PRIVACY TOOLS IN DISTRIBUTED LEDGERS

JANNO SIIM

Simula UiB

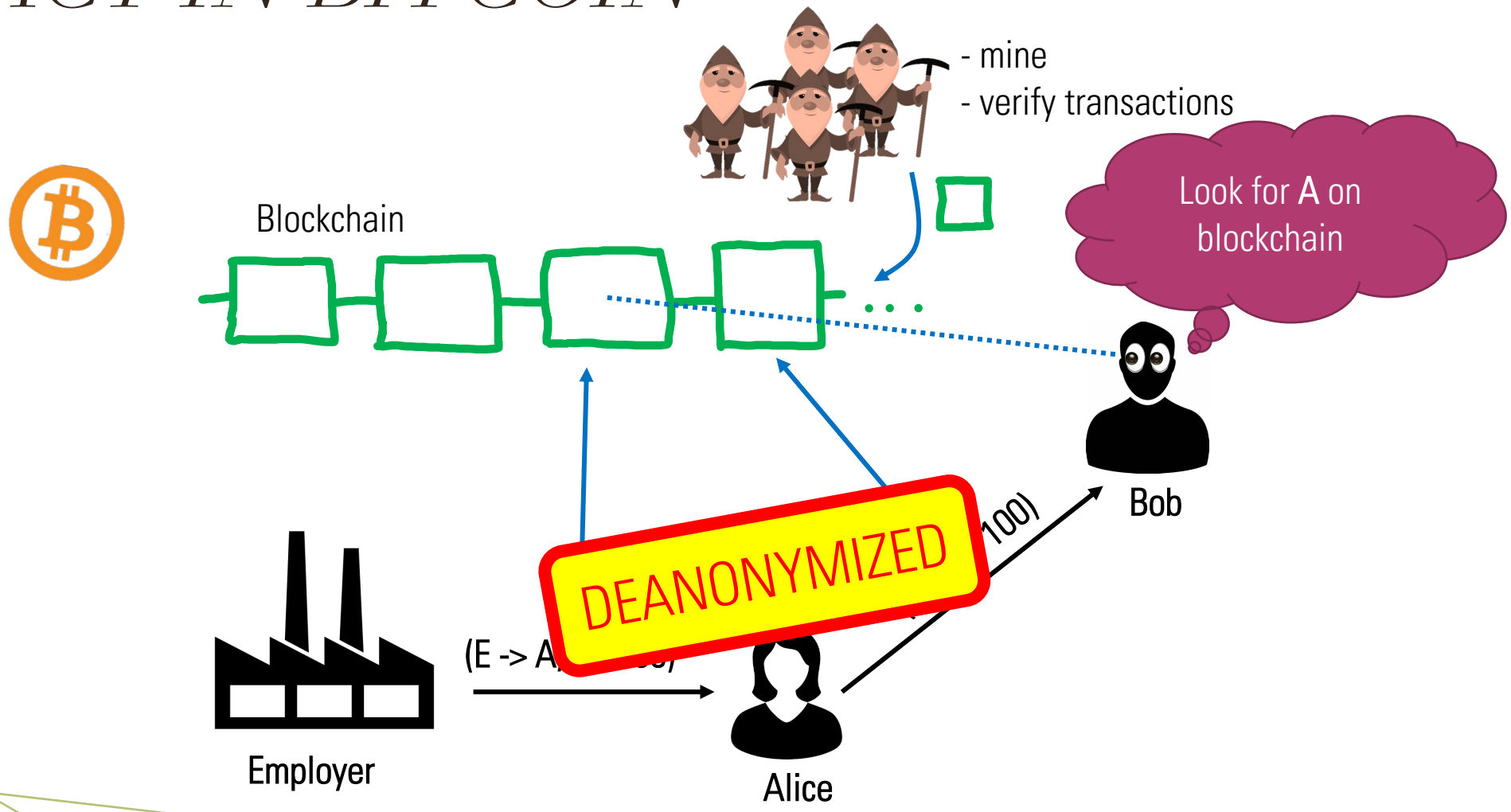
WHAT'S IT ABOUT?

- Privacy challenge in distributed ledgers
- Main tool: ZK-SNARK
- Challenges in ZK-SNARKs
- Alternative approach

PRIVACY

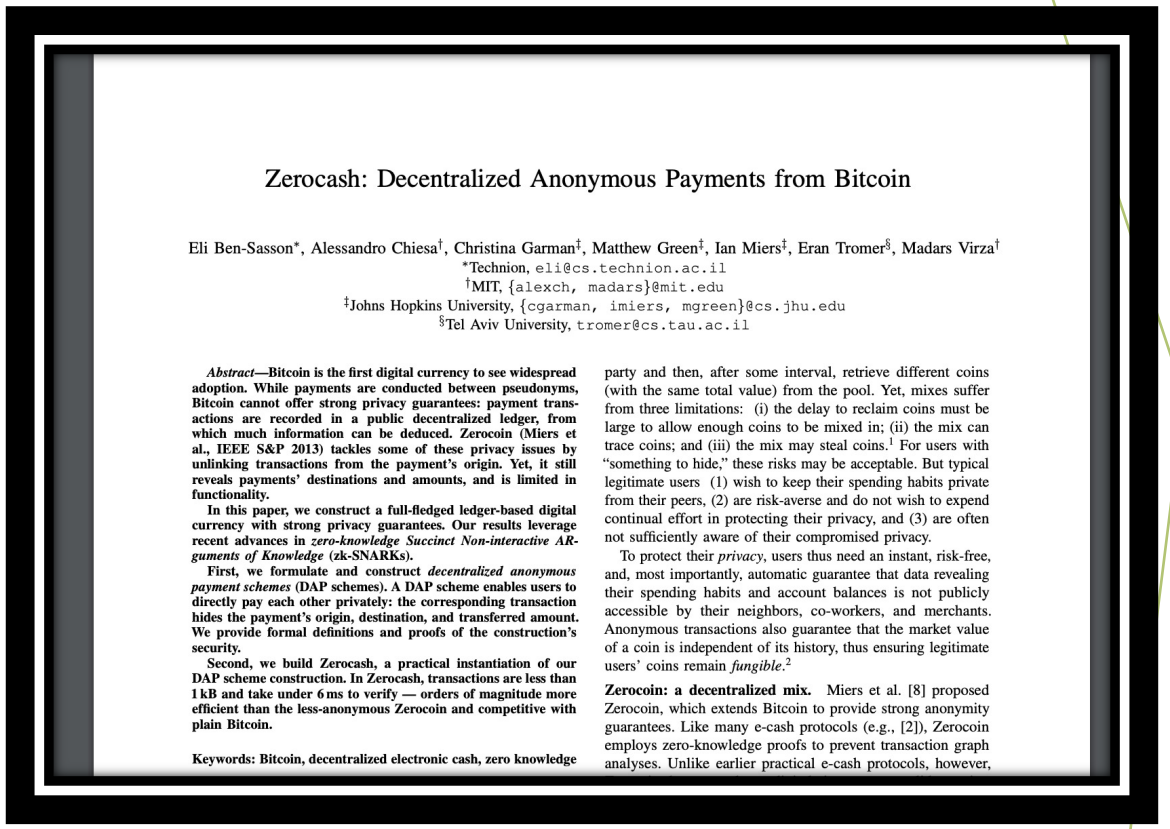
- Privacy is nice (in cryptocurrencies):
  - Neighbor shouldn't know what you bought for dinner
  - Competing company shouldn't know your suppliers
  - ...
- Extent of privacy:
  - Total privacy?
  - Access with court order?
  - Access to central authority?

