CREN Presentation(Snow White)

Jun Hong Peng

Blockchain in a nutshell

We want a distributed system for record keeping

Ideally, everyone should maintain and agree on the same records

Assuming an open network where anyone can join/leave

Assume individual participants are only incentivized by their own selfish interest

Blockchain in a nutshell (cont.)

Goal:

build a system that discourages selfish entities from deviating from the network protocol

& capable of defending against the existence of a few bad actors

namely, a percentage of adversarial participants bellow a certain threshold should not jeopardize the integrity of our record keeping

Blockchain in a nutshell (cont.)

Cannot designate one entity to do all the record keeping and just propagate to the other replicas

Why?

- compromising that one entity = compromising the entire network
- a fixed eternal leader means attackers get ∞ attempts at compromising it

Blockchain in a nutshell (cont.)

One solution is to periodically randomly select a new "leader"

**Not going in to details of proof-of-work protocols and why they work*

Essentially, it is a continuous lottery game

Winner randomly selected at each round

First such simulation of "fair" random selection: Nakamoto's Bitcoin

Take away: proof-of-work is just one such method to achieve "random selection"

Disadvantage of PoW

Time consuming

High energy consumption

Not so random: mining pools, etc.

Adversarial attacks due to time lags: block withholding attack, etc.

Alternatives

Researchers have been looking for more efficient methods of distributing leadership responsibilities

One idea is proof-of-stake:

tie an entity's probability of becoming leader with its "stake" in the network

can further incentivize good behaviour by punishing malicious leaders (e.g. take away some of their stake)

Proof of Stake

All existing proof-of-stake networks lack mathematical security guarantees

Main talking point today:

- A new proof-of-work system: *Snow White*
  - First mathematically provably secure proof-of-work
  - Fixes some existing issues that plague other blockchain systems

I will not go in to details of *Snow White's* proof, rather a description of the protocol

Snow White (provably secure PoS)

Snow White leverages another consensus scheme: Sleepy Consensus

Uses the new "*fruitchain*" to further discourage harmful behaviours

I will be describing the *Sleepy Consensus* and *fruitchain* as well as how they fit in to **Snow White**

Background info (Proof of Work Protocol)

At any given block n, takes inputs:

- h_{n-1}, Hash of previous block
- TX, any subset of the transaction pool
- $\eta \leftarrow \{0,1\}^n$, a random number

- Compute $H(h_{n-1}, TX, \eta)$
- Wins if $H(h_{n-1}, TX, \eta) < d_p$, where d_p is the difficulty parameter

Sleepy Consensus Protocol

A player wins if:

$$PRF_{k_0}(P, t) \oplus PRF_{SK[P]}(t) < D_p$$

k_0: a common public random seed for the first PRF

P: the identifier (public key) of p$layerP$

$SK[P]$: secret key of player P

t: time, all times must be increasing, future times are invalid

Sleepy Consensus (cont.)

What it achieves:

Instead of having participants compete in finding a value smaller than the defined security parameter, we let "fate" decide.

A leader is chosen based on the current time and their *ID*: public key

Think of the idea that all time past, present and future are constant and predetermined. The leader at any given time is already "predetermined"

Takes away the meaningless computational guess work in p-o-w

Sleepy Consensus

Adversary Super Powers:
- Sleep any honest player
- Delay messages

Why we need the time restriction:

all times must be increasing, future times are invalid?

From Prev. Slide:
$PRF_{k_0}(P,t) \oplus PRF_{SK[P]}(t) < D_p$

Without the time restriction, can attack by proof-of-work

$PRF_{k_0}(P,t)$ alone does not work:

k_0 is public

attacker can precompute and figure out which player P will win at what time

attacker can then proceed to sleep those players and stall the network

Sleepy cont.

This is why we need the 2nd part:

$$PRF_{k_0}(P,t) \oplus PRF_{SK[P]}(t) < D_p$$

Because the 2nd portion contains a player's secret key, an adversary will not be able to precompute the final **xored** value of player P at time t without knowing $P's$ secret key

Sleepy cont.

