UniqueChain: A Fast, Provably Secure Proof-of-Stake Based Blockchain Protocol in the Open Setting

Peifang Ni[1,2,3], Hongda Li[1,2,3], Xianning Meng[1,2,3], and Dongxue Pan[1,2,3]

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[2] Data Assurance and Communication Security Research Center, Beijing, China
[3] State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
nipeifang@iie.ac.cn

Abstract. We present "*UniqueChain*", a proof-of-stake based blockchain protocol that achieves secure initialization of newly joining parties without any additional trusted assumptions and fast messages (transactions) confirmation. Specifically, the adversary can send *corrupt* instructions to any parties at any time mildly and have messages delivery delay with an upper bound. Security of our protocol holds if majority of overall stakes are controlled by honest parties.
In "*UniqueChain*", we propose a new form of two-chain structure that consists of two tightly linked chains named *leader* chain and *transaction* chain with two types of corresponding blocks named *leader* block and *transaction* block. To achieve the above guarantees, we formalize a secure bootstrapping mechanism for new parties in open setting and realize *uniqueness* of *transaction* chains held by honest parties. We prove that "*UniqueChain*" satisfies security properties as chain growth, chain quality, common prefix and soundness, and two additional properties as uniqueness and high efficiency.

Keywords: proof-of-stake, secure initialization, uniqueness, high efficiency

1 Introduction

Blockchain, the technique of the most concern, has been investigated in various fields in recent years and believed to make huge changes to the future world, such as finance. Bitcoin, introduced in [23], is the first successful implementation of blockchain. Parties compete to extend chains by solving a computational puzzle (proof-of-work), which is *a moderately hard hash inequality* [11, 26], and the opportunity of a party to be winner is related to the amount of computational power that he has invested.

In bitcoin system, any forks must consume extra computational power and it is rational that a party extends one chain all the time. As a result, the best chain

selection rule-*longest chain* enables the new parties to be initialized securely and freely. So that bitcoin system can scale to a large network in the open setting. The core of bitcoin system has been extracted called bitcoin backbone protocol and analyzed under specific assumptions in [16, 24, 17]. Their works show that, assuming the majority of computational power are controlled by honest parties, bitcoin system satisfies three security properties as chain growth, chain quality and common prefix.

However, due to the essence of proof-of-work (PoW) mechanism, bitcoin system has wasted a huge amount of computational power, which is the non-recyclable physical resource. Proof-of-stake (PoS) is the most desirable mechanism to replace PoW, which enables a party to provide a proof that he is eligible to extend the chain. Precisely, the process of winner selection is related to some properties of the parties themselves, such as the balance of their accounts. It is attractive to design a PoS based blockchain protocol that is as secure as PoW based ones. Unfortunately, we have got the negative results of constructing a blockchain protocol in open setting via PoS mechanism [3]. *Nothing-at-stake* attack enables the adversary to generate an alternative blockchain that could be better than honest parties' local chains even if the minority of stakes are controlled by him. So that the newly parties have no means of distinguishing honest chains from the malicious one. A secure bootstrapping is necessary for new parties to identify the correct version of local chains. Further, efficiency has always been a fundamental problem of constructing blockchain protocols in the scalable and open network. In short, it is meaningful to achieve PoS based consensus among honest parties in open setting and there are three challenges to be handled as posterior corruptions, secure initialization of new parties and low efficiency. Here, we have the following interesting question:

> Is it possible to construct a fast and provably secure proof-of-stake based blockchain protocol in the open setting, which provides secure initialization for new parties without any trusted assumptions?

1.1 Our Contributions

In this paper, we obtain a positive result of the above question. Informally, our contributions can be briefly summarized in the following outline. First, we introduce three main challenges about designing PoS based blockchain protocols in the open setting.

- **Posterior Corruptions.** Unlike PoW based blockchain protocols, where there are no free valid blocks. The essence of PoS mechanism determines that the adversary can rewrite history freely by corrupting the elected honest parties at some point in the future. It means that an (fully) adaptive adversary can generate an alternative blockchain without losing anything to mislead the existing honest parties to make wrong selections.
- **Secure Initialization of New Parties.** As discussed above, *nothing-at-stake* attack enables the adversary to hold a chain that is indistinguishable

from the existing honest parties' and mislead the newly joining parties to choose the wrong version of local chains successfully.

– **Low Efficiency.** Efficiency has always been a problem of blockchain protocols to be solved. Note that, most of existing schemes can only guarantee a common prefix of chains held by honest parties and the main reasons can be listed as follows. (1). Due to the facts that an elected adversary may behave maliciously (i.e., hide new blocks or broadcast more than one blocks at the same time in PoS based blockchain protocols) and more than one parties may be elected synchronously, so it is unavoidable that honest parties hold different views of the latest several blocks, (2). Messages depend tightly on a newly created block. Consequently, it is impossible to confirm messages immediately and eventually when the block that consists of them is received. We stress that the major cause is that honest parties hold different views of the party who is eligible to create a block that consists of messages.

Handling Posterior Corruptions. In our protocol, we consider a mildly adaptive adversary that the *corrupt* instruction takes δ slots to be effective. Inspired by the solution in [3], existing honest parties can compare local chains with the ones received from network. Based on protocol execution, at any time, honest parties will always reject a chain pair if the *leader* chain forks more than $K \in \mathbb{N}$ (the parameter of common prefix property) blocks or the *transaction* chain is longer with more than one blocks from their local ones. In this way, adversary cannot convince the existing honest parties to revise their local chains.

Note that parameter δ should be set reasonably in that (1). a small δ means stronger security, but it will lead the existing honest parties to the adversarial chain (2). a big δ means that protocol is secure with a relative static adversary. We set $\delta \geqslant R+\epsilon$ to ensure the *leader* blocks created by honest parties in current epoch are confirmed before the *corrupt* instruction takes effect, where R is the length of an epoch and ϵ is a secure margin for the last honest leader of an epoch.

Achieving Secure Initialization of Newly Joining Parties. As discussed above, it is impossible for the newly joining parties to be consistent with existing honest parties without any additional help. [3] achieves secure bootstrapping by providing new parties with a list of parties with honest majority. New parties can be bootstrapped from the genesis block [1] that the initial parties provide them with local states.

We can see that these given solutions aim to introduce a lower bound for initializing new parties and the additional trusted assumption is unavoidable. Therefore, we try to discard the trust assumption. Inspired by these existing solutions, instead of providing the new parties with a trusted list of parties, we propose a mechanism that allows the most recent elected parties to provide the new parties with their local states in a self-organized manner.

Precisely, we introduce a new type of transactions called *requesting* transactions Tx_r that allows new parties to require the current state for secure joining. In details, Tx_r with fixed value $v = 0$ has at least one valid inputs and $\tilde{l}$ outputs, where $\tilde{l}$ is the expected number of parties being elected during an epoch. Note that we aim to let new parties obtain the common prefix (confirmed blocks)

of existing honest parties' local *leader* chains. A party P_j can provide the new parties with his local *leader* chain $\mathcal{C}_{loc}$ by broadcasting a genaral (spending) transaction at slot $sl_j^{e_i}$ as $Tx_g = (h_{-1}, v, sl_j^{e_i}, \pi, \omega = (sl_j^{e_i}, c, \sigma))$ successfully if the followings are satisfied. (1). h_{-1} is index of Tx_r created at some slot of epoch e_i. (2). $v = 0$ is value of Tx_g. (3). $\mathcal{H}(pk_j, sl_j^{e_i}, nonce^{e_i}) < T^{e_i}$ denotes that P_j is elected as leader at slot $sl_j^{e_i}$ of epoch e_i with random number $nonce^{e_i}$ and difficult target T^{e_i}. (4). $\mathcal{H}^*(pk_j, nonce^{e_i})\ mod\ \tilde{l}+1 = i$ indicates that P_j can redeem the *ith* output of Tx_r. (5). $c = Enc_{pk}(\mathcal{C'}_{loc})$ is encryption of $\mathcal{C}_{loc}$'s latest $l > K$ blocks under public key of requiring party. (7). $Ver_{pk_j}(Tx_g, \sigma) = 1$ denotes that σ is correct signature of Tx_g under P_j's signing key. Note that $\mathcal{H}, \mathcal{H}^*$ are two hash functions with specific outputs. As soon as all the elected honest parties of epoch e_i have broadcasted their transactions, then new parties can determine the common prefix of exisiting honest parties' local chains and further get a correct local chain. More details are showed in section 4.

At the beginning of an epoch, the leader selection function $\mathcal{H}$ is determined so are the elected parties of the current epoch. So that new parties can join the network at any time. Note that party P_j can provide his whole local chain when $len(\mathcal{C}_{loc}) \leq K$ ($len(\mathcal{C}_{loc})$ denotes length of $\mathcal{C}_{loc}$) and new parties choose one chain randomly in that no blocks have been confirmed by honest parties now.

Achieving High Efficiency of Handling Messages. In PoS based blockchain protocol, there are two main ways to handle messages (1). an elected party is eligible to create a block that consists of messages [20, 9, 1], (2). an elected party first creates an empty block, then the empty block is viewed as a random beacon to select a party to generate a block with payloads [15, 10]. We can see that the processes of leader electing and handling messages are synchronized, so that the messages will be confirmed if and only if the backed block is confirmed by honest parties. As discussed above, honest parties cannot hold a same local chain. Consequently, the newly created blocks cannot be confirmed immediately by honest parties, so are the messages packed in the blocks.

Fast messages confirmation means that once a transaction block is accepted by an honest party, then it is confirmed by all honest parties finally without waiting for being backed by several blocks. In our protocol, we separate these two processes discussed above. More concretely, we introduce two types of blocks as *leader block* and *transaction block* that correspond to two types of blockchains as *leader chain* and *transaction chain*, and propose a new form of two-chain structure. Indeed, an elected party is allowed to create an empty block-*leader block* and then he is eligible to create a *transaction block* that consists of messages if his *leader* block has been confirmed by all the honest parties, which means that the *leader block* is in common part of honest parties' local chains.

The uniqueness of the common part held by honest parties determines the uniqueness of party who is eligible to extend the current *transaction* chain in network. As a result, at any slot, there is at most one valid newly created *transaction* block in network and honest parties hold a same view of the current *transaction* chain, and that is why we call our chain as *unique chain*. During protocol execution, honest parties hold different local *leader* chains that enjoy

a large common prefix with overwhelming probability and a same *transaction* chain. We stress that *transaction* chain will not be extended until the length of *leader* chain is at least $K + 2 \in \mathbb{N}$ and these two chains will not grow together in that the elected adversary may not create or broadcast valid blocks. In short, in our protocol, the honest parties hold forked local *leader* chains that enjoy a common prefix and an unique local *transaction* chain that is at least $K + 1$ blocks shorter than *leader* chain.

UniqueChain Π^{UC}. Based on the above description, our blockchain consists of two types of chains (Fig.2) and four phases (section 2.3). Precisely, we present our protocol in the $\{\mathcal{F}_{init}, \mathcal{F}_{res}, \mathcal{F}_{NET}\}$-hybrid model and prove that with overwhelming probability, the chains held by honest parties satisfy four fundamental security properties as chain growth, chain quality, common prefix and soundness, and two additional properties as uniqueness and high efficiency.

1.2 Related Work

Chaum introduces the first e-cash system with a central bank[8]. Bitcoin is the first fully decentralized currency system [23] introduced by Nakamoto. The success of bitcoin brings us the first scalable consensus protocol in the open setting, where parties can join or leave freely.

Recently, a number of works focus on the investigation of security of bitcoin system. Garay et al. formally analyze the core of Nakamoto's blockchain protocol in synchronous network [16] and propose two security properties as chain quality and common prefix. [21] is the first to define chain growth property formally. Pass et al. extend their works to asynchronous networks [24]. In [17], Garay et al. further consider the difficulty target recalculation function in the adaptive setting. Chain quality, chain growth and common prefix have been considered as the fundamental properties of blockchain protocols. [12, 14, 27, 28] analyze bitcoin system in the rational setting.

Despite the success of PoW based blockchain protocols, they have some inevitable flaws, i.e., consuming a huge amount of non-recyclable physical resources. It is meaningful to construct blockchain protocols that rely on environment-friendly resource. PoS mechanism [2] enables a party to prove ownership of some stakes and a number of works have been studied PoS based blockchain protocols.

Sleepy [25] studies the distributed protocols in a *sleepy* model of computation where players can be on-line (alert) or off-line (asleep), considers a fixed stakeholder distribution and sporadic participation happens at any given point. *Ouroboros* [22], the first PoS based blockchain protocol with rigorous security guarantees, does not consider sporadic participation that parties are fixed in genesis block. The elegant work *Algorand* [18] is an adaptive secure PoS based blockchain protocol. But it requires the elected committee members being online and makes progress if majority of committee members do show up. What's more, the current committee runs a Byzantine agreement protocol, which can only be secure against $\frac{1}{3}$ adversary. Snow White [3] is the first to formally articulate the robust requirements for PoS based blockchain protocols. A negative result is concluded that it is impossible for a newly joining party correctly identifying

the true version of history without additional trusted advices. Further, a mildly adaptive adversary is considered and the *check pointing* idea helps existing parties to choose the best local chains correctly. [9] presents *Ouroboros Praos*, the first PoS based blockchain protocol that guarantees security against a fully adaptive adversary in semi-synchronous setting with cryptographic techniques as verifiable random function and forward secure digital signature scheme. [1] improves *Ouroboros Praos* to achieve dynamic availability as bitcoin system. In this protocol, the new or off-line parties can safely (re-)join and bootstrap their local chain from the genesis block. In fact, the genesis block consists of the initial parties who provide their local states for new parties. However, it ignores a condition that the adaptive adversary may corrupt most of the initial parties and provide the wrong version of local states for the requesting parties.

The two-chain structure blockchain has been studied. [10] proposes a scheme with two types of chains called PoW-chain and PoS-chain via combining PoW and PoS mechanisms efficiently to against a malicious majority of computing power in open setting. [15] shows a PoS based blockchain with two chains to mimic PoW based blockchains. [13] achieves high throughput via proposing two types of blocks called *key-block* and *micro-block*, in which the current leader do not stop handling transactions (*micro-block*) until the next leader is elected (*key-block*). In these proposed schemes, the blocks with messages (transactions) link to the newly-created empty blocks, which have not been confirmed by honest parties. So that they only guarantee the common-prefix property of honest parties' local chains and messages (transactions) are confirmed only when the corresponding block is deep enough. Consequently, fast messages confirmation is still not achieved.

