^{(k)}, r2_{i,j,m}^{(k)}\}_{k=1}^{\ell}$ together by computing $R1, R2$. Then we can have that $E1 \cdot (R1)^e = g^t \cdot (g^X)^e$ which is equal to g^β as $\beta = t + X \cdot e$; and $E2 \cdot (R2)^e = s(F)^t \cdot \Pi_{m=1}^{M_2} \Pi_{k=1}^{\ell} (g^{d_{i,j,m}^{(k)} \lambda_m^k})^e \cdot \Pi_{m=1}^{M_2} (F^{X_m})^e = \alpha \cdot F^\beta$.

Lastly, we prove that $s_{i,j} = F_i(j) = G_{i,j}(0)$, as $\mathsf{MC}_{\{1,i\}}$ uses same $s_{i,j}$ to compute $G_{i,j}(0)$ as $F_i(j)$, which shows that $\mathsf{MC}_{\{1,i\}}$ has reshared its shares correctly. First, we observe that $\sum_{k=1}^{\ell} d_{i,j,m}^{(k)} p^k = G_{i,j}(m)$, and $G_{i,j}(m')$ for $m' = 1, \cdots, t_2$ can be interpolated to a unique polynomial with constant coefficient equal to $s_{i,j}$. Thus we can get $s_{i,j}$ by computing $s_{i,j} = \sum_{m'=1}^{t_2} G_{i,j}(m') u_{i,j,m'}$, where $\{u_{i,j,m'}\}_{m' \in [t_2]}$ are interpolation coefficients. Now we can compute that

$$W_1 = \Pi_{m'=1}^{t_2} (H1_{m'})^{u_{i,j,m'}} = g^{\sum_{m'=1}^{t_2} z'_m u_{i,j,m'}} = g^z$$

$$\begin{aligned} W_2 &= \Pi_{m'=1}^{t_2} (H2_{m'})^{u_{i,j,m'}} \\ &= g^{\sum_{m'=1}^{t_2} G_{i,j}(m') u_{i,j,m'}} \cdot h_m^{\sum_{m'=1}^{t_2} z'_m u_{i,j,m'}} \\ &= g^{s_{i,j}} F^z \end{aligned}$$

Therefore, it's easy to see that

$$g^{y_1} \cdot h^{y_2} = g^{a+s_{i,j} \cdot e} \cdot h^{b+s'_{i,j} \cdot e} = D^e \cdot g^a \cdot h^b = D^e \cdot T_1$$

$$g^{y_3} = g^{c+z \cdot e} = T_2 \cdot W_1^e$$

$$g^{y_1} \cdot F^{y_3} = g^{a+s_{i,j} \cdot e} \cdot F^{c+z \cdot e} = T_3 \cdot W_2^e$$

□

B. Maintaining ZK proof

In this section, we give a ZK proof for maintaining phase to show that the share $s_{i,f} = \sum_{k=1}^{\ell} d_{i,f}^{(k)} p^k = G_i(f)$ party P_f gets, where $d_{i,f}^{(k)}$, is indeed from decrypting $\{c1_{i,f}^{(k)} := g^{y_f^{(k)}}, c2_{i,f}^{(k)} := g^{d_{i,f}^{(k)}} \cdot h_m^{y_m^{(k)}}\}_{k \in [\ell]}$, then he uses the same share to re-share to next voting committee.

Lemma 2. *Assume the DDH problem is hard. The protocol described in Fig.8 is a honest verifier zero-knowledge argument of knowledge of c_0'', x_f,$\{y_t^{(1)}, \cdots y_t^{(\ell)}\}$, m_f such that P_f gets m_f after decrypting $\{c1_{i,f}^{(k)}, c2_{i,f}^{(k)}\}_{k=1,i=1}^{\ell, N_{\mathsf{QUAL}_d}}$, and he use the same m_f to reshare to the next voting committee, $\hat{F}_f(0) = m_f = \sum_{\mathsf{QUAL}_d} s_{i,f}$.*

