The arithmetic of pairing-based proof systems

Youssef El Housni

PhD defense — Palaiseau, November 18, 2022

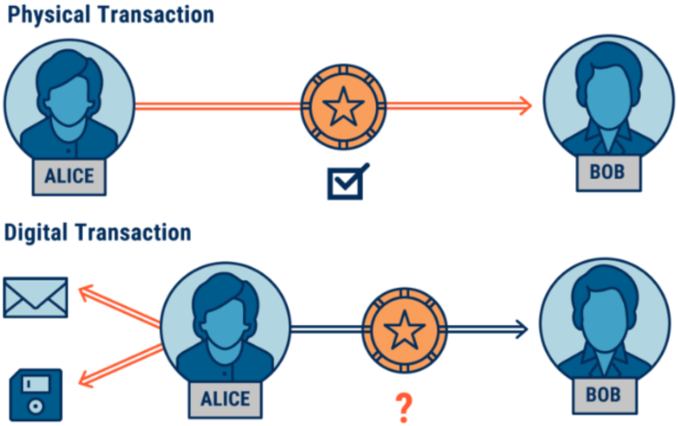
Overview

1. Motivation
2. zk-SNARK
3. SNARK-friendly curves
4. SNARK-friendly 2-chains
5. Pairings in R1CS
6. Multi-scalar multiplication
7. Conclusion

Overview

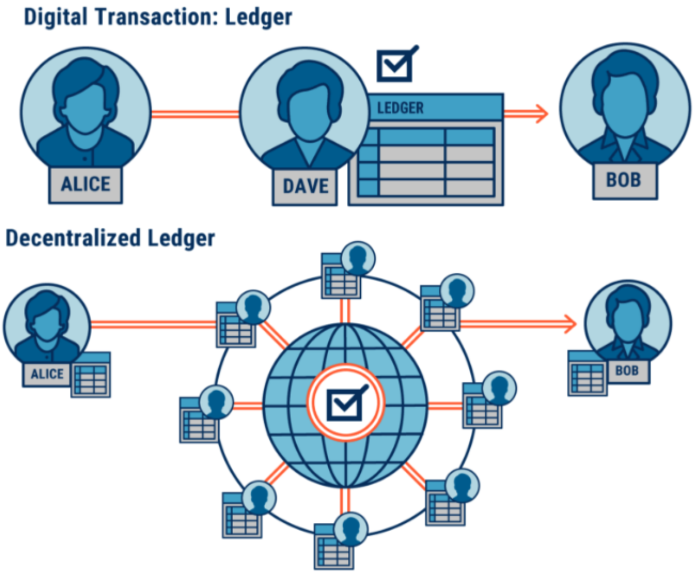
1 Motivation
2 zk-SNARK
3 SNARK-friendly curves
4 SNARK-friendly 2-chains
5 Pairings in R1CS
6 Multi-scalar multiplication
7 Conclusion

The story of Alice and Bob



(Courtesy of CBINSIGHTS)

The story of Alice and Bob



Blockchains

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

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

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

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

Overview

1. Motivation
2. zk-SNARK
3. SNARK-friendly curves
4. SNARK-friendly 2-chains
5. Pairings in R1CS
6. Multi-scalar multiplication
7. Conclusion

Zero-knowledge proofs (ZKP)

Alice

I know the solution to this complex equation

Bob

No idea what the solution is but Alice claims to know it

Challenge
$\longleftarrow$

Response
$\longrightarrow$

Zero-knowledge proofs (ZKP)

Alice

I know the solution to this complex equation

Bob

No idea what the solution is but Alice claims to know it

Challenge $\longleftarrow$

Response $\longrightarrow$

- **Sound**: **Alice** has a wrong solution $\implies$ **Bob** is not convinced.

Zero-knowledge proofs (ZKP)

Alice

I know the solution to
this complex equation

Bob

No idea what the solution is
but Alice claims to know it

$$\xleftarrow{\text{Challenge}}$$

$$\xrightarrow{\text{Response}}$$

- **Sound**: **Alice** has a wrong solution $\implies$ **Bob** is not convinced.
- **Complete**: **Alice** has the solution $\implies$ **Bob** is convinced.

Zero-knowledge proofs (ZKP)

Alice

I know the solution to
this complex equation

Bob

No idea what the solution is
but Alice claims to know it

$\xleftarrow{\text{Challenge}}$

$\xrightarrow{\text{Response}}$

- **Sound**: **Alice** has a wrong solution $\implies$ **Bob** is not convinced.
- **Complete**: **Alice** has the solution $\implies$ **Bob** is convinced.
- **Zero-knowledge**: **Bob** does NOT learn the solution.

Example: Sigma protocol

Alice

Bob

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

Example: Sigma protocol

Alice **Bob**

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

$n \xleftarrow{\$} \mathbb{Z}_r$ $\xrightarrow{A = g^n}$

Example: Sigma protocol

Alice **Bob**

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

$n \xleftarrow{\$} \mathbb{Z}_r$

$$\xrightarrow{A = g^n}$$

$$\xleftarrow{c}$$

$c \xleftarrow{\$} \mathbb{Z}_r$

Example: Sigma protocol

Alice **Bob**

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

$n \xleftarrow{\$} \mathbb{Z}_r$ $\xrightarrow{A = g^n}$

$\xleftarrow{c}$ $c \xleftarrow{\$} \mathbb{Z}_r$

$s = n + c \cdot x$ $\xrightarrow{s}$

Example: Sigma protocol

Alice **Bob**

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

$n \xleftarrow{\$} \mathbb{Z}_r$ $\xrightarrow{A = g^n}$

$\xleftarrow{c}$ $c \xleftarrow{\$} \mathbb{Z}_r$

$s = n + c \cdot x$ $\xrightarrow{s}$ $g^s \stackrel{?}{=} A \cdot y^c$

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

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

Non-Interactive Zero-Knowledge (NIZK) Sigma protocol

Alice

Bob

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

$$n \xleftarrow{\$} \mathbb{Z}_r$$
$$A = g^n$$
$$c = H(A, y)$$
$$s = n + c \cdot x$$

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

$$g^s \stackrel{?}{=} A \cdot y^c$$
$$c \stackrel{?}{=} H(A, y)$$

Non-Interactive Zero-Knowledge (NIZK) Sigma protocol

Alice

Bob

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

$$n \xleftarrow{\$} \mathbb{Z}_r$$

$$\underbrace{g}_{\text{Setup}} ;\ A = g^n$$

$$\underbrace{\pi = (A, c, s)}_{\text{proof}} \longrightarrow$$

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

$$\underbrace{s = n + c \cdot x}_{\text{Prove}}$$

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

$$\underbrace{c \stackrel{?}{=} H(A, y)}_{\text{Verify}}$$

ZKP families

Expressivity

- *specific* statement vs. *general* statement

ZKP families

Expressivity

- *specific* statement vs. *general* statement

Deployability

- *interactive* vs. $non - interactive$ protocol
- *trapdoored* setup vs. *transparent* setup
- *Designated* verifier vs. *any* verifier