1.3 Outline of the Paper

The remainder of the paper is organized as follows. In section 2, we present the preliminaries of our protocol. The ideal functionalities used in our protocol are showed in section 3. Then we give the detailed construction of our protocol in section 4. Security analysis is in section 5. Conclusion is presented in section 6.

2 Preliminaries

In this section, we follow Canetti's formulation of the multiparty protocol execution [6, 7] and Pass's cryptographic model for blockchain protocol [24] to give the formal model of protocol execution and some related definitions.

2.1 The Model of Protocol Execution

Epoch-based execution. Protocol executes in disjoint and consecutive time intervals called *epoch*. More concretely, time is divided into fixed size unites called slot sl and each epoch e consists of $R \in \mathbb{N}$ slots that $e_i = \{sl_j^{e_i}, j \in \{1, ..., R\}\}$

denotes the *ith* epoch. We assume that each party holds almost synchronous local clock, so is the current slot.

The parties. In open setting, parties can join or leave the network safely without any permissions. We classify parties who maintain the system into two sets in the network as the *existing* parties E that have caught up with protocol execution and the *joining* parties J that are in the initializing phase.When a party $P \in J$ is initialized successfully, then we have $E = E \cup \{P\}$ and $J = J/\{P\}$. Further, E consists of honest parties H and corrupted parties C.

The adversary. In decentralized setting, adversary $\mathcal{A}$ is fully adaptive that controls the entire local states of corrupted parties as soon as it sends the *corrupt* instruction. In our epoch-based protocol, we set $\mathcal{A}$ as mildly adaptive that the *corrupt* instruction takes effect after $\delta \geqslant R + \epsilon$ slots since it is sent. Parameter δ guarantees the adversary can control the whole leaders of the current epoch with negligible probability, even if he can foresee the leaders at the starting of the epoch (the time that leader election function is determined). We set a secure margin ϵ that ensures the block created by the last honest leader of an epoch to be confirmed by all honest parties before the *corrupt* instruction takes effect. Note that $\mathcal{A}$ can send *corrupt* instruction at any time of protocol execution, which will take effect in the next epoch.

In order to describe easily, we borrow the flat model from [16] that each party holds one unite of stake and security holds if majority of existing parties are honest. More formally, there exists a constant $\varphi > 0$ such that during protocol execution, we have $\frac{|H|}{|C|} \geq 1 + \varphi$. Moreover, we assume that $|E| \cdot p << 1$, where $p = \frac{T^{e_i}}{2^k}$ (security parameter k) is probability that a party with one unit stake is elected at a given slot of epoch e_i and T^{e_i} is different target determined by the current distribution of stakes in network to ensure a stable growth of chain.

FUNCTIONALITY $\mathcal{F}_{NET}$

$\mathcal{F}_{NET}$ is parameterized by Δ, interacts with an ideal adversary $\mathcal{S}$ and a set of parties $\mathcal{P} \subseteq E$. At slot $sl_j^{e_i}$ $(i \in \mathbb{N}, j \in \{1, ..., R\})$, it proceeds as follows.

- Upon receiving input $(Broadcast, m)$ from a party $P_{i'} \in \mathcal{P}$, sends $(Broadcast, m)$ to $\mathcal{S}$ and records $(P_{i'}, m, b = 0, sl_j^{e_i})$.
- Upon receiving $(Broadcast, m, P'_{i'}, t)$ from $\mathcal{S}$, where $P'_{i'} \in \mathcal{P}$, then
  - if there is a record $(P_{i'}, m, b = 0, sl_j^{e_i})$ and $t \leqslant sl_j^{e_i} + \Delta$, then sends $(m, P'_{i'})$ to all the other parties at time t and sets $b = 1$.
  - else, if $t > sl_j^{e_i} + \Delta$, then sends $(m, P'_{i'})$ to all the other parties at time $sl_j^{e_i} + \Delta$ and sets $b = 1$.
  - else, ignores the message.

Fig. 1. The Communication Network Functionality $\mathcal{F}_{NET}$

Communication Network. In blockchain protocol, the parties communicate with each other via a $diffusion\ mechanism$ that guarantees the messages sent by a party can be eventually received by the other parties. Here, we assume a *slot-synchronous* network and parties (in E or J) have access to a functionality $\mathcal{F}_{NET}$ that is parameterized with an upper bound $\Delta \leqslant 1$ *slot* to reflect the network latency and guarantee that the messages sent by honest parties are delivered within a slot. Formally, $\mathcal{F}_{NET}$ proceeds as follows. Upon receiving an instruction to diffuse messages from a party at slot $sl_j^{e_i}$, then $\mathcal{F}_{NET}$ asks the adversary for the delivery time. If the specified time $t \leq sl_j^{e_i} + \Delta$, then set the delivery time as $t' = t$, else $t' = sl_j^{e_i} + \Delta$. Note that the source of messages can be modified by adversary and no messages delivery is delayed by more than one slot. $\mathcal{F}_{NET}$ is described in Fig.1.

2.2 Notations

Cryptographic Techniques.

1. Collision-Resistant Hash Functions. In our protocol, function $\mathcal{H}$ determines the leaders of each slot and $\mathcal{H}^*$ determines the serial number of the requesting transaction's output that can be redeemed by a party. Note that for each epoch, there is an unique random seed *nonce* for $\mathcal{H}$ and $\mathcal{H}^*$.
   - $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$, where k is security parameter.
   - $\mathcal{H}^* : \{0,1\}^* \to \{0,1\}^{k'}$, where $\tilde{l} = 2^{k'}$ is the expected number of parties being elected during a given epoch.
2. Semantically-Secure Public Key Encryption Scheme. (Gen, Enc, Dec) is denoted as a semantically secure public key encryption scheme.
3. Digital Signature Scheme. (Gen, Sig, Ver) is denoted as an unforgeable digital signature scheme.
4. Commitment Scheme. $(Com, Open)$ is denoted as a commitment scheme with security properties as *correctness*, *binding* and *hiding*.

Two Types of Blocks. In our protocol, we propose two types of blocks $B = (h_{-1}, sl, pk, \sigma)$ and $\tilde{B} = (\tilde{h}_{-1}, h_{-1}, sl, pk, X, nonce, \sigma)$. B denotes *leader* block that is an empty block created by the elected parties of each slot and $\tilde{B}$ denotes *transaction* block that consists of messages created by a party if his former *leader* block B' has been confirmed by the honest parties.

- $B_{i'} = (h_{-1}, sl_j^{e_i}, pk_{i'}, \sigma_{i'})$ and $\mathcal{V}(B_{i'}) = 1$ if and only if:
  - $h_{-1} = \mathcal{H}(B_{i'-1})$. $B_{i'}$ links to its parent *leader* block correctly.
  - $B_{i'}.sl > B_{i'-1}.sl$. Leader chain with a strictly increasing sequence of time.
  - $\mathcal{H}(nonce^{e_i}, sl_j^{e_i}, pk_{i'}) < s * T^{e_i}$. Party $P_{i'}$ with address $pk_{i'}$ and stake s is the leader of $sl_j^{e_i}$ exactly, where $nonce^{e_i}$ determines the leader election function of e_i, T^{e_i} is the difficult target of e_i and $s = 1$.
  - $Ver_{pk_{i'}}([B_{i'}], \sigma_{i'}) = 1$. The signature on $(h_{-1}, sl_j^{e_i}, pk_{i'})$ under $P_{i'}$'s signing key is correct.

– $\tilde{B}_{j'} = (\tilde{h}_{-1}, h'_{-1}, sl_j^{e_i}, pk_{j'}, X, nonce_{j'}, \sigma_{j'})$ and $\tilde{\mathcal{V}}(\tilde{B}_{j'}) = 1$ if and only if:
  - $\tilde{h}_{-1} = \mathcal{H}(\tilde{B}_{j'-1})$. $\tilde{B}_{j'}$ links to its parent *transaction* block correctly.
  - $\tilde{B}_{j'}.sl > \tilde{B}_{j'-1}.sl$. Transaction chain with a strictly increasing sequence of time.
  - $h'_{-1} = \mathcal{H}(B')$. $\tilde{B}_{j'}$ links to its parent *leader* block B' correctly.
  - $Fresh(B') = 1$. B' is the latest confirmed *leader* block by honest parties up to the beginning of $sl_j^{e_i}$.
  - $\tilde{B}_{j'}.pk = B'.pk$. The creator of a confirmed *leader* block $pk_{j'}$ is eligible to create a *transaction* block.
  - $V(X) = 1$. The messages packed in *transaction* block are valid.
  - $nonce_{j'} \in_R \{0,1\}^k$. A random value that is sampled uniformly from $\{0,1\}^k$ and used for generating $nonce^{e_{i+1}}$.
  - $Ver_{pk_{j'}}([\tilde{B}_{j'}], \sigma_{j'}) = 1$. The signature on $(h_{-1}, h'_{-1}, sl_j^{e_i}, pk_{j'}, X)$ under $P_{j'}$'s signing key is correct.

In our protocol, a blockchain $\mathcal{C}^* = \{\mathcal{C}, \tilde{\mathcal{C}}\}$ consists of two chains called *leader* chain $\mathcal{C} = B_0, B_1, ..., B_n$ and *transaction* chain $\tilde{\mathcal{C}} = \tilde{B}_1, ..., \tilde{B}_m$, where B_0 is genesis block created by initial parties, B_n and $\tilde{B}_m$ are the heads of $\mathcal{C}$ and $\tilde{\mathcal{C}}$. Let $len(\mathcal{C}) = n + 1$ and $len(\tilde{\mathcal{C}}) = m$ denote the length of $\mathcal{C}$ and $\tilde{\mathcal{C}}$ respectively. What's more, $n - m \geq K$, where $K \in \mathbb{N}$ is the parameter of *common prefix* property. Let $\mathcal{C}^{\lceil\kappa}$ denotes a chain by pruning the κ rightmost blocks of $\mathcal{C}$ and if $\kappa \geq len(\mathcal{C})$, then $\mathcal{C}^{\lceil\kappa} = \epsilon$ is an empty string. $\mathcal{C}_1 \preceq \mathcal{C}_2$ means that $\mathcal{C}_1$ is a prefix of $\mathcal{C}_2$. $\mathcal{C}^*$ is valid if $\mathcal{V}(B_i) = 1$ and $\tilde{\mathcal{V}}(\tilde{B}_j) = 1$, where $B_i \in \mathcal{C}$ $(i \in \{1, ..., n\})$ and $\tilde{B}_j \in \tilde{\mathcal{C}}$ $(j \in \{1, ..., m\})$. More formally, $\mathcal{C}^*$ is pictured in Fig.2.

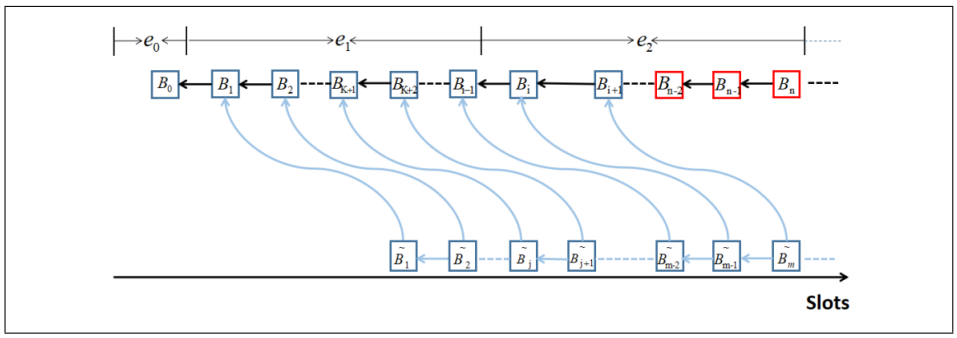

Fig. 2. UniqueChain Structure
The black arrows denote *leader* chain $\mathcal{C}$ and the blue arrows denote the *transaction* chain $\tilde{\mathcal{C}}$. Dark blue blocks denote the blocks that have been confirmed by honest parties and the red blocks denote the unstable blocks. $\tilde{\mathcal{C}}$ starts when block B_1 has been backed by K blocks. Block B_{i+1} is the latest confirmed block held by honest parties and $n - m \geq K$. Note that $\mathcal{C}$ and $\tilde{\mathcal{C}}$ do not grow synchronously at some slots.

Two Types of Transactions. For the secure joining of new parties, we introduce two types of transactions Tx_g and Tx_r. Tx_g denotes the general trans-

actions between payer and payee, and Tx_r is the requesting transaction that enables the new parties to obtain a trusted set of *leader* chains provided by the most recent elected parties. Note that *transaction* Tx *redeems* (*spends*) *transaction* Tx' means that *the output*(s) *of* Tx' *is the input*(s) *of* Tx.

- $Tx_g = (h_{-1}, v, sl, \pi, b = 0, \omega)$ is treated as a general transaction that $P_{i'}$ pays for $P_{j'}$, where h_{-1} is index of the spent transactions Tx'_g, v is transaction value, sl denotes the time that Tx_g is created, π specifies the conditions that Tx_g can be spent, $b = 0$ indicates that Tx_g is a general transaction and ω is witness to make Tx_g be evaluated true $Tx'_g.\pi(Tx_g) = 1$.
- $Tx_r = (h_{-1}, v, sl, \pi, b = 1, \omega)$ is requesting transaction with at least one valid input and $\tilde{l}$ outputs broadcasted by new parties who intend to join and maintain protocol execution. $\tilde{l} = p * |E| * R$ is the expected number of parties being elected during epoch e_i. The other notations $h_{-1}, v, sl, \pi, \omega$ are the same as in Tx_g, $b = 1$ denotes that Tx_r is a requesting transaction. The $i'th$ output of Tx_r can be redeemed by $Tx_g = (h_{-1}, v, sl, \pi, b = 0, \omega)$ broadcasted by party $P_{j'}$ with address $pk_{j'}$ successfully $(Tx_r.\pi(Tx_g) = 1)$ if
  - h_{-1} is index of Tx_r and $v = 0$.
  - $Tx_g.sl = Tx_r.sl + 1$ for $Tx_r.sl < sl_R^{e_i}$ or $Tx_g.sl = sl_1^{e_{i+1}}$ for $Tx_r.sl = sl_R^{e_i}$.
  - $\omega = (sl_{j''}^{e_i}, c, \sigma)$
    - $\mathcal{H}(pk_{j'}, sl_{j''}^{e_i}, nonce^{e_i}) < T^{e_i}$. $P_{j'}$ is a leader of slot $sl_{j''}^{e_i}$.
    - $c = Enc_{pk}(\mathcal{C}'_{loc})$. The encryption of the latest l consecutive blocks $\mathcal{C}'_{loc}$ of $P_{j'}$'s local *leader* chain, where $l > K$.
    - $Ver_{pk_{j'}}([Tx_g], \sigma) = 1$. The signature on $(h_{-1}, v, sl, \pi, b = 0, sl_{j''}^{e_i}, c)$ under $P_{j'}$'s signing key is correct.
  - $\mathcal{H}^*(pk_{j'}, sl_{j''}^{e_i}, nonce^{e_i})\ mod\ \tilde{l} + 1 = i'$. Tx_g can redeem the $i'th$ $(i' \in \{1, ..., \tilde{l}\})$ output of Tx_r.