Proof. **Perfect completeness**

The first step is to verify that prover correctly convert the ciphertexts $\{c1_{f,t}^{(k)}, c2_{f,t}^{(k)}\}_{k=1,t=1}^{\ell, M_{d+2}}$ encrypted by $\mathcal{MC}_{d+2}$'s pk to ciphertexts $\{r1_{f,t}^{(k)}, r2_{f,t}^{(k)}\}_{k=1,t=1}^{\ell, M_{d+2}}$ encrypted by prover's pk. The proof is the same as in Sec. VI-A, thus we omit the description here. Next, we prove that $\mathsf{MC}_{\{d+1,f\}}$ decrypts $\{c1_{i,f}^{(k)}, c2_{i,f}^{(k)}\}_{k=1,i=1}^{\ell, N_{\mathsf{QUAL}_d}}$ correctly. If we set $Q = \sum_{k=1,i=1}^{\ell, N_{\mathsf{QUAL}_d}} y_{i,f}^{(k)} \cdot p^k$, we can observe that $E_2 / E_1^{x_f} = g^{m_f + x_f Q} / (g^{x_f Q}) = g^{m_f}$ as $h_f = g^{x_f}$ and $\sum_{k=1,i=1}^{\ell, N_{\mathsf{QUAL}_d}} d_{i,f}^{(k)} p^{(k)} = m_f$. Furthermore, it's easy to see that $\Pi_{k=0}^{t_{d+2}} (\Delta_{f,k})^{j^k} = g^{\sum_{k=0}^{t_{d+2}} c'_k \cdot j^k} \cdot h^{\sum_{k=0}^{t_{d+2}} c''_k \cdot j^k} = g^{\hat{F}_f(j)} h^{\hat{G}_f(j)}$, where $j \in [0, M_{d+2}]$. When $j = 0$, we can get $\Pi_{k=0}^{t_{d+2}} (\Delta_{f,k}) = g^{\hat{F}_f(0)} h^{\hat{G}_f(0)} = g^{m_f} h^{c_0''}$, which is set as σ by verifier. Combining these two equations, we can get that $E_1^{-x_f} h^{c_0''} = \sigma / E_2$. Hence, we can verify that

$$\begin{aligned} E_1^{-z_1} h^{z_2} = E_1^{-x_f \cdot e - \rho_1} h^{c_0'' \cdot e + \rho_2} &= E_1^{-x_f e} h^{c_0'' e} \cdot E_1^{-\rho_1} h^{\rho_2} \\ &= (\sigma / E_2)^e \cdot E_1^{-\rho_1} h^{\rho_2} = (\sigma / E_2)^e \cdot a_2 \end{aligned}$$

Given $a_1 = g^{\rho_1}$ and $h_f = g^{m_f}$, it is easy to prove that

$$g^{z_1} = g^{x_f \cdot e + \rho_1} = {h_f}^e \cdot a_1$$

Lastly, we prove the correctness of key handover. We have $\sum_{k=1}^{\ell} m_{f,t}^{(k)} p^k = \hat{F}_f(t)$, and $\hat{F}_f(t)$ for $t = 1, \cdots, t_{d+2}$ can be interpolated to a unique polynomial with constant coefficient equal to m_f. Thus we can get m_f by computing $m_f = \sum_{k=1}^{\ell} \hat{F}_f(t) u_{f,t}$. Now we can compute that

$$W_1 := \Pi_{t=1}^{t_{d+2}} (H_{\{t,1\}})^{u_{f,t}} = g^{\sum_{t=1}^{t_{d+2}} z_t u_{f,t}} = g^z$$

$$W_2 := \Pi_{t=1}^{t_{d+2}} (H_{\{t,2\}})^{u_{f,t}} = g^{\sum_{t=1}^{t_{d+2}} \hat{F}_f(t) u_{f,t}} \cdot h_t^{\sum_{t=1}^{t_{d+2}} z_t u_{f,t}} = g^{m_f} h_t^z$$

Therefore, it's easy to see that

$$g^{y_1} \cdot h^{y_2} = g^{a + m_f \cdot e} \cdot h^{b + c_0'' \cdot e} = \sigma^e \cdot g^a \cdot h^b = \sigma^e \cdot T_1$$

$$g^{y_3} = g^{c + z \cdot e} = T_2 \cdot W_1^e$$

$$g^{y_1} \cdot h_t^{y_3} = g^{a + m_f \cdot e} \cdot h_t^{c + z \cdot e} = T_3 \cdot W_2^e$$