ZKP families

Expressivity

- *specific* statement vs. *general* statement

Deployability

- *interactive* vs. *non – interactive* protocol
- *trapdoored* setup vs. *transparent* setup
- *Designated* verifier vs. *any* verifier

Complexity

- prover complexity (Alice)
- verifier complexity (Bob)
- communication complexity (size of the proof and the setup)

ZKP families

Expressivity

- *specific* statement vs. *general* statement

Deployability

- *interactive* vs. $non - interactive$ protocol
- *trapdoored* setup vs. *transparent* setup
- *Designated* verifier vs. *any* verifier

Complexity

- prover complexity (Alice)
- verifier complexity (Bob)
- communication complexity (size of the proof and the setup)

Security

- Cryptographic assumptions
- Cryptographic primitives

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 work [GMR85]
- Non-Interactive ZKP [BFM88]
- Succinct ZKP [K92]
- Succinct Non-Interactive ZKP [M94]

ZKP literature landmarks

- First ZKP work [GMR85]
- Non-Interactive ZKP [BFM88]
- Succinct ZKP [K92]
- Succinct Non-Interactive ZKP [M94]
- Pairing-based succinct NIZK [Groth10]

ZKP literature landmarks

- First ZKP work [GMR85]
- Non-Interactive ZKP [BFM88]
- Succinct ZKP [K92]
- Succinct Non-Interactive ZKP [M94]
- Pairing-based succinct NIZK [Groth10]
- "SNARK" terminology and characterization of existence [BCCT11]
- Pairing-based SNARK in quasi-linear prover time [GGPR13]
- Pairing-based SNARK with shortest proof and verifier time [Groth16]

ZKP literature landmarks

- First ZKP work [GMR85]
- Non-Interactive ZKP [BFM88]
- Succinct ZKP [K92]
- Succinct Non-Interactive ZKP [M94]
- Pairing-based succinct NIZK [Groth10]
- "SNARK" terminology and characterization of existence [BCCT11]
- Pairing-based SNARK in quasi-linear prover time [GGPR13]
- Pairing-based SNARK with shortest proof and verifier time [Groth16]
- SNARK with universal and updatable setup [GKMMM18, BKMM19 (Sonic), GWC19 (PlonK), CHMMVW19 (Marlin),...]

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.

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

A proof is very "short" and "easy" to verify.

Non-interactive

No interaction between the prover and verifier for proof generation and verification (except the proof message).

ARgument of Knowledge

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

Preprocessing zk-SNARK for NP language

F: public NP program, x, z: public inputs, w: private input

$$z := F(x, w)$$

Preprocessing zk-SNARK for NP language

F: public $\mathtt{NP}$ program, x, z: public inputs, w: private input

$$z := F(x, w)$$

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

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

Preprocessing zk-SNARK for NP language

F: public NP program, x, z: public inputs, w: private input

$$z := F(x, w)$$

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

$$\begin{array}{llll} \textit{Setup}: & (pk, vk) & \leftarrow & S(F, 1^\lambda) \\ \textit{Prove}: & \pi & \leftarrow & P(x, z, w, pk) \end{array}$$

Preprocessing zk-SNARK for NP language

F: public $\texttt{NP}$ program, x, z: public inputs, w: private input

$$z := F(x, w)$$

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

$$\begin{array}{llll} \textit{Setup}: & (pk, vk) & \leftarrow & S(F, 1^\lambda) \\ \textit{Prove}: & \pi & \leftarrow & P(x, z, w, pk) \\ \textit{Verify}: & \texttt{false/true} & \leftarrow & V(x, z, \pi, vk) \end{array}$$

Preprocessing zk-SNARK for NP language

F: public NP program, x, z: public inputs, w: private input

$$z := F(x, w)$$

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

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

Anyone

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

pk $\qquad\qquad$ vk

Alice (prover) $\qquad\qquad$ Bob (verifier)

$\pi \leftarrow P(x, z, w, pk) \xrightarrow{\quad\pi\quad} V(x, z, \pi, vk)$?

(Trapdoored) preprocessing zk-SNARK for NP language

F: public $\mathtt{NP}$ program, x, z: public inputs, w: private input

$$z := F(x, w)$$

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 :	$\mathtt{false/true}$	$\leftarrow$	$V(x, z, \pi, vk)$

$$\begin{tikzcd}
 & \begin{array}{c}\text{TTP (secret trapdoor } \tau\text{)}\\ (pk, vk) \leftarrow S(F, \tau, 1^\lambda)\end{array} \arrow[dl, dotted, "pk"'] \arrow[dr, dotted, "vk"] & \\
\begin{array}{c}\text{Alice (prover)}\\ \pi \leftarrow P(x, z, w, pk)\end{array} \arrow[rr, "\pi"] & & \begin{array}{c}\text{Bob (verifier)}\\ V(x, z, \pi, vk)?\end{array}
\end{tikzcd}$$

(Trapdoored) preprocessing zk-SNARK for NP language

F: public $\mathtt{NP}$ program, x, z: public inputs, w: private input

$$z := F(x, w)$$

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

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

MPC (secret trapdoor τ)

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

pk vk

Alice (prover) Bob (verifier)

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

zk-SNARK

Succinctness: A 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:

zk-SNARKs in a nutshell

Main ideas:

1. Reduce a "general statement" satisfiability to a polynomial equation satisfiability.

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.

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.

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. Make the protocol non-interactive.

Arithmetization

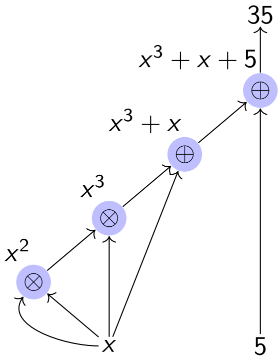
Statement $\rightarrow$ **Arithmetic circuit** $\rightarrow$ Intermediate representation $\rightarrow$ Polynomial identities $\rightarrow$ zk-SNARK proof

$$x^3 + x + 5 = 35 \qquad ({\color{red}x = 3})$$

Arithmetization

Statement $\rightarrow$ **Arithmetic circuit** $\rightarrow$ Intermediate representation $\rightarrow$ Polynomial identities $\rightarrow$ zk-SNARK proof

$$x^3 + x + 5 = 35 \qquad ({\color{red}x = 3})$$



Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ **Intermediate representation** $\rightarrow$ Polynomial identities $\rightarrow$ zk-SNARK proof

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

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

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

witness:

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

$$O \bullet \vec{w} = L \bullet \vec{w} \cdot R \bullet \vec{w}$$

Arithmetization

e.g. Quadratic Arithmetic Program

Statement → Arithmetic circuit → Intermediate representation → **Polynomial identities** → zk-SNARK proof

$$L(X)R(X) - O(X) = H(X)T(X) \qquad (QAP \in \mathbb{F}[X])$$

Arithmetization

e.g. Quadratic Arithmetic Program

Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ Intermediate representation $\rightarrow$ **Polynomial identities** $\rightarrow$ zk-SNARK proof

$$L(X)R(X) - O(X) = H(X)T(X) \qquad (QAP \in \mathbb{F}[X])$$

$$L(\tau)R(\tau) - O(\tau) = H(\tau)T(\tau) \qquad (\textit{trapdoor } \tau \xleftarrow{\$} \mathbb{F})$$

Arithmetization

e.g. Quadratic Arithmetic Program

Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ Intermediate representation $\rightarrow$ **Polynomial identities** $\rightarrow$ zk-SNARK proof

$$
\begin{aligned}
L(X)R(X) - O(X) &= H(X)T(X) && (QAP \in \mathbb{F}[X]) \\
L(\tau)R(\tau) - O(\tau) &= H(\tau)T(\tau) && (\textit{trapdoor } \tau \xleftarrow{\$} \mathbb{F}) \\
C(L(\tau)R(\tau) - O(\tau)) &= C(H(\tau)T(\tau)) && (\textit{Homomorphic commitment})
\end{aligned}
$$

Succinct evaluation of polynomials

Instead of verifying the QAP on the whole domain $\mathbb{F}$ $\rightarrow$ 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 polynomials

Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ Intermediate representation $\rightarrow$ Polynomial identities $\rightarrow$ **zk-SNARK proof**

Let's take the example of polynomial L:

Blind evaluation of polynomials

Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ Intermediate representation $\rightarrow$ Polynomial identities $\rightarrow$ **zk-SNARK proof**

Let's take the example of polynomial L:

- Alice can send L to Bob and he computes $L(\tau)$ $\rightarrow$ breaks the zero-knowledge.

Blind evaluation of polynomials

Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ Intermediate representation $\rightarrow$ Polynomial identities $\rightarrow$ **zk-SNARK proof**

Let's take the example of polynomial L:

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

Blind evaluation of polynomials

Statement $\rightarrow$ Arithmetic circuit $\rightarrow$ Intermediate representation $\rightarrow$ Polynomial identities $\rightarrow$ **zk-SNARK proof**

Let's take the example of polynomial L:

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

$\Longrightarrow$ homomorphic cryptography to evaluate $L(X)$ at τ without Bob learning L nor Alice learning τ.

Blind evaluation of polynomials

$$L(\tau) = l_0 + l_1\tau + l_2\tau^2 + \cdots + l_d\tau^d \in \mathbb{F}$$
$$C(L(\tau)) = l_0 C(1) + l_1 C(\tau) + l_2 C(\tau^2) + \cdots + l_d C(\tau^d)$$

Somewhat homomorphic commitment w.r.t.:

- depth-d **additions** (arbitrary d)
- depth-1 **multiplications** (for $L(\tau) \cdot R(\tau)$ and $H(\tau) \cdot T(\tau)$).

Blind evaluation of polynomials

Somewhat homomorphic commitment w.r.t.:

- depth-d **additions** (arbitrary d)

Blind evaluation of polynomials

Somewhat homomorphic commitment w.r.t.:

- depth-d **additions** (arbitrary d)

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

Blind evaluation of polynomials

Somewhat homomorphic commitment w.r.t.:

- depth-d **additions** (arbitrary d)

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

- depth-1 **multiplications** (for $L(\tau) \cdot R(\tau)$ and $H(\tau) \cdot T(\tau)$).

$$C(\tau_1) = \tau_1 G;\ C(\tau_2) = \tau_2 G$$
$$C(\tau_1) \cdot C(\tau_2) = C(\tau_1 \cdot \tau_2) \qquad (?)$$

Blind evaluation of polynomials

Somewhat homomorphic commitment w.r.t.:

- depth-d **additions** (arbitrary d)

$$C(\tau) = \tau G \qquad (DL)$$

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

- depth-1 **multiplications** (for $L(\tau) \cdot R(\tau)$ and $H(\tau) \cdot T(\tau)$).

$$C(\tau_1) = \tau_1 G;\ C(\tau_2) = \tau_2 G$$

$$C(\tau_1) \cdot C(\tau_2) = C(\tau_1 \cdot \tau_2) \qquad (?)$$

$$\underbrace{e(C(\tau_1), C(\tau_2))}_{\text{product of commitments}} = \underbrace{Z^{\tau_1 \cdot \tau_2}}_{\substack{\text{new commitment to } \tau_1 \cdot \tau_2 \\ \text{(where } Z = e(G, G))}} \qquad \text{(bilinear pairing)}$$

Blind evaluation of QAP

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

$$
\begin{aligned}
e(C(H(\tau)), C(T(\tau)) \cdot e(C(O(\tau)), C(1)) &= e(C(L(\tau)), C(R(\tau))) \\
e(H(\tau)G, T(\tau)G) \cdot e(O(\tau)G, G) &= e(L(\tau)G, R(\tau)G) \\
e(G, G)^{H(\tau)T(\tau)} \cdot e(G, G)^{O(\tau)} &= e(G, G)^{L(\tau)R(\tau)} \\
Z^{H(\tau)T(\tau)+O(\tau)} &= Z^{L(\tau)R(\tau)}
\end{aligned}
$$

Outline of contributions

Outline of contributions

- Blockchain limitations: confidentiality and scalability

Outline of contributions

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)

Outline of contributions

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**

Outline of contributions

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**
- Proof composition for better confidentiality and scalability $\rightarrow$ 2-chains and 2-cycles **[CANS 2020, EuroCrypt 2022, DCC 2022]**

Outline of contributions

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**
- Proof composition for better confidentiality and scalability $\rightarrow$ 2-chains and 2-cycles **[CANS 2020, EuroCrypt 2022, DCC 2022]**
- Pairings in R1CS for fast generation of the composed proof **[ACNS 2023]**

Outline of contributions

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**
- Proof composition for better confidentiality and scalability $\rightarrow$ 2-chains and 2-cycles **[CANS 2020, EuroCrypt 2022, DCC 2022]**
- Pairings in R1CS for fast generation of the composed proof **[ACNS 2023]**
- Multi-scalar multiplication for fast generation of proofs **[(in submission), zprize winner]**

Overview

1. Motivation
2. zk-SNARK
3. SNARK-friendly curves
4. SNARK-friendly 2-chains
5. Pairings in R1CS
6. Multi-scalar multiplication
7. Conclusion

Instantiation

DL:

- $E\colon y^2 = x^3 + ax + b$ 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.

Instantiation

DL:

- $E: y^2 = x^3 + ax + b$ 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.

Pairing-friendly:

- small embedding degree k (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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Instantiation

DL:

- $E: y^2 = x^3 + ax + b$ 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.

