Shielded CSV

Private and Efficient Client-Side Validation

shieldedcsv.org

Jonas Nick — Blockstream

Liam Eagen — Alpen

Robin Linus — ZeroSync

Summary

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

	Bitcoin	**Shielded CSV** 🛡️
Privacy	😜	✅

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

	Bitcoin	**Shielded CSV** 🛡️
Privacy	😉	✅
Blockspace per tx	~600B, on average	64B, always

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

	Bitcoin	**Shielded CSV** 🛡️
Privacy	😜	✅
Blockspace per tx	~600B, on average	64B, always

- **SAME** double-spending security as Bitcoin

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

	Bitcoin	**Shielded CSV** 🛡️
Privacy	😜	✅
Blockspace per tx	~600B, on average	64B, always

- **SAME** double-spending security as Bitcoin
- **NO** online requirement

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

	Bitcoin	**Shielded CSV** 🛡️
Privacy	😜	✅
Blockspace per tx	~600B, on average	64B, always

- **SAME** double-spending security as Bitcoin
- **NO** online requirement
- **NO** inbound capacity problem

Summary

Shielded CSV is a transaction protocol on top of Bitcoin.

	Bitcoin	**Shielded CSV** 🛡️
Privacy	😜	✅
Blockspace per tx	~600B, on average	64B, always

- **SAME** double-spending security as Bitcoin
- **NO** online requirement
- **NO** inbound capacity problem
- **NO** softfork required (*)

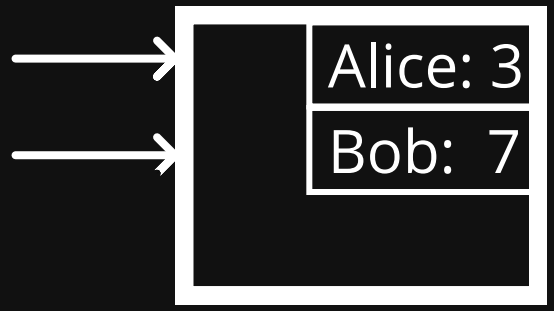
Client-Side Validation (CSV)

Client-Side Validation (CSV)

Shielded CSV  transaction data is embedded in Bitcoin transactions.

Client-Side Validation (CSV)

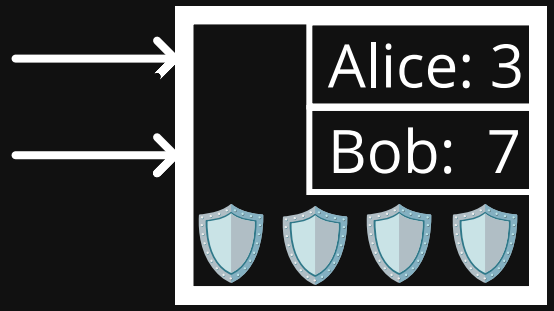
Shielded CSV transaction data is embedded in Bitcoin transactions.



A 2-input, 2-output Bitcoin transaction

Client-Side Validation (CSV)

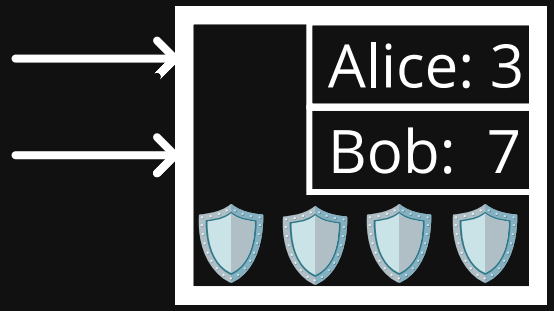
Shielded CSV transaction data is embedded in Bitcoin transactions.



A 2-input, 2-output Bitcoin transaction

Client-Side Validation (CSV)

Shielded CSV 🛡 transaction data is embedded in Bitcoin transactions.

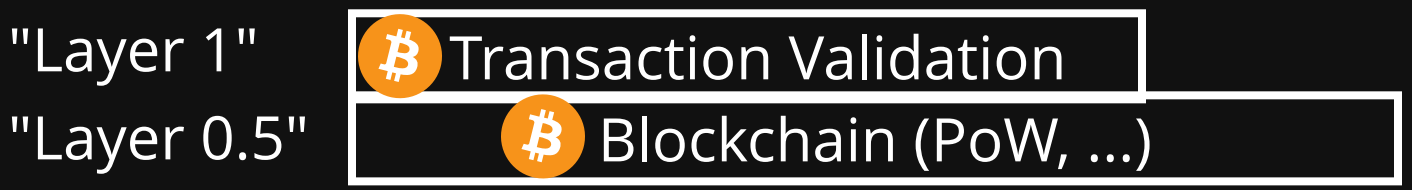


A 2-input, 2-output Bitcoin transaction

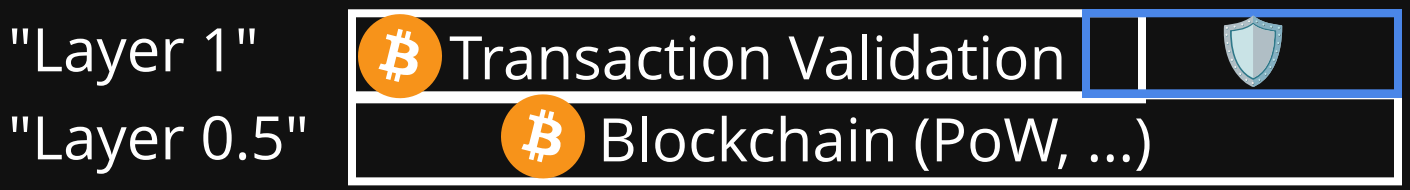
🛡 data is ignored by ordinary Bitcoin nodes, only interpreted "client-side" by 🛡 nodes.

 can be understood as an alternative L1

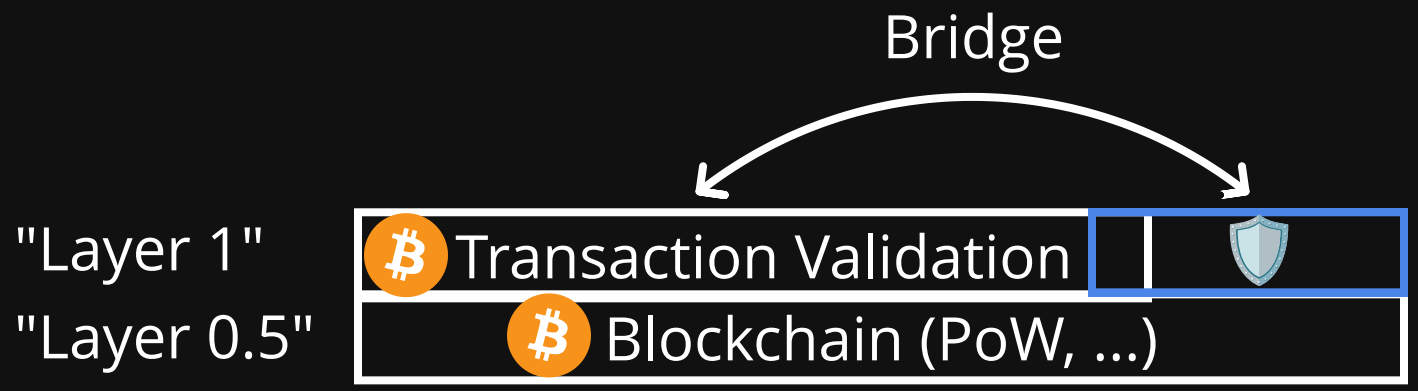
can be understood as an alternative L1



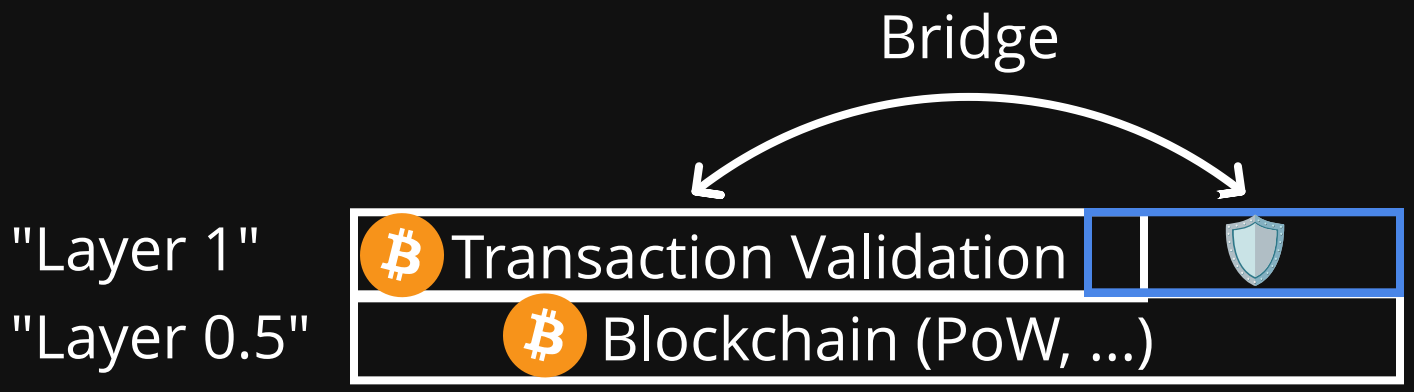
can be understood as an alternative L1



can be understood as an alternative L1



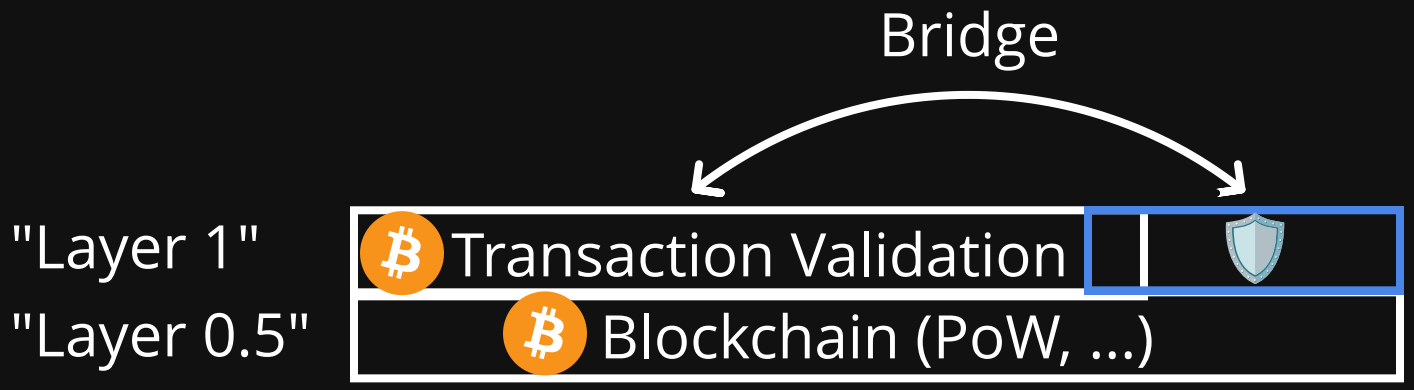
can be understood as an alternative L1



Bridge examples:

- t-of-n federated bridge
- one-way peg
- BitVM
- validity bridge
- ...

can be understood as an alternative L1



Bridge examples:

- t-of-n federated bridge
- one-way peg
- BitVM
- validity bridge
- ...

doesn't need a softfork, but it inherits the security of the bridge (which may be improved by softforks).



