Stealth addresses in their most basic form work as follows:

- Every recipient publishes an elliptic curve point P

(they know the corresponding private key p

)

- If I want to send a payment to you, I generate a random value r

, and I send the payment to an address with the public key P * r

(I can generate the address locally)

- I tell you r

offline so you can claim the funds using the private key p * r

This allows paying a recipient without the public learning even that the recipient has been paid (think of the recipient's key P

as being publicly

associated with some real-world identity, eg. via ENS). However, this protocol has the downside that some offline channel is required to learn r

. One option is to encrypt r

with P

and publish this to the blockchain, which solves the problem but it comes at an efficiency cost: the recipient now has to scan through all stealth transactions to learn about any payments that are for them.

Some of the proposals in theabove link can solve this problem by assuming that the recipient knows which sender will send them funds, but this is not always the case; sometimes, the sender may even want to be anonymous.

The challenge is: can we improve these protocols in such a way that reduces the scanning overhead for the recipient? The approach above has ~64 bytes and one elliptic curve multiplication per transaction in the anonymity set; can we do better?

There are information-theoretic arguments that show some limits: there must be at least O(N)

information that the total

set of recipients must listen to, as it must somehow contain for every recipient the information of whether or not they received a transaction. If each sender only modifies O(1)

elements in this information, then the recipients would have to do an O(N)

-sized scan, or else the senders would have to know which portion of the data the recipient is scanning, which can be used to detect transactions that are more likely going to the recipient.

Note that enlisting off-chain actors to perform complicated work (eg. updating the entire dataset after each transaction) to assist the recipients is acceptable.

Here is one fairly inefficient and complicated (but reasonably elementary in the mathematical sense) solution that seems to work:

- Every recipient r

picks a prime d_r

and generates a hidden-order RSA group (ie. a modulus n_r = p_rq_r

) such that the order of 2 is a multiple of d_r

. They publish n_r

. The list of all d_r

values is published through a mixing protocol, so which d_i

corresponds to which n_i

is unknown. (note: this assumes phi-hiding)

- For every recipient, publicly store a state variable S_i

, an integer modulo n_i

, which is initialized to 1.

- To send a bit to some recipient r

, perform the following procedure. Let n_1 ... n_k

be the list of moduli. For every 1 \le j \le k

, compute w = 2 ^ {d_1 * d_2 * ... * d_{j-1} * d_{j+1} * ... * d_k}

(note that d_j

is skipped). Set S_i \leftarrow S_i * w

.

- Note that the discrete log base 2 of S_r

modulo d_r

changes, but the discrete log base 2 or S_t

modulo d_t

for t \ne r

does not change. Hence, only recipient r

gets a message and other recipients do not, but no one other than r

learns that r

got a message. Recipient r

can check for messages at any time by computing the discrete log using the factors (p_r, q_r)

as a trapdoor.

This does involve a lot

of work, though it could be outsourced and incentivized and verified via some ZK-SNARK-based plasma scheme (the dataset would be stored off-chain to avoid huge on-chain gas costs for rewriting it).

Another alternative is to use fully homomorphic encryption to outsource checking: use FHE to allow some third party to walk through the entire set of encrypted values, decrypt each value, and return the values that decrypt correctly. Given FHE as a “black box protocol” this is conceptually simpler; it also means that complicated constructions to store and maintain data off-chain are not necessary. However, the FHE approach may be more expensive.

Edit: this problem feels very conceptually similar to PIR (either the access variant or the search variant) but not quite the same; the main difference is that each recipient must only be able to read their own values.