CS251 Fall 2020

(cs251.stanford.edu)

Using zkSNARKs

Dan Boneh

Recap: high-level goals

- Private transactions on a public blockchain
- Blockchain scaling, such as proof-based Rollup
- Privately prove compliance, such as a private proof of solvency

Recap: non-interactive proof systems (for NP)

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

public statement in $\mathbb{F}_p^n$ $\boldsymbol{x}$

$\boldsymbol{w}$ secret witness in $\mathbb{F}_p^m$

Let $\boldsymbol{x} \in \mathbb{F}_p^n$. Two standard goals for prover P:

(1) **<u>Soundness</u>**: convince Verifier that $\exists \boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

(e.g., $\exists \boldsymbol{w}$ such that [$H(\boldsymbol{w}) = \boldsymbol{x}$ and $0 < \boldsymbol{w} < 2^{60}$])

(2) **<u>Knowledge</u>**: convince Verifier that P "knows" $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

(e.g., P knows a $\boldsymbol{w}$ such that $H(\boldsymbol{w}) = \boldsymbol{x}$)

Non-interactive Proof Systems (for NP)

A **non-interactive proof system** is a triple (S, P, V):

- **S**(C) $\rightarrow$ public parameters (S_p, S_v) for prover and verifier

  (S_p, S_v) is called a *reference string*

- **P**(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$) $\rightarrow$ proof π

- **V**(S_v, $\boldsymbol{x}$, $\boldsymbol{\pi}$) $\rightarrow$ accept or reject

proof systems: properties (informal)

Prover P(***pp***, x, w) — Verifier V (***pp***, x, π)

proof π →

accept or reject

Complete: $\forall x, w$: $C(x, w) = 0 \Rightarrow$ V(S_v, x, P(S_p, x, w)) = accept

Proof of knowledge: V accepts $\Rightarrow$ P "knows" w s.t. $C(x, w) = 0$

in some cases, **soundness** is sufficient: $\exists w$ s.t. $C(x, w) = 0$

Zero knowledge (optional): (x, π) "reveals nothing" about w

SNARK: succinct argument of knowledge

Goal: P wants to show that it knows $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

Succinct:

- Proof π should be **short** [i.e., $|\pi| = O(\mathbf{log}(|\boldsymbol{C}|), \lambda)$]
- Verifying π should be **fast** [i.e., time(V) = $O(|x|, \mathbf{log}(|\boldsymbol{C}|), \lambda)$]

note: if SNARK is zero-knowledge, then called a **zkSNARK**

zkSNARK applications

Blockchain Applications

Scalability:

- SNARK Rollup (zkSNARK for privacy from public)

Privacy: Private Tx on a public blockchain

- Confidential transactions
- Zcash

Compliance:

- Proving solvency in zero-knowledge
- Zero-knowledge taxes

… but first: 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*) → *com*

- **verify**(*msg*, *com*, *r*) → accept or reject

anyone can verify that commitment was opened correctly

Commitments: security properties

- **binding**: Bob cannot produce two valid openings for ***com***.

  Formally: no efficient adversary can produce

  com, (m_1, r_1), (m_2, r_2)

  such that verify(m_1, ***com***, r_1) = verify(m_2, ***com***, r_2) = accept

  and $m_1 \neq m_2$.

- **hiding**: ***com*** reveals nothing about committed data

  commit(m, r) $\rightarrow$ ***com***, and r is uniform in R $(r \leftarrow R)$,

  then ***com*** is statistically independent of m

Example 1: hash-based commitment

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

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

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

binding: follows from collision resistance of H

hiding: follows from a mild assumption on H

Example 2: Pedersen commitment

G = finite cyclic group = {1, g, g^2, ..., g^{q-1}} where $g^i \cdot g^j = g^{(i+j \bmod q)}$

q = $|G|$ is called the **order** of G. Assume q is a prime number.

Fix h in G and let R = {0, 1 ,..., q-1}. For m, r ∈ R define

$$H(m, r) = g^m \cdot h^r \in G$$

Fact: for a "cryptographic" group G, this H is collision resistant.