BitVM and new Script opcodes only benefit shitcoiners and JPEG degens!!



BitVM and new Script opcodes only benefit shitcoiners and JPEG degens!!

Liar! Shielded CSV proves that BitVM and better Script opcodes can add full privacy and 64 byte transactions to Bitcoin.







Full privacy with only 64 bytes on-chain sounds impossible?



Full privacy with only 64 bytes on-chain sounds impossible?

Shielded CSV Approach



Full privacy with only 64 bytes on-chain sounds impossible?

Shielded CSV Approach

- Only post the **minimal data** necessary to prevent **double-spending** to the blockchain.



Full privacy with only 64 bytes on-chain sounds impossible?

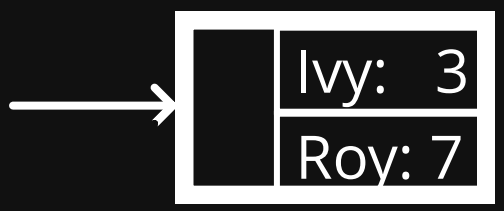
Shielded CSV Approach

- Only post the **minimal data** necessary to prevent **double-spending** to the blockchain.
- Send "the rest of the transaction" to the receiver **directly**.

Basic Shielded CSV 🛡️ recipe

Basic Shielded CSV 🛡️ recipe

1. **Create CSV Transaction**. Consist of inputs and outputs, similar to Bitcoin transactions.



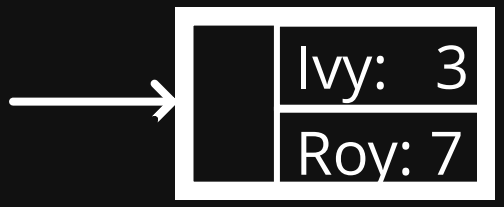
CSV transaction



Sally the Sender

Basic Shielded CSV 🛡️ recipe

1. **Create CSV Transaction**. Consist of inputs and outputs, similar to Bitcoin transactions.
2. **Derive 64-byte "nullifier"** from transaction



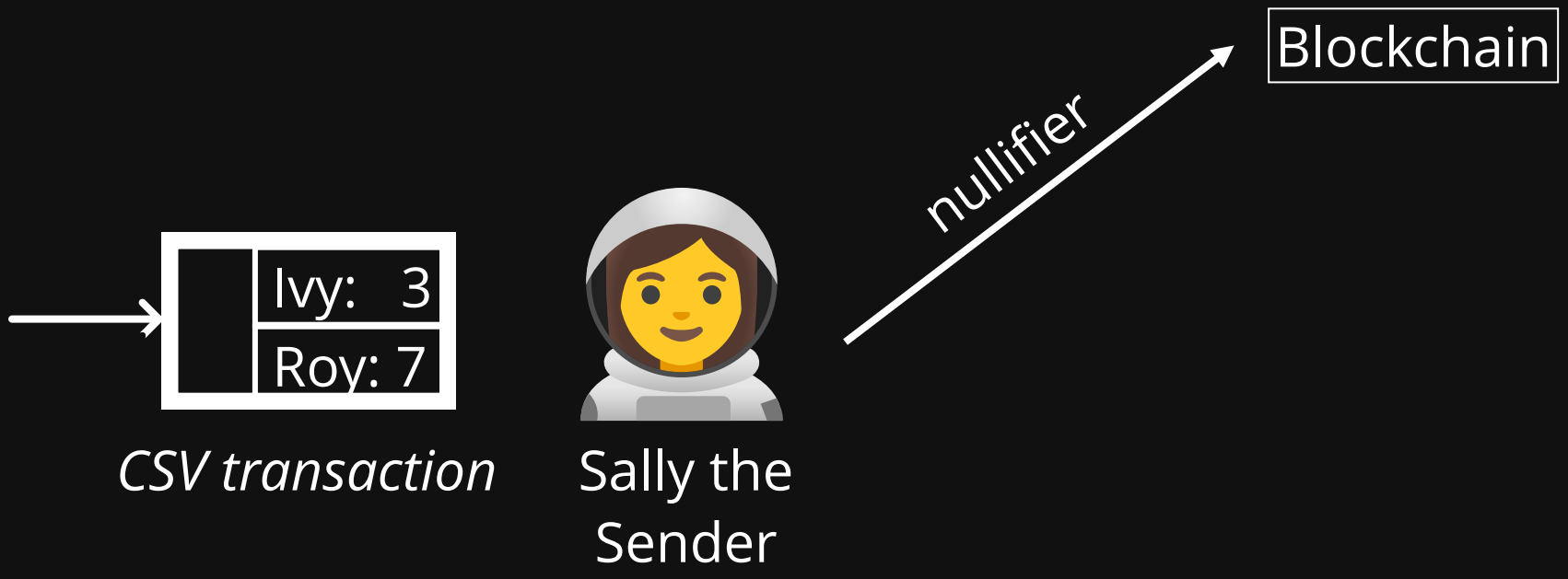
CSV transaction



Sally the Sender

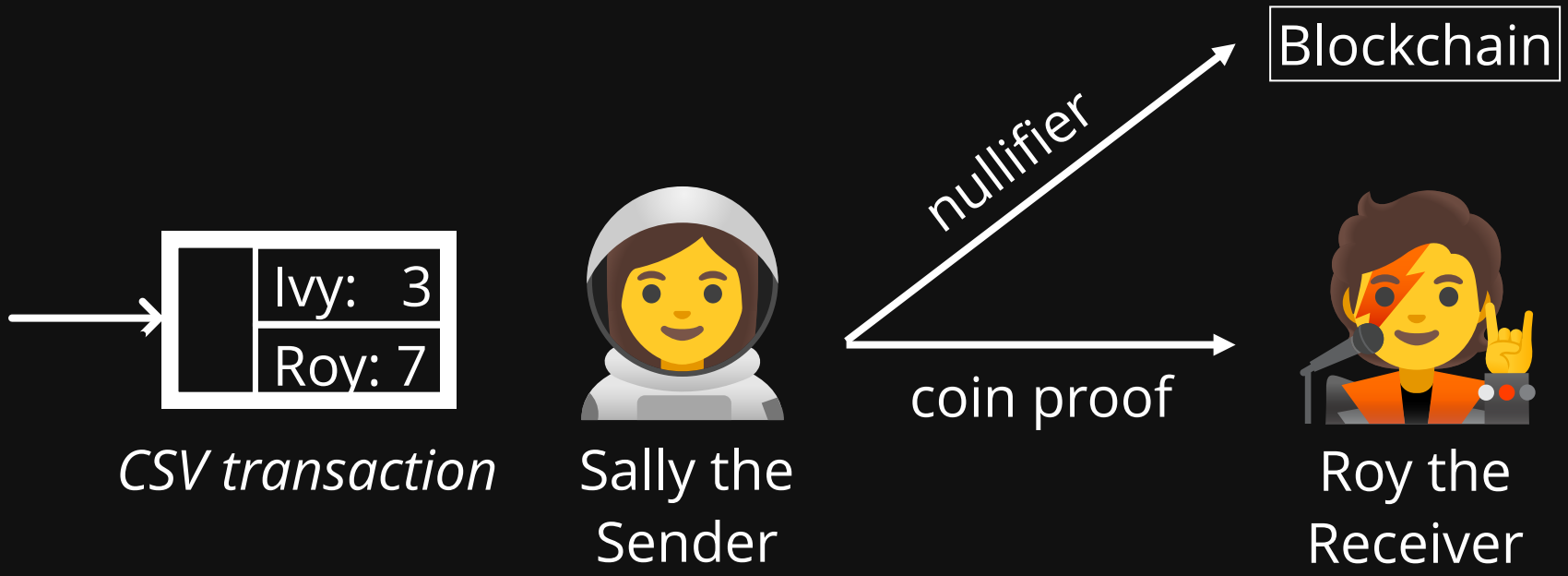
Basic Shielded CSV 🛡️ recipe

1. **Create CSV Transaction**. Consist of inputs and outputs, similar to Bitcoin transactions.
2. **Derive 64-byte "nullifier"** from transaction
3. **Post nullifier** to the blockchain



Basic Shielded CSV 🛡️ recipe

1. **Create CSV Transaction**. Consist of inputs and outputs, similar to Bitcoin transactions.
2. **Derive 64-byte "nullifier"** from transaction
3. **Post nullifier** to the blockchain
4. **Send "coin proof"** to the receiver



Simplified Coin Proof

Simplified Coin Proof

Coin Proof: History of (CSV) transactions connecting to issuance transactions.

Simplified Coin Proof

Coin Proof: History of (CSV) transactions connecting to issuance transactions.



Simplified Coin Proof

Coin Proof: History of (CSV) transactions connecting to issuance transactions.

(The exact details of issuance depend on the type of bridge)



Simplified Coin Proof

Coin Proof: History of (CSV) transactions connecting to issuance transactions.

(The exact details of issuance depend on the type of bridge)





Sally the Sender

coin proof



Roy the Receiver

Simplified Coin Proof

Coin Proof: History of (CSV) transactions connecting to issuance transactions.

(The exact details of issuance depend on the type of bridge)





Sally the Sender

coin proof



Roy the Receiver

Verify:
All txs in the coin proof are valid.