Remarks. With the *uniqueness* property of *transaction* chains held by honest parties (section 5.1), and the tight relation between *leader* chains and *transaction* chains, new parties can only be provided with *leader* chains. Here we explain that our method is practical. (1). It is feasible to treat new parties as ordinary users to broadcast transactions. (2). Tx_r with value $v = 0$ is just treated as a mechanism to help new parties to initiate securely and not a permission for joining the system. Honest parties are willing to provide new parties with local states honestly to increase their power and further guarantee security of system and honest parties' interests. (3). In slot-synchronous network, Tx_r must be received by honest parties at the end of $Tx_r.sl$ and the corresponding redeeming transactions Tx_g broadcasted by honest parties must be received by new parties at the end of the next slot, so that new parties can be initialized within two slots. (4). Under honest majority assumption, more than half of elected parties of an epoch are honest and $\mathcal{H}^*(pk_{j'}, sl_{j''}^{e_i}, nonce^{e_i})\ mod\ \tilde{l} + 1 = i'$ indicates that $P_{j'}$ can only redeem the $i'th$ output of Tx_r successfully, so that more than half of chains obtained by new parties are provided by honest parties. (5). c ensures that only new parties who have sent requesting transactions can get the set of chains and c can be not verified publicly for (4).

2.3 Overview of Protocol Π^{UC}

In this section, we present a high overview of protocol Π^{UC}. In our protocol, the two chains $\mathcal{C}$ and $\tilde{\mathcal{C}}$ are plaited tightly. Precisely, as described in section 2.2 and pictured in Fig.2, a *leader* block B_i links to its former block B_{i-1}, a *transaction* block $\tilde{B}_j$ links to its former block $\tilde{B}_{j-1}$ and a confirmed *leader* block B' that has been in the common part of honest parties' local chains. Informally, our protocol consists of four phases and proceeds as follows.

1.Initialization. The initial epoch e_0 consists of two slots $sl_1^{e_0}$ and $sl_2^{e_0}$. The initial parties agree on a difficult target T^{e_1} and a random value $nonce^{e_1}$ for the first epoch e_1 together, where T^{e_1} is determined by the distribution of initial parties' stakes and $nonce^{e_1}$ is a random beacon for selecting leaders. More concretely, each party P_i first computes and broadcasts the commitment of his stake s_i and uniformly selected values r_i, r_i' as $Com(s_i, r_i; r_i')$. Then he collects the received commitments at the end of $sl_1^{e_0}$, opens and broadcasts messages as (s_i, r_i). Finally, he collects all the valid openings and computes the genesis block as $B_0 := (T^{e_1}, nonce^{e_1})$ locally at the end of $sl_2^{e_0}$. What's more, based on the *uniqueness* property of *transcation* chains held by honest parties, T^{e_i} and $nonce^{e_i}$ are determined by the stakes distribution and the *nonce* in *transaction* blocks of epoch e_{i-1} ($i \in \{2, 3, ...\}$) respectively. During protocol execution, for the initialization of newly joining parties, they are bootstrapped by broadcasting the defined requesting transactions Tx_r at any slot.

2.Fetching Information from Network. During protocol execution, parties can collect information from network that consists of transactions, blockchains, ect. At slot $sl_j^{e_i}$, the elected parties of e_i extract the set of requesting transactions and provide the new parties with their local states (chains) by broadcasting the defined redeeming transactions. The elected parties of $sl_j^{e_i}$ extract the set of chains. The parties whose former *leader* blocks have been backed by K blocks extract the set of chains and transactions. The new parties extract the set of corresponding redeeming transactions to get a trusted set of chains.

3.Update Local State. Upon receiving information from network, each existing party compares them with local state and determines the best local state via executing protocol $BestValid'$ ($Fig.6$). For the new parties, they execute protocol $BestValid''$ ($Fig.7$) to determine the initial state (chain).

4.Extend Chain. At slot $sl_j^{e_i}$, each existing party tries to extend local chain. Formally, party $P_{i'}$ with local chain $\mathcal{C}_{i'}^* = (\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'})$ first determines whether he is elected as a leader by computing $\mathcal{H}(sl_j^{e_i}, pk_{i'}, nonce^{e_i}) \leq T^{e_i}$ and if it is true, he creates a *leader* block B to extend $\mathcal{C}_{i'}$. At the same time, if a *leader* block B' created by $P_{j'}$ with local chain $\mathcal{C}_{j'}^* = (\mathcal{C}_{j'}, \tilde{\mathcal{C}}_{j'})$ has been backed by K blocks in his local chain, then he creates a *transaction* block $\tilde{B}$ to extend $\tilde{\mathcal{C}}_{j'}$. Finally, the elected parties broadcast their updated local states (chains).

2.4 Security Properties

The security properties of blockchain protocols have been well defined [16, 21, 24]. In our protocol, we consider chain growth and chain quality of the *leader*

chains and *transaction* chains held by existing honest parties, common prefix of the *leader* chains held by existing honest parties and chain soundness [15] of *leader* chains held by new parties.

Definition 1 *(Chain Growth). Consider protocol Π^{UC}, chain growth property with parameters g and $\tilde{g}$ states that during protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, for any two existing honest parties P_1 and P_2 with local chains $\mathcal{C}_1^* = \{\mathcal{C}_1, \tilde{\mathcal{C}}_1\}$ at slot sl and $\mathcal{C}_2^* = \{\mathcal{C}_2, \tilde{\mathcal{C}}_2\}$ at slot sl' respectively. Let $t = sl' - sl > 0$, then it holds that $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq g \cdot t$ and $len(\tilde{\mathcal{C}}_2) - len(\tilde{\mathcal{C}}_1) \geq \tilde{g} \cdot t$, where $g \geq \tilde{g}$ are the lower bound of chain growth rate.*

Definition 2 *(Chain Quality). Consider protocol Π^{UC}, chain quality property with parameters $\mu \in (0,1)$, $\tilde{\mu} \in (0,1)$ and $l \in \mathbb{N}$ states that during protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, for any existing honest party P with local chain $\mathcal{C}^* = \{\mathcal{C}, \tilde{\mathcal{C}}\}$. It holds that for any large enough l consecutive blocks of $\mathcal{C}$ and $\tilde{\mathcal{C}}$, the ratios of blocks created by adversary are at most μ and $\tilde{\mu}$ respectively.*

Definition 3 *(Common Prefix of leader chain). Consider protocol Π^{UC}, common prefix property with parameter $K \in \mathbb{N}$ states that during protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, for any two existing honest parties P_1 and P_2 with local chains $\mathcal{C}_1^* = \{\mathcal{C}_1, \tilde{\mathcal{C}}_1\}$ at slot sl and $\mathcal{C}_2^* = \{\mathcal{C}_2, \tilde{\mathcal{C}}_2\}$ at slot sl' respectively. Let $sl' \geq sl$, then it holds that $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$.*

Definition 4 *(Soundness of leader chain). Consider protocol Π^{UC}, soundness property with parameter $K \in \mathbb{N}$ states that during protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, for a new party P_1 initialized with chain $\mathcal{C}_1^* = \{\mathcal{C}_1, \tilde{\mathcal{C}}_1\}$ at slot sl and an existing honest party P_2 with local chain $\mathcal{C}_2^* = \{\mathcal{C}_2, \tilde{\mathcal{C}}_2\}$ at slot sl, then it holds that $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil K} \preceq \mathcal{C}_1$.*

Based on the above four properties, we prove that the *transaction* chains held by honest parties satisfy uniqueness and further high efficiency of handling messages is achieved.

Definition 5 *(Uniqueness of transaction chain). Consider protocol Π^{UC}, uniqueness property states that during protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, for any two honest parties P_1 and P_2 with local chains $\mathcal{C}_1^* = \{\mathcal{C}_1, \tilde{\mathcal{C}}_1\}$ and $\mathcal{C}_2^* = \{\mathcal{C}_2, \tilde{\mathcal{C}}_2\}$ at the end of slot sl respectively. Then it holds that $\tilde{\mathcal{C}}_1 = \tilde{\mathcal{C}}_2$.*

Definition 6 *(High Efficiency). Consider protocol Π^{UC}, high efficiency property states that during protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, if a transaction block $\tilde{B}$ has been confirmed by an existing honest party P at slot sl, then $\tilde{B}$ will be confirmed finally by all the existing honest parties at the end of sl.*

3 Ideal Functionalities

Following Canetti's formulation of the *real world* protocol executions [4], we present our blockchain protocol Π^{UC} ($UniqueChain$) in the $\{\mathcal{F}_{init}\mathcal{F}_{res}, \mathcal{F}_{NET}\}$-hybrid model. Formally, the execution of Π^{UC} is directed by an environment

$\mathcal{Z}(1^k)$ with security parameter k. $\mathcal{Z}$ activates a number of parties (stakeholders) with inputs X. Moreover, $\mathcal{Z}$ communicates with a mildly adaptive adversary $\mathcal{A}$ who controls a certain number of parties freely. In this section, we introduce the ideal functionalities used in our protocol and their implementations are presented in appendix A and B.

$\mathcal{F}_{init}$**.** At the beginning of protocol execution, a genesis block that consists of the difficult target T^{e_1} and random value $nonce^{e_1}$ of epoch e_1 is created. Formally, $\mathcal{F}_{init}$ is parameterized by the initial parties $P_1, ..., P_n$ and their respective stakes $s_1, ..., s_n$, and proceeds as follows. In genesis epoch $e_0 = \{sl_1^{e_0}, sl_2^{e_0}\}$, based on the stakes distribution of initial parties, $\mathcal{F}_{init}$ computes T^{e_1} and choose a random value $nonce^{e_1} \in_R \{0,1\}^k$, and then generates genesis block as $B_0 = (T^{e_1}, nonce^{e_1})$. Now, the initial parties are initialized and get into epoch e_1 to start protocol execution. In non-genesis epoch, upon receiving the request of secure joining from new parties, $\mathcal{F}_{init}$ returns B_0.

$\mathcal{F}_{res}$**.** At any slot, each party has access to $\mathcal{F}_{res}$ to determine if he is eligible to create a *leader* block or a *transaction* block and ask for the validity of blocks. Formally, each party first sends the *register* command for joining and *unregister* command for leaving the execution. At any slot, $\mathcal{F}_{res}$ grants each registered party with one unit of stake, sets a party as leader with probability p. What's more, $\mathcal{F}_{res}$ maintains a set $(Ctr, \mathcal{P}) = \{(Ctr_j^{e_i}, \mathcal{P}_j^{e_i})\}_{i\in\mathbb{N},\ j\in\{1,...,R\}}$, where $Ctr_j^{e_i}$ is a counter and $Ctr_b^{e_0} = 0$ $(b \in \{1,2\})$, and $\mathcal{P}_j^{e_i}$ is a set of elected parties at $sl_j^{e_i}$ and initialized to ϕ (an empty set). If at least one parties are elected at $sl_j^{e_i}$, then set $Ctr_j^{e_i} = Ctr_{j-1}^{e_i} + 1$ and adds the corresponding elected parties to $\mathcal{P}_j^{e_i}$, otherwise, set $Ctr_j^{e_i} = Ctr_{j-1}^{e_i}$ and $\mathcal{P}_j^{e_i} = \phi$. Each elected party can create a *leader* block. Further, if $Ctr_j^{e_i} - K = Ctr_{j'}^{e_i} > 0$ and $\mathcal{P}_{j'}^{e_i} \neq \phi$, then $\mathcal{F}_{res}$ uniformly selects a party $P \in \mathcal{P}_{j'}^{e_i}$ who is eligible to create a *transaction* block, where $Ctr_j^{e_i}$ is the current counter. At any time, each party has access to the verification process of $\mathcal{F}_{res}$ to verify blocks. The formal description of $\mathcal{F}_{res}$ is given in Fig.4.

FUNCTIONALITY $\mathcal{F}_{init}$

$\mathcal{F}_{init}$ is parameterized by initial parties $P_1, ..., P_n$ and their respective stakes $s_1, ..., s_n$, interacts with initial parties, new parties and ideal adversary $\mathcal{S}$. Upon receiving any forms of messages, it first gets the current slot from local clock and proceeds as follows.

- Upon receiving $(Initialize, P_{i'}, s_{i'})$ from $P_{i'}$ at $sl_j^{e_i}$ $(i \in \mathbb{N}, j \in \{1, ..., R\})$
  - if $i = 0$ and $j = 1$, then
    - computes the difficult target T^{e_1}, samples a random value $nonce^{e_1} \in_R \{0,1\}^k$ and creates a genesis block $B_0 = (T^{e_1}, nonce^{e_1})$, then stores B_0 and sends $(Initialized, B_0)$ to $P_{i'}$ at $sl_2^{e_0}$.
  - else, if $i > 0$ and B_0 has been created, then sends $(Initialized, B_0)$ to $P_{i'}$.
  - else, sends $(Error)$ to $P_{i'}$.

Fig. 3. Initialization Functionality $\mathcal{F}_{init}$

FUNCTIONALITY $\mathcal{F}_{res}$

$\mathcal{F}_{res}$ is parameterized by probability p, security parameter k, interacts with an ideal adversary $\mathcal{S}$ and a set of parties E.