⇒ commitment scheme: **commit** and **verify** as in example 1

commit$(m \in R,\ r \leftarrow R) = H(m, r) = g^m \cdot h^r$

An interesting property

$$\textbf{commit}(m \in R,\ r \leftarrow R) = H(m, r) = g^m \cdot h^r$$

Suppose: $\text{commit}(m_1 \in R,\ r_1 \leftarrow R) \rightarrow com_1$

$\text{commit}(m_2 \in R,\ r_2 \leftarrow R) \rightarrow com_2$

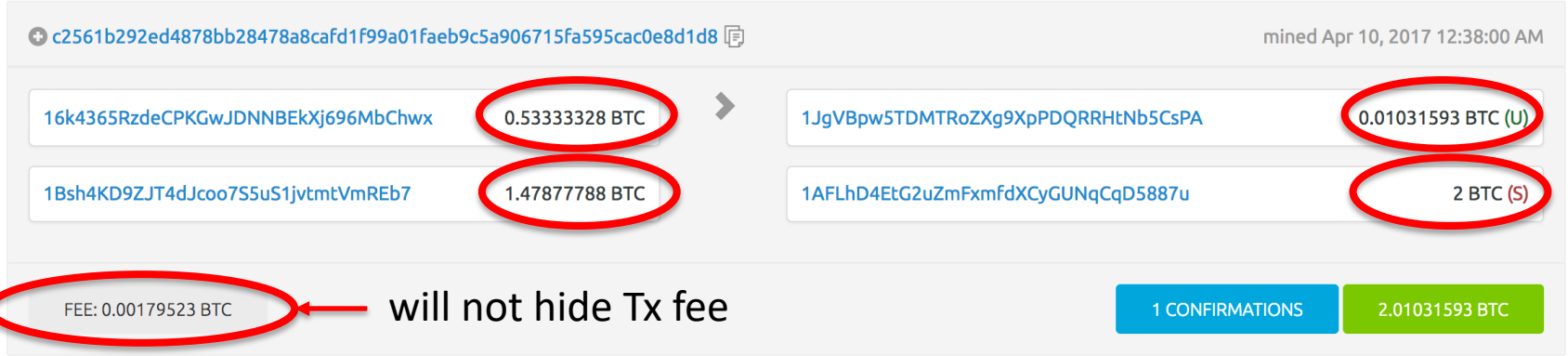
Then: $com_1 \times com_2 = g^{m1+m2} \cdot h^{r1+r2} = \text{commit}(m_1+m_2,\ r_1+r_2)$

$\Rightarrow$ anyone can sum committed value

Confidential Transactions

Confidential Tx (CT)

Goal: hide amounts in Bitcoin transactions.



⇒ businesses cannot use for supply chain payments

Confidential Tx: how?

Bitcoin Tx today:

Google: **30** → Alice: **1**, Google: **29**

8 bytes

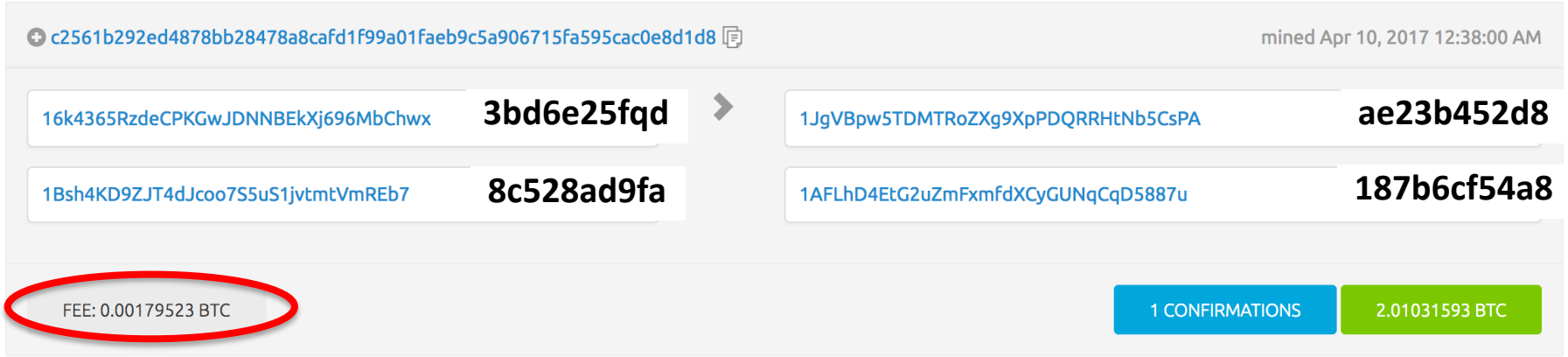
The plan: replace amounts by commitments to amounts

Google: **com$_1$** → Alice: **com$_2$**, Google: **com$_3$**

32 bytes

where **com$_1$** = commit(30, r_1), **com$_2$** = commit(1, r_2), **com$_3$** = commit(29, r_3)

Now blockchain hides amounts



How much was transferred ???