Simplified Nullifier

The nullifier consists of:



Simplified Nullifier



The nullifier consists of:

- **CoinID:** Identifier of the output being spent. This is the hash of the CSV transaction creating the output & index in the transaction.

Simplified Nullifier



The nullifier consists of:

- **CoinID:** Identifier of the output being spent. This is the hash of the CSV transaction creating the output & index in the transaction.
- **TxHash:** Hash of the CSV transaction being created.

Simplified Nullifier



The nullifier consists of:

- **CoinID:** Identifier of the output being spent. This is the hash of the CSV transaction creating the output & index in the transaction.
- **TxHash:** Hash of the CSV transaction being created.

> *The nullifier nullifies the coin, thereby preventing double spending*

Preventing Double Spending



nullifier = (CoinID, TxHash)

Preventing Double Spending



nullifier = (CoinID, TxHash)

Embed nullifier

Blockchain

Preventing Double Spending



nullifier = (CoinID, TxHash)

Embed nullifier

Blockchain

Process nullifiers



Preventing Double Spending



nullifier = (CoinID, TxHash)

Embed nullifier

Blockchain

Process nullifiers



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

nullifier key-value store

Preventing Double Spending



nullifier = (CoinID, TxHash)

Embed nullifier

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



nullifier = (CoinID, TxHash)

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

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



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

Preventing Double Spending



Sally the Sender



Coin, Coin Proof



Roy the Receiver

Preventing Double Spending



Sally the Sender



Coin, Coin Proof



Roy the Receiver

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

nullifier key-value 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.

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!

Preventing Double Spending

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

(what we've seen is not the actual Shielded CSV nullifier, because it has a number of problems)

Publishing Nullifiers

Publishing Nullifiers

Problem: Sender shouldn't have to make a Bitcoin tx for every nullifier (overhead, requires having bitcoin).

Publishing Nullifiers

Problem: Sender shouldn't have to make a Bitcoin tx for every nullifier (overhead, requires having bitcoin).



Publishing Nullifiers

Problem: Sender shouldn't have to make a Bitcoin tx for every nullifier (overhead, requires having bitcoin).



- Publisher is a permissionless role, open to anyone able to make a Bitcoin transaction.

Publishing Nullifiers

Problem: Sender shouldn't have to make a Bitcoin tx for every nullifier (overhead, requires having bitcoin).



- Publisher is a permissionless role, open to anyone able to make a Bitcoin transaction.
- If the nullifier ends up in the blockchain, the publisher reponsible is paid a fee within Shielded CSV.

