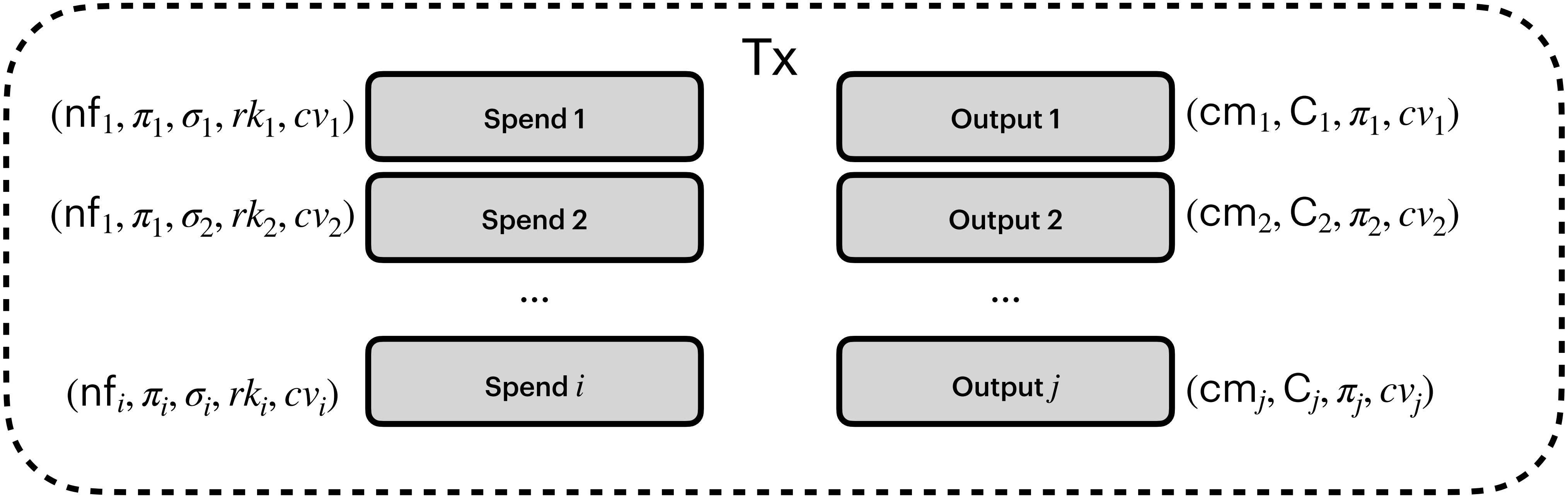
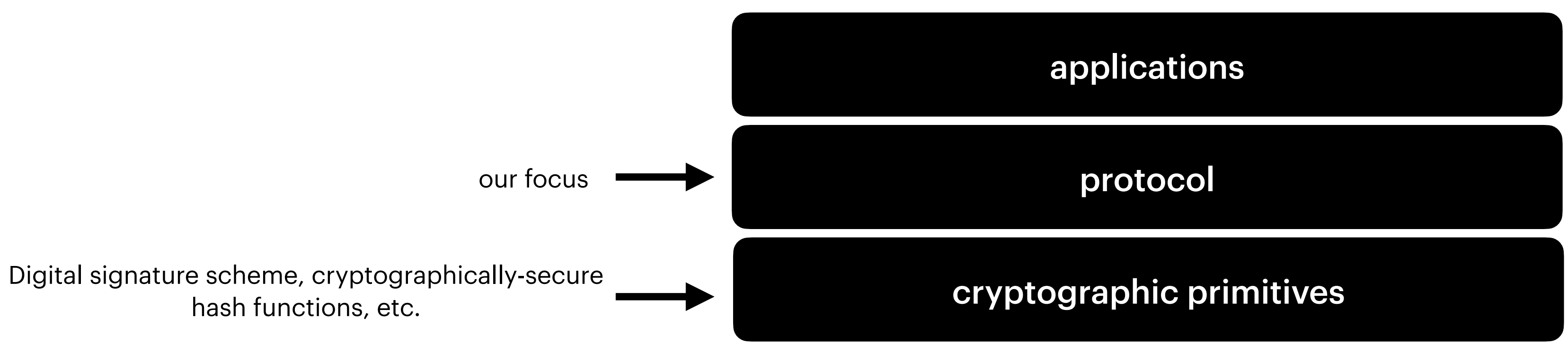
Cryptographic Protocol Design for UTXO-Based Decentralized Anonymous Payments



redshiftzero

Today

- Recap: Bitcoin transaction structure and UTXOs
- Properties of an idealized blockchain:
  - *confidentiality/privacy*
  - *integrity*
- Building a private and decentralized UTXO-based protocol



Motivation

Bitcoin is not private

- Bitcoin is peer-to-peer, non-custodial, and decentralized
- But a passive observer can see:
  - Plaintext values of inputs and outputs
  - *Pseudo-anonymous* sender and recipient identities
- **Privacy in an open society requires anonymous transaction systems**

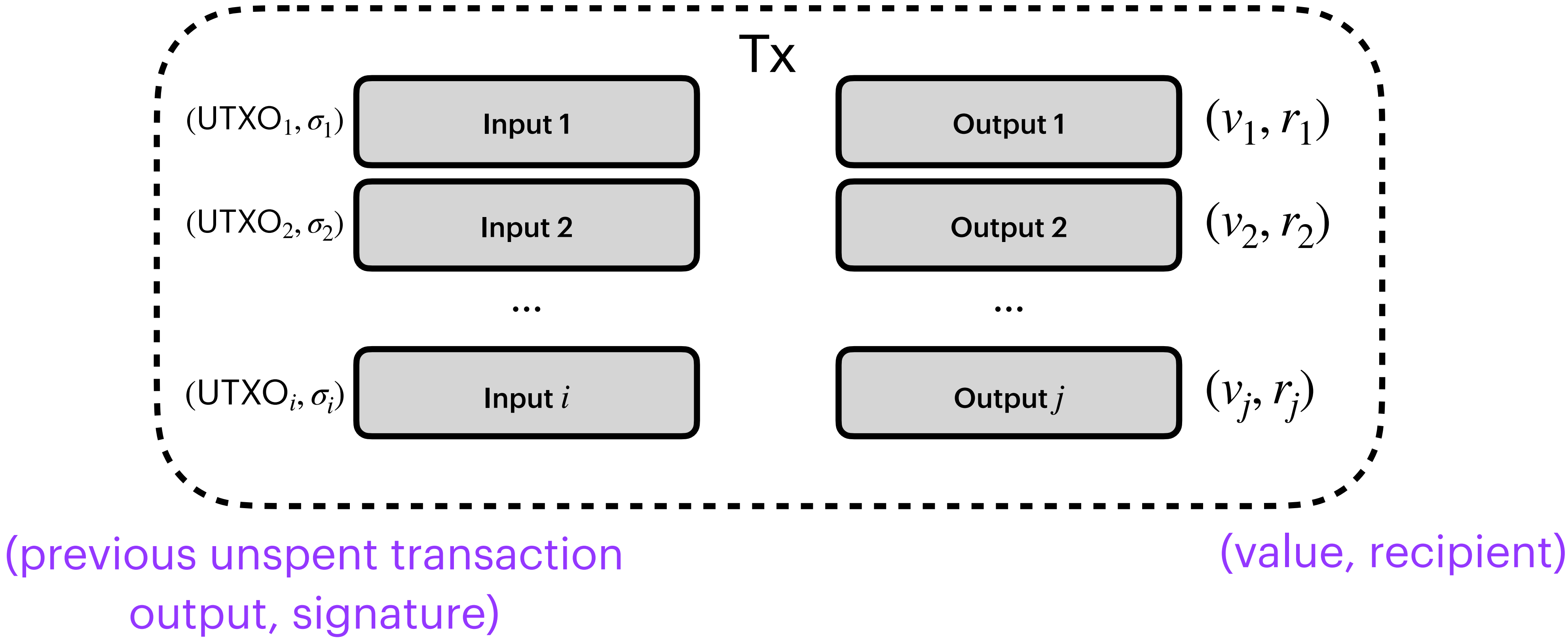
System Invariants

Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities.
- Integrity:
  - You cannot spend other people's coins.
  - You cannot spend coins that don't exist.
  - You cannot spend coins twice.
  - You cannot create or destroy value.

Recap

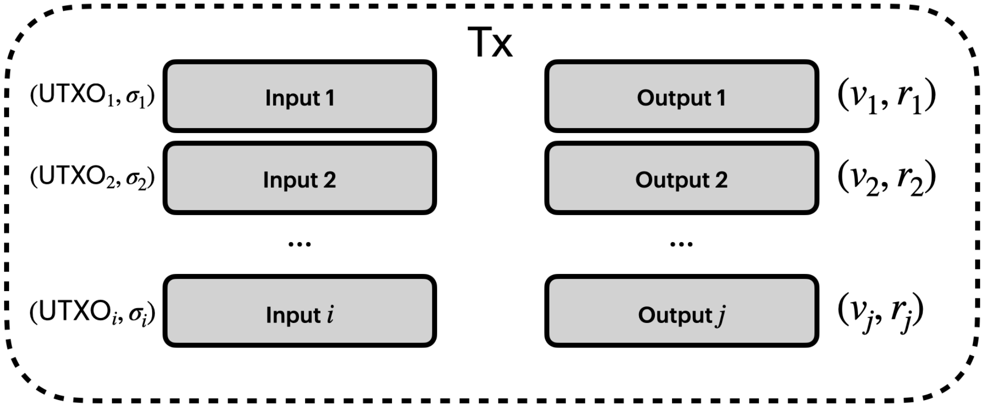
Bitcoin Transaction Structure for a Simple Payment



Bitcoin

System Invariants

~~Privacy and~~ Integrity



- ~~Private:~~
  - ~~A passive observer should not learn anything about values, sender or recipient identities.~~
- Integrity:
  - You cannot spend other people's coins. ✅ Digital signatures
  - You cannot spend coins that don't exist. ✅ Each input must refer to an UTXO
  - You cannot spend coins twice. ✅ Each UTXO can only be claimed once
  - You cannot create or destroy value. ✅ We can calculate the sum of the inputs equals the sum of the outputs

System Invariants
Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities.
- Integrity:
  - You cannot spend other people’s coins.
  - You cannot spend coins that don’t exist.
  - You cannot spend coins twice.
  - You cannot create or destroy value.

