

CS251 Fall 2022
(cs251.stanford.edu)



Privacy on the Blockchain

Dan Boneh

The need for privacy in the financial system

Supply chain privacy:

- A manufacturer does not want to reveal how much it pays its supplier for parts.



Payment privacy:

- A company that pays its employees in crypto wants to keep list of employees and salaries private.
- Endusers need privacy for rent, donations, purchases

Business logic privacy: Can the code of a smart contract be private?

The need for privacy in the financial system

The bottom line:

Blockchains cannot reach their full potential without some form of private transactions

Types of Privacy

Pseudonymity: (weak privacy)

- Every user has a long-term consistent pseudonym (e.g. reddit)
 - Pros: reputation
 - Cons: link to real-world identity can leak over time

Full anonymity: User's transactions are unlinkable

- No one can tell if two transactions are from the same address

A difficult question: privacy from who?

No privacy:

Everyone can see all transactions



Privacy from the public:

Only a trusted operator can see transactions

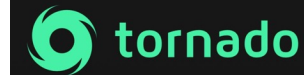


Semi-full privacy:

only “local” law enforcement can see transactions

full privacy:

no one can see transactions



Negative aspects of complete privacy

How to prevent criminal activity?

The challenge:

- How to support positive applications of private payments, but prevent the negative ones?
- Can we ensure legal compliance while preserving privacy?
- Yes! The key technology: **zero knowledge proofs**



Are Bitcoin and Ethereum Private?

The base systems are definitely not ...

Privacy in Ethereum?





- Every account balance is public
- For Dapps: code and internal state are public
- All account transactions are linked to account

etherscan.io:

Address 0x1654b0c3f62902d7A86237...

Balance: 1.114479450024297906 Ether

Ether Value: \$4,286.34 (@ \$3,846.05/ETH)

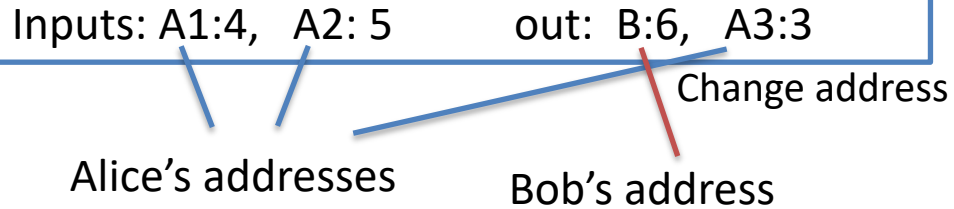
	Txn Hash	Method ⓘ	Block
	0x0269eff8b4196558c07...	Set Approval For...	13426561
	0xa3dacb0e7c579a99cd...	Cancel Order_	13397993
	0x73785abcc7ccf030d6a...	Set Approval For...	13387834
	0x1463293c495069d61c...	Atomic Match_	13387703

Privacy in Bitcoin?

The screenshot shows a Bitcoin transaction with a fee of 0.00179523 BTC. It features two input boxes on the left and two output boxes on the right. Arrows point from the input boxes to the labels 'from addresses' and 'amounts', and from the output boxes to 'to addresses' and 'amounts'.

Address	Amount
16k4365RzdeCPKGwJDNNBEKxj696MbChwx	0.53333328 BTC
1Bsh4KD9ZJT4dJcoo7S5uS1jvtmtVmREb7	1.47877788 BTC
1JgVBpw5TDMTRoZXg9XpPDQRRHtNb5CsPA	0.01031593 BTC (U)
1AFLhD4EtG2uZmFxmfdXCyGUNqCqD5887u	2 BTC (S)

Alice can have many addresses (creating address is free)



Transaction data can be used to link an address to a physical identity

(chainalysis)

Linking an addresses to an identity

inputs: A1: 4, A2: 5 outputs: B: 6, A3: 3

Alice buys a book from a merchant:

- Alice learns one of merchant's address (B)
 - Merchant links three addresses to Alice (A1, A2, A3)
-

Alice uses an exchange (ETH \leftrightarrow USD)

- BSA: a US exchange must do KYC (know your customer)
... collect and verify Alice's ID
- Exchange links Alice to her addresses (A1, A2, A3)

De-anonymization strategy: Idioms of use

A general strategy for de-anonymizing Bitcoin addresses

Heuristic 1:

Two addresses are input to a TX

⇒ both addresses are controlled by same entity

The screenshot displays a Bitcoin transaction interface. At the top, the transaction ID is `c2561b292ed4878bb28478a8cafd1f99a01faeb9c5a906715fa595cac0e8d1d8`, and it was mined on Apr 10, 2017 at 12:38:00 AM. The transaction has two input addresses and two output addresses. The inputs are:

- `16k4365RzdeCPKGwJDNNBEkXj696MbChwx` (0.53333328 BTC)
- `1Bsh4KD9ZJT4dJcoo7S5uS1jvtmtVmREb7` (1.47877788 BTC)

The outputs are:

- `1JgVBpw5TDMTRoZXg9XpPDQRRHtNb5CsPA` (0.01031593 BTC (U))
- `1AFLhD4EtG2uZmFxmfdXCyGUNqCqD5887u` (2 BTC (S))

At the bottom left, the fee is 0.00179523 BTC. At the bottom right, there is a blue button for "1 CONFIRMATIONS" and a green button for "2.01031593 BTC".

De-anonymization strategy: Idioms of use

Heuristic 2:

Change address is controlled by the same user as input address

Which is the change address?

- Heuristic: a new address that receives less than every input

The screenshot displays a Bitcoin transaction interface. At the top, the transaction ID is `c2561b292ed4878bb28478a8cafd1f99a01faeb9c5a906715fa595cac0e8d1d8`, and it was mined on Apr 10, 2017 at 12:38:00 AM. The transaction has two inputs and two outputs. The first input is `16k4365RzdeCPKGwJDNNBEkXj696MbChwx` with a value of 0.53333328 BTC. The second input is `1Bsh4KD9ZJT4dJcoo7S5uS1jvtmtVmREb7` with a value of 1.47877788 BTC. The first output is `1JgVBpw5TDMTRoZXg9XpPDQRRHtNb5CsPA` with a value of 0.01031593 BTC (U). The second output is `1AFLhD4EtG2uZmFxmfdXCyGUNqCqD5887u` with a value of 2 BTC (S). The fee is 0.00179523 BTC. The transaction has 1 confirmation and a total value of 2.01031593 BTC.

Address	Value (BTC)
<code>16k4365RzdeCPKGwJDNNBEkXj696MbChwx</code>	0.53333328
<code>1Bsh4KD9ZJT4dJcoo7S5uS1jvtmtVmREb7</code>	1.47877788
<code>1JgVBpw5TDMTRoZXg9XpPDQRRHtNb5CsPA</code>	0.01031593 (U)
<code>1AFLhD4EtG2uZmFxmfdXCyGUNqCqD5887u</code>	2 (S)

FEE: 0.00179523 BTC

1 CONFIRMATIONS

2.01031593 BTC

A Bitcoin experiment

[Meiklejohn, et al.]

step 1: Heuristic 1 and 2 \Rightarrow 3.3M clusters

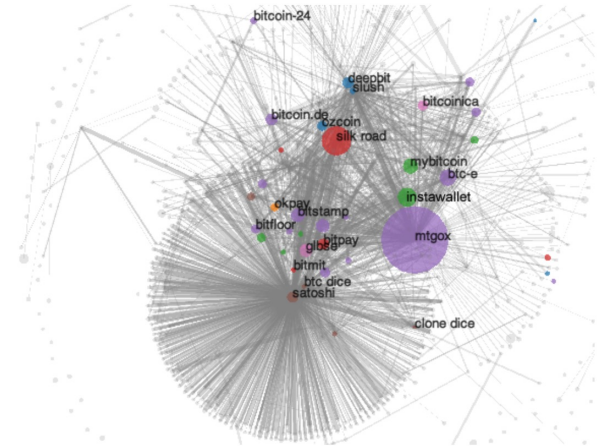
step 2: 1070 addreses identified by interacting with merchants

- Coinbase, Bitpay, ...