Block i



Block i

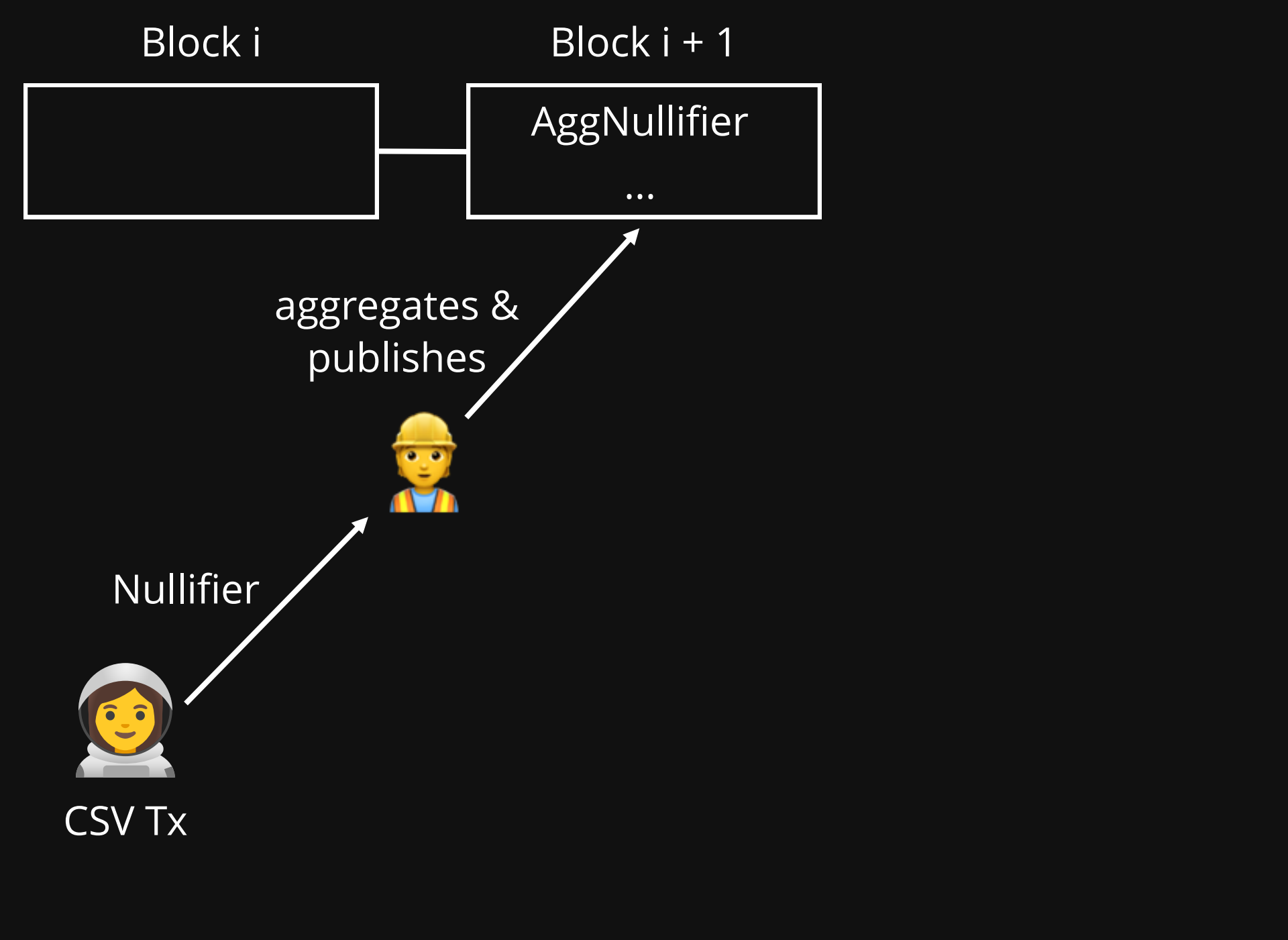


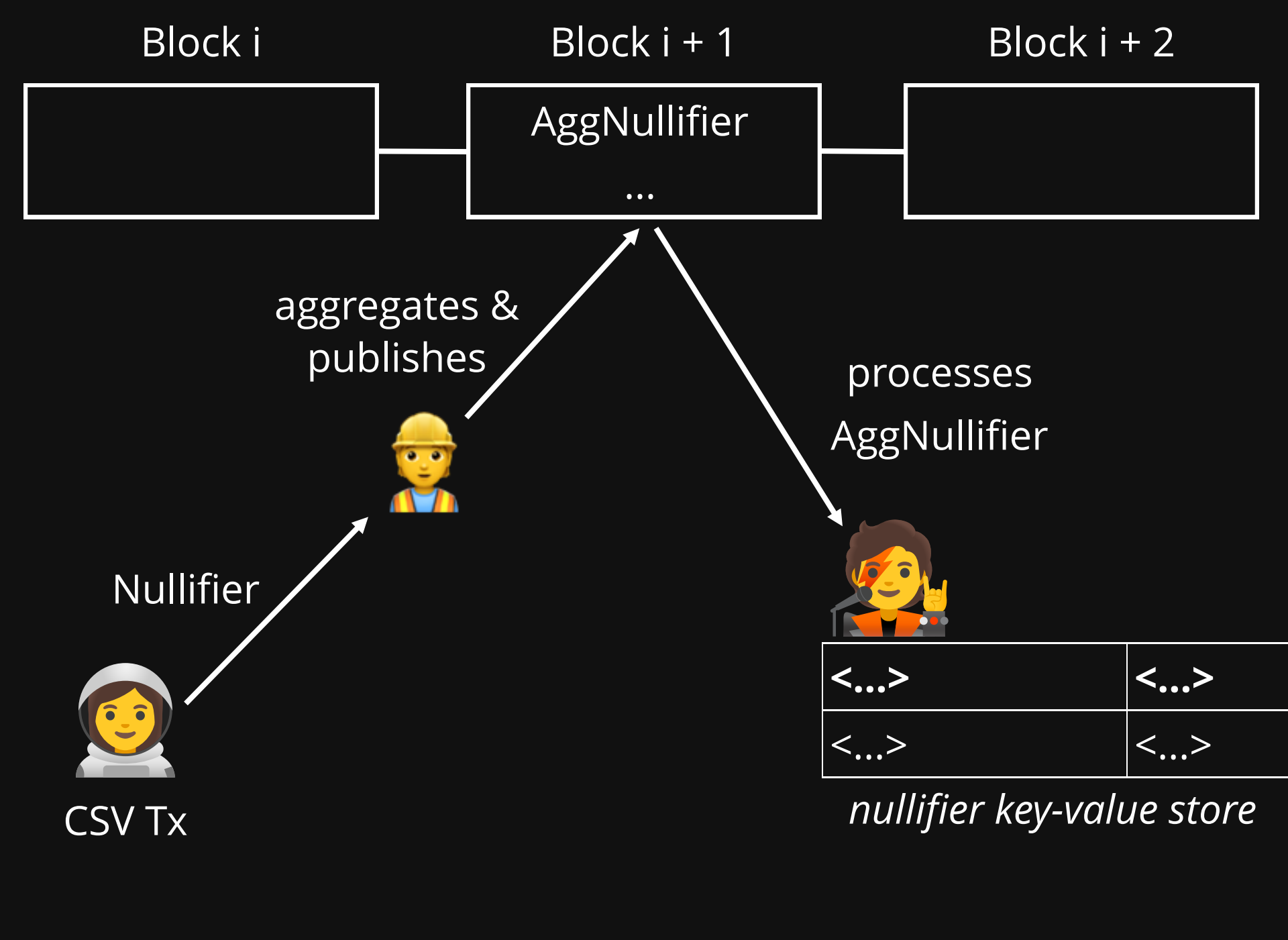


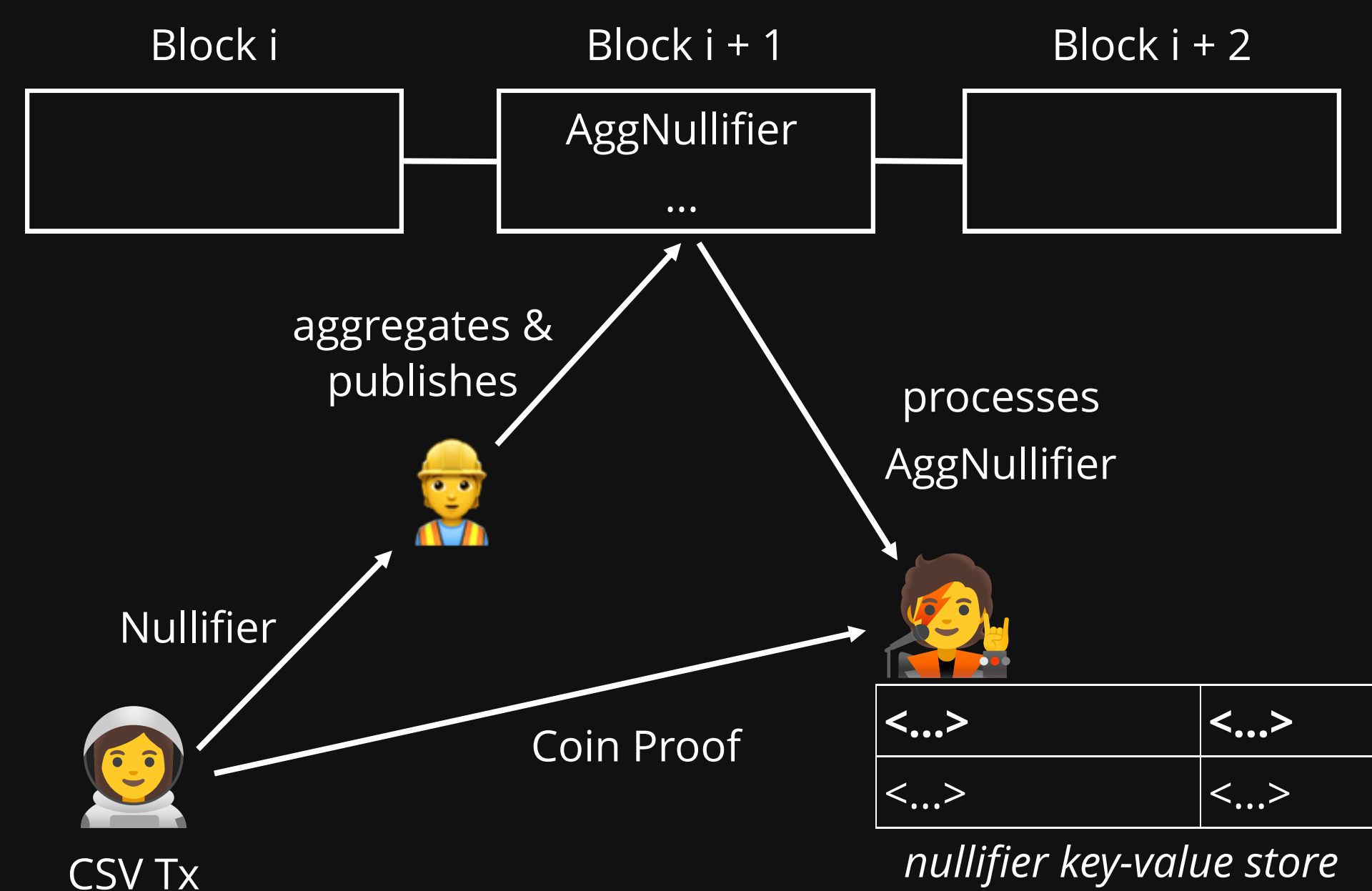
Nullifier



CSV Tx


Block i
Block i + 1
AggNullifier
...
aggregates &
publishes
Nullifier
CSV Tx


Block i
Block i + 1
Block i + 2
AggNullifier
...
aggregates &
publishes
processes
AggNullifier
Nullifier
CSV Tx
<...>
<...>
<...>
<...>
nullifier key-value store


Block i
Block i + 1
Block i + 2
AggNullifier
...
aggregates &
publishes
processes
AggNullifier
Nullifier
Coin Proof
<...>
<...>
<...>
<...>
CSV Tx
nullifier key-value store

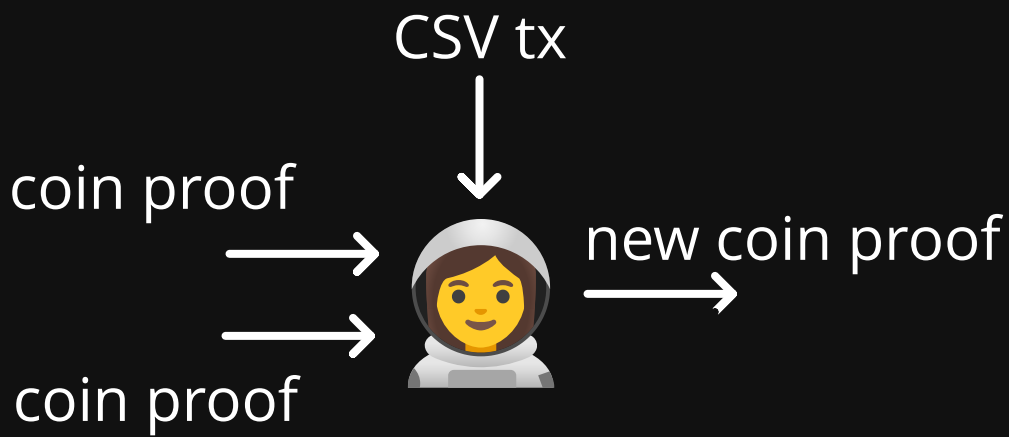
Succinct & Private Coin Proofs

Succinct & Private Coin Proofs

Problem: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?).

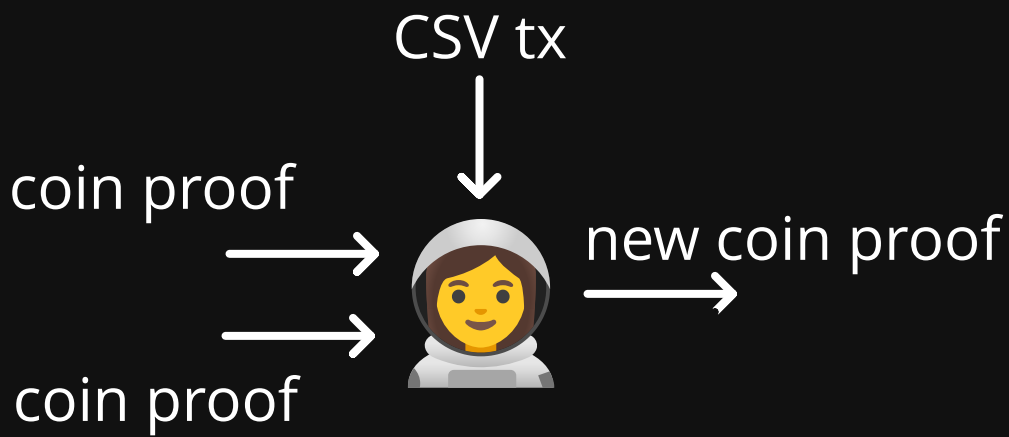
Succinct & Private Coin Proofs

Problem: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?).



Succinct & Private Coin Proofs

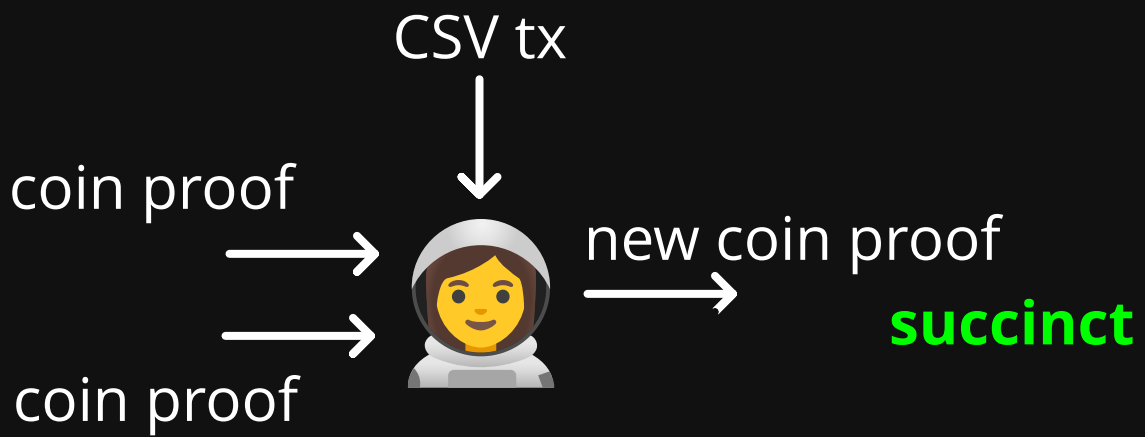
Problem: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?).



Solution: Replace new coin proof with a **cryptographic** proof that the incoming coin proofs and Sally's CSV tx is correct.

Succinct & Private Coin Proofs

Problem: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?).

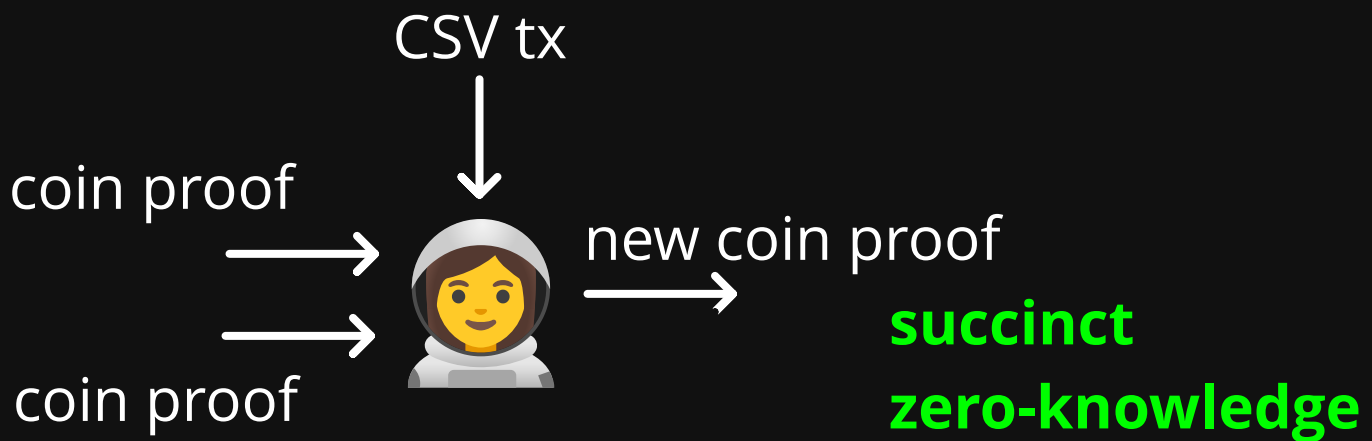


Solution: Replace new coin proof with a **cryptographic** proof that the incoming coin proofs and Sally's CSV tx is correct.

