SNARK construction & Groth 16

ZK-School Beginner class

Mentor: Inseon Yu

Outline

- SNARK Construction
- Review: Basic Knowledge
- Groth 16
- PCP based vs Modern SNARKs

Table of contents

Why do we use SNARK?

- **SNARK:** Succinct Non-interactive Argument of Knowledge

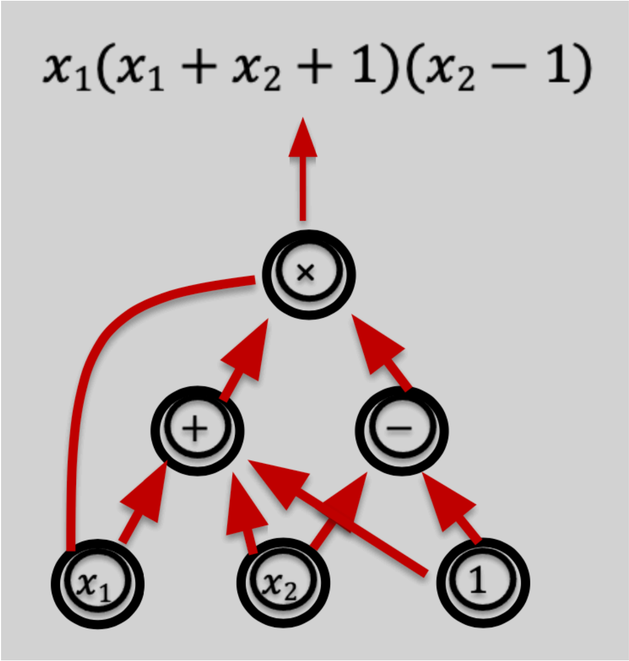
- **Babai-Fortnow-Levin-Szegedy 1991:**

  In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with unreliable software.

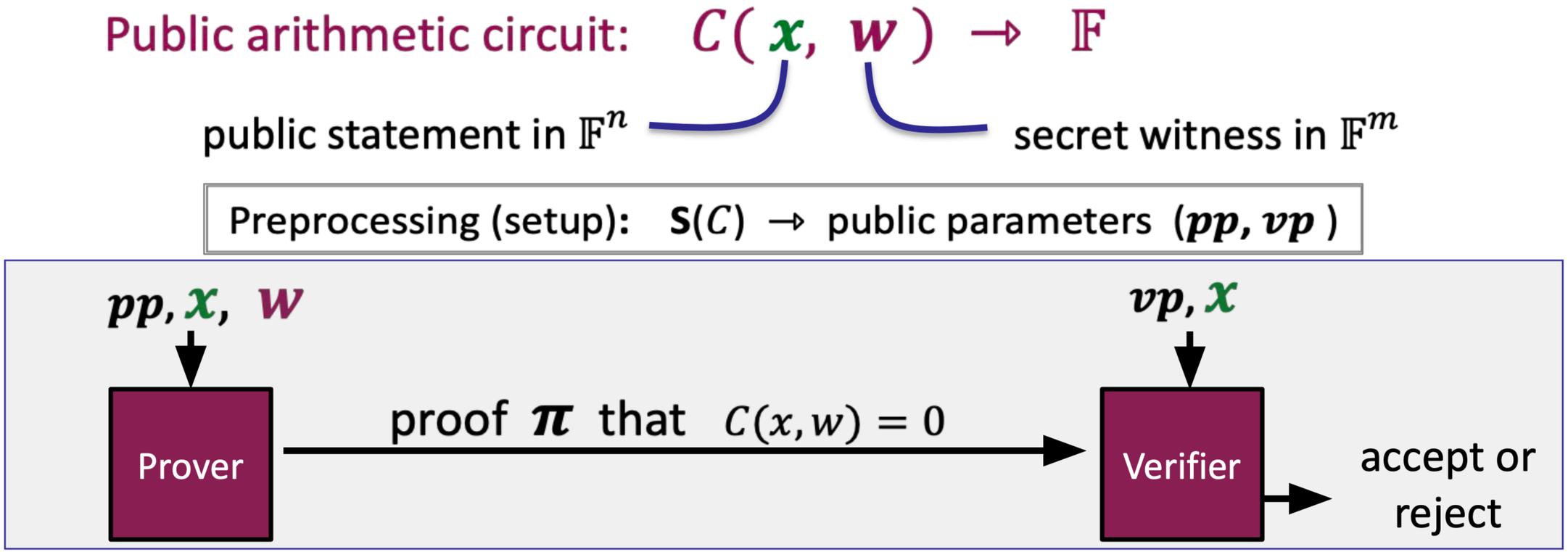
Arithmetic circuit

- Fix a finite field $F_P = \{0, \ldots, p-1\}$ for some prime p.

