Thanks to Wei Jie Koh for review and feedback

# Intro

There has been a lot of interest recently in creating a “Login with Ethereum” button. This seems like a reasonable thing to do. However it does link your login to your transaction history. This can be mitigated with a mixer or using an account with zero funds.

Previuosly use of zk-SNARKS have been focused on cryptocurrency transactions. Here we propose the applications of zk-SNARKS to universal login. Using the same circuit we use in Semaphore, we allow members of a group to login to a service without revealing which member they are.

# Note on statefulness

This is directly applicable to stateless applications, where each user does not have a personal state, such as for VPNs. Email, by contrast, is an example of a stateful application. Stateful dApps can also be supported where all of a user’s information is encrypted and anyone who logs in can requires random peices of information which only a legitimate user can decrypt.

# Semaphore Basics

Please see https://github.com/kobigurk/semaphore for introduction

# Semaphore Anon Login

The authenticator creates a list of public keys that it will accept as logins. It then publishes this list and shares the merkle proofs with the users.

Here we need to set the external_nullifier

so that double logins are not possible.

If we set external_nullifier

equal hash("ANONLOGIN" + URL)

of the website this will allow give each login a uniqe nullifier which should not reaveal any information about the users.

# Many Requests Per Site

We can also support multiple requests to a single site. This is useful with VPN logins or sites that require stronger limits on what users can do.

The VPN provider only wants to accept a limited number of logins from every user their for they only accept semaphore proofs with

1. external_nullifer == hash("ANONLOGIN", URL, day_count , ticket_number
2. signal == hash(target_host_pub_key, cookie)

target_host_pub_key

prevents replay attacks and cookie

is the credentail that is used for the rest of this session.

where day_count

is the number of days since January 1st, 1970 at UTC

and ticket_number

is any number more than 0 and less than max_ticket_number

You can see that each day each user can have a maximum of max_ticket_number - 1

signals that have uiniqe nullifiers for each URL. Thus the authenticator knows that the total number of connections they need to serve is bounded and they can limit the bandwidth of each connection.

## Group Creation

The creation of these groups is still an open problem. Besides centralised mechanisms, we can employ decentralised mechanisms such as:

1. Burning ETH to impose an econmoic cost of account creation.
2. Charging a fee to join the group and pay this fee to the infrastucture provider.
3. Uniqueness DAO: https://www.humanitydao.org
4. Individuality party, where we hold an global party and users participate to join the group.

## Conclusion

Here we describe a login mechanism that allows users to authenticated by an authenticator without reavealing which account they are using.

This allows us to use stronger proofs of individuality and still maintain privacy. The creations of these groups is an open problem and not addressed here.