- Makes coin proof size and verification time **independent** of transaction graph size.

Succinct & Private Coin Proofs

Problem: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?).

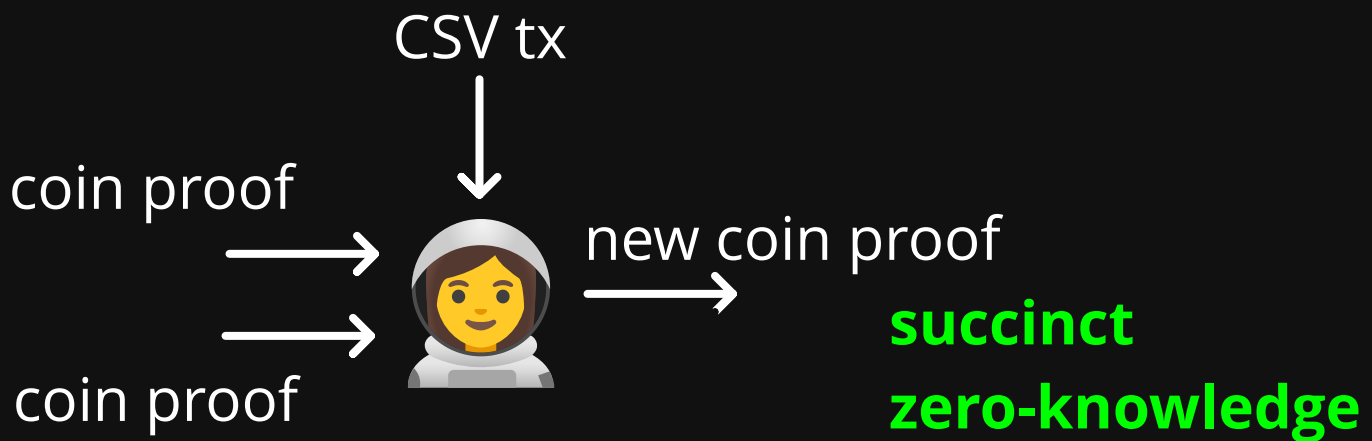


Solution: Replace new coin proof with a **cryptographic** proof that the incoming coin proofs and Sally's CSV tx is correct.

- Makes coin proof size and verification time **independent** of transaction graph size.

Succinct & Private Coin Proofs

Problem: Coin proof includes all ancestor transactions involved in creating the coin (size?, privacy?).



Solution: Replace new coin proof with a **cryptographic** proof that the incoming coin proofs and Sally's CSV tx is correct.

- Makes coin proof size and verification time **independent** of transaction graph size.
- Instantiated with **recursive SNARKs** or **folding schemes**.

shieldedcsv.org 

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.

Whitepaper

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.

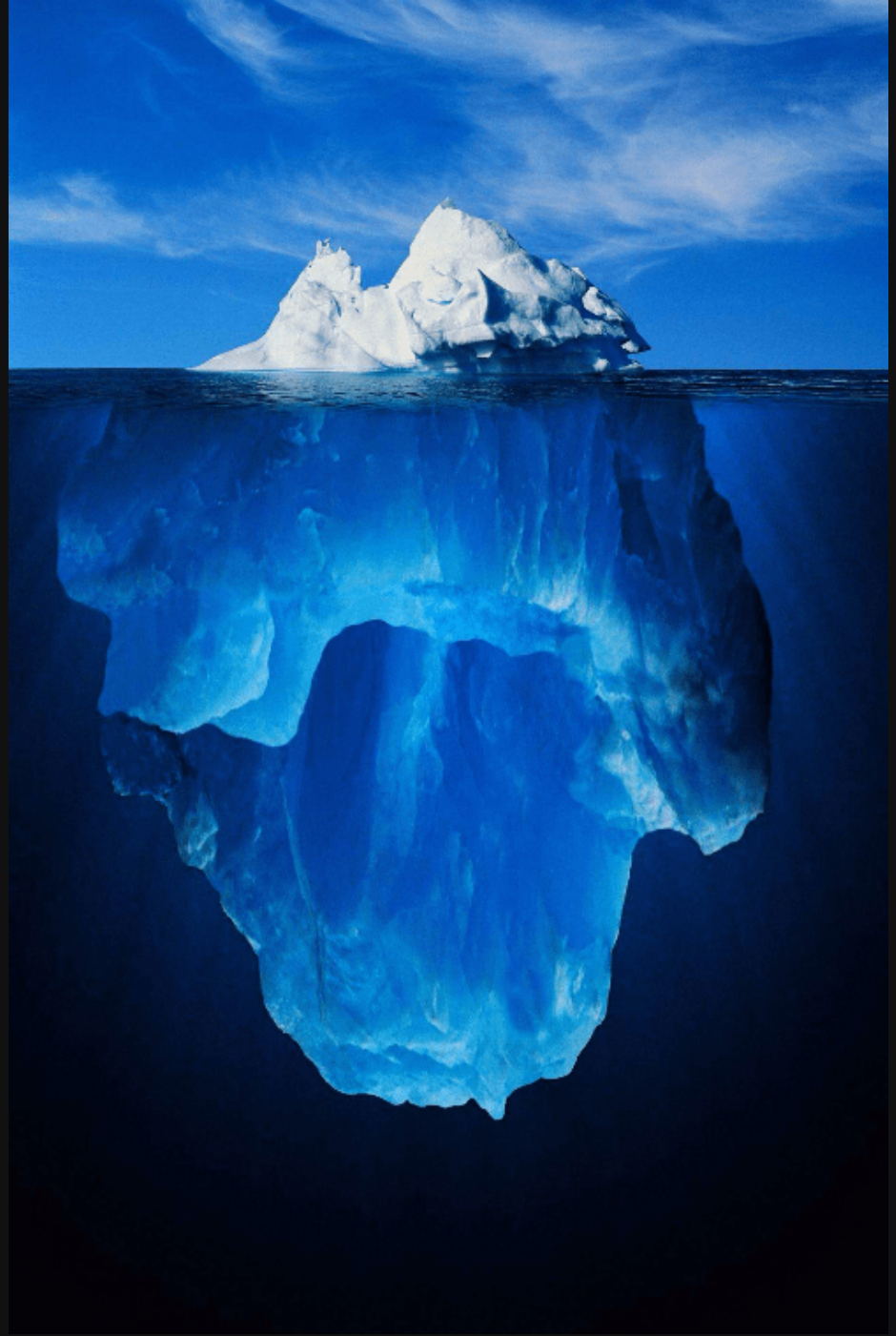
In this work, we introduce Shielded CSV, which improves upon state-of-the-art CSV protocols by providing the first construction that offers truly private transactions. It addresses the issues of traditional private cryptocurrency designs by requiring only 64 bytes of data per transaction, called a *nullifier*, 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.

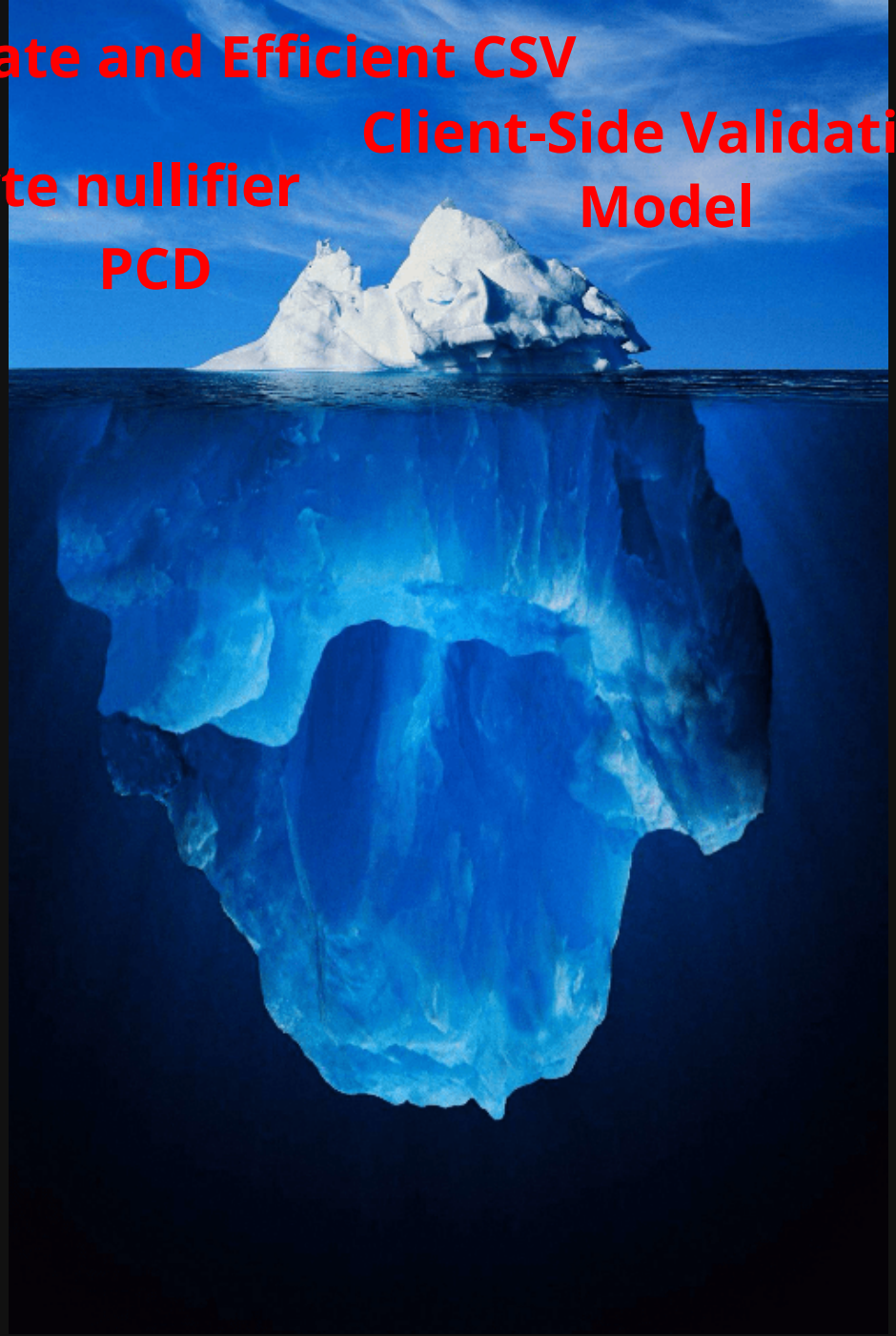
We specify Shielded CSV using the Proof Carrying Data (PCD) abstraction. We then discuss two implementation strategies that we believe to be practical, based on Folding Schemes and Recursive STARKs, respectively. Finally, we propose future extensions, demonstrating the power of the PCD abstraction and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

