Efficient Novel Privacy Preserving POS Protocol

Rashad Aliyev
Khalil Abdellah Lemtaffah

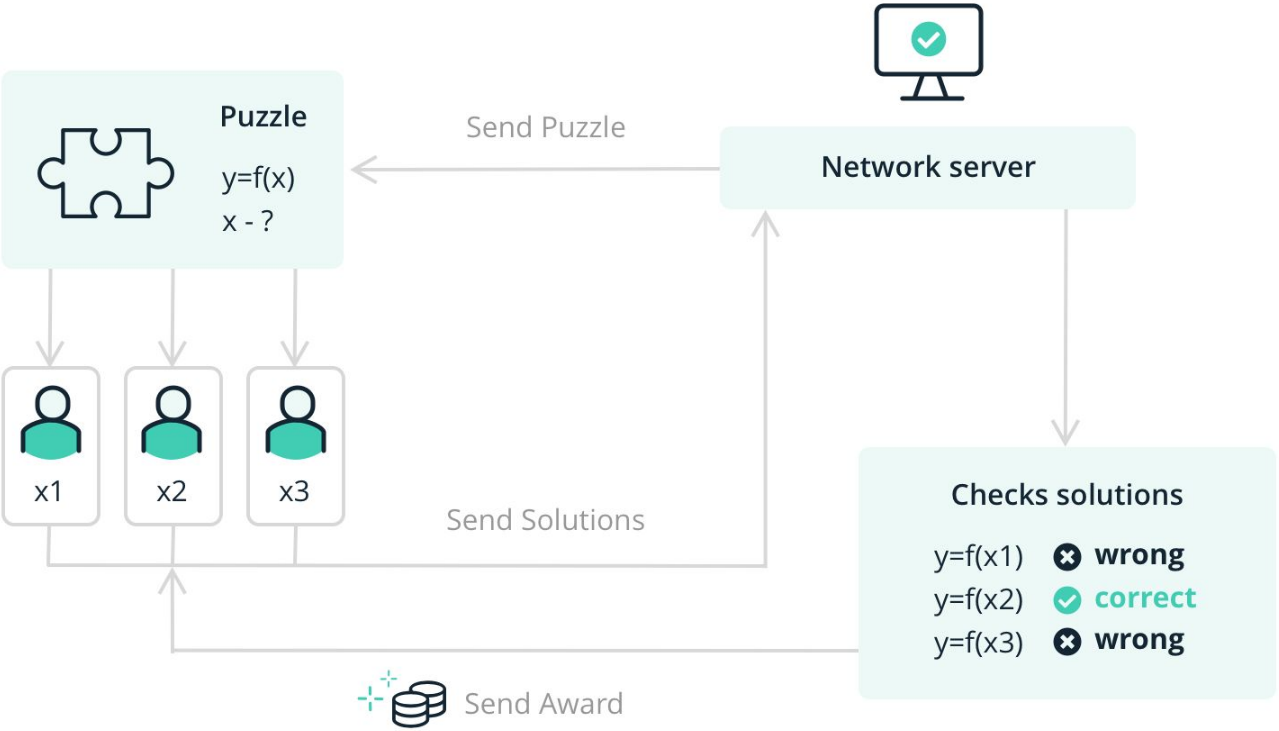
Agenda

- Proof of Work?
- PoW problems
- Proof of Stake?
- Paper overview
- Analysis + Algorand PoC
- Overview of the Scheme
- Conclusion

Proof of Work?

