Optimized and secure pairing-friendly elliptic curve suitable for one layer proof composition

Youssef El Housni[1] Aurore Guillevic[2]

[1]Ernst & Young, Inria and École polytechnique, Paris, France

[2]Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

CANS 2020

Overview

1. Preliminaries
   - Zero-knowledge proof
   - ZK-SNARK
2. Proof composition
   - Notations
   - Techniques
3. Our work
   - Theory
   - Implementation
4. Applications

Zero-knowledge proof

What is a zero-knowledge proof?

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

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.

"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:

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

$$\begin{tikzcd} & \text{Anyone (trusted)} \\ & (pk, vk) \leftarrow S(F, 1^\lambda) \arrow[dl, dotted, "pk"'] \arrow[dr, dotted, "vk"] & \\ \text{Alice (prover)} & & \text{Bob (verifier)} \\ \pi \leftarrow P(x, z, w, pk) \arrow[rr, "\pi"] & & 0/1 \leftarrow V(x, z, \pi, vk) \end{tikzcd}$$

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.

- $E\colon y^2 = x^3 + ax + b$ 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 $\mathtt{NP}$ 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 $(vk_\alpha, vk_\beta) \in \mathbb{G}_1 \times \mathbb{G}_2$.

Proof composition

Blockchains and ZK-SNARKs

A blockchain is a *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

$$\text{Transparent} \xrightarrow[\text{Problem}]{} \text{confidentiality} \qquad \xrightarrow[\text{Solution}]{} \text{ZK-SNARK}$$

π is zero-knowledge

$$\text{Immutable} \xrightarrow[\text{Problem}]{} \text{scalability} \qquad \xrightarrow[\text{Solution}]{} \text{ZK-SNARK}$$

π is $O_\lambda(1)$

$$\text{Paying} \xrightarrow[\text{Problem}]{} \text{cost} \qquad \xrightarrow[\text{Solution}]{} \text{ZK-SNARK}$$

V is $O_\lambda(|\Phi|)$

Example: On Ethereum blockchain ($\lambda \approx 110$-bit), π_{Groth16} is 254 bytes and V_{Groth16} costs $(200k + 8k/\text{input}) \times \text{gas} \approx 0.37\text{eur} + \ldots$.

Proof composition

Question: How much does it cost in space and fees to verify 1000 proofs ?

Answer: 254000 bytes and $>$ 370 euros !

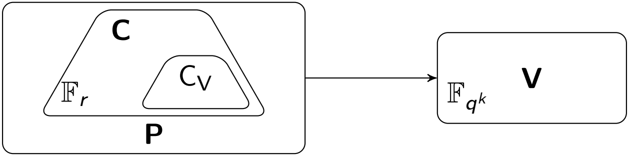
Question: Can we aggregate 1000 proofs into a single constant-size proof ? Can we verify 1000 proofs at the cost of 1 proof ? Can we do both ?

Answer: Since the verification algorithm V (Eq. 1) is an `NP` program, generate a new proof that verifies the correctness of the previous 1000 proofs.

- Scenario 1: The number of proofs to aggregate is not fixed and you need to aggregate on the fly
- Scenario 2: The number of proofs to aggregate is fixed and you need to aggregate only once

Proof composition

A proof of a proof



How easy/difficult is to express V (Eq. 1) as an instance Φ of a NP program in C ?

Remember that V (Eq. 1) 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 $q = r$ (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 [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

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

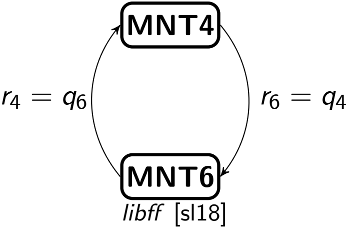
Definition

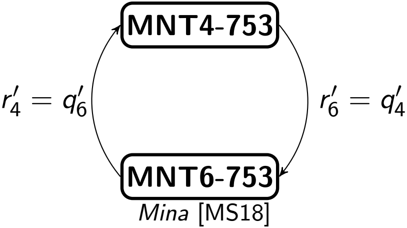
An *m*-cycle of elliptic curves is an *m*-chain, with