Whitepaper

```rust
// Aggregate nullifiers are posted to the blockchain by "publishers"
#[derive(Debug, PartialEq, Clone)]
pub struct AggregateNullifier {
    // Each public key commits to an account update
    pub pks: Vec<PublicKey>,
    // The "R-parts" of the half-aggregate signature commits to the transaction hash
    pub sig: Signature,
    // Commitment to the account ID of the publisher that will receive the fee
    pub fee_acct_comm: Commitment,
}
```

High-level rust spec




Private and Efficient CSV
Client-Side Validation
Model
64-byte nullifier
PCD

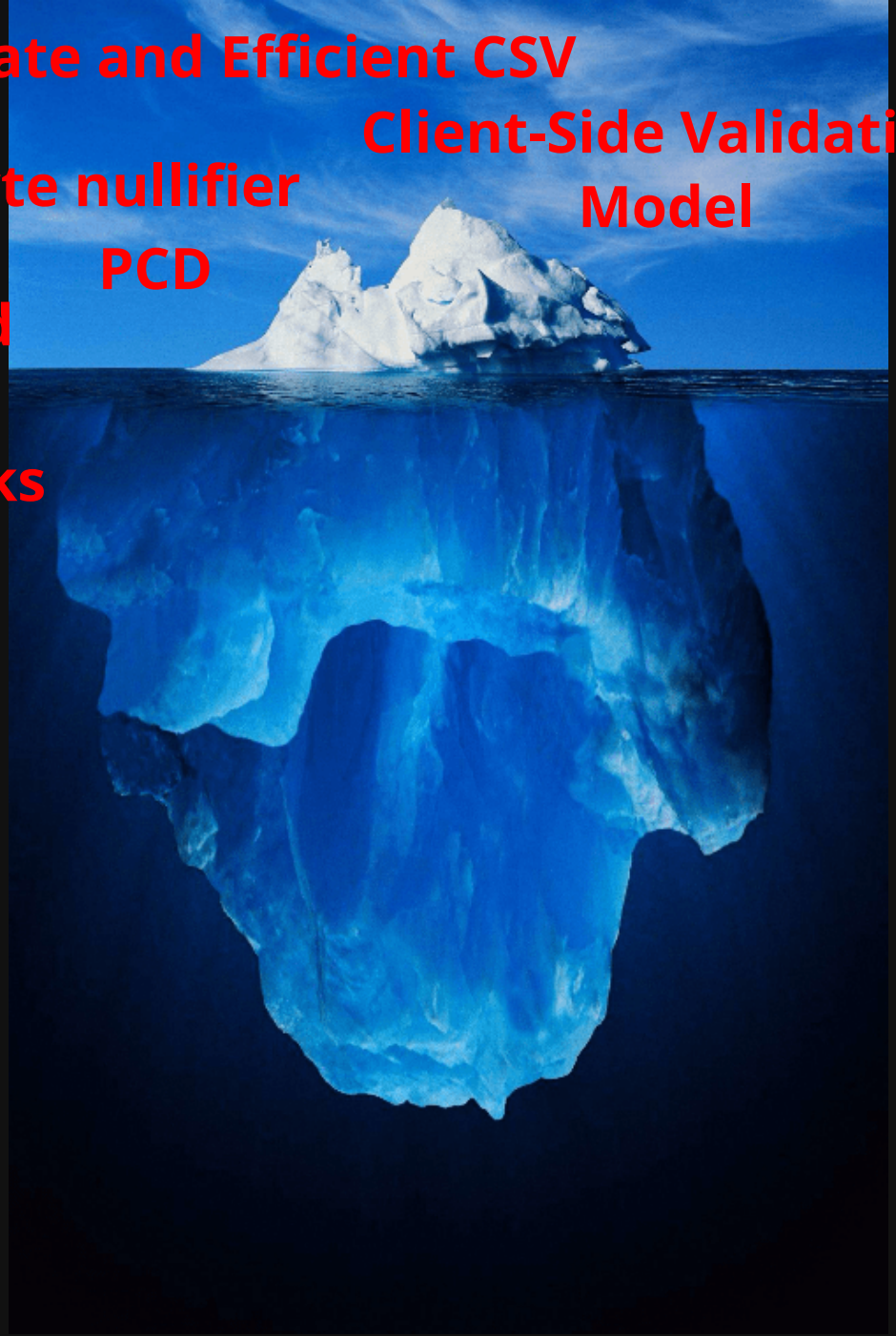
Private and Efficient CSV

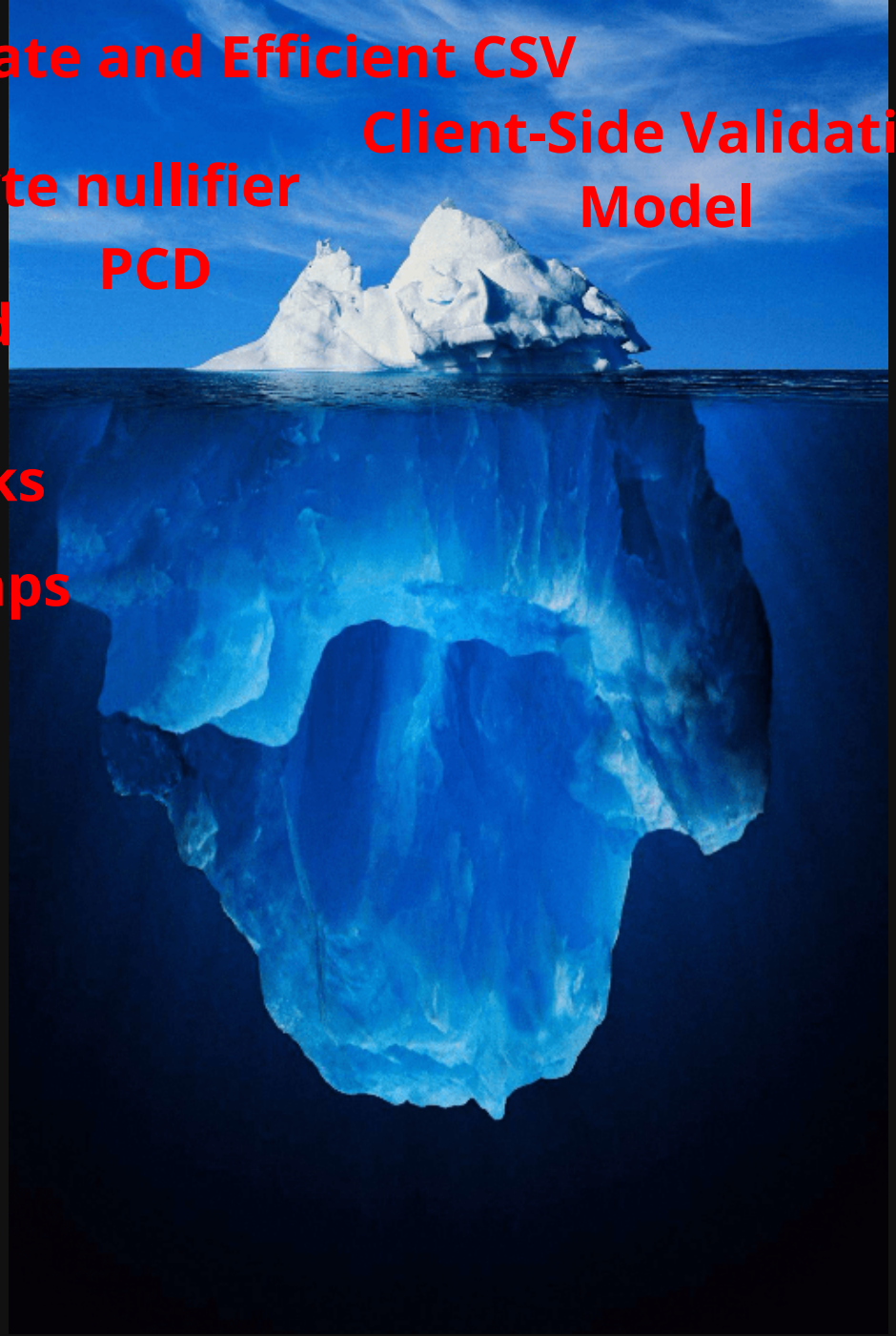
Client-Side Validation Model

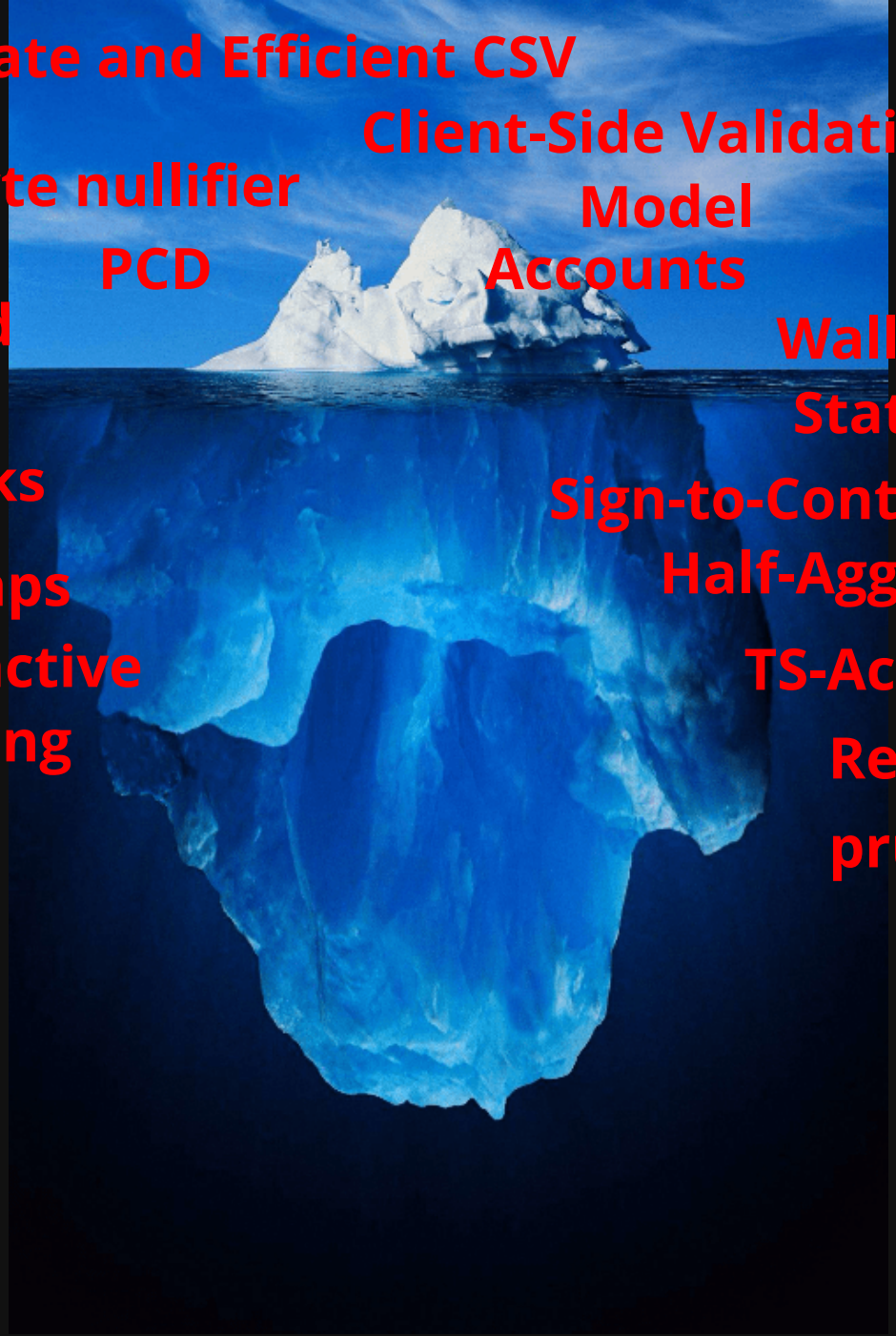
64-byte nullifier PCD

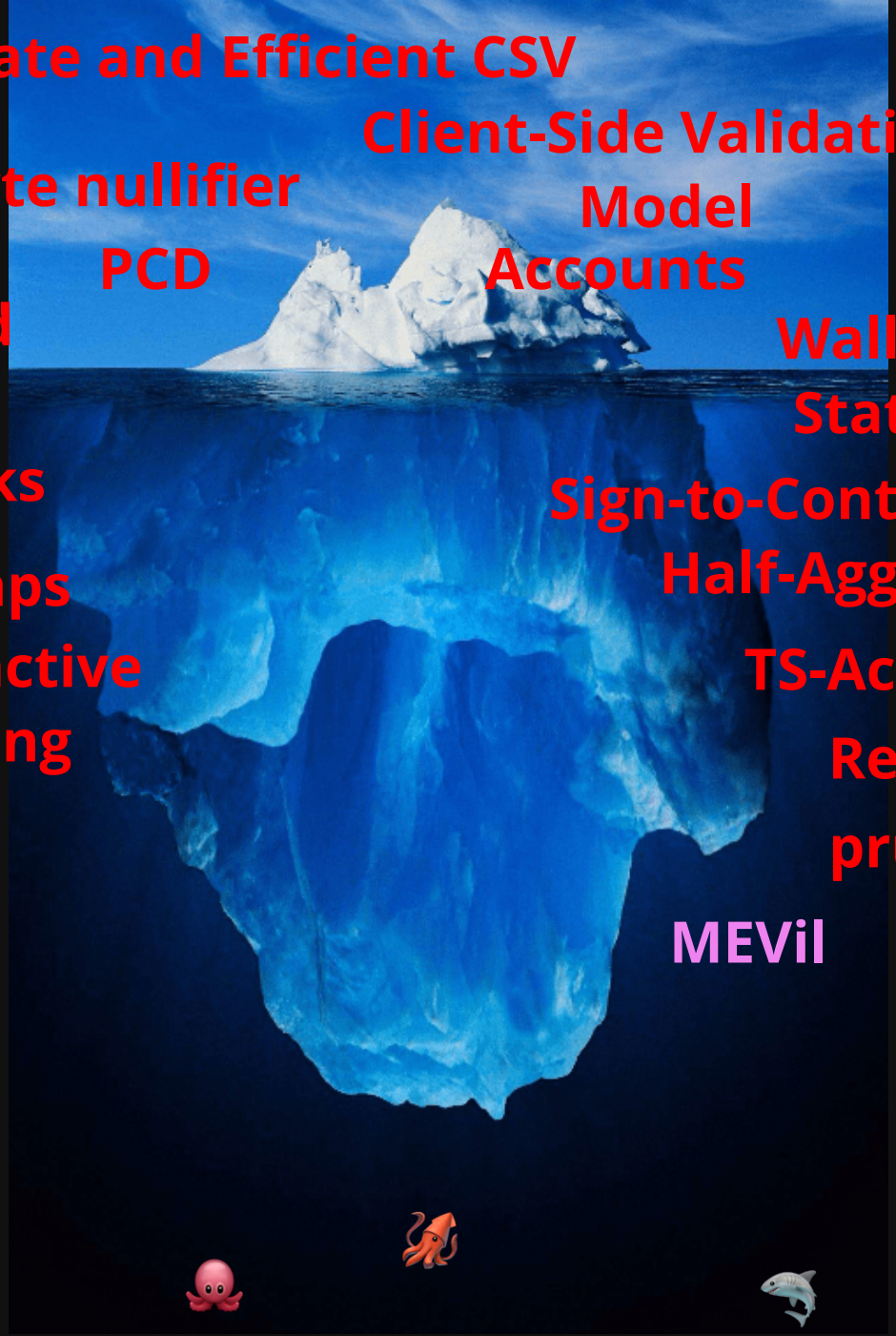
t-of-n Shared Accounts




Private and Efficient CSV
Client-Side Validation
Model
64-byte nullifier
PCD
t-of-n Shared
Accounts
Timelocks


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
t-of-n Shared
Accounts
Timelocks
Atomic Swaps


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
non-interactive
publishing
TS-Accumulator
Reorgs
prunable wallet
state


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
non-interactive
publishing
TS-Accumulator
Reorgs
prunable wallet
state
MEVil


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
non-interactive
publishing
TS-Accumulator
Reorgs
Scriptable spending policies
prunable wallet
state
Light clients
MEVil


