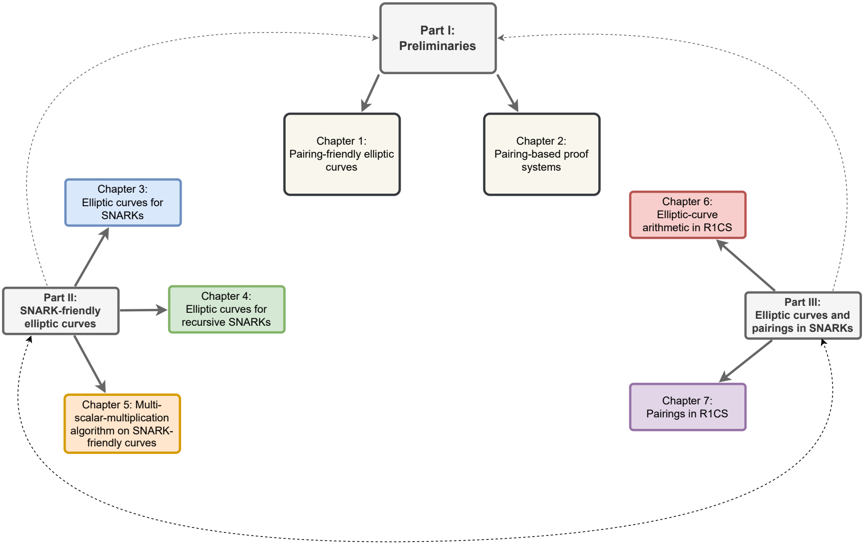
ZK-SNARKs & Elliptic curves

Youssef El Housni

ConsenSys, LIX and Inria, Paris, France

Séminaire — Rennes le 16/09/2022

PhD Thesis

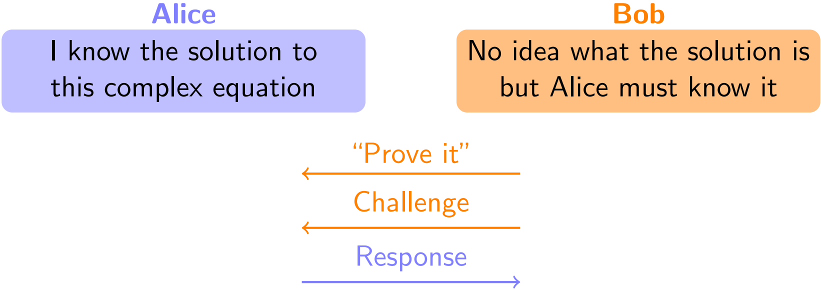

Part I:
Preliminaries
Chapter 1:
Pairing-friendly elliptic curves
Chapter 2:
Pairing-based proof systems
Chapter 3:
Elliptic curves for SNARKs
Part II:
SNARK-friendly elliptic curves
Chapter 4:
Elliptic curves for recursive SNARKs
Chapter 5: Multi-scalar-multiplication algorithm on SNARK-friendly curves
Chapter 6:
Elliptic-curve arithmetic in R1CS
Part III:
Elliptic curves and pairings in SNARKs
Chapter 7:
Pairings in R1CS

Overview

1. Preliminaries
   - Zero-knowledge proof (ZKP)
   - ZK-SNARK
   - proof composition

2. Choice of elliptic curves
   - SNARK curves
   - Implementations

Zero-Knowledge Proofs



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

Zero-Knowledge for public keys: Sigma protocol

Alice **Bob**

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

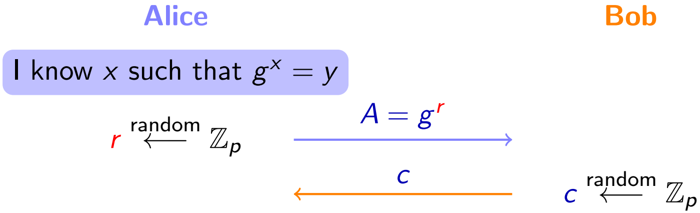
Zero-Knowledge for public keys: Sigma protocol

