FE hackathon

Team name: ZenGo X

Project title: “PPS - stealth addresses based on inner product FE”

Goals:

- Main:
  - Build an efficient crypto system to anonymize payments in existing blockchains
- Secondary:
  - Use simplest FE (contribute to the learning curve of people outside the field)
  - Minimize new security assumptions on top of what is already assumed in blockchain
  - Test and improve GoFE API
  - Solution must be superior to “Naive” stealth address construction

Challenges:

- No centralized authority
- This is an open problem (https://ethresear.ch/t/open-problem-improving-stealth-addresses/7438)

Motivating example:

Edward Snowden wants to collect donations in ETH. He authenticates his ethereum account by tweeting it from his twitter account. Alice believes in Snowden's cause and wants to donate. Her account address is public on the blockchain and can be easily connected to her real world identity. Since ethereum is only pseudo-anonymous, any transaction sent from Alice to Snowden's account will be public on the blockchain forever.

What if we could find a way for Alice to donate to Snowden such that (one of the two holds):

- Anonymity : No third party will be able to tell that Snowden received a donation from Alice (strong claim)
- Plausible deniability: While anyone can observe Alice made “a” donation, no one will be able to distinguish between the case the donation was made to Snowden and the case the donation was made to some other organization (weak claim).

Problem definition :

We have:

n senders **S1,..., Sn,**

m receivers **R1,..., Rm,**

t blockchain blocks **B1,...,Bt**

Si wants to send a payment (blockchain transaction) to receiver **Rj** in block **Bk** without any blockchain observer being able to detect the recipient has been paid.

Naive construction :

- Every recipient **Rj** publishes its public key (== its address) which is an elliptic curve point **Q** to which **Rj** knows the secret key **s** (**Q = s G**).
- Sender **Si** picks a random number **r** and sends the payment to **rQ**
- **Si** communicates **r** to **Rj** in an out-of-band channel
- **Rj** is able to spend the payment using secret key **sr**

Problems with the naive construction:

- Requires out of band communication which we cannot assume (Alice has no secure channel with Snowden)
- Requires the recipient search time linear in **t** (O(t)) . This requires large bandwidth to download all the blocks as **t** can be very large (in our example Snowden online time is limited and he usually connects to the internet via low bandwidth connection and of-course does not run his own full node)

Proposed FE based solution:

We rely on the Simple Functional Encryption Schemes for Inner Products scheme. We take the protocol of section 3: Inner product from DDH. Assuming DDH does not require us to make any new assumptions.

- Setup: Here we need to generate the master public key (**mpk**) and master secret key (**msk**). In the paper they are computed as : **msk = [s_1,...,s_m]** and **mpk = [s_1G, …, s_mG]**[1] . We note that this can be decentralized. For our use case we let each receiver **Ri** pick random **s_i** and publish **H_i = s_i G** . mpk will be the vector with all **H_i**'s.
- KeyDer : In the paper, this is where the centralized authority generates a "inner product" key sk_y = <msk,y>. However, in our case **msk** is decentralized and the best we can hope for is for each party to get only a key of the form sk_yi = [0,0,0..., s_i,...0,0,0] * [0,0,0..., x_i,...0,0,0]T = s_i x_i . Therefore in our scheme KeyDer boils down to each receiver **Rj** picks a random x_j and outputs a secret key sk_yj = s_j x_j[2]
- Send: here we describe how sender **Si** signals receiver **Rj** that a stealth payment was done on block height **Bk**. We rely on the Encrypt definition from the paper scheme.
    - If k = 1 (first block): **Si** samples random r_1 and computes C_1 = r_1G:
        - compute: E_j1 = z_1G + r_1H_j for receiver **Rj** .
        - Compute E_i1 = r_1H_i for receivers **Ri != Rj**
    - If k > 1 : **Si** samples random r_k and computes C_k = r_kG + C_{k-1}:
        - compute: E_jk = z_kG + r_kH_j + E_j{k-1} for receiver **Rj** .
        - Compute E_ik = r_kH_i + E_i{k-1} for receivers **Ri != Rj** .

  For simplicity of our system we take signals z_k = 1 for all k. This choice will become clear in a moment.

---

[1] We use an elliptic curve group and in particular secp256k1 which is used in Bitcoin/Ethereum etc.
[2] We discuss later a scenario where we can leverage more the inner product

- Search: This is a protocol run by each recipient **Rj** periodically when it goes online. We rely on the Decrypt definition from the paper. Assume current block height is **t**
    a. Decrypt: D =x_j E_jt - sk_yj C_jt
    b. Compute v = ec_dlog(d) (easy because the dlog is small).
        - If v=0: stop (no signals)
        - If v>=1 : do a binary search: repeat from step (a) for block t/2 and so on until you find the block height that changed v=0 to v=1

  A more accurate description of the binary search algorithm will be given later.

Completing the solution :

- Our FE system allows us only to signal 0/1 on a specific block height. However, due the additive homomorphism of the simple FE we are able to reduce the search time from linear in the naive solution to logarithmic in **t**.
- We require the sender to additionally encrypt a random number **r** using **Qj,** the receiver pubkey and publish this ciphertext in a smart contract as part of the signaled block
- The payment will be made to address **r Qj**
- Once a receiver identifies a block with a signal, it will know the ciphertext in the smart contract is intended for it and will decrypt it to decipher the stealth address to which the payment was made.
- The property we require here is that the ciphertext in the smart contract will not reveal which public key was used to generate it: we claim this is the common case in public key encryption
- Alternatively to the above, we can take z_k's to be some large random number and let the resulted z_kG to encode **r** and thus get rid of the smart contract ciphertext

Practical considerations:

- There is O(m) complexity for computing all the ciphertexts per one stealth address. This can be significantly reduced in practice as we often care about the anonymity set which is smaller than the total of all possible participants. Let us say that it is enough to hide the receiver identity among 10 recipients (anon-set = 10). We can also think of the plausible deniability case where there are two potential recipients (snowden and non-profit to fight climate change) and there is no way to tell which of the two got the payment.
- While it is clear that if a recipient ciphertext was not updated during a certain block - this recipient cannot be a receiver of a stealth payment in this block, we may assume a more complex network behaviour in practice : decoys (all empty ciphertext updates) can be sent, sender can control some receivers addresses and so on. This let us believe that the construction suggested can be easily turned into a cheap mixer !
- The easiest way to deploy our scheme in practice is via a smart contract that register all N parties (e.g. N = 1000), all play both senders and receivers. This provides an elegant way to keep all data (ciphertexts) etc. on chain and to manage the update from multiple senders (blockchain provides natural order)

- If we do allow for a centralized authority we can leverage the power of the inner product to get additional benefits, for example - auditability or partial audiability for a mixer. We keep this direction as future work
- A malicious sender can corrupt the state for all receivers. It is reasonable to request senders to submit zero knowledge proof for the correctness of their computation (which is a straightforward sigma protocol in the case of simple FE). To save space and computation we can work in a model in which senders and receivers are staking funds in the smart contract and if a receiver claims its state is corrupted it is up to the specific sender (easy to pinpoint who was the sender) to publish the proof. The dispute will be settled by the smart contract slashing the faulty (either receiver or sender) party.

Acknowledgements:

We would like to extend our thanks to :

- Varun Madathil
- István András Seres