ZK-SNARK 101

Youssef El Housni

ConsenSys, LIX and Inria, Paris, France

GT Oisillons 05/02/2021

What is a ZKP?

Zero-Knowledge Proof of Knowledge

Alice

I know the solution to this complex equation

Bob

No idea what the solution is but Alice must know it

"Prove it" ←

Challenge ←

Response →

Example: Sigma protocol

Zero-Knowledge for public keys (Sigma protocol)

Alice | **Bob**

I know x such that $g^x = y$

Alice: $r \xleftarrow{\text{random}} \mathbb{Z}_p$ — sends $A = g^r$ → Bob

Bob: $c \xleftarrow{\text{random}} \mathbb{Z}_p$ — sends c → Alice

Alice: $s = r + c \cdot x$ — sends s → Bob

Bob: $g^s \stackrel{?}{=} A \cdot y^c$

with $A \cdot y^c = g^r \cdot g^{x \cdot c}$

then $g^r \cdot g^{x \cdot c} = g^{r + x \cdot c}$

Example: non-interactive Sigma protocol

Non-Interactive Zero-Knowledge (NIZK)

Alice | **Bob**

I know x such that $g^x = y$

$$r \xleftarrow{\text{random}} \mathbb{Z}_p$$

$$A = g^r$$

$$c = H(A, y)$$

$$s = r + c \cdot x$$

$$\pi = (A, c, s) \longrightarrow$$

$$g^s \stackrel{?}{=} A \cdot y^c$$

$$c \stackrel{?}{=} H(A, y)$$

ZKP families

- *specific* statement vs *general* statement
- *interactive* vs *non-interactive* protocol
- *transparent* setup vs *trapdoored* setup vs *no* setup
- *Any* verifier vs *given* verifier
- prover complexity (Alice)
- verifier complexity (Bob)
- communication complexity (size of the proof and the setup)
- security assumptions, cryptographic primitive...
- ...

"I have a *sound*, *complete* and *zero-knowledge* proof that a statement is true". [GMR85]

Sound

False statement $\implies$ cheating prover cannot convince honest verifier.

Complete

True statement $\implies$ honest prover convinces honest verifier.

Zero-knowledge

True statement $\implies$ verifier learns nothing other than statement is true.

Zero-knowledge proof

ZK-SNARK: Zero-Knowledge Succinct Non-interactive ARgument of Knowledge

"I have a *computationally sound*, *complete*, *zero-knowledge*, *succinct*, *non-interactive* proof that a statement is true and that I know a related secret".

Succinct

Honestly-generated proof is very "short" and "easy" to verify.

Non-interactive

No interaction between the prover and verifier for proof generation and verification.

ARgument of Knowledge

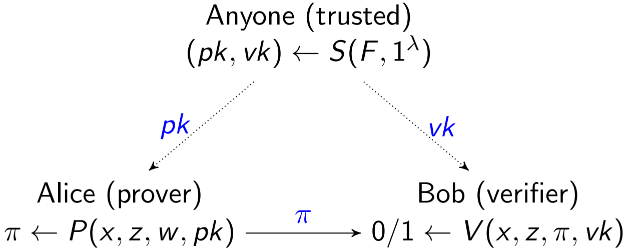
Honest verifier is convinced that a comptutationally bounded prover knows a secret information.

Zero-knowledge proof

Preprocessing ZK-SNARK of NP language

Let F be a public NP program, x and z be public inputs, and w be a private input such that $z := F(x, w)$.
A ZK-SNARK consists of algorithms S, P, V s.t. for a security parameter λ:

Trapdoored Setup:	(pk_τ, vk_τ)	$\leftarrow$	$S(F, \tau, 1^\lambda)$
Prove:	π_w	$\leftarrow$	$P(x, z, w, pk)$
Verify:	$0/1$	$\leftarrow$	$V(x, z, \pi, vk)$



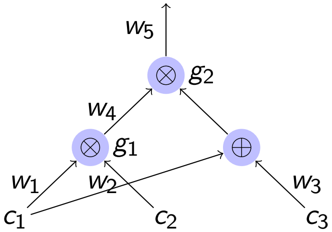
ZK-SNARKs in a nutshell

main ideas:

1. Reduce a "general statement" satisfiability to a polynomial equation satisfiability.
2. Use Schwartz-Zippel lemma to succinctly verify the polynomial equation with high probability.
3. Use homomorphic hiding cryptography to blindly verify the polynomial equation.
4. Use Fiat-Shamir transform to make the protocol non-interactive.