- **Registration**
  1. Upon receiving $(Register, P_{i'})$ from party $P_{i'} \in E$, if there is a record $(P_{i'}, re_{i'} = 1)$, then ignore the message. Otherwise, send $(Register, P_{i'})$ to $\mathcal{S}$. Upon receiving $(Registered, P_{i'})$ from $\mathcal{S}$, then record $(P_{i'}, re_{i'} = 1)$ and send $(Registered)$ to $P_{i'}$. ($P_{i'}$ registered)
  2. Upon receiving $(Unregister, P_{i'})$ from party $P_{i'} \in E$, if there is no record $(P_{i'}, re_{i'} = 1)$, then ignore the message. Otherwise, update record $(P_{i'}, re_{i'} = 0)$ and send $(Unregistered)$ to $P_{i'}$. ($P_{i'}$ unregistered)
- **Stake Election.** At slot $sl_j^{e_i}$ $(i \in \mathbb{N}, j \in \{1, ..., R\})$.
  1. **Creating Leader Block**: we set $Ctr_j^{e_i} = Ctr_{j-1}^{e_i}$ or $Ctr_j^{e_i} = Ctr_R^{e_{i-1}}$ for $j = 1$, and $\mathcal{P}_j^{e_i} = \phi$, every registered party is granted with one unite stake $s_{j,i'}^{e_i} = 1$.
     - Upon receiving $(L\text{-}Elect, P_{i'})$ from $P_{i'}$, proceed as follows.
       * If there is a record $(P_{i'}, re_{i'} = 1, s_{j,i'}^{e_i} = 1)$, then
         · with probability p, choose $h \in_R \{0,1\}^k$, then set $\mathcal{P}_j^{e_i} = \mathcal{P}_j^{e_i} \cup P_{i'}$, record the entry $((P_{i'}, re_{i'} = 1, s_{j,i'}^{e_i} = 0), \mathcal{P}_j^{e_i})$ and send $(L\text{-}Elected, P_{i'}, f = 1)$ to $P_{i'}$. ($P_{i'}$ is elected)
         · with probability $1 - p$, record the entry $((P_{i'}, re_{i'} = 1, s_{j,i'}^{e_i} = 0), \mathcal{P}_j^{e_i})$ and send $(L\text{-}Elected, P_{i'}, f = 0)$ to $P_{i'}$. ($P_{i'}$ is not elected)
       * Otherwise, send $(L\text{-}Elected, P_{i'}, f = 0)$ to $P_{i'}$. ($P_{i'}$ is not elected)

       If $\mathcal{P}_j^{e_i} \neq \phi$, then set $Ctr_j^{e_i} = Ctr_j^{e_i} + 1$. Update record as $(\mathcal{P}_j^{e_i}, Ctr_j^{e_i})$. (Record the elected parties $\mathcal{P}_j^{e_i}$ and the corresponding counter $Ctr_j^{e_i}$ of $sl_j^{e_i}$)
     - Upon receiving $(Compute, B_{-1}, P_{i'})$ from $P_{i'}$. (Compute the index of the former *leader* block)
       * If $P_{i'} \in \mathcal{P}_j^{e_i}$ and there is a record (B_{-1}, h_{-1}), then send $(Computed, h_{-1})$ to $P_{i'}$. Otherwise, choose a random value $h_{-1} \in \{0,1\}^k$, record (B_{-1}, h_{-1}) and send $(Computed, h_{-1})$ to $P_{i'}$.
       * Otherwise, send $(Error)$ to $P_{i'}$.
     - Upon receiving $(Sign, P_{i'}, B)$ from $P_{i'}$.
       * If there is a record $P_{i'} \in \mathcal{P}_j^{e_i}$, then send $(Sign, P_{i'}, B)$ to adversary. Upon receiving $(Signed, (P_{i'}, B), \sigma)$ from the adversary, record (B, σ) and then send $(Signed, (B, \sigma))$ to $P_{i'}$.
       * Otherwise, send $(Error)$ to $P_{i'}$.
  2. **Creating Transaction Block**:
     - Upon receiving $(T\text{-}Elect, P_{j'})$ from $P_{j'}$, then compute $Ctr_j^{e_i} - K = Ctr_{j''}^{e_i}$. If $Ctr_{j''}^{e_i} > 0$ and $\mathcal{P}_{j''}^{e_i} \neq \phi$, then uniformly choose a party $P_{\tilde{j}} \in_R \mathcal{P}_{j''}^{e_i}$. (Only one party $P_{\tilde{j}}$ is elected to create transaction block).
       * If $P_{j'} = P_{\tilde{j}}$, then record $(P_{j'}, Ctr_j^{e_i}, Ctr_{j''}^{e_i})$ and send $(T\text{-}Elected, P_{j'}, \tilde{f} = 1)$ to $P_{j'}$. (P_j is elected)
       * Otherwise, send $(T\text{-}Elected, P_{j'}, \tilde{f} = 0)$ to $P_{j'}$. ($P_{j'}$ is not elected)
     - Upon receiving $(Compute, \tilde{B}_{-1}, B_{-1}, P_{j'})$. (Compute the index of the former *transaction* block and the corresponding *leader* block)
       * If there is a record $(P_{j'}, Ctr_j^{e_i}, Ctr_{j''}^{e_i})$, then if there is a record $((\tilde{B}_{-1}, \tilde{h}_{-1}), (B_{-1}, h_{-1}))$, send $(Computed, \tilde{h}_{-1}, h_{-1})$ to $P_{j'}$. Otherwise, choose $\tilde{h}_{-1}, h_{-1} \in \{0,1\}^k$, record $((\tilde{B}_{-1}, \tilde{h}_{-1}), (B_{-1}, h_{-1}))$ and send $(Computed, \tilde{h}_{-1}, h_{-1})$ to $P_{j'}$.
       * Otherwise, send $(Error)$ to $P_{j'}$.
     - Upon receiving $(Sign, P_{j'}, \tilde{B})$ from $P_{j'}$.
       * If there is a record $(P_{j'}, Ctr_j^{e_i}, Ctr_{j''}^{e_i})$, then send $(Sign, P_{j'}, \tilde{B})$ to adversary. Upon receiving $(Signed, (\tilde{B}, \tilde{\sigma}))$ from the adversary, then record $(\tilde{B}, \tilde{\sigma})$ and send $(Signed, (\tilde{B}, \tilde{\sigma}))$ to $P_{j'}$.
       * Otherwise, send $(Error)$ to $P_{j'}$.
- **Verification** Upon receiving $(Verify, P_{i'}', B(\tilde{B}), \sigma(\tilde{\sigma}))$ from a party $P_{i'} \in E$.
  - If there is a record of the form $(P_{i'}', B(\tilde{B}), \sigma(\tilde{\sigma}))$, then send $f'(\tilde{f}') = 1$ to $P_{i'}$.
  - Otherwise, send $f'(\tilde{f}') = 0$ to $P_{i'}$.

Fig. 4. Resource functionality $\mathcal{F}_{res}$

4 Protocol Π^{UC}

In this section, we present the detailed description of our protocol Π^{UC} in the $\{\mathcal{F}_{init}, \mathcal{F}_{res}, \mathcal{F}_{NET}\}$-hybrid model and give the corresponding sub-protocols.

4.1 The Formal Description of Π^{UC}

We now present protocol Π^{UC} in $\{\mathcal{F}_{init}\mathcal{F}_{res}, \mathcal{F}_{NET}\}$-hybrid model. First each party is initialized via $\mathcal{F}_{init}$ to get genesis block B_0 and then gets information from the network via $\mathcal{F}_{NET}$ that consists of blockchains, transactions, ect. Note that new parties only extract the corresponding redeeming transactions received from network. Then, each party performs some validations via $BestValid'$ (in Fig.6)or $BestValid''$ (in Fig.7) to get the best local state and try to extend local chain via $\mathcal{F}_{res}$. Finally, each party updates and broadcasts local state via $\mathcal{F}_{NET}$. More details are presented in Fig.5.

4.2 The Best Chain Algorithm: $BestValid$

In decentralized setting, at any slot $sl_j^{e_i}$ ($i \in \mathbb{N}$, $j \in \{1, ..., R\}$) each party determines the local state independently. Algorithm $BestValid$ allows honest parties to hold the best local states. In our protocol, we introduce two algorithms $BestValid'$ and $BestValid''$ for the existing parties and new parties respectively.

$BestValid'$**.** It is parameterized with two content validation predicates $\mathcal{V}(\cdot)$ and $\tilde{\mathcal{V}}(\cdot)$ to determine the validity of *leader* blocks and *transaction* blocks, and parameter $K \in \mathbb{N}$. It takes a set of chains $\mathbb{C}_j^{e_i}$ and party $P_{i'}$'s local chain $\mathcal{C}_{loc}^*$ as inputs and proceeds as follows. A detailed description is showed in Fig.6.

- Discard chains in $\mathbb{C}_j^{e_i}$ that the *leader* chains fork more than K blocks or the *transaction* chains fork more than one blocks from local one. This comes from the *common prefix* property of *leader* chains and the uniqueness of *transaction* chains held by honest parties (theorem 3 and theorem 7).
- Discard invalid chains in the updated set $\mathbb{C}_{j,1}^{e_i}$. Validation predicates $\mathcal{V}(\cdot)$ and $\tilde{\mathcal{V}}(\cdot)$ evaluate every *leader* block and *transaction* block in $\mathbb{C}_{j,1}^{e_i}$ sequentially.
- Discard chains in the updated set $\mathbb{C}_{j,2}^{e_i}$, where there are more than one different blocks with the same slot created by a same party. The elected adversary may create several valid blocks and send to different honest parties. What's more we believe that each honest elected party only creates one valid block.
- Compare local chain with the chains in the updated set $\mathbb{C}_{j,3}^{e_i}$ and determine the best local one.

$BestValid''$**.** It is parameterized with parameters $K \in \mathbb{N}$ and $l > K$, and content validation predicates $\mathcal{V}(\cdot)$ and $\tilde{\mathcal{V}}(\cdot)$. It takes two sets of chains $\mathbb{C}_j^{e_i}$ and $\mathbb{C}'^{e_i}_j$ as inputs and proceeds as follows. A detailed description is showed in Fig.7.

- Discard chain $\mathcal{C}'_{i'} \in \mathbb{C}'^{e_i}_j$ if $len(\mathcal{C}'_{i'}) \neq l$, where $i' \in \{1, ..., |\mathbb{C}'^{e_i}_j|\}$. We believe honest parties must provide chains correctly. What's more, $l > K$ ensures new parties to get the latest confirmed part of honest parties' local chains, where l is the parameter of *chain quality* property of honest *leader* chains.

PROTOCOL Π^{UC}

The protocol is parameterized by content validation predicates $\mathcal{V}$ and $\tilde{\mathcal{V}}$, interacts with parties $P_{i'}$ ($i' \in \{1, 2, ...\}$) and adversary $\mathcal{A}$ at slot $sl_j^{e_i}$ ($i \in \mathbb{N}, j \in \{1, 2, ..., R\}$) of epoch e_i. Proceeds as follows.

1. **Initialization.**
   - if $i = 0$ and $j = 1$, then party $P_{i'}$ sends $(Initialize, P_{i'}, s_{i'})$ to $\mathcal{F}_{init}$ and gets $(Initialized, B_0)$.
   - else, party $P_{i'}$ sends $(Initialize, P_{i'})$ to $\mathcal{F}_{init}$, gets $(Initialized, B_0)$ and sends $(Broadcast, Tx_r, P_{i'})$ to $\mathcal{F}_{NET}$.

   Return $(Initialized, P_{i'})$ to the environment $\mathcal{Z}$.
2. **Fetching Information from Network.** Each party fetches information from $\mathcal{F}_{NET}$.
   - Collect the information $\mathcal{M}_j^{e_i}$ received during $sl_j^{e_i}$.
     - the existing elected parties of $sl_j^{e_i}$ extract the set of chains $\mathbb{C}_j^{e_i}$ from $\mathcal{M}_j^{e_i}$. (The elected parties of $sl_j^{e_i}$ are eligible to extend *leader* chain).
     - the existing elected parties of e_i
       - extract the requesting transactions $\mathbb{T}_{j,r}^{e_i}$ from $\mathcal{M}_j^{e_i}$.
       - create and broadcast general transactions Tx_g to redeem transactions in $\mathbb{T}_{j,r}^{e_i}$. (The elected parties of e_i provide the new parties with their local states by broadcasting the redeeming transactions).
     - the parties whose former *leader* blocks have been backed by K blocks extract the set of chains $\mathbb{C}_j^{e_i}$ and the set of transactions $\mathbb{T}_j^{e_i} = \{\mathbb{T}_{j,g}^{e_i}, \mathbb{T}_{j,r}^{e_i}\}$ from $\mathcal{M}_j^{e_i}$. (The parties are eligible to create *transaction* blocks that consist of payloads)
     - the new parties
       - extract the set of chains $\mathbb{C}_j^{e_i}$ from $\mathcal{M}_j^{e_i}$.
       - extract the corresponding valid redeeming transactions Tx_g from $\mathbb{T}_{j,g}^{e_i}$, decrypt c in each Tx_g and get a set of chains $\mathbb{C}'^{e_i}_j$. (The new parties get a trusted set of *leader* chains).
3. **Update Local State.** After receiving information from $\mathcal{F}_{NET}$, then
   - each existing party with local chain $\mathcal{C}_{loc}^*$, updates local chain as $\mathcal{C}_{loc}^* := BestValid'(\mathcal{C}_{loc}^*, \mathbb{C}_j^{e_i})$.
   - each new party gets the initial local state as $\mathcal{C}_{loc}^* := BestValid''(\mathbb{C}'^{e_i}_j, \mathbb{C}_j^{e_i})$.
4. **Extend Chain.** Each party $P_{i'}$ tries to extend local chain $\mathcal{C}_{loc}^* = \{\mathcal{C}, \tilde{\mathcal{C}}\}$. We assume that $P_{i'}$ has registered to $\mathcal{F}_{res}$ and been granted with stakes $s_{i'}$.
   - Upon receiving $(Input\text{-}Stake, P_{i'})$ from environment $\mathcal{Z}$, $P_{i'}$ extends chain $\mathcal{C}$.
     - send $(L\text{-}Elect, B, P_{i'})$ to $\mathcal{F}_{res}$ and then receive $(L\text{-}Elected, P_{i'}, f)$.
     - if $f = 1$, then send $(Compute, B_{-1}, P_{i'})$ to $\mathcal{F}_{res}$ and receive $(Computed, h_{-1})$.
     - send $(Sign, P_{i'}, B)$ to $\mathcal{F}_{res}$ and receive $(Signed, (B, \sigma))$.
     - set $\mathcal{C} := \mathcal{C} \,||\, B$, $\mathcal{C}_{loc}^* := \{\mathcal{C}, \tilde{\mathcal{C}}\}$ and send $(Broadcast, \mathcal{C}_{loc}^*, P_{i'})$ to $\mathcal{F}_{NET}$.

