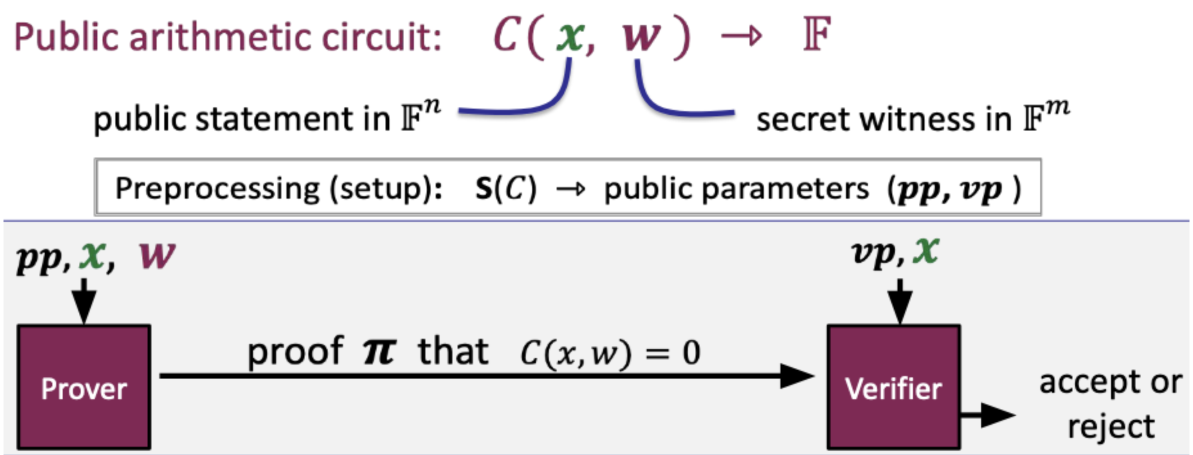
Research Presentation 1

Content Subj



We have three constrains :

- Complete

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

- Knowledge soundness

$$\text{V accepts} \ \Rightarrow\ \text{P “knows” } w \text{ s.t. } C(x,w) = 0$$

- Zero Knowledge

$$(C, pp, vp, x, \pi) \quad \text{“reveal nothing new” about } w$$

SNARK : Succinct ARgument of Knowledge :

- Prover: Short proof : len(pi) = sublinear (| W |)
- Verifier: fast to verify

Definitions: knowledge soundness

Formally: (S, P, V) is (adaptively) **knowledge sound** for a circuit C if

for every poly. time adversary $A = (A_0, A_1)$ such that

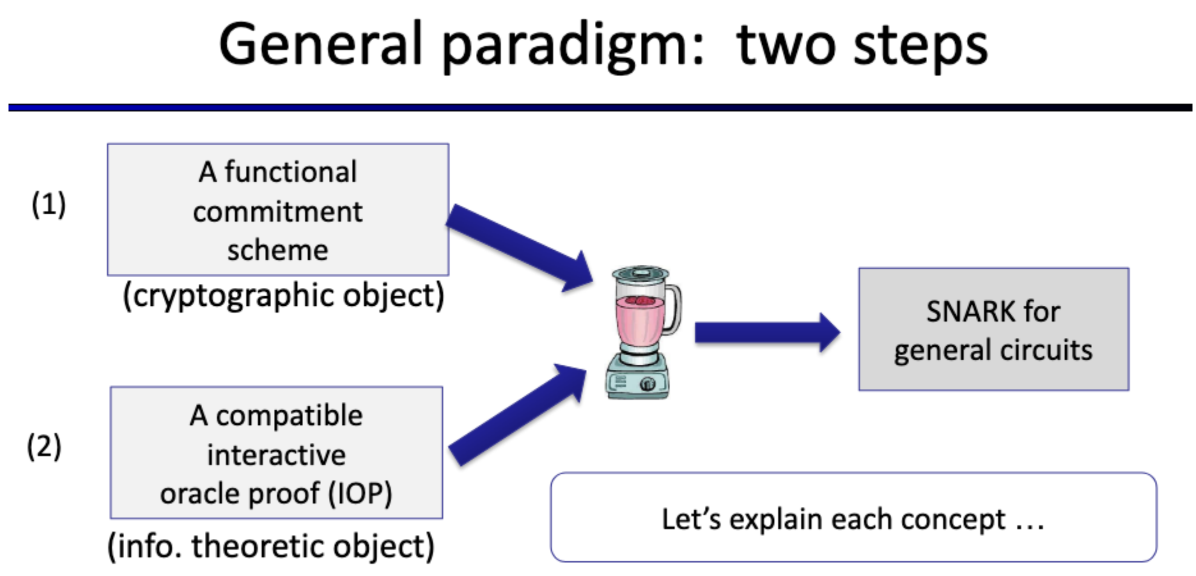
$gp \leftarrow S_{\text{init}}(\,)$, $(C, x, \text{st}) \leftarrow A_0(gp)$, $(pp, vp) \leftarrow S_{\text{index}}(C)$, $\pi \leftarrow A_1(pp, x, \text{st})$:

$\Pr[\, V(vp, x, \pi) = \text{accept} \,] > 1/10^6$ (non-negligible)

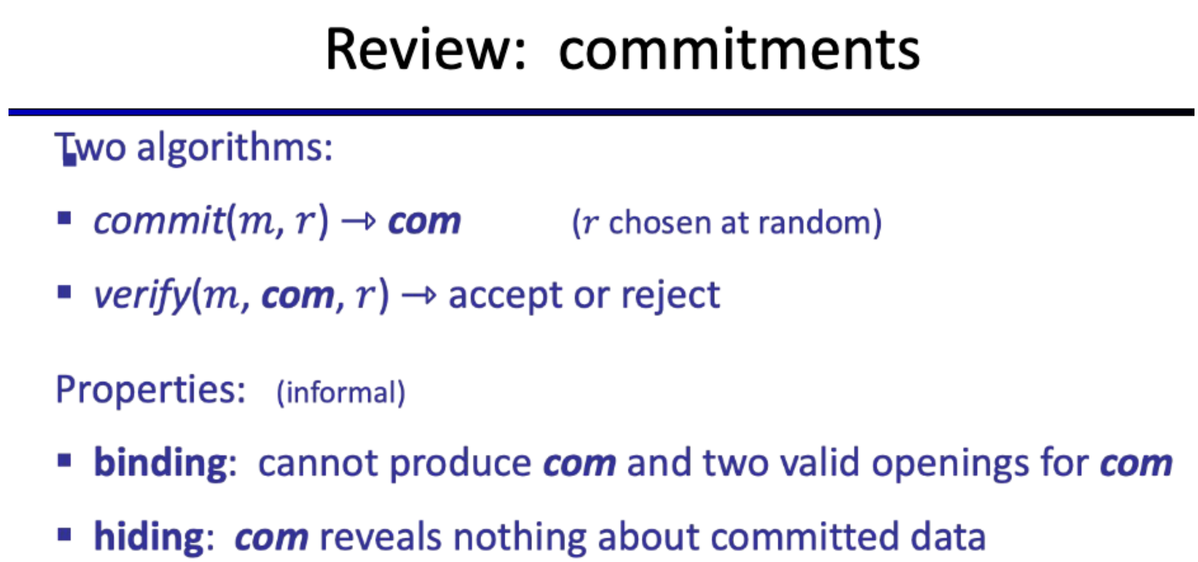
there is an efficient **extractor** E (that uses A) s.t.

$gp \leftarrow S_{\text{init}}(\,)$, $(C, x, \text{st}) \leftarrow A_0(gp)$, $\boxed{w \leftarrow E(gp, C, x):}$

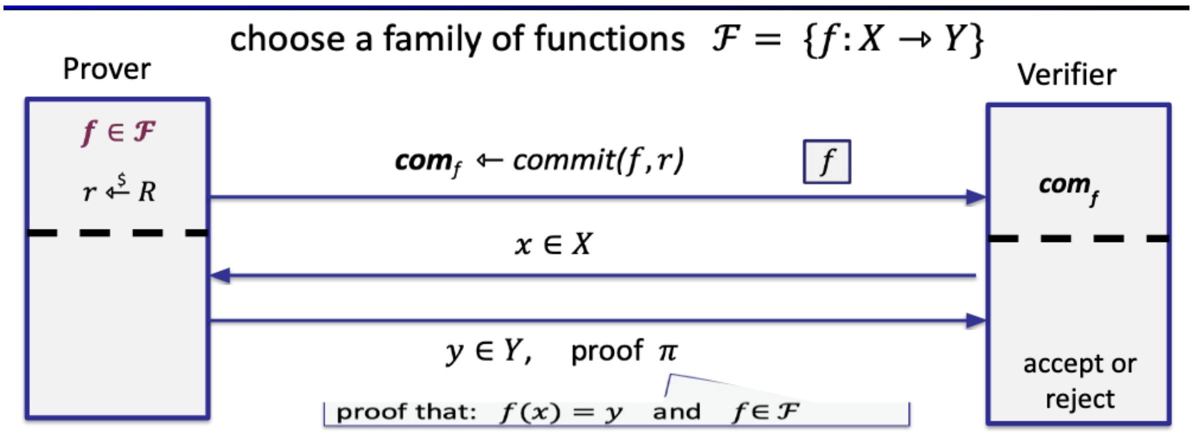
$\Pr[\, C(x, w) = 0 \,] > 1/10^6 - \epsilon$ (for a negligible ϵ)



I know about the first part but the second part is still vague for me.



