Shielded CSV

Private and Efficient Client-Side Validation

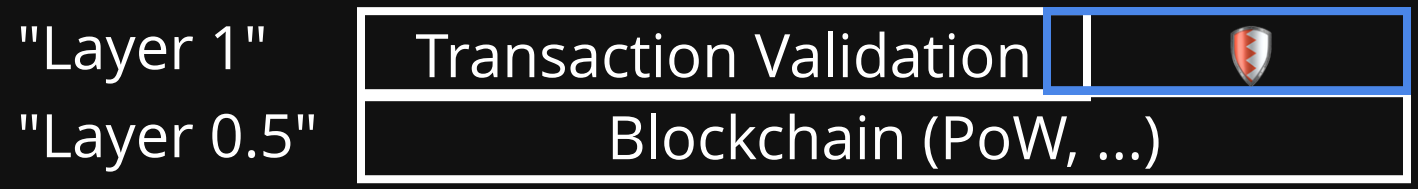
shieldedcsv.org

Jonas Nick — Blockstream

Liam Eagen — Alpen

Robin Linus — ZeroSync

Summary



- Shielded CSV is a transaction protocol to **create an L1 on top of a blockchain** that allows embedding arbitrary data (e.g., Bitcoin).
- It **inherits double-spend security** from the underlying blockchain.
- The amount of data embedded into the blockchain is **64 bytes**.
- Coins and coin proofs are sent directly to the receiver through a **private one-way communication channel**.
- Coin proofs are **succinct**.
- Shielded CSV is fully **private**.

Motivation

1. A more efficient design for **private cryptocurrencies**
    - ZK proofs are **not on the blockchain** and are **not verified** by full nodes. Can be built on existing blockchain.
2. Improve privacy of **Bitcoin**.
    - Use Bitcoin as the underlying blockchain. **Requires bridge** (BitVM, one-way peg, federated peg, ...).

Toy CSV: 🐸 Issuance

Transaction 1

On-Chain

Off-Chain

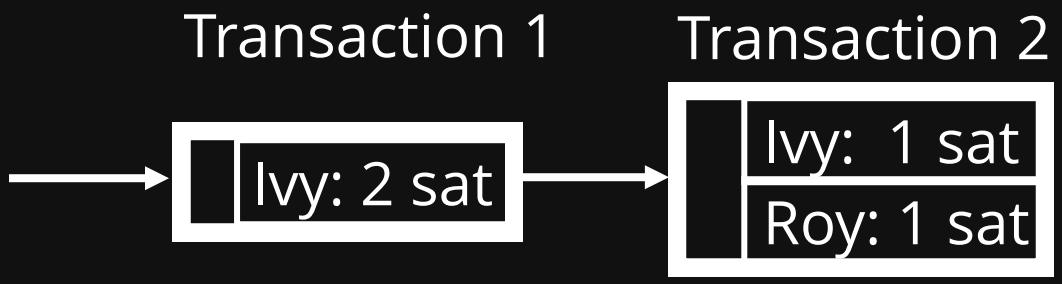
Sign("I issue 10 🐸-coins in transaction 1, output 1. Redeemable for physical frogs.")