Po

Proc rld
at th



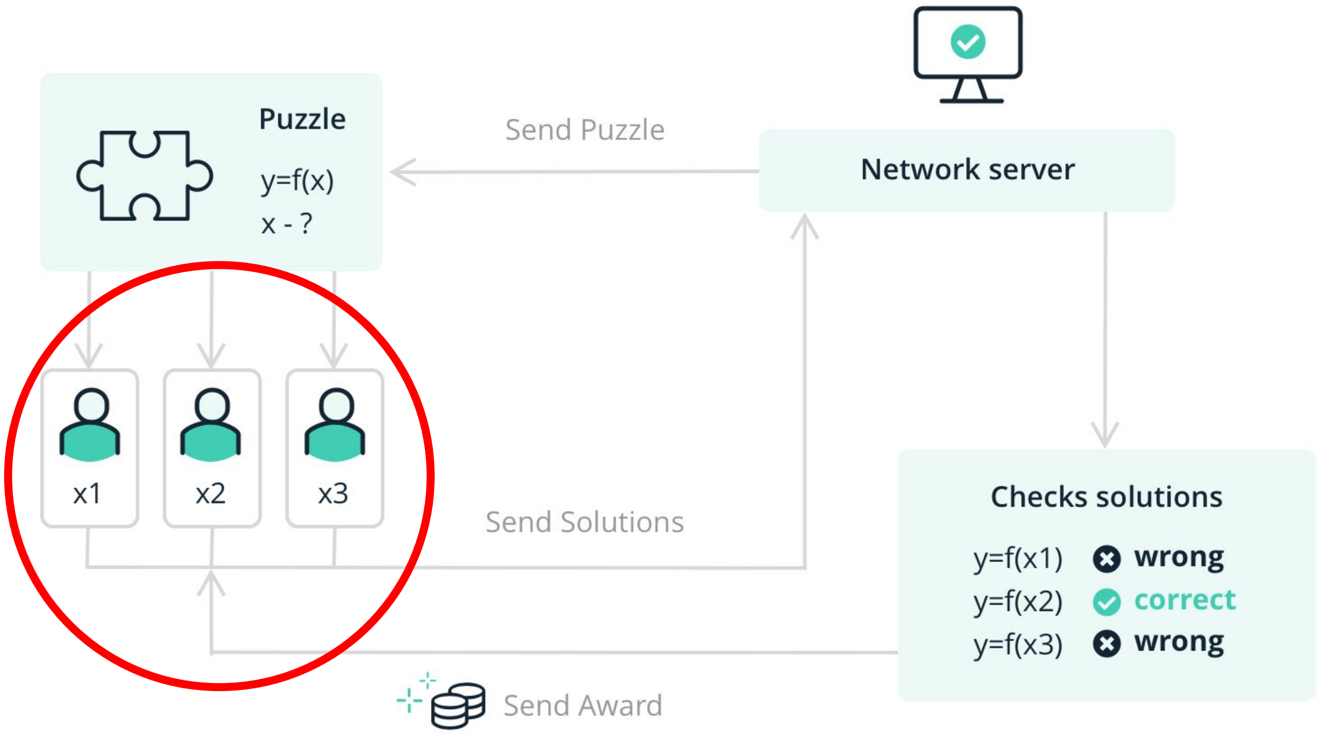
PoW problems

PoW Problems

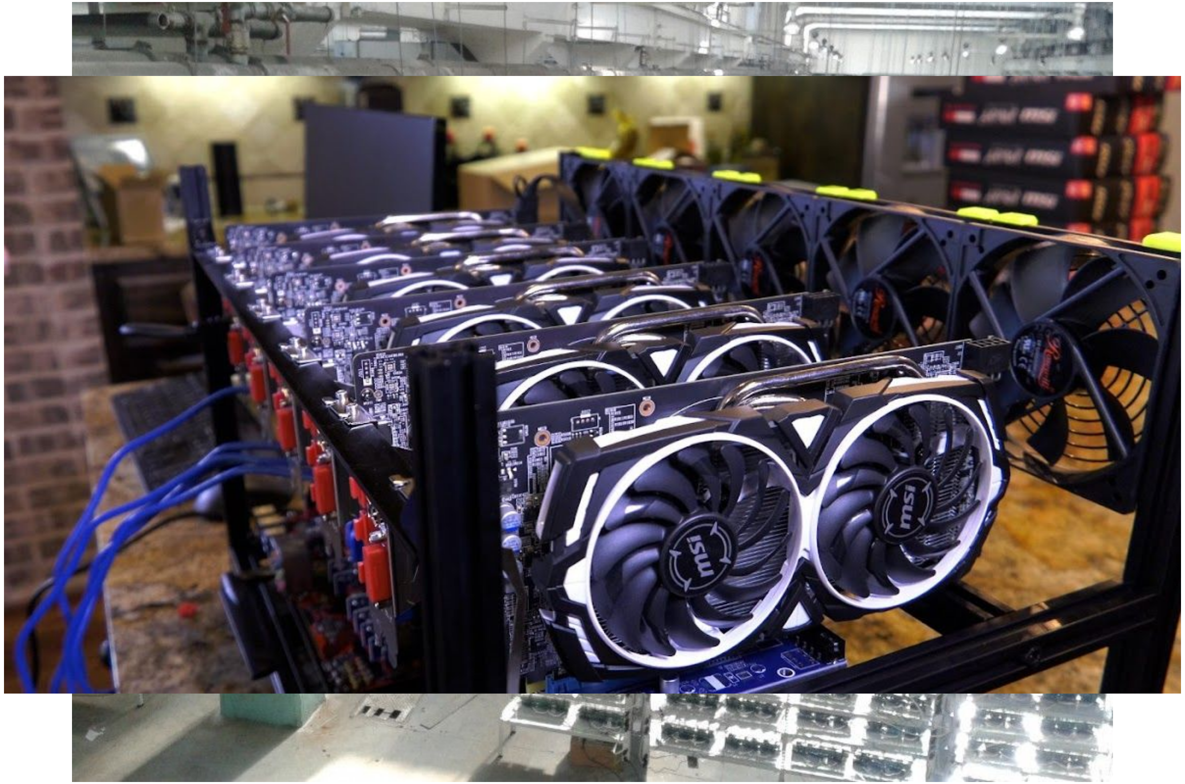
The need of **more** miners

=

more power consumption



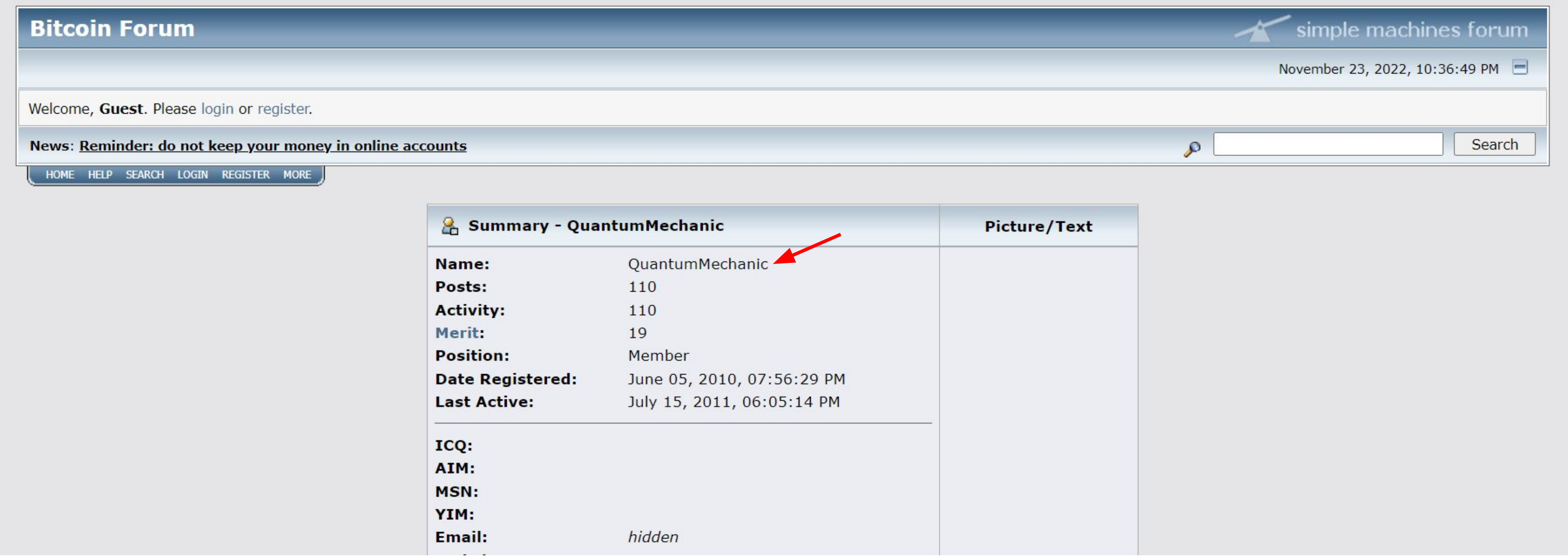
PoW Problems



PoW Problems

We needed a solution for this!!!

PoW Problems



PoW Problems

https://bitcointalk.org/index.php?topic=27787.0

Bitcoin Forum simple machines forum

November 23, 2022, 10:35:57 PM

Welcome, **Guest**. Please login or register.

News: **Reminder: do not keep your money in online accounts** Search

HOME HELP SEARCH LOGIN REGISTER MORE

Bitcoin Forum > Bitcoin > Development & Technical Discussion > **Proof of stake instead of proof of work**

« previous topic next topic »

Pages: [**1**] 2 » All print

Author	Topic: Proof of stake instead of proof of work (Read 32365 times)

QuantumMechanic (OP)
Member

Activity: 110
Merit: 19

Proof of stake instead of proof of work
July 11, 2011, 04:12:45 AM
Merited by ETFbitcoin (3), Vod (2), webtricks (2), d5000 (1), drays (1) #1

I've got an idea, and I'm wondering if it's been discussed/ripped apart here yet:

I'm wondering if as bitcoins become more widely distributed, whether a transition from a proof of work based system to a proof of stake one might happen. What I mean by proof of stake is that instead of your "vote" on the accepted transaction history being weighted by the share of computing resources you bring to the network, it's weighted by the number of bitcoins you can prove you own, using your private keys.

For those that don't want to be actively verifying transactions, and so that not all private keys need to be facing the network, votes could be delegated to other addresses via some kind of nonstandard Bitcoin transaction. In this way, voting power would accumulate with trusted delegates instead of miners. New bitcoins and transaction fees could be randomly and periodically distributed to delgates, weighted by the number of votes they've accumulated, thereby incentivising diversity of the delegates and direct voters.

If the implementation could be done, it proved to maintain at least a similar level of privacy and trustworthiness, and it only minimally complicated the UX, I'm thinking that a proof of stake based fork could out-compete a proof of work one due to much lower transaction fees, since its network wouldn't need to support the cost of the miners' computing resources. (Note that the vote delegation scheme has bandwith/storage overhead that would offset these savings by some amount which would hopefully be relatively small.)