The problem: how will miners verify Tx?

Google: $\mathbf{com_1}$ ⇀ Alice: $\mathbf{com_2}$, Google: $\mathbf{com_3}$

$\mathbf{com_1}$ = commit(30, r_1), $\mathbf{com_2}$ = commit(1, r_2), $\mathbf{com_3}$ = commit(29, r_3)

Solution: zkSNARK (special purpose, optimized for this problem)

- Google: (1) privately send r_2 to Alice
  (2) construct a zkSNARK π where

statement = x = ($\mathbf{com_1}$, $\mathbf{com_2}$, $\mathbf{com_3}$)
witness = w = (m_1, r_1, m_2, r_2, m_3, r_3)

and circuit $C(x,w)$ outputs 0 if:

CT arithmetic circuit

(i) $\mathbf{com_i}$ = commit(m_i, r_i) for i=1,2,3,
(ii) $m_1 = m_2 + m_3 + \text{TxFees}$,
(iii) $m_2 \geq 0$ and $m_3 \geq 0$

The problem: how will miners verify Tx?

- Google: (1) privately send r_2 to Alice
  (2) construct zkSNARK proof π that Tx is valid
  (3) append π to Tx (need short proof! $\Rightarrow$ zkSNARK)

Tx: proof π , Google: **com$_1$** $\rightharpoonup$ Alice: **com$_2$**, Google: **com$_3$**

- Miners: accept Tx if proof π is valid (need fast verification)
  $\Rightarrow$ learn Tx is valid, but amounts are hidden

Optimized proof?

Note: Alice needs r_2 to spend her UTXO

otherwise: she cannot construct proof π

statement = x = (com_1, com_2, com_3)

witness = w = (m_1, r_1, m_2, r_2, m_3, r_3)

circuit $C(x,w)$ outputs 0 if:

(i) $com_i = commit(m_i, r_i)$,

~~(ii) $m_1 = m_2 + m_3 + TxFees$,~~

(iii) $m_2 \geq 0$ and $m_3 \geq 0$

Easy to check with Pedersen:

set $com = com_1/com_2 \cdot com_3 \cdot g^{TxFees}$

prove that $com = commit(0, r)$

remaining proof is ≈400 bytes

Zcash (simplified)

Zcash

Goal: fully private payments … like cash, but across the Internet

challenge: will governments allow this ???

Zcash blockchain supports two types of TXOs:

- transparent TXO (as in Bitcoin)
- shielded (anonymized)

a Tx can have both types of inputs, both types of outputs

Addresses and TXOs

H_1, H_2, H_3: cryptographic hash functions.

sk needed to spend TXO for address pk

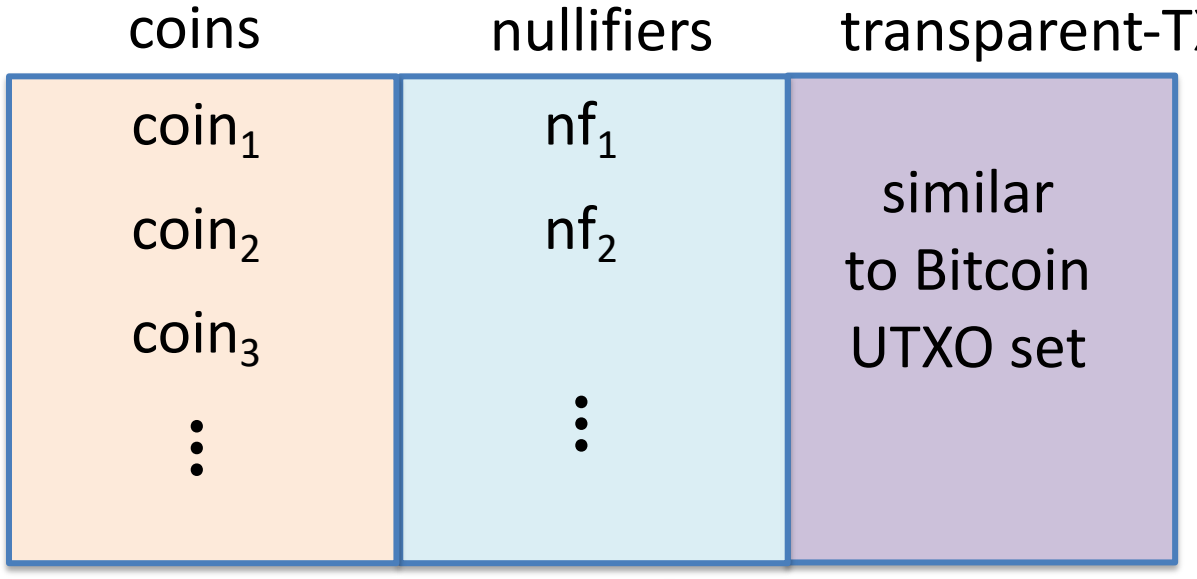
(1) shielded address: random sk $\leftarrow$ X, pk = H_1(sk)