System Invariants

Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities. ✅ Encrypt all of these
- Integrity:
  - You cannot spend other people's coins. ❌ Digital signatures link transactions
  - You cannot spend coins that don't exist. ❌ Can't directly reference UTXOs
  - You cannot spend coins twice. ❌ Can't directly reference UTXOs
  - You cannot create or destroy value. ❌ We can't directly calculate the sum of the inputs equals the sum of the outputs because the values are encrypted

2013 IEEE Symposium on Security and Privacy

Zerocoin: Anonymous Distributed E-Cash from Bitcoin

Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin
The Johns Hopkins University Department of Computer Science, Baltimore, USA
{*imiers, cgarman, mgreen, rubin*}*@cs.jhu.edu*

Zerocash: Decentralized Anonymous Payments from Bitcoin
(extended version)

Eli Ben-Sasson[*] Alessandro Chiesa[†] Christina Garman[‡] Matthew Green[‡]
Ian Miers[‡] Eran Tromer[§] Madars Virza[†]

May 18, 2014

Zcash Protocol Specification
Version 2021.1.15 [NU5 proposal]

Daira Hopwood[†]
Sean Bowe[†] — Taylor Hornby[†] — Nathan Wilcox[†]

September 1, 2021

Cryptographic Primitive: ZKPs

Introduction

- A *Zero Knowledge Proof (ZKP)* demonstrates a statement $\phi(w)$ is true, without revealing anything more about w other than the statement.

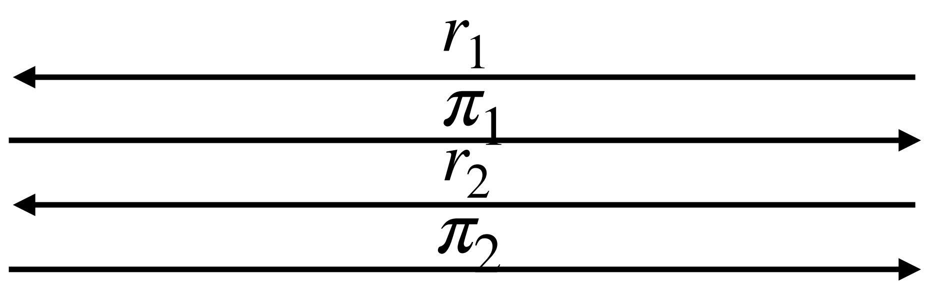
Cryptographic Primitive: ZKPs

Interactive Setting

Verifier 👩🏽

Prover 👱

- I want to convince the verifier a statement $\phi(w)$ is true without revealing my private data w.



Generates random challenge r_1

Generates random challenge r_2

At the end, verifier decides based on prover responses whether to accept.

Cryptographic Primitive: ZKPs

Key Properties

- **Completeness:**
  - An honest prover can convince an honest verifier that the statement ($\phi(w)$) is true.
- **Soundness:**
  - A dishonest prover cannot convince an honest verifier that a proof of a false statement is true.
- **Zero-Knowledge:**
  - Nothing more is revealed other than the truth of $\phi(w)$

Cryptographic Primitive: ZKPs

Example: Billiard Balls

Prover 👱

- I want to convince the verifier these two billiard balls are different colors

Colorblind Verifier 👩🏽

Cryptographic Primitive: ZKPs

Example: Billiard Balls

Prover 👱

- I want to convince the verifier these two billiard balls are different colors

Colorblind Verifier 👩🏽

Tosses coin r_1, switches balls if heads

Did I switch? ←

Yes/no →

Cryptographic Primitive: ZKPs

Example: Billiard Balls

Prover

- I want to convince the verifier these two billiard balls are different colors

Colorblind Verifier

Did I switch?

Yes/no

Tosses coin r_1, switches balls if heads

Continue until your friend is convinced

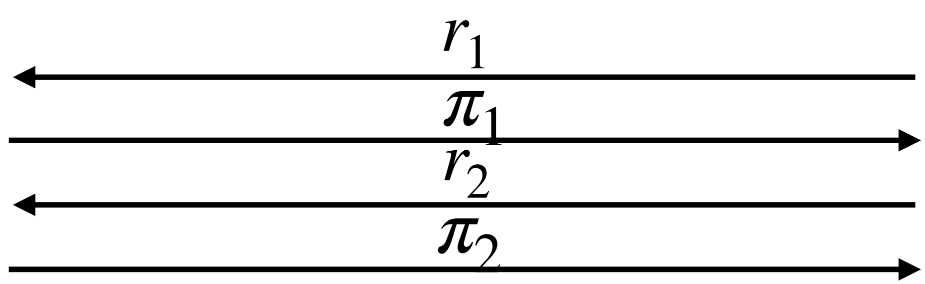
Cryptographic Primitive: ZKPs

Interactive Setting

Verifier

Prover

- I want to convince the verifier a statement $\phi(w)$ is true without revealing my private data w.



Generates random challenge r_1

Generates random challenge r_2

At the end, verifier decides based on prover responses whether to accept.

Cryptographic Primitive: ZKPs

Non-Interactive Setting

- I want to convince the verifier a statement $\phi(w)$ is true without revealing my private data w.

<u>Prover</u> 👱

$\pi \leftarrow \text{Prove}(\phi, w, \ldots)$

<u>Verifier</u> 👩🏽

$0/1 \leftarrow \text{Vfy}(\phi, \ldots)$

$$\xrightarrow{\quad\quad\quad\quad \pi \quad\quad\quad\quad}$$

Cryptographic Primitive: zk-SNARKs

Introduction

zk-SNARK:

- **Z**ero **K**nowledge
- **S**uccinct
- **N**on-interactive
- **AR**gument of
- **K**nowledge

Cryptographic Primitive: zk-SNARKs

Basic Algorithms (for pre-processing SNARK)

- **Preprocess/Setup:** Given the statements/computation ϕ, produce public parameters pp for proving/verification.
  - "Trusted Setup"
- **Prove:** Given $\phi(w)$ and public parameters pp, produce a proof π
- **Verify:** Given a proof π and public parameters pp, accept or reject the proof

Cryptographic Primitive: zk-SNARKs

Circuit Programming

π

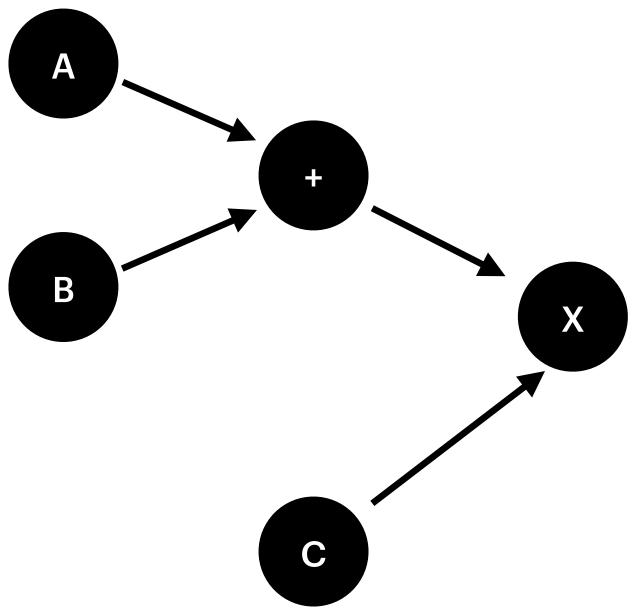
- *Public input*: The data that will be public on the blockchain
- *Witnesses*: The data we want to be private
- *Statements*: The things we want to prove about the witnesses and check against the public inputs

- I want to convince the verifier I know the pre-image w given hash output $H(w)$.

Prover 👦

Cryptographic Primitive: zk-SNARKs

Represent the computation as an arithmetic circuit



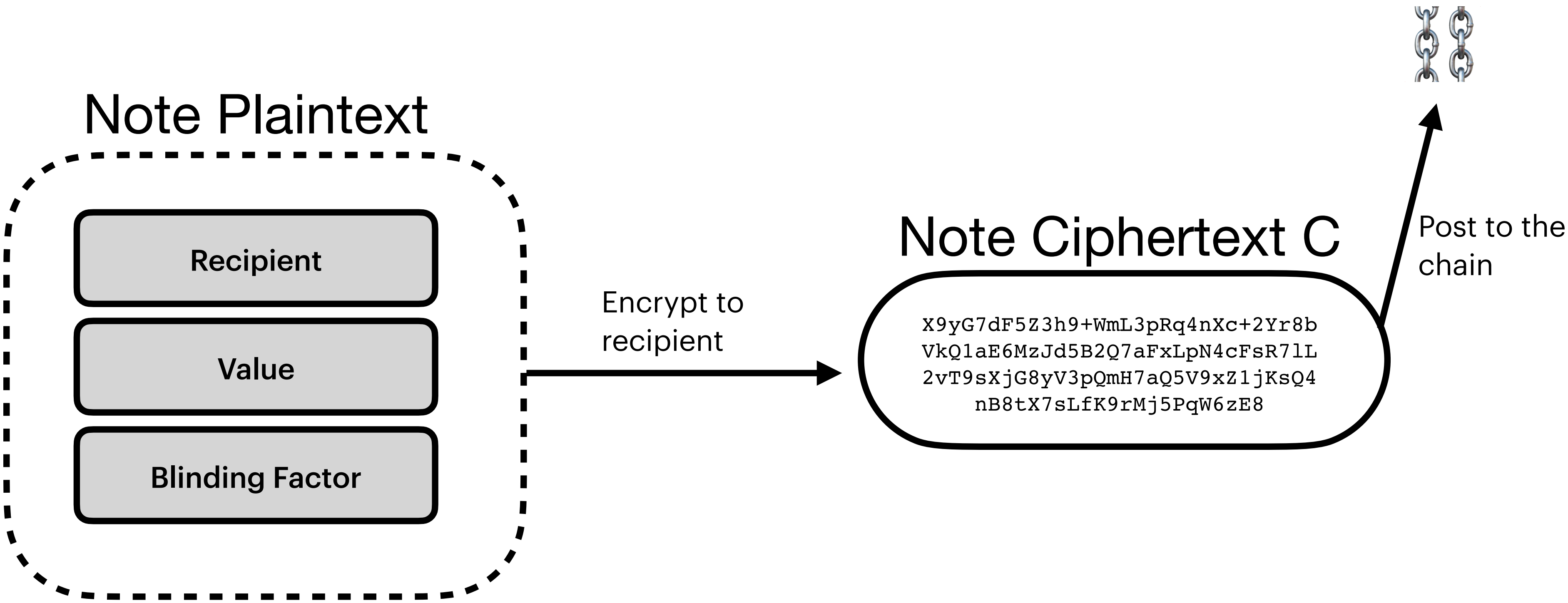
- Gates: Addition, multiplication
- A, B, C: constants or inputs
- (A + B) x C

System Invariants
Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities.
- Integrity:
  - You cannot spend other people's coins.
  - You cannot spend coins that don't exist.
  - You cannot spend coins twice.
  - You cannot create or destroy value.