Alice **Bob**

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

$r \xleftarrow{\text{random}} \mathbb{Z}_p \quad \xrightarrow{A = g^r}$

Zero-Knowledge for public keys: Sigma protocol



Alice

Bob

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

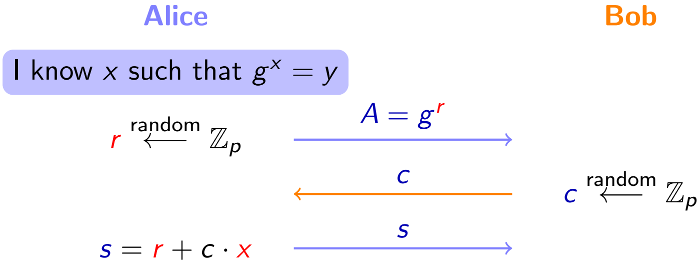
$r \xleftarrow{\text{random}} \mathbb{Z}_p$

$\xrightarrow{A = g^r}$

$\xleftarrow{c}$

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

Zero-Knowledge for public keys: Sigma protocol



Alice

Bob

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

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

$A = g^r$

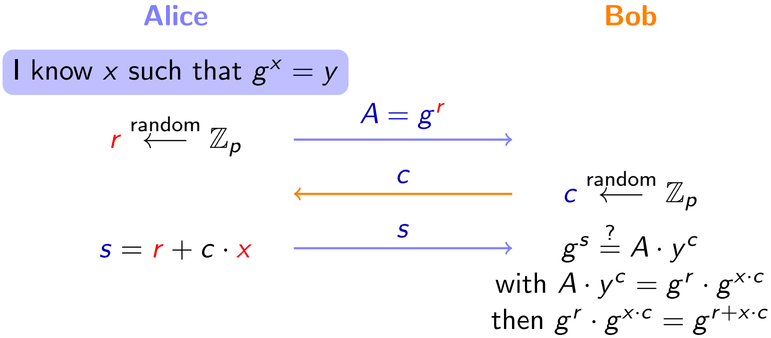
c

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

s

$s = r + c \cdot x$

Zero-Knowledge for public keys: Sigma protocol



Alice

Bob

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

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

$\xrightarrow{A = g^r}$

$\xleftarrow{c}$

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

$s = r + c \cdot x$

$\xrightarrow{s}$

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

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

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

Non-Interactive Zero-Knowledge (NIZK) Sigma protocol

Alice

Bob

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

$$r \overset{\text{random}}{\longleftarrow} \mathbb{Z}_p$$

$$A = g^r$$

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

$$s = r + c \cdot x \quad \xrightarrow{\pi = (A, c, s)} \quad g^s \overset{?}{=} A \cdot y^c$$

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

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...
- ...

Blockchains and ZKP

A blockchain is a public peer-to-peer *decentralized*, *transparent, immutable, paying* ledger.

- *Transparent*: everything is visible to everyone
- *Immutable*: nothing can be removed once written
- *Paying*: everyone should pay a fee to use

Transparent $\xrightarrow[\text{Problem}]{}$ confidentiality $\xrightarrow[\text{Solution}]{}$ ZKP

setup, prover?, verifier?

Immutable $\xrightarrow[\text{Problem}]{}$ scalability $\xrightarrow[\text{Solution}]{}$ ZKP

Communication complexity

Paying $\xrightarrow[\text{Problem}]{}$ cost $\xrightarrow[\text{Solution}]{}$ ZKP

Verifier complexity, *prover?*

ZKP literature landmarks

- First ZKP paper [GMR85]
- Non-Interactive ZKP [BFM88]
- Succinct ZKP [K92]
- Succinct Non-Interactive ZKP [M94]
- Succinct NIZK without the PCP Theorem [Groth10]
- "SNARK" terminology and characterization of existence [BCCT11]
- Succinct NIZK without PCP Theorem and Quasi-linear prover time [GGPR13]
- Succinct NIZK without with constant-size proof and constant-time verifier (Groth16)
- First succinct NIZK with universal and updatable setup [Sonic19]
- Active research and implementation on SNARK with universal and updatable setup [PLONK19]
- ...

Zero-knowledge proof

What is a zero-knowledge proof?

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

Sound

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

Complete

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

Zero-knowledge

True statement $\Longrightarrow$ 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 computationally bounded prover knows a secret information.