~~PRIVACY~~ IN BITCOIN



SOLUTION: ZCASH

- How to solve the issue?
- Elegant solution from 2014: Zerocash



Zerocash: Decentralized Anonymous Payments from Bitcoin

Eli Ben-Sasson*, Alessandro Chiesa†, Christina Garman‡, Matthew Green‡, Ian Miers‡, Eran Tromer§, Madars Virza†
*Technion, eli@cs.technion.ac.il
†MIT, {alexch, madars}@mit.edu
‡Johns Hopkins University, {cgarman, imiers, mgreen}@cs.jhu.edu
§Tel Aviv University, tromer@cs.tau.ac.il

***Abstract*—Bitcoin is the first digital currency to see widespread adoption. While payments are conducted between pseudonyms, Bitcoin cannot offer strong privacy guarantees: payment transactions are recorded in a public decentralized ledger, from which much information can be deduced. Zerocoin (Miers et al., IEEE S&P 2013) tackles some of these privacy issues by unlinking transactions from the payment's origin. Yet, it still reveals payments' destinations and amounts, and is limited in functionality.**

In this paper, we construct a full-fledged ledger-based digital currency with strong privacy guarantees. Our results leverage recent advances in *zero-knowledge Succinct Non-interactive ARguments of Knowledge* (zk-SNARKs).

First, we formulate and construct *decentralized anonymous payment schemes* (DAP schemes). A DAP scheme enables users to directly pay each other privately: the corresponding transaction hides the payment's origin, destination, and transferred amount. We provide formal definitions and proofs of the construction's security.

Second, we build Zerocash, a practical instantiation of our DAP scheme construction. In Zerocash, transactions are less than 1 kB and take under 6 ms to verify — orders of magnitude more efficient than the less-anonymous Zerocoin and competitive with plain Bitcoin.

Keywords: Bitcoin, decentralized electronic cash, zero knowledge

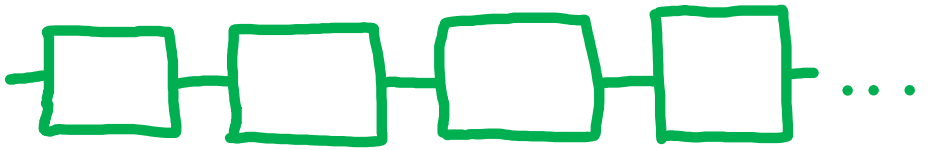
party and then, after some interval, retrieve different coins (with the same total value) from the pool. Yet, mixes suffer from three limitations: (i) the delay to reclaim coins must be large to allow enough coins to be mixed in; (ii) the mix can trace coins; and (iii) the mix may steal coins.[1] For users with "something to hide," these risks may be acceptable. But typical legitimate users (1) wish to keep their spending habits private from their peers, (2) are risk-averse and do not wish to expend continual effort in protecting their privacy, and (3) are often not sufficiently aware of their compromised privacy.

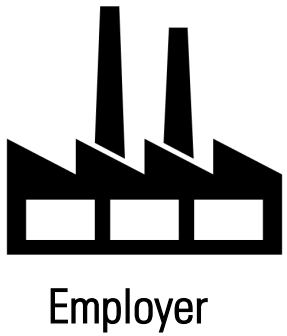
To protect their *privacy*, users thus need an instant, risk-free, and, most importantly, automatic guarantee that data revealing their spending habits and account balances is not publicly accessible by their neighbors, co-workers, and merchants. Anonymous transactions also guarantee that the market value of a coin is independent of its history, thus ensuring legitimate users' coins remain *fungible*.[2]

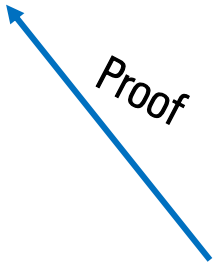
Zerocoin: a decentralized mix. Miers et al. [8] proposed Zerocoin, which extends Bitcoin to provide strong anonymity guarantees. Like many e-cash protocols (e.g., [2]), Zerocoin employs zero-knowledge proofs to prevent transaction graph analyses. Unlike earlier practical e-cash protocols, however,