Arithmetization of the statement

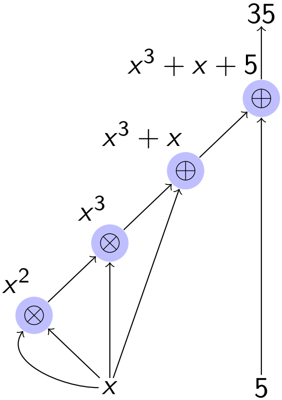
1. Statement $\rightarrow$
2. Arithmetic circuit $\rightarrow$
3. Rank 1 Constraint System (R1CS) $\rightarrow$
4. Quadratic Arithmetic Program (QAP) $\rightarrow$
5. zkSNARK Proof



Let's try an example!

Arithmetic circuit

$$x^3 + x + 5 = 35 \qquad (x = 3)$$



Let's try an example!
Rank 1 Constraint System (R1CS)

constraints:

$$o = l \cdot r$$

$$\begin{aligned} a &= x \cdot x \\ b &= a \cdot x \\ c &= (b + x) \cdot 1 \\ d &= (c + 5) \cdot 1 \end{aligned}$$

witness:

$$\begin{aligned} \vec{w} &= \begin{pmatrix} one & x & d & a & b & c \end{pmatrix} \\ &= \begin{pmatrix} 1 & 3 & 35 & 9 & 27 & 30 \end{pmatrix} \end{aligned}$$

Let's try an example!

Rank 1 Constraint System (R1CS)

constraints vectors:

$$\vec{o_1} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$
$$\vec{l_1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$
$$\vec{r_1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$
$$\dots$$
$$\vec{o_4} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$
$$\vec{l_1} = \begin{pmatrix} 5 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\vec{r_1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

verify:

$$\vec{o_i} \bullet \vec{w} = \vec{l_i} \bullet \vec{w} \cdot \vec{r_i} \bullet \vec{w}$$

Let's try an example!

Rank 1 Constraint System (R1CS)

$$L = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$O = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Lagrange polynomial interpolation:

$$L = \begin{pmatrix} L_1(1) & L_2(1) & L_3(1) & L_4(1) & L_5(1) & L_6(1) \\ L_1(2) & L_2(2) & L_3(1) & L_4(2) & L_5(2) & L_6(2) \\ L_1(3) & L_2(3) & L_3(3) & L_4(3) & L_5(3) & L_6(3) \\ L_1(4) & L_2(4) & L_3(4) & L_4(4) & L_5(4) & L_6(4) \end{pmatrix}$$

$$L_1(x) = -5 + 9.166x - 5x^2 + 0.833x^3$$

$$L_2(x) = 8 - 11.333x + 5x^2 - 0.666x^3$$

$$L_3(x) = 0$$

$$L_4(x) = -6 + 9.5x - 4x^2 + 0.5x^3$$

$$L_5(x) = 4 - 7x + 3.5x^2 - 0.5x^3$$

$$L_6(x) = -1 + 1.833x - x^2 + 0.166x^3$$

Let's try an example!

Quadratic Arithmetic Program (QAP)

$$R_i(x) = \begin{pmatrix} 3.0 & -5.166 & 2.5 & -0.333 \\ -2.0 & 5.166 & -2.5 & 0.333 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

$$O_i(x) = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ -1.0 & 1.833 & -1.0 & 0.166 \\ 4.0 & -4.333 & 1.5 & -0.166 \\ -6.0 & 9.5 & -4.0 & 0.5 \\ 4.0 & -7.0 & 3.5 & -0.5 \end{pmatrix}$$

Let's try an example!

Quadratic Arithmetic Program (QAP)

Now, the polynomials L_i, R_i, O_i should verify

$$\begin{pmatrix} L_1(x) \\ L_2(x) \\ L_3(x) \\ L_4(x) \\ L_5(x) \\ L_6(x) \end{pmatrix}^\intercal \bullet \begin{pmatrix} 1 \\ 3 \\ 35 \\ 9 \\ 27 \\ 30 \end{pmatrix} \cdot \begin{pmatrix} R_1(x) \\ R_2(x) \\ R_3(x) \\ R_4(x) \\ R_5(x) \\ R_6(x) \end{pmatrix}^\intercal \bullet \begin{pmatrix} 1 \\ 3 \\ 35 \\ 9 \\ 27 \\ 30 \end{pmatrix} = \begin{pmatrix} O_1(x) \\ O_2(x) \\ O_3(x) \\ O_4(x) \\ O_5(x) \\ O_6(x) \end{pmatrix}^\intercal \bullet \begin{pmatrix} 1 \\ 3 \\ 35 \\ 9 \\ 27 \\ 30 \end{pmatrix} \tag{1}$$

at $x = 1, 2, 3$ and 4

Let's try an example!
QAP

We rewrite this equation as

$$\underbrace{\sum_{i=1}^{6} L_i(x)w_i}_{\text{L(x)}} \cdot \underbrace{\sum_{i=1}^{6} R_i(x)w_i}_{\text{R(x)}} = \underbrace{\sum_{i=1}^{6} O_i(x)w_i}_{\text{O(x)}}, \qquad \text{at } x = 1, 2, 3, 4$$

which means $t(x) = \prod_{i=1}^{4}(x - i)$ divides $L(x) \cdot R(x) - O(x)$.

