Redactable Blockchain in the Permissionless Setting

无权限设置中的可编辑区块链

2019 IEEE

区块链的不可篡改性

- 区块链的不可篡改性建立在哈希函数与其特殊的链式数据结构之上。
- 不可篡改性使得区块链能够在去中心化的同时解决信任问题。

带来的问题：

无许可的区块链已完全去中心化，并允许任何用户以少量费用将交易发布到链上，因此恶意用户可以将交易发布到包含非法和/或有害数据。

传统的解决方案

基于变色龙散列可修改的的许可区块链设置

G. Ateniese, B. Magri, D. Venturi, and E. Andrade, “Redactable blockchain–or–rewriting history in bitcoin and friends,” in Security and Privacy (EuroS&P), 2017 IEEE European Symposium on. IEEE, 2017, pp. 111–126.

J. Camenisch, D. Derler, S. Krenn, H. C. Po ¨hls, K. Samelin, and D. Slamanig, “Chameleon-hashes with ephemeral trapdoors,” in IACR International Workshop on Public Key Cryptography. Springer, 2017, pp. 152–182.

作者的方案

- 基于共识的投票机制（consensus-based voting）

  if a redaction gathers enough votes the operation is performed on the chain.

- 不依赖于繁琐的加密工具或信任假设
- 提供了公共可验证性和责任性
- 可集成到比特币中

预备知识

区块： $B := < s, x, ctr >, s \in \{0,1\}^k, x \in \{0,1\}^*, ctr \in \mathbb{N}$

有效区块： $\mathrm{Valid}ateBlock^D(B) := H\big(ctr, G(s,x)\big) < D$, $H: \{0,1\}^* \to \{0,1\}^k$, $G: \{0,1\}^* \to \{0,1\}^k, D \in \mathbb{N}$

区块链(C)：区块的序列，区块链中最右的区块称为 $Head(C)$

将区块链C添加新有效区块B'变成C'： $C' := C||B', Head(C') := B'$

$B' :=< s', x', ctr' >, s' = H(ctr, G(s,x))$

空链： $C := \varepsilon$

链长度： $len(C)$ 移除链C中最右的q个区块： $C^{\lceil q}$ 移除链C中最左的q个区块： ${}^{q\rceil}C$

预备知识

一致性（Persistence）

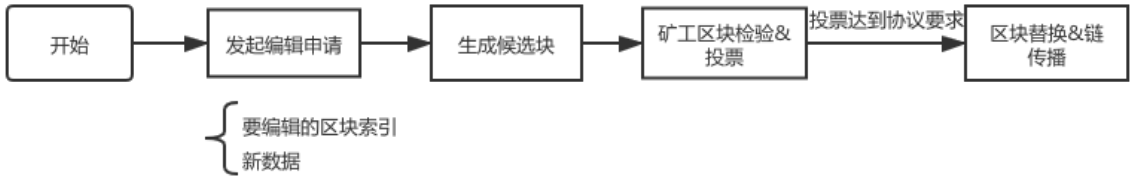
- Once a user in the system announces a particular transaction as stable, all of the remaining users when queried will either report the transaction in the same position in the ledger or will not report any other conflicting transaction as stable.

- A system parameter k determines the number of blocks that stabilise a transaction. That is, a transaction is stable if the block containing it has at least k blocks following it in the blockchain.