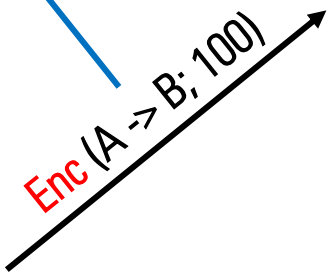
ZCASH

- Privacy: **YES**
- Verifiability: **NO**





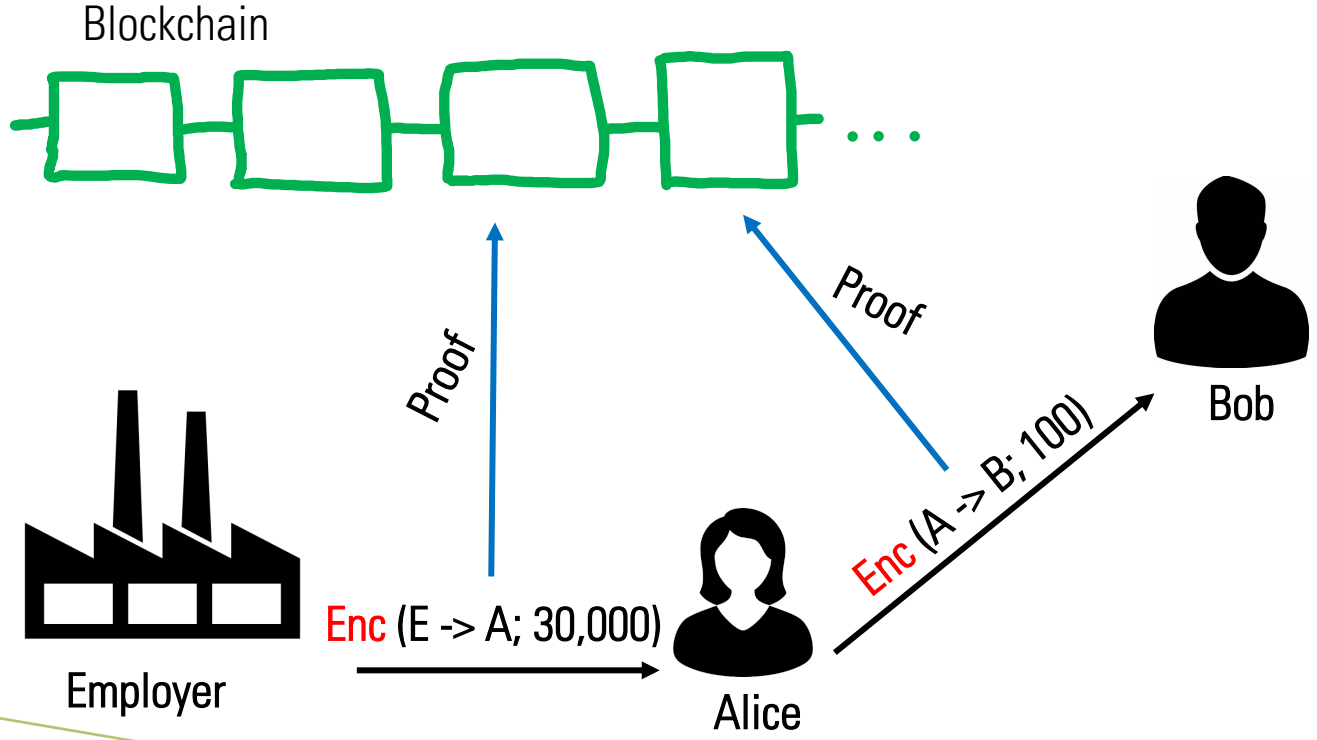




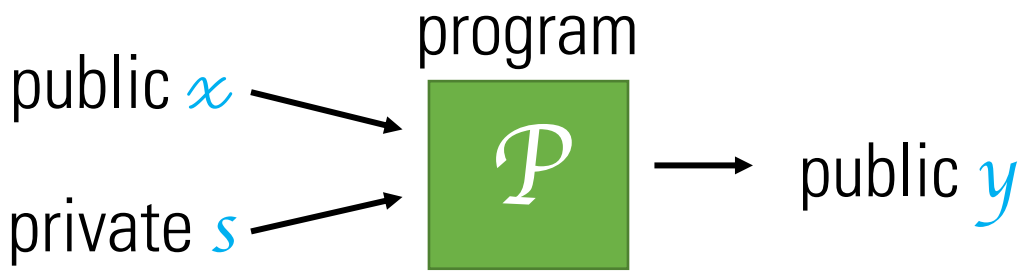
ZCASH

Zero-Knowledge Proof

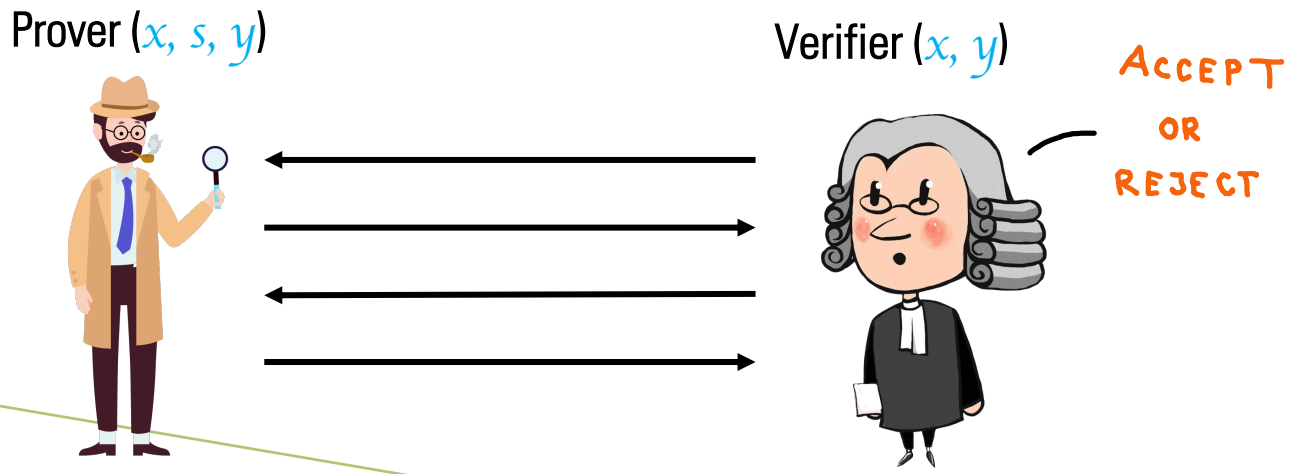
- Privacy: if proof doesn't leak
- Verifiability: if proof is unforgable