□

VII. Reputation Management Scheme

A. 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 $\mathsf{Rep}_{\mathsf{Fld}}^{(k)}(u_i)$ of user u_i in field Fld in the k-th treasury period is aggregated by four reputation factors, $\mathsf{Rep}_{\mathsf{Fld}}^{(k)}(u_i) := (\mathsf{Rep\text{-}RW}^{(k)}(u_i), \mathsf{Rep\text{-}PC}_{\mathsf{Fld}}{}^{(k)}(u_i), \mathsf{Rep\text{-}PP}_{\mathsf{Fld}}{}^{(k)}(u_i), \mathsf{Rep\text{-}TS}^{(k)}(u_i))$. $\mathsf{Rep\text{-}RW}^{(k)}(u_i)$ 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. $\mathsf{Rep\text{-}PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$ 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. $\mathsf{Rep\text{-}PP}_{\mathsf{Fld}}{}^{(k)}(u_i)$ is partial

Maintaining ZK argument

Common Knowledge: Commitment $\{\Delta_{f,k}\}_{k=0}^{t_{d+2}}$ of $\{c'_k\}$, qualified parties set QUAL_d with size N_{QUAL_d}, ℓ, p, public key h_f, $u_{f,1}, \cdots, u_{f,t_{d+2}}$ are distinct points in $\mathcal{Z}_q$ defining Lagrange polynomial $\hat{F}_f(z)$ such that $\hat{F}_f(0) = m_f$.
Common Reference String: Commitment key h.

Statement: Ciphertexts $\{c1_{i,f}^{(k)} = g^{y_{i,f}^k}, c2_{i,f}^{(k)} = g^{d_{i,f}^k} \cdot (h_f)^{y_{i,f}^{(k)}}\}_{k=1,i=1}^{\ell, N_{\mathsf{QUAL}_d}}$, $\hat{F}_f(0) = m_f = \sum_{i=1}^{N_{\mathsf{QUAL}_d}} s_{i,f}$, public keys h_f and $\{h_t\}_{t=1}^{M_{d+2}}$, $\{c1_{f,t}^{(k)} := g^{y_t^{(k)}}, c2_{f,t}^{(k)} := g^{d_{f,t}^{(k)}} \cdot h_t^{y_t^{(k)}}\}_{k=1,t=1}^{\ell, M_{d+2}}$

Witness: The free term c_0'' of $\hat{G}_f(z)$, secret key x_f, $\{y_t^{(1)}, \cdots y_t^{(\ell)}\}_{k=1,t=1}^{\ell, M_{d+2}}$, from $\mathcal{Z}_q$, m_f, $\{g^{d_{i,f}^{(k)}}\}_{k=1,i=1}^{\ell, N_{\mathsf{QUAL}_d}}$, $\{g^{d_{f,t}^{(k)}}\}_{t=1}^{M_{d+2}}$.
Protocol:

- The prover P does the followings:
  - Select $\tau \leftarrow \mathcal{Z}_q$;
  - Compute $F := g^\tau$; %Prover sets his pk as F, and sk as τ.
  - For $t = 1, \cdots, M_{d+2}$
    - For $k = 1, \cdots, \ell$
      - Select $\delta_{f,t}^{(k)} \leftarrow \mathcal{Z}_q$;
      - Compute $r1_{f,t}^{(k)} := g^{\delta_{f,t}^{(k)}}$, $r2_{f,t}^{(k)} := g^{d_{f,t}^{(k)}} \cdot (F)^{\delta_{f,t}^{(k)}}$. %Convert $\{c1_{f,t}^{(k)}, c2_{f,t}^{(k)}\}_{k=1,t=1}^{\ell, M_{d+2}}$ to $\{r1_{f,t}^{(k)}, r2_{f,t}^{(k)}\}_{k=1,t=1}^{\ell, M_{d+2}}$.
  - Set $\{\lambda_t\}_{t=1}^{M_{d+2}} \leftarrow \mathsf{hash}(\langle c1_{f,t}^{(k)}, c2_{f,t}^{(k)}, r1_{f,t}^{(k)}, r2_{f,t}^{(k)} \rangle_{k=1,t=1}^{\ell, M_{d+2}})$;
  - For $t = 1, \cdots, M_{d+2}$
    - Select $\omega_t, \upsilon_t \leftarrow \mathcal{Z}_q$;
    - Compute $Y_t := y_t^{(1)}\lambda_t + y_t^{(2)}\lambda_t^2 + \cdots + y_t^{(\ell)}\lambda_t^\ell$, $X_t := \delta_t^{(1)}\lambda_t + \delta_t^{(2)}\lambda_t^2 + \cdots + \delta_t^{(\ell)}\lambda_t^\ell$;
    - Compute $A2_t := h_t^{\omega_t} / F^{\upsilon_t}$;
  - Compute $A1 := \Pi_{t=1}^{M_{d+2}} g^{\omega_t - \upsilon_t}$;
  - Select $\rho_1, \rho_2 \leftarrow \mathcal{Z}_q$, compute $a_1 := g^{\rho_1}, a_2 := E_1^{-\rho_1} h^{\rho_2}$;
  - Select $a, b, c \leftarrow \mathcal{Z}_q$, compute $T_1 := g^a \cdot h^b$, $T_2 := g^c$, $T_3 := g^a \cdot F^c$;
  - Compute $E_1 = \Pi_{i=1}^{N_{\mathsf{QUAL}_d}} \Pi_{k=1}^{\ell} (c1_{i,f}^{(k)})^{p^k}$, $E_2 = \Pi_{i=1}^{N_{\mathsf{QUAL}_d}} \Pi_{k=1}^{\ell} (c2_{i,f}^{(k)})^{p^k}$
  - Set $e \leftarrow \mathsf{hash}(\langle c1_{f,t}^{(k)}, c2_{f,t}^{(k)}, r1_{f,t}^{(k)}, r2_{f,t}^{(k)} \rangle_{k=1,t=1}^{\ell, M_{d+2}}, \langle \lambda_t, A2_t \rangle_{i=1}^{M_{d+2}}, A1, a_1, a_2, T_1, T_2, T_3, E_1, E_2)$;
  - For $t = 1, \cdots, M_{d+2}$
    - Compute $v1_t := \omega_t + Y_t \cdot e$, $v2_t := \upsilon_t + X_t \cdot e$;
- Compute $v1 := \sum_{t=1}^{M_{d+2}} v1_t$, $v2 := \sum_{t=1}^{M_{d+2}} v2_t$;
- Compute $z_1 := x_f \cdot e + \rho_1$, $z_2 := c_0'' \cdot e + \rho_2$;
- For $t = 1, \cdots, M_{d+2}$
  - Compute $z_t := \sum_{k=1}^{\ell} \delta_t^{(k)} p^k$
- Compute $z := \sum_{t'=1}^{t_{d+2}} z_{t'} u_{f,t'}$
- $y_1 := a + m_f \cdot e$, $y_2 := b + c_0'' \cdot e$, $y_3 := c + z \cdot e$;
- $P \rightarrow V : (\langle c1_{f,t}^{(k)}, c2_{f,t}^{(k)}, r1_{f,t}^{(k)}, r2_{f,t}^{(k)} \rangle_{k=1,t=1}^{\ell, M_{d+2}}, \langle A1_t, A2_t, v1_t, v2_t \rangle_{t=1}^{M_{d+2}}, a_1, a_2, T_1, T_2, T_3, E_1, E_2, z_1, z_2, y_1, y_2, y_3)$.

Verification:

- Compute the followings:
  - $\lambda \leftarrow \mathsf{hash}(\langle c1_{f,t}^{(k)}, c2_{f,t}^{(k)}, r1_{f,t}^{(k)}, r2_{f,t}^{(k)} \rangle_{k=1,t=1}^{\ell, M_{d+2}})$;
  - $e \leftarrow \mathsf{hash}(\langle c1_{f,t}^{(k)}, c2_{f,t}^{(k)}, r1_{f,t}^{(k)}, r2_{f,t}^{(k)} \rangle_{k=1,t=1}^{\ell, M_{d+2}}, \langle A1_t, A2_t \rangle_{i=1}^{M_{d+2}}, \lambda, a_1, a_2, T_1, T_2, T_3, E_1, E_2)$;
  - $v1 := \sum_{t=1}^{M_{d+2}} v1_t$, $v2 := \sum_{t=1}^{M_{d+2}} v2_t$;
  - For $t = 1, \cdots, M_{d+2}$
    - $C2_t = (c2_{f,t}^{(1)})^\lambda \cdot (c2_{f,t}^{(2)})^{\lambda^2} \cdots (c2_{f,t}^{(\ell)})^{\lambda^\ell}$, $R2_t = (r2_{f,t}^{(1)})^\lambda \cdot (r2_{f,t}^{(2)})^{\lambda^2} \cdots (r2_{f,t}^{(\ell)})^{\lambda^\ell}$;
    - $H_{\{t,1\}} := \Pi_{k=1}^{\ell} (r1_{f,t}^{(k)})^{p^k}$, $H_{\{t,2\}} := \Pi_{k=1}^{\ell} (r2_{f,t}^{(k)})^{p^k}$;
  - $C1 = \Pi_{t=1}^{M_{d+2}} \Pi_{k=1}^{\ell} (c1_{f,t}^{(k)})^{\lambda_t^k}$,$R1 = \Pi_{t=1}^{M_{d+2}} \Pi_{k=1}^{\ell} (r1_{f,t}^{(k)})^{\lambda_t^k}$
  - $W_1 := \Pi_{t'=1}^{t_{d+2}} (H_{\{t',1\}})^{u_{f,t'}}$, $W_2 := \Pi_{t'=1}^{t_{d+2}} (H_{\{t',2\}})^{u_{f,t'}}$;
  - $\sigma := \Pi_{k=0}^{t_{n+2}} (\Delta_{f,k})$;
  - $E_1 = \Pi_{k=1,i=1}^{\ell, \mathsf{QUAL}_d} (c1_{i,f}^{(k)})^{p^k}$, $E_2 = \Pi_{k=1,i=1}^{\ell, \mathsf{QUAL}_d} (c2_{i,f}^{(k)})^{p^k}$.
- Check the followings:
  - $A1 \cdot (C1/R1)^e = g^{v1-v2}$; % Correctness of ciphertext conversion.
  - For $t = 1, \cdots, M_{d+2}$
    - $A2_t \cdot (C2_t/R2_t)^e = h_t^{v1_t} / F^{v2_t}$. % Correctness of ciphertext conversion.
  - $E_1^{-z_1} h^{z_2} = (\sigma/E_2)^e \cdot a_2$, $g^{z_1} = {h_f}^e \cdot a_1$. % Correctness of decryption.
  - $g^{y_1} \cdot h^{y_2} = \sigma^e \cdot T_1$, $g^{y_3} = W_1^e \cdot T_2$, $g^{y_1} \cdot (F)^{y_3} = W_2^e \cdot T_3$; % Correctness of key handover.

Fig. 8: Maintaining ZK argument

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, $\mathsf{Rep\text{-}TS}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}TS}^{(k)}(u_i)$ value is zero as they didn't contribute in short-list genration.

To assist understanding of the following reputation algorithm RepCal in Fig. 9, we summarised the frequently used notations in Table I. 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 $\mathsf{Role} \in \{\mathcal{O}^{(k)} \cup \mathcal{VT}^{(k)} \cup \mathcal{E}^{(k)} \cup \mathcal{T}_k\}$ 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 main-chain, 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 9 as $\mathsf{Rep\text{-}RW}^{(k)}(u_i)$. 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 $\mathsf{Rep\text{-}RW}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}RW}^{(k)}(u_i)$. 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 $\mathsf{Rep\text{-}PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$) 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 $\mathsf{Rep\text{-}PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}RW}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}PC}_{\mathsf{Fld}}{}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}PP}_{\mathsf{Fld}}{}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}PP_{Fld}}^{(k)}(u_i)$ will simply be user's winning rate $\mathsf{PWR_{Fld}}^{[k_0,\ldots,k]}$.