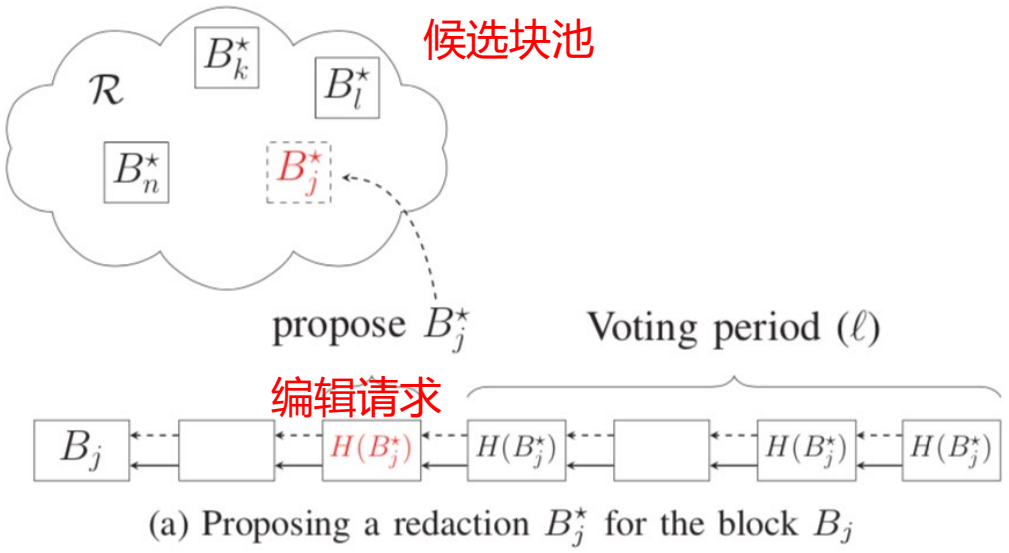
活性（Liveness）

- If all the honest users in the system attempt to include a certain transaction into their ledger, then after the passing of time corresponding to u slots which represents the transaction confirmation time, all users, when queried and responding honestly, will report the transaction as being stable.

核心思想



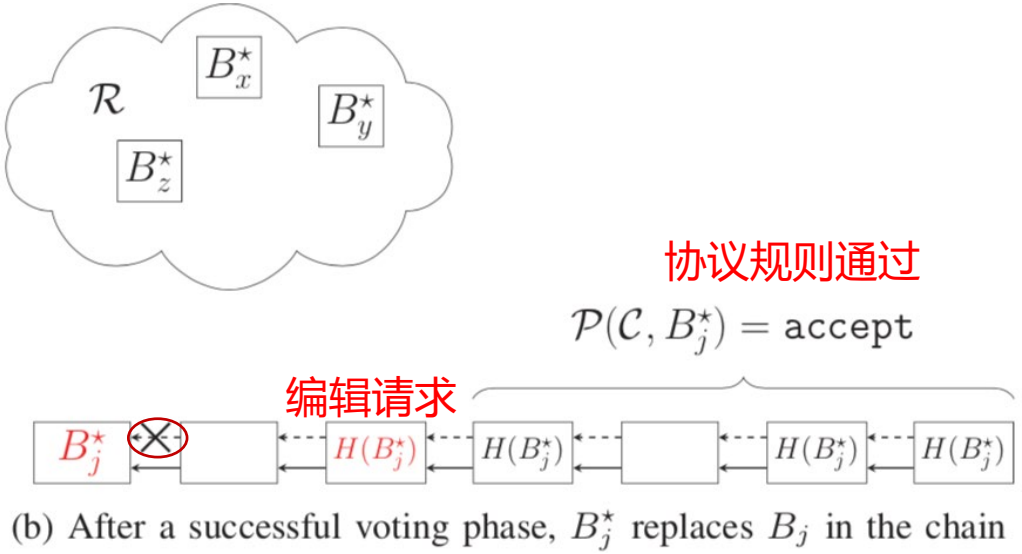
候选块池



(a) Proposing a redaction $B_j^\star$ for the block B_j

协议规则通过

$\mathcal{P}(\mathcal{C}, B_j^\star) = \texttt{accept}$



(b) After a successful voting phase, $B_j^\star$ replaces B_j in the chain

Blockchain Policy

Definition 1 (Policy). *A candidate block* $B^\star$ *generated in round* r *is said to satisfy the policy* $\mathcal{P}$ *of chain* $\mathcal{C} := (B_1, \ldots, B_n)$, *i.e.,* $\mathcal{P}(\mathcal{C}, B^\star) = \texttt{accept}$, *if it holds that* $B_{r+\ell} \in \mathcal{C}^{\lceil k}$ *and the ratio of blocks between* B_r *and* $B_{r+\ell}$ *containing* $H(B^\star)$ *(a vote for* $B^\star$*) is at least* ρ, *for* $k, \ell \in \mathbb{N}$, *and* $0 < \rho \leq 1$, *where* k *is the persistence parameter,* ℓ *is the voting period, and* ρ *is the ratio of votes necessary within the voting period* ℓ.