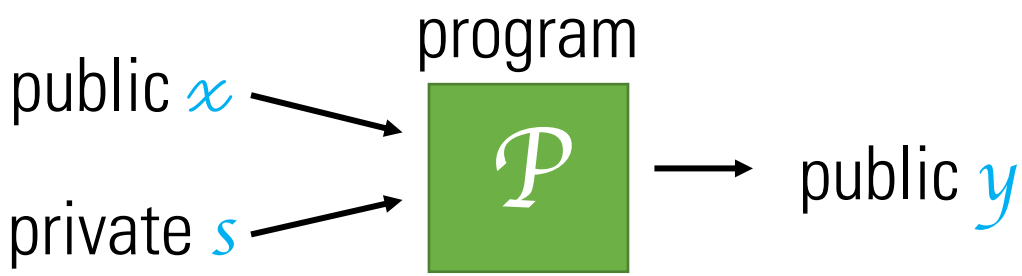
ZERO-KNOWLEDGE (ZK) PROOF



- Prover claims: there is s such that $\mathcal{P}(x, s) = y$



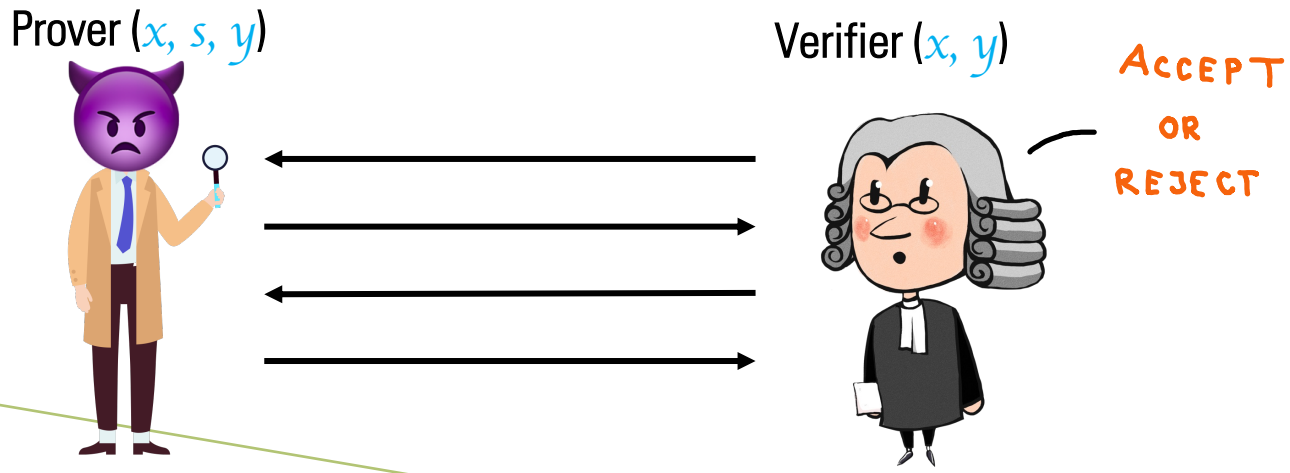
ZERO-KNOWLEDGE (ZK) PROOF



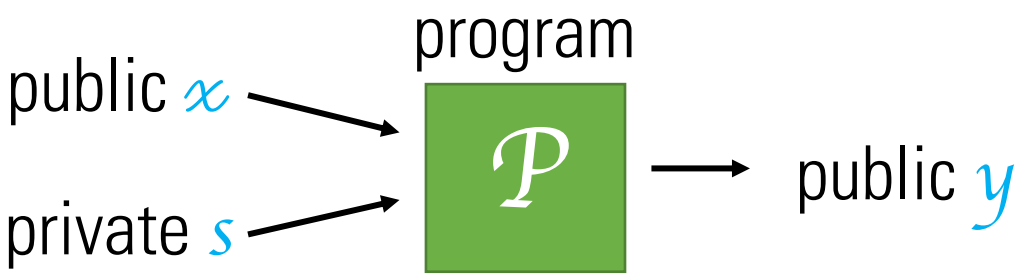
PROPERTIES

Soundness (unforgability):
- prover cannot convince verifier if $\mathcal{P}(x, s) \neq y$

- Prover claims: there is s such that $\mathcal{P}(x, s) = y$



ZERO-KNOWLEDGE (ZK) PROOF

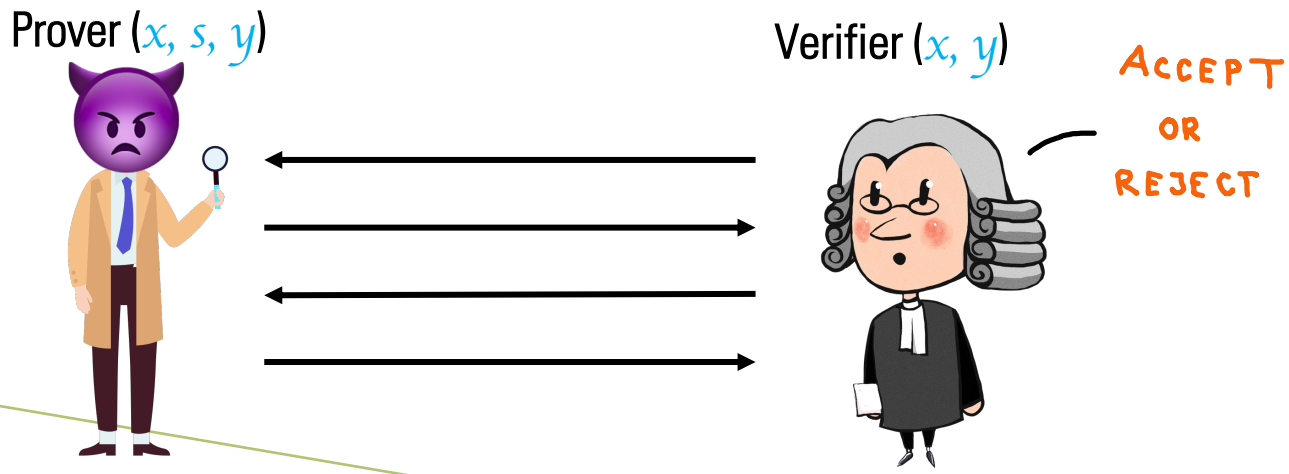


PROPERTIES

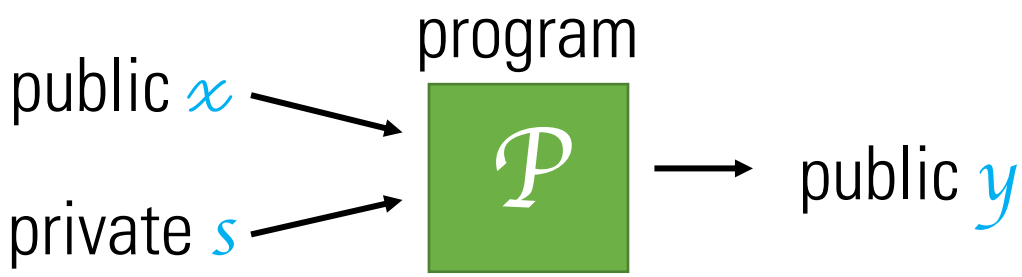
Soundness (unforgability):
- prover cannot convince verifier if $\mathcal{P}(x, s) \neq y$