Ivy

the Issuer

Transaction 1

Toy CSV: Transaction

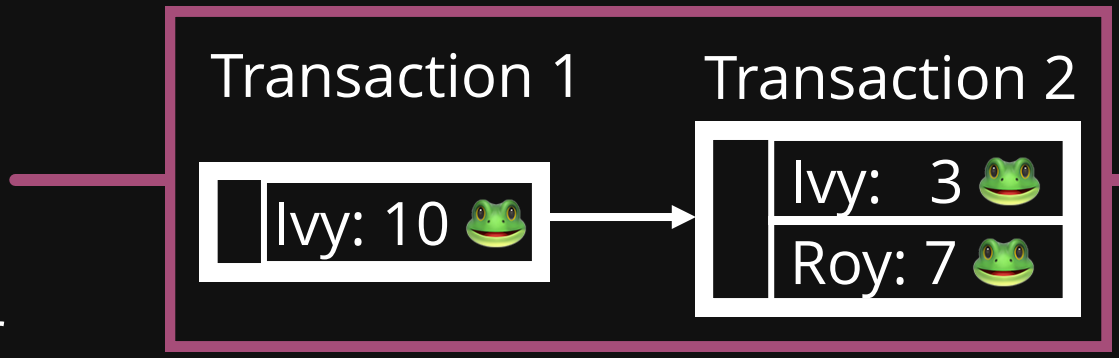


On-Chain

Off-Chain

"Coin Proof":

Ivy
the Issuer



Roy
the Receiver

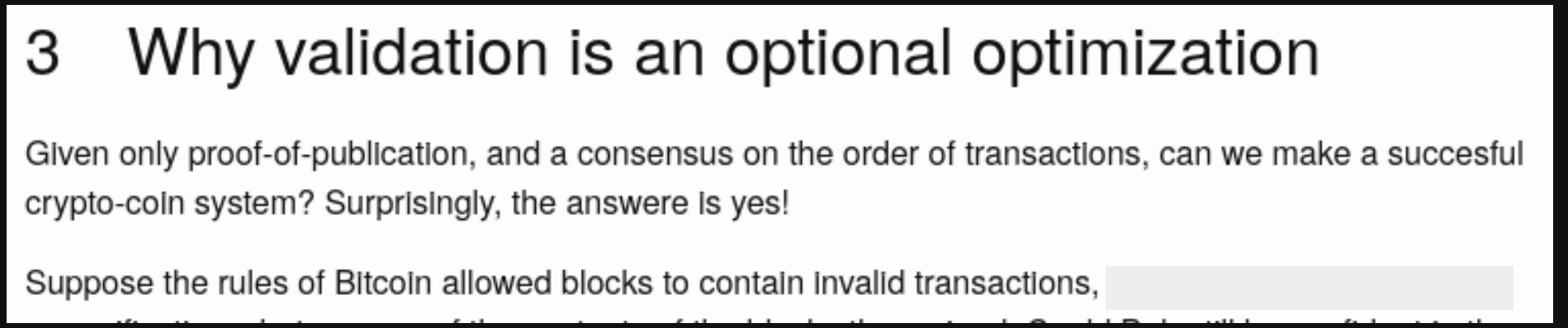
Transaction graph connecting Roy's output
to an issuance transaction

Comparing CSV

Privacy	Coin Proof Size	Blockchain Space
RGB		
Taproot Assets		
Intmax2		
Shielded CSV		

What is CSV really?

Insight: Transaction validation does not need to be part of the consensus rules.



3 Why validation is an optional optimization

Given only proof-of-publication, and a consensus on the order of transactions, can we make a succesful crypto-coin system? Surprisingly, the answere is yes!