Protocol Description

数据结构定义： $B :=< s, x, ctr, y >$ $y\epsilon\{0,1\}^k$是区块的旧状态

将区块链C添加新有效区块B'变成C'： $C' := C||B', Head(C') := B'$

$B' :=< s', x', ctr', y' >, s' = H(ctr, G(s, x), y)$

Protocol Description

一个不变的区块链协议的抽象描述：

链的更新：$\{\mathcal{C}', \perp\} \leftarrow \Gamma.\mathsf{updateChain}:$

链的有效性检查：$\{0,1\} \leftarrow \Gamma.\mathsf{validateChain}(\mathcal{C}):$

区块的有效性检查：$\{0,1\} \leftarrow \Gamma.\mathsf{validateBlock}(B):$

广播：$\Gamma.\mathsf{broadcast}(x):$

可编辑的区块链协议Γ'在Γ 的基础上做了如下改动：

- 修改validateChain 和 validateBlock
- 新增两个接口：proposeEdit与validateCand

编辑提议：$B_j^\star \leftarrow \Gamma'.\mathsf{proposeEdit}(\mathcal{C}, j, x^\star):$

候选块有效性检查：$\{0,1\} \leftarrow \Gamma'.\mathsf{validateCand}(B_j^\star, \tilde{\mathcal{C}}):$

Prot

The protocol Γ' consists of a sequence of rounds r, and is parameterised by the liveness and persistence parameters, denoted by u, k, respectively, and by a policy $\mathcal{P}$ that among other rules and constraints, determines the parameter ℓ (that is the duration of the voting period) and ρ (that is the threshold of votes within the period ℓ for a candidate block to be accepted and incorporated into the chain). A pictorial representation of the protocol can be found in Fig. 1.

Initialisation. Set the chain $\mathcal{C} \leftarrow$ genesis, set round $r \leftarrow 1$ and initialise an empty list of candidate blocks for edits $\mathcal{R} := \emptyset$.

For each round r of the protocol, we describe the following sequence of execution.

Chain update. At the beginning of a new round r, the nodes try to update their local chain by calling $\mathcal{C} \leftarrow \Gamma'.\mathsf{updateChain}$.

Candidate blocks pool. Collect all candidate blocks $B_j^\star$ from the network and add $B_j^\star$ to the pool of candidate blocks $\mathcal{R}$ iff $\Gamma'.\mathsf{validateCand}(\mathcal{C}, B_j^\star) = 1$; otherwise discard $B_j^\star$.

Editing the chain. For all candidate blocks $B_j^\star \in \mathcal{R}$ do:

- If $\mathcal{P}(\mathcal{C}, B_j^\star) = \mathtt{accept}$, then build the new chain as $\mathcal{C} \leftarrow \mathcal{C}^{\lceil (n-j+1) \rceil} || B_j^\star || ^{\lfloor j \rfloor} \mathcal{C}$ and remove $B_j^\star$ from $\mathcal{R}$. For policy $\mathcal{P}$ to accept $B_j^\star$, it must be the case that the ratio of votes for $B_j^\star$ within its voting period (ℓ blocks) is at least ρ.
- If $\mathcal{P}(\mathcal{C}, B_j^\star) = \mathtt{reject}$, then remove $B_j^\star$ from $\mathcal{R}$. For policy $\mathcal{P}$ to reject $B_j^\star$ it must be the case that the ratio of votes for $B_j^\star$ within its voting period (ℓ blocks) is less than ρ.
- If $\mathcal{P}(\mathcal{C}, B_j^\star) = \mathtt{voting}$, then do nothing.

Creating a new block. Collects all the transaction data x from the network for the r-th round and tries to build a new block B_r by performing the following steps:

- *(Voting for candidate blocks).* For all candidate blocks $B_j^\star \in \mathcal{R}$ that the node is willing to endorse, if $\mathcal{P}(\mathcal{C}, B_j^\star) = \mathtt{voting}$ then set $x \leftarrow x || H(B_j^\star)$.
- Create a new block $B := \langle s, x, ctr, G(s,x) \rangle$, such that $s = H(ctr', G(s', x'), y')$, for $\langle s', x', ctr', y' \rangle \leftarrow \mathsf{Head}(\mathcal{C})$.
- Extend its local chain $\mathcal{C} \leftarrow \mathcal{C} || B$ and iff $\Gamma'.\mathsf{validateChain}(\mathcal{C}) = 1$ then broadcast $\mathcal{C}$ to the network.

