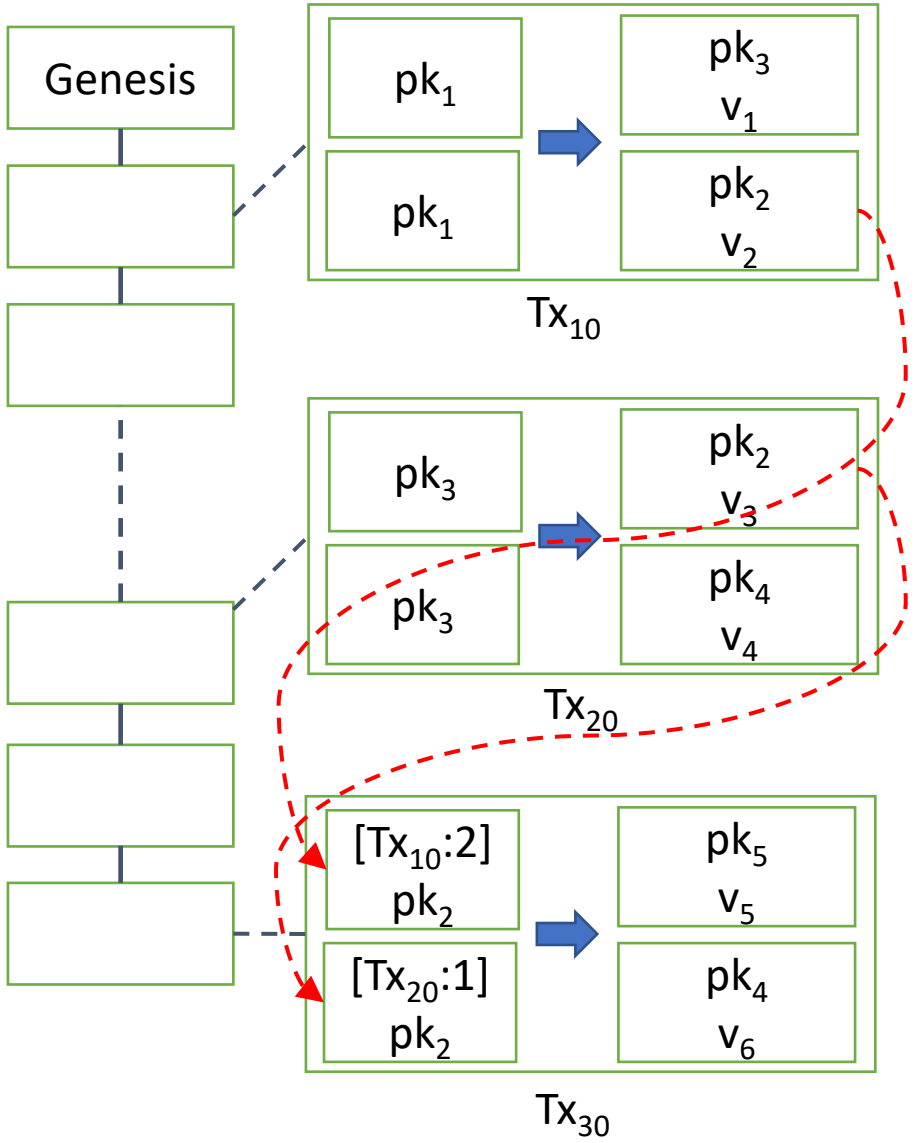
Lecture 18: Data Privacy via ZK Cryptography

https://web3.princeton.edu/principles-of-blockchains/

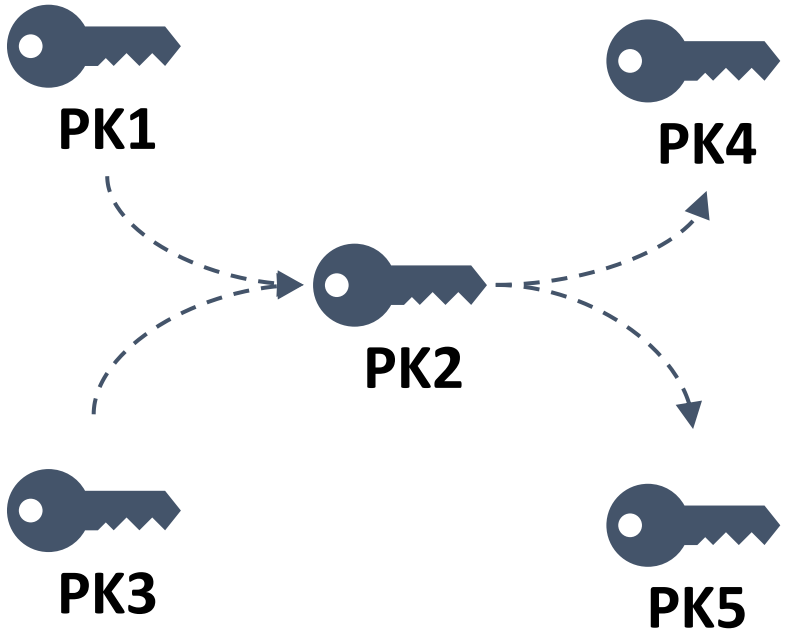
Professor Pramod Viswanath
Princeton University

This lecture:
Zero knowledge (ZK) cryptography library
Zcash architecture – Bitcoin + Data privacy

UTXO Model

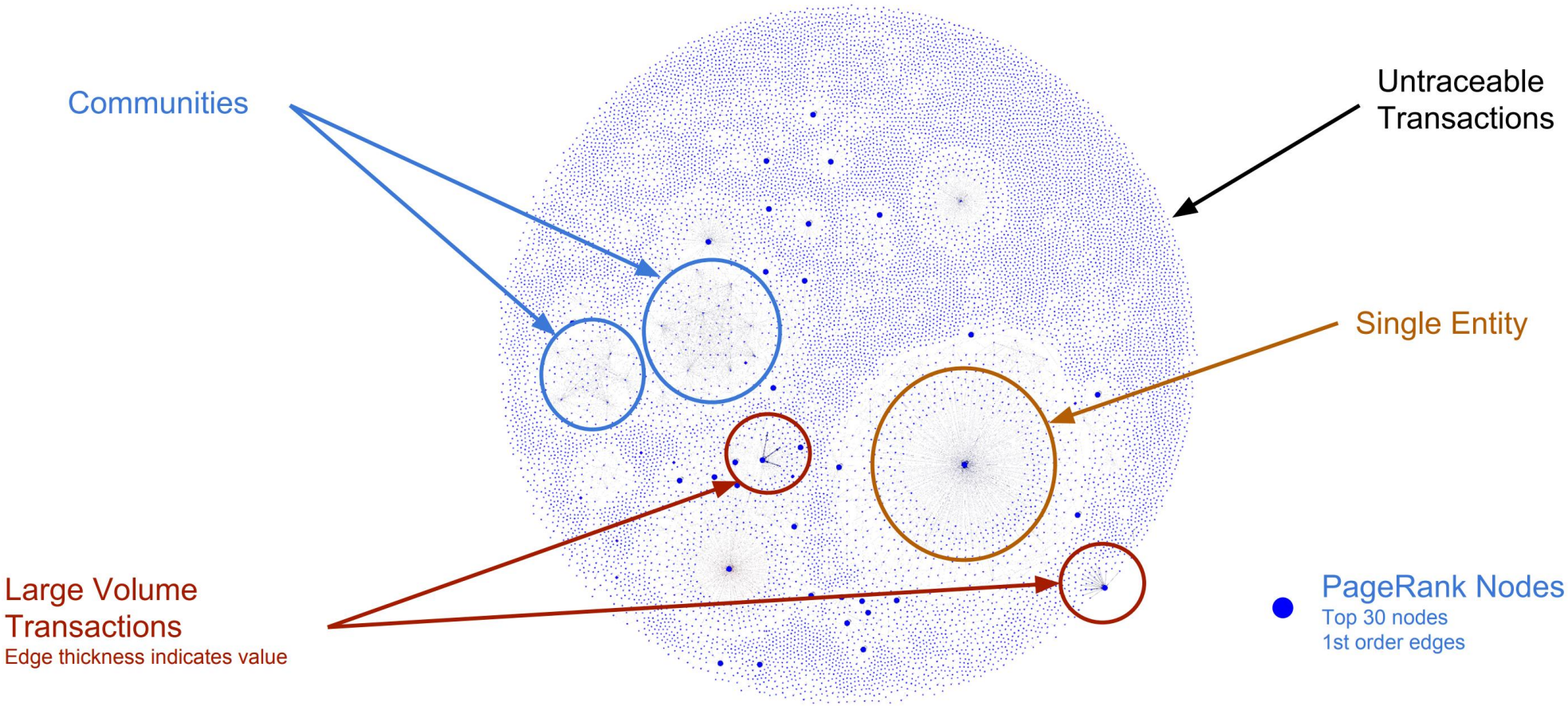


- Bitcoin: Pseudonymous
  - Privacy: public key
  - One can create many public keys