Suppose the rules of Bitcoin allowed blocks to contain invalid transactions,

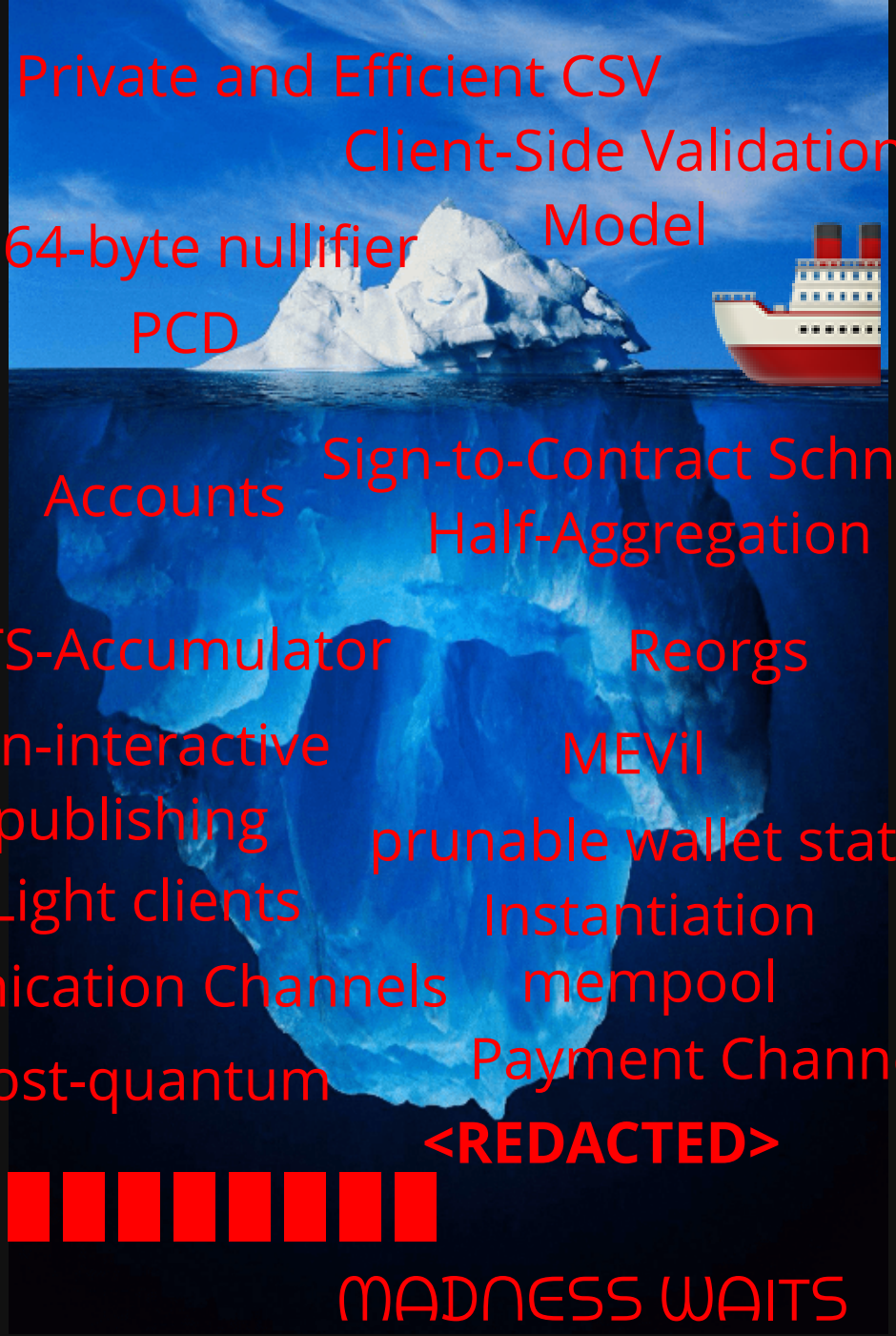
petertodd.org, 2013

... then transactions are only validated "client-side" and simply ignored if they are invalid.

Taking CSV seriously

1. Derive **short** piece of data ("*nullifier*") from the CSV transaction.
2. Post nullifier to blockchain to prevent **double-spending.**

- Toy CSV: nullifier is **Bitcoin tx**
- Shielded CSV: nullifier is **64-byte** blob


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Sign-to-Contract Schnorr
Accounts
Half-Aggregation
TS-Accumulator
Reorgs
non-interactive
MEVil
publishing
prunable wallet state
Light clients
Instantiation
Communication Channels
mempool
Payment Channels
Post-quantum
<REDACTED>
MADNESS WAITS

Shielded CSV: Definitions

- **CSV Transaction**: similar to Bitcoin transactions. Consist of inputs and outputs.

- **Coin**: Tx output. Consists of amount & public key.
- **CoinID:** Tx hash & coin index. Tx inputs contain CoinIDs.
- **Coin Proof**: History of transactions connecting to issuance transactions.