ZK-SNARK

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:

$$\text{Setup:} \quad (pk, vk) \quad \leftarrow \quad S(F, \tau, 1^\lambda)$$

ZK-SNARK

Preprocessing ZK-SNARK of NP language

Let F be a public $\texttt{NP}$ 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:

Setup:	(pk, vk)	$\leftarrow$	$S(F, \tau, 1^\lambda)$
Prove:	π	$\leftarrow$	$P(x, z, w, pk)$

ZK-SNARK

Preprocessing ZK-SNARK of NP language

Let F be a public $\texttt{NP}$ 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:

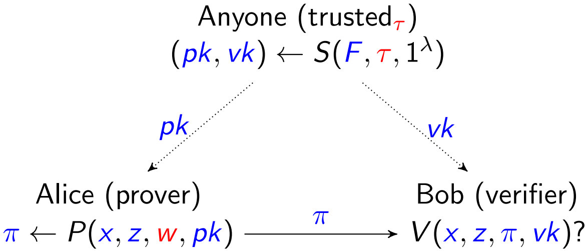
Setup:	(pk, vk)	$\leftarrow$	$S(F, \tau, 1^\lambda)$
Prove:	π	$\leftarrow$	$P(x, z, w, pk)$
Verify:	$\texttt{false/true}$	$\leftarrow$	$V(x, z, \pi, vk)$

ZK-SNARK

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:

Setup:	(pk, vk)	$\leftarrow$	$S(F, \tau, 1^\lambda)$
Prove:	π	$\leftarrow$	$P(x, z, w, pk)$
Verify:	`false/true`	$\leftarrow$	$V(x, z, \pi, vk)$



Anyone (trusted$_\tau$)

$(pk, vk) \leftarrow S(F, \tau, 1^\lambda)$

pk

vk

Alice (prover)

$\pi \leftarrow P(x, z, w, pk) \xrightarrow{\pi}$

Bob (verifier)

$V(x, z, \pi, vk)$?

ZK-SNARK

Succinctness: An honestly-generated proof is very "short" and "easy" to verify.

Definition [BCTV14b]

A succinct proof π has size $O_\lambda(1)$ and can be verified in time $O_\lambda(|F| + |x| + |z|)$, where $O_\lambda(.)$ is some polynomial in the security parameter λ.

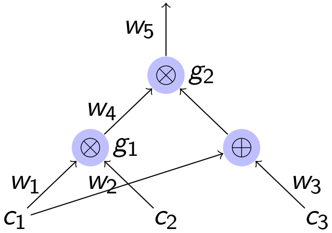
ZK-SNARKs in a nutshell

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

Statement → Arithmetic circuit → Rank 1 Constraint System (R1CS) → Quadratic Arithmetic Program (QAP) → zkSNARK Proof



$$U(x)V(x) - W(x) = H(x)T(x) \qquad (QAP)$$

$$U(\tau)V(\tau) - W(\tau) = H(\tau)T(\tau)$$

$$\mathtt{HH}(U(\tau)V(\tau) - W(\tau) = H(\tau)T(\tau))$$

Blind evaluation of QAP

Instead of verifying the QAP on the whole domain $\mathbb{F}$ → verify it in a single random point $\tau \in \mathbb{F}$.

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 $\mathbb{F}$.

Blind evaluation of QAP

Let's take the example of polynomial U:

- Alice can send U to Bob and he computes $U(\tau) \rightarrow$ This breaks the zero-knowledge.
- Bob can send τ to Alice and she computes $U(\tau) \rightarrow$ This breaks the soundness.

We need a homomorphic hiding cryptographic primitive to evaluate $U(x)$ at τ without Bob learning U nor Alice learning τ.

Blind evaluation of QAP

$$U(\tau) = u_0 + u_1\tau + u_2\tau^2 + \cdots + u_d\tau^d$$
$$HH(U(\tau)) = u_0 + u_1 HH(\tau) + u_2 HH(\tau^2) + \cdots + u_d HH(\tau^d)$$

Homomorphic hiding function w.r.t.:

- d **additions** (arbitrary d)
- **1 multiplication** (for $U \cdot V$ and $H \cdot T$).

Blind evaluation of QAP

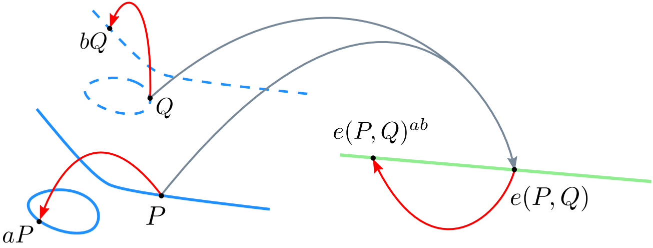
bilinear pairings

A non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

non-degenerate: $\forall P \in \mathbb{G}_1,\ P \neq \mathcal{O},\ \exists Q \in \mathbb{G}_2, e(P,Q) \neq 1_{\mathbb{G}_T}$

$\forall Q \in \mathbb{G}_2,\ Q \neq \mathcal{O},\ \exists P \in \mathbb{G}_1, e(P,Q) \neq 1_{\mathbb{G}_T}$

bilinear: $e([a]P,[b]Q) = e(P,[b]Q)^a = e([a]P,Q)^b = e(P,Q)^{ab}$



1

[1] Courtesy of Diego F. Aranha.