Transaction / public key graph

Transaction Graph allows Deanonymization



Typical transaction graph for a day

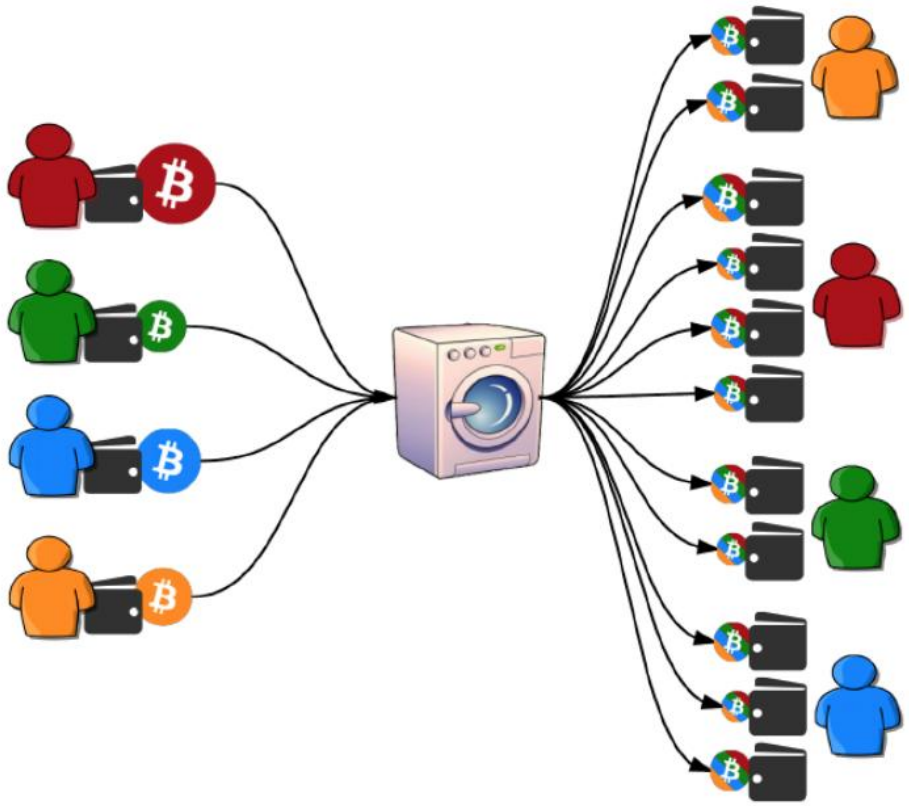
https://arxiv.org/pdf/1502.01657.pdf

Commercial services built around forensics

- **Chainalysis**
- Used by financial institutions and nation states
- Recover losses from malware attacks
  - Wannacry ransomware, 2017
- Part good (when the “bad guys” are tracked) but
- Part weird (when “routine transactions” are exposed willy-nilly)

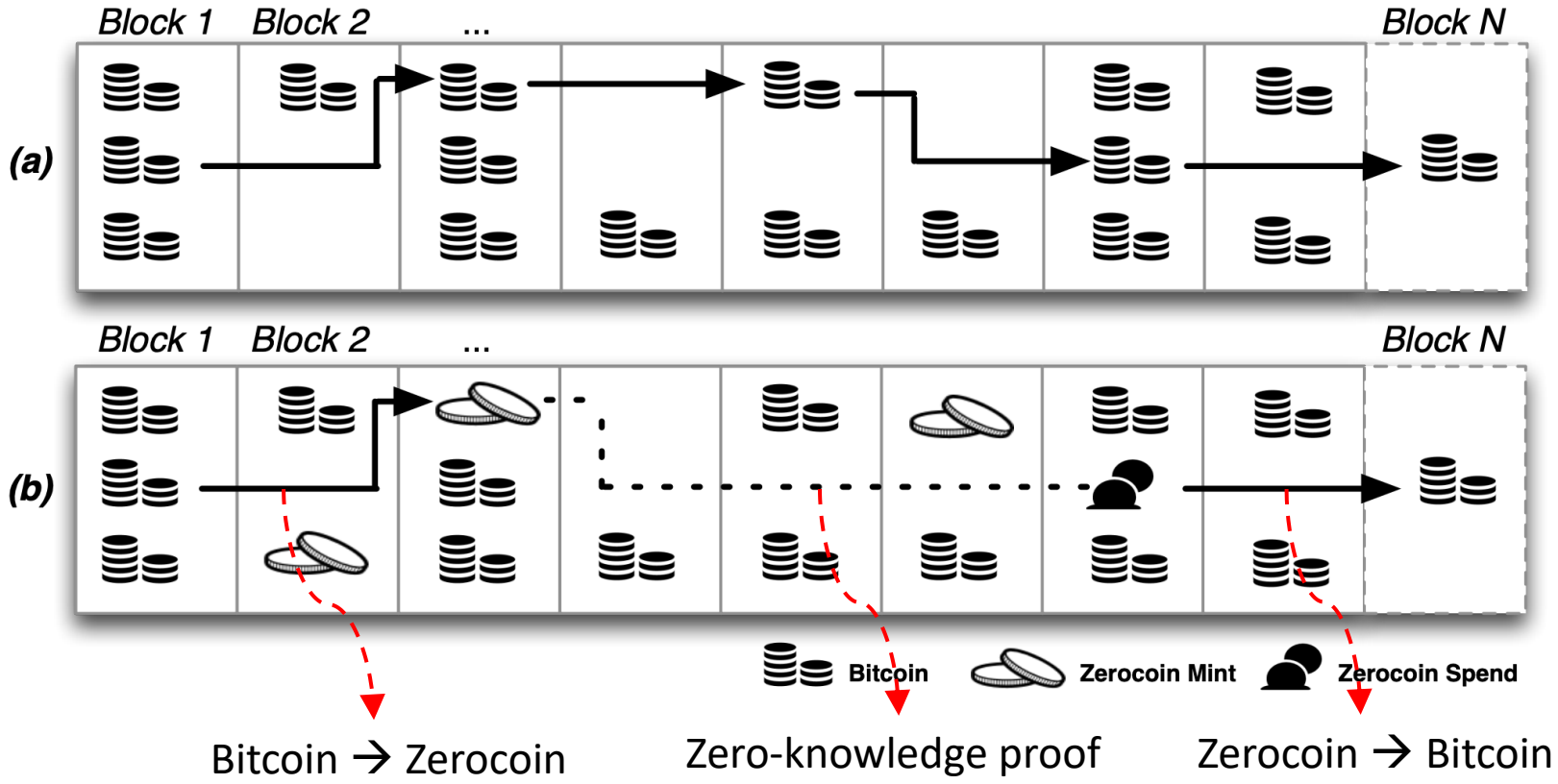
Trusted Third-party Mixer

- Laundry service
- Exchanges the coins (the public keys)
- Centralized third-party
  - The mixer can trace / steal the coins
- Example: Coinbase
  - Offers different public keys for each transaction



Zerocoin (2013)

- Decentralized laundry system
  - Large overhead, doesn't allow transact, split, aggregate zerocoins