Pairing-friendly:

- small embedding degree k (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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

SNARK-friendly:

- $r - 1 \equiv 0 \mod 2^L$ for some large $L \in \mathbb{N}^*$ ($\mathbb{F}_r$ FFT-friendly)

Instantiation

DL:

- $E: y^2 = x^3 + ax + b$ 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.

Pairing-friendly:

- small embedding degree k (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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

SNARK-friendly:

- $r - 1 \equiv 0 \mod 2^L$ for some large $L \in \mathbb{N}^*$ ($\mathbb{F}_r$ FFT-friendly)

BLS12-381: q-bit=381, r-bit=255, $k = 12$, $L = 32$

SNARK-friendly curves from the literature

[D. F. Aranha, Y.EH, A. Guillevic - DCC 2022]

Curve	seed x	L	$r = \#\mathbb{G}_1$ (bits)	p, $\mathbb{G}_1$ (bits)	$p^{k/d}$, $\mathbb{G}_2$ (bits)	$p \equiv 3 \bmod 4$	security (bits) $\mathbb{G}_1$	$\mathbb{F}^*_{p^k}$
BN-256 [PHGR13]	1868033^3 $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 19$	5	256	256	512	✓	128	103
BN-254 [BFR+13]	$2^{62} - 2^{54} + 2^{44}$ $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 7$	45	254	254	508	×	127	102
GMV6-183 [BCG+13]	`0x8eed757d90615e40000000` $\mathrm{HW}(-26x-2) = 16$	31	181	183	549	NA	90	71
BN-254 [BCTV14b]	`0x44e992b44a6909f1` $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 22$	28	254	254	508	✓	127	103
BLS12-381 [Bow17]	`-0xd201000000010000` $\mathrm{HW}(x) = 6$	32	255	381	762	✓	127	126

Families of SNARK-friendly curves [D. F. Aranha, Y.EH, A. Guillevic - DCC 2022]

Family, $r, p \in \mathbb{N},\ t \in \mathbb{Z}$	$r \equiv 1 \bmod 2^L$	$p \equiv 3 \bmod 4$	$\mathbb{G}_2$ coord. in
BN	$x \equiv 2570880382155688433 \bmod 2^{63} \Rightarrow 2^{64} \mid r-1$	✓	$\mathbb{F}_{p^2}$
any x	$x \equiv 0 \bmod 2^{L-1} \Rightarrow 2^L \mid r-1,\ 2^L \mid p-1$	✗	
BLS12	$x \equiv 1 \bmod 3 \cdot 2^{L-1} \Rightarrow 2^L \mid r-1,\ 2^{L-1} \mid p-1$	✗	
$x \equiv 1$	$x \equiv 2^{L-1}-1 \bmod 3 \cdot 2^{L-1} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	$\mathbb{F}_{p^2}$
mod 3	$x \equiv 2^{L/2} \bmod 3 \cdot 2^{L/2} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	
BLS24	$x \equiv 1 \bmod 3 \cdot 2^{L-2} \Rightarrow 2^L \mid r-1,\ 2^{L-2} \mid p-1$	✗	
$x \equiv 1$	$x \equiv 2^{L-1}-1 \bmod 3 \cdot 2^{L-2} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	$\mathbb{F}_{p^4}$
mod 3	$x \equiv 2^{L/4} \bmod 3 \cdot 2^{L/4} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	
MNT4, $t = x+1$	$x \equiv 0 \bmod 2^{L/2} \Rightarrow 2^L \mid r-1, 2^{L/2} \mid p-1$	✗	$\mathbb{F}_{p^2}$
MNT6	$x \equiv 0 \bmod 2^{L-1} \Rightarrow 2^L \mid r-1, 2^{2L} \mid p-1$	✗	$\mathbb{F}_{p^3}$
GMV6($h=4$) any x	$x \equiv 0 \bmod 2^{L-1} \Rightarrow 2^L \mid r-1,\ 2^{L-1} \mid p-1$	NA	$\mathbb{F}_{p^3}$
KSS16 ($x \equiv \pm 25 \bmod 70$)	$\pm 1439818652098642188 5, \pm 3745661661312336140 5$ mod $35 \cdot 2^{62} \Rightarrow 2^{64} \mid r-1,\ p \equiv 1 \bmod 4$	✗	$\mathbb{F}_{p^4}$
KSS18 ($x \equiv 14 \bmod 42$)	$x = 14 \cdot 2^{L/3} \bmod 42 \cdot 2^{L/3} \Rightarrow 2^L \mid r-1,\ 12 \mid p-1$	NA	$\mathbb{F}_{p^3}$

New SNARK-friendly curves

[D. F. Aranha, Y.EH, A. Guillevic - DCC 2022]

Curve	x	L	$r = \#\mathbb{G}_1$ (bits)	p, $\mathbb{G}_1$ (bits)	$p^{k/d}$, $\mathbb{G}_2$ (bits)	$p \equiv 3 \bmod 4$	security (bits) $\mathbb{G}_1$	security (bits) $\mathbb{F}^*_{p^k}$
BN383	`0x49e69d16fdc80216226909f1` $\mathsf{HW}_{2\text{-NAF}}(6x+2) = 30$	44	383	383	766	✓	191	123
BLS24-317	`0xd9018000` $\mathsf{HW}_{2\text{-NAF}}(x) = 6$	60	255	317	1268	✓	127	160
KSS16-329	`0x38fab7583` $\mathsf{HW}(x) = 12$	19	255	329	1316	✓	127	140
KSS18-345	`0xc0c44000000` $\mathsf{HW}(x) = 6$	78	254	345	690	NA	127	150

https://github.com/yelhousni/gnark-crypto

Co-factor clearing and subgroup membership

[Y.EH, A. Guillevic, T. Piellard - AfricaCrypt 2022]

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

- Pairing groups: $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are sub-groups of some prime order r.
- They are defined over some larger groups of composite orders $\underbrace{c_{1,2,T}}_{\text{co-factors}} \times r$

Co-factor clearing and subgroup membership

[Y.EH, A. Guillevic, T. Piellard - AfricaCrypt 2022]

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

- Pairing groups: $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are sub-groups of some prime order r.
- They are defined over some larger groups of composite orders $\underbrace{c_{1,2,T}}_{\text{co-factors}} \times r$

Let P be a random element of order $c_1 \times r$

- Co-factor clearing: $P' \in \mathbb{G}_1 \leftarrow [c_1]P$

Co-factor clearing and subgroup membership

[Y.EH, A. Guillevic, T. Piellard - AfricaCrypt 2022]

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

- Pairing groups: $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are sub-groups of some prime order r.
- They are defined over some larger groups of composite orders $\underbrace{c_{1,2,T}}_{\text{co-factors}} \times r$

Let P be a random element of order $c_1 \times r$

- Co-factor clearing: $P' \in \mathbb{G}_1 \leftarrow [c_1]P$

Let Q be a random element of order $c_{1,2,T} \times r$

- Subgroup membership testing: $[r]Q \stackrel{?}{=} \mathcal{O}$

Co-factor clearing and subgroup membership

[Y.EH, A. Guillevic, T. Piellard - AfricaCrypt 2022]

Proposition ($\mathbb{G}_1$ co-factor clearing)

Many curve families have the $\mathbb{G}_1$ cofactor of the form $c_1 = 3\ell^2$. To clear this cofactor, the map $P \mapsto [\ell]P$ is sufficient for all curves in [FST10] except KSS and 6.6 where $k \equiv 2, 3 \mod 6$.