Blind evaluation of QAP

Blind evaluation can be achieved with *black-box* pairings:

$$\begin{aligned} e(H(\tau)G_1, T(\tau)G_2) \cdot e(W(\tau)G_1, G_2) &= e(U(\tau)G_1, V(\tau)G_2) \\ e(G_1, G_2)^{H(\tau)T(\tau)} \cdot e(G_1, G_2)^{W(\tau)} &= e(G_1, G_2)^{U(\tau)V(\tau)} \\ C_{te}^{H(\tau)T(\tau)+W(\tau)} &= C_{te}^{U(\tau)V(\tau)} \end{aligned}$$

Notations

Pairing-based zkSNARK

- $E\colon y^2 = x^3 + ax + b$ elliptic curve defined over $\mathbb{F}_q$, q a prime power.
- r prime divisor of $\#E(\mathbb{F}_q) = q + 1 - t$, t Frobenius trace.
- $-D$ CM discriminant, $4q = t^2 + Dy^2$ for some integer y.
- d degree of twist.
- k embedding degree, smallest integer $k \in \mathbb{N}^*$ s.t. $r \mid q^k - 1$.
- $\mathbb{G}_1 \subset E(\mathbb{F}_q)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$ two groups of order r.
- $\mathbb{G}_T \subset \mathbb{F}^*_{q^k}$ group of r-th roots of unity.
- pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Proof composition

A proof

Example: Groth16 [Gro16]

Given an instance $\Phi = (a_0, \ldots, a_\ell) \in \mathbb{F}_r^\ell$ of a public NP program F

- $(pk, vk) \leftarrow S(F, \tau, 1^\lambda)$ where

$$vk = (vk_{\alpha,\beta}, \{vk_{\pi_i}\}_{i=0}^{\ell}, vk_\gamma, vk_\delta) \in \mathbb{G}_T \times \mathbb{G}_1^{\ell+1} \times \mathbb{G}_2 \times \mathbb{G}_2$$

- $\pi \leftarrow P(\Phi, w, pk)$ where

$$\pi = (A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \qquad (O_\lambda(1))$$

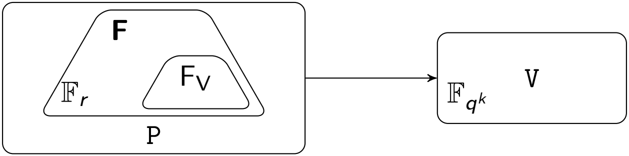
- $0/1 \leftarrow V(\Phi, \pi, vk)$ where V is

$$e(A, B) = vk_{\alpha,\beta} \cdot e(vk_x, vk_\gamma) \cdot e(C, vk_\delta) \qquad (O_\lambda(|\Phi|)) \qquad (1)$$

and $vk_x = \sum_{i=0}^{\ell} [a_i] vk_{\pi_i}$ depends only on the instance Φ and $vk_{\alpha,\beta} = e(vk_\alpha, vk_\beta)$ can be computed in the trusted setup for $(vk_\alpha, vk_\beta) \in \mathbb{G}_1 \times \mathbb{G}_2$.

Recursive ZK-SNARKs

An arithmetic mismatch



F any program is expressed in $\mathbb{F}_r$

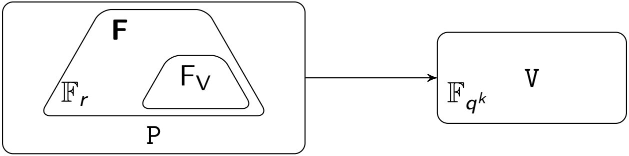
P proving is performed over $\mathbb{G}_1$ (and $\mathbb{G}_2$) (of order r)

V verification (eq. 1) is done in $\mathbb{F}^*_{q^k}$

F_V program of V is natively expressed in $\mathbb{F}^*_{q^k}$ not $\mathbb{F}_r$