Let's try an example!

Schwartz-Zippel lemma

The QAP is the set of polynomials $L_i(x), R_i(x), O_i(x)$ and $t(x)$ in $\mathbb{F}[x]$.
given the witness w, Alice (the prover) computes $L(x), R(x), O(x)$ and $H(x)$ such as

$$L(x) \cdot R(x) - O(x) = t(x) \cdot H(x)$$

Bob (the verifier) needs to verify

$$\tau \xleftarrow{\text{random}} \mathbb{F}$$

$$L(\tau) \cdot R(\tau) - O(\tau) = t(\tau) \cdot H(\tau)$$

Let's try an example!

Schwartz-Zippel lemma

Schwartz–Zippel lemma

Any two distinct polynomials of degree d over a field $\mathbb{F}$ can agree on at most a $d/|\mathbb{F}|$ fraction of the points in $\mathbb{F}$.

Homomorphic hiding w.r.t. an arbitrary number of additions

$$L(\tau) = l_0 + l_1\tau + l_2\tau^2 + \cdots + l_d\tau^d$$
$$L(\tau)G = l_0G + l_1\tau G + l_2\tau^2 G + \cdots + l_d\tau^d G$$

for $G \in \mathbb{G}$ a group with hard discrete logarithm (e.g. elliptic curves).

but we need the homomorphic hiding w.r.t. **only one** multiplication as well (for $L \cdot R$ and $t \cdot H$).

- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- bilinear: $e(aG_1, bG_2) = e(G_1, bG_2)^a = e(aG_1, G_2)^b = e(aG_1, G_2)^{ab}$
- non-degenerate: $e(G_1, G_2) \neq 1_{\mathbb{G}_T}$

$$
\begin{aligned}
e(H(\tau)G_1, t(\tau)G_2) \cdot e(O(\tau)G_1, G_2) &= e(L(\tau)G_1, R(\tau)G_2) \\
e(G_1, G_2)^{H(\tau)t(\tau)} \cdot e(G_1, G_2)^{O(\tau)} &= e(G_1, G_2)^{L(\tau)R(\tau)} \\
C^{H(\tau)t(\tau)+O(\tau)} &= C^{L(\tau)R(\tau)}
\end{aligned}
$$

Let's try an example!

The q-power knowledge of exponent assumption [Groth10]

We need to force Alice to send the right (hidden) polynomials (instead of random points on the curve)

$$L(\tau) = l_0 + l_1\tau + l_2\tau^2 + \cdots + l_d\tau^d$$
$$P = L(\tau)G_1 = l_0 G_1 + l_1\tau G_1 + l_2\tau^2 G_1 + \cdots + l_d\tau^d G_1$$
$$Q = \alpha L(\tau)G_2 = l_0\alpha G_2 + l_1\tau\alpha G_2 + l_2\alpha\tau^2 G_2 + \cdots + l_d\alpha\tau^d G_2$$

Alice computes P and Q and Bob verifies that $e(P, \alpha G_2) = e(G_1, Q)$. The same goes for all polynomials.

Let's try an example!

Zero-knowledge

The same circuit $\implies$ the same QAP polynomials

$$L(x) \cdot R(x) - O(x) = t(x) \cdot H(x)$$

randomize L, R, O

$$\begin{aligned}
L_\gamma &= L(x) + \gamma_L \cdot t(x) \\
R_\gamma &= R(x) + \gamma_R \cdot t(x) \\
O_\gamma &= O(x) + \gamma_O \cdot t(x) \\
L_\gamma(x) \cdot R_\gamma(x) &- O_\gamma(x) = t(x) \cdot H_\gamma(x)
\end{aligned}$$

Let's try an example!