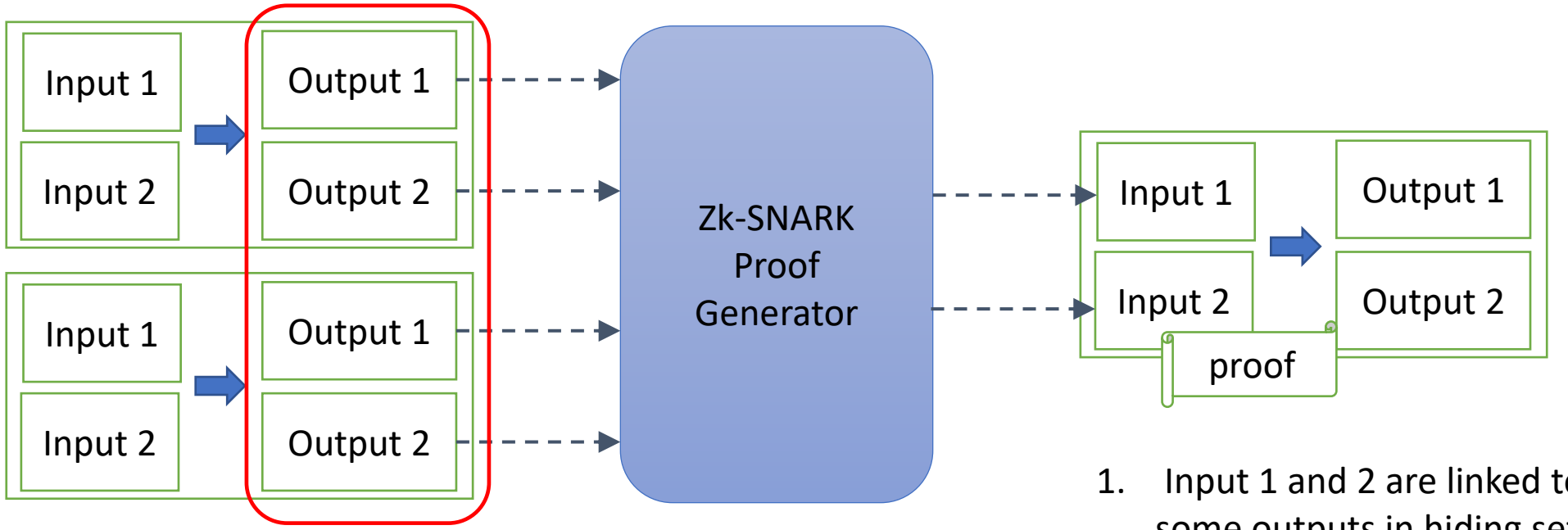


Zcash (2014)

- Extends Bitcoin’s protocol
  - New transaction types
- More information hiding
  - payment's origin, destination, and amount
- Separate anonymous currency
  - Zerocoin vs Basecoin
  - Support split, merge, transact zerocoins
- Use new cryptographic primitives: Zk-SNARK libraries
  - *Zero-knowledge Succinct Non-interactive ARguments of Knowledge*
  - Short and easy to verify

Zcash (2014)

- Special transaction



Hiding set

1. Input 1 and 2 are linked to some outputs in hiding set
2. Value of input 1,2 is no more than those in the outputs

Zk-SNARK: zero knowledge non-interactive succinct argument of knowledge

Language and NP

Definition 1 (Language) A ***language*** L is a set of statements s.t.

$$L(x) = 1 \text{ if } x \in L$$

Example: x is a number, $L(x)$ is an indicator of whether x is composite

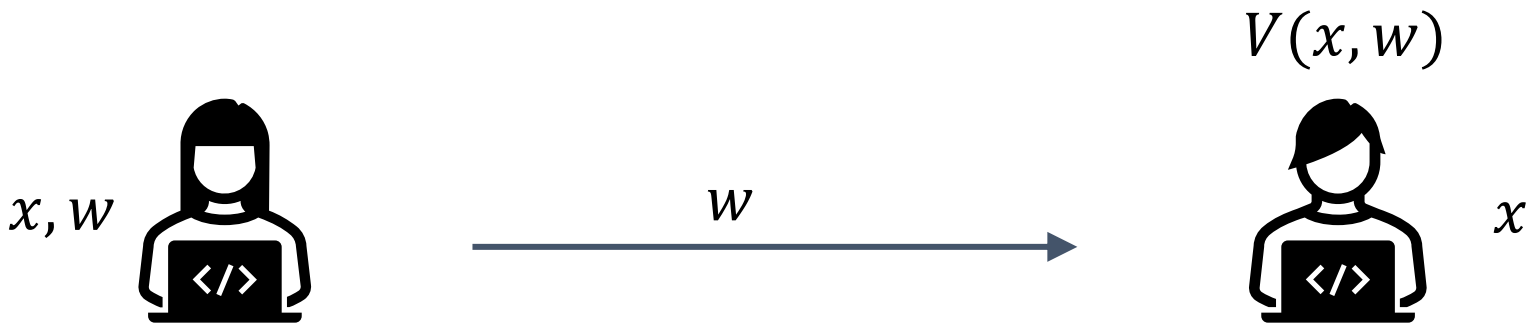
Definition 2 (NP) We define ***NP*** to be the class of languages L that have a polynomial time ***Verifier*** V, s.t.

$$L(x) = 1 \iff \exists w, \text{s.t. } \mathrm{V}(x, w) = 1$$

Example: x is a number, $L(x)$ is an indicator of whether x is composite, witness is the prime factorization of x, and verifier can testify the product of w in polynomial time.

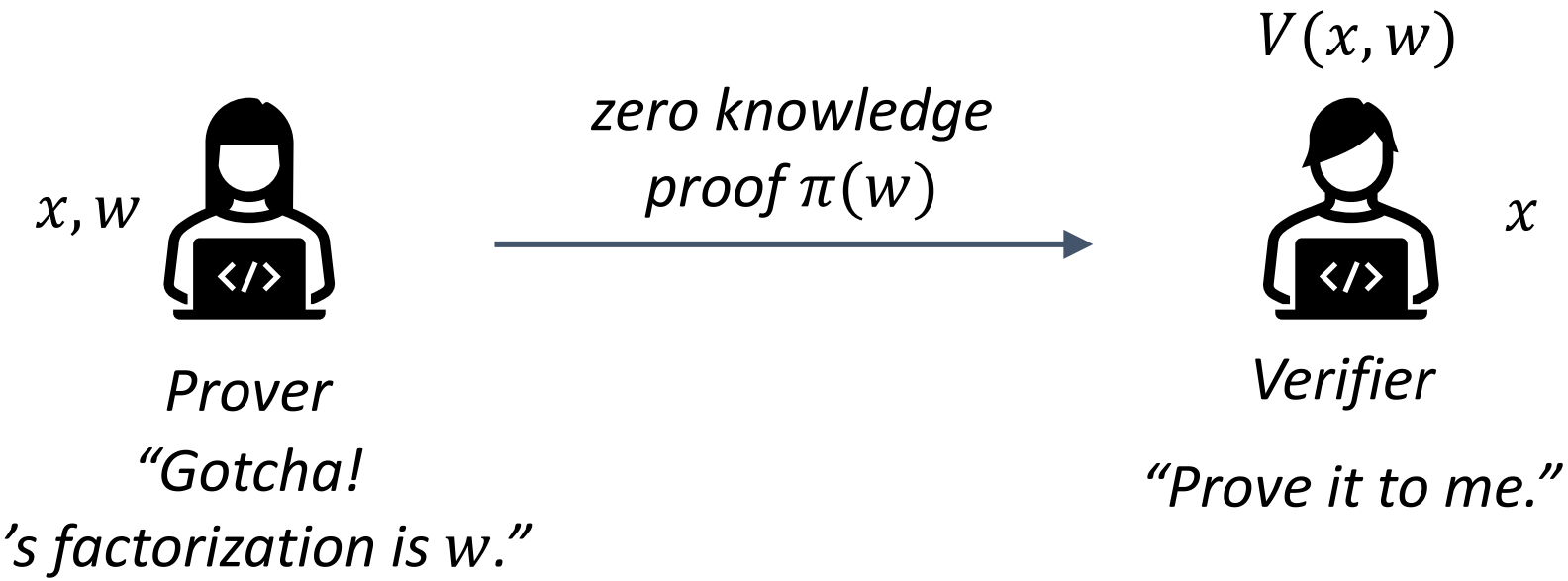
Prover and Zero-knowledge

Given a language L in NP, e.g., verify whether x is a composite.



Prover and Zero-knowledge

Given a language L in NP, e.g., verify whether x is a composite.