We can use hash functions for constructing a standard commitment.



Four important functional commitments :

- Polynomial commitment.
- Multi linear commitment.
- Vector Commitment(Merkle Trees)
- Inner Product commitments.

Polynomial Commitment:

prover commits to a poly f(x) in Fp[x]

evaluation : for public u, v in Fp prover can convince the verifier that the committed poly satisfies **f(u) = v** and **deg(f) ≤ d**

The trivial commitment scheme is not a polynomial commitment

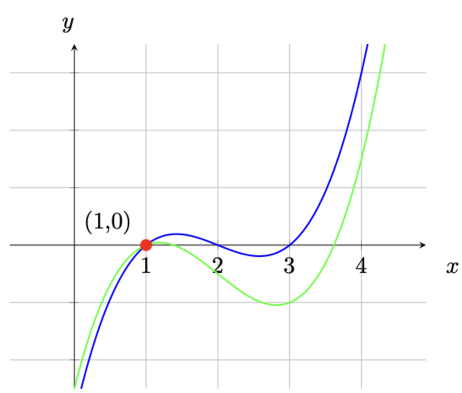
- commit($f = \sum_{i=0}^{d} a_i X^i, r$): output $\text{com}_f \leftarrow H((a_0, \dots, a_d), r)$
- eval: prover sends $\pi = ((a_0, \dots, a_d),\ r)$ to verifier;
  verifier accepts if $f(u) = v$ and $H((a_0, \dots, a_d), r) = \text{com}_f$

The problem: the proof π is not succinct.

Proof size and verification time are **linear** in d

Simple Protocol for the proof

- Downside of such a proving protocol is that one must do the number of checks proportionate to the number of elements.



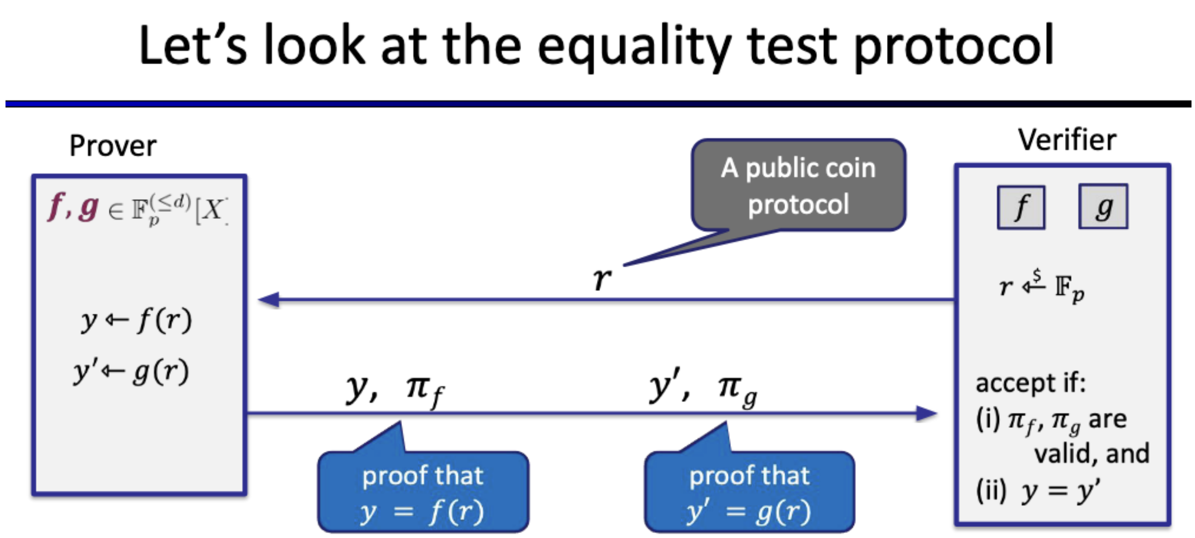
lets take blue curve as P(x) and the green one as Q(x). if we have two non-equal polynomials of degree at most **d**, they can intersect at no more than d points!