     Return $(Return\text{-}Stake, P_{i'})$ to environment $\mathcal{Z}$.
   - Upon receiving $(Input\text{-}Stake, X, P_{i'})$ from the environments $\mathcal{Z}$, where X is the block payloads, $P_{i'}$ extends chain $\tilde{\mathcal{C}}$.
     - send $(T\text{-}Elect, P_{i'})$ to $\mathcal{F}_{res}$ and receive $(T\text{-}Elected, P_{i'}, \tilde{f})$.
     - if $\tilde{f} = 1$, then send $(Compute, \tilde{B}_{-1}, B_{-1}, P_{i'})$ to $\mathcal{F}_{res}$ and receive $(Computed, \tilde{h}_{-1}, h_{-1})$.
     - send $(Sign, P_{i'}, \tilde{B})$ to $\mathcal{F}_{res}$ and receive $(Signed, (P_{i'}, \tilde{B}), \tilde{\sigma})$.
     - set $\tilde{\mathcal{C}} := \tilde{\mathcal{C}} \,||\, \tilde{B}$, $\mathcal{C}_{loc}^* := \{\mathcal{C}, \tilde{\mathcal{C}}\}$ and send $(Broadcast, \mathcal{C}_{loc}^*, P_{i'})$ to $\mathcal{F}_{NET}$.

     Return $(Return\text{-}Stake, P_{i'})$ to environment $\mathcal{Z}$.

Fig. 5. The Unique Chain Protocol Π^{UC}

$BestValid'$

$BestValid'$ with parameter $K \in \mathbb{N}$ and two content validation predicates $\mathcal{V}(\cdot)$ and $\tilde{\mathcal{V}}(\cdot)$. At slot $sl_j^{e_i}$ $(i \in \mathbb{N},\ j \in \{1, ..., R\})$, it takes a set of chains $\mathbb{C}_j^{e_i} := \{(\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'}), i' \in \{1, 2, ..., |\mathbb{C}_j^{e_i}|\}\}$ and party P's local chain $\mathcal{C}_{loc}^* := (\mathcal{C}, \tilde{\mathcal{C}})$ as inputs and proceeds as follows.

1. For each $\mathcal{C}_{i'}^* = (\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'}) \in \mathbb{C}_j^{e_i}$, do
   - if $\mathcal{C}_{i'}$ forks from $\mathcal{C}_{loc}$ with more than K blocks or $\tilde{\mathcal{C}}_{i'}$ forks from $\tilde{\mathcal{C}}_{loc}$ with more than one blocks, then
     - remove $(\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'})$ from $\mathbb{C}_j^{e_i}$ and set $\mathbb{C}_{j,1}^{e_i} := \mathbb{C}_j^{e_i}/\mathcal{C}_{i'}^*$
2. For each $\mathcal{C}_{i'}^* = (\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'}) \in \mathbb{C}_{j,1}^{e_i}$, where $i' \in \{1, .., |\mathbb{C}_{j,1}^{e_i}|\}$, do
   - for each *leader* block $B \in \mathcal{C}_{i'}$ and *transaction* block $\tilde{B} \in \tilde{\mathcal{C}}_{i'}$
     - if $\mathcal{V}(B) \neq 1$ or $\tilde{\mathcal{V}}(\tilde{B}) \neq 1$, then
       - remove $(\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'})$ from $\mathbb{C}_{j,1}^{e_i}$ and set $\mathbb{C}_{j,2}^{e_i} = \mathbb{C}_{j,1}^{e_i}/\mathcal{C}_{i'}^*$
3. For each given $\mathcal{C}_{i'}^* \in \mathbb{C}_{j,2}^{e_i}$ $(i' \in \{1, .., |\mathbb{C}_{j,2}^{e_i}|\})$, do
   - for $\mathcal{C}_{j'}^* \in \mathbb{C}_{j,2}^{e_i}/\mathcal{C}_{i'}^*$, if $(B.sl = B'.sl \wedge B.pk = B'.pk) \vee (\tilde{B}.sl = \tilde{B}'.sl \wedge \tilde{B}.pk = \tilde{B}'.pk)$, where $B, \tilde{B} \in \mathcal{C}_{i'}^*,\ B', \tilde{B}' \in \mathcal{C}_{j'}^*$. Then
     - remove $\mathcal{C}_{i'}^*$ and $\mathcal{C}_{j'}^*$ from $\mathbb{C}_{j,2}^{e_i}$, and set $\mathbb{C}_{j,3}^{e_i} := \mathbb{C}_{j,2}^{e_i}/(\mathcal{C}_{i'}^*, \mathcal{C}_{j'}^*)$
4. For each $\mathcal{C}_{i'}^* = (\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'}) \in \mathbb{C}_{j,3}^{e_i}$, where $i' \in \{1, .., |\mathbb{C}_{j,3}^{e_i}|\}$, do
   - set $\mathcal{C}_{best}^* := \mathcal{C}_{loc}^* = (\mathcal{C}, \tilde{\mathcal{C}})$
   - if $len(\mathcal{C}_{i'}) > len(\mathcal{C})$ or $len(\tilde{\mathcal{C}}_{i'}) - len(\tilde{\mathcal{C}}) = 1$, then set $\mathcal{C}_{best}^* := \mathcal{C}_{i'}^* = (\mathcal{C}_{i'}, \tilde{\mathcal{C}}_{i'})$
   - otherwise, set $\mathcal{C}_{best}^* := \mathcal{C}_{loc}^* = (\mathcal{C}, \tilde{\mathcal{C}})$

Return $\mathcal{C}_{best}^*$

Fig. 6. The Best Valid Chain Protocol $BestValid'$

- Clarify the chains in the updated set $\mathbb{C}'^{e_i}_{j,1}$ into two sets $\mathbb{C}'^{e_i}_{j,11}$ and $\mathbb{C}'^{e_i}_{j,12}$. As described in $BestValid'$ that the *leader* chains held by honest parties cannot be forked by more than K blocks. Let $\mathbb{C}'^{e_i}_{j,11}$ be the set of chains that fork more than K blocks from the chains in $\mathbb{C}'^{e_i}_{j,12}$, so that ensures the chains held by honest parties are in one set.
- Compare the size of $\mathbb{C}'^{e_i}_{j,11}$ and $\mathbb{C}'^{e_i}_{j,12}$, and choose the one whose size is bigger than $\frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, i.e., $\mathbb{C}'^{e_i}_{j,11}$. Based on the security assumption, at any time of protocol execution, the majority of parties are honest, so are the elected parties during an epoch. As a result, we believe that $\mathbb{C}'^{e_i}_{j,11}$ must consists of all the chains sent by honest parties. Now, we stress that the new parties have determined the correct version of honest parties' local *leader* chains.
- Find a set of *transaction* chains from $\mathbb{C}_j^{e_i}$ that match with the *leader* chains in $\mathbb{C}'^{e_i}_{j,11}$ and get a set of chain pairs $\mathbb{C}''^{e_i}_{j,11}$. Clarify the set $\mathbb{C}''^{e_i}_{j,11}$ into several subsets $\{\mathbb{C}''^{e_i}_{j,111}, ..., \mathbb{C}''_{j,11n}\}$. in our protocol, the honest parties hold a same local *transaction* chain, which will be proved in the *uniqueness* property. So let $\mathbb{C}''^{e_i}_{j,11i}$ be the set of chain pairs that with a same *transaction* chain.

Without loss of generality, we assume that the size of $\mathbb{C}''^{e_i}_{j,111}$ is the biggest one and if $|\mathbb{C}''^{e_i}_{j,111}| > \frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, then continue the process. Otherwise, halt.
- Verify every chain in $\mathbb{C}''^{e_i}_{j,111}$ and discard the invalid ones. If the updated set $\mathbb{C}'''^{e_i}_{j,111}$ satisfies $|\mathbb{C}'''^{e_i}_{j,111}| > \frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, then uniformly choose a chain $\mathcal{C}'^{*} \in \mathbb{C}'''^{e_i}_{j,111}$ and set the corresponding chain $\mathcal{C}^{*} \in \mathbb{C}^{e_i}_{j}$ as the current local chain.

BestValid''

BestValid'' with parameters $K \in \mathbb{N}$ and $l > K$, and two content validation predicates $\mathcal{V}(\cdot)$ and $\tilde{\mathcal{V}}(\cdot)$. It takes two sets of chains $\mathbb{C}^{e_i}_{j}$ and $\mathbb{C}'^{e_i}_{j}$ as inputs, and proceeds as follows.

1. For each $\mathcal{C}'_{i'} \in \mathbb{C}'^{e_i}_{j}$, where $i' \in \{1, ..., |\mathbb{C}'^{e_i}_{j}|\}$, do
   - if $len(\mathcal{C}'_{i'}) \neq l$, then remove $\mathcal{C}'_{i'}$ from $\mathbb{C}'^{e_i}_{j}$ and set $\mathbb{C}'^{e_i}_{j,1} := \mathbb{C}'^{e_i}_{j}/\mathcal{C}'_{i'}$.
2. For a given chain $\mathcal{C}'_1 \in \mathbb{C}'^{e_i}_{j,1}$, set $\mathbb{C}'^{e_i}_{j,11} = \{\mathcal{C}'_1\}$ and $\mathbb{C}'^{e_i}_{j,12} = \phi$
   - for each chain $\mathcal{C}'_{i'} \in \mathbb{C}'^{e_i}_{j,1}$, where $i' \in \{2, ..., |\mathbb{C}'^{e_i}_{j,1}|\}$
     - if $\mathcal{C}'_{i'}$ forks more than K blocks from $\mathcal{C}'_1$, then set $\mathbb{C}'^{e_i}_{j,12} := \mathbb{C}'^{e_i}_{j,12} \cup \mathcal{C}'_{i'}$.
     - else, set $\mathbb{C}'^{e_i}_{j,11} := \mathbb{C}'^{e_i}_{j,11} \cup \mathcal{C}'_{i'}$.
3. Compute the size of sets $\mathbb{C}'^{e_i}_{j,11}$ and $\mathbb{C}'^{e_i}_{j,12}$ as n_{11} and n_{12}
   - if $n_{11} > \frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, then set $\mathbb{C}'^{e_i}_{j,11} := \mathbb{C}'^{e_i}_{j,11}$
   - else, if $n_{12} > \frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, then set $\mathbb{C}'^{e_i}_{j,11} := \mathbb{C}'^{e_i}_{j,12}$.
   - else, halt.
4. Find a set of *transaction* chains from $\mathbb{C}^{e_i}_{j}$ that match with the *leader* chains in $\mathbb{C}'^{e_i}_{j,11}$ and get a set of chain pairs $\mathbb{C}''^{e_i}_{j,11}$.
5. For a given chain pair $\mathcal{C}'^{*}_1 = (\mathcal{C}'_1, \tilde{\mathcal{C}}'_1) \in \mathbb{C}''^{e_i}_{j,11}$, set $\mathbb{C}''^{e_i}_{j,111} = \{(\mathcal{C}'_1, \tilde{\mathcal{C}}'_1)\}$.
   - for every chain $\mathcal{C}'^{*}_{i'} = (\mathcal{C}'_{i'}, \tilde{\mathcal{C}}'_{i'}) \in \mathbb{C}''^{e_i}_{j,11}$, where $i' \in \{2, ..., |\mathbb{C}''^{e_i}_{j,11}|\}$
     - if $\tilde{\mathcal{C}}'_{i'} = \tilde{\mathcal{C}}'_{h}$, where $1 \leq h < i'$, then set $\mathbb{C}''^{e_i}_{j,11h} := \mathbb{C}''^{e_i}_{j,11h} \cup (\mathcal{C}'_{i'}, \tilde{\mathcal{C}}'_{i'})$.
     - else, add $(\mathcal{C}'_{i'}, \tilde{\mathcal{C}}'_{i'})$ to $\mathbb{C}''^{e_i}_{j,11i'}$
   - set $\mathbb{C}''^{e_i}_{j,111}$ be the set with the biggest size in $\{(\mathbb{C}''^{e_i}_{j,11u}), 1 \leq u \leq |\mathbb{C}''^{e_i}_{j,11}|\}$
     - if $|\mathbb{C}''^{e_i}_{j,111}| > \frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, then for every chain $\mathcal{C}'^{*}_{i'} = (\mathcal{C}'_{i'}, \tilde{\mathcal{C}}'_{i'}) \in \mathbb{C}''^{e_i}_{j,111}$, where $i' \in \{1, ..., |\mathbb{C}''^{e_i}_{j,111}|\}$
       - for every $B \in \mathcal{C}'_{i'}$ and $\tilde{B} \in \tilde{\mathcal{C}}'_{i'}$, if $\mathcal{V}(B) \neq 1$ or $\tilde{\mathcal{V}}(\tilde{B}) \neq 1$, then remove $\mathcal{C}'^{*}_{i'}$ from $\mathbb{C}''^{e_i}_{j,111}$ and set $\mathbb{C}'''^{e_i}_{j,111} = \mathbb{C}''^{e_i}_{j,111}/\mathcal{C}'^{*}_{i'}$.
     - else, halt.
6. If $|\mathbb{C}'''^{e_i}_{j,111}| > \frac{1}{2}|\mathbb{C}'^{e_i}_{j}|$, then uniformly choose a chain $\mathcal{C}'^{*} = (\mathcal{C}', \tilde{\mathcal{C}}') \in \mathbb{C}'''^{e_i}_{j,111}$, find a chain $(\mathcal{C}, \tilde{\mathcal{C}}) \in \mathbb{C}^{e_i}_{j}$ that matches with $(\mathcal{C}', \tilde{\mathcal{C}}')$ and set $\mathcal{C}^{*}_{loc} := (\mathcal{C}_{loc}, \tilde{\mathcal{C}}_{loc}) = (\mathcal{C}, \tilde{\mathcal{C}})$

Return $\mathcal{C}^{*}_{loc}$

Fig. 7. The Best Valid Chain Protocol *BestValid''*

5 Security Analysis of Π^{UC}

In this section, we give a detailed security analysis of our protocol Π^{UC}. Formally, we first prove that Π^{UC} satisfies the security properties defined in section

2.4 and then conclude that Π^{UC} indeed solves the challenges of designing PoS based blockchain protocols in the open setting better.

Main Parameters. Before showing the proofs of security properties, we present some parameters that make the description of proofs more convenient.

1. Let $\alpha := 1 - (1-p)^{|H|}$ be the probability that at least one honest party is elected to create *leader* block at a given slot.
2. Let $\beta := 1 - (1-p)^{|C|}$ be the probability that at least one corrupted party is elected to create *leader* block at a given slot.
3. $X_j^{e_i}$ and $Y_j^{e_i}$ are boolean random variables, where $i \geq 1,\ j \in \{1, ..., R\}$. Let $X_j^{e_i} = 1$ if at least one honest party is elected at the jth slot of epoch e_i, otherwise, $X_j^{e_i} = 0$. Let $Y_j^{e_i} = 1$ if at least one corrupted party is elected at the jth slot of epoch e_i, otherwise, $Y_j^{e_i} = 0$. Further, we have that $\Pr[X_j^{e_i} = 1] = \alpha$ and $\Pr[Y_j^{e_i} = 1] = \beta$.