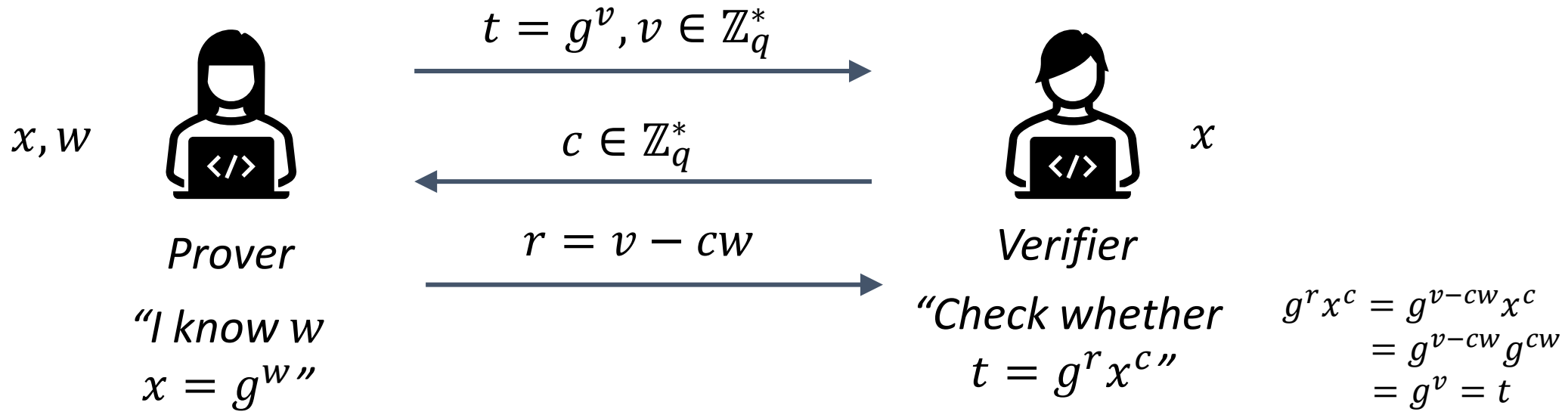


Security Requirements

- Completeness
  - An honest prover with a valid witness can always convince an honest verifier
- Soundness
  - An honest verifier cannot accept a proof if $x \notin L$
- Zero-knowledge
  - The proof does not reveal any information about w

Zero-knowledge Proof

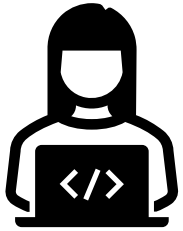
- Schnorr DLOG - Given $x = g^w$, find w.
  - g: a generator of a cyclic group with prime-order q



Zero-knowledge Proof

- Schnorr DLOG - Given $x = g^w$, find w.
  - g: a generator of a cyclic group with prime-order q

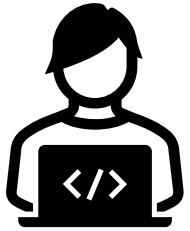
$t = g^v, v \in \mathbb{Z}_q^*$
$c = \mathcal{H}(g, x, t)$
$r = v - cw$



Prover

"I know w
$x = g^w$*"*

(t, r)



x

Verifier

"Check whether
$t = g^r x^c$*"*

Zero-knowledge Proof

- Time to generate the proof
  - Running time of $V(x, w)$ is T
  - Efficient generator $O(T \log T)$
- Time to verify the proof
  - $O(\log T)$ theoretically
  - Much worse in practice

Zero-knowledge Proof

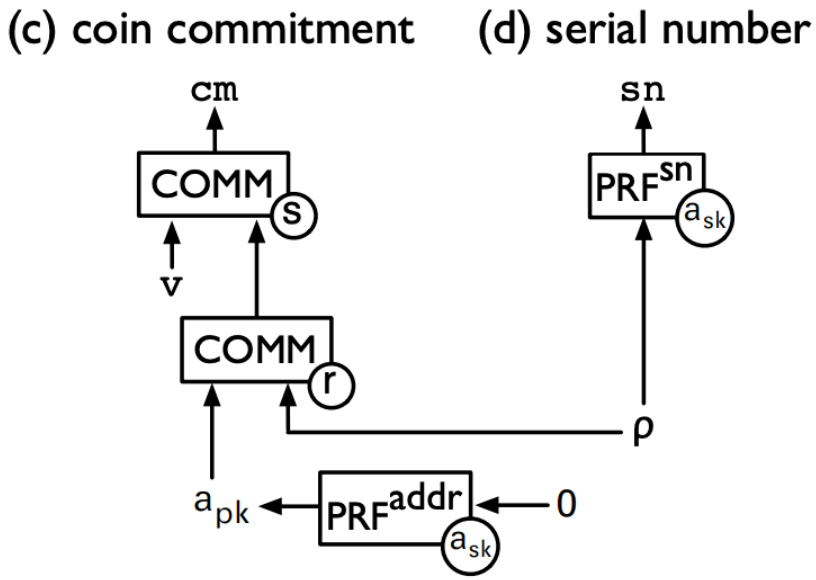
- Proving process
  - Interactive: earlier example
  - Non-interactive: zk-SNARK / zk-STARK
    - Zk-SNARK: trusted setup
    - Zk-STARK: transparent

- **Remarkable Fact**:

  All languages in NP possess zero-knowledge proofs.

Data Structures

- Address
  - $\mathrm{addr}_{\mathrm{pk}}$, $\mathrm{addr}_{\mathrm{sk}}$
- Coin c
  - Coin commitment $\mathrm{cmt}(c)$
  - Coin value $v(c)$
  - Coin serial number $\mathrm{sn}(c)$
  - Coin address $\mathrm{addr}_{\mathrm{pk}}(c)$



- Pour transaction
  - $(rt, sn_1^{old}, sn_2^{old}, cmt_1^{new}, cmt_2^{new}, v_{pub}, \mathrm{info}, \mathrm{proof})$

Problem Statement

- A pour transaction

$$x = (rt, sn_1^{old}, sn_2^{old}, cmt_1^{new}, cmt_2^{new}, v_{pub}, \text{info}, \text{proof})$$

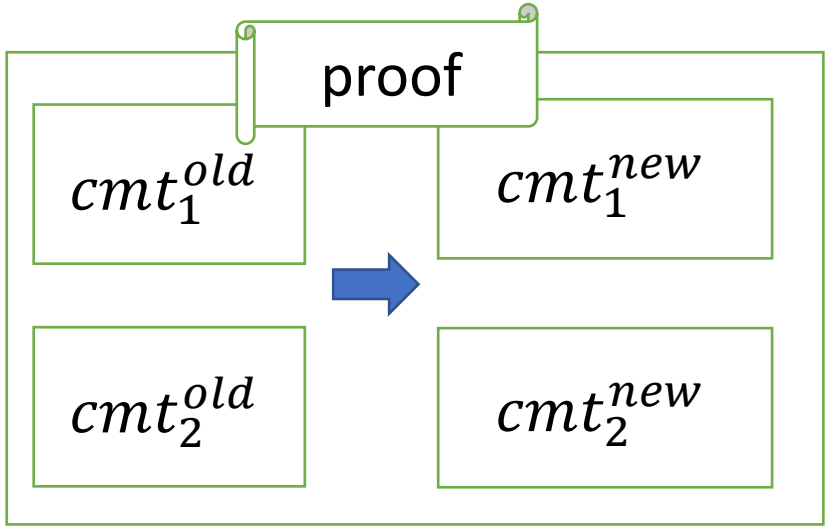
  - Consume two old coins, c_1^{old}, c_2^{old}
  - Create two new coins, c_1^{new}, c_2^{new}
  - Not reveal the information of coins (e.g., public keys)
- Zk-SNARK proof
  - NP statement "*Given the Merkle-tree root rt, serial number , sn_1^{old} , sn_2^{old} , and coin commitments cmt_1^{new}, cmt_2^{new}, I know coins c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new} and address secret key and they satisfy the following conditions…*"

Problem Statement

- A pour transaction x
- $L(x) = 1$: x is a valid transaction
- Witness $w = \left(c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}, \mathrm{addr}_{sk}\left(c_1^{old}\right), \mathrm{addr}_{sk}\left(c_2^{old}\right)\right)$
- Problem: design the structure of transaction such that no information about the coins is leaked
  - E.g., transaction relationship, public keys, values etc.

First Attempt: commitment

- $x = \left(cmt_1^{old}, cmt_2^{old}, cmt_1^{new}, cmt_2^{new}, \text{proof}\right)$
- Not reveal values, public keys etc.