Some other potential improvements this system could offer:

- Possibly quicker, more definite confirmation of transactions, depending on how it can be implemented.
- The "voting power" may be more trustworty, since it would accumulate in a bottom-up fashion via a network of trust, instead of in the somewhat arbitrary way it accumulates now. (Note the potential problem of vote-buying here.)
- It would remove the physical point of failure of bitcoin mining equipment, which can be confiscated or made illegal to run.
- It could be used to provide stakeholders a means of making their voices heard (via the delegated voting system it establishes) when it comes to proposals for software updates and protocol changes.

Anyway, I just wanted to throw the idea out here to see if there are any obvious reasons why it couldn't be implemented, and to hopefully spark a discussion amongst those better qualified than me.

Cheers.

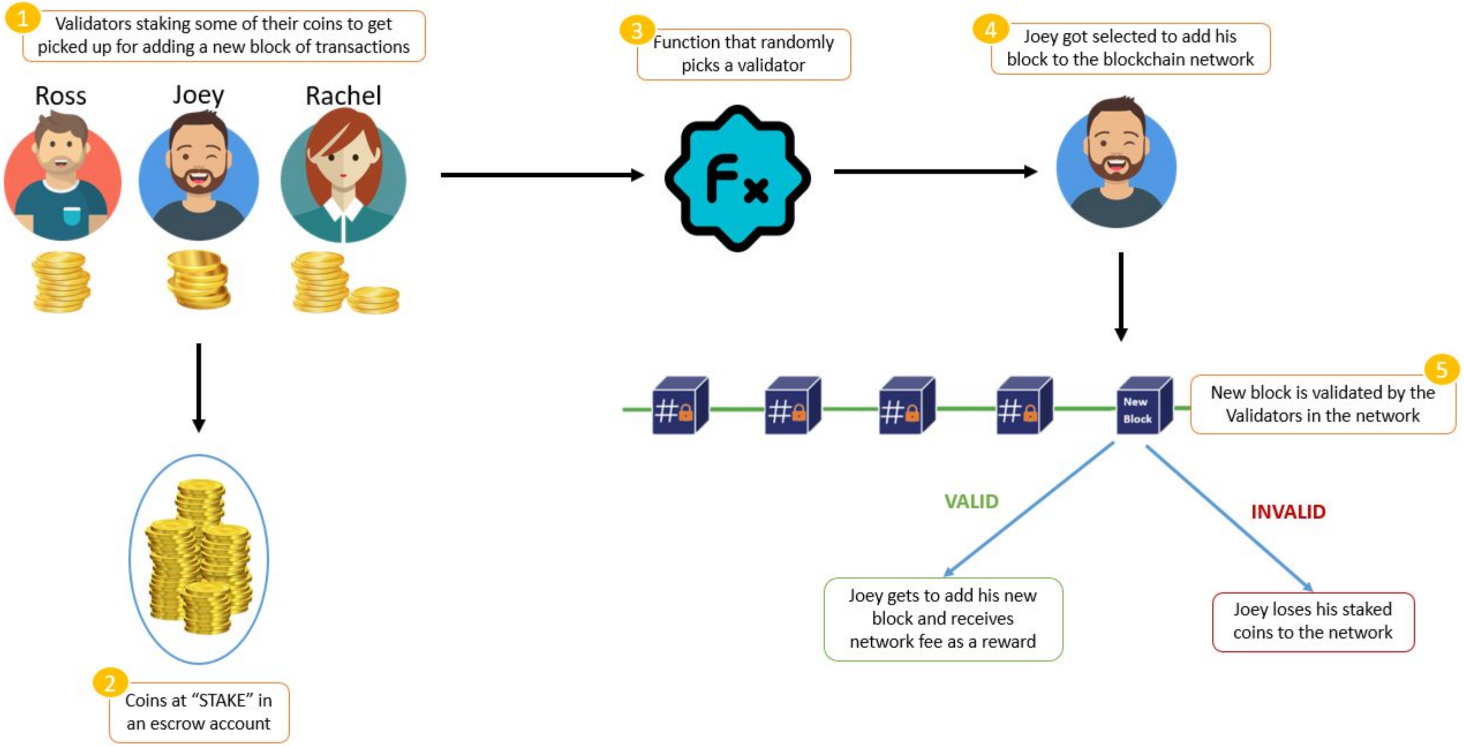
Proof of Stake?

Proof of Stake

Proof of Stake uses way <u>**less**</u> energy by distributing the verification process over the decentralized network.

If you try to game the system, you risk losing your crypto funds since you stake them.