Co-factor clearing and subgroup membership

[Y.EH, A. Guillevic, T. Piellard - AfricaCrypt 2022]

Proposition ($\mathbb{G}_1$ co-factor clearing)

Many curve families have the $\mathbb{G}_1$ cofactor of the form $c_1 = 3\ell^2$. To clear this cofactor, the map $P \mapsto [\ell]P$ is sufficient for all curves in [FST10] except KSS and 6.6 where $k \equiv 2, 3 \mod 6$.

Theorem ($\mathbb{G}_1$ and $\mathbb{G}_2$ membership testing)

Let $q' = q$ or resp. q^k and $c' = c_1$ or resp. c_2. If ψ acts as the multiplication by λ on $E(\mathbb{F}_{q'})[r]$ and $\gcd(\chi(\lambda), c') = 1$ *then*

$$\psi(Q) = [\lambda]Q \iff Q \in E(\mathbb{F}_{q'})[r]$$

with χ the characteristic polynomial of ψ.

Co-factor clearing and subgroup membership

[Y.EH, A. Guillevic, T. Piellard - AfricaCrypt 2022]

Proposition ($\mathbb{G}_1$ co-factor clearing)

Many curve families have the $\mathbb{G}_1$ cofactor of the form $c_1 = 3\ell^2$. To clear this cofactor, the map $P \mapsto [\ell]P$ is sufficient for all curves in [FST10] except KSS and 6.6 where $k \equiv 2, 3 \mod 6$.

Theorem ($\mathbb{G}_1$ and $\mathbb{G}_2$ membership testing)

Let $q' = q$ or resp. q^k and $c' = c_1$ or resp. c_2. If ψ acts as the multiplication by λ on $E(\mathbb{F}_{q'})[r]$ and $\gcd(\chi(\lambda), c') = 1$ then

$$\psi(Q) = [\lambda]Q \iff Q \in E(\mathbb{F}_{q'})[r]$$

with χ the characteristic polynomial of ψ.

Proposition ($\mathbb{G}_T$ membership testing)

For $z \in \mathbb{F}_{p^k}^*$ and Φ_k the k-th cyclotomic polynomial, we have:

$$z^{\Phi_k(p)} = 1 \text{ and } z^p = z^{t-1} \text{ and } \gcd(p + 1 - t, \Phi_k(p)) = r \implies z^r = 1 \ .$$

Overview

1. Motivation
2. zk-SNARK
3. SNARK-friendly curves
4. SNARK-friendly 2-chains
5. Pairings in R1CS
6. Multi-scalar multiplication
7. Conclusion

A pairing-based SNARK

Example: Groth16 [Gro16]

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

A pairing-based SNARK

Example: Groth16 [Gro16]

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

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

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

A pairing-based SNARK

Example: Groth16 [Gro16]

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

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

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

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

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

A pairing-based SNARK

Example: Groth16 [Gro16]

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

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

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

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

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

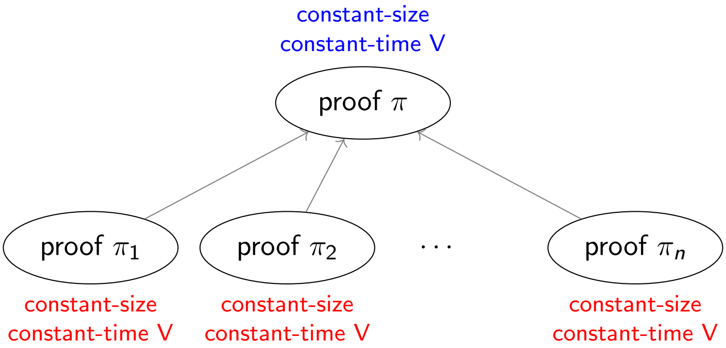
- Verify: $0/1 \leftarrow V(\Phi, \pi, vk)$ where V is

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

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

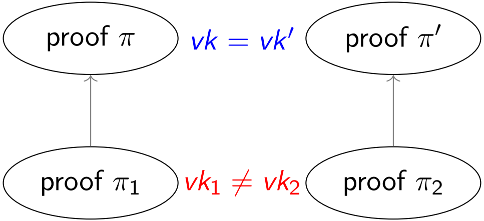
Proof composition: why?

Aggregation:

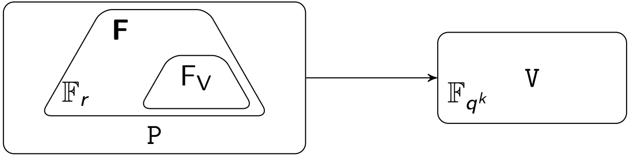


Proof composition: why?

Decentralized private computation (DPC):

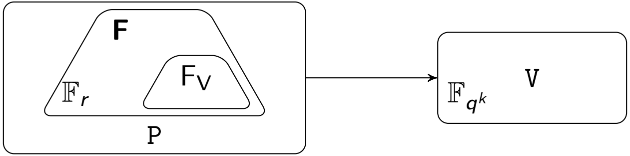


Proof composition: how?



- **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 $\mathbb{F}_r$

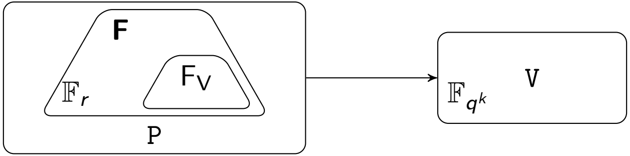
Proof composition: how?



- **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)

Proof composition: how?



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

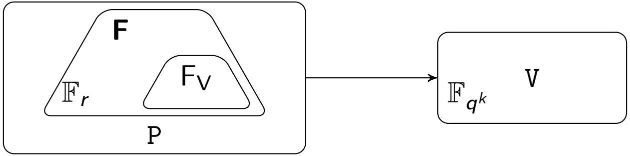
P proving is performed over $\mathbb{G}_1$ (and $\mathbb{G}_2$) (of order r)

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)

Proof composition: how?