(2) **shielded TXO** (note) owned by address pk:

- TXO owner has (from payer): value v and r $\leftarrow$ R
- on blockchain: coin = H_2((pk, v) , r) (commit to pk, v)

pk: addr. of owner, v: value of coin, r: random chosen by payer

The blockchain



just Merkle root … append only tree
(coins are never removed)

explicit list:
one entry per **spent coin**

Transactions: an example

owner of **coin** = H_2((pk, v) , r) (Tx input)

wants to send **coin** funds to: shielded pk', v' (Tx output)

(v = v' + v'') transp. pk'', v''

step 1: construct new coin: **coin'** = H_2((pk', v') , r')

by choosing random r' ⇐ R (and sends v', r' to owner of pk')

step 2: compute **nullifier** for spent coin **nf** = H_3(sk, index of coin in Merkle tree)

nullifier **nf** is used to "cancel" **coin** (no double spends)

key point: miners learn that some coin was spent, but not which one!

Transactions: an example

step 3: construct a zkSNARK proof π for

statement = x = (current Merkle root, **coin'**, **nf**, v'')

witness = w = (sk, (v, r), (pk', v', r'), Merkle proof for **coin**)

C(x, w) outputs 0 if: with **coin** := H_2((pk=H_1(sk), v), r) check

The Zcash circuit

(1) Merkle proof for **coin** is valid,

(2) **coin'** = H_2((pk', v') , r')

(3) v = v' + v'' and v' ≥ 0 and v'' ≥ 0,

(4) **nf** = H_3(sk, index-of-coin-in-Merkle-tree)

from Merkle proof

What is sent to miners

step 4: send (**coin'**, **nf**, transparent-TXO, proof π) to miners,

send (v' , r') to owner of pk'

step 5: miners verify

(i) proof π and transparent-TXO

(ii) verify that **nf** is not in nullifier list (prevent double spending)

if so, add **coin'** to Merkle tree, add **nf** to nullifier list,

add transparent-TXO to UTXO set.

Summary

- Tx hides which coin was spent

  ⇒ **coin** is never removed from Merkle tree,
  but cannot be double spent thanks to nullifer

  note: prior to spending **coin**, only owner knows **nf**:

  $$\mathbf{nf} = H_3(sk, \text{index of coin in Merkle tree})$$

- Tx hides address of **coin'** owner

- Miners can verify Tx is valid, but learn nothing about Tx details.

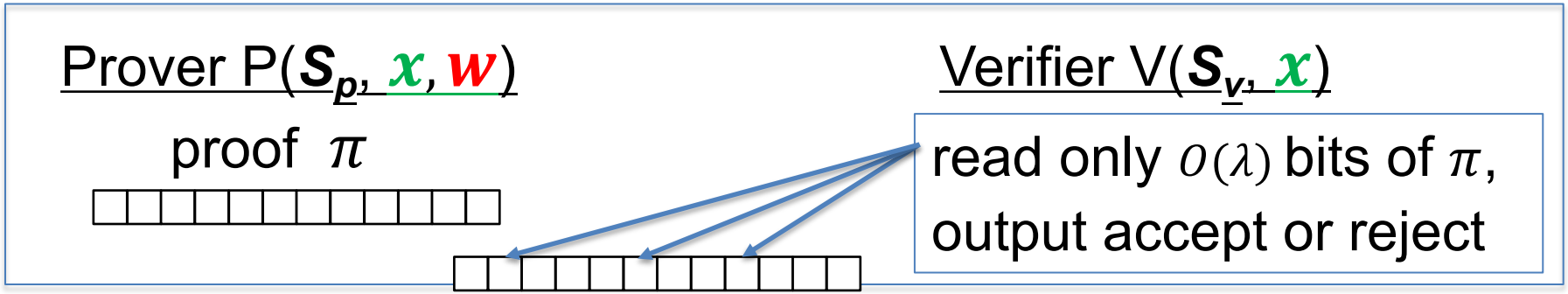
A simple PCP-based SNARK

[Kilian’92, Micali’94]