step 3: now 15% of all addresses identified

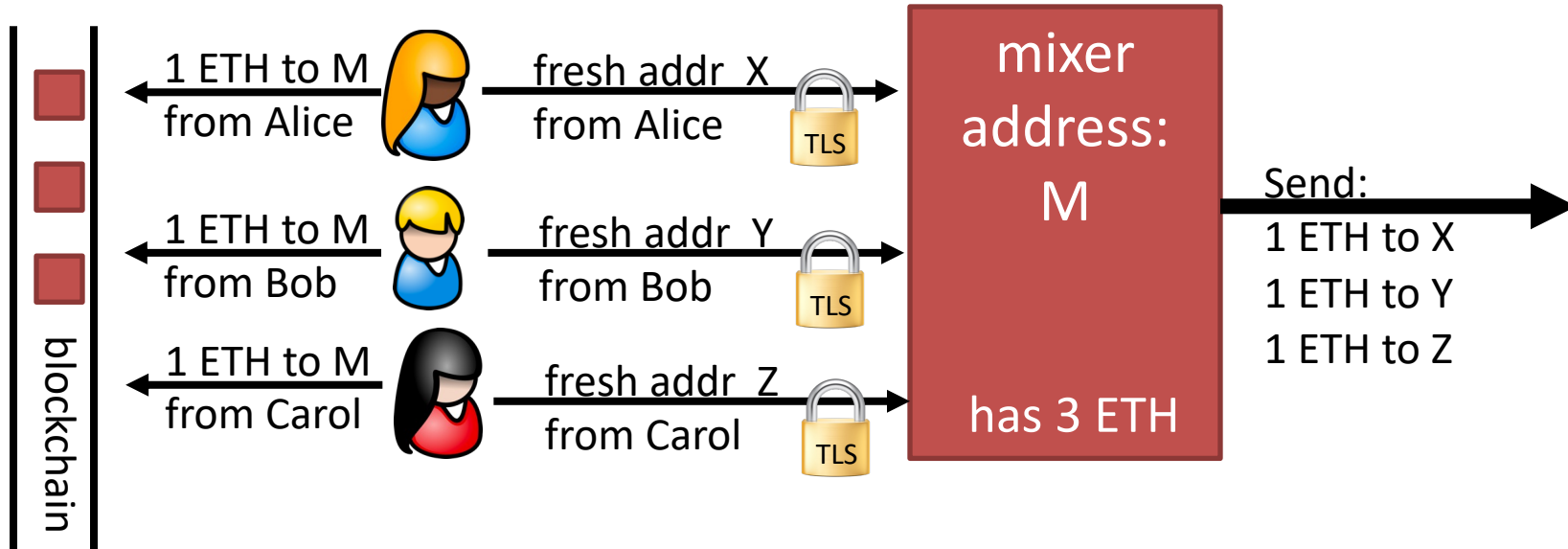
- Learn total assets for all clusters

Commercial efforts: Chainalysis, Elliptic, ...



Private coins on a Public Blockchain

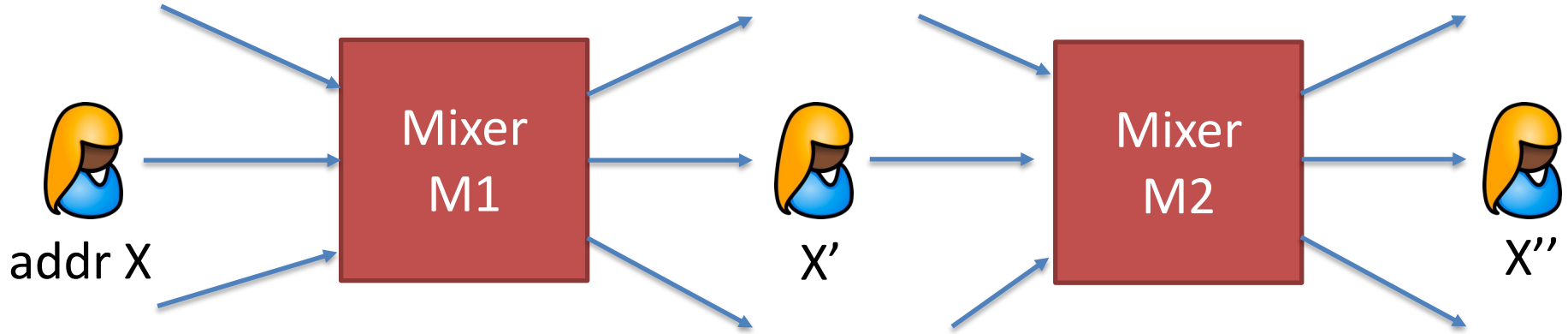
Attempt 1: simple mixing



Observer knows Y belongs to one of {Alice, Bob, Carol} but does not know which one
⇒ anonymity set of size 3.

