I am looking to create a simple smart contract based algorithm

that provides common random number generation

, provided that t

out of N

participants are honest, where t * 2 > N

The algorithm has the following properties

- at the end, all honest participants know the common random number R
- malicious participants are not able to influence R

or make the algorithm stuck.

Preliminary setup

Each participant j

register her public keys P[j]

with a smartcontract DRNGManager

.

Together with the public key P[j]

each participant submits to DRNGManager

a ZK proof that P[j]

is valid and that she knows the corresponding private key.

Commit phase

(10 minutes)

Each participant j

generates a random EC polynomial of degree t

POLY[j]

.

The participant then generates a vector of polynomial evaluations A[i] = [POLY[j]]

at N

integer points i

.

The participant will then encrypt the evaluations to obtain a vector of encrypted polynomial evaluations G_[j] = [Encrypt(POLY[j])]

.

It then submits to DRNGManager

- vector G[j][i]
- commitment to POLY[j]
- a ZK-proof that G[j][i]

were correctly generated from POLY[j]

.

DRNGManager verifies ZK-proof on receipt

After the commit phase, DRNGManager

will contain j

valid vectors G[j]

, where j >= t

.

Reveal phase

. (10 minutes)

Each participant j

will be able to decrypt and reveal to DRNGManager

a vector of points POLYj

. The participants will then submit these vector to DRNGManager

together with a ZK proof that reveal was done correctly.

After the reveal phase, DRNGManager

will include k

reveals, where k >= t

.

RNG computation phase

. For each committed polynomial POLY[j]

, each participant is then able calculate random number R[j]

= POLYj

. The common random is then XOR of all R[j]