Propose an edit. The node willing to propose an edit for the block B_j, for $j \in [n]$, creates a candidate block $B_j^\star \leftarrow \Gamma'.\mathsf{proposeEdit}(\mathcal{C}, j, x^\star)$ using the new data $x^\star$, and broadcasts it to the network by calling $\Gamma'.\mathsf{broadcast}(B_j^\star)$.

Figure 2: Accountable permissionless editable blockchain protocol $\Gamma'_{\mathcal{P}}$

Protocol Description

Algorithm 1: validateChain (implements Γ'.validateChain)

input : Chain $\mathcal{C} = (B_1, \cdots, B_n)$ of length n.
output: $\{0, 1\}$

1: $j := n$;
2: **if** $j = 1$ **then return** Γ'.validateBlock(B_1);
3: **while** $j \geq 2$ **do**
4: $\quad B_j := \langle s_j, x_j, ctr_j, y_j \rangle$; ▷ $B_j :=$ Head$(\mathcal{C})$ when $j = n$
5: $\quad$ **if** Γ'.validateBlock$(B_j) = 0$ **then return** 0;
6: $\quad$ **if** $s_j = H(ctr_{j-1}, G(s_{j-1}, x_{j-1}), y_{j-1})$ **then** $j := j - 1$;
7: $\quad$ **else if** $(s_j = H(ctr_{j-1}, y_{j-1}, y_{j-1})) \wedge (\Gamma'$.validateCand$(\mathcal{C}, B_{j-1}) = 1) \wedge (\mathcal{P}(\mathcal{C}, B_{j-1}) =$ accept$)$ **then** $j := j - 1$;
8: $\quad$ **else return** 0;
9: **return** 1;

Algorithm 2: validateBlock (implements Γ'.validateBlock)

input : Block $B := \langle s, x, ctr, y \rangle$.
output: $\{0, 1\}$

1: Validate data x, if *invalid* **return** 0;
2: **if** $H(ctr, G(s, x), y) < D \vee H(ctr, y, y) < D$ **then return** 1;
3: **else return** 0;

Protocol Description

Algorithm 3: proposeEdit (implements Γ'.proposeEdit)

input : Chain $\mathcal{C} = (B_1, \cdots, B_n)$ of length n, an index $j \in [n]$, and the new data $x_j^\star$.

output: A candidate block $B_j^\star$.

1: Parse $B_j := \langle s_j, x_j, ctr_j, y_j \rangle$;
2: Build the candidate block $B_j^\star := \langle s_j, x_j^\star, ctr_j, y_j \rangle$;
3: **return** $B_j^\star$;

Algorithm 4: validateCand (implements Γ'.validateCand)

input : Chain $\mathcal{C} = (B_1, \cdots, B_n)$ of length n, and a candidate block $B_j^\star$ for an edit.

output: $\{0, 1\}$

1: Parse $B_j^\star := \langle s_j, x_j^\star, ctr_j, y_j \rangle$;
2: **if** Γ'.validateBlock$(B_j^\star) = 0$ **then return** 0;
3: Parse $B_{j-1} := \langle s_{j-1}, x_{j-1}, ctr_{j-1}, y_{j-1} \rangle$;
4: Parse $B_{j+1} := \langle s_{j+1}, x_{j+1}, ctr_{j+1}, y_{j+1} \rangle$;
5: **if** $s_j^\star = H(ctr_{j-1}, y_{j-1}, y_{j-1}) \wedge s_{j+1} = H(ctr_j, y_j, y_j)$ **then return** 1;
6: **else return** 0;

INTEGRATING INTO BITCOIN

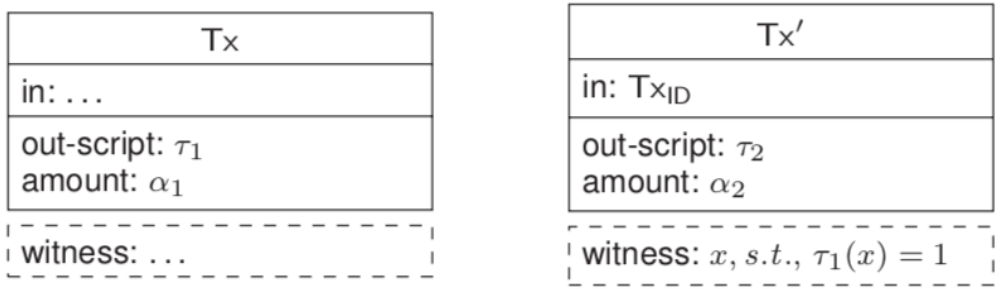


Figure 3: The structure of a transaction in Bitcoin. The transaction Tx′ is spending the output τ_1 of transaction Tx.

