Decentralized Random Number Generation

Peter Robinson, July 5, 2018

CONSENSYS

Overview

- Decentralised random number generation algorithms suffer from the ***Last Actor Problem***, in which the last participant to reveal their share can manipulate the generated random value by withholding their share.
- An encrypted share threshold scheme is proposed which prevents this attack.

Background Material

Traditional DRNG Algorithm

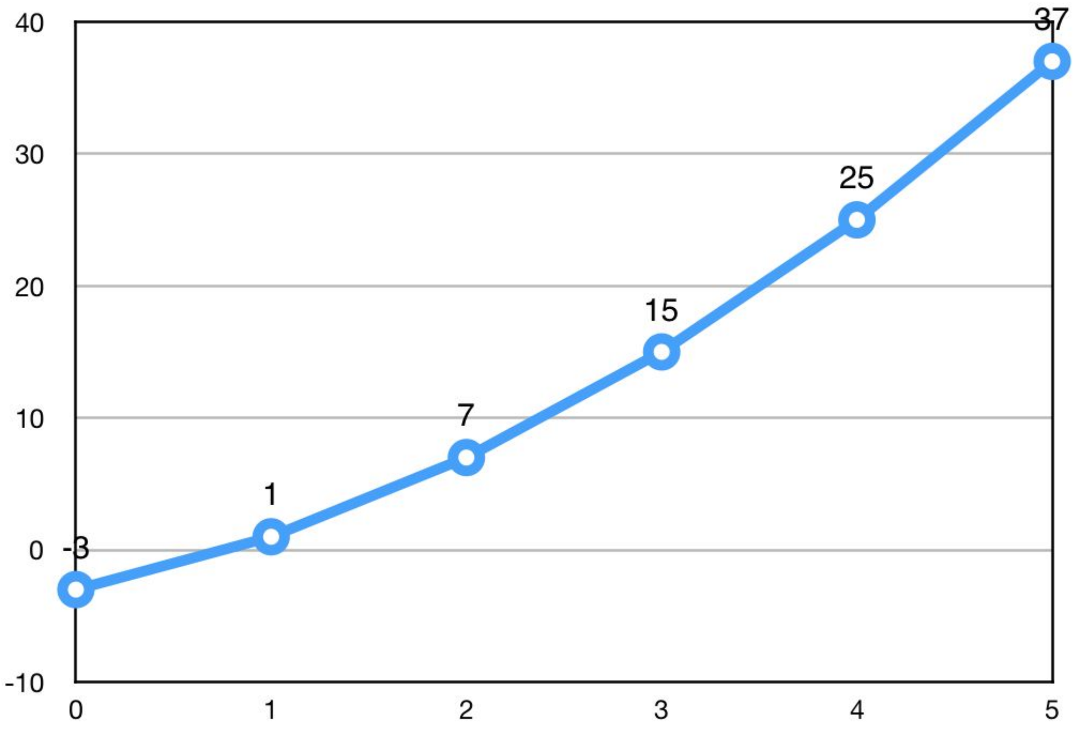
- **N** participants register.
- All participants generate random value R_i and calculate a commitment value:
  a. C_i = Message Digest (R_i)
- All participants post their commitment value: C_i
- Once all commitments are posted, all participants post their random value R_i.
- The random output is calculated as:
  a. Random Output = R_0 XOR R_1 XOR R_2 XOR… R_N
- Participants who do not reveal their random values are fined.
- The random result could be used to determine who gets to generate the next block, with associated block generation reward.

Last Actor Problem

- One of the participants could wait for the other **`N-1`** random values to be posted. They could choose to post their value or withhold their value, thus influencing the output value.
    a. The fine issued for not revealing their random value may be less than the benefit.
- Multiple participants could collude, to determine which set of their random values could be withheld to yield the optimal result.