First Attempt: commitment

Definition 2 (NP) We define ***NP*** to be the class of languages L that have a polynomial time ***Verifier*** V, s.t.

$$L(x) = 1 \iff \exists w, \text{s.t. } \mathrm{V}(x, w) = 1$$

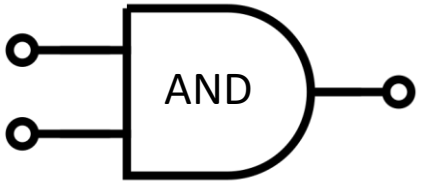
- $x = \left(cmt_1^{old}, cmt_2^{old}, cmt_1^{new}, cmt_2^{new}, \text{proof}\right)$
- Witness $w = \left(c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}, \text{addr}_{sk}\left(c_1^{old}\right), \text{addr}_{sk}\left(c_2^{old}\right)\right)$
- $V(x, w) = 1$ if
  - The commitments of four coins are correct
  - $v\left(c_1^{old}\right) + v\left(c_2^{old}\right) \geq v(c_1^{new}) + v(c_2^{new})$
  - addr_{sk} of two old coins matches addr_{pk}

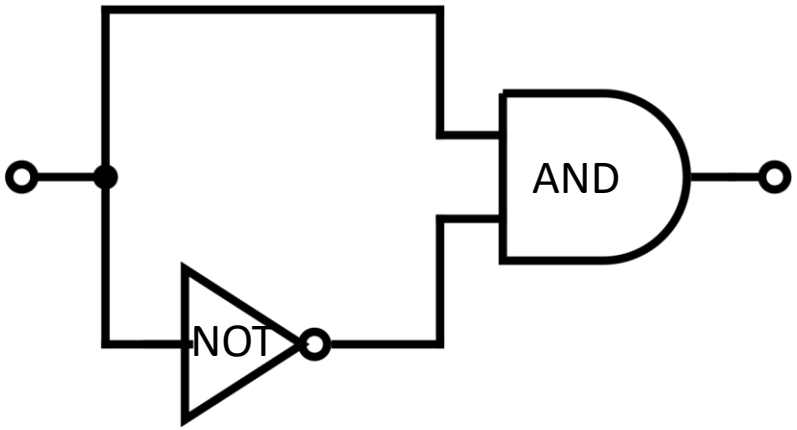
First Attempt: commitment

- Zk-SNARK proof
  - *"Given coin commitments $cmt_1^{old}, cmt_2^{old}, cmt_1^{new}, cmt_2^{new}$, I know coins $c_1^{old}, c_2^{old}, c_1^{new}, c_2^{new}$ and address secret keys and they satisfy the following conditions…"*
  - The sender knows the two old coins and new coins
  - The coins satisfy $v\left(c_1^{old}\right) + v\left(c_2^{old}\right) \geq v(c_1^{new}) + v(c_2^{new})$
  - The sender has access to $\mathrm{addr}_{\mathrm{sk}}(c_1^{old})$ and $\mathrm{addr}_{\mathrm{sk}}(c_2^{old})$

First Attempt: commitment

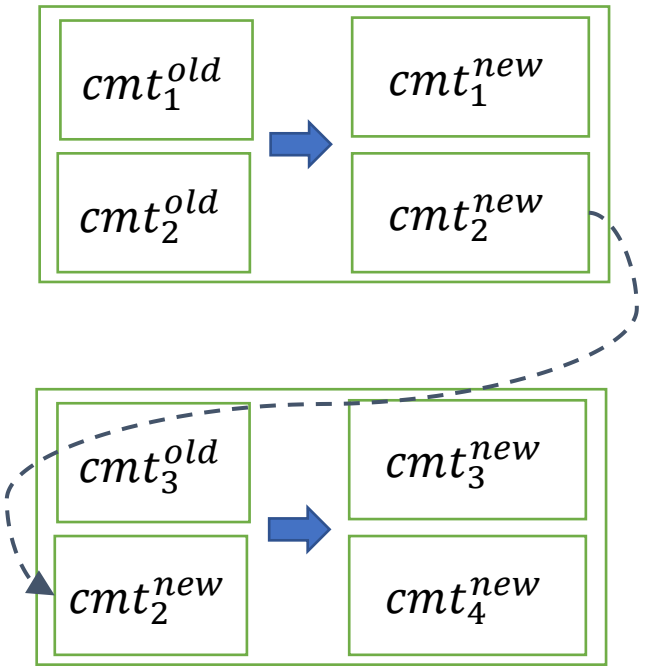
- Constructions of Zk-SNARK proof
  - Solved by satisfiability circuit
  - A triple of polytime algorithms (KeyGen, Prove, Verify)





First Attempt: commitment

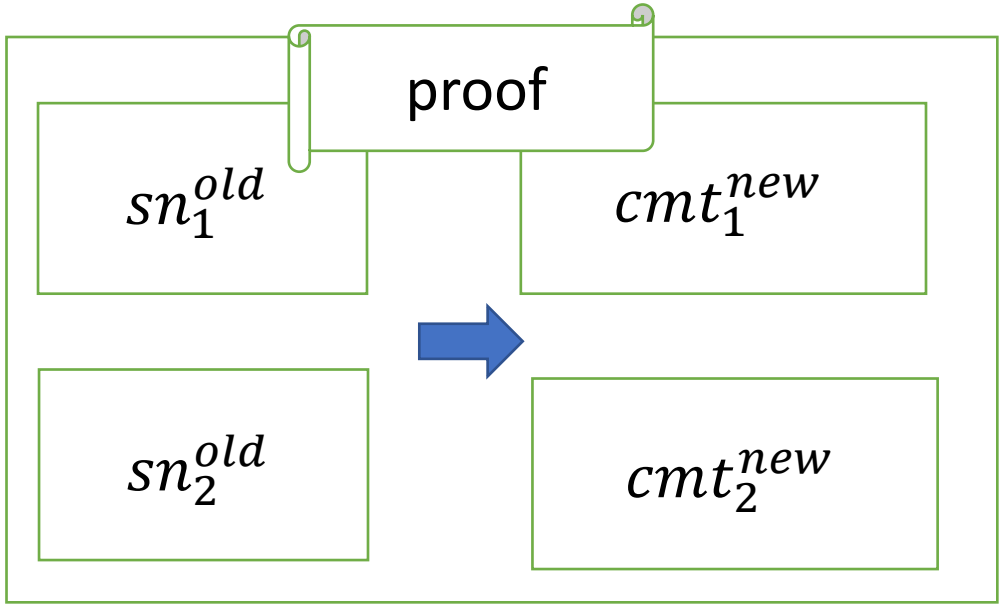
- Anyone who has access to w can create the proof
- Anyone who has access to x can verify the proof



Commitments are still traceable

Second Attempt: two commitments

- Two types of commitments
  - commitment and serial number
- $x = (sn_1^{old}, sn_2^{old}, cmt_1^{new}, cmt_2^{new}, \text{ proof}, \text{ rt})$

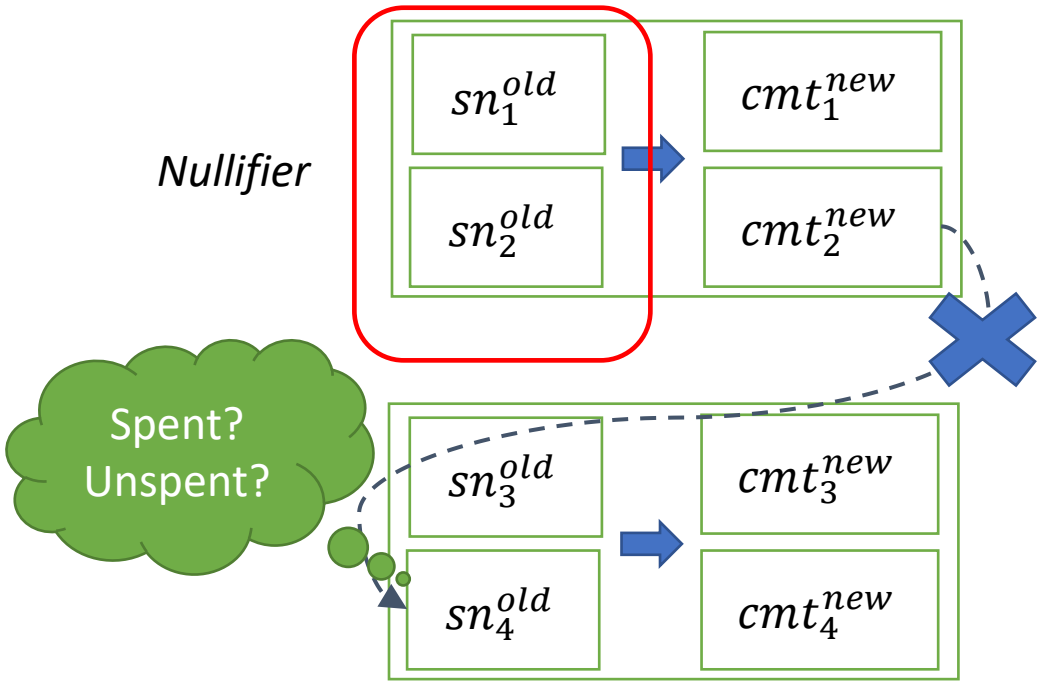


Second Attempt: two commitments

- $V(x, w) = 1$ if
  - The serial numbers of two old coins are correct
  - The commitments of two new coins are correct and verified by Merkle root
  - $v\left(c_1^{old}\right) + v\left(c_2^{old}\right) \geq v(c_1^{new}) + v(c_2^{new})$
  - $\mathrm{addr}_{\mathrm{sk}}$ of two old coins matches $\mathrm{addr}_{\mathrm{pk}}$