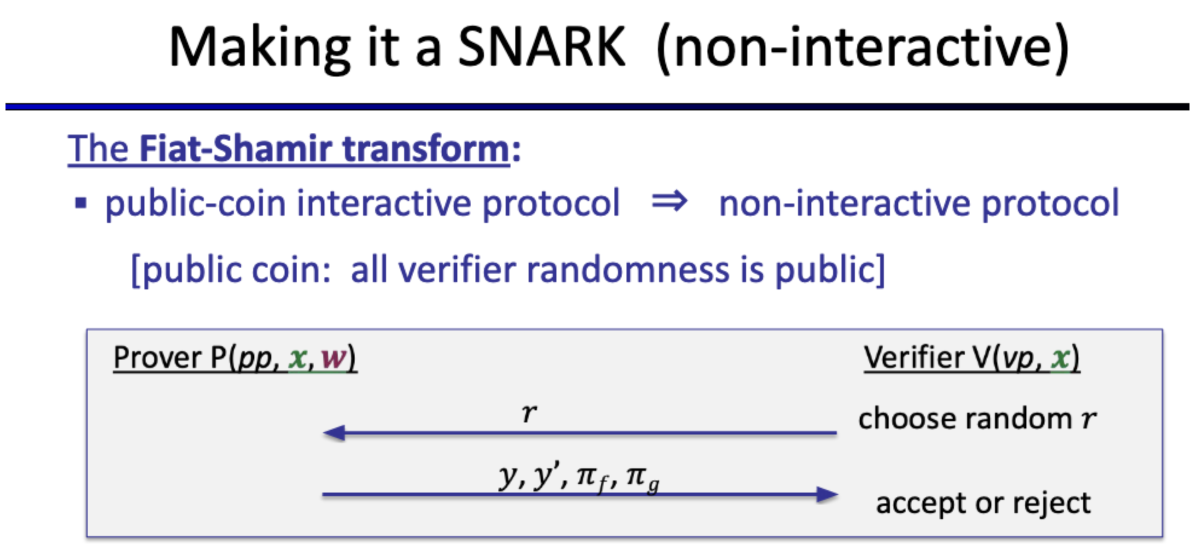
suppose that we have f(x) in Fp[x] (with degree d). we know that the domain of f(x) is p (cause we chose the polynomial from finite field P) so the probability to guess the right root is **d / p**

Suppose p = 2 ^ 256   and   d ≤ 2^40   ⟶   d/p is negligible

So we can say for random r if f(r) = 0 then the f is identically zero with high probability. Therefore, if we want to examine the equality of two polynomials P(x) and Q(x) :

P(x) = Q(x)   ⟶   P(x) - Q(x) = 0





- Fiat-Shamir is not secure all the time.

Homomorphic Enc / Modular arithmetic

That is exactly what homomorphic encryption is designed for. Namely, it allows to encrypt a value and be able to apply arithmetic operations on such encryption. There are multiple ways to achieve homomorphic properties of encryption, and we will briefly introduce a simple one. The general idea is that we choose a base natural number g (say 5) and to encrypt a value we exponentiate g to the power of that value.

- Verifier
  - samples a random value s, i.e., secret
  - calculates encryptions of s for all powers i in $0, 1, ..., d$, i.e.: $E(s^i) = g^{s^i}$
  - evaluates unencrypted *target polynomial* with s: $t(s)$
  - encrypted powers of s are provided to the prover: $E(s^0), E(s^1), ..., E(s^d)$
- Prover
  - calculates polynomial $h(x) = \frac{p(x)}{t(x)}$
  - using encrypted powers $g^{s^0}, g^{s^1}, \ldots, g^{s^d}$ and coefficients $c_0, c_1, \ldots, c_n$ evaluates $E\left(p(s)\right) = g^{p(s)} = \left(g^{s^d}\right)^{c_d} \cdots \left(g^{s^1}\right)^{c_1} \cdot \left(g^{s^0}\right)^{c_0}$ and similarly $E\left(h(s)\right) = g^{h(s)}$
  - the resulting g^p and g^h are provided to the verifier
- Verifier
  - The last step for the verifier is to checks that $p = t(s) \cdot h$ in encrypted space: $g^p = \left(g^h\right)^{t(s)} \quad \Rightarrow \quad g^p = g^{t(s) \cdot h}$