Recursive ZK-SNARKs

An arithmetic mismatch



F any program is expressed in $\mathbb{F}_r$

P proving is performed over $\mathbb{G}_1$ (and $\mathbb{G}_2$) (of order r)

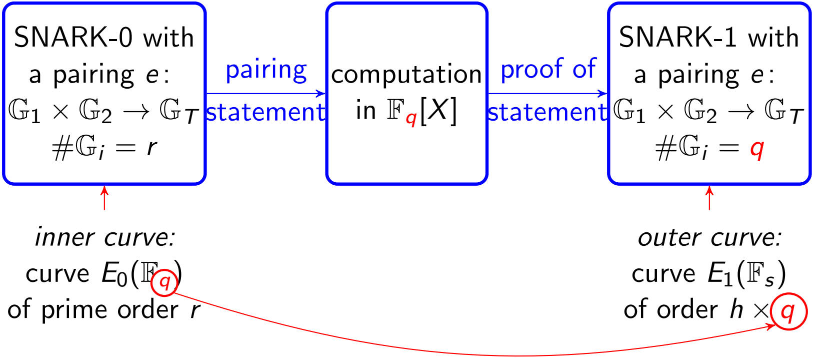
V verification (eq. 1) is done in $\mathbb{F}^*_{q^k}$

F_V program of V is natively expressed in $\mathbb{F}^*_{q^k}$ not $\mathbb{F}_r$

- 1st attempt: choose a curve for which $q = r$ (impossible)
- 2nd attempt: simulate $\mathbb{F}_q$ operations via $\mathbb{F}_r$ operations ($\times \log q$ blowup)
- 3rd attempt: use a cycle/chain of pairing-friendly elliptic curves [CFH+15, BCTV14a, BCG+20]

Recursive ZK-SNARKs

A proof of a proof



SNARK-0 with a pairing e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ $\#\mathbb{G}_i = r$

pairing statement

computation in $\mathbb{F}_q[X]$

proof of statement

SNARK-1 with a pairing e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ $\#\mathbb{G}_i = q$

inner curve: curve $E_0(\mathbb{F}_q)$ of prime order r

outer curve: curve $E_1(\mathbb{F}_s)$ of order $h \times q$

Given q, search for a pairing-friendly curve E_1 of order $h \cdot q$ over a field $\mathbb{F}_s$

Proof composition

cycles and chains of pairing-friendly elliptic curves

Definition

An *m*-chain of elliptic curves is a list of distinct curves

$$E_1/\mathbb{F}_{q_1}, \ldots, E_m/\mathbb{F}_{q_m}$$

where $q_1, \ldots, q_m$ are large primes and

$$\#E_2(\mathbb{F}_{q_2}) = q_1, \ldots, \ \#E_i(\mathbb{F}_{q_i}) = q_{i-1}, \ldots, \ \#E_m(\mathbb{F}_{q_m}) = q_{m-1} \ . \tag{2}$$

Definition

An *m*-cycle of elliptic curves is an *m*-chain, with

$$\#E_1(\mathbb{F}_{q_1}) = q_m \ . \tag{3}$$

Choice of elliptic curves

ZK-curves

- SNARK
  - $E/\mathbb{F}_q$ BN, BLS12, BW12?, KSS16? ... [FST10]
    - pairing-friendly
    - $r-1$ highly 2-adic (efficient FFT)
- Recursive SNARK (2-cycle)
  - $E_1/\mathbb{F}_{q_1}$ and $E_2/\mathbb{F}_{q_2}$ MNT4/MNT6 [FST10, Sec.5], ? [CCW19]
    - both pairing-friendly
    - $r_2 = q_1$ and $r_1 = q_2$
    - $r_{\{1,2\}} - 1$ highly 2-adic (efficient FFT)
    - $q_{\{1,2\}} - 1$ highly 2-adic (efficient FFT)
- Recursive SNARK (2-chain)
  - $E_1/\mathbb{F}_{q_1}$ BLS12 ($seed \equiv 1 \bmod 3 \cdot 2^{large}$) [BCG$^+$20], ?
    - pairing-friendly
    - $r_1 - 1$ highly 2-adic
    - $q_1 - 1$ highly 2-adic
  - $E_2/\mathbb{F}_{q_2}$ Cocks–Pinch algorithm
    - pairing-friendly
    - $r_2 = q_1$