- Arithmetic circuit $C : F^n \to F$



NARK: Non-Interactive Argument of Knowledge



NARK: Non-Interactive Argument of Knowledge

- $S(C) \rightarrow (pp, vp)$
- $P(pp, x, w) \rightarrow \pi$
- $V(vp, x, \pi) \rightarrow$ accept or reject
- Completeness: $\forall x, w : C(x, w) = 0 \rightarrow Pr[V(vp, x, P(pp, x, w)) = accept] = 1$
- Soundness: V accepts -> P “knows” w s.t. C(x,w) = 0
- ZK(optional)

SNARK: Succinct NARK

- $S(C) \rightarrow (pp, vp)$
- $P(pp, x, w) \rightarrow short\ \pi;\ len(\pi) = O(\log|C|)$
- $V(vp, x, \pi) \rightarrow$ accept or reject; $time(V) = O(|x|, \log|C|)$

Types of Preprocessing

- Trusted Setup per circuit
- Universal Trusted Setup
- Transparent

	Proof size	Verifier time	Setup
Groth'16	≈ 200 Bytes O(1)	≈ 1.5 ms O(1)	per circuit
PLONK/Marlin	≈ 400 Bytes O(1)	≈ 3 ms O(1)	Universal
Bulletproof	≈ 1.5 KB O(log|C|)	≈ 3 sec O(|C|)	Transparent
STARK	≈ 100 KB O(log^2|C|)	≈ 10 ms O(log^2|C|)	Transparent

(For a circuit with ≈2^20 gates)

Before we go any further…

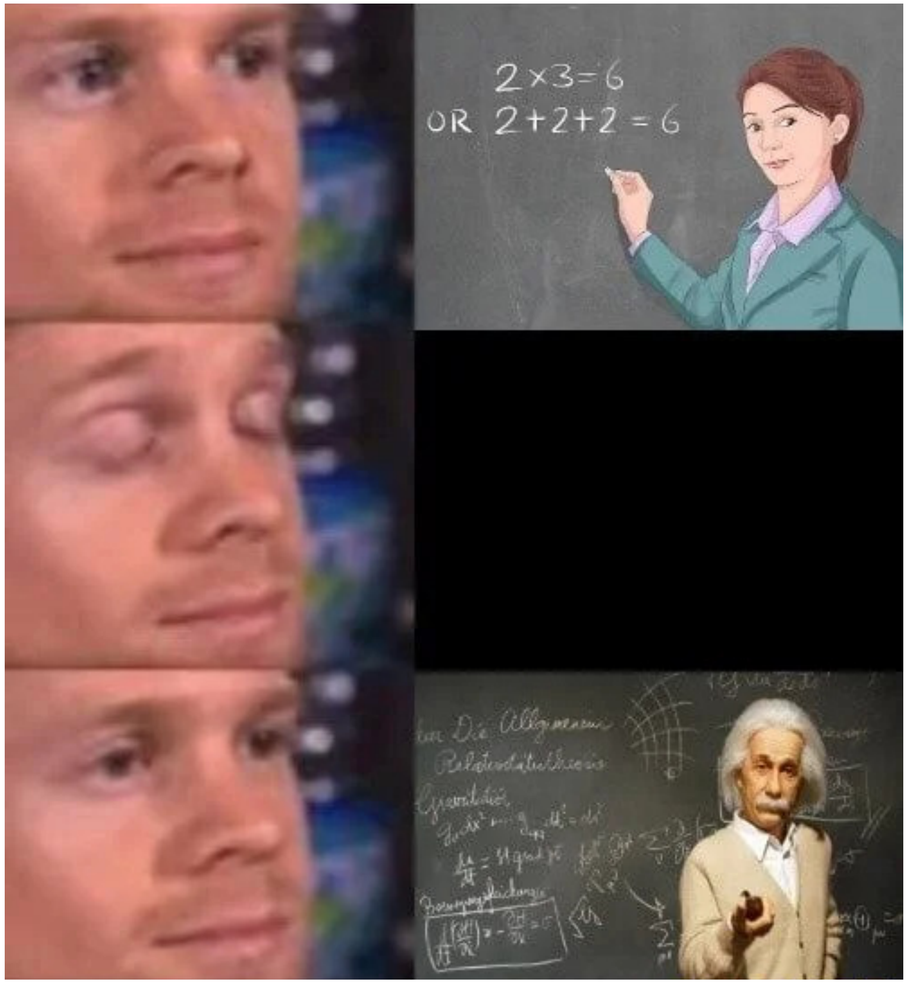


Table of contents

Review: Elliptic Curves

- Weierstrass normal form:

$$y^2 = x^3 + ax + b,\ \ 4a^3 + 27b^2 \neq 0$$

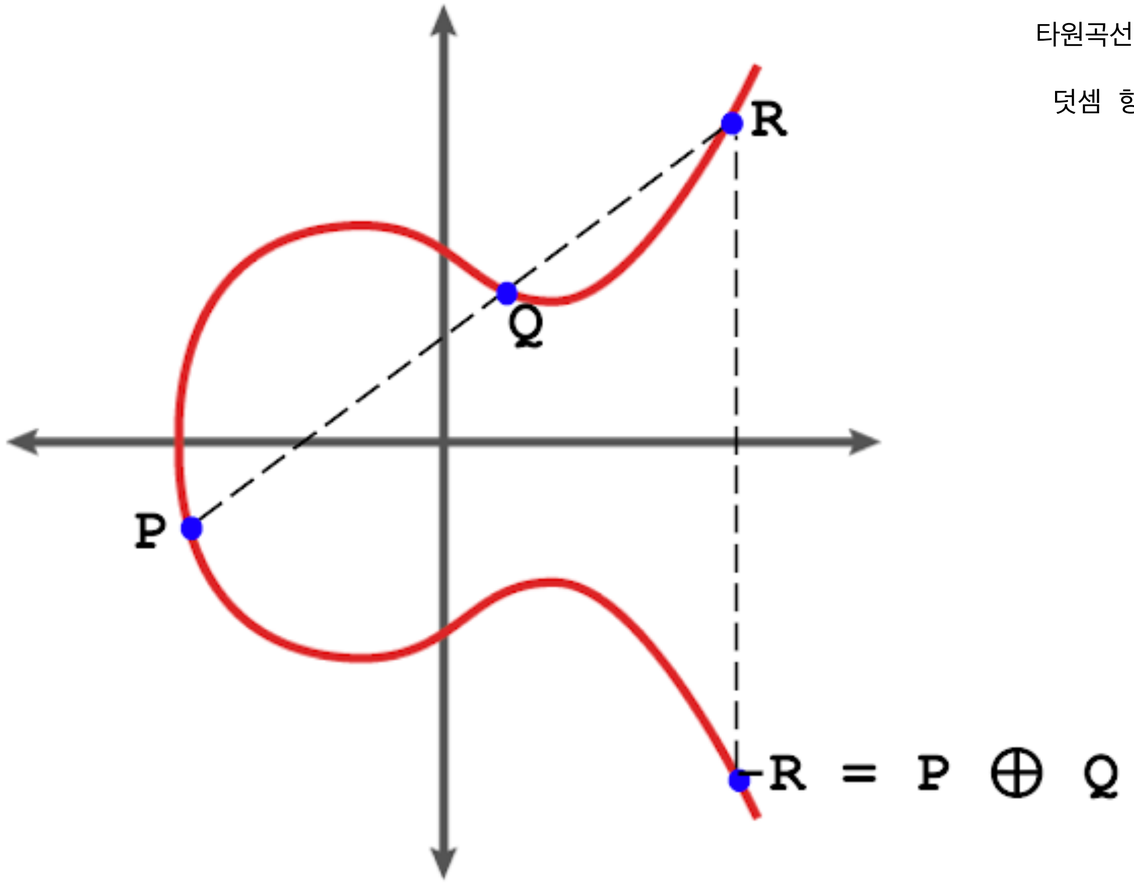
- Secp256k1: $y^2 = x^3 + 7$

  이더리움, 비트코인 키생성시 사용

- BLS12-381: $y^2 = x^3 + 4$

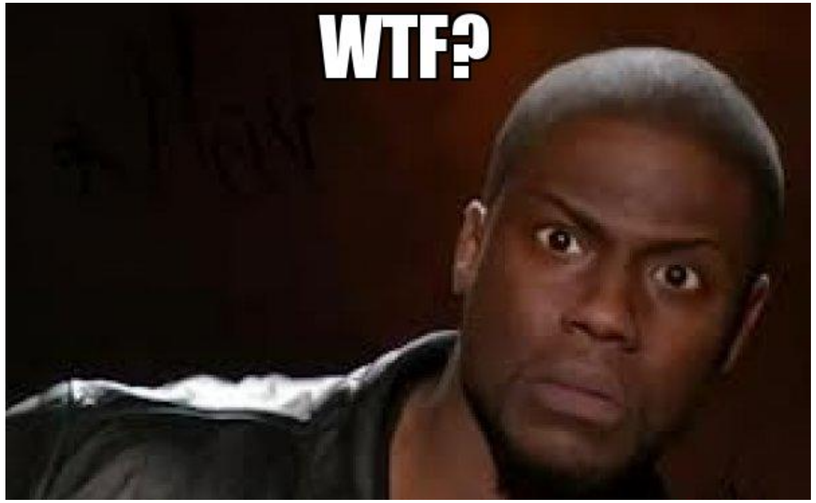
  오늘공부할꺼임

Review: Elliptic Curves



타원곡선의 정의를 내려보면

덧셈 항등원 정의



Review: Finite field

DLP
g랑 x를 알고있으면 y를 알기쉽지만
g랑 y를 알아도 x는 구하기어렵다

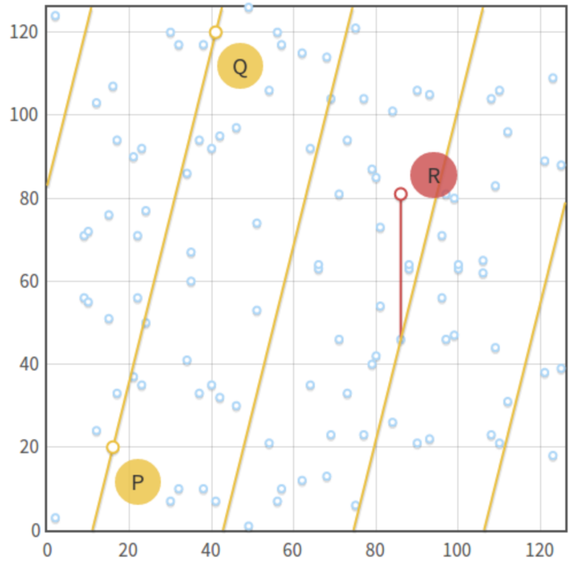
- Discrete Logarithm Problem: $g^x \equiv y \mod p$

  Easy to get **y** from g and x,

  but hard to know **x** from g and y

- Elliptic Curve Discrete Logarithm Problem (ECDLP):

  $aG \equiv b \mod p$



Review: Elliptic curves pairing

- **DH problem:** It's difficult to get abG from aG and bG.

- **CDH(Computational Diffie-Hellman):**

  Solving the exact value of abG from aG, bG

  cbG 를가지고 aG, bG를 구한다