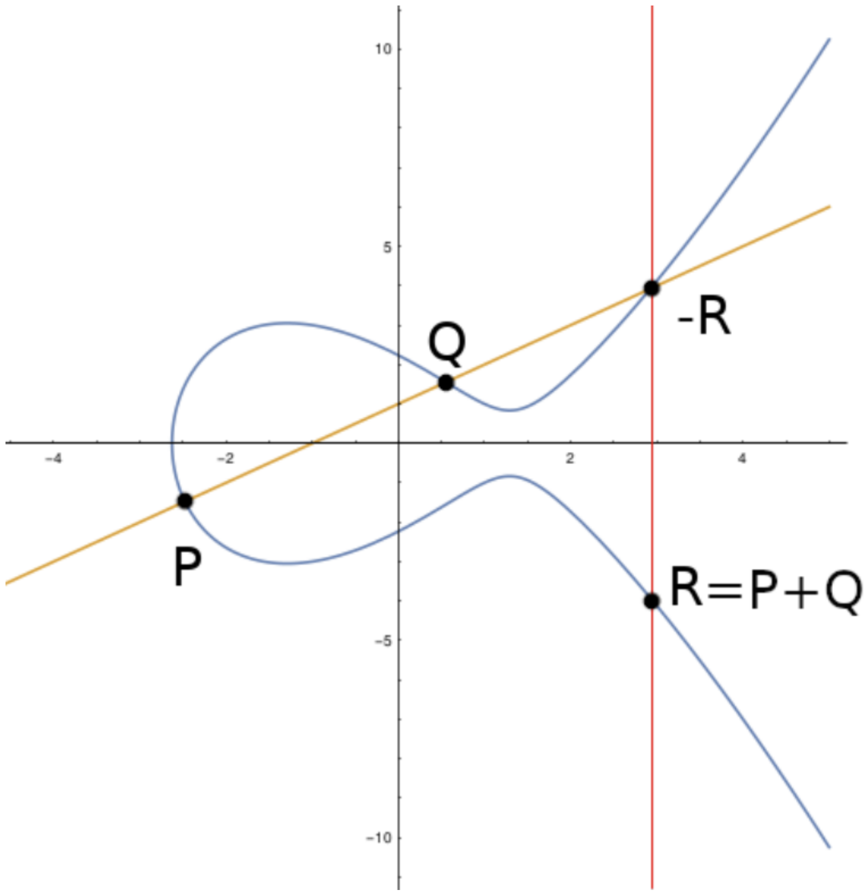
Knowledge of Exponent Assumption (KEA)

- Verifier chooses random s, α and provides evaluation for $x = s$ for power 1 and its “shift”: $(g^s, g^{\alpha \cdot s})$
- Prover applies the coefficient c: $\left((g^s)^c, (g^{\alpha \cdot s})^c\right) = (g^{c \cdot s}, g^{\alpha \cdot c \cdot s})$
- Verifier checks: $(g^{c \cdot s})^{\alpha} = g^{\alpha \cdot c \cdot s}$

Elliptic Curve

Y^2 = X^3 + ax + b

4a^3 + 27b^2 ≠ 0

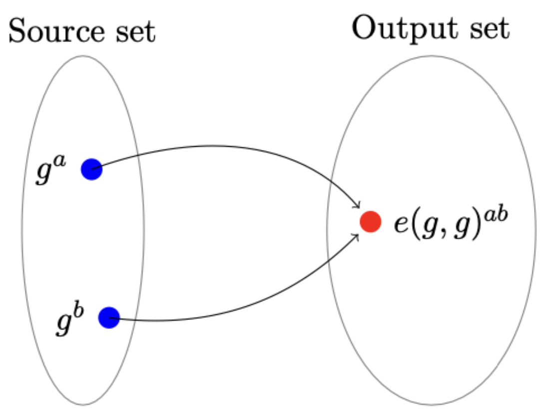


- How to calculate 2P ? I know how but how determine the slope the line ?
- Discrete Log in E.C
    - K * G = G + G + ... + G
    - Given G and K * G it is hard to calculate K
- Encryption in E.C.

- Alice chooses a random k.

- Bob chooses a random b and keep it secret
- Bob creates bG and send it as B to Alice.
- Alice send tuple (KG, M + KB)
- Bob will calculate M + KB - bKG = M

Pairing in Cryptography



G : source group (Points of an elliptic curve)

Gt: Target group (an element in a finite field)

$$e(g^a, g^b) = e(g^b, g^a) = e(g^{ab}, g^1) = e(g^1, g^{ab}) = e(g^1, g^a)^b = e(g^1, g^1)^{ab} = \dots$$

why these pairings are useful in cryptography ?

$$e(g^a, g^b) = e(g^b, g^a)$$

one for encrypting and the another one for decrypting can be used.

Consequence of Pairing

DDH in G is easy :

- input : g , g^x , g^y , g^z in G
- to test if :   e(g, g^z) ?= e(g^x, g^y)