On choosing suitable elliptic curves for ZK-SNARK

Youssef El Housni

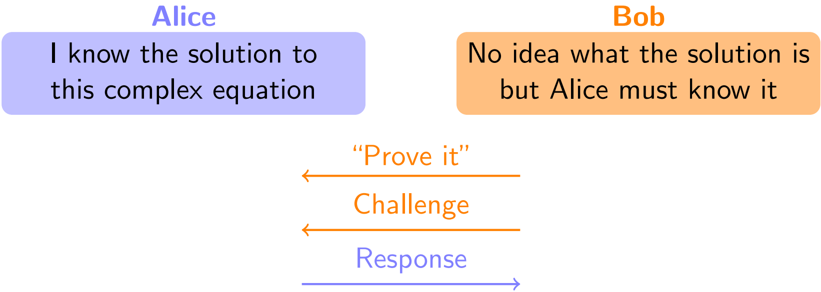
ConsenSys, LIX and Inria, Paris, France

GT CARAMBA 09/02/2021

Overview

1. Preliminaries
   - Zero-knowledge proof (ZKP)
   - ZK-SNARK
   - proof composition
2. Choice of elliptic curves
   - Theory
   - Implementation
3. Future work

Zero-Knowledge Proof of Knowledge, Informally



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 $g^x = y$

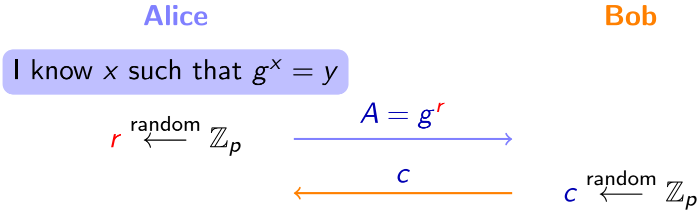
Zero-Knowledge for public keys: Sigma protocol

Alice **Bob**

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

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

Zero-Knowledge for public keys: Sigma protocol



Alice

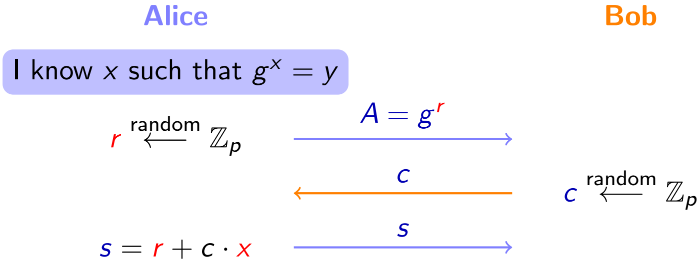
Bob

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

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

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

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 $g^x = y$

$r \overset{\text{random}}{\longleftarrow} \mathbb{Z}_p \qquad \xrightarrow{A = g^r}$

$\xleftarrow{c} \qquad c \overset{\text{random}}{\longleftarrow} \mathbb{Z}_p$

$s = r + c \cdot x \qquad \xrightarrow{s} \qquad g^s \overset{?}{=} A \cdot y^c$

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 (NIKZ) Sigma protocol

Alice

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

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

$$A = g^r$$

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

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

$$\xrightarrow{\pi = (A, c, s)}$$

Bob

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

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

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

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

setup, prover?, verifier?

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

Communication complexity

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

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]
- ...

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 $\implies$ 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 computationally bounded prover knows a secret information.

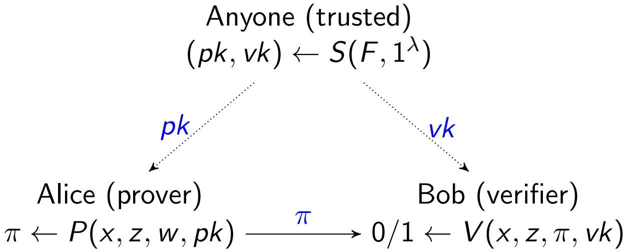
Zero-knowledge proof

Preprocessing ZK-SNARK of NP language

F: public NP program, x, z: public inputs
w: private input s.t. $z := F(x, w)$

A ZK-SNARK consists of algorithms S, P, V s.t. for a security parameter λ:

$$\begin{array}{rlcl} \textit{Trapdoored}\ \text{Setup:} & (pk_\tau, vk_\tau) & \leftarrow & S(F, \tau, 1^\lambda) \\ \text{Prove:} & \pi_w & \leftarrow & P(x, z, w, pk) \\ \text{Verify:} & 0/1 & \leftarrow & V(x, z, \pi, vk) \end{array}$$



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

Alice (prover)
$\pi \leftarrow P(x, z, w, pk) \xrightarrow{\pi} 0/1 \leftarrow V(x, z, \pi, vk)$
Bob (verifier)

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.