  결과값을 나와서 키값을 알고 검증이된다.

- **DDH(Decisional Diffie-Hellman):**

  Determining if abG is valid

  dbG를 연산할수없지만 검증은할수있다.
  키없이 검증할수있는거

Can we make DDH possible while CDH is impossible…?

Review: Elliptic curves pairing

- **Base Field**

  - $p = 3$

  - $F_p = (0,1,2)$

  - $k = 2$

- **Extended Field ($z^2 + 1 = 0$)**

  - $F_{p^2} = (0,\ 1,\ 2,\ z,\ z+1,\ z+2,\ 2z,\ 2z+1,\ 2z+2)$

  $\rightarrow (z+1) + (z+2) = 2z$

  $\rightarrow (z+1) * z = z^2 + z = z - 1 = z + 2$

Review: Elliptic curves pairing

$$y^2 = x^3 \rightarrow$$

$x \rightarrow x^3$	$y \rightarrow y^2$
0 → 0	0 → 0
1 → 1	1 → 1
2 → 2	2 → 1
z → 2z	z → 2
z+1 → 2z+1	z+1 → 2z
z+2 → 2z+2	z+2 → z
2z → z	2z → 2
2z+1 → z+1	2z+1 → 2
2z+2 → z+2	2z+2 → 2z

$$G2 = \{(0,0), (1,1), (1,2), (2,z), (2,2z), (2,2z+1), (z, z+1), (z,2z+2), (2z, z+2)\}$$