Pinocchio-ish Protocol

- Setup: sample $\tau, \alpha \xleftarrow{\text{random}} \mathbb{F}_r^*$ and computes $\tau^i G_1, \alpha\tau^i G_2, L_i(\tau)G_1, \alpha L_i(\tau)G_1$ (same for R_i, O_i, t)
- Prove (Alice): $L(\tau)G_1$ and $\alpha L(\tau)G_2$ (same for R, O, H)
- Verify (Bob):
  $e(H(\tau)G_1, t(\tau)G_2) \cdot e(O(\tau)G_1, G_2) \stackrel{?}{=} e(L(\tau)G_1, R(\tau)G_2)$ and
  $e(L(\tau)G_1, \alpha G_2) \stackrel{?}{=} e(G_1, \alpha L(\tau)G_2)$ (same for R, O, H)

Setup. The setup $\mathcal{S}$ receives an R1CS instance $\phi = (k, N, M, \mathbf{a}, \mathbf{b}, \mathbf{c})$ and then samples a proving key pk and a verification key vk as follows. First, $\mathcal{S}$ reduces the R1CS instance ϕ to a QAP instance $\Phi = (k, N, M, \mathbf{A}, \mathbf{B}, \mathbf{C}, D)$ by running the algorithm qapI. Then, $\mathcal{S}$ samples random elements $t, \alpha, \beta, \gamma, \delta$ in $\mathbb{F}$ (this is the randomness that must remain secret). After that, $\mathcal{S}$ evaluates the polynomials in $\mathbf{A}, \mathbf{B}, \mathbf{C}$ at the element t, and computes

$$\mathbf{K}^{\mathsf{vk}}(t) := \left(\frac{\beta \mathbf{A}_i(t) + \alpha \mathbf{B}_i(t) + \mathbf{C}_i(t)}{\gamma}\right)_{i=0,\dots,k}$$

$$\mathbf{K}^{\mathsf{pk}}(t) := \left(\frac{\beta \mathbf{A}_i(t) + \alpha \mathbf{B}_i(t) + \mathbf{C}_i(t)}{\delta}\right)_{i=k+1,\dots,N}$$

and

$$\mathbf{Z}(t) := \left(\frac{t^j Z_D(t)}{\delta}\right)_{j=0,\dots,M-2} .$$

Finally, the setup algorithm computes encodings of these elements and outputs pk and vk defined as follows:

$$\mathsf{pk} := \left([\alpha]_1, \begin{matrix}[\beta]_1, [\delta]_1 \\ [\beta]_2, [\delta]_2\end{matrix}, [\mathbf{A}(t)]_1, \begin{matrix}[\mathbf{B}(t)]_1 \\ [\mathbf{B}(t)]_2\end{matrix}, \begin{matrix}[\mathbf{K}^{\mathsf{pk}}(t)]_1 \\ [\mathbf{Z}(t)]_1\end{matrix}\right)$$

$$\mathsf{vk} := \left(e(\alpha, \beta), [\gamma]_2, [\delta]_2, [\mathbf{K}^{\mathsf{vk}}(t)]_1\right) .$$

Prover. The prover $\mathcal{P}$ receives a proving key pk, input x in $\mathbb{F}^k$, and witness w in $\mathbb{F}^{N-k}$, and then samples a proof π as follows. First, $\mathcal{P}$ extends the x-witness w for the R1CS instance ϕ to a x-witness (w, h) for the QAP instance Φ by running the algorithm qapW. Then, $\mathcal{P}$ samples random elements r, s in $\mathbb{F}$ (this is the randomness that imbues the proof with zero knowledge). Next, letting $z := 1\|x\|w$, $\mathcal{P}$ computes three encodings obtained as follows

$$[A_r]_1 := [\alpha]_1 + \sum_{i=0}^{N} z_i [\mathbf{A}_i(t)]_1 + r[\delta]_1 ,$$

$$[B_s]_1 := [\beta]_1 + \sum_{i=0}^{N} z_i [\mathbf{B}_i(t)]_1 + s[\delta]_1$$

$$[B_s]_2 := [\beta]_2 + \sum_{i=0}^{N} z_i [\mathbf{B}_i(t)]_2 + s[\delta]_2 .$$

Then $\mathcal{P}$ uses these two compute a fourth encoding:

$$[K_{r,s}]_1 := s[A_r]_1 + r[B_s]_1 - rs[\delta]_1 + \sum_{i=k+1}^{N} z_i [\mathbf{K}_i^{\mathsf{pk}}(t)]_1 + \sum_{j=0}^{M-2} h_j [\mathbf{Z}_j(t)]_1 .$$

Prover:
4 MSM (80%)
7 FFT (20%)

The output proof is $\pi := ([A_r]_1, [B_s]_2, [K_{r,s}]_1)$.

Verifier. The verifier $\mathcal{V}$ receives a verification key vk, input x in $\mathbb{F}^k$, and proof π, and, letting $x_0 := 1$, checks that the following holds:

$$e([A_r]_1, [B_s]_2) = e(\alpha, \beta) + e\left(\sum_{i=0}^{k} x_i [\mathbf{K}_i^{\mathsf{vk}}(t)]_1, [\gamma]_2\right) + e([K_{r,s}]_1, [\delta]_2) .$$

Verifier:
1 MSM
4 pairings