

Let's say someone builds a donation/crowdfunding contract on Aztec and we only allow 64 nullifiers/tx.

If 1000 people donate to this contract, the contract has 1000 notes. When the "operator" wants to withdraw the money, they have to make  $\text{ceil}(1000/64) = 16$

transactions (since you can only nullify 64 notes at a time). The more aggressive the kernel limits, the more transactions would be required. This can be a cumbersome and time-consuming process.

So what are the options? How would contracts on Aztec operate at scale?

1. Use partial notes combined with homomorphic properties of Pederson commitments
2. Use a linear homomorphic encryption instead of our note abstractions

## Using Pederson commitments

I commit to donating a value X. Store the Pederson commitment publicly onchain. As others donate, the contract homomorphically adds these commitments too. At withdrawal time, individual donors would need to reveal their commitments (amount, random value used for commitment) for the operator to withdraw the actual amount.

Implementation:

1. Using Partial notes - Every new donor adds their commitment. At the end, the operator comes to "complete" this note
2. Not using notes/nullifiers at all - this requires fundamental changes to the token contract:

```
fn transfer_private(amount, to_key) { // increase to balance` curr_balance_hash = storage.balance(to_key).read() if
curr_balance_hash == 0 { // first time transferring new_balance_hash = amount * GENERATOR_AMT + rand *
GENERATOR_RAND + domain_seperator * GENERATOR_DOMAIN_SEP } else { // add to existing balance homomorphically.
new_balance_hash = curr_balance_hash + amountX + GENERATOR_AMT_X; }
```

```
storage.balance(to_key).write(new_balance_hash)
```

```
// reduce `from` balance
```

```
...
```

```
}
```

Pro:

- Just creates one note at the end (at withdraw time).

Con:

- Requires donors to interact back to reveal their amounts. Also, if one participant doesn't reveal, then the whole scheme breaks. While this might not work for the donation case, there might be other applications where this is an interesting idea
- of course, the donors could reveal at donation time, but this leaks the amount donated (although preserves user's address)
- of course, the donors could reveal at donation time, but this leaks the amount donated (although preserves user's address)
- Changes to token contract.

## Use Linear Homomorphic Encryption (LHE)

Commitments become a problem when multiple parties are involved since each has to be individually revealed. It might not work for some applications and it leaks privacy. One idea is to use encryption instead of commit-reveal schemes.

Let,

$pk_K$

= key of the donation contract

$x$

,  $y$

,  $\rightarrow$  individual donation amounts

Then,

let  $a = \text{Encrypt}(pk_K, x)$

let  $b = \text{Encrypt}(pk_K, y)$

Define  $f(a,b)$

such that  $\text{Decrypt}(f(a,b)) = x+y$

Regev, LWE or Elgamal seem to be nice schemes for this. There is an implementation of elgamal in Noir but it uses babyjubjub curve (which [@joshc](#) has used).

Pro:

- Privacy

Con:

- Potentially Slow
- Complex
- Similar changes to token contract as in the previous scheme

## Next steps:

1. Are there any other ideas/better schemes to explore?
2. Various aztec community members have asked for an implementation of Elgamal on Grumpkin. Is this worth exploring?
3. What other applications would be impacted by this? Specifically, which ones would be okay with the pederson commitment approach?

Note: the real answer to this problem may be to enforce limits on who can participate. E.g. instead of 1000 people donating 0.1 eth, what if you limit 10 people to donate 10 ETH each! But it is still useful to think about more technical solutions to bypass this problem.

This post is an amalgamation of various ideas/convos with different people. H/t [@Mike](#) [@kashbri](#) [@Maddiaa](#) (for his LWE-Auction work) [@cooper-aztecLabs](#) for all being fun to work with.

## DISCLAIMER

The information set out herein is for discussion purposes only and does not represent any binding indication or commitment by Aztec Labs and its employees to take any action whatsoever, including relating to the structure and/or any potential operation of the Aztec protocol or the protocol roadmap. In particular: (i) nothing in these posts is intended to create any contractual or other form of legal relationship with Aztec Labs or third parties who engage with such posts (including, without limitation, by submitting a proposal or responding to posts), (ii) by engaging with any post, the relevant persons are consenting to Aztec Labs' use and publication of such engagement and related information on an open-source basis (and agree that Aztec Labs will not treat such engagement and related information as confidential), and (iii) Aztec Labs is not under any duty to consider any or all engagements, and that consideration of such engagements and any decision to award grants or other rewards for any such engagement is entirely at Aztec Labs' sole discretion. Please do not rely on any information on this forum for any purpose - the development, release, and timing of any products, features or functionality remains subject to change and is currently entirely hypothetical. Nothing on this forum should be treated as an offer to sell any security or any other asset by Aztec Labs or its affiliates, and you should not rely on any forum posts or content for advice of any kind, including legal, investment, financial, tax or other professional advice.