Review: Elliptic curves pairing

3차원으로 뽑아서 2점을찍어서 Gt를 뽑을수있나봄

- **Pairing :**

  $e : G1 \times G2 \rightarrow GT$

  Actual computation → Twist

- **Embedding degree k :**

  - $p^k - 1 = 0 \pmod{r}$

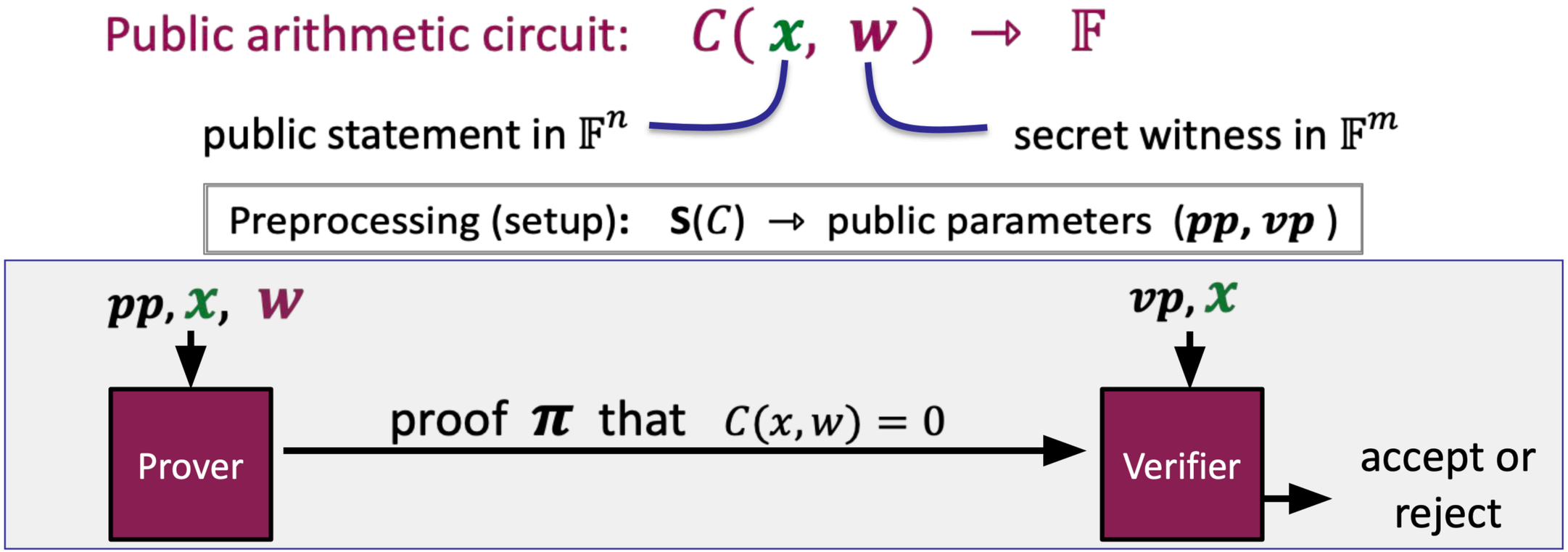
  - BLS12-381: k = 12, order = 381bit

- **Bilinearity :**

  $e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab} = e(P, aQ)^b = e(bP, aQ)$

Table of contents

Groth 16


Public arithmetic circuit: $C(x, w) \rightarrow \mathbb{F}$
public statement in $\mathbb{F}^n$
secret witness in $\mathbb{F}^m$
Preprocessing (setup): $\mathbf{S}(C) \rightarrow$ public parameters (pp, vp)
pp, x, w
vp, x
Prover
proof π that $C(x, w) = 0$
Verifier
accept or reject

Circuit → R1CS(Rank-1 Constraint System)

Computational problem: $x^3 + x + 5 = 35$

1. Flattening:

   sym_1 = x * x

   y = sym_1 * x

   sym_2 = y + x

   ~out = sym_2 + 5

2. Gates to R1CS

$$s \cdot a_1 * s \cdot b_1 - s \cdot c_1 = 0$$
$$s \cdot a_2 * s \cdot b_2 - s \cdot c_2 = 0$$
$$\vdots$$
$$s \cdot a_n * s \cdot b_n - s \cdot c_n = 0$$

[‘~one’, ‘x’, ‘~out’, ‘sym_1’, ‘y’, ‘sym_2’]

A:	B:	C:
[0, 1, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0]	[0, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 1, 0]	[1, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1]
[5, 0, 0, 0, 0, 1]	[1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0]

R1CS → QAP(Quadratic Arithmetic Program)

- Lagrange Interpolation:
  For x_i, y_i with $i \in [0,n)$, and $x_i \neq x_j \forall i \neq j$ $\rightarrow f$ of degree less then n exist .s.t $f(x_i) = y_i$

Making the Polynomial that passes through (1,3), (2,2), (3,4)

1. (2,0), (3,0)
   $\rightarrow (x-2)*(x-3)$

2. (1,3), (2,0), (3,0)
   $\rightarrow \frac{(x-2)*(x-3)*3}{((1-2)*(1-3))}$

3. (1,3), (2,2), (3,4)
   $\rightarrow \frac{(x-2)*(x-3)*3}{((1-2)*(1-3))} + \frac{(x-1)*(x-3)*2}{((2-1)*(2-3))} + \frac{(x-1)*(x-2)*4}{((3-1)*(3-2))}$

QAP(Quadratic Arithmetic Program)

- QAP:

$$A = [A_1(x), A_2(x), A_3(x), A_4(x), A_5(x), A_6(x)]$$

$$B = [B_1(x), B_2(x), B_3(x), B_4(x), B_5(x), B_6(x)]$$

$$C = [C_1(x), C_2(x), C_3(x), C_4(x), C_5(x), C_6(x)]$$

A(R1CS)	B(R1CS)	C(R1CS)
[0, 1, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0]	[0, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 1, 0]	[1, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1]
[5, 0, 0, 0, 0, 1]	[1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0]