- **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]

2-cycles and 2-chains

A 2-cycle of elliptic curves:

$$\begin{tikzcd} E_1(\mathbb{F}_q) \arrow[d, bend left, "\#E_0(\mathbb{F}_p) = q"] \\ E_0(\mathbb{F}_p) \arrow[u, bend left, "\#E_1(\mathbb{F}_q) = p"] \end{tikzcd}$$

A 2-chain of elliptic curves:

$$\begin{tikzcd} E_1(\mathbb{F}_q) \\ E_0(\mathbb{F}_p) \arrow[u, "\#E_1(\mathbb{F}_q) = h \cdot p"'] \end{tikzcd}$$

2-chains of elliptic curves

SNARK-0 with a pairing e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ $\#\mathbb{G}_i = r_0$

pairing statement

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

proof of statement

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

inner curve:
curve $E_0(\mathbb{F}_p)$
of prime order r_0

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

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

SNARK-friendly curves, 2-cycles and 2-chains

- SNARK
  - $E/\mathbb{F}_q$ — BN, BLS12, BW12?, KSS16? ... [FST10]
    - pairing-friendly
    - $2^L \mid r - 1$
- Recursive SNARK (2-cycle)
  - $E_0/\mathbb{F}_p$ and $E_1/\mathbb{F}_q$ — MNT4/MNT6 [FST10, Sec.5], ? [CCW19]
    - both pairing-friendly
    - $\#E_1(\mathbb{F}_q) = p$ and $\#E_0(\mathbb{F}_p) = q$
    - $2^L \mid p - 1$
    - $2^L \mid q - 1$
- Recursive SNARK (2-chain)
  - $E_0/\mathbb{F}_p$ — BLS12 ($x \equiv 1 \bmod 3 \cdot 2^L$) [BCG+20], ?
    - pairing-friendly
    - $2^L \mid r_0 - 1$ ($r_0 \mid \#E_0(\mathbb{F}_p)$)
    - $2^L \mid p - 1$
  - $E_1/\mathbb{F}_q$ — Cocks–Pinch algorithm [ZEXE]
    - pairing-friendly
    - $p \mid \#E_1(\mathbb{F}_q)$

2-chains: outer curve $E_1/\mathbb{F}_q$

- q is a prime or a prime power
- t is relatively prime to q
- ~~r is prime~~
- $\cancel{r \mid q^k - 1}$
- $\cancel{r \mid q + 1 - t}$

r is a **fixed** chosen prime
s.t. $r \mid q + 1 - t$
and $r \mid q^k - 1$

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

2-chains: outer curve $E_1/\mathbb{F}_q$

- q is a prime or a prime power
- t is relatively prime to q
- ~~r is prime~~
- ~~$r \mid q^k - 1$~~
- ~~$r \mid q + 1 - t$~~

$\left.\right\}$ r is a **fixed** chosen prime s.t. $r \mid q + 1 - t$ and $r \mid q^k - 1$

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

Algorithm: Cocks–Pinch method

Fix k and D and choose a prime r s.t. $k|r-1$ and $\left(\frac{-D}{r}\right) = 1$;
Compute $t = 1 + x^{(r-1)/k}$ for x a generator of $(\mathbb{Z}/r\mathbb{Z})^\times$;
Compute $y = (t-2)/\sqrt{-D} \bmod r$;
Lift t and y in $\mathbb{Z}$;
Compute $q = (t^2 + Dy^2)/4$ (in $\mathbb{Q}$);

2-chains: outer curve $E_1/\mathbb{F}_q$

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

2-chains: outer curve $E_1/\mathbb{F}_q$

- $\rho = \log_2 q / \log_2 r \approx 2$ (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: Brezing–Weng method

Fix k and D and choose an irreducible polynomial $r(x) \in \mathbb{Z}[x]$ with positive leading coefficient s.t. $\sqrt{-D}$ and the primitive k-th root of unity ζ_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 $\mathbb{Q}[x]$;

2-chains: outer curve $E_1/\mathbb{F}_q$

- $\rho = \log_2 q / \log_2 r \approx 2$ (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: Brezing–Weng method

Fix k and D and choose an irreducible polynomial $r(x) \in \mathbb{Z}[x]$ with positive leading coefficient s.t. $\sqrt{-D}$ and the primitive k-th root of unity ζ_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 ?

2-chains: outer curve $E_1/\mathbb{F}_q$

[Y.EH, A. Guillevic - CANS 2020]

1. Cocks–Pinch method
   - $k = 6$ and $-D = -3$ $\implies$ 128-bit security, $\mathbb{G}_2$ coordinates in $\mathbb{F}_q$ (pairing over $\mathbb{F}_q$ instead if $\mathbb{F}_{q^3}$), GLV multiplication over $\mathbb{G}_1$ and $\mathbb{G}_2$
   - restrict search to size$(q) \leq 768$ bits $\implies$ smallest machine-word size

2-chains: outer curve $E_1/\mathbb{F}_q$

[Y.EH, A. Guillevic - CANS 2020]

1. Cocks–Pinch method
   - $k = 6$ and $-D = -3$ $\implies$ 128-bit security, $\mathbb{G}_2$ coordinates in $\mathbb{F}_q$ (pairing over $\mathbb{F}_q$ instead if $\mathbb{F}_{q^3}$), 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 $r(x) = q_{\mathrm{BLS12}}(x)$
   - $q(x) = (t^2(x) + 3y^2(x))/4$ factors $\implies$ $q(x_0)$ cannot be prime
   - lift in $\mathbb{Z}$ $t = r \times h_t + t(x_0)$ and $y = r \times h_y + y(x_0)$ [FK19, GMT20]

2-chains: outer curve $E_1/\mathbb{F}_q$

[Y.EH, A. Guillevic - CANS 2020]

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

Our curve, $k = 6$, $D = 3$, $r = q_{\mathrm{BLS12}}$
$r(x) = (x^6 - 2x^5 + 2x^3 + x + 1)/3 = q_{\mathrm{BLS12-377}}(x)$ $t(x) = x^5 - 3x^4 + 3x^3 - x + 3 + h_t r(x)$ $y(x) = (x^5 - 3x^4 + 3x^3 - x + 3)/3 + h_y r(x)$ $q(x) = (t^2 + 3y^2)/4$ $q_{h_t=13,h_y=9}(x) = (103x^{12} - 379x^{11} + 250x^{10} + 691x^9 - 911x^8$ $-79x^7 + 623x^6 - 640x^5 + 274x^4 + 763x^3 + 73x^2 + 254x + 229)/9$

SNARK-0: inner curves

[Y.EH, A. Guillevic - EuroCrypt 2022]

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)

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

SNARK-0: inner curves

[Y.EH, A. Guillevic - EuroCrypt 2022]

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)

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

Universal SNARK

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

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

SNARK-1: outer curves

[Y.EH, A. Guillevic - EuroCrypt 2022]

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 ≈ 768 bits with $(t \mod x) \mod r \equiv 0$ or 3

SNARK-1: outer curves

[Y.EH, A. Guillevic - EuroCrypt 2022]

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 $\approx$ 768 bits with $(t \mod x) \mod r \equiv 0$ or 3

Universal SNARK

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

$\rightarrow$ BW ($k = 6$) family of $\approx$ 704 bits with $(t \mod x) \mod r \equiv 0$ or 3
$\rightarrow$ CP ($k = 8$) family of $\approx$ 640 bits
$\rightarrow$ CP ($k = 12$) family of $\approx$ 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

[Y.EH, A. Guillevic - EuroCrypt 2022]

Short list of 2-chains with some additional nice engineering properties:

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

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

(on aAMD EPYC 7R32 AWS (c5a.24xlarge) machine)

`https://github.com/ConsenSys/gnark-crypto`

Overview

1 Motivation

2 zk-SNARK

3 SNARK-friendly curves

4 SNARK-friendly 2-chains

5 Pairings in R1CS

6 Multi-scalar multiplication

7 Conclusion

Cost of pairing-based SNARKs

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.

