Semaphore RLN, rate limiting nullifier

Thanks to HarryR for review and suggesting using shamir secret sharing here instead of weak key cryptography.

Thanks to Mikerah for review.

Intro

Using semaphore we use nullifiers to limit the rate at which someone can signal. However if someone breaks this we cannot punish them because we cant link the signal to them.

This has lead to Anon rep proposal. This solution should work well in some situations. It imposes a high cost requiring every member of the group to take action in the case of a fork. This can be mitigated with a stake but the gasBlockLimit imposes an upper limit on the number of participants these groups can have.

This excludes a bunch of interesting user cases so here we propose a new kind of nullifier that for a single external_nullifier

reveals no information but if someone repeats the same external_nullifier

with different signal (ie they spam) it reveals their private key allowing us to remove them from the tree.

We hope to build upon this work to define

1. mix networks with strong privacy AND spam resistant.
2. anonymous rate limited universal logins; We limit participants to make more than x requests per second and if you do you get kicked out.

Background

We have a merkle tree that stores a list of our members. Each member has a public key in the tree to which they know the private key.

Each member in the group can signal. They use a snark to prove that they are a member of the group without revealing which member they are. Each signal has the following public parameters

1. Signal; What you are saying
2. External_nullfiers; This prevents the linking of differnt signals together.
3. nullifiers; hash(extrenal_nullfier , leaf_private_key)

which is a finger print uniuq to this member for this external_nullifier

.

The main weakness here is that it does not let us remove anyone for bad behavior as every time they use a different external_nullifier

they are anonymous again.

We define one class of “bad behavior” being making more than X signals per second. Where X is some parameter that we define about our system.

To enforce this we can set the external_nullifier

to be a time stamp and each epoch every member is allowed to make one signal. Participants can ignore a signal if it comes from a future epoch or a epoch in the past.

Method

Firstly we have a smart contract that allows anyone to deposit some currency and join our group. At any point a users can be removed from the group if someone calls a function passing their private key as an input. Anyone who does this will receive 33% of the slashed stake the remainder is burned.

So we want to have a way to force anyone who spams the network to reveal their private key so they can be burned.

In the previous snark the nullifier is hash(external_nullifier, leaf_private_key)

We need to add a few things to the snark

1. We generate a nullifier_private_key

based upon hash(external_nullifier, leaf_private_key)

1. We encrypt the leaf_private_key

with the nullfier_private_key

and reveal the result as a public input.

1. We user shamir secret sharing to encode the nullifier_private_key

so that 51% of the shares are required to reconstruct the secret.

1. We then use signal

to randomly select 50% of the shares and reveal them.

Now each time you create a signal with the same external_nullifier

but different signal

it

1. calculates the same nullifier_private_key

key and encrypts your leaf_private_key

with it.

1. It calculates the same shamir secret and shares.
2. But it reveals a different 50% of the shares.

The chances are very small that a different signal will result in the same 50% of the share being revealed. So we are confident that any spammer will reveal their whole private key on their second message.

Limitations

1. In the p2p context its difficult to identify users who are spamming because sharing every message would be very expensive.

We should also include as public parameter the hash(external_nullifier, leaf_private_key, constant)

so its easy to see if two signals were created with the same member for the same external_nullifier

but different signals. Furthermore we could share a bloom filter of these with our peers who can then request the snark from us and slash the participant.