Check Unspent Coins

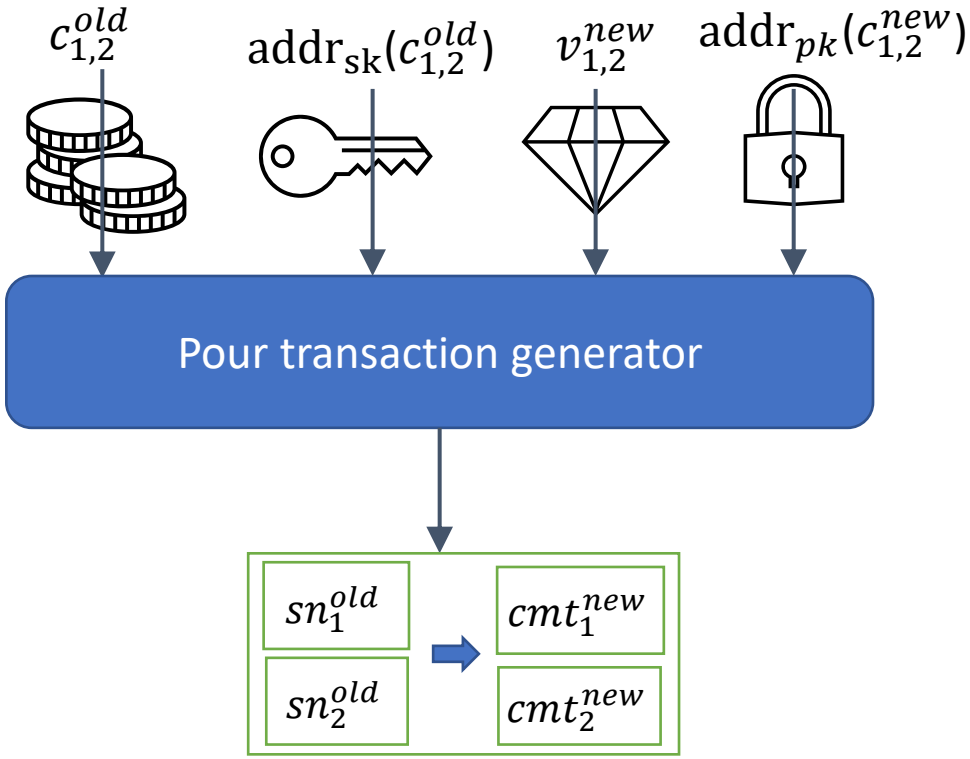
- How to make sure the old coins are unspent?
  - Keep record of all serial numbers appeared before
  - For new transactions, check if the serial numbers are in the nullifier



Protocol Summary

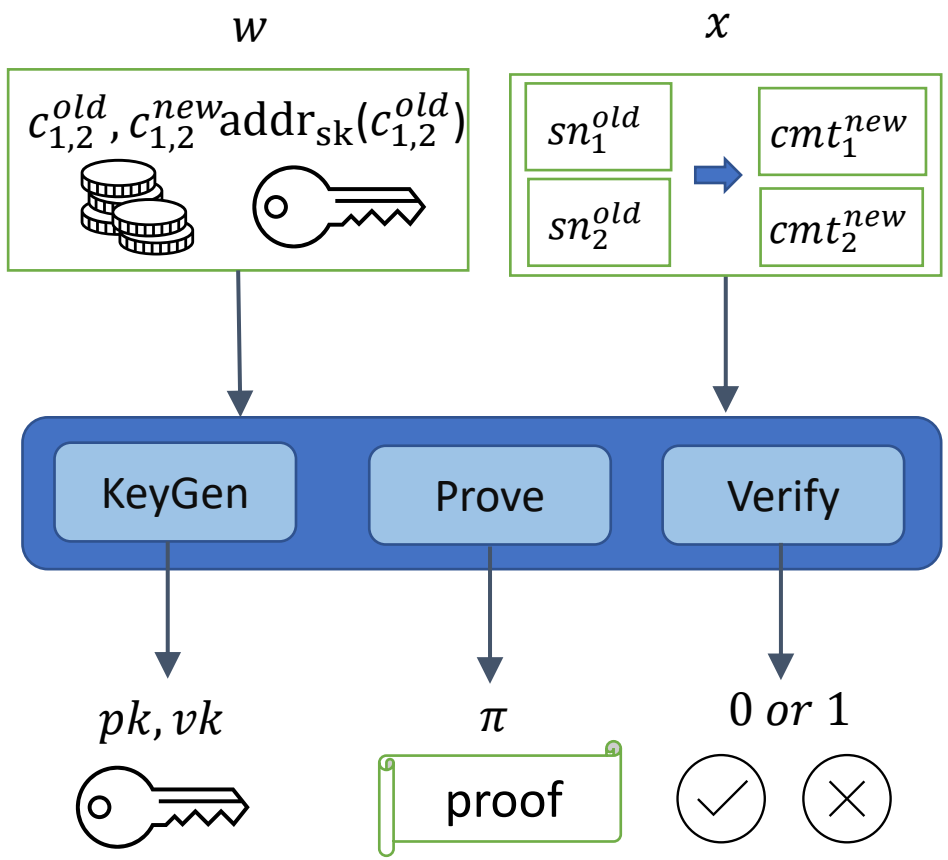
- Create pour transaction $(rt, sn_1^{old}, sn_2^{old}, cmt_1^{new}, cmt_2^{new}, v_{pub}, \text{info})$
  - Two old coins, c_1^{old}, c_2^{old}
  - Secret key of two old coins
  - New values: v_1^{new}, v_2^{new}
  - Public value: v_{pub}, s.t.

    $$v(c_1^{old}) + v(c_2^{old}) \geq v(c_1^{new}) + v(c_2^{new}) + v_{pub}$$
  - New addresses (public keys of new coins)