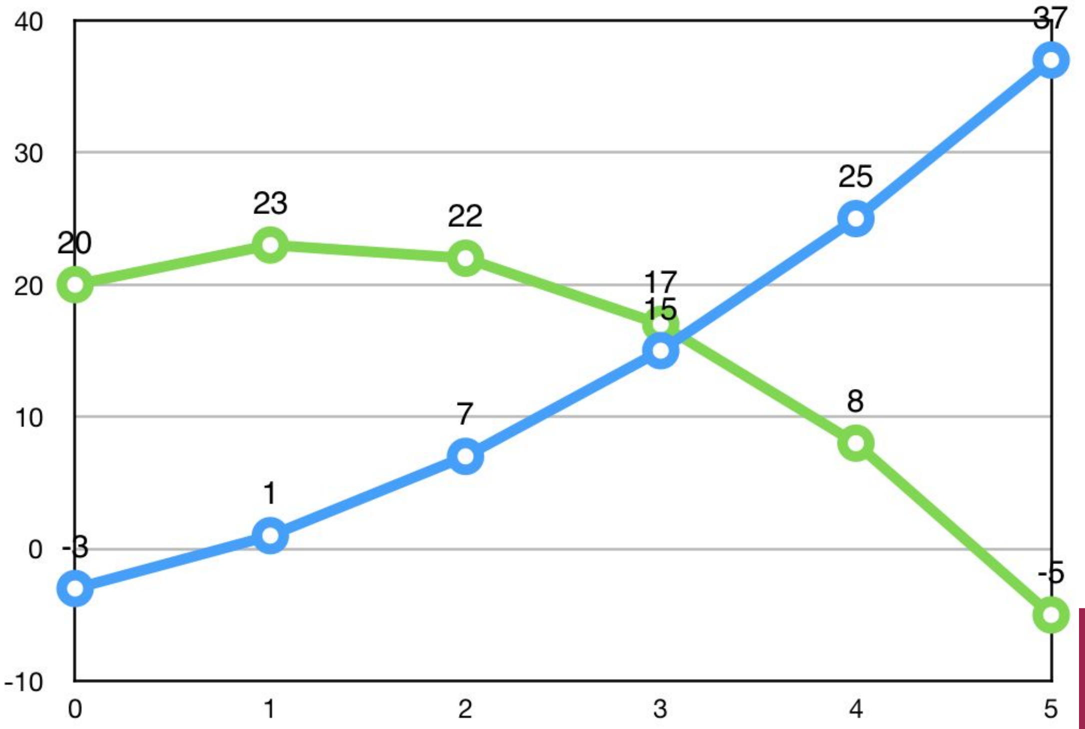
Polynomials

- $y = x^2 + 3x - 3$
- An equation of order 2 is defined by any three points.



Polynomials

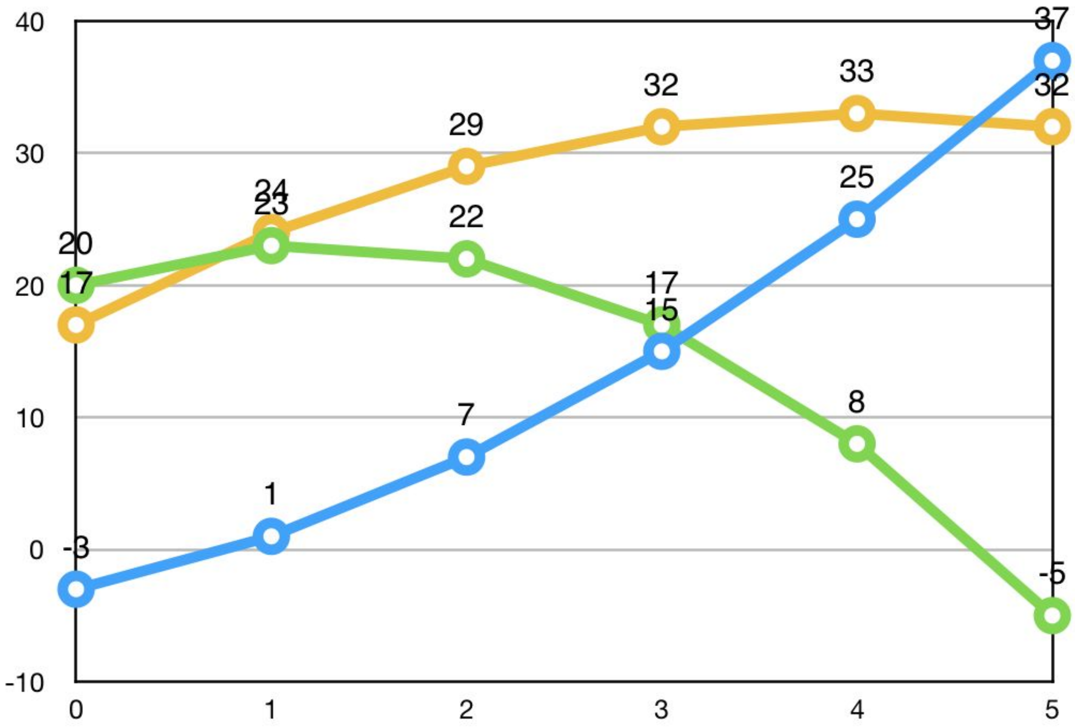
- $y = x^2 + 3x - 3$
- $y = -2x^2 + 5x + 20$