Knowledge soundness:
- prover **knows** s

- Prover claims: there is s such that $\mathcal{P}(x, s) = y$



ZERO-KNOWLEDGE (ZK) PROOF

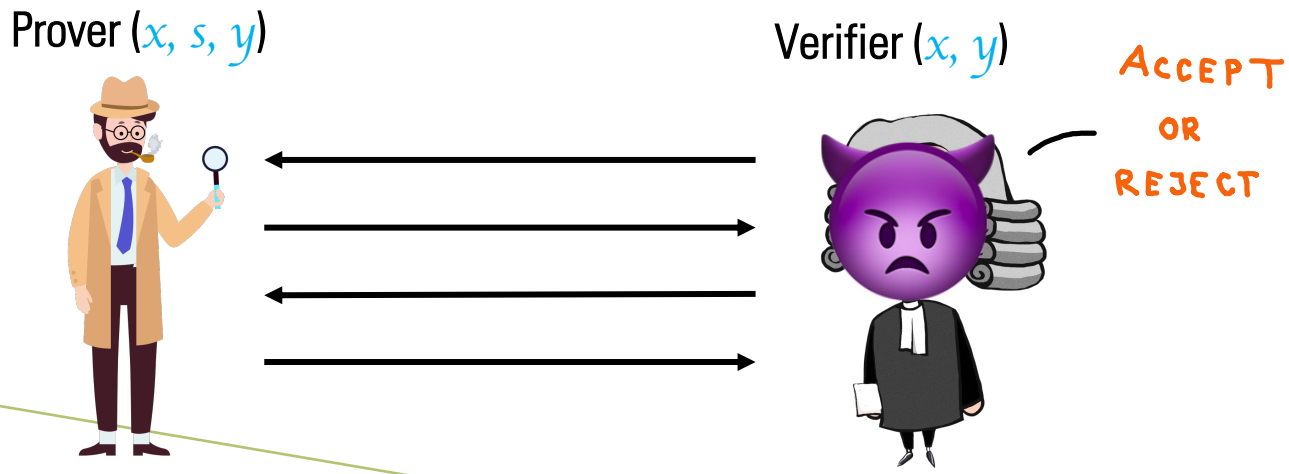


PROPERTIES

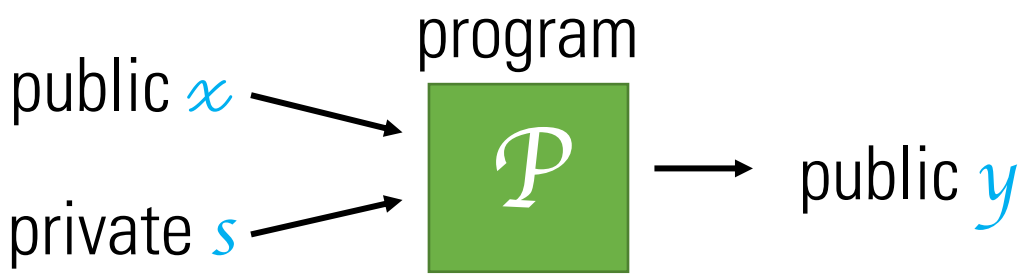
Soundness (unforgability):
- prover cannot convince verifier if $\mathcal{P}(x, s) \neq y$

Knowledge soundness:
- prover **knows** s

- Prover claims: there is s such that $\mathcal{P}(x, s) = y$



ZERO-KNOWLEDGE (ZK) PROOF



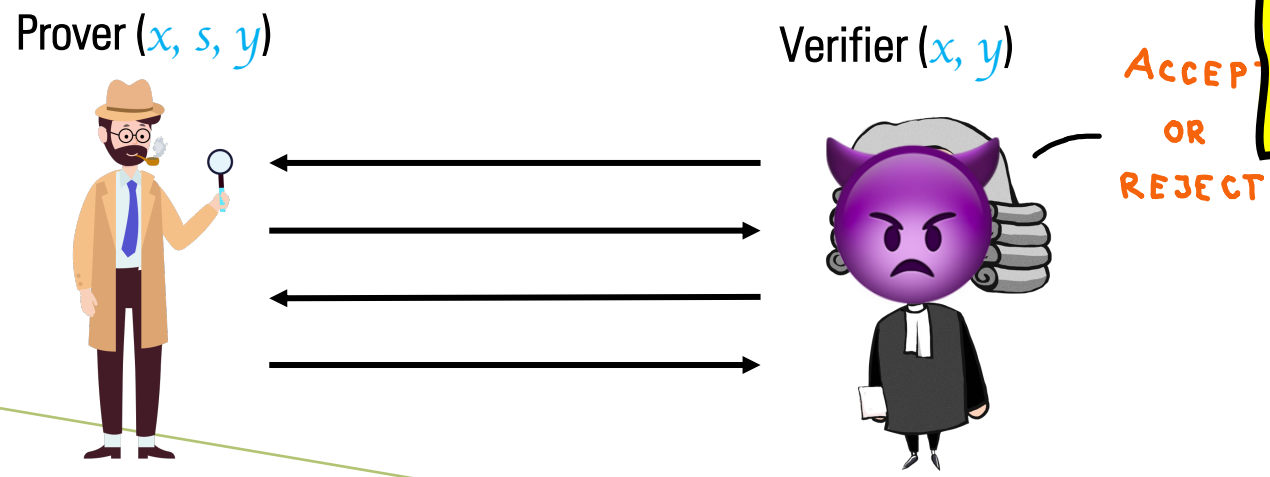
PROPERTIES

Soundness (unforgability):
- prover cannot convince verifier if $\mathcal{P}(x, s) \neq y$