Proof of Stake



Paper Overview

Paper Overview

Efficient Novel Privacy Preserving PoS Protocol

Proof-of-concept with Algorand

Kamilla Stevenson*
Norwegian University of Science and Technology
Trondheim, Norway
kamillastevenson@gmail.com

Oda Skoglund*
Norwegian University of Science and Technology
Trondheim, Norway
odaskoglund7@gmail.com

Mayank Raikwar*
Norwegian University of Science and Technology
Trondheim, Norway
mayank.raikwar@ntnu.no

Danilo Gligoroski
Norwegian University of Science and Technology
Trondheim, Norway
danilog@ntnu.no

Paper Overview

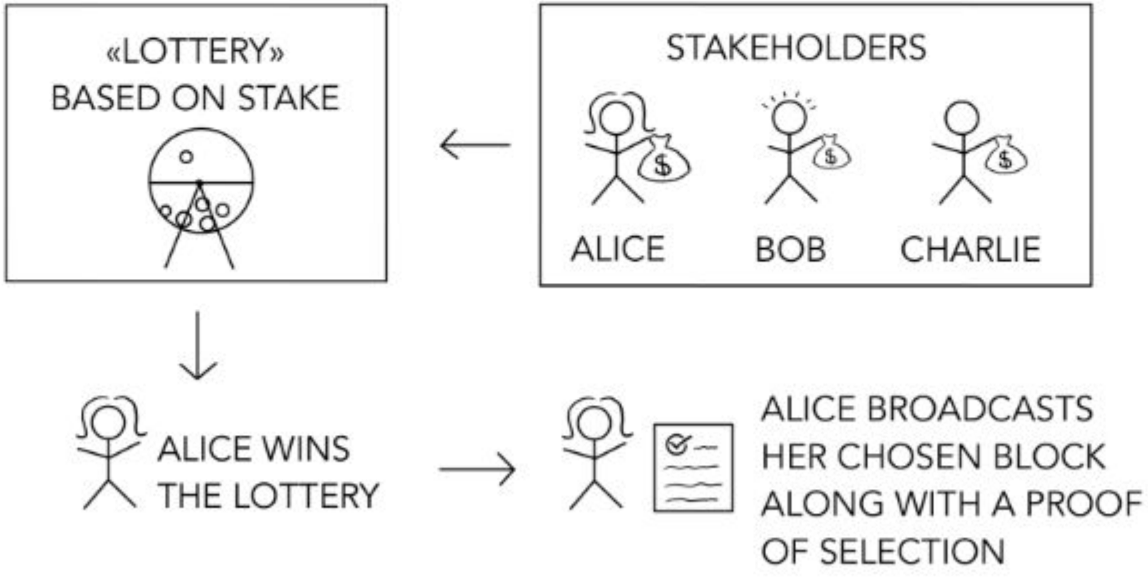


Figure 1: A Simplified Illustration of Selection in PoS

Paper Overview

- The identity of the selected PoS stakeholder is revealed to all participants → This leads to a privacy issue
- Deducting the stake of a participant by frequency analysis.
- The need of privacy in PoS to be equally competitive to PoW

Paper Overview

Newer variations were introduced:

- PPoS (Privacy Preserving) → Zero Knowledge Proof
- Ouroboros Crypsinous → PPoS distributed ledger

But! Those variations were proven that they are still insufficient to protect the stakeholder's identity privacy.

Paper Overview

- Baldimtsi proposed separating the identity and the stake from the validation phase (privacy achieved).
- Trapdoor permutation functionality was used in this research

But! This proposal suffers from high communication complexity and large proof size.

The researchers of our paper improved the scheme of Baldimtsi to be more real world practical.

Analysis + Algorand PoC

Analysis + Algorand PoC

Preliminaries:

Non-Interactive Zero Knowledge Proof (NIZK)

$NIZK = (NIZK.Setup, NIZK.Prove, NIZK.Verify)$.

- $NIZK.Setup(1^\lambda) \rightarrow crs$: Produces a common reference string crs.
- $NIZK.Prove(crs, stmt, w) \rightarrow \pi$: Generates a proof π .
- $NIZK.Verify(crs, stmt, \pi) \rightarrow 0/1$: Verifies the proof π. Outputs 1 if the proof verifies, else 0.

Fully Homomorphic Encryption (FHE)

$FHE = (FHE.Setup, FHE.KeyGen, FHE.Enc, FHE.Dec, FHE.Eval)$.

- $FHE.Setup(1^\lambda) \rightarrow params$: Outputs global parameters.
- $FHE.KeyGen(params) \rightarrow (pk, sk)$: Outputs a public-private key-pair.
- $FHE.Enc(params, pk, \mu) \rightarrow c$: Given a message $\mu \in R_M$, outputs a ciphertext c.
- $FHE.Dec(params, sk, c) \rightarrow \mu^*$: Given a ciphertext c, outputs a message $\mu^* \in R_M$.
- $FHE.Eval(pk, f, c_1, ..., c_l) \rightarrow c_f$: Given the inputs as public key pk, a function $f : R^l_M \rightarrow R_M$ which is an arithmetic circuit over R_M, and a set of l ciphertexts $c_1, ..., c_l$, outputs a ciphertext c_f.