From Prev. Slide:
$PRF_{k_0}(P,t) \oplus PRF_{SK[P]}(t) < D_p$

Since $SK[P]$ is private, no one except P knows it

P can use different $SK[P]$ values until finds one satisfying:

$$PRF_{k_0}(P,t) \oplus PRF_{SK[P]}(t) < D_p$$

We need a zero-knowledge way to make sure that P is indeed using its real secret key without actually knowing the value of $SK[P]$

solution: Verifiable Random Function (VRF)

VRF

Recall:

- CDH assumption:
  - Given $g, g^a \; and g^b$ hard to find g^{ab} etc.
- DDH assumption
  - Given $g^a \; and \; g^b, g^{ab}$ indistinguishable from g^R
- Inverse CDH & DDH assumption

Bilinear Mapping

$$e(g^a, g^b) = e(g, g)^{ab}$$

*CDH, DDH, inverse CDH+DDH assumption applies for Bilinear Mapping as well

VRF cont.

Let us define the following

VRF function:

$$\mathrm{VRF}_{SK[P]}(t) = e(g,g)^{\frac{1}{t+SK[P]}}$$

Proof String:

$$\mathrm{Proof}_{SK[P]}(t) = g^{\frac{1}{t+SK[P]}}$$

Given t and PK, we want to prove the valid SK is used in $VRF_{SK[P]}(t)$ without knowing SK

VRF cont.

Goal:

a player P joins the network, chooses a secret key $(SK[P])$ and broadcasts the corresponding public key $PK[P] = g^{SK[P]}$ to the network

at some time t, P would broadcast a value he claims to be the output of

$$\mathrm{VRF}_{SK[P]^*}(t)$$

Without learning the value of $SK[P]$ others on the network would like to verify

$$SK[P]^* = SK[P]$$

VRF algorithm

Consider 2 functions:

$\text{Prove}(\text{SK}^*, t)$ & $\text{Verify}(\text{VRF}_{\text{SK}_1^*}(t), \text{Proof}_{\text{SK}_2^*}(t), t)$

Prove:

Take input SK^*, t

return: $\text{VRF}_{SK^*}(t)$ and $\text{Proof}_{SK^*}(t)$

From Prev. Slides:

$PK = g^{SK}$

$VRF_{SK}(t) = e(g,g)^{\frac{1}{t+SK}}$

$\text{Proof}_{sk}(t) = g^{\frac{1}{t+s}}$

$e(g^a, g^b) = e(g,g)^{ab}$

VRF algorithm

From Prev. Slides:

$PK = g^{SK}$

$VRF_{SK}(t) = e(g,g)^{\frac{1}{t+SK}}$

$\text{Proof}_{sk}(t) = g^{\frac{1}{t+SK}}$

$e(g^a, g^b) = e(g,g)^{ab}$

$\text{Verify}\left(\text{VRF}_{\text{SK}_1^*}(t), \text{Proof}_{\text{SK}_2^*}(t), t\right)$:

takes the output of Prove(SK, t) as inputs

if $\boldsymbol{e\left(g, \text{Proof}_{sk_2^*}(t)\right) == \text{VRF}_{sk_1^*}(t)}$ && $\boldsymbol{e\left(g^t PK, Proof_{SK_2^*}(t)\right) == e(g,g)}$

return TRUE

return FALSE

First condition checks if $SK_1^* == SK_2^*$:

LHS:

$$e\left(g, \text{Proof}_{\text{SK}_2^*}(t)\right) = e\left(g, \frac{1}{g^{t+\text{SK}_2^*}}\right) = e(g,g)^{\frac{1}{t+\text{SK}_2^*}}$$

RHS:

$$\text{VRF}_{SK_1^*}(t) = e(g,g)^{\frac{1}{t+\text{SK}_1^*}}$$

$\therefore$ Equality only holds if $\text{SK}_1^* == \text{SK}_2^*$

VRF algorithm

From Prev. Slides:

$PK = g^{SK}$

$VRF_{SK}(t) = e(g,g)^{\frac{1}{t+SK}}$

$\text{Proof}_{sk}(t) = g^{\frac{1}{t+SK}}$

$e(g^a, g^b) = e(g,g)^{ab}$

$\text{Verify}(\text{VRF}_{\text{SK}_1^*}(t), \text{Proof}_{\text{SK}_2^*}(t), t)$:

takes the output of Prove(SK, t) as inputs

if $\boldsymbol{e\left(g, \text{Proof}_{sk_2^*}(t)\right) == \text{VRF}_{sk_1^*}(t)}$ && $\boldsymbol{e\left(g^t PK, Proof_{SK_2^*}(t)\right) == e(g,g)}$

return TRUE

return FALSE

Second Condition checks if the SK^* used to generate Proof & Verify is same as the SK corresponding to the public key:

LHS:

$$e\left(g^t PK, \text{Proof}_{SK_2^*}(t)\right) = e\left(g^t g^{SK}, g^{\frac{1}{t+SK_2^*}}\right) = e(g,g)^{\frac{t+SK}{t+SK_2^*}}$$

LHS == RHS iff SK == SK_2^*

∴ We have a zero-knowledge proof varifying the SK^* used in VRF equals SK

FruitChain (inspiration)

Another important mechanism of $SnowWhite$ is $FruitChain$

In current PoW, there is only 1 winner at each timestep

It takes 2-4 years for an average participant to win 1 round

This leads to people joining mining pools (from decentralized to centralized)

PoW also suffers from "block withholding" attacks (always accepts longest chain)

FruitChain (background info)

To understand $FruitChain$, we must first dive into economics and game theory

In a normal "first price" auction:

- Players are incentivized to bet their max valuation
- In fact, betting their max valuation is a dominant strategy

In "2nd price" auction:

- Only a weakly dominant strategy to bet at one's max valuation

2nd Price Auction

Unlike a normal auction, where the highest bidder wins and pays his respective bid, a 2nd Price auction winner pays the next highest bid (in other words, the highest losing bid)

$E.g.$:

2 player auction: P_1, P_2

P_1 bids 100 for the object, P_2 bids 10

P_1 wins the bid, but pays the price P_2 bidded. that is P_1 pays 10

For details of the strategic game implication of this and the Vickery auction, refer to Google and Wikipedia

FruitChain

At any given round, we define the hash for that round as

$$h := H(h_{-1}, \eta, F, TX)$$

h_{-1}: hash of last block
η: nounce
F: set of fruits
TX: set of transactions
$[h]_{:k}$: prefix of h
$[h]_{-k}$: suffix (last k $bits$)

If $[h]_{:k} < D_p$ (mined block)

append block to chain and broadcast

If $[h]_{-k} < D_{p(\text{fruit})}$ (mined fruit)

broadcast fruit, add it to the existing pool of fruits

FruitChain

At a round n, instead of rewarding the miner who found block n, equally reward owners of the **fruits** contained in the n^{th} block

This achieves the mining pool affect without collaboration

- since each block must contain a set of fruits
- and the reward goes to the fruit owners not the miner of the block

Block-withholding does not work

Block withholding

This attack works against p-o-w networks

Upon discovering a valid block b_{n+1} for a chain C, adversary A does not broadcast his solution to the network. chain length of C is n

Instead, he tries to further extend his own new chain $C_A = C \parallel b_n$, while the rest of the network is still working on C

If some other player solved the $n + 1$ block, A would quickly publish his chain to split the network (split is possible due to propagation delay)

essentially, A gained unfair advantage of extra time solving his own chain C_A

This attack not only gives A extra time, it can also overwrite a block mined by some other player.

Snow White (combining everything)

Snow White uses the **Sleepy Consensus** and **FruitChain** to build a non-proof-of-work system that also eliminates selfish mining

Sleepy Consensus can be turned into proof-of-stake by associating each unit of network currency with an unique ID (public key)

Thus, more stake one has = higher chance of becoming leader

Snow White (dealing with Nothing-at-Stake attacks)

- Lock a portion of the sitting committee's funds while they are in power
- Past committee has power to rewrite their blocks with out penalties once they transfer funds out
- If there are attempts of rewriting histories, new nodes will not be able to distinguish
- Solution: a list of trusted nodes