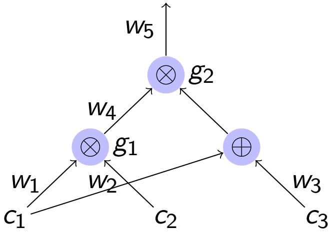
ZK-SNARKs in a nutshell

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



$$U_i(x)V_i(x) - W_i(x) = H(x)_i T(x) \qquad (QAP)$$

$$U_i(\tau)V_i(\tau) - W_i(\tau) = H_i(\tau)T_i(\tau)$$

$$\mathtt{HH}(U_i(\tau)V_i(\tau) - W_i(\tau) = H_i(\tau)T_i(\tau))$$

Arithmetization example

QAP

- F program with $N = n_{in} + n_{out} \in \mathbb{F}$ I/O
- circuit of depth m
- QAP $\equiv u_i(x)$, $v_i(x)$ and $w_i(x)$, $i \in 0, 1 \ldots m$ and $t(x)$ of degree d in $\mathbb{F}[x]$.

$c_1, \ldots, c_N \in \mathbb{F}$ is a valid assignment of $F \iff \exists c_{N+1}, \ldots, c_m \in \mathbb{F}$ s.t. $t(x)|p(x)$, where $p(x)$ is:

$$(u_0(x) + \sum_{i=1}^{m} c_i u_i(x)) + (v_0(x) + \sum_{i=1}^{m} c_i v_i(x)) - (c_0(x) + \sum_{i=1}^{m} c_i w_i(x)) \quad (1)$$

Blind evaluation of QAP

Alice has a set of polynomials of degree d and she wants to convince Bob that they verify the QAP (eq. 1). Instead of verifying the QAP on the whole domain $\mathbb{F}$, she can 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 $\mathbb{F}$.

So, if we choose the field $\mathbb{F}$ carefully, $\tau \xleftarrow{\$} \mathbb{F}$ is assumed to be picked at random and since $t(x)h(x), p(x)$ are non-zero polynomials, the possibility of a false proof to verify is bounded by a negligible fraction.

Blind evaluation of QAP

Let's take the example of polynomial U_i:

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

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

Blind evaluation of QAP

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

So we need a homomorphic hiding function wrt **d additions**. But we also have to compute multiplications $U \cdot V$ and $h \cdot t$. So we need the homomorphic property wrt to **one multiplication** as well.

$$(u_0(\tau) + \sum_{i=1}^{m} c_i u_i(\tau)) + (v_0(\tau) + \sum_{i=1}^{m} c_i v_i(\tau)) - (c_0(\tau) + \sum_{i=1}^{m} c_i w_i(\tau))$$
$$= h(\tau)t(\tau)$$

Blind evaluation of QAP

This can be achieved with bilinear pairings:

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

$$e(h(\tau)G_1, t(\tau)G_2) \cdot e(\hat{W}(\tau)G_1, G_2) = e(\hat{U}(\tau)G_1, \hat{V}(\tau)G_2)$$

$$e(G_1, G_2)^{h(\tau)t(\tau)} \cdot e(G_1, G_2)^{\hat{W}(\tau)} = e(G_1, G_2)^{\hat{U}(\tau)\hat{V}(\tau)}$$

$$Cte^{h(\tau)t(\tau)+\hat{W}(\tau)} = Cte^{\hat{U}(\tau)\hat{V}(\tau)}$$

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

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

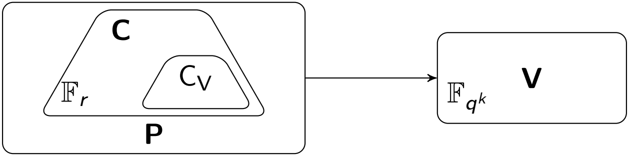
- $0/1 \leftarrow V(\Phi, \pi, vk)$ where V is

$$e(A, B) = vk_{\alpha,\beta} \cdot e(vk_x, vk_\gamma) \cdot e(C, vk_\delta) \qquad (O_\lambda(|\Phi|)) \qquad (2)$$

  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 $(vk_\alpha, vk_\beta) \in \mathbb{G}_1 \times \mathbb{G}_2$.

Proof composition

A proof of a proof



How easy/difficult is to express `V` (Eq. 2) as an instance Φ of a `NP` program in `C` ?

Remember that `V` (Eq. 2) lies in $\mathbb{F}_{q^k}$ and `C` in $\mathbb{F}_r$, where q is the field size of an elliptic curve E and r its prime subgroup order.

