Pseudonym Pairs: A foundation for proof-of-personhood in the web 3.0 jurisdiction

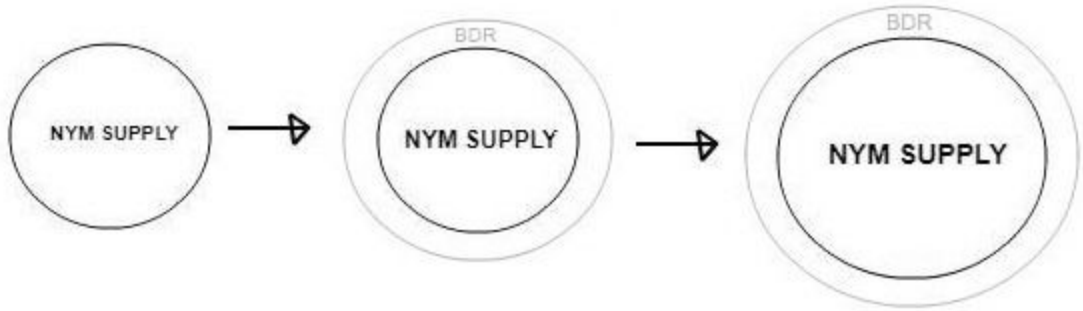
ABSTRACT: Pseudonym Pairs is a dApp for global proof-of-personhood, through monthly pseudonym events that last 20 minutes, where every single person on Earth is randomly paired together with another person, 1-on-1, to verify that the other is a person, in a pseudo-anonymous context. The events provide NYM tokens, global personhood tokens, untraceable from month to month and disposable, a sort of “temporary access tokens” similar to festival bracelets. The proof-of-personhood is that you are with the same person for the whole event.

1-on-1 verification of (pseudo-anonymous) personhood

Within the 1-on-1 pairs, people can socialize as they want, and can be seen as being employed in government positions, expected to stay within the pair for the entire duration of the pseudonym event. The 1-on-1 pairs is the standard organization, requiring mutual verification. In the case of a problem, such as a bot attacker, or, a person not showing up, people can break up their pair, to be assigned to be verified by another pair (2-on-1), similar to how people are verified at the “virtual border”. (see below)

How to opt-in to Pseudonym Pairs

The population is used to secure a "virtual border" around the network, and “border tokens” (BDR) can be bought to apply at the “virtual border” and meet a random pseudonym pair, that verify the person that opts-in. The “border tokens” are distributed through the population, each person can issue 1 BDR, and each time BDR is issued, the ability to issue one more BDR is given to a random person within the pseudonym pool, distributing the ability to invite new people onto the population as a whole, making it possible for the network to accept new people multiple times its population size, so that it can grow from 0 to potentially 5 billion people.



Example of how to bootstrap a Pseudonym Pairs network

The network, as a collective or swarm, can choose how fast it wants to grow, based on what they discover is secure enough. If 6 new people are allowed in per pair, then the growth is 4-fold each month, and the population is 2*4^n after n events, starting with a single pair, 2*4^0 = two people. With that growth rate, you get to (2*)1 million people after 10 events, and (2*)1 billion after 15 events.

The growth rate can be decreased gradually, approaching zero as the entire global population has been accounted for. For example, 4-fold growth for 10 events up to 2 million people, then 2-fold another 10 events, 2 billion people within 20 events. Then, grow with 1 person per pair for 4 events, 1.5-fold, and the network grows to 10 billion people within 24 events, 2 years exactly.

The population sorts themselves into pairs

The pair sorting is invoked by each person, people are sorted into two lists (together forming pairs), and the lists are continuously shuffled with each new person who invokes sortMe(). This sorting mechanism keeps the computational cost per person low, and forms complete pairs regardless of how many of the people who registered choose to commit with sortMe().

```
function sortMe() atTime(0, pseudonymEvent) {
    uint8 idx;
    uint totalSorted = pairingUtility[0].counter + pairingUtility[1].counter;
    idx = totalSorted % 2;
    pairingUtility[idx].counter++;
    totalSorted++;
    pseudonymID[msg.sender] = totalSorted;
    uint pos = pairingUtility[idx].counter;
    uint randomNumber = 1 + labyrinth.generateRandomNumber() % (pos - 1);
    pairingUtility[idx].index[pos].push(randomNumber);
    pairingUtility[idx].index[randomNumber] = pos;
}
```

You simply sort yourself into a pair, at any time from the end of the previous event, up until the next event, and this will cost you a tiny amount of GAS. Every other person is sorted in pairingUtility[1], instead of pairingUtility[0], combined, a list of pairs when the event begins. When you call SortMe(), the pairingUtility counts how many have been sorted in the pairingUtility you are assigned to, and then randomly places you in the position another person had, moving that other person to the end of the list. Over time, the pairs get shuffled.