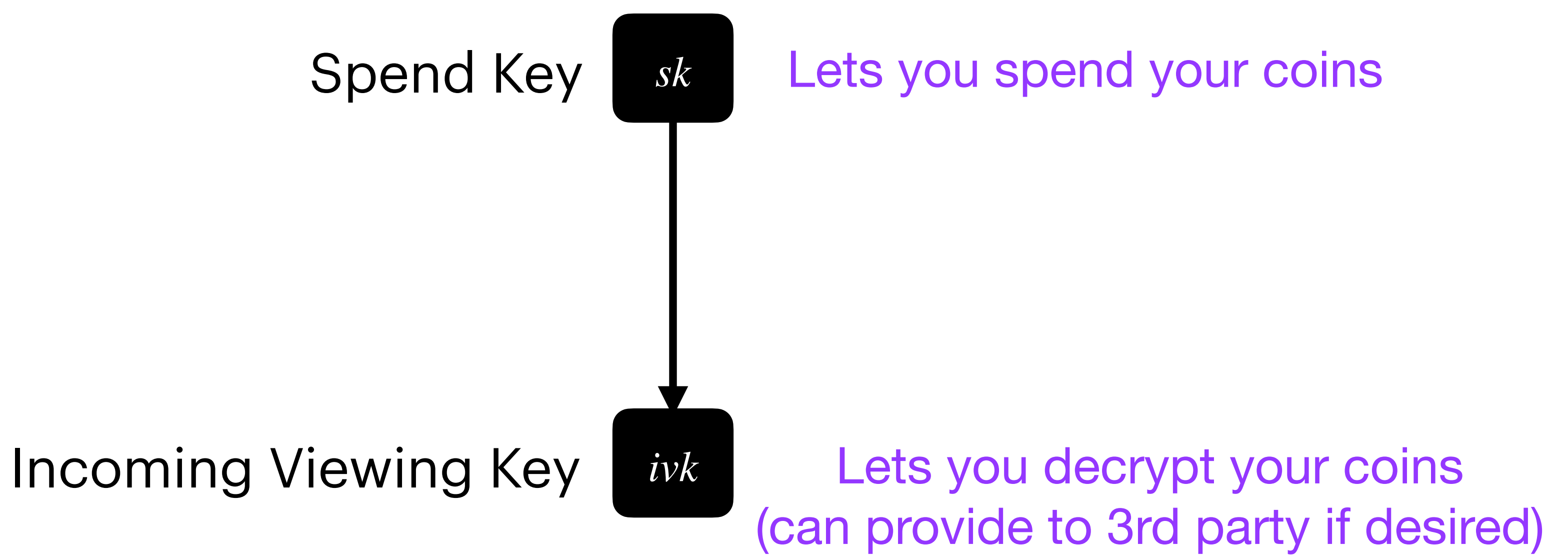
Privacy: Encrypted Notes

Value in the system is carried by *Notes*



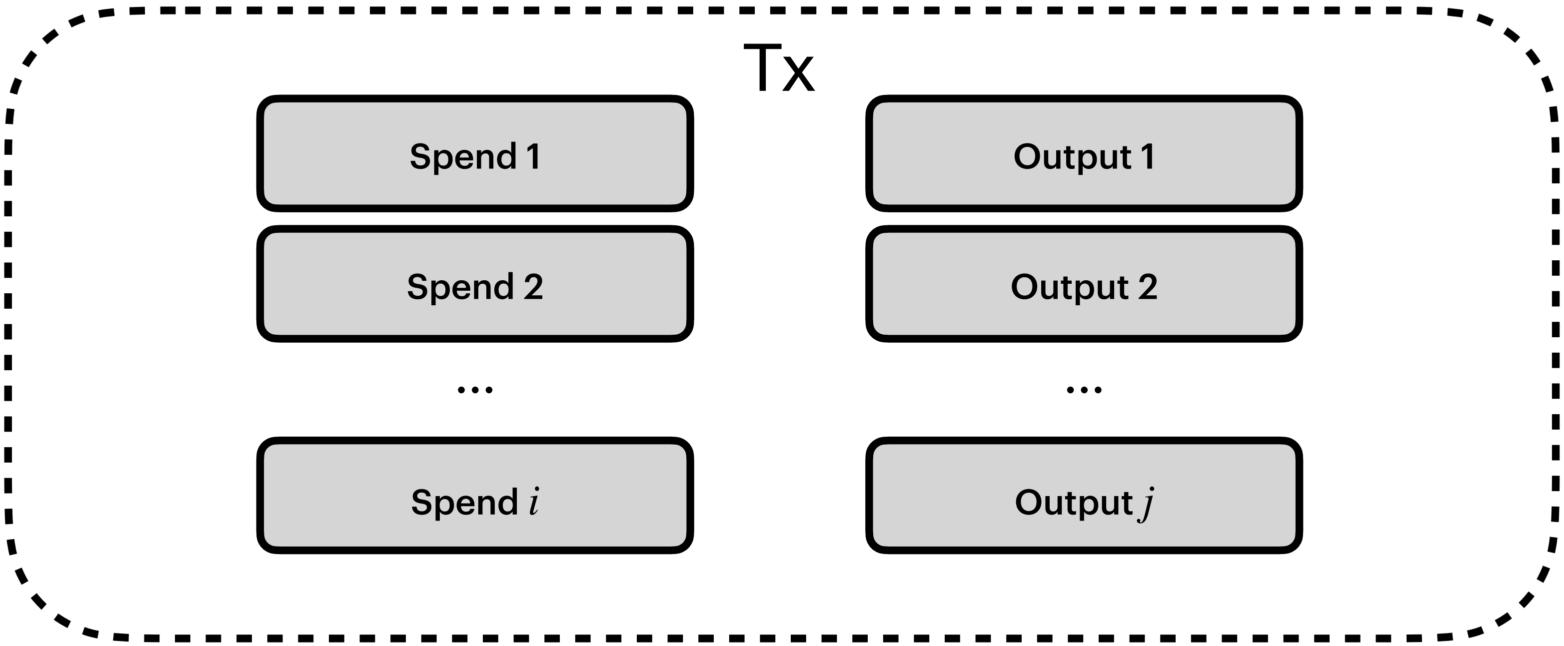
ZCash-Style Key Hierarchy

Spending and Viewing Capabilities are Separated



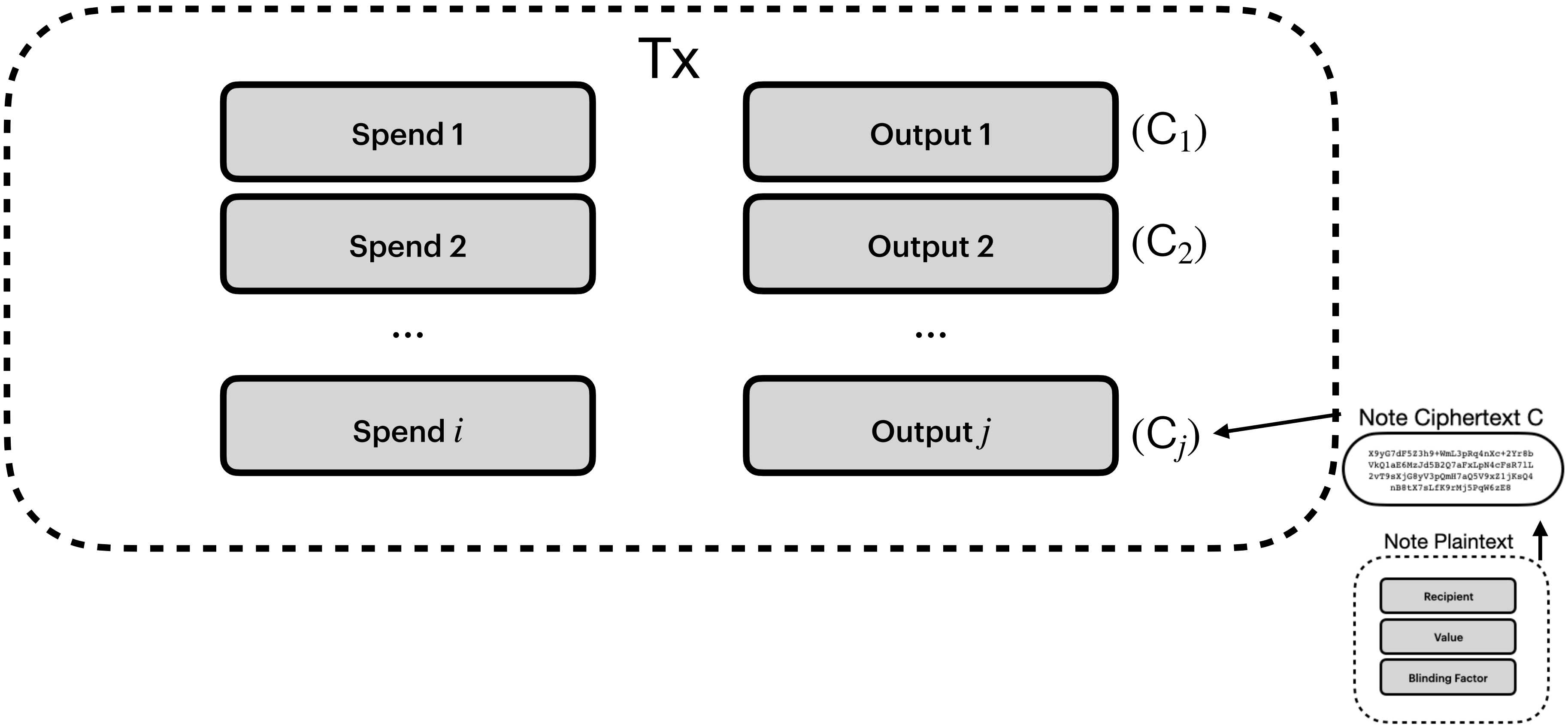
Shielded Transaction Structure

Shielded Transactions consist of *Actions*



Shielded Transaction Structure

Shielded Transactions consist of *Actions*



System Invariants
Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities. ✅
- Integrity:
  - You cannot spend other people's coins.
  - You cannot spend coins that don't exist.
  - You cannot spend coins twice.
  - You cannot create or destroy value.

Integrity: You cannot spend coins that don’t exist

- We need nodes to keep track of *privately*:
  1. All notes that exist in the system
  2. All notes that have been spent in the system

- We need to derive a quantity from each note that:
  - Binds us to the value and recipient
  - Hides the value and recipient

Cryptographic Primitive: Commitment Schemes

- A *cryptographic commitment scheme* is basically an envelope:

- Key properties:
  - **Binding**: I cannot change my value once it is sealed inside the envelope.
  - **Hiding**: No one else can look at the value, because it is inside the sealed envelope.