Remarks. In proof-of-stake based blockchain protocols, even if more than one parties are elected at a slot, the chain can only be extended by one block, which is different from the proof-of-work based blockchain protocols.

5.1 Proofs of Security Properties

In this section, we present the proofs of security properties with respect to *leader* chains and *transaction* chains held by honest parties.

The security analysis of leader chain.

1. Achieving chain growth property.

   Lemma 1. *During protocol execution, for a given slot $sl_j^{e_i}$, suppose that an honest party P_1 holds leader chain $\mathcal{C}_1$ at the beginning of $sl_j^{e_i}$ and $X_j^{e_i} = 1$. Then at the end of $sl_j^{e_i}$, for any honest party P_2 with chain $\mathcal{C}_2$ and with overwhelming probability, it holds that $len(\mathcal{C}_2) - len(\mathcal{C}_1) = 1$.*

   Proof. Consider protocol execution, at any time, a broadcasted block must be received by all the parties within a slot, so that the chains held by honest parties must with the same length at the end of any slot. Further, an elected honest party must create a *leader* block honestly that will be received by all the honest parties at the end of current slot. Note that, when $X_j^{e_i} = 1$, the honest parties may receive several valid blocks and choose different blocks to extend local chains. So that each honest party's local chain is increased by one block with overwhelming property. This completes the proof.

   Lemma 2. *During protocol execution, suppose that an honest party P_1 holds chain $\mathcal{C}_1$ at the beginning of $sl_j^{e_i}$. Then at the beginning of $sl_{j+t}^{e_i}$ $(t > 0)$, for any honest party P_2 with chain $\mathcal{C}_2$. With overwhelming probability, it holds that $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq \sum_j^{j+t-1} X_j^{e_i}$.*

Proof. From lemma 1, we can see that honest parties' local *leader* must be extended by one block when $X_j^{e_i} = 1$, moreover, a broadcasted valid block that created by a corrupted party can also be received and accepted by honest parties. So that $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq \sum_j^{j+t-1} X_j^{e_i}$ for $t > 0$. This completes the proof.

Theorem 1. (*Chain Growth). During protocol execution* $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, *for any two existing honest parties* P_1 *and* P_2 *with local leader chains* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *at the beginning of* $sl_j^{e_i}$ *and* $sl_{j+t}^{e_i}$ *respectively, where* $t > 0$. *Then the probability that* $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq g \cdot t$, *where* $g = (1-\varepsilon)\alpha$, *is at least* $1 - e^{-\Omega(t)}$.

Proof. From the definition of variable $X_j^{e_i}$, we have that $\Pr[X_j^{e_i} = 1] = \alpha$. Let $\omega = \sum_j^{j+t-1} X_j^{e_i}$, then by Chernoff bound, we have $\Pr[\omega < (1-\varepsilon)\alpha \cdot t] < e^{-\Omega(t)}$. From lemma 2, we have $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq \sum_j^{j+t-1} X_j^{e_i}$. Thus,

$$\Pr[len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq \textstyle\sum_j^{j+t-1} X_j^{e_i} \geq (1-\varepsilon)\alpha \cdot t] \geq 1 - e^{-\Omega(t)}$$

Let $g = (1-\varepsilon)\alpha$. This completes the proof.

2. Achieving chain quality property.

Lemma 3. *During protocol execution, any* $l \in \mathbb{N}$ *consecutive blocks of a chain* $\mathcal{C}$ *are created in at least* $\frac{l}{\alpha+\beta}$ *consecutive slots.*

Proof. Gathering all the resources in the network, we have that $\mathcal{C}$ will be extended by one block when $X_j^{e_i} = 1$ or $Y_j^{e_i} = 1$ at a slot. At slot $sl_j^{e_i}$, $\mathcal{C}$ is extended by one block with probability $\alpha + \beta$. Let S be a set of consecutive slots during which the l consecutive blocks are created. So we get that $|S| \geq \frac{l}{\alpha+\beta}$. This completes the proof.

Theorem 2. (*Chain Quality*). *During protocol execution* $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, *for any existing honest party* P *with local chain* $\mathcal{C}$, *for any* $l \in \mathbb{N}$ *consecutive blocks of* $\mathcal{C}$. *Then the probability that the ratio of blocks created by adversary is at most* $\mu = \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{1}{1+\varphi}$ *is at least* $1 - e^{-\Omega(l)}$.

Proof. From lemma 3, we have that these l consecutive blocks are created in at least $\frac{l}{\alpha+\beta}$ consecutive slots. Further, we let $X(S)$ and $Y(S)$ be the number of blocks that are created during S by honest and corrupted parties respectively. By Chernoff bound, with overwhelming probability, we have that $Y(S) \leq (1+\varepsilon)\beta|S|$ and $X(S) \geq (1-\varepsilon)\alpha|S|$. Then, we get the following inequality:

$$\frac{Y(S)}{l} \leq \frac{Y(S)}{X(S)} \leq \frac{(1+\varepsilon)\beta|S|}{(1-\varepsilon)\alpha|S|} = \frac{(1+\varepsilon)\beta}{(1-\varepsilon)\alpha} = \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{p|C|}{p|H|} \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{1}{1+\varphi}$$

Where the second equality follows from the fact that $|E| \cdot p << 1$ and the last inequality follows from $\frac{|H|}{|C|} \geq 1 + \varphi$. Let $\mu = \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{1}{1+\varphi}$, this completes the proof.

3. Achieving common prefix property.
   First, we analysis two cases that may cause the honest parties' local *leader* chains diverge for more than K blocks.
   - Case 1: At some slot, the adversary broadcasts a hidden *leader* chain that forks more than K blocks from honest parties' local ones.
   - Case 2: For K consecutive slots, there are more than one parties are elected as leaders at each slot. Note that, the elected parties can be honest or corrupted as long as they create valid *leader* blocks and broadcast them.

   Based on the $BestValid'$ protocol (Fig.6), we know that honest parties cannot choose a chain that forks more than K blocks from local chains. So that Case 1 happens with eligible probability.
   For Case 2, we let A_1 denotes the event that more than one parties are elected as leaders at a given slot and the block created by each of them is valid, A_2 denotes the event that A_1 happens for K consecutive slots. Then we have:

$$\Pr[A_2]=(\Pr[A_1])^K \leq (1-(1-p)^{|E|}-|E|p(1-p)^{|E|-1})^K$$

   Based on the assumption that $p \cdot |E| << 1$, we have that

$$\Pr[A_2] \leq p^2|E|(|E|-1) \approx 0$$

   Theorem 3. *(Common Prefix). During protocol execution $EXEC_{\Pi^{UC}}$, for any two existing honest parties P_1 and P_2 with local leader chains $\mathcal{C}_1$ at slot $sl_{i'}^{e_i}$ and $\mathcal{C}_2$ at slot $sl_{j'}^{e_j}$ (i and j can be same) respectively. Let $sl_{j'}^{e_j} \geq sl_{i'}^{e_i}$, then the probability that $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$ is at least $1-e^{-\Omega(K)}$.*

   Proof. Based on the analysis of the two cases above, with overwhelming probability, we have that at any slot, the *leader* chains held by honest parties cannot diverge for more than K blocks. What's more, in our slot-synchronous network, the chains held by honest parties with the same length at the end of some slot. Assume that P_2 holds chain $\mathcal{C}_3$ at $sl_{i'}^{e_i}$, then we have $\mathcal{C}_1^{\lceil K} = \mathcal{C}_3^{\lceil K}$. Further, we have that $\mathcal{C}_3^{\lceil K} \preceq \mathcal{C}_2$ (that follows from $BestValid'$). So the probability that $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$ is at least $1-e^{-\Omega(K)}$. This completes the proof.

4. Achieving soundness property
   In the proof-of-work based blockchain protocol, a new party can take the longest chain among a set of chains received from the network as the best local chain. However, as we have discussed, the longest chain strategy is useless and a new party can be mislead by adversary easily in the proof-of-stake based blockchain protocol. We say that a new party is initialized securely, if he holds a *leader* chain $\mathcal{C}_1$ that satisfies $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil K} \preceq \mathcal{C}_1$, where $\mathcal{C}_2$ is *leader* chain held by an existing honest party. The main challenge for new parties joining securely is getting the correct version of common part held by honest parties and that is also the main idea of our $BestValid''$ protocol. We stress that the *transaction* chain $\tilde{\mathcal{C}}_1$ held by a new

party satisfies $\tilde{\mathcal{C}}_1 = \tilde{\mathcal{C}}_2$, where $\tilde{\mathcal{C}}_2$ is a *transaction* chain held by an existing honest party and that can be guaranteed by the *uniqueness* property of *transaction* chains held by honest parties.

Theorem 4. *(Soundness). During protocol execution $EXEC_{\Pi^{\cup\mathcal{C}},\mathcal{A},\mathcal{Z}}$, for a new party P_1 who is initialized with leader chain $\mathcal{C}_1$ and an existing honest party P_2 with local leader chain $\mathcal{C}_2$ at sl. Then the probability that $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil K} \preceq \mathcal{C}_1$ is at least $1 - e^{-\Omega(K)}$.*

Proof. After executing the $BestValid''$ protocol, suppose that a new party P_1 obtains a *leader* chain $\mathcal{C}_1$ and an existing honest party P_2 with local *leader* chain $\mathcal{C}_2$ at sl. Then with overwhelming probability, it holds that $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil K} \preceq \mathcal{C}_1$.
Consider a contradiction that $\mathcal{C}_1^{\lceil K} \npreceq \mathcal{C}_2$ (or $\mathcal{C}_2^{\lceil K} \npreceq \mathcal{C}_1$), we assume that B is the last common block of $\mathcal{C}_1$ and $\mathcal{C}_2$, then B must be at least K deeps in $\mathcal{C}_1$ and $\mathcal{C}_2$. Further, we assume that $\mathcal{C'}_1$ is the latest l blocks of $\mathcal{C}_1$ ($l > K$), so that $\mathcal{C'}_1^{\lceil K} \npreceq \mathcal{C}_2$. However, based on the execution of protocol $BestValid''$, we have that $\mathcal{C'}_1$ must not fork more than K blocks with the chains in set $\mathbb{C'}_{j,11}^{e_i}$, which consists of all the *leader* chains of the honest elected parties in current epoch. Further, based on the common prefix property of *leader* chains held by honest parties, it holds that $\mathcal{C'}_1^{\lceil K} \preceq \mathcal{C}_2$ and then $\mathcal{C}_1^{\lceil K} \preceq \mathcal{C}_2$. This completes the proof.

.

The security analysis of transaction chain.

1. Achieving chain growth property.

   Lemma 4. *During protocol execution, suppose that a leader block B is backed by K blocks at the beginning of slot $sl_j^{e_i}$ in a leader chain held by an honest party and created by an honest party P. Then the transaction chain $\tilde{\mathcal{C}}$ held by any honest party must be extended with one transaction block $\tilde{B}$ at the end of $sl_j^{e_i}$.*

   Proof. From protocol execution, we believe that an elected honest party must create and broadcast a block honestly, so that the newly-created block can be received by all the honest parties within a slot. Based on the *common prefix* property of *leader* chains held by honest parties (theorem 3), at the beginning of $sl_j^{e_i}$, B has been in the common part of honest parties' local chains and its creator P is eligible to create a *transaction* block $\tilde{B}$. So that at the end of $sl_j^{e_i}$, $\tilde{B}$ must be received by all the other honest parties in that P is an honest party. This completes the proof.

   Theorem 5. *During protocol execution $EXEC_{\Pi^{\cup\mathcal{C}},\mathcal{A},\mathcal{Z}}$, for any two existing honest parties P_1 and P_2 with local transaction chains as $\tilde{\mathcal{C}}_1$ and $\tilde{\mathcal{C}}_2$ at the beginning of slot $sl_j^{e_i}$ and $sl_{j+t}^{e_i}$ respectively, where $t > 0$. Then the probability that $len(\tilde{\mathcal{C}}_2) - len(\tilde{\mathcal{C}}_1) \geq \tilde{g} \cdot t$, where $\tilde{g} = (1-\varepsilon)\alpha \cdot \frac{1+\varphi}{2+\varphi}$ is at least $1 - e^{-\Omega(t)}$.*

Proof. Note that we consider the *chain growth* property of *transaction* chain when the length of the corresponding *leader* chain is at least K.
Gathering all the resources in the network, at a given slot $sl_j^{e_i}$, the expected number of elected parties is $p|E|$. At any slot, Let A_1 denotes the event that the confirmed *leader* block is created by an honest party, A_2 denotes the event that the *transaction* chain will grow with one block as the corresponding *leader* chain grows with one block. We have

$$\Pr[A_1] \geq \frac{p|H|}{p|E|} \geq \frac{1+\varphi}{2+\varphi}$$

Following from the result of lemma 4, with overwhelming probability, we have that $\Pr[A_2] \geq \alpha \cdot \Pr[A_1]$. Further, we have

$$\Pr[len(\tilde{\mathcal{C}}_2) - len(\tilde{\mathcal{C}}_1) \geq (1-\varepsilon)\alpha \cdot \frac{1+\varphi}{2+\varphi} \cdot t] \geq 1 - e^{-\Omega(t)},$$

where $\tilde{g} = (1-\varepsilon)\alpha \cdot \frac{1+\varphi}{2+\varphi}$. This completes the proof.

2. Achieving chain quality property.

Lemma 5. *During protocol execution, an honest party with local transaction chain $\tilde{\mathcal{C}}$, then the probability that a block $\tilde{B} \in \tilde{\mathcal{C}}$ is created by the adversary is at most $\frac{|C|}{|E|}$.*

Proof. From protocol execution, we know that $\tilde{B}$ is created by the adversary if and only if the corresponding *leader* block B is created by him. Based on the analysis of theorem 5, we get that the probability that B is created by the adversary is at most $\frac{|C|}{|E|}$. This completes the proof.