- 1st attempt: choose a curve for which $q = r$ (DLP broken)
- 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 [BCTV14a, BCG+20]

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

$$\#E_2(\mathbb{F}_{q_2}) = q_1, \ldots,\ \#E_i(\mathbb{F}_{q_i}) = q_{i-1}, \ldots,\ \#E_m(\mathbb{F}_{q_m}) = q_{m-1}\ . \tag{3}$$

Definition

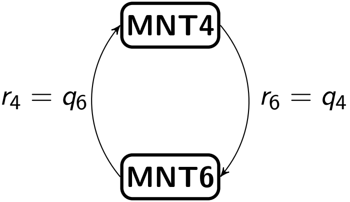
An *m*-cycle of elliptic curves is an *m*-chain, with

$$\#E_1(\mathbb{F}_{q_1}) = q_m\ . \tag{4}$$

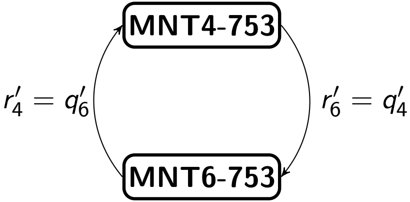
Proof composition

cycles and chains of pairing-friendly elliptic curves

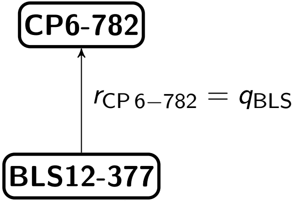
(A joint work with Aurore Guillevic,
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France)



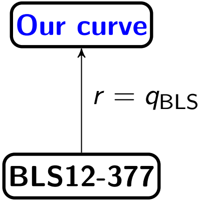
libff [sl18]



Mina [MS18]



Zexe [BCG+20]



This work [EG20]

Proof composition

cycles and chains of pairing-friendly elliptic curves

$E/\mathbb{F}_q$	q	r	k	d	a, b	λ
MNT4	$q_4 = r_6$ (298b)	$r_4 = q_6$ (298b)	4	2	$a = 2, b = *$	77
MNT6	$q_6 = r_4$ (298b)	$r_6 = q_4$ (298b)	6	2	$a = 11, b = *$	87
MNT4-753	$q'_4 = r'_6$ (753b)	$r'_4 = q'_6$ (753b)	4	2	$a = 2, b = *$	113
MNT6-753	$q'_6 = r'_4$ (753b)	$r'_6 = q'_4$ (753b)	6	2	$a = 11, b = *$	137
BLS12-377	q_{BLS} (377b)	r_{BLS} (253b)	12	6	$a = 0, b = 1$	125
CP6	$q_{\text{CP 6}}$ (782b)	$r_{\text{CP 6}} = q_{\text{BLS}}$ (377b)	6	2	$a = 5, b = *$	138
This work	q (761b)	$r = q_{\text{BLS}}$ (377b)	6	6	$a = 0, b = -1$	126

Table: 2-cycle and 2-chain examples.

Recall that $E/\mathbb{F}_q : y^2 = x^3 + ax + b$ has a subgroup of order r, an embedding degree k, a twist of order d and an approximate security of λ-bit.

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 $k \in \mathbb{N}^*$)~~
- $4q - t^2 = Dy^2$ (for $D < 10^{12}$) and some integer y

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

```
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;
```

Our work

Limitations and improvements

- $\rho = \log_2 q / \log_2 r \approx 2$ (because $q = f(t^2, y^2)$ and $t, y \xleftarrow{\$} \bmod r$).
- 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.

Our work

Notes

- $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k}) \cong E'[r](\mathbb{F}_{q^{k/d}})$ for a twist E' of degree d.
- When $-D = -3$, there exists a twist E' of degree $d = 6$.
- Associated with a choice of $\xi \in \mathbb{F}_{q^{k/6}}$ s.t. $x^6 - \xi \in \mathbb{F}_{q^{k/6}}[x]$ is irreducible, the equation of E' can be either
    - $y^2 = x^3 + b/\xi$ and we call it a D-twist or
    - $y^2 = x^3 + b \cdot \xi$ and we call it a M-twist.
- For the D-type, $E' \to E : (x, y) \mapsto (\xi^{1/3}x, \xi^{1/2}y)$,
- For the M-type $E' \to E : (x, y) \mapsto (\xi^{2/3}x/\xi, \xi^{1/2}y/\xi)$

Our work

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]

Our work

The suggested curve: BW6-761

We found the following curve $E : y^2 = x^3 - 1$ over $\mathbb{F}_q$ of 761-bit. The parameters are expressed in polynomial forms and evaluated at the seed $x_0 = \texttt{0x8508c00000000}$. For pairing computation we use the M-twist curve $E' : y^2 = x^3 + 4$ over $\mathbb{F}_q$ to represent $\mathbb{G}_2$ coordinates.