A simple construction: PCP-based SNARK

The PCP theorem: Let $C(x, w)$ be a circuit where $x \in \mathbb{F}_p^n$.

there is a proof system that for every x proves $\exists w: C(x, w) = 0$ as follows:

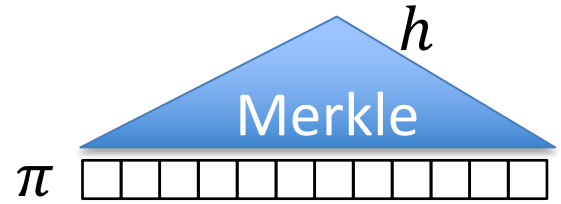


V always accepts valid proof. If no w, then V rejects with high prob.

size of proof is $poly(|C|)$. (not succinct)

Converting a PCP proof to a SNARK

Prover P(S_p, x, w) Verifier V(S_v, x)



Merkle root h → 1 hash

open $O(\lambda)$ positions of π →

← $O(\lambda \log |C|)$ hashes

$O(\lambda)$ opening and Merkle proofs →

output accept or reject

Verifier sees $O(\lambda \log |C|)$ data ⇒ succinct proof.

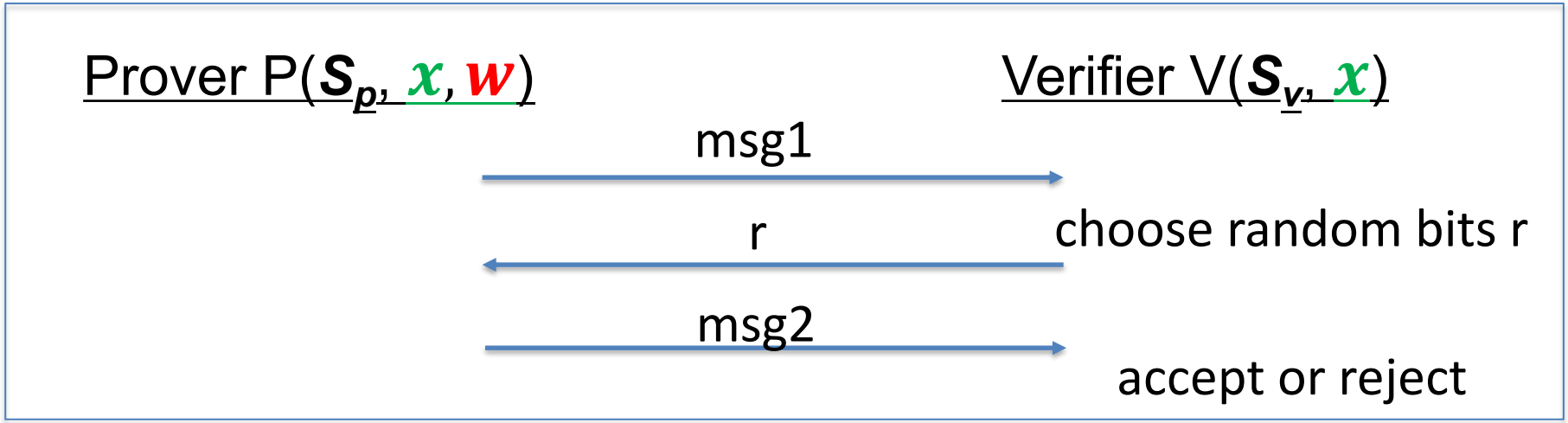
Problem: **interactive**

Making the proof non-interactive

The **Fiat-Shamir heuristic:**

- public-coin interactive protocol ⇒ non-interactive protocol

  public coin: all verifier randomness is public (no secrets)



Making the proof non-interactive

Fiat-Shamir heuristic: $H: M \to R$ a cryptographic hash function

- idea: prover generates random bits on its own (!)

Prover P(S_p, x, w)

generate msg1

r ⇐ H(x, msg1)

generate msg2

π = (msg1, msg2)

$|\pi| = O(\lambda \log |C|)$

Verifier V(S_v, x)

r ⇐ H(x, msg1)

accept or reject

Thm: this is a secure SNARK assuming H is a random oracle

Are we done?

Simple transparent SNARK from the PCP theorem

- Use Fiat-Shamir heuristic to make non-interactive
- We will apply Fiat-Shamir in many other settings

The bad news: an impractical SNARK --- Prover time too high

Better SNARKs: next lecture! Goal: Time(Prover) = O(|C|)

END OF LECTURE

Next lecture: How to build an efficient SNARK