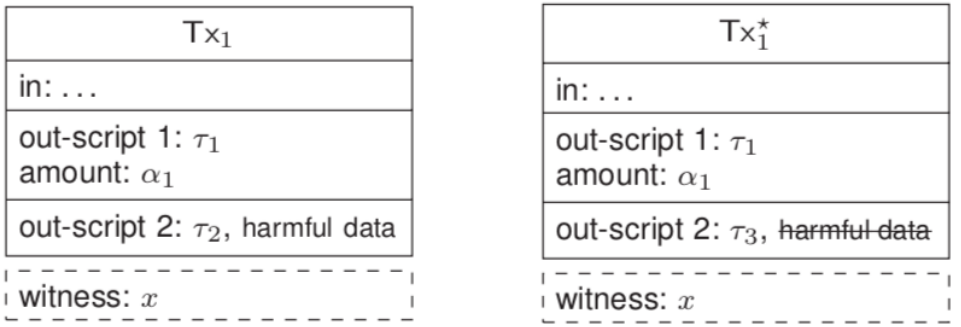
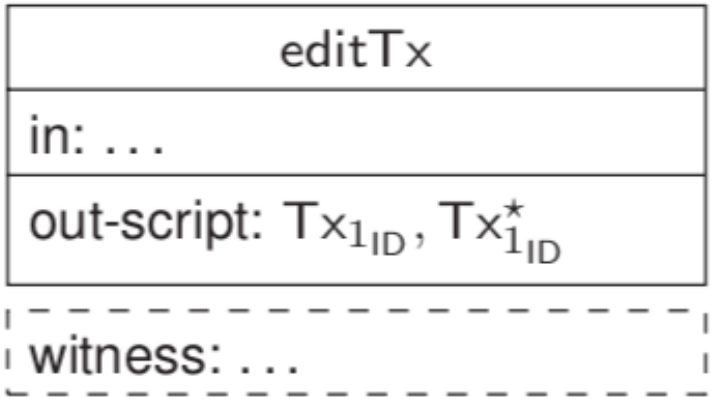


Figure 4: The transaction Tx_1 on the left contains harmful data, and the candidate transaction $\mathsf{Tx}_1^\star$ on the right contains a copy of all the fields of Tx_1, with exception of the harmful data.

INTEGRATING INTO BITCOIN



发起编辑请求流程：

构建交易$editTx$，包含$Tx_{1_{ID}}$与$Tx^{*}_{1_{ID}}$，用以表示被编辑的交易的哈希和候选交易的哈希

广播$editTx$与$Tx^{*}_{1_{ID}}$，将$editTx$添加到区块链中，将$Tx^{*}_{1_{ID}}$添加到候选交易池

检查$Tx^{*}_{1_{ID}}$是否通过有效验证，有效则进入投票环节

INTEGRATING INTO BITCOIN

编辑策略：如果满足以下条件，则提议的修订将被批准为有效

- 除了被移除的有害信息，其他信息需要与原交易完全相同
- 只能删除永远无法花费的交易
- 它没有为链中其他编辑候选交易进行投票
- 在相应的editTx在链中稳定之后，它在1024个连续块（投票周期）中获得了超过50%的选票。

给候选交易Tx_1^*投票：矿工在其新生成区块中把 $\text{editTx}_{\text{ID}} = H(\text{Tx}_{1_{\text{ID}}} || \text{Tx}^{\star}_{1_{\text{ID}}})$ 作为交易添加进去。

投票周期结束后，该候选交易从候选池中移除。

INTEGRATING INTO BITCOIN

区块验证：验证区块内的所有交易

- 未编辑过的交易：与过去验证方法一致
- 编辑过的交易：存储原交易过去的old state,通过old state $Tx_{1_{ID}}$ 去保证与witness的关联关系

Tx_1
in: . . .
out-script 1: τ_1 amount: α_1
out-script 2: τ_2, harmful data
witness: x

$\mathsf{Tx}_1^\star$
in: . . .
out-script 1: τ_1 amount: α_1
out-script 2: τ_3, ~~harmful data~~
witness: x

Figure 4: The transaction Tx_1 on the left contains harmful data, and the candidate transaction $\mathsf{Tx}_1^\star$ on the right contains a copy of all the fields of Tx_1, with exception of the harmful data.