Sally the Sender

Coin, Coin Proof

Roy the Receiver

Verify:
All txs in the coin proof are valid.

Preventing Double Spending

```
Nullifier := (CoinID, TxHash)
```

Embed nullifier (<some CoinID>, <some TxHash>)

IGNORED by Roy

Blockchain

Process nullifiers

CoinID	TxHash
<some CoinID>	<...>
<other CoinID>	<...>

nullifier key-value store

Process nullifiers:

Ignore nullifiers whose CoinID is already in the KV-store.

Preventing Double Spending

Sally the Sender

Coin, Coin Proof

Roy the Receiver

CoinID	TxHash
<some CoinID>	<...>
<other CoinID>	<...>

nullifier key-value store

Verify Coin Proof:

1. All txs in the coin proof are valid.
2. Every coin spent in the coin proof must be present in the KV-store and the tx hashes must match.

Preventing Double Spending

This design does prevent double spending with only a small nullifier!

- **Problem:** Insecure! Anyone can nullify any coin.
  - **Solution:** Add signature to nullifier.
- **Inefficiency:** One nullifier per spent coin.
  - **Solution:** Introduce "accounts", a special type of TXO.
  - **Result:** one nullifier per account state update that can spend an arbitrary number of coins.
- **Problem**: Posting nullifiers requires a dedicated on-chain tx.
  - **Solution**: *Publishers* collect nullifiers and post them all at once with a single on-chain tx. Anyone can become a publisher.

Towards 64-bytes Nullifiers

```
Nullifier := (CoinID, TxHash
```

Accounts

(insecure, one nullifier per coin)

```
Nullifier := (Nullifier_PubKey, TxHasl
```

Signature

(insecure, one nullifier per tx)

```
Nullifier := (Nullifier_PubKey, TxHash, Signature
```

Sign-To-Contract

128 bytes

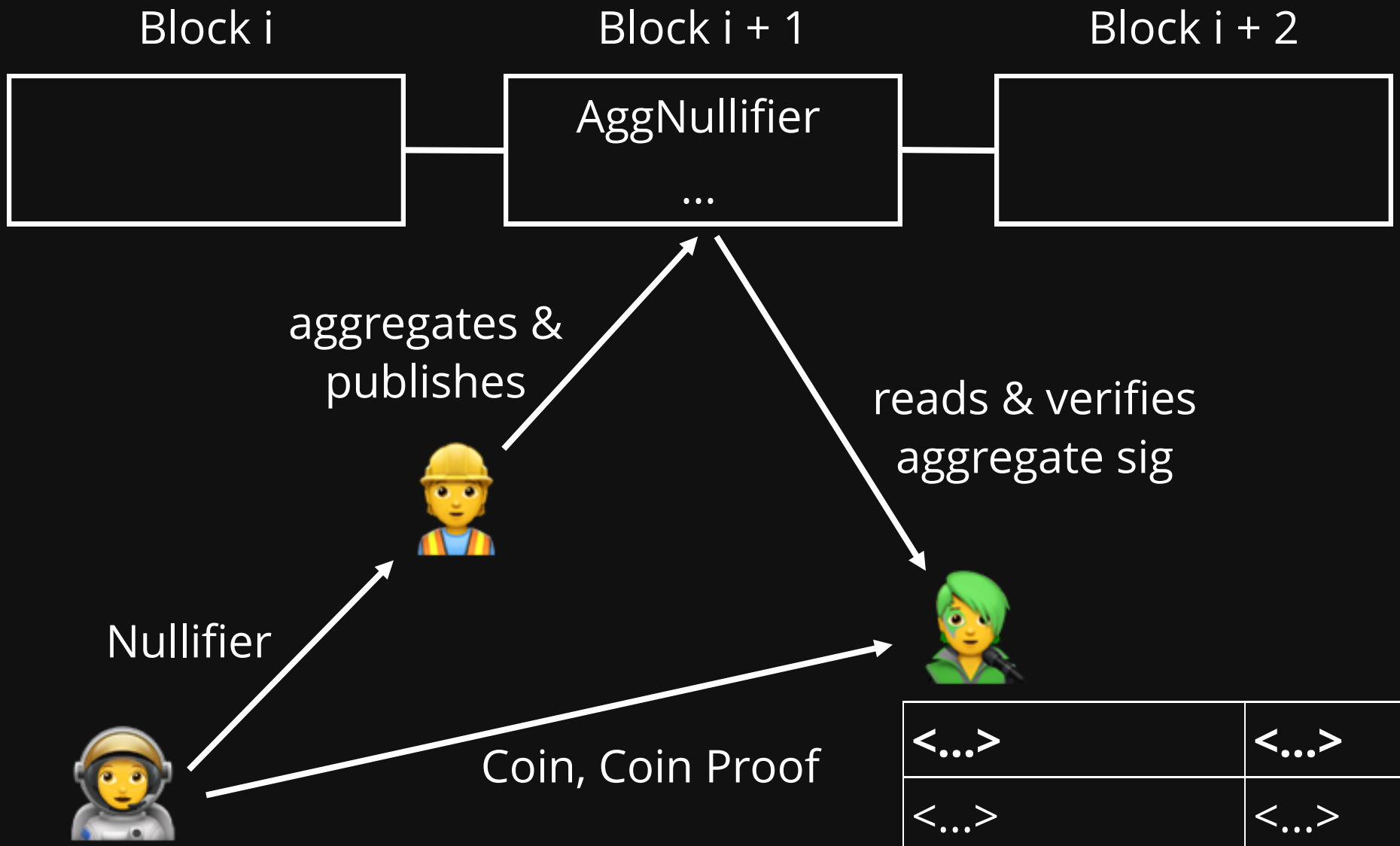
```
Nullifier := (Nullifier_PubKey, Signature
```