Problems: (i) mixer M knows shuffle, (ii) mixer can abscond with 3 ETH !!

Increasing the anonymity set



M1: mix n inputs from n users \Rightarrow X' has anonymity set size = n

M2: mix output from m mixers \Rightarrow X'' has anonymity set size = $n \times m$

Privacy: as long as one of M1 or M2 are honest

Secure mixing without a mixer?

Problem: Mixer can abscond with funds or reveal the shuffle.

Can we securely mix without a trusted mixer? Answer: yes!

- on Bitcoin: **CoinJoin** (used by, e.g., Wasabi wallet)
- on Ethereum: **Tornado cash**
 - ... a large scale single mixer using ZK proofs – next lecture

CoinJoin: Bitcoin Mixing without Mixer

The setup: Alice, Bob, and Carol want to mix together.

Alice owns UTXO **A1:5**, Bob owns UTXO **B1:3**, Carol owns **C1:2**



A1: 5, A3 (change addr)

A2 (post mix address over Tor)



B1: 3, B3 (change addr)

B2 (post mix address over Tor)



(same as Alice and Bob)



A1: 5, A3

B1: 3, B3

C1: 2, C3

B2, A2, C2

public forum

mix
addresses

CoinJoin: Bitcoin Mixing without Mixer

CoinJoin TX: all three prepare and sign the following Tx:

inputs (not private): A1: 5, B1: 3, C1: 2

outputs (private): B2: 2, A2: 2, C2: 2

outputs (not private): A3: 3, B3: 1

mix addresses

Mixed UTXOs all have same value = min of inputs (2 in this case)

All three post sigs on Pastebin \Rightarrow one of them posts Tx on chain.

Coinjoin drawbacks

In practice: each CoinJoin Tx mixes about 40 inputs

- Large Tx: 40 inputs, 80 outputs

All participants must sign CoinJoin Tx for it to be valid

⇒ ensures all of them approve the CoinJoin Tx

... but any one of them can disrupt the process

Beyond simple mixing

Private Tx on a public blockchain

Can we have private transactions on a public blockchain?

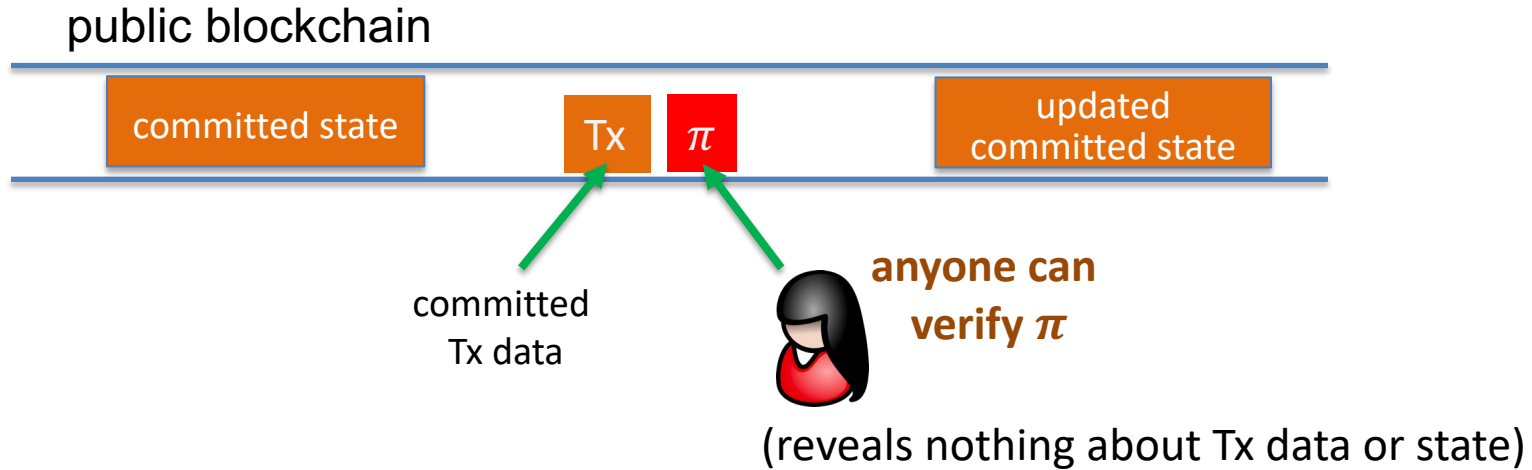
Naïve reasoning:

universal verifiability \Rightarrow transaction data must be public
otherwise, how we can verify Tx ??