A	**B**	**C**
[-5.0, 9.166, -5.0, 0.833]	[3.0, -5.166, 2.5, -0.333]	[0.0, 0.0, 0.0, 0.0]
[8.0, -11.333, 5.0, -0.666]	[-2.0, 5.166, -2.5, 0.333]	[0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0]	[0.0, 0.0, 0.0, 0.0]	[-1.0, 1.833, -1.0, 0.166]
[-6.0, 9.5, -4.0, 0.5]	[0.0, 0.0, 0.0, 0.0]	[4.0, -4.333, 1.5, -0.166]
[4.0, -7.0, 3.5, -0.5]	[0.0, 0.0, 0.0, 0.0]	[-6.0, 9.5, -4.0, 0.5]
[-1.0, 1.833, -1.0, 0.166]	[0.0, 0.0, 0.0, 0.0]	[4.0, -7.0, 3.5, -0.5]

$$\rightarrow Q(x) = A(x) * B(x) - C(x) = H(x) * (x-1)(x-2)(x-3)(x-4)$$

Setup

- BLS12-381

- What we use for the setup
  - Toxic waste: $\alpha, \beta, \gamma, \delta, \tau \in F_p$
  - m variables, of which l as public input
  - n constraints
  - $Z(x)$ from QAP
  - $L_i(x) = \beta * A_i(x) + \alpha * B_i(x) + C_i(x)$

CRS(Common Reference String)

- **Proving Key**

  - G1 elements: $(\alpha, \delta, \tau, \tau^2, \ldots, \tau^{n-1}, L_{l+1}(\tau)/\delta, \ldots, L_m(\tau)/\delta, Z(\tau)/\delta, \tau * Z(\tau)/\delta, \ldots, \tau^{n-2}Z(\tau)/\delta)_1$

  - G2 elements: $(\beta, \delta, 1, \tau, \tau^2, \ldots, \tau^{n-1})_2$

- **Verification Key**

  - G1 elements: $(1, L_0(\tau)/\gamma, L_1(\tau)/\gamma, \ldots, L_l(\tau)/\gamma)_1$

  - G2 elements: $(1, \gamma, \delta)_2$

  - GT element: $(\alpha_1 * \beta_2)_t$

Proof Generation

Given a witness vector $w = (1, w_1, w_2, \ldots, w_m)$, and two random elements r, s,

a proof is made of three points $\underline{A, B \text{ and } C}$

- $A(G1) = \alpha + w_0 * A_0(\tau) + w_1 * A_1(\tau) + \ldots + w_m * A_m(\tau) + r * \delta$
- $B(G2) = \beta + w_0 * B_0(\tau) + w_1 * B_1(\tau) + \ldots + w_m * B_m(\tau) + s * \delta$
- $C(G1) = w_{l+1} * L_{l+1}(\tau)/\delta + \ldots + w_m * L_m(\tau)/\delta + H(\tau) * Z(\tau)/\delta + s * A_1 + r * B_1 - r * s * \delta$

Verification

To verify a proof, we'll just need to check the following equality:

$$A_1 * B_2 = \alpha_1 * \beta_2 + (w_0 * L_0(\tau)/\gamma + \ldots + w_l * L_l(\tau)/\gamma)_1 * \gamma_2 + C_1 * \delta_2$$

- Left term $A * B$:

$$[\alpha + A(\tau) + r * \delta] * [\beta + B(\tau) + s * \delta]$$
$$= \alpha * \beta + \alpha * B(\tau) + s * \alpha * \delta + A(\tau) * B(\tau) + s * A(\tau) * \delta + r * \delta * \beta + r * B(\tau) * \delta + s * r * \delta * \delta$$
$$= A(\tau) * B(\tau) + \alpha * \beta + \alpha * B(\tau) + s * \alpha * \delta + s * A(\tau) * \delta + r * \delta * \beta + r * B(\tau) * \delta + s * r * \delta * \delta$$

- Right term:

$$= \alpha * \beta + L(\tau) + H(\tau) * Z(\tau) + s * \alpha * \delta + s * r * \delta * \delta + r * \beta * \delta + r * B(\tau) * \delta + s * r * \delta * \delta - r * s * \delta * \delta$$
$$= \alpha * \beta + \beta * A(\tau) + \alpha * B(\tau) + C(\tau) + H(\tau) * Z(\tau) + s * \alpha * \delta + s * A(\tau) * \delta + s * r * \delta * \delta + r * \delta * \beta + r * B(\tau) * \delta$$
$$= C(\tau) + H(\tau) * Z(\tau) + \alpha * \beta + \alpha * B(\tau) + s * \alpha * \delta + s * A(\tau) * \delta + r * \delta * \beta + r * B(\tau) * \delta + s * r * \delta * \delta$$