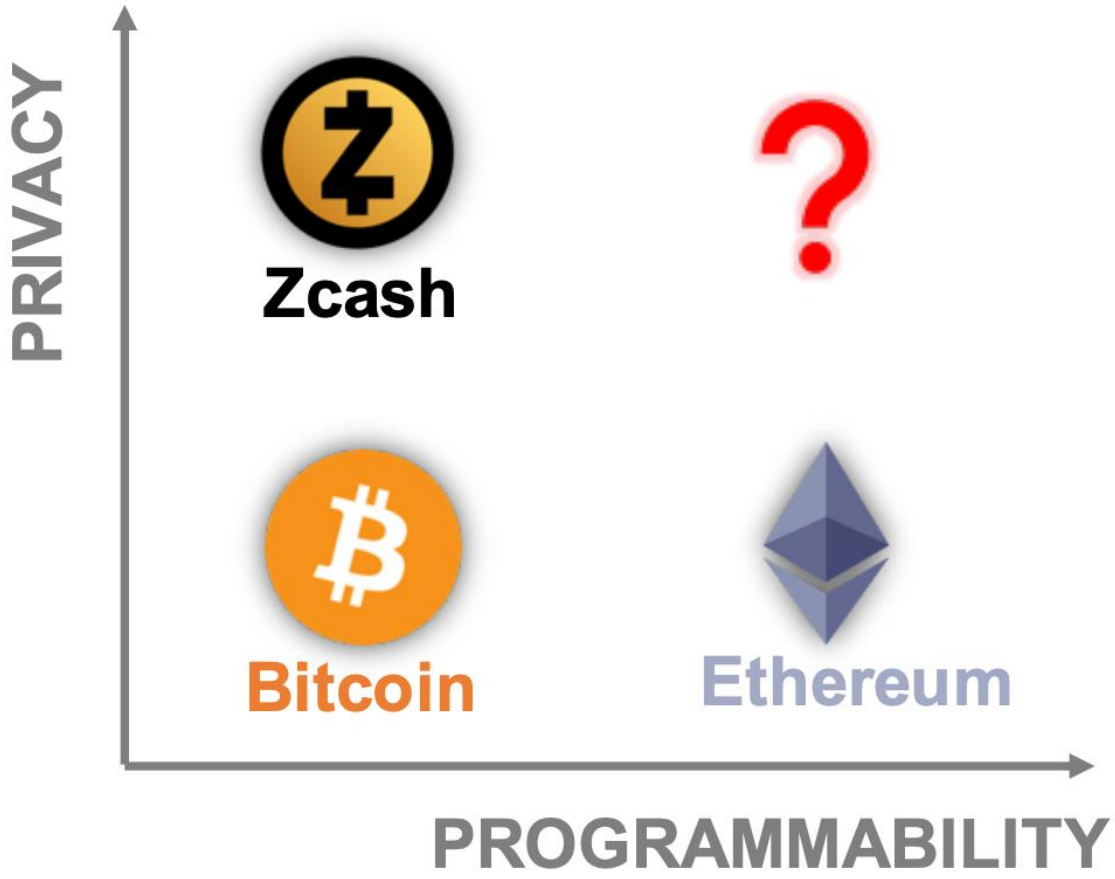


Protocol Summary

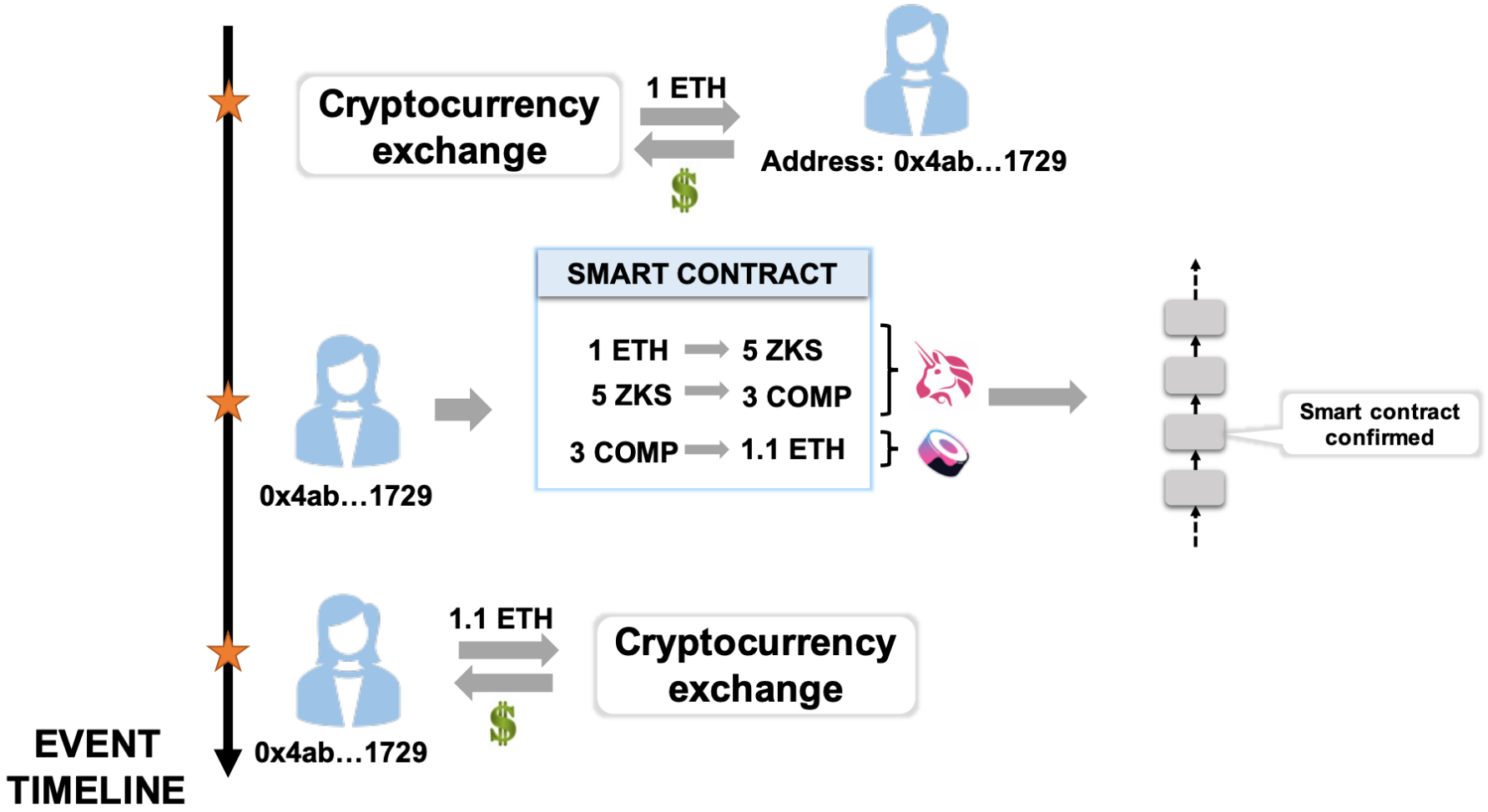
- Generate the proof
  - Library: zk-SNARK
  - Trusted setup: proving key pk and verifying key vk
  - $\pi = \mathrm{Prove}(pk,\ w,\ x)$
  - $\mathrm{Verify}(vk,\ x,\ \pi) = 0\ or\ 1$