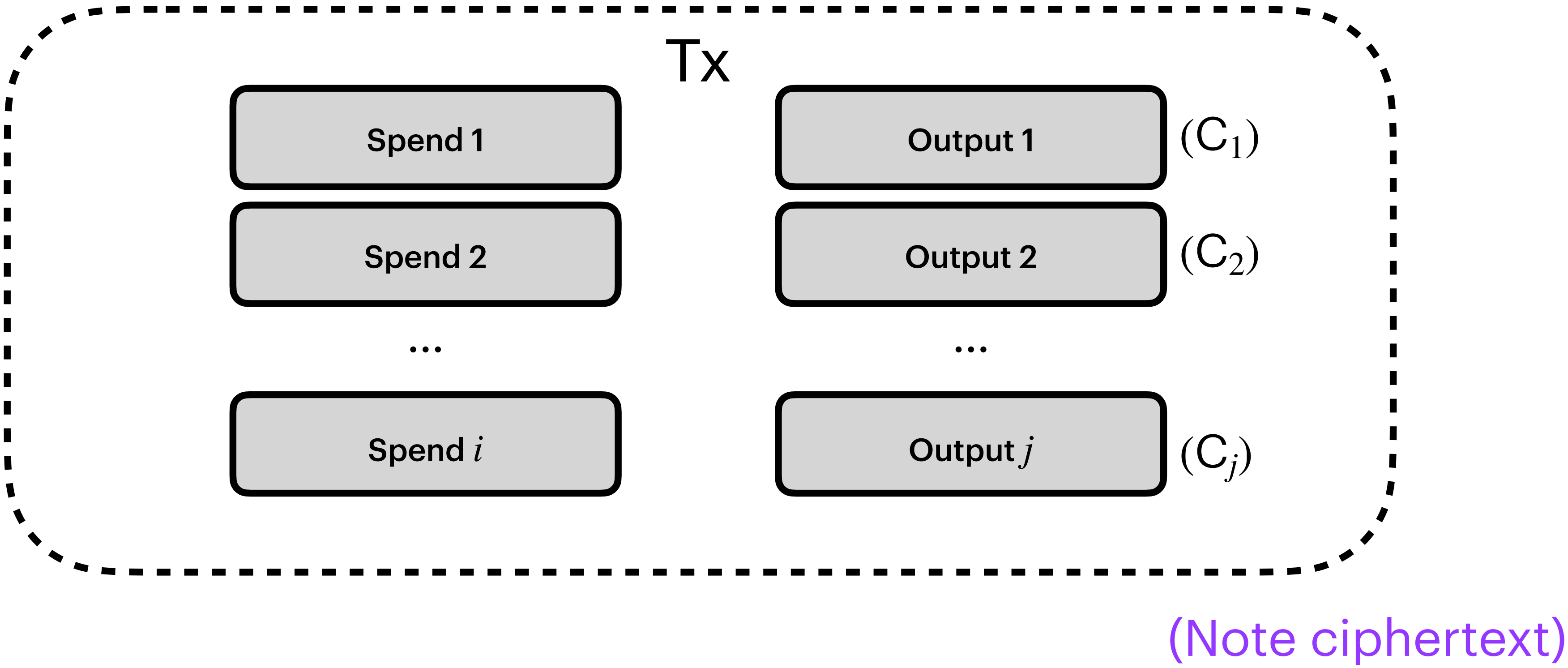
Cryptographic Primitive: Commitment Schemes

- Two phases: Commit and Reveal
- Commit:
  - $\text{cm} = \text{Commit}(m, \text{randomness})$

- Reveal:
  - Verify(m, cm, randomness) = [accept, reject]

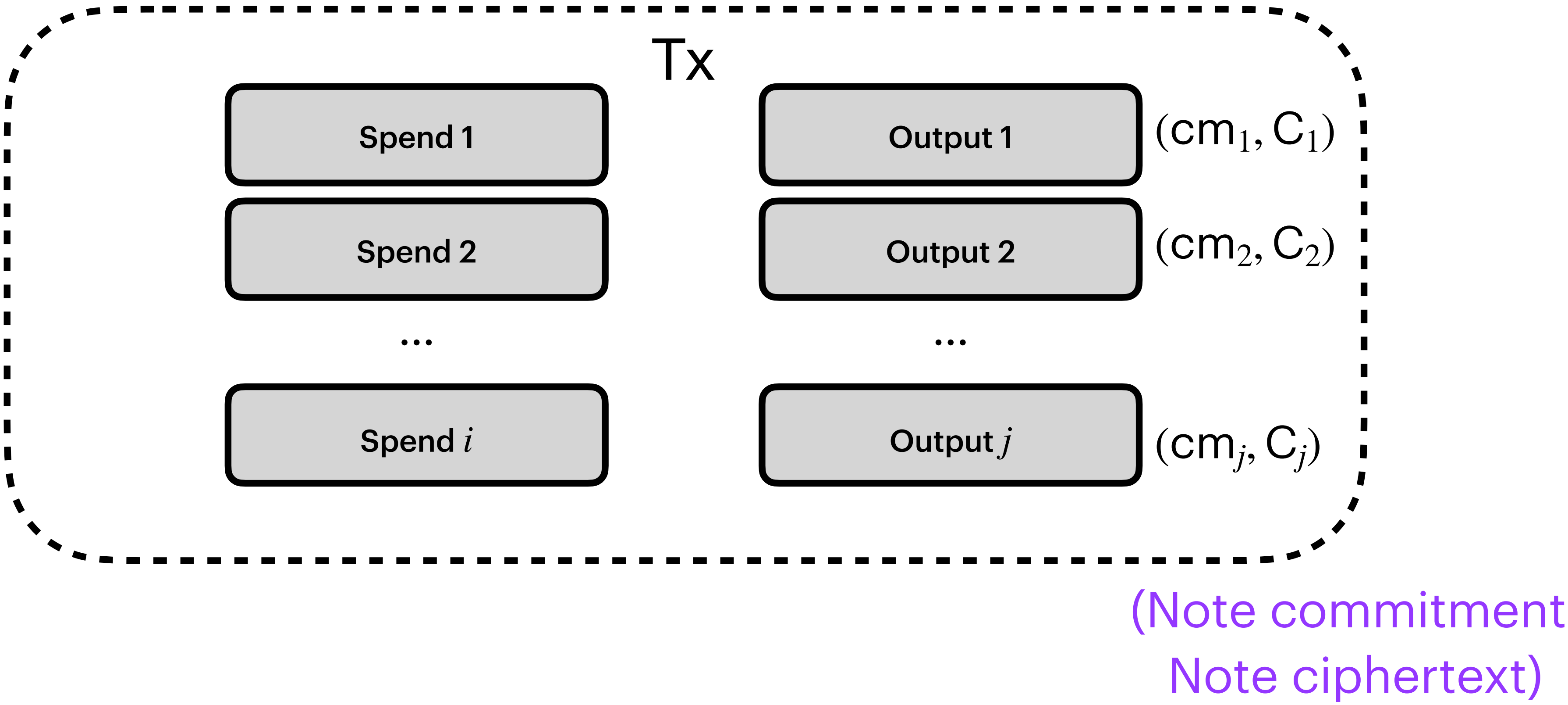
Shielded Transaction Structure

Shielded Transactions consist of *Actions*



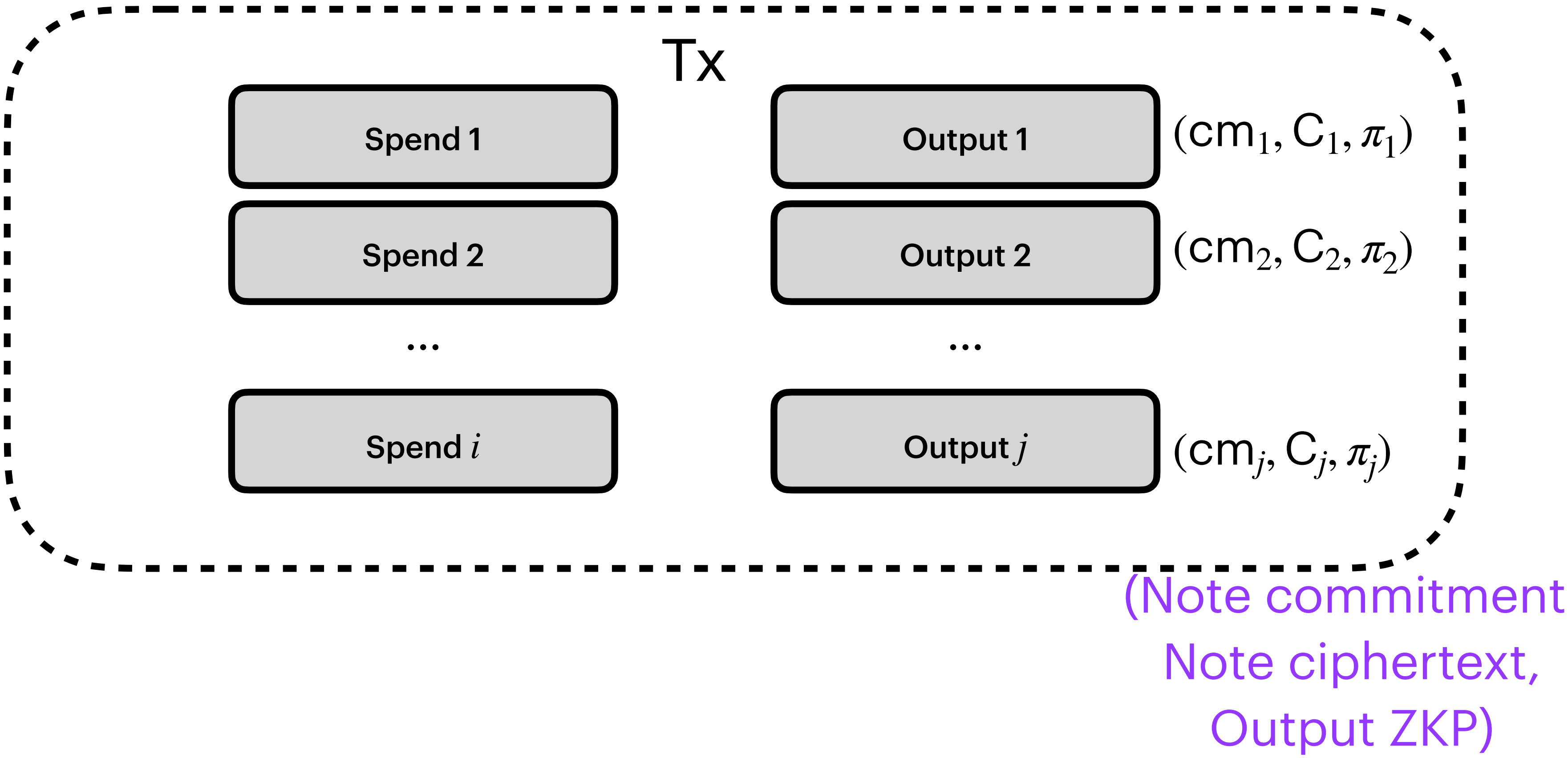
Shielded Transaction Structure

Shielded Transactions consist of *Actions*



Shielded Transaction Structure

Shielded Transactions consist of *Actions*



Output ZKP

π

- *Public inputs*:
  - Note commitment
  - [...]
- *Private witnesses*:
  - Note: Value, address, blinding factor
  - [...]
- *Statements*:
  - The note commitment is well-formed.
  - [...]

