IEEE International Conference on Blockchain and Cryptocurrency
2–5 May 2022 // Virtual Conference

Bridging Sapling: Private Cross-Chain Transfers

Aleixo Sanchez, Alistair Stewart and Fatemeh Shirazi

ETH zürich

web3 foundation

Motivation

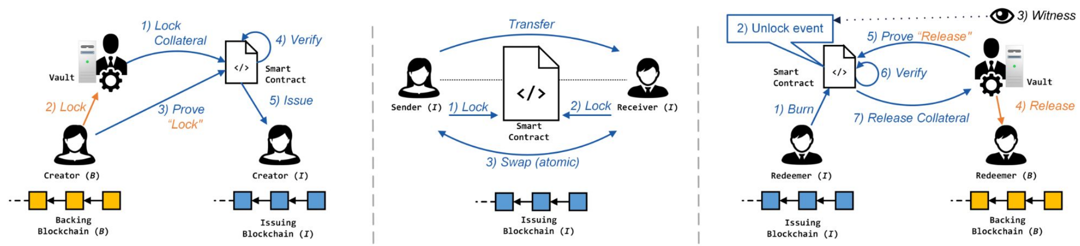
- Mounting interest in privacy features in blockchain
  NIZK, non-interactive zero knowledge
  - Zcash: strong privacy guarantees thanks to zk-SNARKs

- Users can only durably rely on the privacy of a protocol by remaining confined to it
  -> exchange private tokens for transparent, then onwards

- Objective: design a protocol that enables cross-chain transfers, assuming one specification across chains

Background
XCLAIM

- Asset migration ("wrapped" assets)
- Smart-contract capable issuing chain ↔ backing chain
- Trustless intermediaries: collateralised "vaults"
- Cross-chain state verification
- Auditable
  → private transactions?



Issue Transfer / Swap Redeem

A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, “XCLAIM: Trustless, interoperable, cryptocurrency-backed assets,” in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 03 2019, pp. 193–210.

Background

Zcash/Sapling

- "Shielded payment scheme" on top of Bitcoin
- Leverages zk-SNARKS to provide strong privacy guarantees
- Transparent TXs + **shielded** transfers
- Transparent value pool <-> shielded value pool

Transaction model:

- Bitcoin's UTXOs -> notes
- Shielded **Spend** & **Output** transfers
- **Note commitment tree**: incremental Merkle tree
- **Nullifier set**: one nullifier associated with each note, published when spent

Background

Zcash/Sapling

Spend transfers allow one to prove that:

- they know a note with note commitment in the note commitment tree
- the revealed value commitment was derived from the value in the note
- they know private key to receiving address
- the revealed nullifier is computed correctly

Background
Zcash/Sapling

As for Output transfers:

- the created note is valid
- value commitment is derived from the value in the note
- the derived note commitment is computed correctly

Background

Zcash/Sapling

Soundness of transferred amounts:

- **Value commitment** in both Spend and Output transfers:
  Homomorphic Pedersen commitments
- **Binding signature:**

$$\mathsf{bvk} := \left(\bigoplus_{i=1}^{n} \mathsf{cv}_i^{\mathsf{old}}\right) \ominus \left(\bigoplus_{j=1}^{m} \mathsf{cv}_j^{\mathsf{new}}\right) \ominus \mathsf{ValueCommit}_0\left(\mathsf{v}^{\mathsf{balance}}\right).$$

$$\mathsf{bsk} := \left(\boxplus_{i=1}^{n} \mathsf{rcv}_i^{\mathsf{old}}\right) \boxminus \left(\boxplus_{j=1}^{m} \mathsf{rcv}_j^{\mathsf{new}}\right).$$

ZCLAIM

Setup

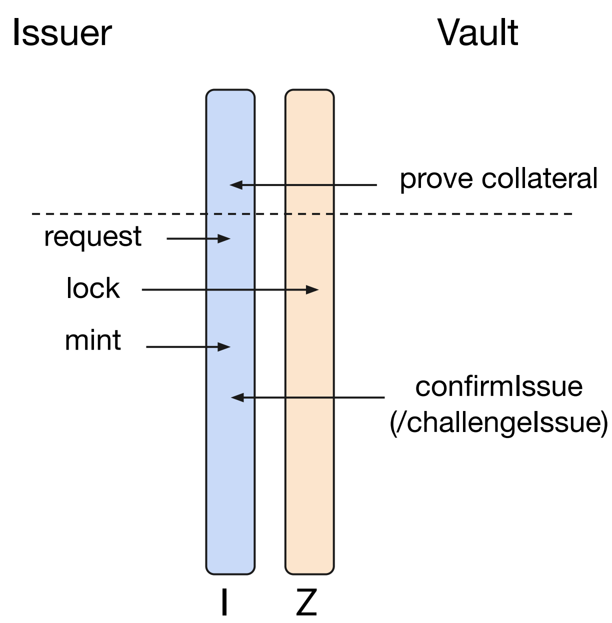
Chains:

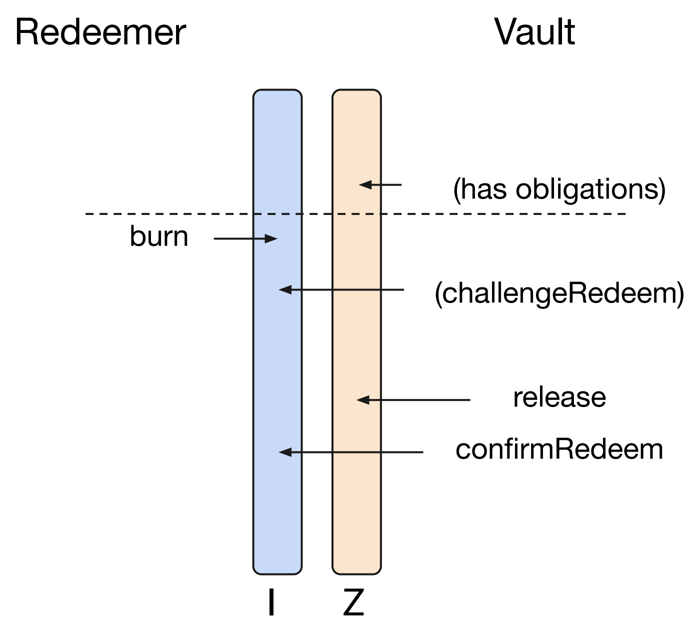
- Backing chain: Zcash (only shielded scheme)
- Issuing chain: must support Zcash crypto, we assume **Sapling** implemented

Actors:

- Vaults:
    - decentralised custodians
    - lock collateral in issuing chain's currency
    - earn fees from issuing/redeeming
- Issuers: anyone minting wrapped (shielded!) ZEC on issuing chain
- Redeemers: anyone redeeming wrapped ZEC for ZEC

ZCLAIM
Issuing & Redeeming





ZCLAIM

Mint transfers

Mint transfer: "modified Spend transfer". Issuer:

- includes Zcash note commitment and Merkle proof in transfer
- proves in ZK that:
  - note is addressed to vault
  - commitments to locked value v and issued value (=v-fees) are correct
  - note commitment trapdoor was derived from **nonce** → prevents replay attacks
- publishes C^{enc} (note plaintext encrypted to vault)

→ Vault decrypts C, publishes **DH shared secret** if wrong (along with SNARK proving its validity)

→ Improved "happy path" efficiency

ZCLAIM

Burn transfers

Burn transfer: "modified Output transfer". Redeemer:

- Requests vault to create note with note commitment cm
- proves in zk-SNARK:
  - note with note commitment cm is valid
  - commitments to burned value v and value to be redeemed in note ($=v$-fees) are correct
- encrypts note plaintext C to vault

→ Vault decrypts C, publishes **DH shared secret** if wrong (along with zk-SNARK proving its validity)

→ Vault creates note with note commitment cm on Zcash and submits Merkle proof to issuing chain

ZCLAIM
Collateralisation

ZEC locked with a vault is hidden - how do we ensure proper collateralisation?

ZEC may have been "reused" to issue funds

- this is fine
- ***ZEC obligations***: amount of wrapped ZEC issued - redeemed through vault
- collateralisation ratio: collateral/ZEC obligations

→ Vaults proactively prove collateralisation ratio above σ_{std}

- every time they want to **accept lock transactions**
- before **exchange rate** drops by x%

  → otherwise, **liquidation** - users can burn wZEC in exchange for the vault's collateral at a discount

Splitting Strategy

Vault learns amount locked with them

$\longrightarrow$ split total t among several vaults

- Choose individual amounts s.t. if vault learns v

$$\Pr[T = t \mid V = v] \approx \Pr[T = t]$$

Limitations & Future Work

- Limited use cases outside of privacy
- Splitting strategy requires a considerable number of transaction -> not realistic on every chain
- Strict availability requirements on vaults

Beyond:

- Protocol can be adapted to any implementation of Sapling
- Adaptation to newer versions in the works
- Extension to multi-asset shielded pools

Thank you!

Contact: aleixo@web3.foundation

github.com/alxs/zclaim