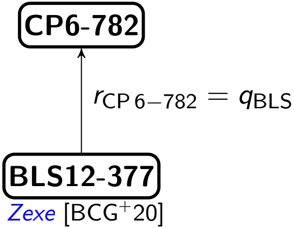
$$\#E_1(\mathbb{F}_{q_1}) = q_m \ . \qquad (3)$$

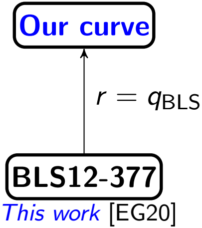
Proof composition

cycles and chains of pairing-friendly elliptic curves









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	$q_{\mathrm{CP\,6}}$ (782b)	$r_{\mathrm{CP\,6}} = q_{\mathrm{BLS}}$ (377b)	6	2	$a = 5, b = *$	138
This work	q (761b)	$r = q_{\mathrm{BLS}}$ (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.

Our work

ZK-curves

- SNARK
  - $E/\mathbb{F}_q$ — BN, BLS12, BW12?, KSS16? ... [FST10]
    - pairing-friendly
    - $r-1$ highly 2-adic
- 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
    - $q_{\{1,2\}} - 1$ highly 2-adic
- Recursive SNARK (2-chain)
  - $E_1/\mathbb{F}_{q_1}$ — BLS12 ($seed \equiv 1 \bmod 3 \cdot 2^{adicity}$) [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
    - $r_2 = q_1$

Our work

Snarky 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 over CP

- $\rho = \log_2 q / \log_2 r \approx 2$ (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

1. 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 ζ_k are in $K = \mathbb{Q}[x]/r(x)$;
2. Choose $t(x) \in \mathbb{Q}[x]$ be a polynomial representing $\zeta_k + 1$ in K;
3. Set $y(x) \in \mathbb{Q}[x]$ be a polynomial mapping to $(\zeta_k - 1)/\sqrt{-D}$ in K;
4. 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 size$(q) \leq 768$ bits $\implies$ smallest machine-word size
2. Brezing–Weng method
   - choose $r(x) = q_{\mathrm{BLS\,12-377}}(x)$
   - $q(x) = (t^2(x) + 3y^2(x))/4$ 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]

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

Our work

Features

- The curve is over 761-bit instead of 782-bit, we save one machine-word of 64 bits.
- The curve 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).
- The curve parameters have polynomial expressions, allowing fast implementation.
- The curve has a very efficient optimal ate pairing.
- The curve has CM discriminant $-D = -3$, allowing fast GLV multiplication on both $\mathbb{G}_1$ and $\mathbb{G}_2$.
- The curve has fast subgroup checks and fast cofactor multiplication on $\mathbb{G}_1$ and $\mathbb{G}_2$ via endomorphisms.
- The curve has fast and secure hash-to-curve methods for both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Our work

Cost estimation of a pairing

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

$e(P, Q) = (f_{x_0+1,Q}(P) f^{q}_{x_0^3-x_0^2-x_0,Q}(P))^{(q^6-1)/r}$ $\quad$ 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 [SL20] 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. 1
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}

Applications
Industrial blockchain projects

- **Aleo**: private applications (https://aleo.org/)
  - already in use: https://developer.aleo.org/autogen/advanced/the_aleo_curves/overview
- **Celo**: batched verification of BLS signatures (https://celo.org/)
  - already in use: https://github.com/celo-org/celo-bls-snark-rs
- **Clearmatics Zecale**: general purpose zk-SNARK aggregator (https://www.clearmatics.com)
  - already in use: https://github.com/clearmatics/zecale
- **EY Midnight ZVM™** : private smart contracts on Ethereum
  - *to be released soon*

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., pages 1–35, 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.

References V

SCIPR-LAB.
Zexe (zero knowledge execution), 2020.
https://github.com/scipr-lab/zexe.