Integrity: You cannot spend coins that don't exist

Nodes keep track of all note commitments in the system

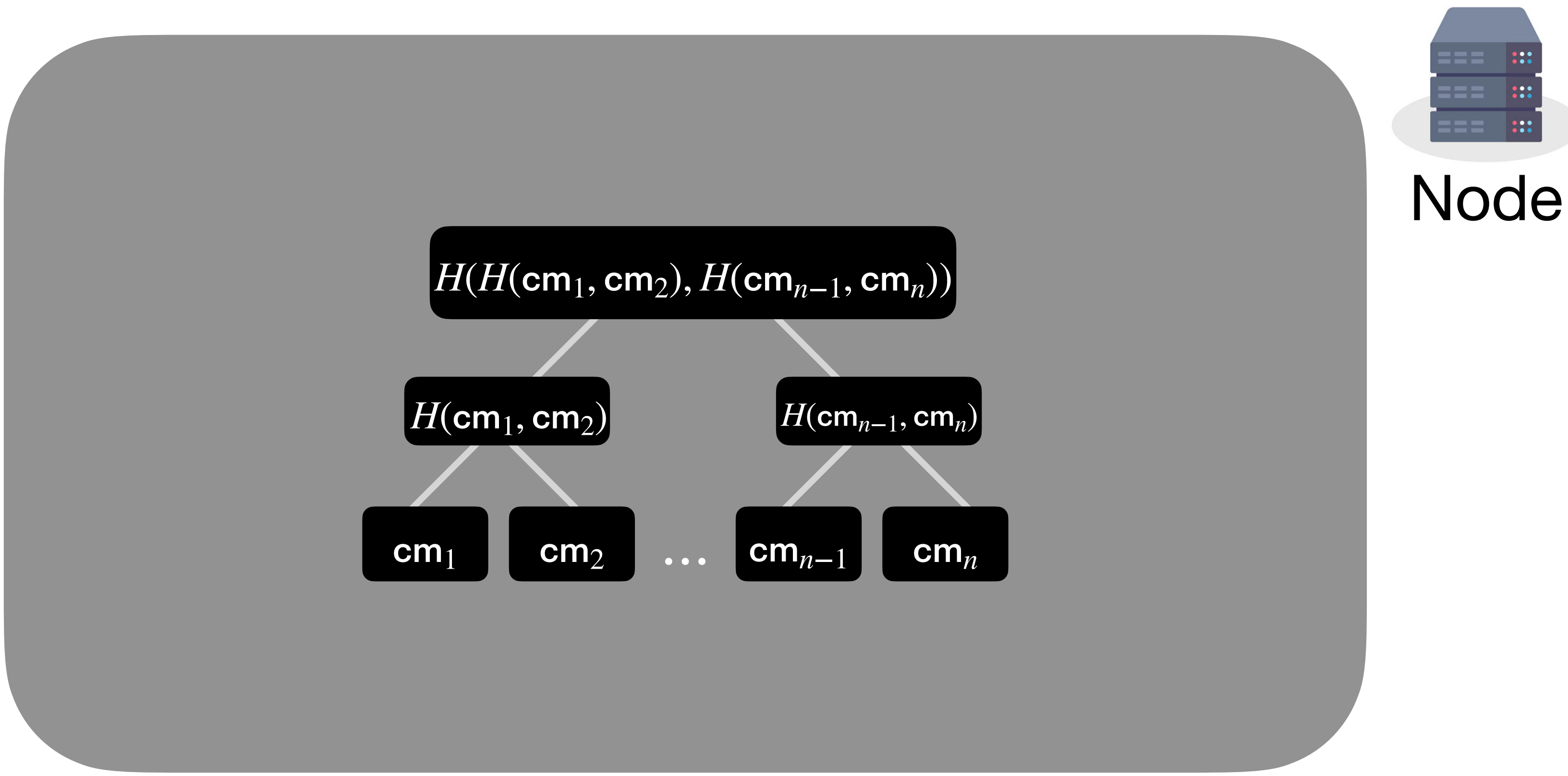
Node

cm_1 cm_2 $\cdots$ cm_{n-1} cm_n

Problem: To spend a note with commitment cm clients need to prove: $(cm = cm_1) \vee (cm = cm_2) \vee \ldots \vee (cm = cm_N)$

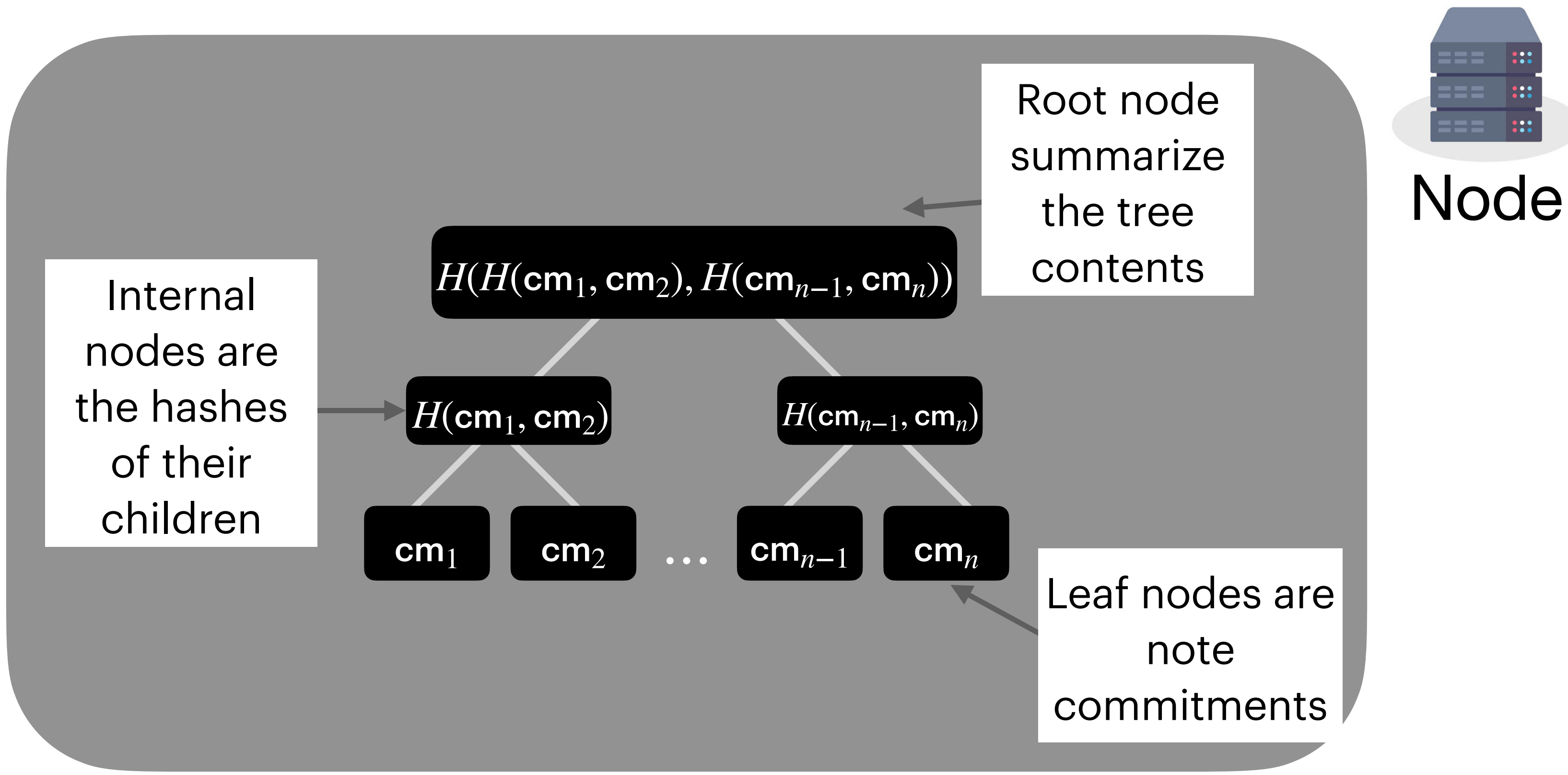
Integrity: You cannot spend coins that don't exist