crypto magic \Rightarrow private Tx on a publicly verifiable blockchain

Crypto tools: **commitments** and **zero knowledge proofs**

A paradigm for Private Tx



Committed data: short (hiding) commitment on chain

Proof π : succinct *zero-knowledge proof* that

- (1) committed Tx data is consistent with committed current state, and
- (2) committed updated state is correct

Review: cryptographic commitments

Cryptographic commitment: emulates an envelope



Many applications: e.g., a DAPP for a sealed bid auction

- Every participant **commits** to its bid,
- Once all bids are in, everyone opens their commitment

Cryptographic Commitments

Syntax: a commitment scheme is two algorithms

• commit(*msg*, *r*) \rightarrow *com*

secret randomness

commitment string

• verify(*msg*, *com*, *r*) \rightarrow accept or reject

anyone can verify that commitment was opened correctly

Commitments: security properties

- **binding**: Bob cannot produce two valid openings for **com**
More precisely: no efficient adversary can produce
 $\mathbf{com}, (m_1, r_1), (m_2, r_2)$
such that $\text{verify}(m_1, \mathbf{com}, r_1) = \text{verify}(m_2, \mathbf{com}, r_2) = \text{accept}$
and $m_1 \neq m_2$.

- **hiding**: **com** reveals nothing about committed data
 $\text{commit}(m, r) \rightarrow \mathbf{com}$, and r is sampled uniformly in a set R ,
then **com** is statistically independent of m

Example: hash-based commitment

Fix a hash function $H: M \times R \rightarrow C$ (e.g., SHA256)

where H is collision resistant, and $|R| \gg |C|$

- $\text{commit}(m \in M, r \leftarrow R)$: $\mathbf{com} = H(m, r)$
- $\text{verify}(m, \mathbf{com}, r)$: accept if $\mathbf{com} = H(m, r)$

binding: follows from collision resistance of H

hiding: follows from a mild assumption on H

What is a zk-SNARK?

Succinct zero knowledge proofs:
an important tool for privacy on the blockchain

What is a zk-SNARK ?

(intuition)

SNARK: a succinct proof that a certain statement is true

Example statement: “I know an m such that $\text{SHA256}(m) = 0$ ”

- **SNARK:** the proof is “**short**” and “**fast**” to verify
[if m is 1GB then the trivial proof (the message m) is neither]
- **zk-SNARK:** the proof “reveals nothing” about m

zk-SNARK: Blockchain Applications

Private Tx on a public blockchain:

- Tornado cash, Zcash, IronFish
- Private Dapps: Aleo

Compliance:

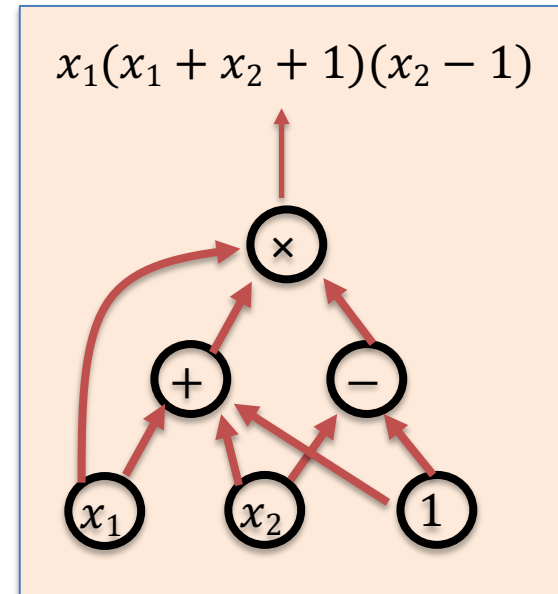
- Proving that private Tx are in compliance with banking laws
- Proving solvency in zero-knowledge

Scalability: privacy in a zk-SNARK Rollup (next week)