	Setup	Prove	Verify
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 (PLONK-KZG)	$d_{\geq n+a}\ \mathtt{M}_{\mathbb{G}_1},\ 1\ \mathtt{M}_{\mathbb{G}_2}$	$9(n+a)\ \mathtt{M}_{\mathbb{G}_1}$	$2\ \mathtt{P},\ 18\ \mathtt{M}_{\mathbb{G}_1}$

Cost of pairing-based SNARKs

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.

	Setup	Prove	Verify
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 (PLONK-KZG)	$d_{\geq n+a}\ \mathtt{M}_{\mathbb{G}_1},\ 1\ \mathtt{M}_{\mathbb{G}_2}$	$9(n+a)\ \mathtt{M}_{\mathbb{G}_1}$	$2\ \mathtt{P},\ 18\ \mathtt{M}_{\mathbb{G}_1}$

F_V: program that checks $\mathtt{V}$ (eq. 1) ($\ell = 1$, $n = 90000$)

Pairings out-circuit

ate pairing

$$
\begin{aligned}
e : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T \\
(P, Q) &\mapsto f_{t-1,Q}(P)^{(q^k-1)/r}
\end{aligned}
$$

- $f_{t-1,Q}(P)$ is the Miller function
- $f \mapsto f^{(q^k-1)/r}$ is the final exponentiation

Examples: For polynomial families in the seed x,

BLS12 $e(P, Q) = f_{x,Q}(P)^{(q^{12}-1)/r}$

BLS24 $e(P, Q) = f_{x,Q}(P)^{(q^{24}-1)/r}$

[BN06, AKL+11, ABLR14, ABLR14, Sco19] [HHT20, AFK+13, GF16, GS10, Kar13]

Pairings out-circuit: Miller algorithm

Algorithm: MillerLoop(s, P, Q)

Output: $m = f_{s,Q}(P)$

$m \leftarrow 1$; $R \leftarrow Q$

for *b from the second most significant bit of s to the least* **do**

  $\ell \leftarrow \ell_{R,R}(P)$; $R \leftarrow [2]R$; $v \leftarrow v_{[2]R}(P)$ Doubling Step

  $m \leftarrow m^2 \cdot \ell / v$

  if $b = 1$ **then**

    $\ell \leftarrow \ell_{R,Q}(P)$; $R \leftarrow R + Q$; $v \leftarrow v_{R+Q}(P)$ Addition Step

    $m \leftarrow m \cdot \ell / v$

return m

Pairings out-circuit: Miller algorithm

Algorithm: MillerLoop(s, P, Q)

Output: $m = f_{s,Q}(P)$

$m \leftarrow 1$; $R \leftarrow Q$

for *b from the second most significant bit of s to the least* **do**

  $\ell \leftarrow \ell_{R,R}(P)$; $R \leftarrow [2]R$; Doubling Step

  $m \leftarrow m^2 \cdot \ell$

  if $b = 1$ **then**

    $\ell \leftarrow \ell_{R,Q}(P)$; $R \leftarrow R + Q$; Addition Step

    $m \leftarrow m \cdot \ell$

return m

Pairings out-circuit: Miller algorithm

Algorithm: MillerLoop(s, P, Q)

Output: $m = f_{s,Q}(P)$

$m \leftarrow 1$; $R \leftarrow Q$

for *b from the second most significant bit of s to the least* **do**

  $\ell \leftarrow \ell_{R,R}(P)$; $R \leftarrow [2]R$; Doubling Step

  $m \leftarrow m^2 \cdot \ell$

  if $b = 1$ **then**

    $\ell \leftarrow \ell_{R,Q}(P)$; $R \leftarrow R + Q$; Addition Step

    $m \leftarrow m \cdot \ell$

return m

Pairings in-circuit (R1CS)

[Y.EH - ACNS 2023]

	Time	Constraints
BLS12-377	< 1 ms	$\approx$ **80 000**

Inverses, in R1CS, cost (almost) as much as multiplications !

- Miller loop:
  - Affine coordinates $\rightarrow\approx$ 19k (`arkworks`)
  - Division in extension fields
  - Double-and-Add in affine
  - lines evaluations (1/y, x/y)
  - Loop with short addition chains
  - Torus-based arithmetic
- Final Exponentiation:
  - Karatsuba cyclotomic squarings
  - Torus-based arithmetic
  - Exp. with short addition chains

19k $\rightarrow\approx$ 11k (`gnark`)

Pairings in-circuit (R1CS)

[Y.EH - ACNS 2023]

e.g. For BLS12-377,

https://github.com/ConsenSys/gnark

	Constraints
Pairing	**11535**
Groth16 verifier	19378
BLS sig. verifier	14888
KZG verifier	20679

For BLS24-315, a pairing is **27608** contraints .

More optimizations in mind:

- Quadruple-and-Add Miller loop [CBGW10]
- Fixed argument Miller loop (KZG, BLS sig) [CS10]
- ~~Longa's sums of products Mul [Lon22]~~

Overview

1. Motivation
2. zk-SNARK
3. SNARK-friendly curves
4. SNARK-friendly 2-chains
5. Pairings in R1CS
6. **Multi-scalar multiplication**
7. Conclusion

Multi-Scalar-Multiplication (MSM)

[Y.EH and G. Botrel - *In submission*]

$a_1P_1 + a_2P_2 + \cdots + a_nP_n$ with $P_i \in \mathbb{G}_1$ (or $\mathbb{G}_2$) and $a_i \in \mathbb{F}_r(|r| = \text{b}-bit)$

- Step 1: reduce the b-bit MSM to several c-bit MSMs for some chosen fixed $c \leq b$
- Step 2: solve each c-bit MSM efficiently
- Step 3: combine the c-bit MSMs into the final b-bit MSM

Multi-Scalar-Multiplication (MSM)

[Y.EH and G. Botrel - *In submission*]

$a_1P_1 + a_2P_2 + \cdots + a_nP_n$ with $P_i \in \mathbb{G}_1$ (or $\mathbb{G}_2$) and $a_i \in \mathbb{F}_r(|r| =\text{b}-bit)$