Nodes maintain an incremental Merkle tree of all note commitments



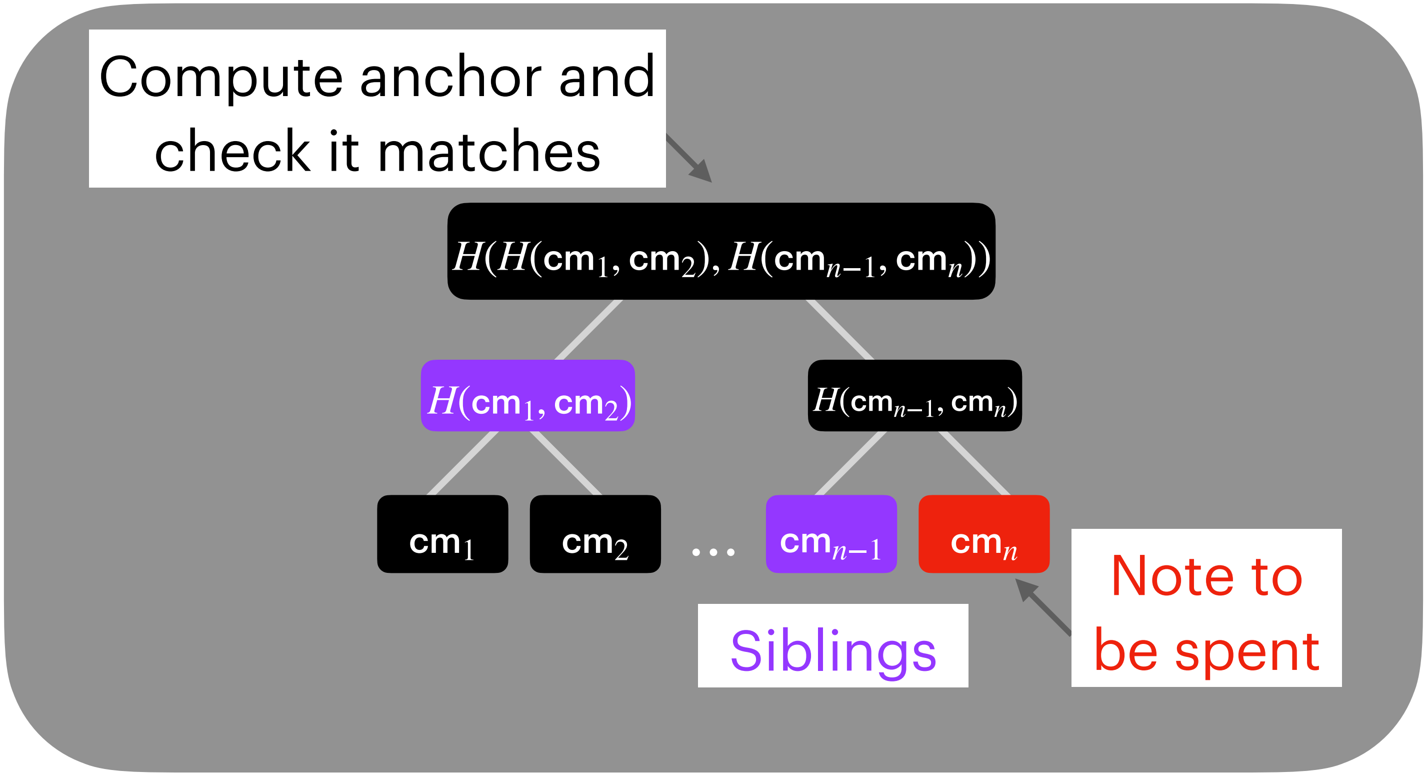
Integrity: You cannot spend coins that don't exist

Nodes maintain an incremental Merkle tree of all note commitments



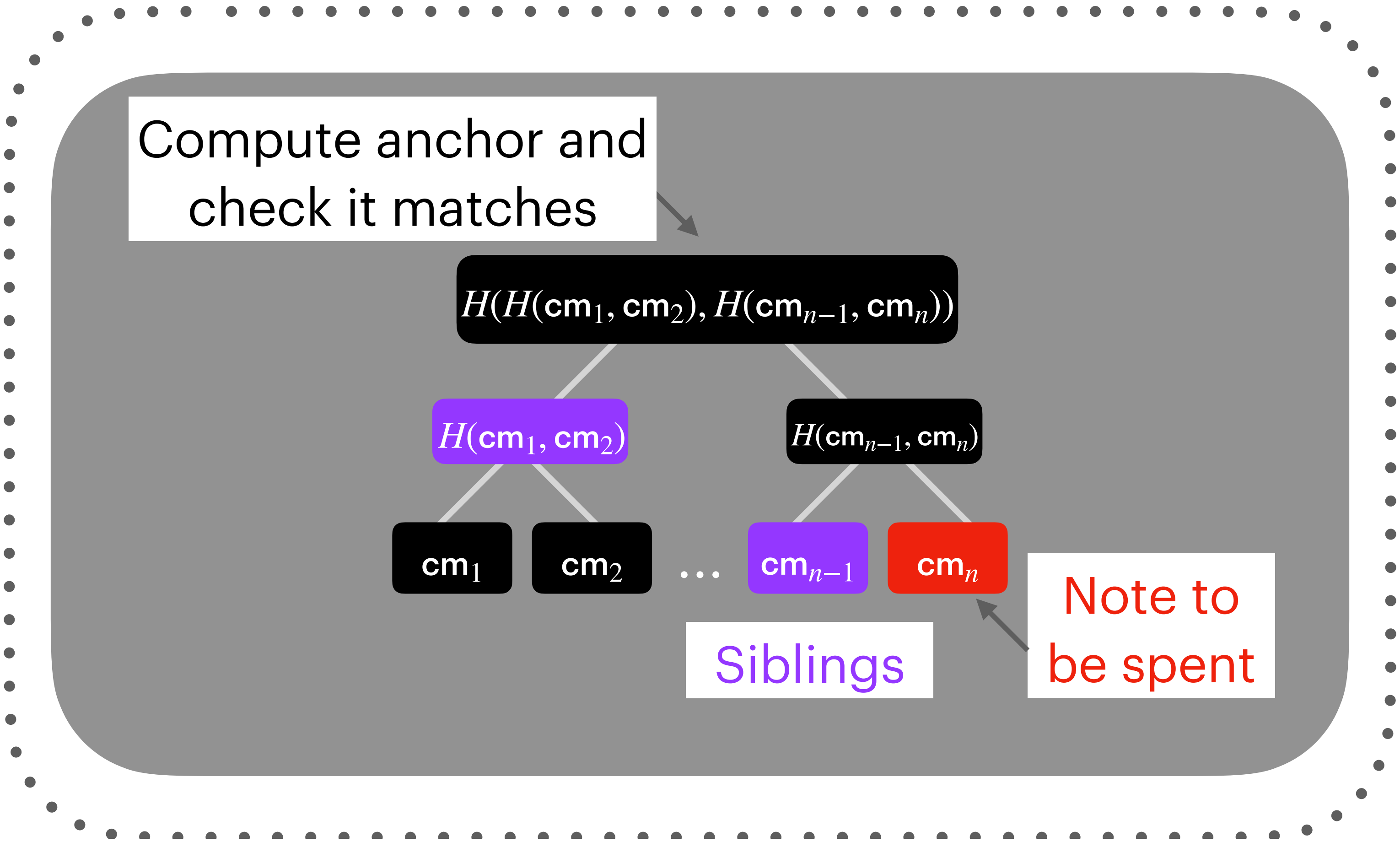
Integrity: You cannot spend coins that don't exist

Merkle proofs let us demonstrate our note is in the system



Integrity: You cannot spend coins that don't exist

When we spend, we provide a Merkle proof in ZK

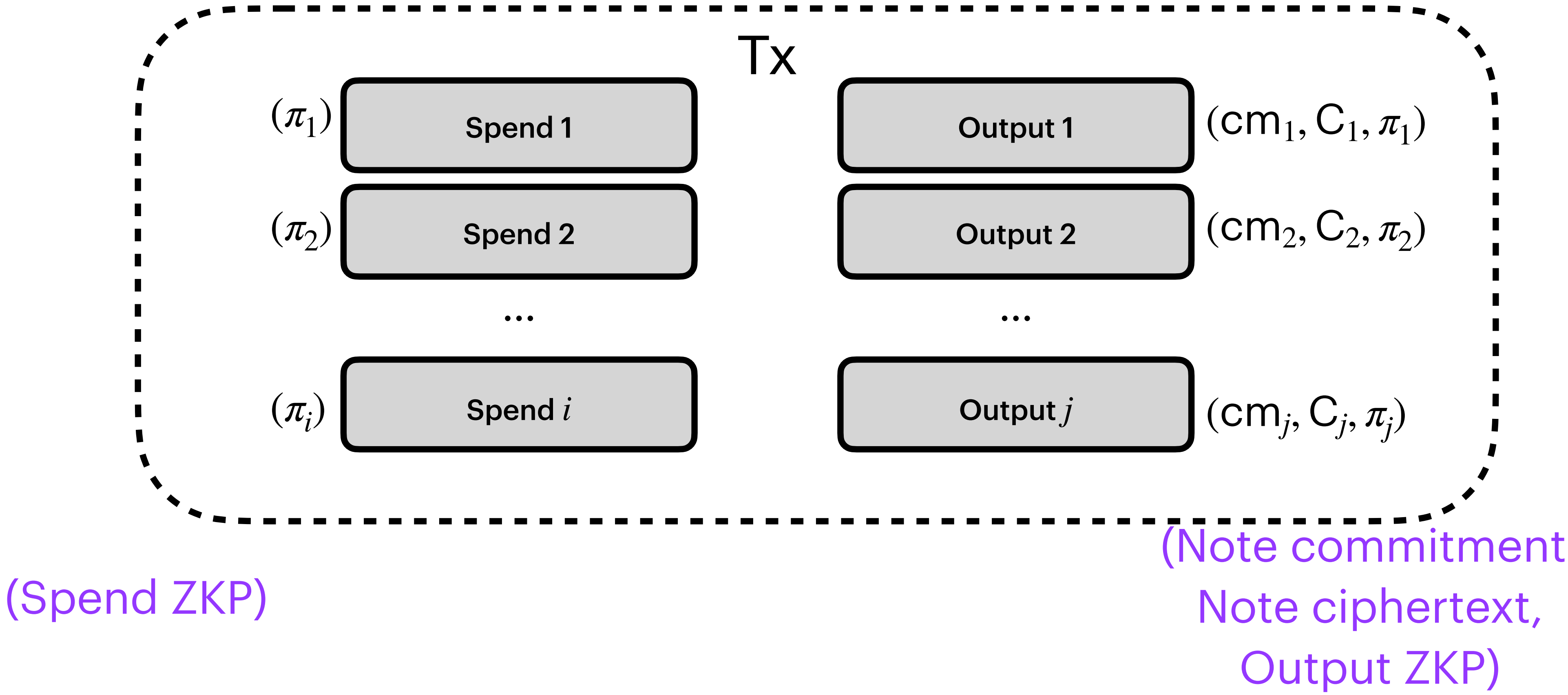


π

- *Public input*: Anchor of merkle tree
- *Witnesses*: Siblings and note commitment

Shielded Transaction Structure

Shielded Transactions consist of *Actions*

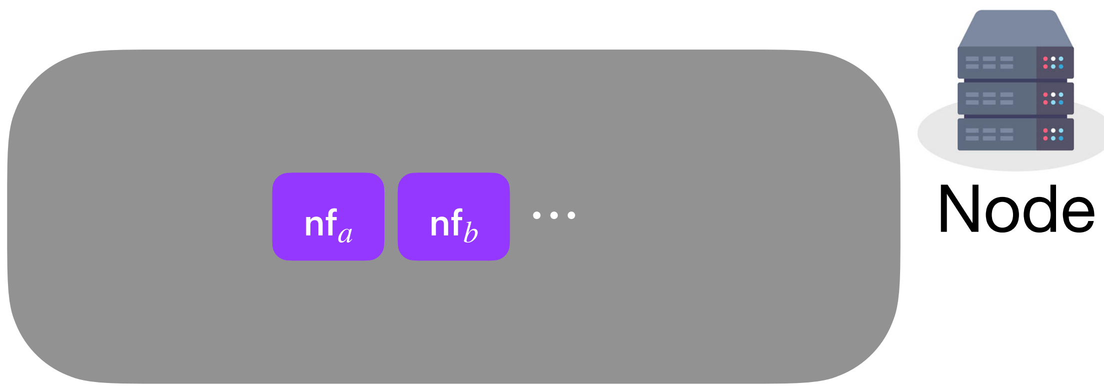


System Invariants
Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities. ✅
- Integrity:
  - You cannot spend other people's coins.
  - You cannot spend coins that don't exist. ✅
  - You cannot spend coins twice.
  - You cannot create or destroy value.