Polynomials

- $y = x^2 + 3x - 3$
- $y = -2x^2 + 5x + 20$

- Adding equations:
- $y = -x^2 + 8x + 17$



Shamir's Threshold Scheme

- Split a secret into **n** shares.
- Any **m** of **n** shares can be combined to yield the secret.
- No information about the secret is revealed if less than **m** shares are revealed.

REF: https://cs.jhu.edu/~sdoshi/crypto/papers/shamirturing.pdf

Shamir's Threshold Scheme

Define an equation:

- $Y(x) = a_0 + a_1 X + a_2.X^2 + ... + a_{m-1}.X_{m-1} \bmod P$

Where:

- P is a large prime number.
- a_0 is the secret value.
- m is the threshold number of shares.

Shamir's Threshold Scheme

Process:

- Randomly generate coefficients $\mathbf{a}_0$ to $\mathbf{a}_{m-1}$ in the range **1** to **P-1**.
- Securely distribute **n** shares.
- Any **m** shares can be combined to yield $\mathbf{a}_0$.

Algorithm

Set-up

- Deploy a smart contract to an Ethereum blockchain.
- Choose large prime number `P`.

Registration

To register, each participant:

- Generates an ephemeral ECC key pair (could also be RSA).
- Publishes their ephemeral ECC public key by sending a transaction to the contract setting the public key value.
- Each participant has an x value. The participant’s x value is calculated as:
    - x_i = (Participant’s Ethereum Address) mod P

Generate Random Coefficients

All participants:

- Generate `m - 1` random coefficients for an equation in the range `1` to `P - 1`.
- Calculate the `y` values for each of the participant `x` values.

Post Commitment

All participants:

- Post to the contract the message digest of the **`y`** values for each of the participant **`x`** values.

Any participant which does not post commitment values drops out of the random number generation process and is fined.

Generate Random Coefficients

All participants:

- Post to the contract the encrypted **`y`** values for each of the participant **`x`** values, encrypted against the public keys of each other participant.

Any participant which does not post all of the encrypted y values drops out of the random number generation process and is fined.

Post Private Keys and Calculate Random

All participants:

- Post their private decryption keys.

Once **m** private keys are posted, the contract has enough information to calculate the random value and check for correctness.

Correctness can be checked for by decrypting the encrypted **y** values, checking commitments, and checking that the order of the curve that each entity posted is **m - 1**.

Random value = $\sum_1^n a_{0i} \bmod P$.

Attacks

Attacks

`m` colluding attackers could:

- Decrypt the encrypted `y` values.
- Choose which attacker should withhold their private key, thus manipulating the output.

`n-m+1` colluding attackers could:

- Withhold their private keys, thus preventing decryption of `m` values.
- Denial of service.

Conclusion

Conclusion

Encrypted Share Threshold Scheme with Commitments:

- Leaderless.
- Withstands up to m of n attackers before output can be manipulated.
  - Manipulation limited to combinations of pre-committed-to values.
- Withstands up to $n - m + 1$ attackers attempting to take service offline.

More Reading

More reading on alternative schemes:

- https://ethresear.ch/t/leaderless-k-of-n-random-beacon/2046
- https://dfinity.org/pdf-viewer/pdfs/viewer?file=../library/dfinity-consensus.pdf (section 7.3).