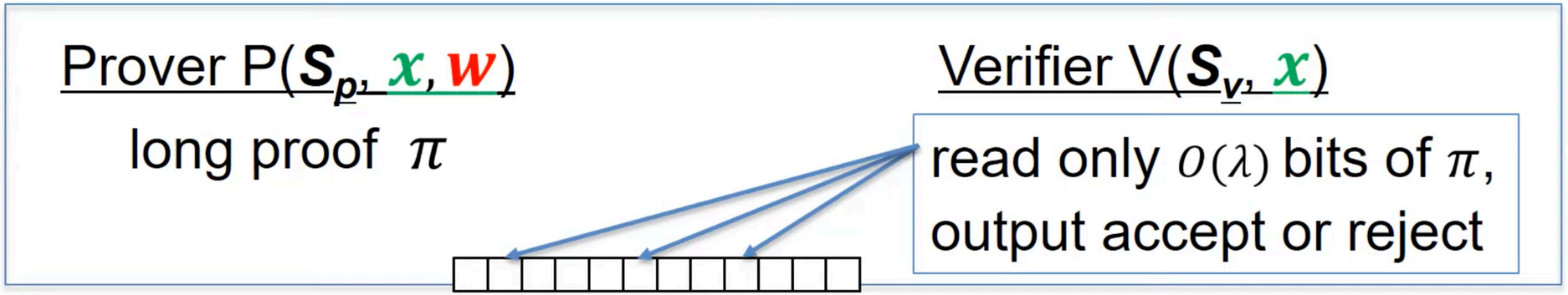
공통부분 지워서 계산을하면

$\rightarrow A(\tau) * B(\tau) = C(\tau) + H(\tau) * Z(\tau)$ (Schwarts-Zippel Lema)

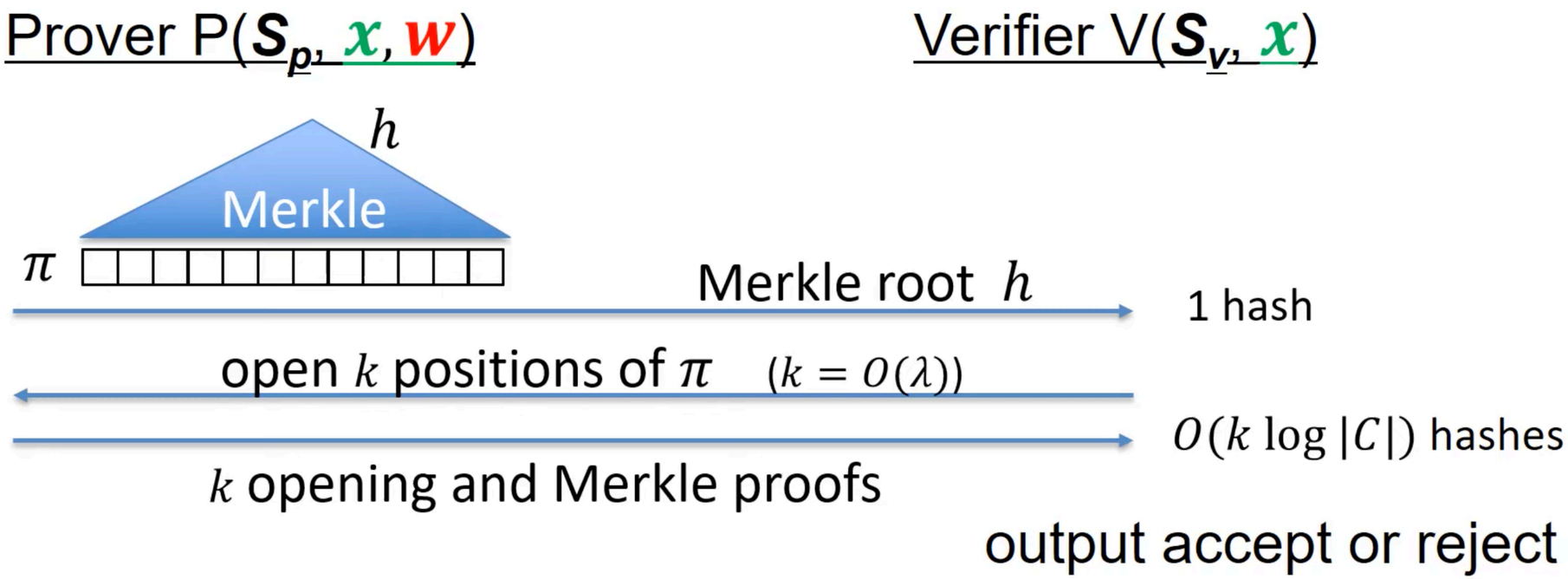
Table of contents

PCP based SNARK

PCP(Probabilistic Checkable Proof):