Signature Half-Aggregation

96 bytes

```
AggNullifier := (Nullifier_PubKeys, AggSi
```

64 bytes

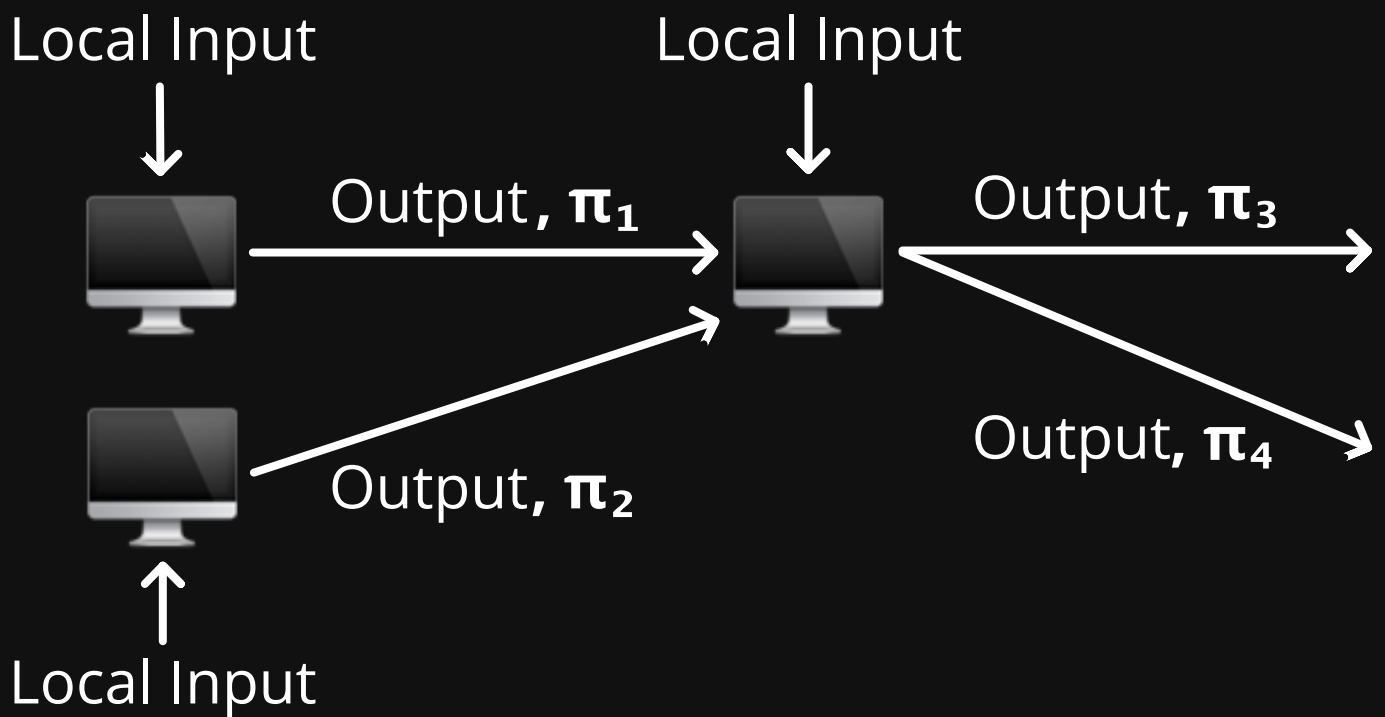



```
Tx := (AcctState, Coins,
       NewAcctState, NewCoins)
```