Tx_1
Tx_2
Tx_3
⋮

(a) Non-redacted.

$\mathsf{Tx}_1^\star$, $\mathsf{Tx}_{1_{\mathsf{ID}}}$
Tx_2
Tx_3
⋮

(b) Redacted transaction Tx_1.

Figure 6: List of transactions contained within a block before (left) and after (right) redacting a transaction in the block.

INTEGRATING INTO BITCOIN

区块验证：验证区块是否满足POW验证

Value	Description
`hash_prev`	hash of the previous block header
`merkle_root`	root of the merkle tree (whose the leaves are the transactions)
`difficulty`	the difficulty of the proof-of-work
`timestamp`	the timestamp of the block
`nonce`	nonce used in proof-of-work
`old_merkle_root`	root of the merkle tree of old set of transactions

Figure 5: Structure of the Bitcoin block header. The last highlighted field (`old_merke_root`) is only included in the block header of the extended (editable) protocol.

Algorithm 2: validateBlock (implements Γ'.validateBlock)

input : Block $B := \langle s, x, ctr, y \rangle$.
output: $\{0, 1\}$

1: Validate data x, if *invalid* **return** 0;
2: **if** $H(ctr, G(s, x), y) < D \vee H(ctr, y, y) < D$ **then**
return 1;
3: **else return** 0;