Analysis + Algorand PoC

Algorand is based on choosing committee members using a sortition protocol.

A member can be a potential:

- Block leader
- Verifier

Analysis + Algorand PoC

The scheme is based on Byzantine protocol, where the members pass the information through a gossip protocol. Still, only on single stake setting...

Analysis + Algorand PoC

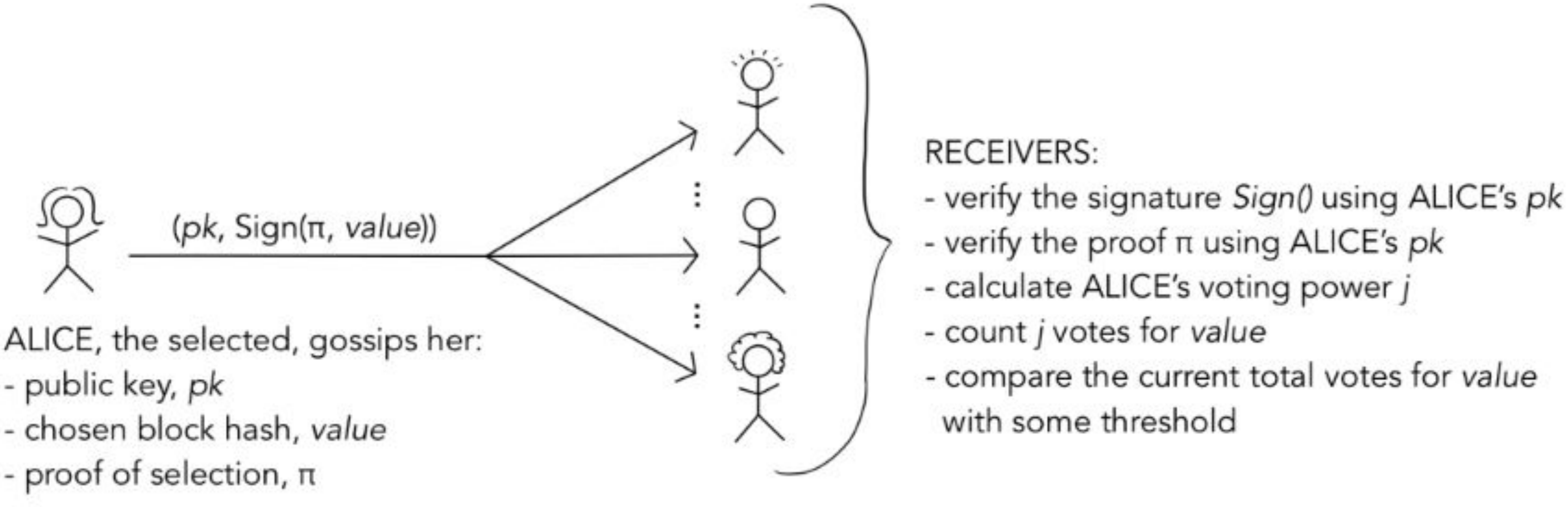


Figure 2: A Simplified Illustration of Consensus in Algorand

Analysis + Algorand PoC

To fix the identity leaks, Baldimtsi presented a flexible anonymous selection functionality using Zero Knowledge Proofs. Like the following scheme:

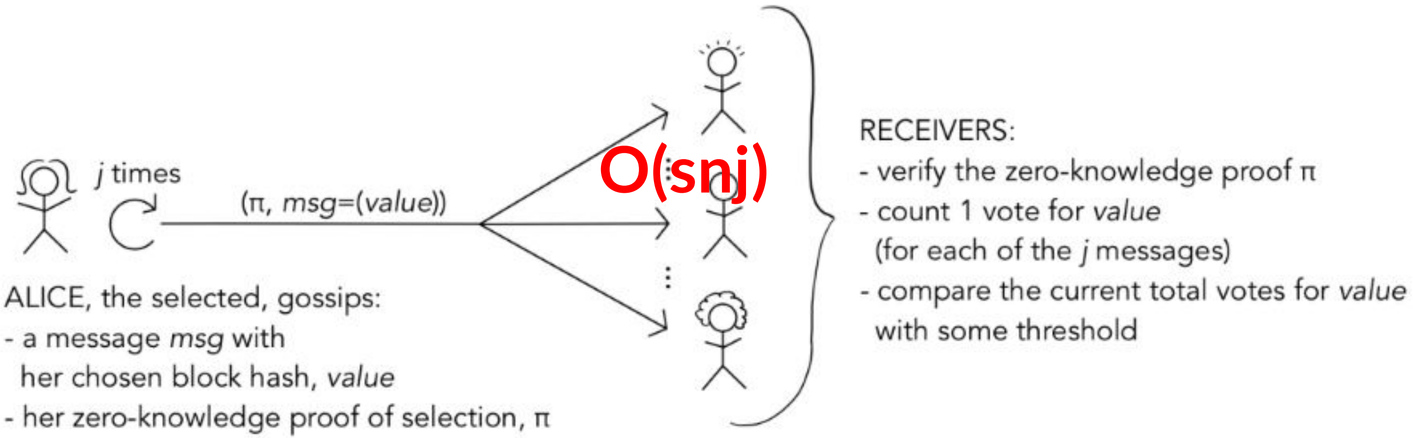


Figure 3: A Simplified Illustration of Baldimtsi et al.'s scheme with Algorand in Multi-stake setting

Overview of the Scheme

Overview of the Scheme

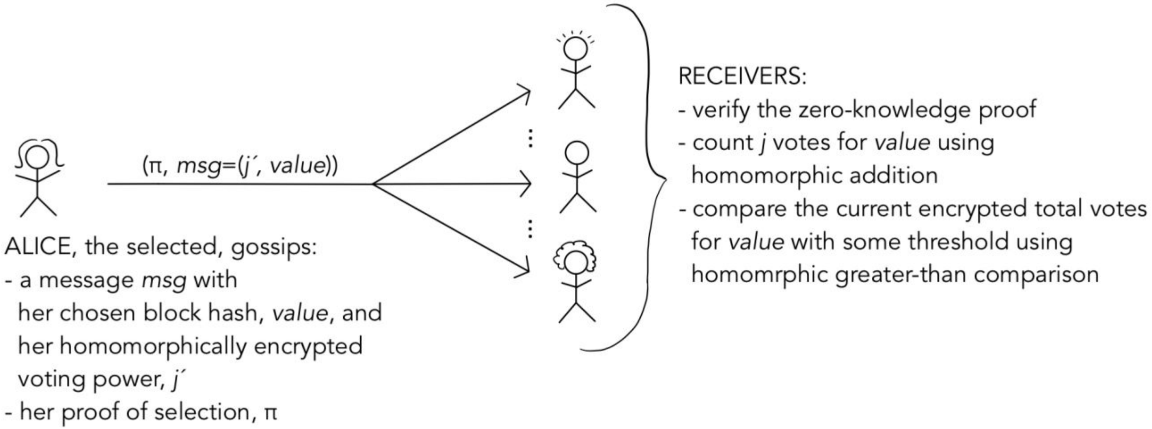


Figure 4: A Simplified Illustration of our scheme with Algorand in Multi-stake setting

Outline of Protocol

(1) Participant P_i, calls $ComitteeVote()$ with inputs tag and $value$.
(2) $CommitteeVote()$ calls $Sortition()$.
(3) $Sortition()$ calls $EligibilityCheck()$.
(4) $EligibilityCheck()$ receives the trapdoor permutation, $\vec{V}_{tag}$, from $ProcessRO()$, computes its inverse, v_i, and calls $Eligible()$.
(5) $Eligible()$ calculates the voting power, j_i, and returns it to the protocol $EligibilityCheck()$.
(6) $EligibilityCheck()$ receives j_i and returns $(j_i, v_i, \vec{V}_{tag})$ to the $Sortition()$.
(7) $Sortition()$ receives $(j_i, v_i, \vec{V}_{tag})$, and applies homomorphic encryption to j_i yielding j_i', and further composes the message, $msg_i = (H(ctx.last\_block), value, j_i')$.
(8) $Sortition()$ calls $CreateProof()$, with inputs msg_i, tag, v_i, $\vec{V}_{tag}$ and j_i.
(9) $CreateProof()$ creates a zero-knowledge proof, π_i, on msg_i and tag, and returns π_i to $Sortition()$
(10) $Sortition()$ receives π_i and returns (π_i, msg_i) to $CommitteeVote()$.
(11) $CommitteeVote()$ gossips the message, $m_i = (tag, \pi_i, msg_i)$.
(12) Upon receiving the gossiped message m_i, participant P_h calls the $CountVotes()$, for tag.
(13) $CountVotes()$ calls $ProcessMsg()$, for m_i.
(14) $ProcessMsg()$ acquires tag, π_i, and msg_i from m_i.
(15) $ProcessMsg()$ calls $Verify()$, on (tag, π_i, msg_i).
(16) $Verify()$ checks that π_i is a valid proof for msg_i and tag and returns 1.
(17) $ProcessMsg()$, upon receiving 1, sets $votes'$ equal to j_i' and returns ($votes$', $value$).
(18) $CountVotes()$ receives $(votes', value)$ and adds the $votes'$ to the $counts'[value]$ using homomorphic addition.
(19) $CountVotes()$ checks if $counts'[value]$ is larger than the threshold using homomorphic greater-than comparison, and if it is, returns $value$.