PCP based SNARK



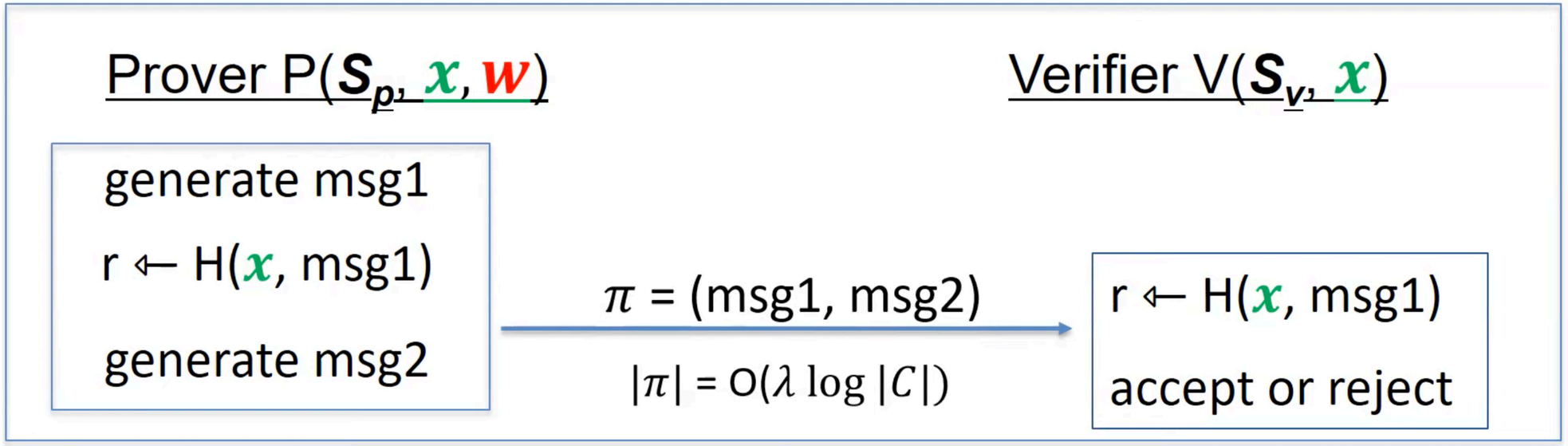
Verifier sees $O(\lambda \log |C|)$ data $\Rightarrow$ succinct proof.

Problem: **interactive**

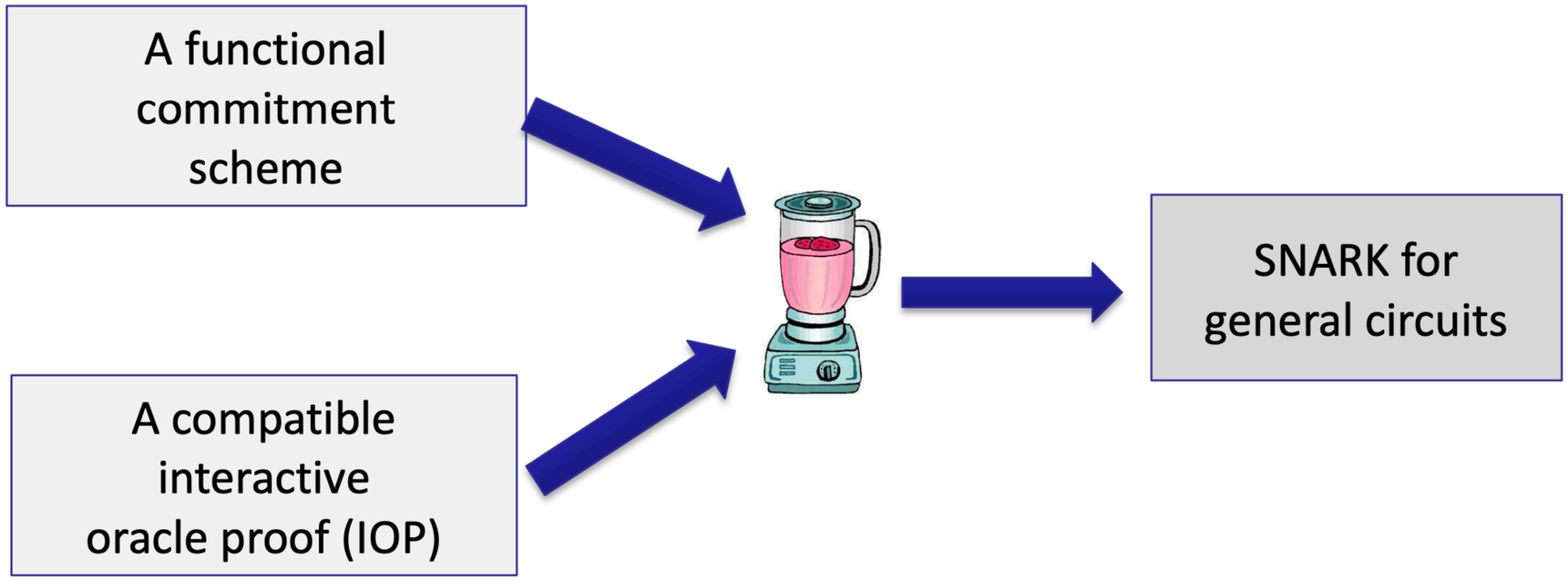
PCP based SNARK

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

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



Modern SNARKs



Thank you!!

Coming up: PCS