Theorem 6. *During protocol execution $EXEC_{\Pi^{\cup C}}$, for an existing honest party P with local transaction chain $\tilde{\mathcal{C}}$, for any $l \in \mathbb{N}$ consecutive blocks of $\tilde{\mathcal{C}}$. Then with probability at least $1 - e^{-\Omega(l)}$, it holds that the ratio of blocks created by adversary is at most $\tilde{\mu} = \frac{1}{1+\varphi}$.*

Proof. Based on the result of lemma 3, these l consecutive *transaction* blocks are created in at least $\tilde{S} = \frac{l}{\alpha+\beta}$ consecutive slots. Following from the result of lemma 5, at any slot, the probability that a *leader* block created by adversary is confirmed by honest parties is at most $\frac{|C|}{|E|}$. What's more, the *transaction* chain will be extended with one block when a *leader* block is confirmed by honest parties. So we have that the number of *transaction* blocks created by adversary in the l consecutive blocks is at most $\frac{|C|}{|E|} \cdot \frac{l}{\alpha+\beta}$.
Let $\tilde{X}(\tilde{S})$ and $\tilde{Y}(\tilde{S})$ denote the number of *transaction* blocks created by honest paries and adversary during $\tilde{S}$ respectively, then we have

$$\frac{\tilde{Y}(\tilde{S})}{l} \leq \frac{\tilde{Y}(\tilde{S})}{\tilde{X}(\tilde{S})} \leq \frac{\frac{|C|}{|E|} \cdot \frac{l}{\alpha+\beta}}{\frac{|H|}{|E|} \cdot \frac{l}{\alpha+\beta}} = \frac{|C|}{|H|} \leq \frac{1}{1+\varphi}$$

Let $\tilde{\mu} = \frac{1}{1+\varphi}$, this completes the proof.

3. Achieving uniqueness property.

Lemma 6. *During protocol execution $EXEC_{\Pi^{UC},\mathcal{A},\mathcal{Z}}$, for a given slot $sl_j^{e_i}$, there are at most one valid transaction block in the network. (Based on $BestValid'$, we do not consider the condition that adversary may create and broadcast more than one valid transaction blocks at the same time)*

Proof. Consider a contradiction that there are two different valid *transaction* blocks $\tilde{B}_1$ and $\tilde{B}_2$ created by P_1 and P_2 respectively at $sl_j^{e_i}$. Let B_1 and B_2 are the corresponding *leader* blocks that are linked by $\tilde{B}_1$ and $\tilde{B}_2$ respectively. Based on protocol execution, B_1 and B_2 must be backed by K blocks and held by two different honest parties P_1 with local *leader* chains $\mathcal{C}_1$ and P_2 with local *leader* chains $\mathcal{C}_2$ at $sl_j^{e_i}$. Further, we get that that $\mathcal{C}_1^{\lceil K} \not\preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil K} \not\preceq \mathcal{C}_1$, which contradicts the *common prefix* property of *leader* chains held by honest parties (theorem 3). This completes the proof.

Theorem 7. *During protocol execution $EXEC_{\Pi^{UC}}$, at the end of any slot $sl_j^{e_i}$, for any two existing honest parties P_1 and P_2 with local transaction chains $\tilde{\mathcal{C}}_1$ and $\tilde{\mathcal{C}}_2$ respectively. Then with overwhelming probability, it holds that $\tilde{\mathcal{C}}_1 = \tilde{\mathcal{C}}_2$.*

Proof. Consider a contradiction that $\tilde{\mathcal{C}}_1 \neq \tilde{\mathcal{C}}_2$, which means that $len(\tilde{\mathcal{C}}_1) \neq len(\tilde{\mathcal{C}}_2)$ or there are some blocks are different in these two chains.
For condition 1, we assume that $len(\tilde{\mathcal{C}}_1) < len(\tilde{\mathcal{C}}_2)$. Without loss of generality, let $len(\tilde{\mathcal{C}}_2) - len(\tilde{\mathcal{C}}_1) = 1$. Based on lemma 6, it must be the case that the last block $\tilde{B}_{len(\tilde{\mathcal{C}}_2)}$ of $\tilde{\mathcal{C}}_2$ is not accepted or received by P_1 at the end of $sl_j^{e_i}$. Further, we can see that $\tilde{B}_{len(\tilde{\mathcal{C}}_2)}$ must be valid and created at the beginning of $sl_j^{e_i}$ or before in that it has been accepted by an honest party P_2. What's more, based on the slot-synchronous network assumption, $\tilde{B}_{len(\tilde{\mathcal{C}}_2)}$ must have been received by all the honest parties at the end of $sl_j^{e_i}$. As a result, if $\tilde{B}_{len(\tilde{\mathcal{C}}_2)}$ has been confirmed by an honest party, then with overwhelming probability that it must be confirmed by all the honest parties. So that condition 1 happens with negligible probability.
For condition 2, we assume the last blocks of $\tilde{\mathcal{C}}_1$ and $\tilde{\mathcal{C}}_2$ are different, which are denoted as $\tilde{B}_1$ and $\tilde{B}_2$. What's more, Based on protocol execution, these two blocks must be valid and received by all the honest parties at the end of $sl_j^{e_i}$, which contradicts to the result of lemma 6.
This completes the proof.

4. Achieving high efficiency

Theorem 8. *During protocol execution $EXEC_{\Pi^{UC}}$, at slot sl, once a transaction block $\tilde{B}$ is confirmed by an honest party P, then with overwhelming probability, $\tilde{B}$ must be confirmed finally by all the honest parties at the end of sl.*

Proof. Based on the *uniqueness* property of *transaction* chains held by honest parties (theorem 7), at the end of each slot, all the honest parties hold

a same view of local *transaction* chain. So with overwhelming probability, if $\tilde{B}$ is confirmed by an honest party, then it will be confirmed by all honest parties within a slot. This completes the proof.

5.2 Further Discussions

Compared with previous related works [20, 9, 1, 15, 10, 3, 18, 13], "$UniqueChain$" solves the challenges of designing proof-of-stake based blockchain protocols better (section 1.1). Precisely, (1). new parties can join protocol execution at any time and be initialized securely without any additional trusted assumptions. With honest majority assumption, we let the elected parties of the current epoch provide new parties with their local chains and no party can provide more than one chains, so the majority of these chains are provided by honest parties. Further, we propose a best chain selection rule $BestValid''$ for new parties to select a correct local chain from the received chains. Note that the set of parties who provide new parties with local chains are not fixed and predicted, and they are self organized. What's more, we also consider the fairness of new parties that a new party can be initialized securely if and only if he has broadcasted a valid request (requesting transaction). (2). We have identified that the essential cause of low efficiency for handling messages is that honest parties hold different views of the last several blocks of local chains, so it always takes a long time to uniform the honest parties' views. In this protocol, based on the *common prefix* property of *leader* chains held by honest parties, we propose a new way to link the two chains. Formally, we use the uniqueness of the confirmed *leader* blocks (the common part of *leader* chains held by honest parties) to select an unique party to handle messages, which guarantees the uniqueness of valid block with messages in the network at any time. As a result, once a block with messages is added by an honest party to his local chain, then due to the latency of network, it must be confirmed by all the honest parties within a slot.

However, an elected honest party of the current epoch is responsible to help the new parties to be initialized securely and might be eligible to create a *transaction* at some point of future, so that our protocol cannot support a fully adaptive adversary as in [9, 1]. We will continue to optimize our protocol to achieve the better security properties.

6 Conclusion

In this work, we propose a fast and provably secure proof-of-stake based blockchain protocol Π^{UC} in open setting, which executes in a slot-synchronous network and with a mildly adaptive adversary. Based on the honest majority assumption, except for three fundamental security properties of blockchain protocols as chain growth, chain quality and common prefix, our protocol also achieves soundness of chains held by newly joining parties without any additional trusted assumptions and uniqueness of chains that consist of messages held by honest parties, which guarantees the high efficiency of handling messages.

References

1. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. computer and communications security **2018**, 913–930 (2018)
2. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. Computer Science pp. 142–157 (2014)
3. Bentov, I., Pass, R., Shi, E.: Snow white: Provably secure proofs of stake. IACR Cryptology ePrint Archive **2016**, 919 (2016)
4. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: IEEE Symposium on Foundations of Computer Science. p. 136 (2001)
5. Canetti, R.: Universally composable signature, certification, and authentication. In: IEEE Workshop on Computer Security Foundations (2004)
6. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology **13**(1), 143–202 (2000)
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000), `https://eprint.iacr.org/2000/067`
8. Chaum, D.: Blind Signatures for Untraceable Payments. Springer US (1983)
9. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain pp. 66–98 (2018)
10. Duong, T., Fan, L., Veale, T., Zhou, H.: Securing bitcoin-like backbone protocols against a malicious majority of computing power. IACR Cryptology ePrint Archive **2016**, 716 (2016)
11. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: International Cryptology Conference on Advances in Cryptology. pp. 139–147 (1993)
12. Eyal, I.: The miner's dilemma. Computer Science pp. 89–103 (2014)
13. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-ng: a scalable blockchain protocol. networked systems design and implementation pp. 45–59 (2016)
14. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International Conference on Financial Cryptography & Data Security (2014)
15. Fan, L., Zhou, H.: iching: A scalable proof-of-stake blockchain in the open setting (or, how to mimic nakamoto's design via proof-of-stake). IACR Cryptology ePrint Archive **2017**, 656 (2017)
16. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications **9057**, 281–310 (2015)
17. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty pp. 291–323 (2017)
18. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. symposium on operating systems principles **2017**, 51–68 (2017)
19. Hofheinz, D., Mllerquade, J.: Universally composable commitments using random oracles. In: Theory of Cryptography Conference (2004)
20. Kiayias, A., Konstantinou, I., Russell, A., David, B., Oliynykov, R.: A provably secure proof-of-stake blockchain protocol. IACR Cryptology ePrint Archive **2016**, 889 (2016)
21. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols (2015)

22. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: International Cryptology Conference (2017)
23. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted (2008)
24. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 643–673 (2017)
25. Pass, R., Shi, E.: The sleepy model of consensus. In: International Conference on the Theory & Application of Cryptology & Information Security (2017)
26. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release Crypto. Massachusetts Institute of Technology (1996)
27. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin (2015)
28. Schrijvers, O., Bonneau, J., Dan, B., Roughgarden, T.: Incentive compatibility of bitcoin mining pool reward functions (2016)

A The Implementation of $\mathcal{F}_{init}$

We denote φ_{init} as the ideal protocol of $\mathcal{F}_{init}$, where the parties are dummy that they only forward messages sent by environment $\mathcal{Z}$ to $\mathcal{F}_{init}$ and then forward the messages sent by $\mathcal{F}_{init}$ to environment $\mathcal{Z}$. Further, we denote Π_{init} as the protocol that implements φ_{init} securely. Informally, the genesis epoch consists of two slots $e_0 = \{sl_1^{e_0}, sl_2^{e_0}\}$, each party $P_{i'}$ first commits to his local stake $s_{i'}$ ($s_{i'} = 1$ in our protocol) and a random value $r_{i'} \in \{0,1\}^k$, and broadcasts the commitment $C_{i'}$ via $\mathcal{F}_{NET}$. Then, $P_{i'}$ collects all the received commitments and opens his commitment to others via $\mathcal{F}_{NET}$. Finally, they determine the difficult target T^{e_1} and random value $nonce^{e_1}$ for epoch e_1. During protocol execution, the new parties obtain these messages from the genesis block B_0. Π_{init} is formally described in Fig.8.

Let $EXEC^{\mathcal{F}_{init}}_{\varphi_{init},\mathcal{S},\mathcal{Z}}$ be the random variable that denotes the joint outputs of all the parties by executing protocol φ_{init} with adversary $\mathcal{S}$ and environment $\mathcal{Z}$. Let $EXEC^{\mathcal{F}_{NET}}_{\Pi_{init},\mathcal{A},\mathcal{Z}}$ be the random variable that denotes the joint outputs of all the parties by executing protocol Π_{init} with adversary $\mathcal{A}$ and environment $\mathcal{Z}$. We have the following lemma.

Lemma 7. *Consider the ideal protocol described above and the real protocol Π_{init} (Fig.8), it holds that these two random variables $EXEC^{\mathcal{F}_{NET}}_{\Pi_{init},\mathcal{A},\mathcal{Z}}$ and $EXEC^{\mathcal{F}_{init}}_{\varphi_{init},\mathcal{S},\mathcal{Z}}$ are indistinguishable.*

Proof. Consider the adversary $\mathcal{A}$ for Π_{init}, we construct the adversary $\mathcal{S}$ for φ_{init} with security parameter k. Note that $\mathcal{S}$ maintains a local table $\mathcal{T}$.

Upon receiving $(Initialize, P_{i'}, s_{i'})$ from $\mathcal{A}$, if it has a record $B_0 \in \mathcal{T}$, then send $(Initialized, B_0)$ to $\mathcal{A}$. Otherwise, pass the message to the functionality $\mathcal{F}_{init}$ and receive $(Initialized, B_0)$ from $\mathcal{F}_{init}$, then record B_0 and send $(Initialized, B_0)$ to $\mathcal{A}$.

Now, we can see that for each query from $\mathcal{A}$, the form of output is $(Initialized, B_0)$, where $B_0 = (T^{e_1}, nonce^{e_1})$, T^{e_1} is determined by the distribution of initial parties' stakes and $nonce^{e_1}$ is sampled uniformly from $\{0,1\}^k$. Therefore, $EXEC^{\mathcal{F}_{NET}}_{\Pi_{init},\mathcal{A},\mathcal{Z}}$ and $EXEC^{\mathcal{F}_{init}}_{\varphi_{init},\mathcal{S},\mathcal{Z}}$ are indistinguishable.

Protocol Π_{init}

Π_{init} is parameterized by security parameter k, interacts with parties $P_{i'}$ ($i' \in \{1, 2, ...\}$), adversary $\mathcal{A}$ and environment $\mathcal{Z}$. For each $P_{i'}$, it proceeds as follows.
Upon receiving $(Initialize, P_{i'}, s_{i'})$ from $\mathcal{Z}$, $P_{i'}$ gets the current slot $sl_j^{e_i} (i \in \mathbb{N}, j \in \{1, ..., R\})$ from the local clock.

- If $i = 0$ and $j = 1$, then chooses random values $r'_{i'}, r_{i'} \in \{0, 1\}^k$, computes commitment as $C_{i'} := Com(r'_{i'}; s_{i'}, r_{i'})$ and sends $(Broadcast, P_{i'}, C_{i'})$ to $\mathcal{F}_{NET}$.
  - collects all the received commitments $C := \{C_{i'}, i' \in \{1, 2, ...\}\}$, opens commitment as $(s_{i'}, r_{i'})$ and sends $(Broadcast, P_{i'}, (s_{i'}, r_{i'}))$ to $\mathcal{F}_{NET}$ at $sl_2^{e_i}$.
  - collects all the received openings $O := \{(s_{i'}, r_{i'}), i' \in \{1, 2, ...\}\}$, computes the difficult target T^{e_1} according to the distribution of stakes $\{s_{i'}, i' \in \{1, 2, ...\}\}$ and random value $nonce^{e_1} := \oplus_{i'} r_{i'}$
  - set $B_0 = (T^{e_1}, nonce^{e_1})$
- Else, collects the message received from $\mathcal{F}_{NET}$, extracts the set $\mathbb{C}_j^{e_i}$ of chains and sets B_0 as the first common block of the majority of *leader* chains in $\mathbb{C}_j^{e_i}$

Output $(Initialized, B_0)$ to the environment $\mathcal{Z}$.