Bridging between blockchains: zkBridge

Arithmetic circuits

- Fix a finite field $\mathbb{F} = \{0, \dots, p - 1\}$ for some prime $p > 2$.
- **Arithmetic circuit:** $C: \mathbb{F}^n \rightarrow \mathbb{F}$
 - directed acyclic graph (DAG) where internal nodes are labeled $+$, $-$, or \times
inputs are labeled $1, x_1, \dots, x_n$
 - defines an n -variate polynomial with an evaluation recipe
- $|C| = \# \text{ gates in } C$



Interesting arithmetic circuits

Examples:

- $C_{\text{hash}}(h, \mathbf{m})$: outputs 0 if $\text{SHA256}(\mathbf{m}) = h$, and $\neq 0$ otherwise

$$C_{\text{hash}}(h, \mathbf{m}) = (h - \text{SHA256}(\mathbf{m})) , \quad |C_{\text{hash}}| \approx 20\text{K gates}$$

- $C_{\text{sig}}(\text{pk}, m, \sigma)$: outputs 0 if σ is a valid ECDSA signature on m with respect to pk

(preprocessing) NARK: Non-interactive ARgument of Knowledge

Public arithmetic circuit: $C(x, w) \rightarrow \mathbb{F}$

public statement in \mathbb{F}^n

secret witness in \mathbb{F}^m

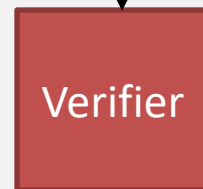
Preprocessing (setup): $S(C) \rightarrow$ public parameters (pp, vp)

pp, x, w



proof π that $C(x, w) = 0$

vp, x



accept or reject

(preprocessing) NARK: Non-interactive ARgument of Knowledge

A preprocessing NARK is a triple (S, P, V) :

- $S(C) \rightarrow$ public parameters (pp, vp) for prover and verifier
- $P(pp, \mathbf{x}, \mathbf{w}) \rightarrow$ proof π
- $V(vp, \mathbf{x}, \pi) \rightarrow$ accept or reject

NARK: requirements (informal)

Prover P($pp, \mathbf{x}, \mathbf{w}$)

Verifier V(vp, \mathbf{x}, π)



Complete: $\forall \mathbf{x}, \mathbf{w}: C(\mathbf{x}, \mathbf{w}) = 0 \Rightarrow \Pr[V(vp, \mathbf{x}, P(pp, \mathbf{x}, \mathbf{w})) = \text{accept}] = 1$

Adaptively knowledge sound: V accepts \Rightarrow P “knows” \mathbf{w} s.t. $C(\mathbf{x}, \mathbf{w}) = 0$
(an extractor E can extract a valid \mathbf{w} from P)

Optional: Zero knowledge: $(C, pp, vp, \mathbf{x}, \pi)$ “reveal nothing” about \mathbf{w}

SNARK: a Succinct ARgument of Knowledge

A succinct preprocessing NARK is a triple (S, P, V):

- $S(C) \rightarrow$ public parameters (pp, vp) for prover and verifier

- $P(pp, x, w) \rightarrow$ short proof π ; $|\pi| = O_\lambda(\log(|C|))$

- $V(vp, x, \pi)$ fast to verify ; $\text{time}(V) = O_\lambda(|x|, \log(|C|))$

short “summary” of circuit

Why preprocess C ??

SNARK: a Succinct ARgument of Knowledge

A succinct preprocessing NARK is a triple (S, P, V) :

- $S(C) \rightarrow$ public parameters (pp, vp) for prover and verifier
- $P(pp, x, w) \rightarrow$ short proof π ; $|\pi| = O_\lambda(\log(|C|))$
- $V(vp, x, \pi)$ fast to verify ; $\text{time}(V) = O_\lambda(|x|, \log(|C|))$

SNARK: (S, P, V) is **complete**, **knowledge sound**, and **succinct**

zk-SNARK: (S, P, V) is a SNARK and is **zero knowledge**

The trivial SNARK is not a SNARK

- (a) Prover sends w to verifier,
- (b) Verifier checks if $C(x, w) = 0$ and accepts if so.

Problems with this:

- (1) w might be secret: prover does not want to reveal w to verifier
- (2) w might be long: we want a “short” proof
- (3) computing $C(x, w)$ may be hard: we want a “fast” verifier

END OF LECTURE

Next lecture:
more on zk-SNARKs and their applications