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

BW6-761

Features

- is over 761-bit instead of 782-bit, we save one machine-word of 64 bits.
- has an embedding degree $k = 6$ and a twist of order $d = 6$, allowing $\mathbb{G}_2$ coordinates to be in $\mathbb{F}_q$ (factor 6 compression).
- parameters have polynomial expressions, allowing fast implementation.
- has a very efficient optimal ate pairing.
- has faster cyclotomic squaring in $\mathbb{F}_{q^6}$.
- has CM discriminant $-D = -3$, allowing fast GLV multiplication on both $\mathbb{G}_1$ and $\mathbb{G}_2$.
- has fast subgroup checks and fast cofactor multiplication on $\mathbb{G}_1$ and $\mathbb{G}_2$ via endomorphisms.
- has fast and secure hash-to-curve methods for both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Our work

Cost estimation of a pairing

$e(P,Q) = f_{t-1,Q}(P)^{(q^6-1)/r}$ $(t-1)$ of 388 bits, $Q \in \mathbb{F}_{q^3}$

$e(P,Q) = (f_{x_0+1,Q}(P) f^q_{x_0^3-x_0^2-x_0,Q}(P))^{(q^6-1)/r}$ x_0 of 64 bits, $Q \in \mathbb{F}_q$

$$(q^6-1)/r = \underbrace{(q^3-1)(q+1)}_{\text{easy part}}\underbrace{(q^2-q+1)/r}_{\text{hard part}} = \begin{cases} \text{easy part} \times (w_0 + qw_1) \\ \text{easy part} \times f(x_0, q^i) \end{cases}$$

Curve	Prime	Pairing	Miller loop	Exponentiation	Total
BLS12	377-bit	ate	6705 m_{384}	7063 m_{384}	13768 m_{384}
CP6	782-bit	ate	47298 m_{832}	10521 m_{832}	57819 m_{832}
This	761-bit	opt. ate	7911 m_{768}	5081 m_{768}	12992 m_{768}

m_b base field multiplication, b bitsize in Montgomery domain on a 64-bit platform

x4.5 less operations in a smaller field by one machine-word

Our work

Rust implementation timings

Implemented in `ZEXE` Rust library [**?**] and tested on a 2.2 GHz Intel Core i7 x86_64 processor with 16 Go 2400 MHz DDR4 memory running macOS Mojave 10.14.6. Rust compiler is Cargo 1.43.0.

Pull request url: `https://github.com/scipr-lab/zexe/pull/210`

Curve	Pairing	Miller loop	Exponentiation	Total	Eq. 2
BLS12	ate	0.7ms	1.3ms	2ms	3.4ms
CP6-782	ate (ZEXE)	76.1ms	8.1ms	84.2ms	309.4ms
Our curve	opt. ate	2.5ms	3ms	5.5ms	10.5ms

x15 faster to compute a pairing

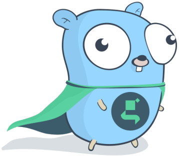
x29 faster to verify a Groth16 proof

N.B.: Affine pairing on CP6-761 can be optimized by implementing faster inverse in F_{q^3}

Implemenetation

zkTeam at consenSys

- goff: Fast finite field arithmetic in Golang
  `(github, hackmd)`
- gurvy: Elliptic curves arithmetic in Golang (BN, BLS12, BW6)
  `(github, hackmd)`
- gnark: Zero-knowledge proof in Golang (Groth16, recursive Groth16)
  `(github, hackmd)`



Future work

- Can we construct cycles of embedding degrees greater than 6?
- In particular, can we construct a 2-cycle of higher embedding degrees?
- Can we characterize all 2-chains with arbitrary embedding degrees?

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.

Youssef El Housni and Aurore Guillevic.
Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition.
Cryptology ePrint Archive, Report 2020/351, 2020.
`https://eprint.iacr.org/2020/351`.

References III

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.

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.

References IV

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.

Izaak Meckler and Evan Shapiro.
Coda: Decentralized cryptocurrency at scale.
O(1) Labs whitepaper, 2018.
`https://cdn.codaprotocol.com/v2/static/coda-whitepaper-05-10-2018-0.pdf`.

scipr lab.
libff: C++ library for finite fields and elliptic curves., 2018.
`https://github.com/scipr-lab/libff`.