Succinct & Private Coin Proofs

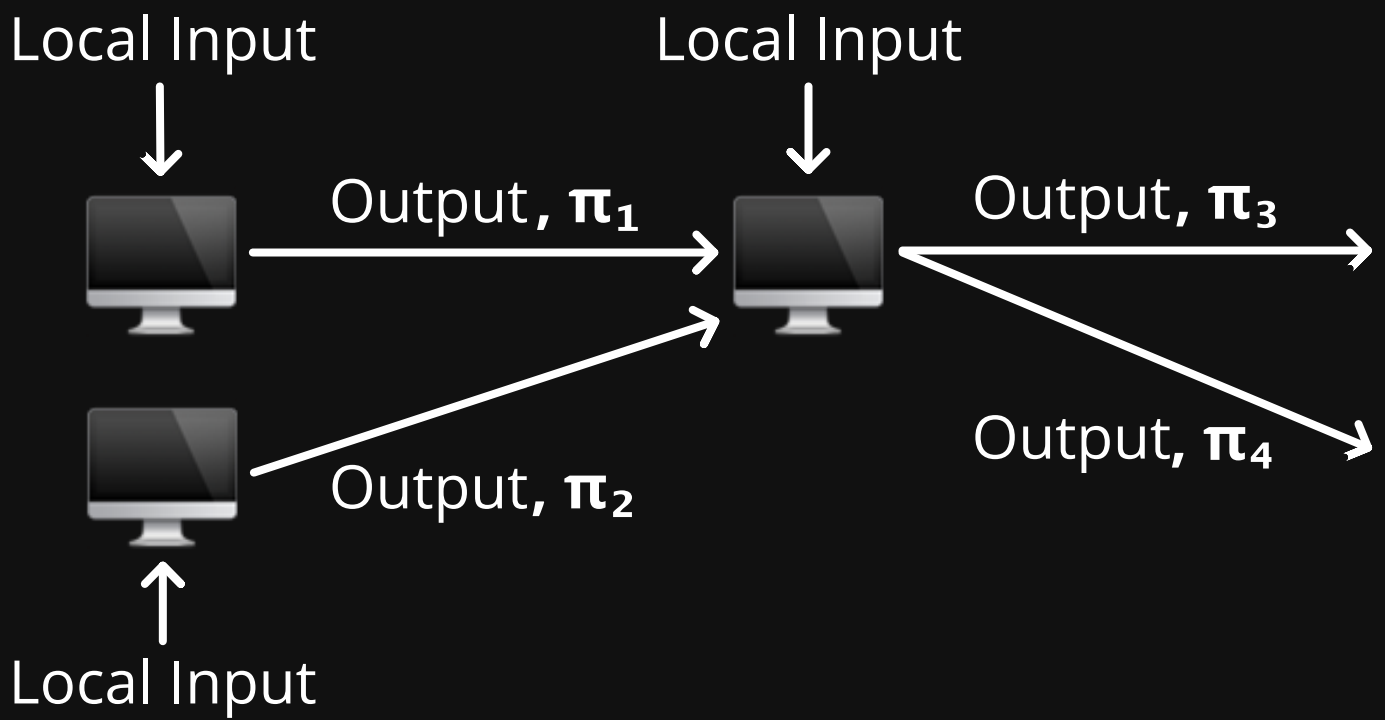
- **Problem**: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?)
- **Solution**: Wrap the protocol in a "Proof-Carrying Data" (PCD) scheme