Choice of elliptic curves

Curve $E_2/\mathbb{F}_{q_2}$

- q is a prime or a prime power
- t is relatively prime to q
- ~~r is prime~~
- ~~r divides $q+1-t$~~
- ~~r divides q^k-1 (smallest $k \in \mathbb{N}^*$)~~

r is a **fixed** chosen prime
that divides $q+1-t$
and q^k-1 (smallest $k \in \mathbb{N}^*$)

- $4q - t^2 = Dy^2$ (for $D < 10^{12}$) and some integer y

Algorithm 1: Cocks–Pinch method

```
1 Fix $k$ and $D$ and choose a prime $r$ s.t. $k | r-1$ and $(\frac{-D}{r}) = 1$;
2 Compute $t = 1 + x^{(r-1)/k}$ for $x$ a generator of $(\mathbb{Z}/r\mathbb{Z})^\times$;
3 Compute $y = (t-2)/\sqrt{-D} \bmod r$;
4 Lift $t$ and $y$ in $\mathbb{Z}$;
5 Compute $q = (t^2 + Dy^2)/4$ (in $\mathbb{Q}$);
6 back to 1 if $q$ is not a prime integer;
```

2-chains

Limitations and improvements

- $\rho = \log_2 q / \log_2 r \approx 2$ (because $q = f(t^2, y^2)$ and $t, y \overset{\$}{\leftarrow} \bmod r$).
- The curve parameters (q, r, t) are not expressed as polynomials.

```
Algorithm 2: Brezing–Weng method
Fix $k$ and $D$ and choose an irreducible polynomial $r(x) \in \mathbb{Z}[x]$ with
  positive leading coefficient [1] s.t. $\sqrt{-D}$ and the primitive $k$-th root of
  unity $\zeta_k$ are in $K = \mathbb{Q}[x]/r(x)$;
Choose $t(x) \in \mathbb{Q}[x]$ be a polynomial representing $\zeta_k + 1$ in $K$;
Set $y(x) \in \mathbb{Q}[x]$ be a polynomial mapping to $(\zeta_k - 1)/\sqrt{-D}$ in $K$;
Compute $q(x) = (t^2(x) + Dy^2(x))/4$ in $\mathbb{Q}[x]$;
```

- $\rho = 2 \max(\deg t(x), \deg y(x)) / \deg r(x) < 2$
- $r(x), q(x), t(x)$ but does $\exists\, x_0 \in \mathbb{Z}^*, r(x_0) = r_{\text{fixed}}$ and $q(x_0)$ is prime ?

[1] conditions to satisfy Bunyakovsky conjecture which states that such a polynomial produces infinitely many primes for infinitely many integers.

2-chains

Suggested construction: combines CP and BW

1. Cocks–Pinch method
   - $k = 6$ and $-D = -3 \implies$ 128-bit security, $\mathbb{G}_2$ coordinates in $\mathbb{F}_q$, GLV multiplication over $\mathbb{G}_1$ and $\mathbb{G}_2$
   - restrict search to $\text{size}(q) \leq 768$ bits $\implies$ smallest machine-word size
2. Brezing–Weng method
   - choose $r(x) = q_{\text{BLS}\,12-377}(x)$
   - $q(x) = (t^2(x) + 3y^2(x))/4$ factors $\implies$ $q(x_0)$ cannot be prime
   - lift $t = r \times h_t + t(x_0)$ and $y = r \times h_y + y(x_0)$ [FK19, GMT20]

2-chains [CANS2020]

The suggested curve: BW6-761

$E : y^2 = x^3 - 1$ over $\mathbb{F}_q$ of 761-bit with seed $x_0 = \mathtt{0x8508c00000000}$ and polynomials:

Our curve, $k = 6$, $D = 3$, $r = q_{\mathrm{BLS\,12-377}}$
$r(x) = (x^6 - 2x^5 + 2x^3 + x + 1)/3 = q_{\mathrm{BLS\,12-377}}(x)$ $t(x) = x^5 - 3x^4 + 3x^3 - x + 3 + h_t r(x)$ $y(x) = (x^5 - 3x^4 + 3x^3 - x + 3)/3 + h_y r(x)$ $q(x) = (t^2 + 3y^2)/4$ $q_{h_t=13,h_y=9}(x) = (103x^{12} - 379x^{11} + 250x^{10} + 691x^9 - 911x^8$ $-79x^7 + 623x^6 - 640x^5 + 274x^4 + 763x^3 + 73x^2 + 254x + 229)/9$

Inner curves [EC2022]