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
TS-Accumulator
non-interactive
publishing
Reorgs
Scriptable spending policies
prunable wallet
state
Light clients
MEVil
Instantiation


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
non-interactive
publishing
TS-Accumulator
Reorgs
Scriptable spending policies
prunable wallet
state
Light clients
MEVil
Instantiation
"mempool"


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
non-interactive
publishing
TS-Accumulator
Reorgs
Scriptable spending policies
prunable wallet
state
Light clients
MEVil
Instantiation
"mempool"
Communication Channels

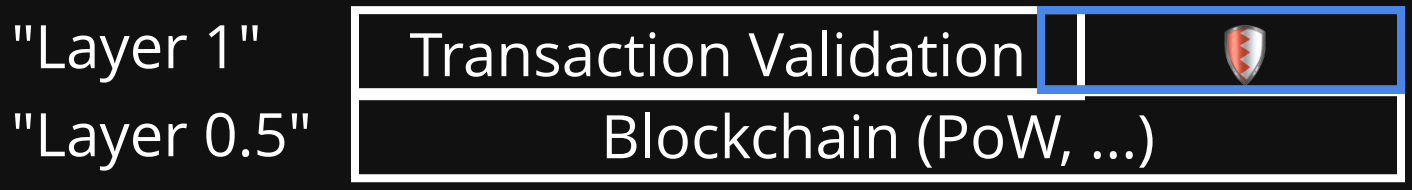

Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
non-interactive
publishing
TS-Accumulator
Reorgs
Scriptable spending policies
prunable wallet
state
Light clients
MEVil
Instantiation
"mempool"
Communication Channels
Post-quantum


Private and Efficient CSV
Client-Side Validation
64-byte nullifier
Model
PCD
Accounts
t-of-n Shared
Accounts
Wallet
State
Timelocks
Sign-to-Contract Schnorr
Half-Aggregation
Atomic Swaps
TS-Accumulator
non-interactive
publishing
Reorgs
Scriptable spending policies
prunable wallet
state
Light clients
MEVil
Instantiation
"mempool"
Communication Channels
Post-quantum
Payment Channels

shieldedcsv.org

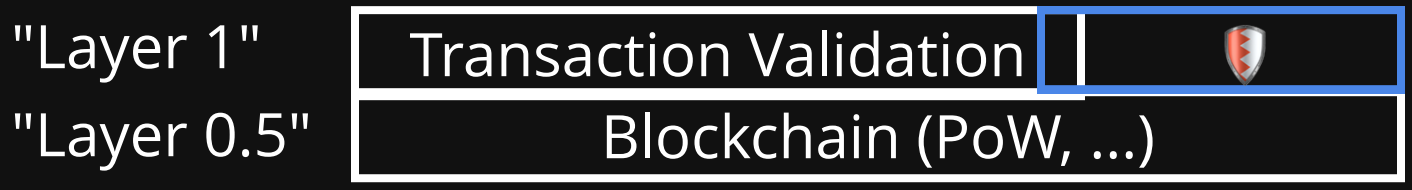


shieldedcsv.org



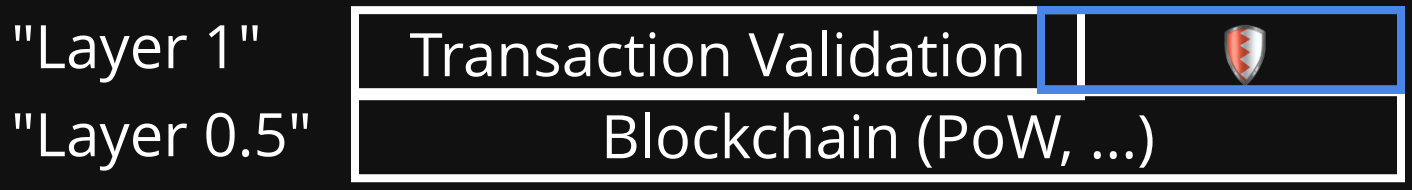
- Shielded CSV is a transaction protocol to **create an L1 on top of a blockchain** that allows embedding arbitrary data (e.g., Bitcoin).