Integrity: You cannot double spend

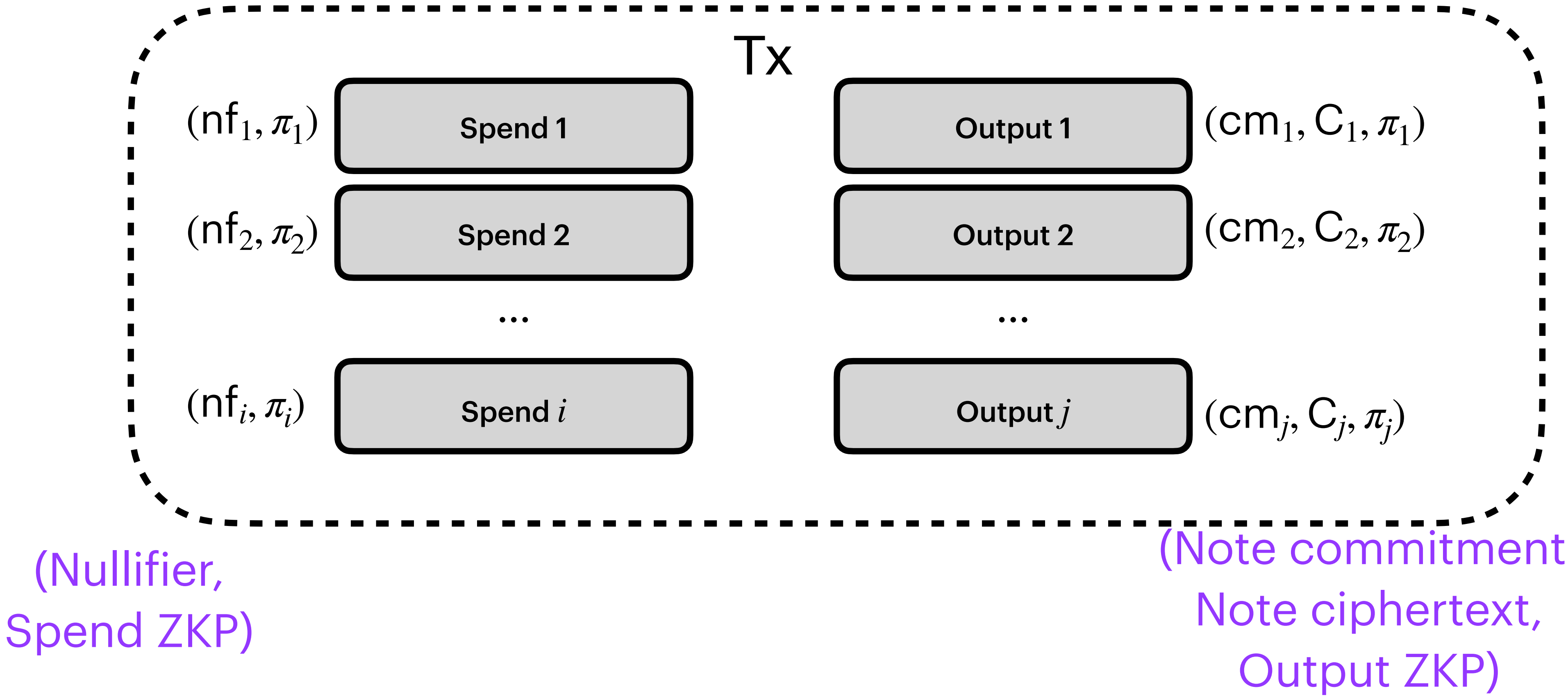
All nodes maintain a nullifier set



- Spend:
  - Reveals the nullifier associated with a note in the system, but does not link to *which* note
  - One-to-one mapping between notes and nullifiers: once revealed, that same note is considered invalid in future spends

Shielded Transaction Structure

Shielded Transactions consist of *Actions*



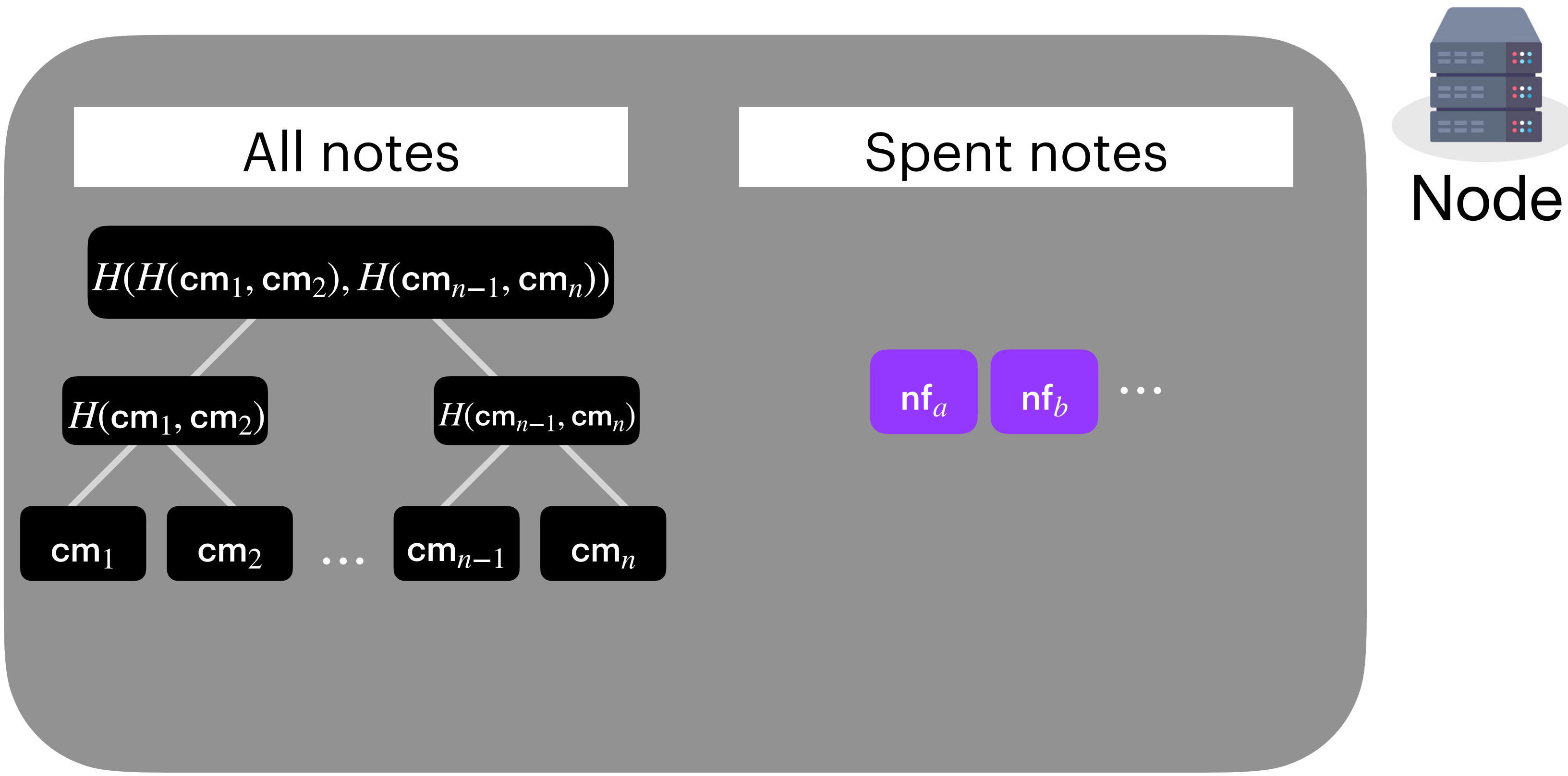
Spend ZKP

π

- *Public inputs*:
  - Anchor of merkle tree
  - Nullifier
  - [...]
- *Private witnesses*:
  - Note: Value, address
  - Merkle proof: Siblings in the tree
  - [...]
- *Statements*:
  - The Merkle proof demonstrates the note commitment exists in the tree with the provided public anchor.
  - The nullifier is correctly derived.
  - [...]

Node State Management

All nodes maintain a Merkle tree of all notes and a nullifier set of spent notes



System Invariants

Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities. ✅
- Integrity:
  - You cannot spend other people's coins.
  - You cannot spend coins that don't exist. ✅
  - You cannot spend coins twice. ✅
  - You cannot create or destroy value.

Cryptographic Primitive: Re-randomizable Signatures

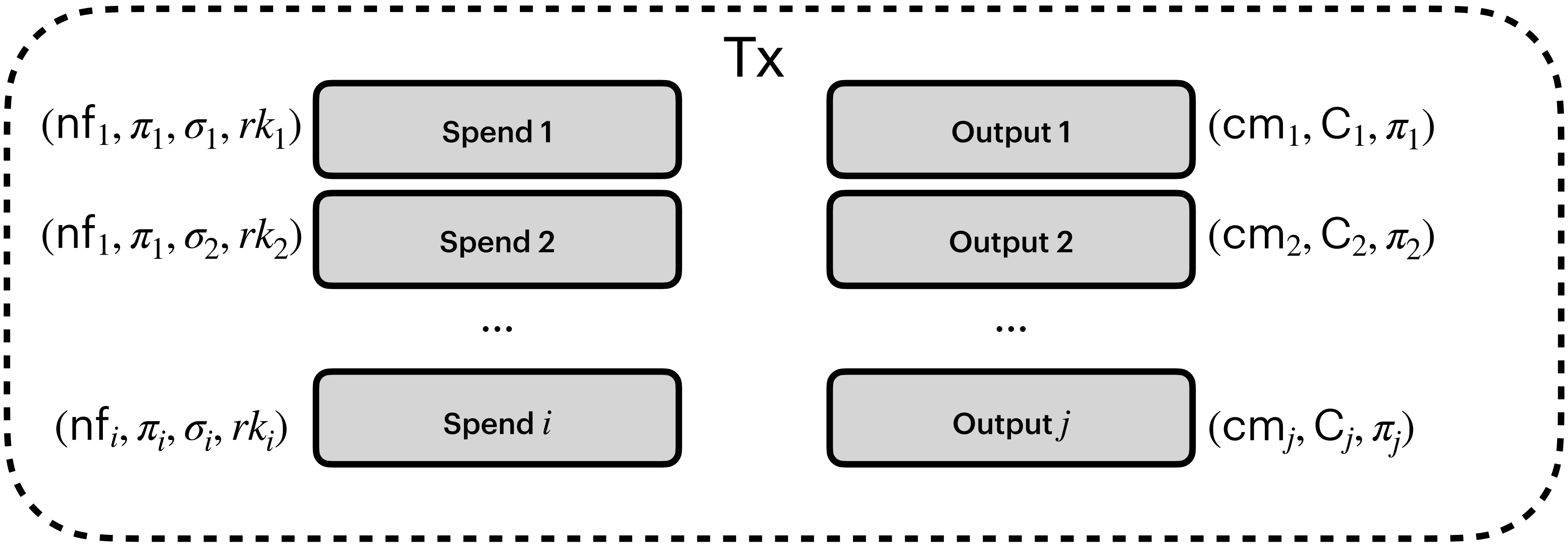
- Why can't we use regular signature schemes in privacy-preserving protocols?
  - Trivial linkage of spends due to trial verification of targeted public keys
- Instead:
  - We derive a one-time use ("randomized") key rk from our real key ak, and use that:

$$rk = ak + [\alpha]B$$

  - We also need to demonstrate in ZK that the randomized key (public on the transaction) is a correct randomization given the witnessed real key ak and randomizer α