Fig. 8. The Initialization Protocol Π_{init}

B The Implementation of $\mathcal{F}_{res}$

As described above, we denote φ_{res} as the ideal protocol of $\mathcal{F}_{res}$, where the parties are dummy that they only forward the messages sent by environment $\mathcal{Z}$ to $\mathcal{F}_{res}$ and then forward the messages sent by $\mathcal{F}_{res}$ to $\mathcal{Z}$. Further, we denote Π_{res} as the protocol that implements φ_{res} securely. Informally, at any slot, first each party determines whether he is the leader by computing a hash function. Moreover, a party also checks if his *leader* block is deep enough in local chain and determines whether he is eligible to crete a *transaction* block. After that, any parties can verify the validity of *leader* blocks and *transaction* blocks. We show Π_{res} in the $\{\mathcal{F}_{RO}, \mathcal{F}_{SIG}\}$-*hybrid* model (Fig.9), where functionalities $\mathcal{F}_{RO}$ and $\mathcal{F}_{SIG}$ have been well defined in [19] and [5] respectively.

Let $EXEC^{\mathcal{F}_{res}}_{\varphi_{res},\mathcal{S},\mathcal{Z}}$ be the random variable that denotes the joint outputs of all the parties by executing protocol φ_{res} with adversary $\mathcal{S}$ and environment $\mathcal{Z}$. Let $EXEC^{\mathcal{F}_{RO},\mathcal{F}_{SIG}}_{\Pi_{res},\mathcal{A},\mathcal{Z}}$ be the random variable that denotes the joint outputs of all the parties by executing protocol Π_{res} with adversary $\mathcal{A}$ and environment $\mathcal{Z}$. We have the following lemma.

Lemma 8. *Consider the ideal protocol describes above and the real protocol Π_{res} in Fig.9, it holds that these two random variables $EXEC^{\mathcal{F}_{res}}_{\varphi_{res},\mathcal{S},\mathcal{Z}}$ and $EXEC^{\mathcal{F}_{RO},\mathcal{F}_{SIG}}_{\Pi_{res},\mathcal{A},\mathcal{Z}}$ are indistinguishable.*

Proof. Consider the adversary $\mathcal{A}$ for Π_{res}, we construct the adversary $\mathcal{S}$ for φ_{res} with security parameter k. Note that $\mathcal{S}$ maintains a local table $\mathcal{T}$. At $sl_j^{e_i}$ $(i \in \mathbb{N}, j \in \{1, ..., R\})$ and it proceeds as follows.

1. Simulating Registration Phase
   Upon receiving $(Register, P_{i'})$ from $\mathcal{A}$, send $(Register, P_{i'})$ to $\mathcal{F}_{res}$ and obtain $(Registered, P_{i'})$, then send $(Registered, P_{i'})$ to $\mathcal{A}$. Upon receiving $(Unregister, P_{i'})$ from $\mathcal{A}$, send $(Unregister, P_{i'})$ to $\mathcal{F}_{res}$ and obtain $(Unregistered, P_{i'})$, then send $(Unregistered, P_{i'})$ to $\mathcal{A}$.
2. Simulating Stake Election Phase
   - Creating Leader Blocks
     - Upon receiving $(L\text{-}Elect, P_{i'})$ from $\mathcal{A}$, if there is a record $(P_{i'}, h)$, then send h to $\mathcal{A}$. Otherwise, send $(L\text{-}Elect, P_{i'})$ to $\mathcal{F}_{res}$ and obtain $(L\text{-}Elected, P_{i'}, f)$. If $f = 1$, choose random value $h \in \{0,1\}^k$ such that $h \leq T^{e_i}$. Otherwise, choose random value $h \in \{0,1\}^k$ such that $h > T^{e_i}$. Then record $(L\text{-}Elected, P_{i'}, h)$ and send h to A.
     - Upon receiving $(Compute, B_{-1}, P_{i'})$ from $\mathcal{A}$, if their is a record (B_{-1}, h_{-1}), then send h_{-1} to $\mathcal{A}$. Otherwise, send $(Compute, B_{-1}, P_{i'})$ to $\mathcal{F}_{res}$ and obtain $(Computed, h_{-1})$, then record (B_{-1}, h_{-1}) and send h_{-1} to $\mathcal{A}$.
     - Upon receiving $(Sign, B, P_{i'})$ from $\mathcal{A}$, if there is a record (B, σ), then send $(Signed, (B, P_{i'}), \sigma)$ to $\mathcal{A}$. Otherwise, send $(Sign, B, P_{i'})$ to $\mathcal{F}_{res}$ and obtain $(Signed, B, \sigma)$, then record (B, σ) and send $(Signed, (B, P_{i'}), \sigma)$ to $\mathcal{A}$.
   - Creating Transaction Blocks
     - Upon receiving $(T\text{-}Elect, P_{j'})$ from $\mathcal{A}$, if there is a record $(P_{j'}, \tilde{f})$, then send $\tilde{f}$ to $\mathcal{A}$. Otherwise, send $(T\text{-}Elect, P_{j'})$ and obtain $(T\text{-}Elected, P_{j'}, \tilde{f})$, then record $(P_{j'}, \tilde{f})$ and send $\tilde{f}$ to $\mathcal{A}$.
     - Upon receiving $(Compute, \tilde{B}_{-1}, B_{-1}, P_{j'})$ from $\mathcal{A}$, if there is a record $(\tilde{B}_{-1}, B_{-1}, \tilde{h}_{-1}, h_{-1})$, then send $(\tilde{h}_{-1}, h_{-1})$ to $\mathcal{A}$. Otherwise, send $(Compute, \tilde{B}_{-1}, B_{-1}, P_{j'})$ to $\mathcal{F}_{res}$ and obtain $(Computed, \tilde{h}_{-1}, h_{-1})$, then record $(\tilde{B}_{-1}, B_{-1}, \tilde{h}_{-1}, h_{-1})$ and send $(\tilde{h}_{-1}, h_{-1})$ to $\mathcal{A}$.
     - Upon receiving $(Sign, \tilde{B}, P_{j'})$ from $\mathcal{A}$, if there is a record $(\tilde{B}, \tilde{\sigma})$, then send $(Signed, (\tilde{B}, P_{j'}), \tilde{\sigma})$ to $\mathcal{A}$. Otherwise, send $(Sign, \tilde{B}, P_{j'})$ to $\mathcal{F}_{res}$ and obtain $(Signed, (\tilde{B}, \sigma))$, then record $(\tilde{B}, \tilde{\sigma})$ and send $(Signed, (\tilde{B}, P_{j'}), \tilde{\sigma})$ to $\mathcal{A}$.
3. Simulating Verification Phase
   - Upon receiving $(Verify, (B, P'_{i'}), \sigma)$ from $\mathcal{A}$, if there is a record $((P'_{i'}, h), (B, \sigma))$, then send $(h, y'_{i'} = 1)$ to $\mathcal{A}$. Otherwise, choose $h \in \{0,1\}^k$ such that $h > T^{e_i}$ and send $(h, y'_i = 0)$ to $\mathcal{A}$ or choose $h \in \{0,1\}^k$ such $h > T^{e_i}$ and send $(h, y'_{i'} = 1)$ to $\mathcal{A}$ or choose $h \in \{0,1\}^k$ such that $h \leq T^{e_i}$ and send $(h, y'_{i'} = 0)$ to $\mathcal{A}$.
   - Upon receiving $(Verify, (\tilde{B}, P'_{i'}), \tilde{\sigma})$ from $\mathcal{A}$, if there is a record $((P'_{i'}, \tilde{f}), (\tilde{B}, \sigma))$, then send $\tilde{y'}_{i'} = \tilde{f}$ to $\mathcal{A}$. Otherwise, send $\tilde{y'}_{i'} = 0$ to $\mathcal{A}$.

Now, from the above simulation, we can see that the environment $\mathcal{Z}$ gets what he should gets in the real protocol execution. Precisely, for each *Elect*, *Compute* and *Sign* query, $\mathcal{F}_{ree}$ chooses uniformly from $\{0,1\}^k$. For each $Verify$ query, $\mathcal{S}$ responds according to the records in $\mathcal{T}$, which also come from $\mathcal{F}_{res}$. In fact, $\mathcal{S}$ just transfers message between $\mathcal{Z}$ and $\mathcal{F}_{res}$. As a result, we have that $EXEC^{\mathcal{F}_{res}}_{\varphi_{res},\mathcal{S},\mathcal{Z}}$ and $EXEC^{\mathcal{F}_{RO},\mathcal{F}_{SIG}}_{\Pi_{res},\mathcal{A},\mathcal{Z}}$ are indistinguishable.

Protocol Π_{res}

Π_{res} is parameterized by probability p, security parameter k, interacts with parties $P_{i'} \in E$ ($i' \in \{1, 2, ...|E|\}$), adversary $\mathcal{A}$ and environment $\mathcal{Z}$. For each $P_{i'}$, at slot $sl_j^{e_i}$ ($i \in \mathbb{N}, j \in \{1, ..., R\}$), it proceeds as follows.

- **Registration.**
  1. Upon receiving $(Register, P_{i'})$ from $\mathcal{Z}$, if $P_{i'}$ has been registered with $re_{i'} = 1$, then ignore the message. Otherwise, set $re_{i'} = 1$, record $(P_{i'}, re_{i'} = 1)$ and send $(Registered, P_{i'})$ to $\mathcal{Z}$.
  2. Upon receiving $(Unregister, P_{i'})$ from environment $\mathcal{Z}$, if $P_{i'}$ has not been registered with $re_{i'} = 1$, then ignore the message. Otherwise, set $re_{i'} = 0$, record $(P_{i'}, re_{i'} = 0)$ and send $(Unregistered, P_{i'})$ to $\mathcal{Z}$.
- **Stake Election.**
  1. **Creating leader Blocks**: every registered party $P_{i'}$ with one unit stake $s_{j,i'}^{e_i} = 1$ proceeds as follows.
     - Upon receiving $(L\text{-}Elect, P_{i'})$ from $\mathcal{Z}$
       - If there is a record $(P_{i'}, re_{i'} = 1, s_{j,i'}^{e_i} = 1)$, then query $\mathcal{F}_{RO}$ with input $(pk_{i'}, sl_j^{e_i})$ and obtain h. If $h \leq T^{e_i}$, send $(L\text{-}Elected, P_{i'}, f = 1)$ to $\mathcal{Z}$, otherwise, send $(L\text{-}Elected, P_{i'}, f = 0)$ to $\mathcal{Z}$.
       - Otherwise, if $re_{i'} = 0$ or $s_{j,i'}^{e_i} = 0$, then send $(L\text{-}Elected, P_{i'}, f = 0)$ to $\mathcal{Z}$.
     - Upon receiving $(Compute, B_{-1}, P_{i'})$ from $\mathcal{Z}$, query $\mathcal{F}_{RO}$ with input (B_{-1}) and obtain h_{-1}, then send $(Computed, h_{-1}, P_{i'})$ to $\mathcal{Z}$.
     - Upon receiving $(Sign, B, P_{i'})$ from $\mathcal{Z}$, send $(Sign, B, P_{i'})$ to $\mathcal{F}_{SIG}$, obtain $(Signed, (B, P_{i'}), \sigma)$, then send $(Signed, (B, P_{i'}), \sigma)$ to $\mathcal{Z}$.
  2. **Creating Transaction Blocks**: if a *leader* block $B_{j'}$ is backed by K' blocks created by party $P_{j'}$, then $P_{j'}$ proceeds as follows.
     - Upon receiving $(T\text{-}Elect, P_{j'})$ from $\mathcal{Z}$, if $K' = K$, then send $(T\text{-}Elected, P_{j'}, \tilde{f} = 1)$ to $\mathcal{Z}$. Otherwise, send $(T\text{-}Elected, P_{j'}, \tilde{f} = 0)$ to $\mathcal{Z}$.
     - Upon receiving $(Compute, \tilde{B}_{-1}, B_{-1}, P_{j'})$ from $\mathcal{Z}$, query $\mathcal{F}_{RO}$ with input $(\tilde{B}_{-1}, B_{-1})$ and obtain $(\tilde{h}_{-1}, h_{-1})$, then send $(Computed, \tilde{h}_{-1}, h_{-1}, P_{j'})$ to $\mathcal{Z}$.
     - Upon receiving $(Sign, \tilde{B}, P_{j'})$ from $\mathcal{Z}$, then send $(Sign, \tilde{B}, P_{j'})$ to $\mathcal{F}_{SIG}$ and obtain $(Signed, (\tilde{B}, P_{j'}), \tilde{\sigma})$, then send $(Signed, (\tilde{B}, P_{j'}), \tilde{\sigma})$ to $\mathcal{Z}$.
- **Verification**: each existing party $P_{i'} \in E$ proceeds as follows. Upon receiving $(Verify, (B, P'_{i'}), \sigma)$ or $(Verify, (\tilde{B}, P'_{i'}), \tilde{\sigma})$ from $\mathcal{Z}$.
  - Send $(pk'_{i'}, sl_j^{e_i})$ to $\mathcal{F}_{RO}$ and obtain h, if $h \leq T^{e_i}$, then set $y_{i'} = 1$, otherwise set $y_{i'} = 0$.
  - Send $(Verify, (B, P'_{i'}, \sigma))$ or $(Verify, (\tilde{B}, P'_{i'}, \tilde{\sigma}))$ to $\mathcal{F}_{SIG}$ and obtain $y'_{i'}$ or $\tilde{y'}_{i'}$
  - If $y_{i'} = 1 \wedge y'_{i'} = 1$ or $\tilde{y'}_{i'} = 1$, then send $(Verified, f' = 1)$ or $(Verified, \tilde{f}' = 1)$ to $\mathcal{Z}$. Otherwise, send $(Verified, f' = 0)$ or $(Verified, \tilde{f}' = 0)$ to $\mathcal{Z}$

Fig. 9. The Resource Protocol Π_{res}