- Step 1: reduce the b-bit MSM to several c-bit MSMs for some chosen fixed $c \leq b$
- Step 2: solve each c-bit MSM efficiently
- Step 3: combine the c-bit MSMs into the final b-bit MSM

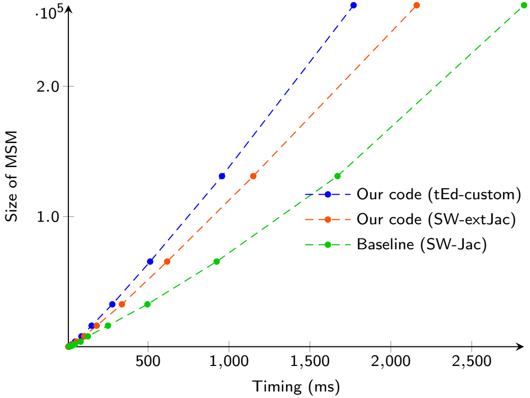
$\rightarrow$ Overall cost is: $b/c(n + 2^{c-1}) + (b - c - b/c - 1)$

- Mixed re-additions: to accumulate P_i in the c-bit MSM buckets with cost $b/c(n - 2^{c-1} + 1)$
- Additions: to combine the bucket sums with cost $b/c(2^c - 3)$
- Additions and doublings: to combine the c-bit MSMs into the b-bit MSM with cost $b - c + b/c - 1$
  - $b/c - 1$ additions and
  - $b - c$ doublings

Our MSM code vs. the ZPrize baseline (BLS12-377 $\mathbb{G}_1$)

[Y.EH and G. Botrel - *In submission*]

- All inner curves have a twisted Edwards form $-y^2 + x^2 = 1 + dx^2y^2$
- We use a custom coordinates system $(y - x : y + x : 2dxy) \rightarrow (7\mathtt{m}$ per addition)
- 2-NAF buckets, Parallelism, software optimizations...



Samsung Galaxy A13 5G (Model SM-A136ULGDXAA with SoC MediaTek Dimensity 700 (MT6833))

Overview

1. Motivation
2. zk-SNARK
3. SNARK-friendly curves
4. SNARK-friendly 2-chains
5. Pairings in R1CS
6. Multi-scalar multiplication
7. Conclusion

Industrial Impact

gnark

CONSENSYS

celo

Aleo

EY

Go Ethereum

ESPRESSO SYSTEMS

Summary

Summary

- Blockchain limitations: confidentiality and scalability

Summary

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)

Summary

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**

Summary

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**
- Proof composition for better confidentiality and scalability $\rightarrow$ 2-chains and 2-cycles **[CANS 2020, EuroCrypt 2022, DCC 2022]**

Summary

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**
- Proof composition for better confidentiality and scalability $\rightarrow$ 2-chains and 2-cycles **[CANS 2020, EuroCrypt 2022, DCC 2022]**
- Pairings in R1CS for fast generation of the composed proof **[ACNS 2023]**

Summary

- Blockchain limitations: confidentiality and scalability
- pairing-based zk-SNARKs are a solution (constant-size proof and fast verification)
- What are SNARK-friendly curves? Fast arithmetic? **[DCC 2022, AfricaCrypt 2022]**
- Proof composition for better confidentiality and scalability $\rightarrow$ 2-chains and 2-cycles **[CANS 2020, EuroCrypt 2022, DCC 2022]**
- Pairings in R1CS for fast generation of the composed proof **[ACNS 2023]**
- Multi-scalar multiplication for fast generation of proofs **[(in submission), zprize winner]**

Perspectives

- Is it possible to find a silver bullet construction of elliptic curves that can address all the efficiency/security requirements?
- Are there more efficient cycles of pairing-friendly curves? How to generate them?

Perspectives

- Is it possible to find a silver bullet construction of elliptic curves that can address all the efficiency/security requirements?
- Are there more efficient cycles of pairing-friendly curves? How to generate them?
- Can we get rid of the FFT-friendliness?
  - Field-agnostic SNARKs [Brakedown, Orion]
  - FFT over non-smooth fields [ECFFT]

Diego F. Aranha, Paulo S. L. M. Barreto, Patrick Longa, and Jefferson E. Ricardini.
The realm of the pairings.
In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 3–25. Springer, Heidelberg, August 2014.

Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez.
Implementing pairings at the 192-bit security level.
In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 177–195. Springer, Heidelberg, May 2013.

References II

Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio Cesar López-Hernández.
Faster explicit formulas for computing pairings over ordinary curves.
In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 48–68. Springer, Heidelberg, May 2011.

Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza.
SNARKs for C: Verifying program executions succinctly and in zero knowledge.
In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

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.

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.

References IV

Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath Setty, Andrew J. Blumberg, and Michael Walfish.
Verifying computations with state.
In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 341–357, New York, NY, USA, 2013. Association for Computing Machinery.
ePrint with major differences at `ePrint 2013/356`.

Paulo S. L. M. Barreto and Michael Naehrig.
Pairing-friendly elliptic curves of prime order.
In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006.

Sean Bowe.
BLS12-381: New zk-SNARK elliptic curve construction.
Zcash blog, March 11 2017.
`https://blog.z.cash/new-snark-curve/`.

Craig Costello, Colin Boyd, Juan Manuel González Nieto, and Kenneth Koon-Ho Wong.
Avoiding full extension field arithmetic in pairing computations.
In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 203–224. Springer, Heidelberg, May 2010.

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 VI

- 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.

- Craig Costello and Douglas Stebila.
  Fixed argument pairings.
  In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 92–108. Springer, Heidelberg, August 2010.

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

References VII

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

Loubna Ghammam and Emmanuel Fouotsa.
On the computation of the optimal ate pairing at the 192-bit security level.
Cryptology ePrint Archive, Report 2016/130, 2016.
`https://eprint.iacr.org/2016/130`.

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.

Robert Granger and Michael Scott.
Faster squaring in the cyclotomic subgroup of sixth degree extensions.
In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 209–223. Springer, Heidelberg, May 2010.

Daiki Hayashida, Kenichiro Hayasaka, and Tadanori Teruya.
Efficient final exponentiation via cyclotomic structure for pairings over families of elliptic curves.
Cryptology ePrint Archive, Report 2020/875, 2020.
https://eprint.iacr.org/2020/875.

References IX

Koray Karabina.
Squaring in cyclotomic subgroups.
Math. Comput., 82(281):555–579, 2013.

Patrick Longa.
Efficient algorithms for large prime characteristic fields and their application to bilinear pairings and supersingular isogeny-based protocols.
Cryptology ePrint Archive, Report 2022/367, 2022.
https://eprint.iacr.org/2022/367.

Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova.
Pinocchio: Nearly practical verifiable computation.
In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

References X

Michael Scott.
Pairing implementation revisited.
Cryptology ePrint Archive, Report 2019/077, 2019.
https://eprint.iacr.org/2019/077.