Integrity: You cannot spend other people's coins

Re-Randomizable Digital Signatures



(Nullifier, Spend ZKP, Signature, Randomized Verification Key)

(Note commitment, Note ciphertext, Output ZKP)

Spend ZKP

- *Public inputs*:
  - Anchor of merkle tree
  - Nullifier
  - Randomized spend verification key rk
  - [...]
- *Private witnesses*:
  - Note: Value, address
  - Merkle proof: Siblings in the tree
  - Spend authorization key
  - Randomizer
  - [...]
- *Statements*:
  - The Merkle proof demonstrates the note commitment exists in the tree with the provided public anchor.
  - The nullifier is correctly derived.
  - The randomized spend verification key is correctly derived from the witnessed spend authorization key and randomizer
  - [...]

π

System Invariants

Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities. ✅
- Integrity:
  - You cannot spend other people's coins. ✅
  - You cannot spend coins that don't exist. ✅
  - You cannot spend coins twice. ✅
  - You cannot create or destroy value.

Cryptographic Primitive: Commitment Schemes

- Two phases: Commit and Reveal
- Commit:
  - cm $=$ Commit(m, randomness)

- Reveal:
  - Verify(m, cm, randomness) = [accept, reject]

Cryptographic Primitive: Commitment Schemes
Pedersen Commitments

- Two phases: Commit and Reveal
  - Commit: $cm = [m]G + [\text{randomness}]H$

- Reveal:
  - $cm == [m]G + [\text{randomness}]H$

Cryptographic Primitive: Commitment Schemes

Additive Homomorphism

- Pedersen commitments are additively homomorphic.
- Given two commitments:

$$cm_1 = \text{Commit}(m_1, \text{randomness}_1) = [m_1]G + [\text{randomness}_1]H$$

$$cm_2 = \text{Commit}(m_2, \text{randomness}_2) = [m_2]G + [\text{randomness}_2]H$$

- The addition $cm_1 + cm_2$ is:

$$\begin{aligned}
&cm_1 + cm_2 \\
&= [m_1]G + [\text{randomness}_1]H + [m_2]G + [\text{randomness}_2]H \\
&= [m_1 + m_2]G + [\text{randomness}_1 + \text{randomness}_2]H \\
&= \text{Commit}(m_1 + m_2, \text{randomness}_1 + \text{randomness}_2)
\end{aligned}$$

Cryptographic Primitive: Commitment Schemes

Value Commitments

- Add value commitment cv to each action in the tx:

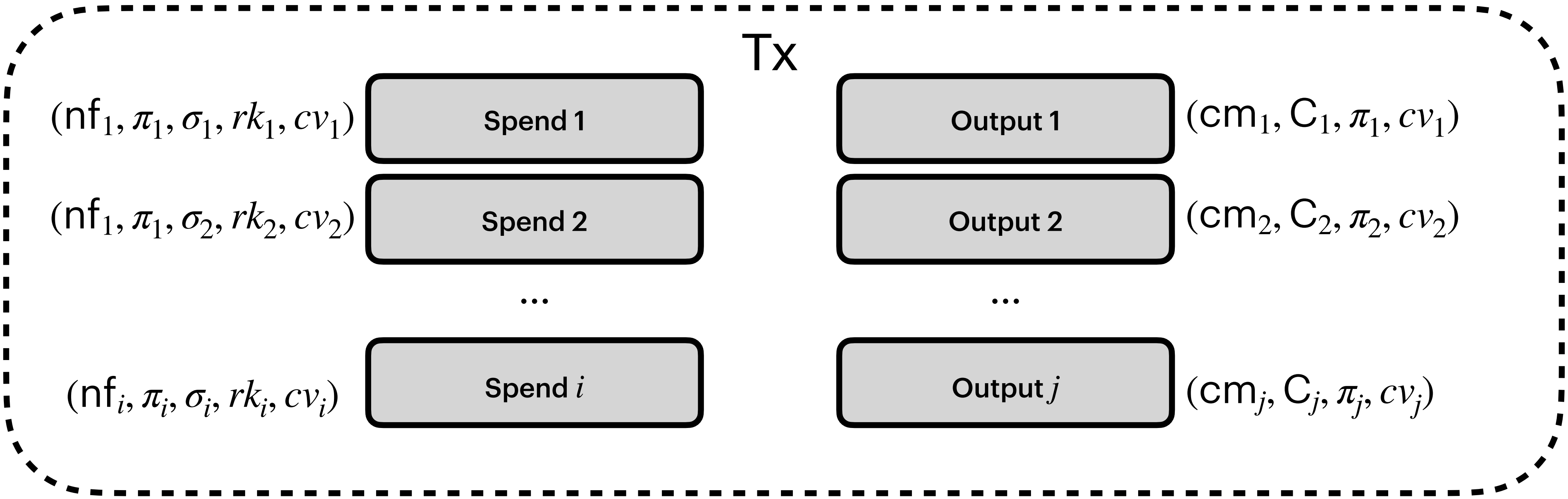
$$cv = \boxed{[v]}G + [\text{randomness}]H$$

Amount, where:

- Spend value is positive (releasing value into the tx)
- Output value is negative (consuming value from the tx)

Integrity: You cannot create or destroy value.

Additively homomorphic value commitments let us verify value balance is 0

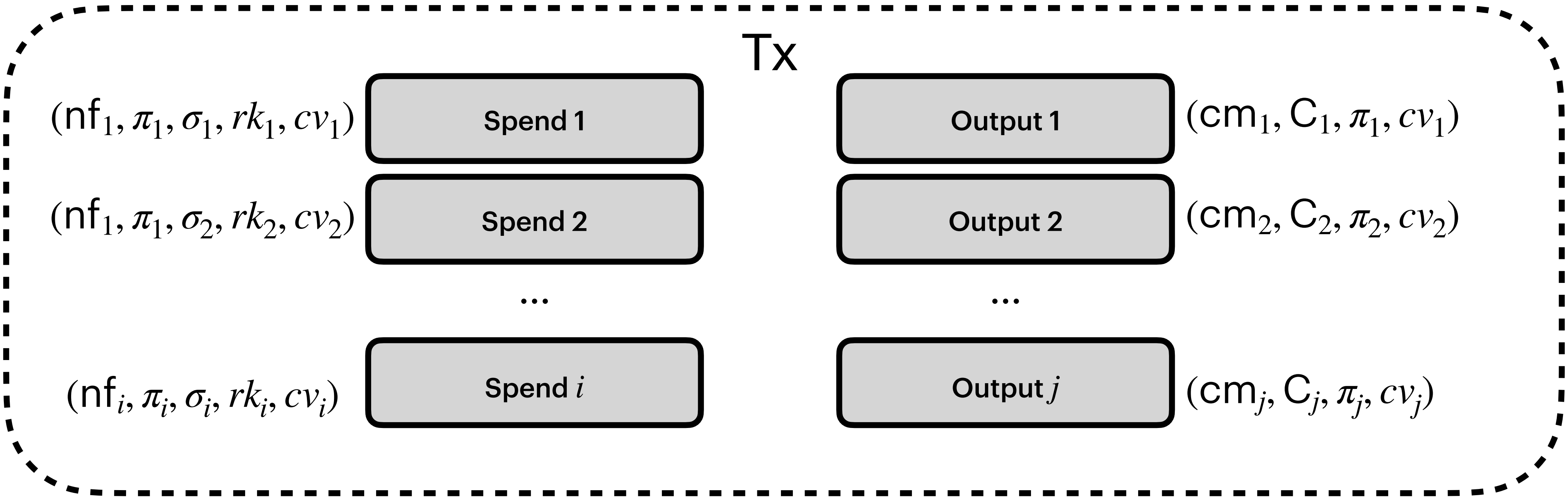


(Nullifier, Spend ZKP, Signature, Randomized Verification Key, Value commitment)

(Note commitment, Note ciphertext, Output ZKP, Value commitment)

Integrity: You cannot create or destroy value.

Additively homomorphic value commitments let us verify value balance is 0



$$\sum_i \sum_j (cv_{i,j}) \stackrel{?}{=} \mathsf{Commit}(0, \sum_i \sum_j (\mathsf{randomness}_{i,j}))$$

System Invariants

Privacy and Integrity

- Private:
  - A passive observer should not learn anything about values, sender or recipient identities. ✅
- Integrity:
  - You cannot spend other people's coins. ✅
  - You cannot spend coins that don't exist. ✅
  - You cannot spend coins twice. ✅
  - You cannot create or destroy value. ✅

Summary

How to do decentralized anonymous payments

- All state (notes) is **encrypted** on the blockchain.
  - The blockchain never sees recipient, sender, or value.
- **Note commitment tree** (incremental Merkle tree) maintains an **append-only** data store of all state in the system.
  - Spends of a note must demonstrate in ZK the note commitment is in the commitment tree.
- State is nullified/deleted by revealing a **nullifier** (once, double spend protection) that goes into the **nullifier set**. Observers cannot link nullifier to notes/state that was invalidated.
- Spends also must demonstrate control of the note via a **randomized signature**.
- Value conservation is provided through the **additively homomorphic** property of **value commitments**.

Extra