Knowledge soundness:
- prover **knows** s

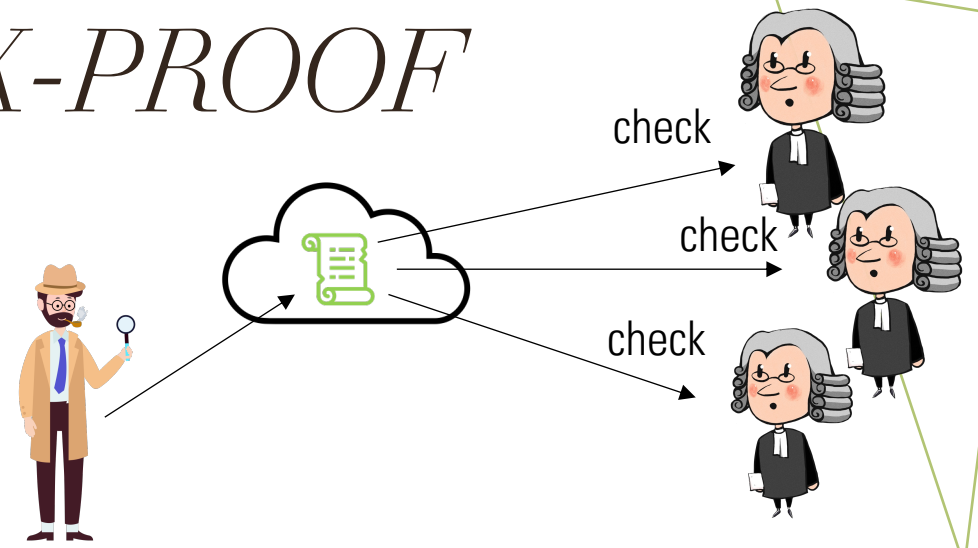
Zero-knowledge (privacy):
- s is not leaked
- even more: only leaked information is that $\mathcal{P}(x, s) = y$

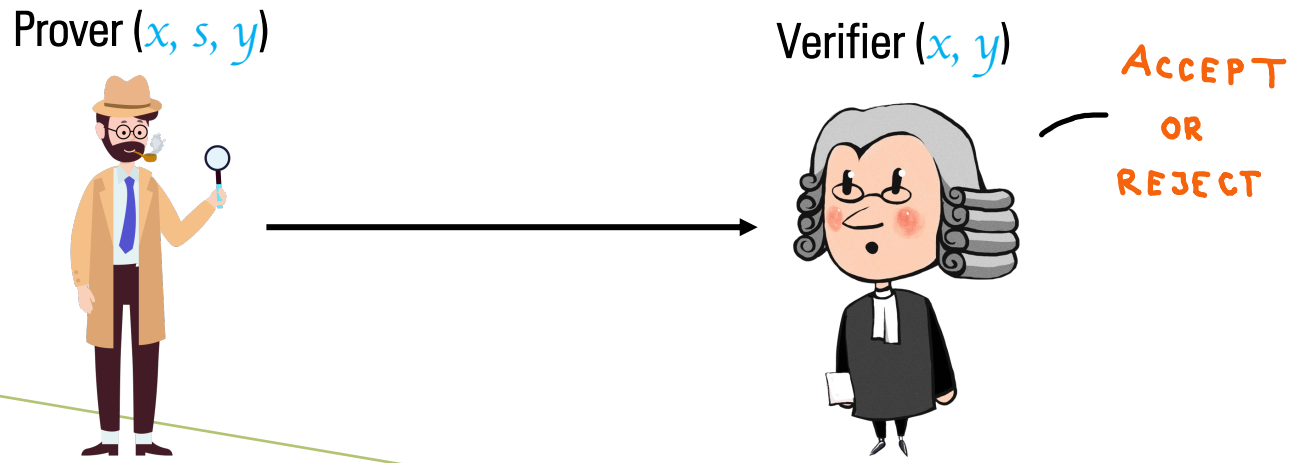
- Prover claims: there is s such that $\mathcal{P}(x, s) = y$



NON-INTERACTIVE ZK-PROOF

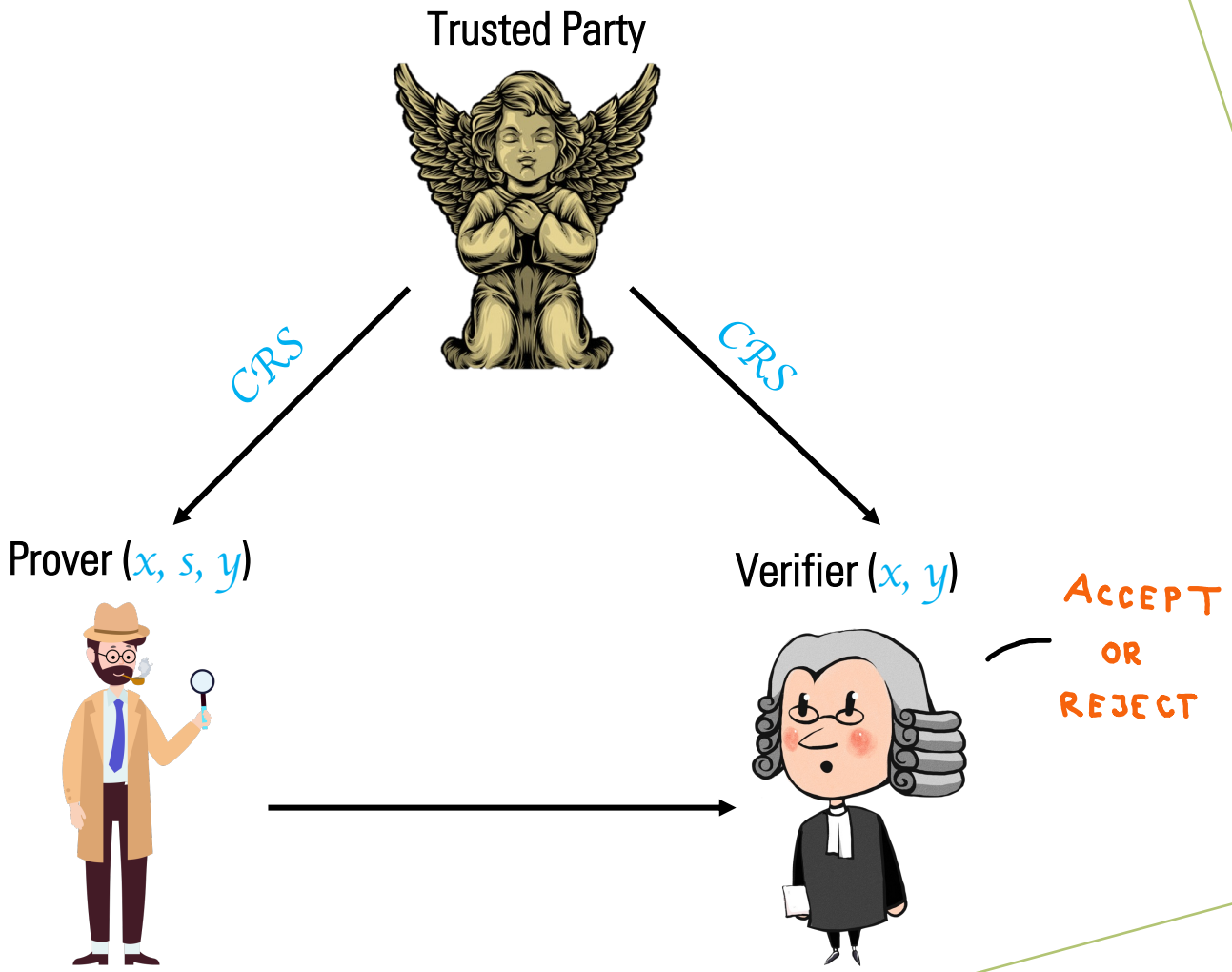
- What other properties are needed?
- Proof should be verifiable by many verifiers
- Proof should be non-interactive
- Mathematically impossible!