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 $\mathsf{Rep\text{-}TS}^{(k)}(u_i)$ 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 $\mathsf{Rep\text{-}TS} := \omega \cdot log_{10}(\mathsf{ConsR}(u_i)^{(k)}) + 1$, 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 $w_1 \cdot \mathsf{Rep\text{-}RW}^{(k)}(u_i) + w_2 \cdot \mathsf{Rep\text{-}PC_{Fld}}^{(k)}(u_i) + w_3 \cdot \mathsf{Rep\text{-}PP_{Fld}}^{(k)}(u_i) + w_4 \cdot \mathsf{Rep\text{-}TS}^{(k)}(u_i)$. However, when weighting Rep-RW and $\mathsf{Rep\text{-}PC_{Fld}}$, 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 $\mathsf{Rep\text{-}Pub_{Fld}} \in [0, 1]$. 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 I: 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
$\mathsf{Rep_{Fld}}(u_i)^{(k)}$	reputation score of user u_i in field Fld in k-th treasury period
$\mathsf{Role} \in \{\mathcal{O}^{(k)} \cup \mathcal{VT}^{(k)} \cup \mathcal{E}^{(k)} \cup \mathcal{T}_k\}$	The role of user in current treasury period
$\{\mathsf{VPR}^{(t)}(u_i)\}_{t\in[k_0,\ldots,k]}$	the voting power ratio casted by u_i in t-th period
$\{\mathsf{ACC_{Fld}}^{(t)}(u_i)\}_{t\in[T]}$	the percentage of proposals supported by u_i in filed Fld entered the list of winning projects when he joined the t-th time
$Y_{\mathsf{Fld}}^{[k_0,\ldots,k]}(u_i)$	the number of projects u_i voted YES by himself in filed Fld from k_0-th period to k-th period
$N_{\mathsf{Fld}}^{[k_0,\ldots,k]}(u_i)$	the number of projects u_i voted NO by himself in filed Fld from k_0-th period to k-th period
$\mathsf{Sign}^{(k)}(u_i) \in \{0, 1\}$	if u_i is a treasury committee member in period k and he signed the short-list, then $\mathsf{Sign}^{(k)}(u_i) = 1$; otherwise, $\mathsf{Sign}^{(k)}(u_i) = 0$.
$\mathsf{ConsR}(u_i)^{(k)}$	consistent rate of short-listed proposals and decision made by voters
ω	treasury committee smoothing parameter
$\mathsf{PWR_{Fld}}^{[k_0,\ldots,k]}$	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
$\mathsf{CR}(u_i)$	User's credibility, if u_i is honest, $\mathsf{CR}(u_i) = 1$; otherwise $\mathsf{CR}(u_i) = 0$
$\mathsf{Rep\text{-}Pub_{Fld}}$	the optional external source of reputation for u_i

B. 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. 10.

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

Algorithm RepCal

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

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

Regularity of Work:

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

Quality of total productive contributions:

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

Funded Percentage:

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

Treasury Short-List Contribution:

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

Reputation Factors Aggregation:

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

Output:

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

Fig. 9: Reputation Computation Algorithm RepCal

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. The root hash of these two Merkle trees are included in the block header, as Transaction Merkel Root Hash and Reputation Merkel Root Hash.

Each record in the reputation list contains a user's public key, the consensus reputation value of the voter, the vote message. Since the reputation of a node will be updated once the node's status (the assets, transaction behaviour, consensus participation) changes, it is time-consuming to record the updated reputation value considering the node behaviour in this block. In our design, nodes' reputation will be updated once the new block is confirmed in the network, whilst, the block generation process will continue to proceed to the next block height based on the latest consensus reputation values. Once the reputation updating is completed, the updated reputation is the used for the block next to the current processed block. [JJ: not all blocks have reputation values right?]

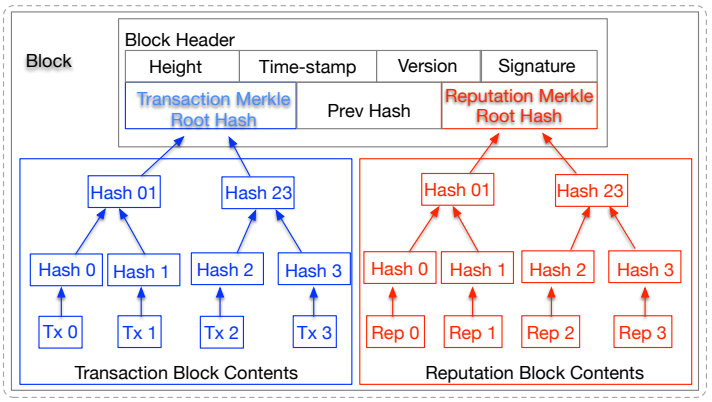

Fig. 10: Block Structure

VIII. Implementation and Performance

IX. Related Work

A. Voting in blockchain

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

B. 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 centralized and decentralized 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 decentralized 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 decentralized 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.

X. Conclusion