SNARK-0

Groth16 SNARK

- 128-bit security
- pairing-friendly
- efficient $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and pairing
- $p-1 \equiv r-1 \equiv 0 \mod 2^L$ for large input $L \in \mathbb{N}^*$ (FFTs)

→ BLS ($k=12$) family of roughly 384 bits with seed $x \equiv 1 \mod 3 \cdot 2^L$

Universal SNARK

- 128-bit security
- pairing-friendly
- efficient $\mathbb{G}_1$, $\cancel{\mathbb{G}_2, \mathbb{G}_T}$ and pairing
- $p-1 \equiv r-1 \equiv 0 \mod 2^L$ for large $L \in \mathbb{N}^*$ (FFTs)

→ BLS ($k=24$) family of roughly 320 bits with seed $x \equiv 1 \mod 3 \cdot 2^L$

Outer curves [EC2022]

SNARK-1

Groth16 SNARK

- 128-bit security
- pairing-friendly
- efficient $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and pairing
- $r' = p \ (r' - 1 \equiv 0 \mod 2^L)$

$\rightarrow$ BW ($k = 6$) family of roughly 768 bits with $(t \mod x) \mod r \equiv 0$ or 3

Universal SNARK

- 128-bit security
- pairing-friendly
- efficient $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and pairing
- $r' = p \ (r' - 1 \equiv 0 \mod 2^L)$

$\rightarrow$ BW ($k = 6$) family of roughly 704 bits with $(t \mod x) \mod r \equiv 0$ or 3

$\rightarrow$ CP ($k = 8$) family of roughly 640 bits

$\rightarrow$ CP ($k = 12$) family of roughly 640 bits

All $\mathbb{G}_i$ formulae and pairings are given in terms of x and some $h_t, h_y \in \mathbb{N}$.

Implementation and benchmark

Short-list of curves

We short list few 2-chains of the proposed families that have some additional nice engineering properties

- Groth16: BLS12-377 and BW6-761
- Universal: BLS24-315 and BW6-633 (or BW6-672)

Table: Cost of $\mathtt{S}$, $\mathtt{P}$ and $\mathtt{V}$ algorithms for Groth16 and Universal. n =number of multiplication gates, a =number of addition gates and ℓ =number of public inputs. $\mathtt{M}_{\mathbb{G}}$ =multiplication in $\mathbb{G}$ and $\mathtt{P}$=pairing.

	$\mathtt{S}$	$\mathtt{P}$	$\mathtt{V}$
Groth16	$3n\ \mathtt{M}_{\mathbb{G}_1},\ n\ \mathtt{M}_{\mathbb{G}_2}$	$(4n-\ell)\ \mathtt{M}_{\mathbb{G}_1},\ n\ \mathtt{M}_{\mathbb{G}_2}$	$3\ \mathtt{P},\ \ell\ \mathtt{M}_{\mathbb{G}_1}$
Universal	$d_{\geq n+a}\ \mathtt{M}_{\mathbb{G}_1},\ 1\ \mathtt{M}_{\mathbb{G}_2}$	$9(n+a)\ \mathtt{M}_{\mathbb{G}_1}$	$2\ \mathtt{P},\ 18\ \mathtt{M}_{\mathbb{G}_1}$

Implementation and benchmark

https://github.com/ConsenSys/gnark (Go)

F_V: program that checks V (eq. 1) ($\ell = 1$, $\cancel{n = 90000}$ $n = 19378$)
[Housni22] "Pairings in R1CS"

Table: Groth16 (ms)

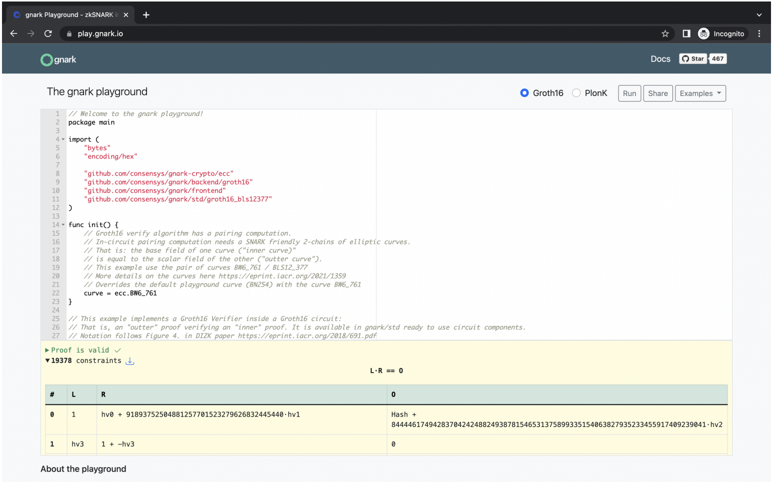
	S	P	V
BLS12-377	387	34	1
BLS24-315	501	54	4
BW6-761	1226	114	9
BW6-633	710	69	6
BW6-672	840	74	7