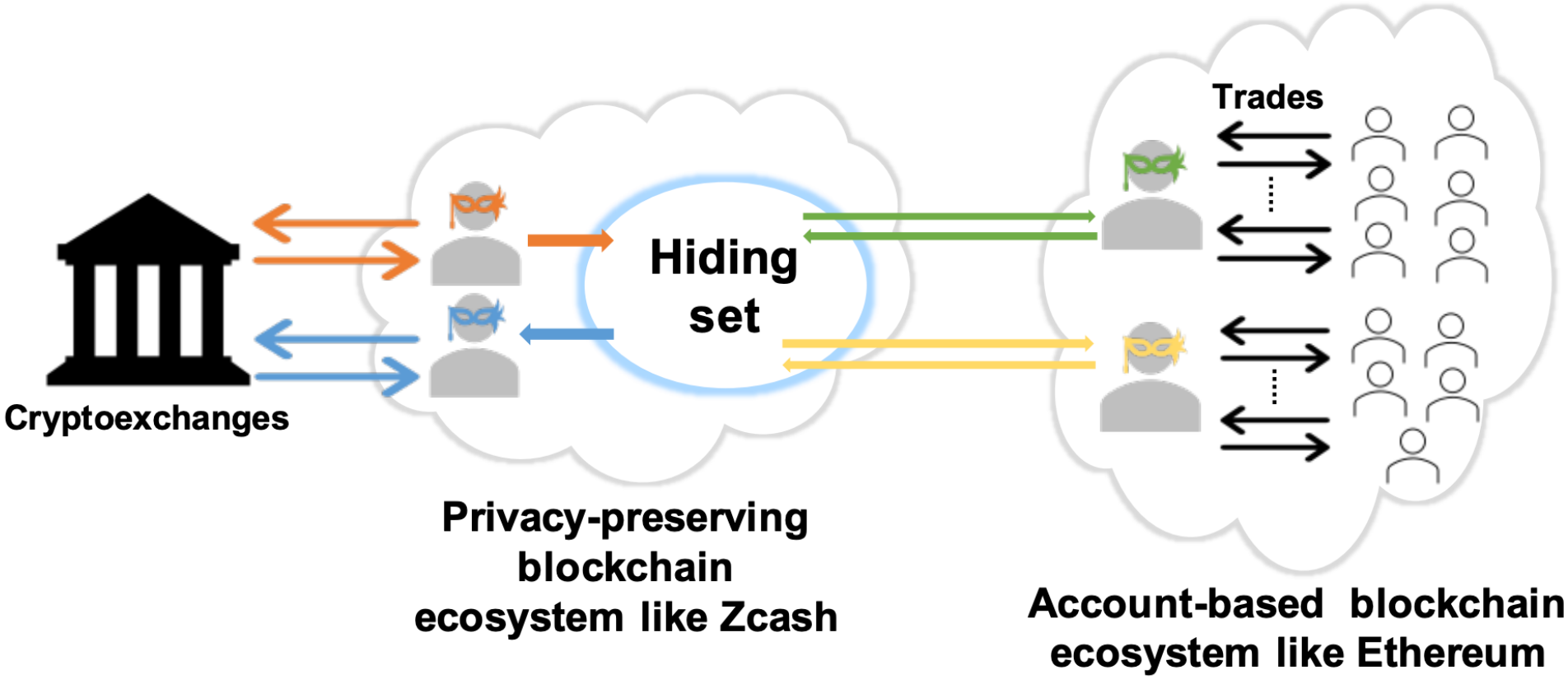
Privacy and Programmability



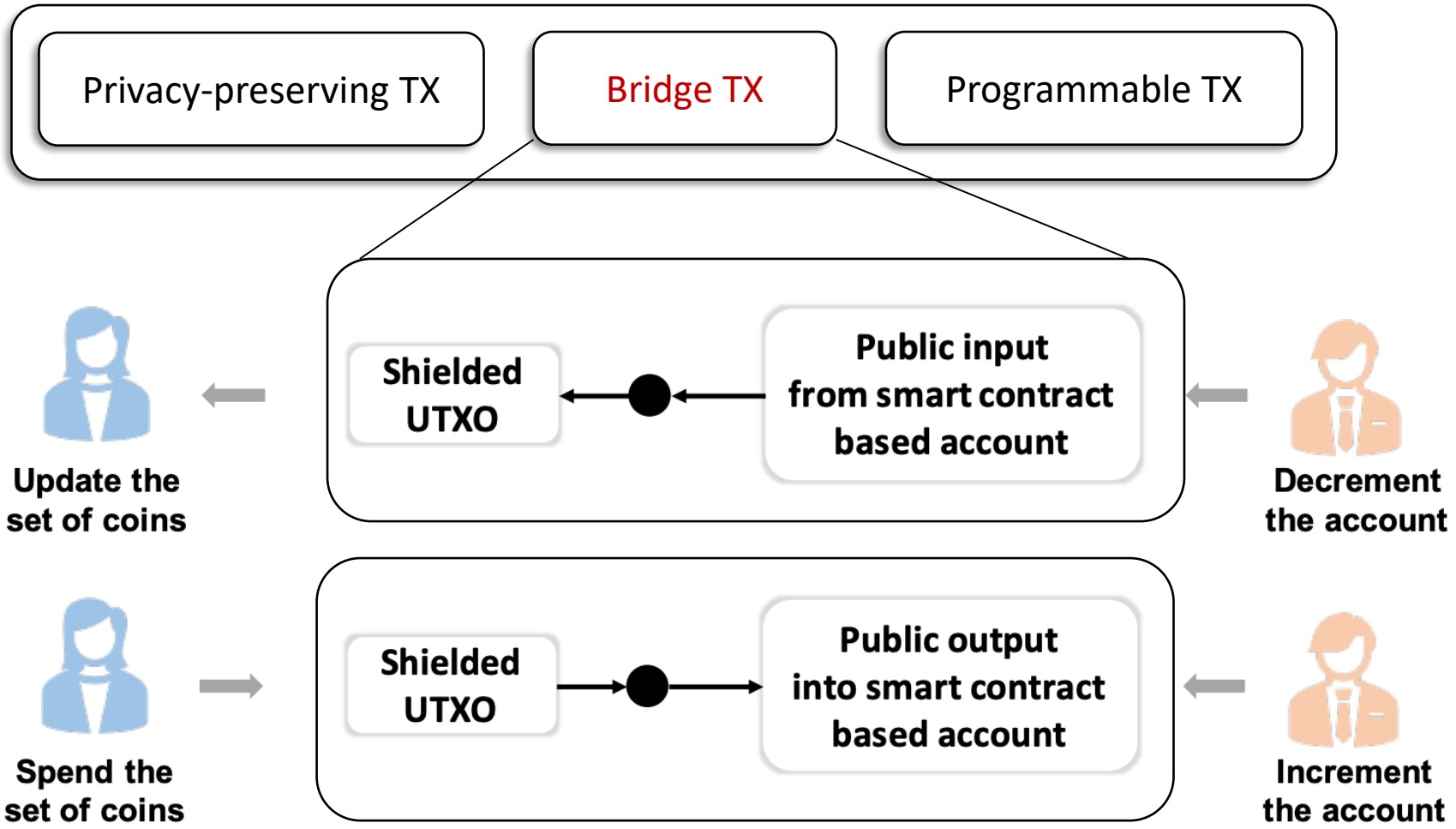
Privacy Issue in Account-based System

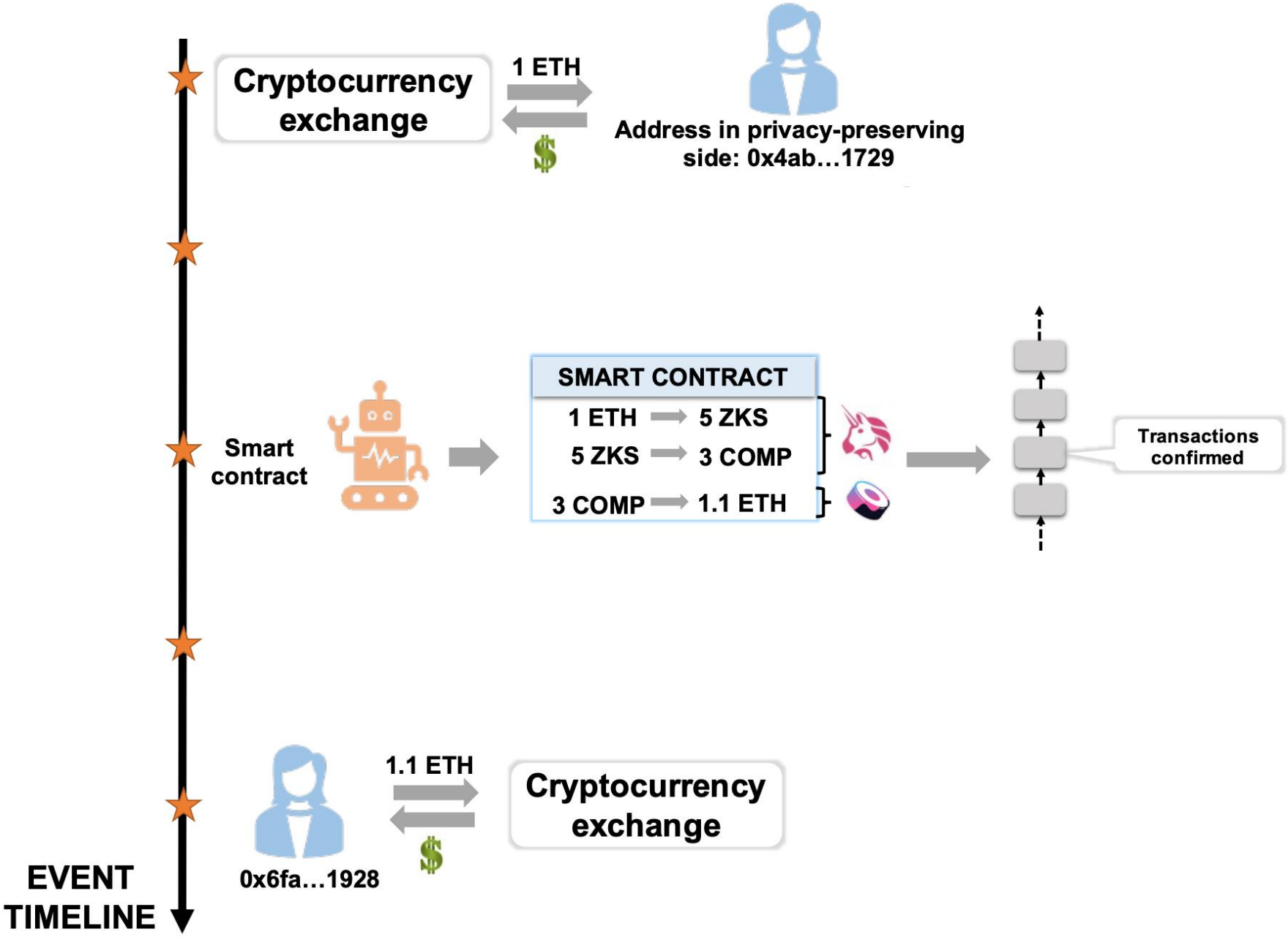


Key Idea: Bridging



Construction: Privacy Bridge




Cryptocurrency exchange
1 ETH
Address in privacy-preserving side: 0x4ab…1729
Smart contract
SMART CONTRACT
1 ETH → 5 ZKS
5 ZKS → 3 COMP
3 COMP → 1.1 ETH
Transactions confirmed
1.1 ETH
Cryptocurrency exchange
0x6fa…1928
EVENT TIMELINE

Tornado Cash

- Zcash based privacy for account-based model
- Tornado cash was a popular Ethereum smart contract
- Widely used by malicious parties (e.g., money laundering)
- Widely also used involuntarily as ”crypto hygiene”

- Tornado cash usage declared illegal by US DoT 8/8/2022
  - Developer Alexey Pertsev arrested

Summary

- Zcash provides privacy for UTXO-based model
- Privacy in account-based model
  - Bridging provides account privacy
  - Data is still public
- Data privacy: homomorphic encryption
- General privacy-preserving architectures
  - Zkay, Zether, Kachina…
- Active area of research and development

Attendance : NFT Drop

https://poap.website/political-sport-become

- Mint token to Metamask.
- Submit tx hash for attendance claim.

- Instructions in Ed pinned posts.