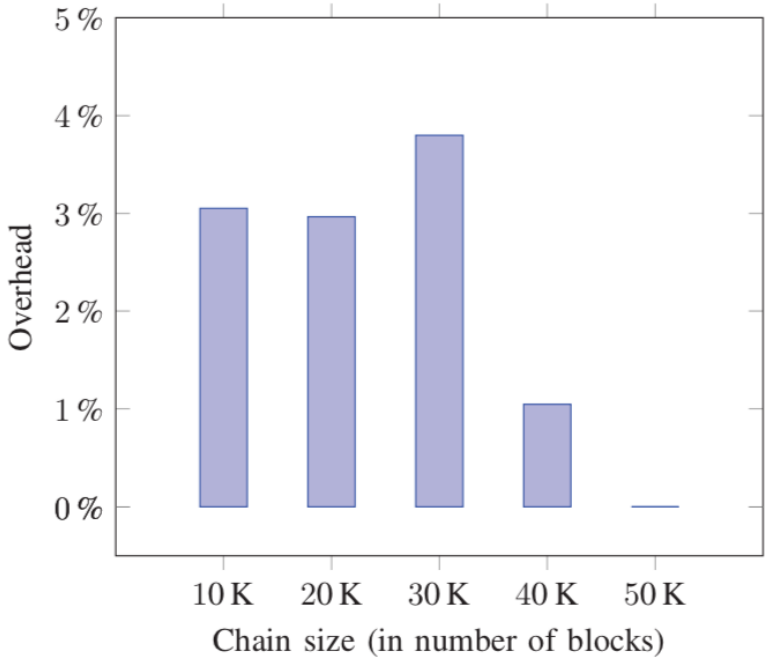
INTEGRATING INTO BITCOIN

链验证：矿工需要验证链中的所有区块。

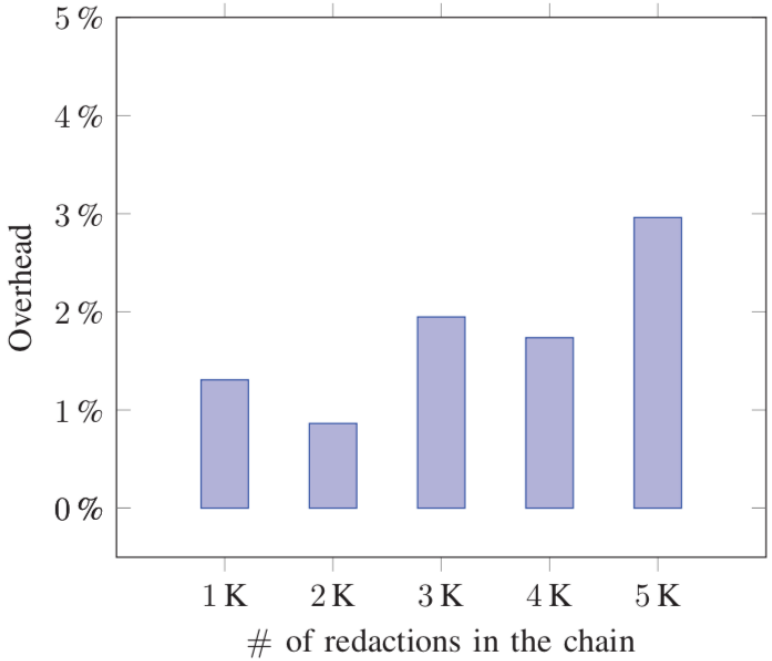
- 矿工可以通过检测与下一个区块验证其哈希链接来判断该区块是否已被编辑；
- 如果是经过编辑的区块，则矿工会根据编辑策略验证该修订是否已获批准。
- 如果满足以下任何条件，则矿工将链视为无效链：
  - （1）根据验证策略未被批准修订的区块，
  - （2）修订后的区块的Merkle根值对于交易集而言是不正确的
  - （3）链上未执行先前批准的编辑。

实验过程

- Intel Xeon Gold 6132 CPU @ 2.60GHz • 128GB of RAM
- Debian Linux 4.9.0-6-amd64
- Python 3.5.3.

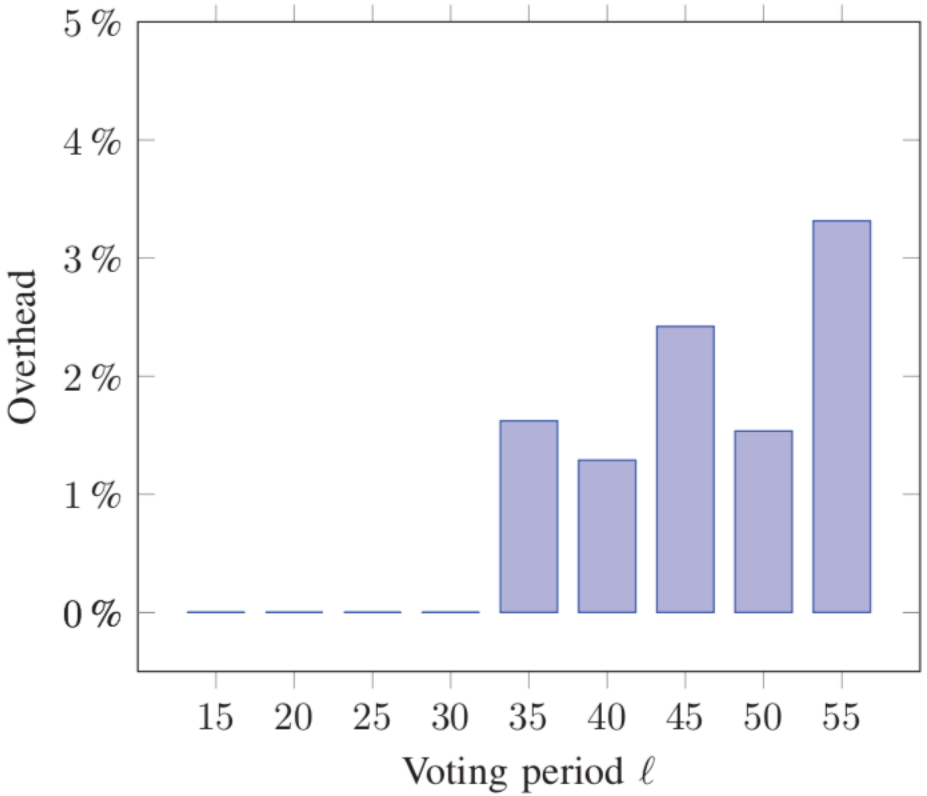


Overhead Compared to Immutable Blockchain.



Overhead by Number of Redactions.

实验过程



Discussion

- Scrutiny of Candidate Blocks. 矿工会仔细检查候选块吗?
- Denial of Service. 恶意编辑请求
- False Victim
- Double Spend Attacks

存在的问题

- 不允许编辑货币交易
- 无法阻止本地账本检索旧数据
- 共识延迟

感谢您的观看

THANK YOU FOR YOUR WATCHING