Protocol

Protocol EligibilityCheck(tag)

Call $ProcessRO(tag)$ and receive $\vec{V}_{tag}$
Compute $v_i = f^{-1}_{TRP.sk_i}(\vec{V}_{tag}[i])$
Call $Eligible(v_i, stake_i, tag)$ and receive j_i
Output $j_i, v_i, \vec{V}_{tag}$

Protocol Eligible$\{v_i, stake_i,$tag$\}$

$p \leftarrow \frac{\tau}{totalStake}$
$j_i \leftarrow 0$
while $2^{\frac{v_i}{len(v_i)}} \notin \left[\sum_{k=0}^{j_i} B(k; w, p), \sum_{k=0}^{j_i+1} B(k; w, p)\right]$
    do $j_i \leftarrow j_i + 1;$
Output j_i

Protocol

Protocol CreateProof($msg_i, tag, v_i, \vec{V}_{tag}, j_i, params$)

1. Compute $C_i^v = F(PRF.sk_i, v_i||tag)$
2. Let $rt_{\vec{V}_{tag}}$ be the root of $MTree(\vec{V}_{tag})$
3. Let $path_{\vec{V}_{tag}[i]}$ be the path to $\vec{V}_{tag}[i]$ in $MTree(\vec{V}_{tag})$
4. Let rt_{pk} be the root of $MTree(pk)$
5. Let $path_{pk_i}$ be the path to pk_i in $MTree(pk)$
6. Let rt_{cm} be the root of $MTree(cm)$
7. Let $path_{cm}$ be the path to cm_i in $MTree(cm)$
8. Compute $\sigma_i = SIG.Sign(SIG.sk_i, msg_i||tag)$
9. Let $x = (rt_{\vec{V}_{tag}}, rt_{pk}, rt_{cm}, tag, msg_i, C_i^v, \vec{V}_{tag}, params)$
10. Let $w = (i, j_i, stake_i, PRF.sk_i, v_i, \sigma_i, pk_i, path_{pk_i},$
    $path_{\vec{V}_{tag}[i]}, path_{cm}, cm_i)$
11. Compute $\pi_{NIZK} := NIZK.Prove(crs, x, w)$
12. Set $\pi_i := (rt_{\vec{V}_{tag}}, rt_{pk}, rt_{cm}, C_i^v, \pi_{NIZK})$
13. Output π_i

Protocol

Protocol Verify($msg, tag, \pi, params$)

1: Call $ProcessRO(tag)$ and receive $\vec{V}_{tag}$
2: Parse $\pi = (rt_{\vec{V}_{tag}}, rt_{pk}, rt_{cm}, C, \pi_{NIZK})$
3: Set $x = (rt_{\vec{V}_{tag}}, rt_{pk}, rt_{cm}, tag, msg, C, \vec{V}_{tag}, params)$
4: Check that $NIZK.Verify(crs, x, \pi_{NIZK}) =?1$
5: If yes, output 1; else output 0

Protocol

procedure SORTITION($value, tag, params$)
    $\left\langle j_i, v_i, \vec{V}_{tag} \right\rangle \leftarrow EligibilityCheck(tag)$
    $\pi_i \leftarrow null$
    $j'_i \leftarrow 0$
    if $j_i > 0$ **then**
        $j'_i = FHE.Enc(params, FHE.pk, j_i)$
        $msg_i = (H(ctx.last\_block), value, j'_i)$
        $\pi_i \leftarrow CreateProof(msg_i, tag, v_i, \vec{V}_{tag}, j_i, params)$
    return $\langle \pi_i, msg_i \rangle$
end procedure

Protocol

```
procedure CountVotes(ctx, tag, T, τ, λ)
    start ← Time()
    counts' ← {}
    msgs ← incomingMsgs[tag].iterator()
    while TRUE do
        m ← msgs.next()
        if m =⊥ then
            if Time() > start + λ then return TIMEOUT
        else
            ⟨votes', value⟩ ← ProcessMsg(ctx, m)
            if votes' = 0 then continue;
            counts'[value] =
FHE.ADD(FHE.pk_a, f, counts'[value], votes')
            x' ← counts'[value]
            y' ← FHE.Enc(params, T · τ + 1)
            b ← FHE.GreaterThan(x', y')
            if b return value
end procedure
```

Conclusion

Evaluation of the Scheme

- Aim is to achieve full privacy concerning stake and identity. Baldimtsi et al.'s scheme provides privacy of identity and homomorphic encryption provides privacy of stake.
- Better complexity in terms of computation and communication. O(sn) complexity compared to Baldimtsi et al.'s scheme with O(snj).

Conclusion

By removing the need for multiple unlinkable proofs in the multi-stake setting through the use of homomorphic encryption, the scheme performs better than Baldimtsi et al.'s scheme in multi-stake instantiation of Algorand.