shieldedcsv.org



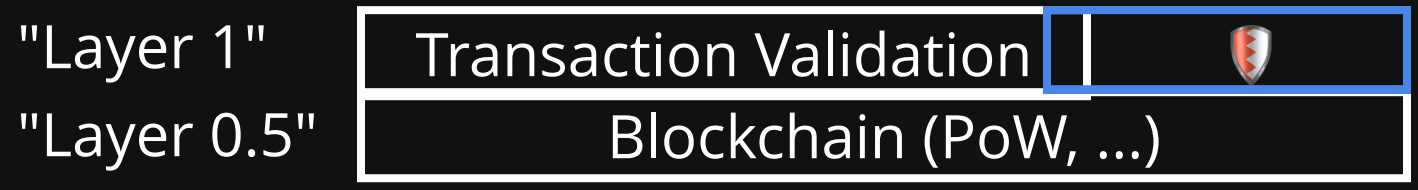
- Shielded CSV is a transaction protocol to **create an L1 on top of a blockchain** that allows embedding arbitrary data (e.g., Bitcoin).
- It **inherits double-spend security** from the underlying blockchain.

shieldedcsv.org



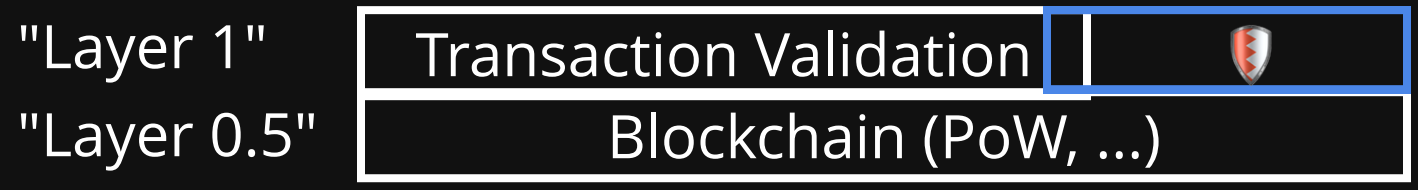
- Shielded CSV is a transaction protocol to **create an L1 on top of a blockchain** that allows embedding arbitrary data (e.g., Bitcoin).
- It **inherits double-spend security** from the underlying blockchain.
- The amount of data embedded into the blockchain is **64 bytes** per transaction.

shieldedcsv.org



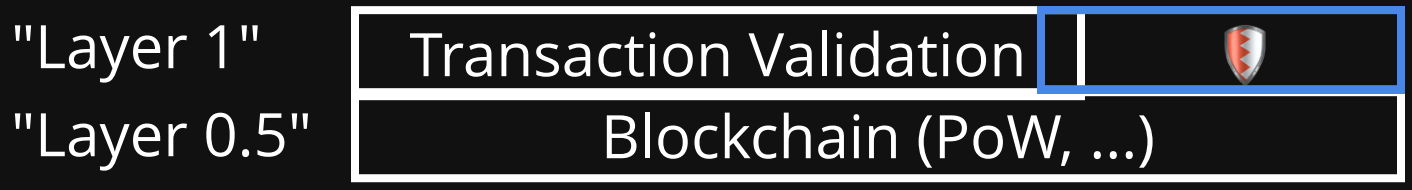
- Shielded CSV is a transaction protocol to **create an L1 on top of a blockchain** that allows embedding arbitrary data (e.g., Bitcoin).
- It **inherits double-spend security** from the underlying blockchain.
- The amount of data embedded into the blockchain is **64 bytes** per transaction.
- Coin proofs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

shieldedcsv.org



- 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** per transaction.
- Coin proofs are sent directly to the receiver through a **private one-way communication channel**.
- Coin proofs are **succinct**.

shieldedcsv.org



- 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** per transaction.
- Coin proofs are sent directly to the receiver through a **private one-way communication channel**.
- Coin proofs are **succinct**.
- Shielded CSV is fully **private**.
- **No softfork** required. But security is inherited from the bridge.

You've unlocked the backup slides

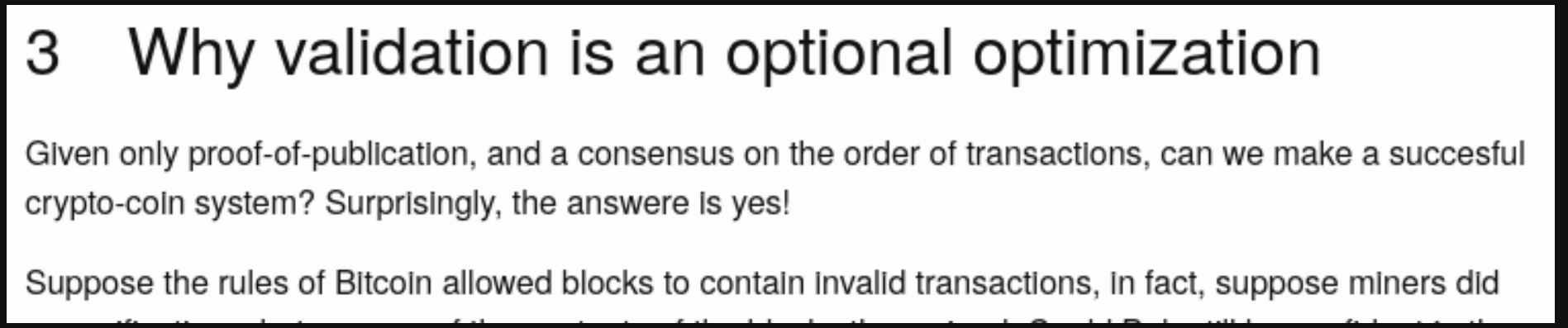
What is CSV really?

What is CSV really?

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

What is CSV really?

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



3 Why validation is an optional optimization

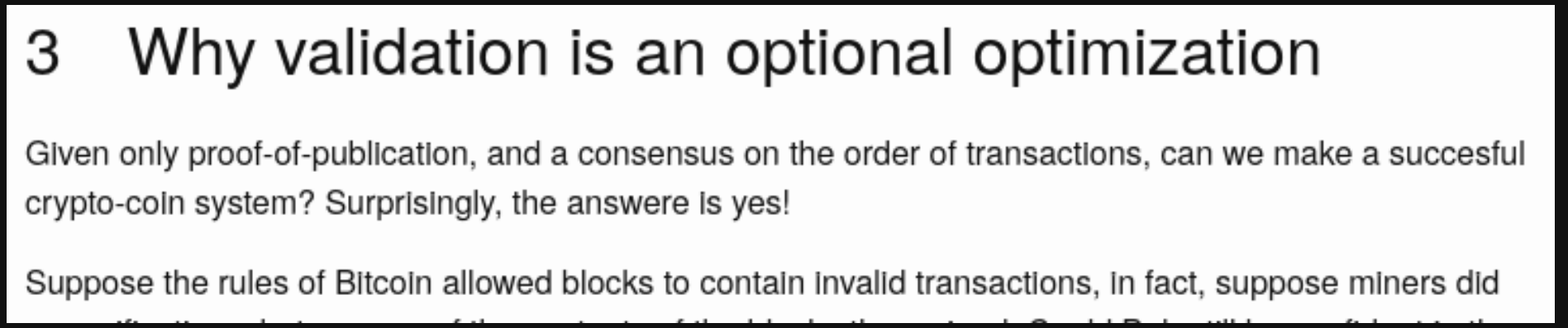
Given only proof-of-publication, and a consensus on the order of transactions, can we make a succesful crypto-coin system? Surprisingly, the answere is yes!

Suppose the rules of Bitcoin allowed blocks to contain invalid transactions, in fact, suppose miners did

petertodd.org, 2013

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, in fact, suppose miners did

petertodd.org, 2013

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

Comparing CSV

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

Comparing CSV

Privacy	Coin Proof Size	Blockchain Space
RGB		BitcoinTx(num_ins, num_outs)
Taproot Assets		BitcoinTx(num_ins, num_outs)
Intmax2		Asymptotic to 4 to 5 bytes (interactive)
Shielded CSV		Asymptotic to 64 bytes (non-interactive)

Comparing CSV

Privacy	Coin Proof Size	Blockchain Space
RGB	$O(\|\mathsf{tx_history}\|)$	$\mathsf{BitcoinTx}(\mathsf{num_ins}, \mathsf{num_outs})$
Taproot Assets	$O(\|\mathsf{tx_history}\|)$	$\mathsf{BitcoinTx}(\mathsf{num_ins}, \mathsf{num_outs})$
Intmax2	$O(1)$	Asymptotic to 4 to 5 bytes (interactive)
Shielded CSV	$O(1)$	Asymptotic to 64 bytes (non-interactive)

Comparing CSV

	Privacy	Coin Proof Size	Blockchain Space
RGB	Encrypted amounts & assets, LN-compatible	$O(\|\mathsf{tx_history}\|)$	$\mathsf{BitcoinTx}(\mathsf{num_ins}, \mathsf{num_outs})$
Taproot Assets	No, LN-compatible	$O(\|\mathsf{tx_history}\|)$	$\mathsf{BitcoinTx}(\mathsf{num_ins}, \mathsf{num_outs})$
Intmax2	No	$O(1)$	Asymptotic to 4 to 5 bytes (interactive)
Shielded CSV	Yes	$O(1)$	Asymptotic to 64 bytes (non-interactive)