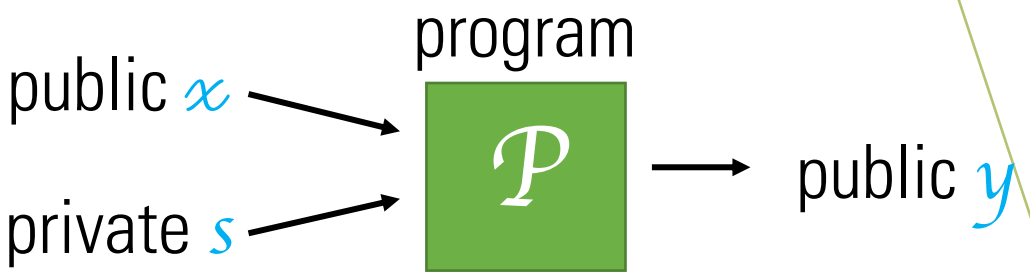


COMMON REFERENCE STRING (CRS)

- Trusted setup phase
- Avoids impossibility results



EFFICIENCY



- What else?
- Succinctness:
  - Proof size: much smaller than s
  - Verifier much faster than recomputing $\mathcal{P}(x, s)$
- ZK-SNARK = Zero-Knowledge Succinct Non-interactive ARgument of Knowledge
- Prover's speed: roughly the same as computing $\mathcal{P}(x, s) = y$

EARLY RESULTS

- ZK-proof proposed in 1985 (Goldwasser, Micali, Rackoff)
  - Turing award, Gödel Prize
  - theoretical results for specific programs $\mathcal{P}$
- 80s-90s:
  - ZK-proof for all efficient programs $\mathcal{P}$
  - non-interactive zero-knowledge
  - ZK-SNARKs (CS-proofs)
  - many theoretical results
  - impractical efficiency for arbitrary programs $\mathcal{P}$
  - good efficiency for some specific problems: Σ-protocols

PRACTICAL SNARKS

Groth

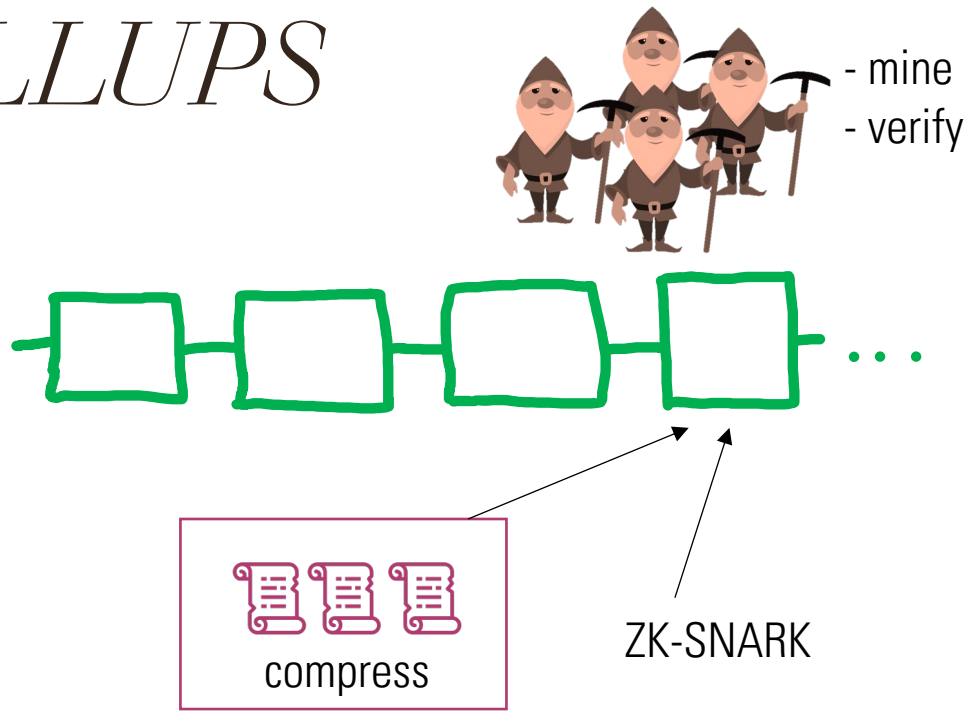
Lipmaa

2000s:

- Pairing-based cryptography
- First efficient ZK-SNARKs
  - almost good enough for real life
- Better mathematical modeling of programs (Quadratic Span Programs, Quadratic Arithmetic Programs, ...)
  - practical efficiency
  - Pinocchio ZK-SNARK, Groth16 ZK-SNARK, ...
  - Proof size: ~1500bits (for any program!)