Proof-Carrying Data (PCD) [2010]



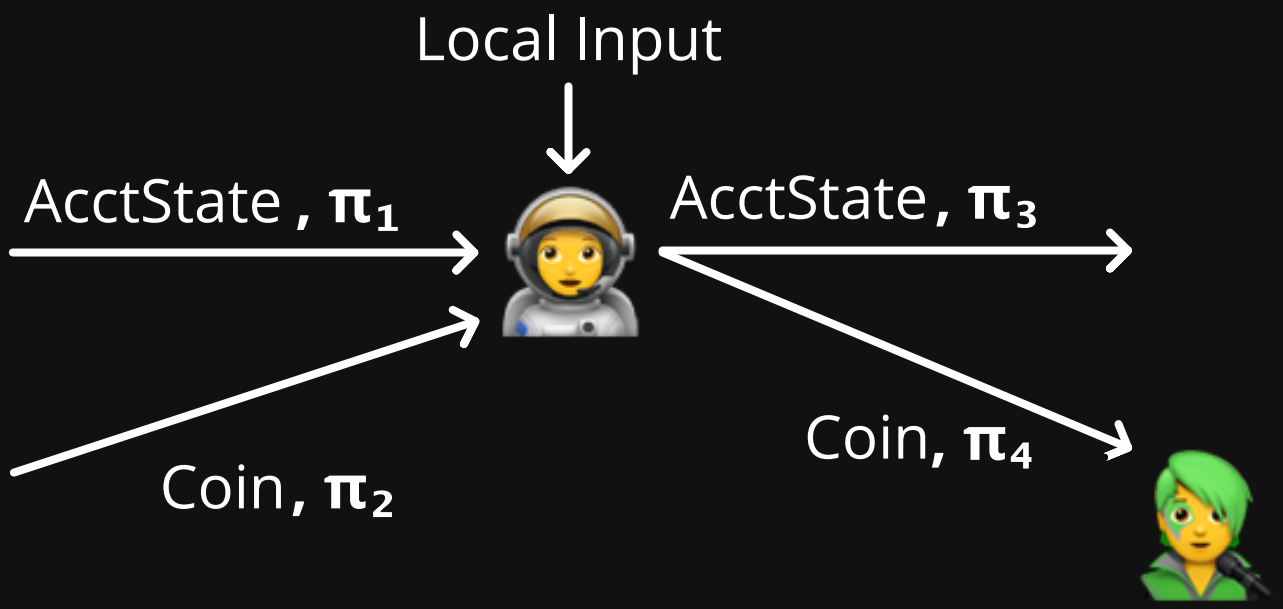
- π_i: proof that the **entire** preceding computation graph is correct.
  - Size and Verification time **independent** of graph size.
  - **Zero-Knowledge** for incoming inputs and and outputs.
- PCD can be instantiated with **recursive SNARKs** or **Folding schemes**.