Profitability of collusion attacks

The only attack vector I see in Pseudonym Pairs, collusion attacks, they follow an inverse square law, the return decreases more and more the fewer people attack the network.

$$\frac{colluders}{population}^2$$

If 25% of the population attacks the network, they get $\frac{1}{4}^2$ = 6.25% bots, if 10% of the population attacks the network, $\frac{1}{10}^2$ = 1%.
The colluders get *colluders/population* more than they get otherwise. If 10% of the entire human population collude and together attack the network, they get 1/10 more than baseline, if 5% of the entire population attack the network, they get 1/20, 5% (0.05x personhood tokens per attacker, exactly how that is divided between the attackers, they get to decide for themselves.)

Scheduling the Pseudonym Event

The Pseudonym Event is global, and singular, and so to be fair, the event is scheduled to a random hour, and cycles over 24 hours. An example schedule, that fits with prior norm systems, is to happen on weekends in the 7-day week, 13 months with 28 days. The event could be set for 07:00 on Saturday 22 December (UTC), which is 21:00 Saturday UTC+14:00, the earliest time zone on Earth (areas in this zone are the first to see a new day), and on a random hour that then varies from 21:00 Saturday to 21:00 Sunday UTC+14:00, 07:00 Saturday to 07:00 Sunday UTC, and 19:00 Friday to 19:00 Saturday UTC−12:00.

```
uint genesisTimestamp = 1545462000; // Saturday 22 December 2018 07:00:00 (UTC),
                                    // pseudonym events scheduled to a random
                                    // hour on the weekend across all time zones

uint pseudonymEvent;

function setRandomHour() internal {
    uint step = eventScheduler.counter % 24;
    uint randomNumber = step + labyrinth.generateRandomNumber() % (24 - step);
    if(eventScheduler.index[step] == 0) eventScheduler.index[step] = step;
    if(eventScheduler.index[randomNumber] == 0) eventScheduler.index[randomNumber]
= randomNumber;
    pseudonymEvent = 28 days + eventScheduler.index[randomNumber] - 20 minutes;
    eventScheduler.index[randomNumber] = eventScheduler.index[step];
}
```

The personhood tokens are mixed, making them untraceable

When the pseudonym event is over and people have been verified, all personhood tokens are mixed, through the entire population. The mixing is simple, people continuously join mixers, incrementally increasing the number of mixers over time as people invoke joinMixer(), in mixers of 4 or so people that use ring signatures, and a personhood token is issued to their new public key, and registration for the next event with another new key (keys from two separate mixers.)

On collusion attacks that simultaneously attack the border

An attack vector I was asked about in Pseudonym Pairs was a combination of collusion attacks with an attack of the “virtual border”. To be exact, collusion attacks can sustain this many bots:

$$\frac{colluders}{population}^2$$

The success-rate follows an inverse square law, so quite low returns.

A simultaneous attack of the border with bots up to *colluders/population* of total population (colluders are able to issue that many border tokens) provides one bot in *colluders/population* of the pairs already controlled, an inverse cube law, in my opinion negligible.

If 10% of the entire population attacks their network, they can sustain bots in 1% of all pairs, and their simultaneous attack of the border will give them an extra bot in 0.1% of all pairs. Since the attack vectors follow an inverse square law and inverse cube law, when 5% of the entire population attack their network, they get bots for 0.25% of all pairs, and an extra bot in 0.0125% of all pairs. Overall, the bots the attackers control increases with *collusion/population/2*, as they get only one extra per pair where the border attacks succeed.

Ideally both attack vectors would be combined. The attack requires a large population that colludes, and is as a heist very low reward.

Borderless personhood tokens for a global population

The Pseudonym Pairs protocol has no way of distinguishing between people, since it treats any human being as equivalent, it cannot shut certain people out. It is borderless in that the protocol cannot know how many people it has counted unless it assumes it is everyone.

References

Pseudonym Parties: An Offline Foundation for Online Accountable Pseudonyms
https://pdos.csail.mit.edu/papers/accountable-pseudonyms-socialnets08.pdf (2008)