APPLICATION: ROLLUPS

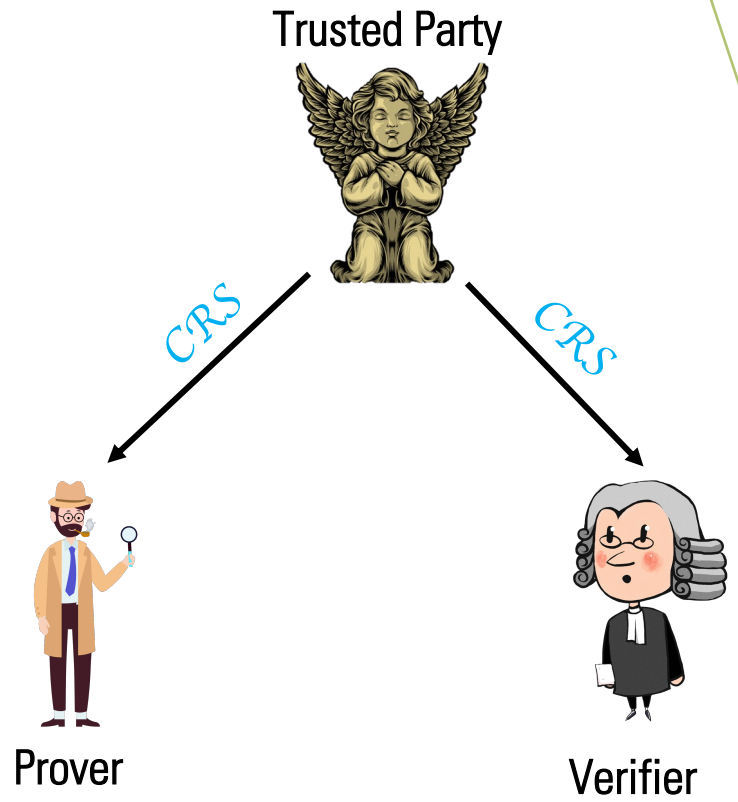
- Forget about privacy
- Blockchain scalability problem
- Rollups:
  - compress transactions
  - give ZK-SNARK to prove correctness
- Zero-knowledge doesn't matter
- Soundness and Succinctness



OPEN PROBLEMS

<u>Trusted setup:</u>

- distributed ledger ≠ trusted setup 😰
- New $\mathcal{CRS}$ for each program $\mathcal{P}$ 😰
- solutions:
  - multi-party computation for $\mathcal{CRS}$
    - cumbersome
    - have to run for each $\mathcal{P}$
  - universal ZK-SNARKs – same $\mathcal{CRS}$ for all $\mathcal{P}$
  - transparent ZK-SNARKs – $\mathcal{CRS}$ is public random string
  - How to get as good efficiency?



OPEN PROBLEMS

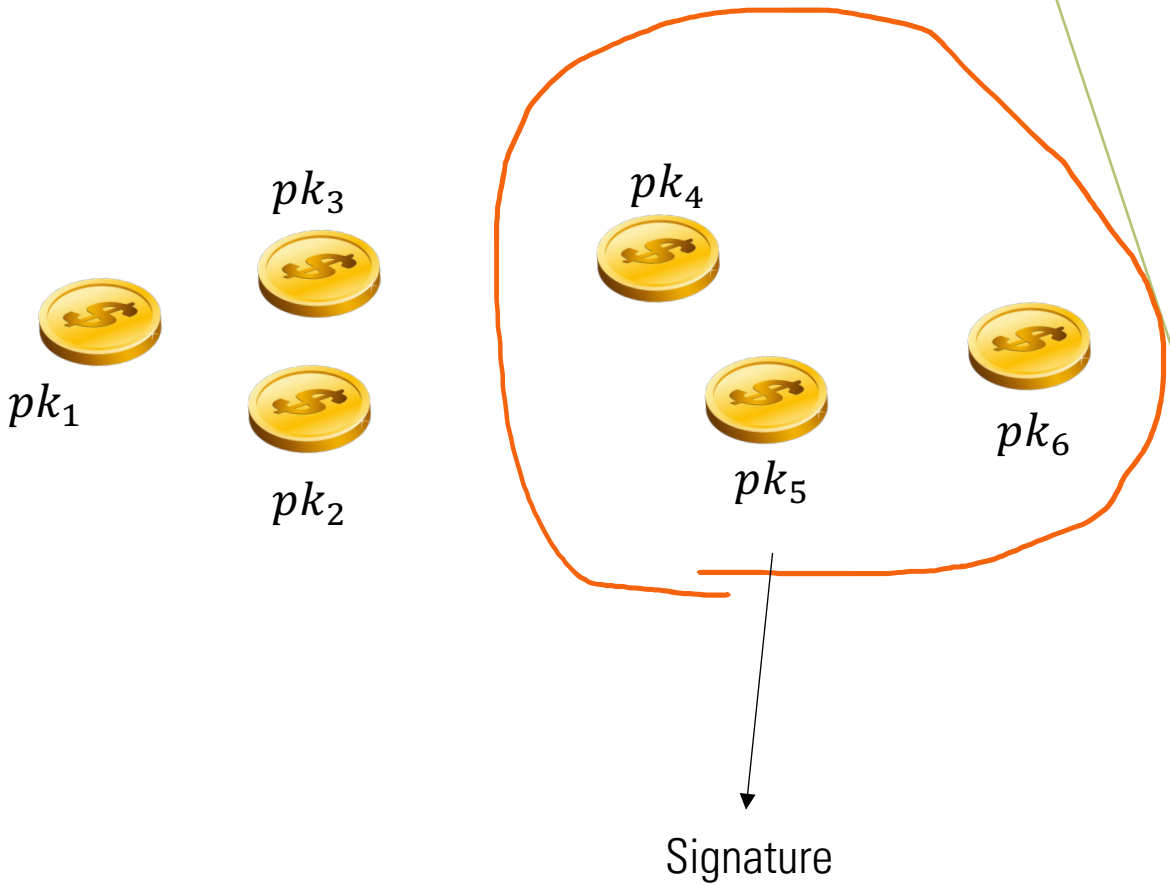
Security assumptions:

- Cryptography based on assumption
  - **Falsifiable assumptions:** computing X is hard
    - feel safer
  - **Non-falsifiable assumptions:**
    - hash functions give random outputs
    - if you compute X, then you know Y (knowledge assumptions)
    - realistic NF assumptions?
- Post-quantum security:
  - Most SNARKs insecure against quantum
  - Some candidates (less efficient)

INSUFFICIENT FOR SNARKS!

ALTERNATIVE

- Traceable ring signatures (Monero)
- Signer is private in the ring
- Double spending protection:
  - cannot sign twice without detection



QUESTIONS