Proof-Carrying Data (PCD) [2010]



- 🖥️: Shielded CSV transaction
- **Output**: Account state or coin
- **Local Input**: Account update proofs, etc
- **π_i:** Coin proof

Shielded CSV Coin Proofs



- Coin proof π_4 proves to Roy that all transactions are correct and have been nullified
  - succinct and ZK
- Why PCD? Framework abstracts away some of the complexity.

Shielded CSV is ...

- an instantiation of **PCD,**
- the definition of **correct computation in PCD,**
  - we specify this in Rust,
- a spec for how the **nullifier key-value store** is updated.

This was just a tiny glimpse of the paper

- Blockchain reorganizations
- "Trustless" Publishing & Fees
- Security definitions of the primitives (accumulators, etc.)
- t-of-n Shared Accounts
- Atomic Swaps
- Wallet State
- Nullifier Accumulator based on Merkle Tree of Merkle Trees
- ...

Future Work

- More complete specification & test vectors
- Instantiation of primitives
- BitVM (?) Bridging
- Communication channels / UX
- Other drawbacks of CSV paradigm?
- "mempool"
- Efficient timelocks
- Payment channels
- Light clients
- Scriptable spending policies
- ...

shieldedcsv.org

Shielded CSV: Private and Efficient Client-Side Validation

Jonas Nick[1], Liam Eagen[2], and Robin Linus[3]

[1]Blockstream
[2]Alpen Labs
[3]ZeroSync

September 20, 2024

Abstract

Cryptocurrencies allow mutually distrusting users to transact monetary value over the internet without relying on a trusted third party.

Bitcoin, the first cryptocurrency, achieved this through a novel protocol used to establish consensus about an ordered transaction history. This requires every transaction to be broadcasted and verified by the network, incurring communication and computational costs. Furthermore, transactions are visible to all nodes of the network, eroding privacy, and are recorded permanently, contributing to increasing storage requirements over time. To limit resource usage of the network, Bitcoin currently supports an average of 11 transactions per second.

Most cryptocurrencies today still operate in a substantially similar manner. Private cryptocurrencies like Zcash and Monero address the privacy issue by replacing transactions with proofs of transaction validity. However, this enhanced privacy comes at the cost of increased communication, storage, and computational requirements.

Client-Side Validation (CSV) is a paradigm that addresses these issues by removing transaction validation from the blockchain consensus rules. This approach allows sending the coin along with a validity proof directly to its recipient, reducing communication, computation and storage cost. CSV protocols deployed on Bitcoin today [1, 2] do not fully leverage the paradigm's potential, as they still necessitate the overhead of publishing ordinary Bitcoin transactions. Moreover, the size of their coin proofs is proportional to the coin's transaction history, and provide limited privacy. A recent improvement is the Intmax2 [3] CSV protocol, which writes significantly less data to the blockchain compared to a blockchain transaction and has succinct coin proofs.

In this work, we introduce Shielded CSV, which improves upon state-of-the-art CSV protocols by providing the first construction that offers truly private transactions. It addresses the issues of traditional private cryptocurrency designs by requiring only 64 bytes of data per transaction, called a *nullifier*, to be written to the blockchain. Moreover, for each nullifier in the blockchain, Shielded CSV users only need to perform a single Schnorr signature verification, while non-users can simply ignore this data. The size and verification cost of coin proofs for Shielded CSV receivers is independent of the transaction history. Thus, one application of Shielded CSV is adding privacy to Bitcoin at a rate of 100 transactions per second, provided there is an adequate bridging mechanism to the blockchain.

We specify Shielded CSV using the Proof Carrying Data (PCD) abstraction. We then discuss two implementation strategies that we believe to be practical, based on Folding Schemes and Recursive STARKs, respectively. Finally, we propose future extensions, demonstrating the power of the PCD abstraction and the extensibility of Shielded CSV. This highlights the significant potential for further improvements to the Shielded CSV framework and protocols built upon it.