Table: Universal (ms)

	S	P	V
BLS12-377	87	215	4
BLS24-315	76	173	1
BW6-761	294	634	9
BW6-633	170	428	6
BW6-672	190	459	7

Play with gnark!

Write SNARK programs at `https://play.gnark.io/`
Example: Proof of Groth16 `V` program (eq. 1)



The gnark playground

● Groth16 ○ PlonK Run Share Examples

```go
// Welcome to the gnark playground!
package main

import (
    "bytes"
    "encoding/hex"

    "github.com/consensys/gnark-crypto/ecc"
    "github.com/consensys/gnark/backend/groth16"
    "github.com/consensys/gnark/frontend"
    "github.com/consensys/gnark/std/groth16_bls12377"
)

func init() {
    // Groth16 verify algorithm has a pairing computation.
    // In-circuit pairing computation needs a SNARK friendly 2-chains of elliptic curves.
    // That is: the base field of one curve ("inner curve)"
    // is equal to the scalar field of the other ("outter curve").
    // This example use the pair of curves BW6_761 / BLS12_377
    // More details on the curves here https://eprint.iacr.org/2021/1359
    // Overrides the default playground curve (BN254) with the curve BW6_761
    curve = ecc.BW6_761
}

// This example implements a Groth16 Verifier inside a Groth16 circuit:
// That is, an "outter" proof verifying an "inner" proof. It is available in gnark/std ready to use circuit components.
// Notation follows Figure 4. in DIZK paper https://eprint.iacr.org/2018/691.pdf
```

```
▶ Proof is valid ✓
▼19378 constraints
```

L·R == O

#	L	R	O
0	1	hv0 + 918937525048812577015232796268324454 40·hv1	Hash + 84444617494283704242488249387815465313758993351540638279352334559174092390 41·hv2
1	hv3	1 + -hv3	0

About the playground

Conclusion

papers Optimized and secure pairing-friendly elliptic curvE2 suitable for one layer proof composition (**CANS 2022**)

Families of SNARK-friendly 2-chains of elliptic curves (**EUROCRYPT 2022**)

A survey of elliptic curves for proof systems (**DCC 2022**)

implementations github/ConsenSys/gnark-crypto (Go)

gitlab/inria/snark-2-chains (SageMath/MAGMA)

other papers Co-factor clearing and subgroup membership on pairing-friendly elliptic curves (**AFRICACRYPT 2022**)

Pairings in Rank-1 Constraint System (**In submission**)

THANK YOU!

References I

Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu.
Zexe: Enabling decentralized private computation.
In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1059–1076, Los Alamitos, CA, USA, may 2020. IEEE Computer Society.

Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza.
Scalable zero knowledge via cycles of elliptic curves.
In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014.

References II

Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza.
Succinct non-interactive zero knowledge for a von neumann architecture.
In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 781–796. USENIX Association, August 2014.

Alessandro Chiesa, Lynn Chua, and Matthew Weidner.
On cycles of pairing-friendly elliptic curves.
SIAM Journal on Applied Algebra and Geometry, 3(2):175–192, 2019.

References III

Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur.
Geppetto: Versatile verifiable computation.
In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 253–270. IEEE Computer Society, 2015.
ePrint 2014/976.

Georgios Fotiadis and Elisavet Konstantinou.
TNFS resistant families of pairing-friendly elliptic curves.
Theoretical Computer Science, 800:73–89, 31 December 2019.

David Freeman, Michael Scott, and Edlyn Teske.
A taxonomy of pairing-friendly elliptic curves.
Journal of Cryptology, 23(2):224–280, April 2010.

References IV

Aurore Guillevic, Simon Masson, and Emmanuel Thomé.
Cocks–Pinch curves of embedding degrees five to eight and optimal ate pairing computation.
Des. Codes Cryptogr., 88:1047–1081, March 2020.

Jens Groth.
On the size of pairing-based non-interactive arguments.
In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.