Pairings in Rank-1 Constraint Systems

Youssef El Housni

Consensys (Linea)

SIAM-AG 2023 — July 12th, 2023

Overview

Overview

1. Preliminaries
   - SNARKs
   - Bilinear pairings
2. Motivations
   - Applications
3. Pairings in-circuit
   - R1CS
   - Optimizations

Preprocessing ZK-SNARK of NP language

Let F be a public NP program, x and z be public inputs, and w be a private input such that

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

.

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

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

Preprocessing ZK-SNARK of NP language

Let F be a public NP program, x and z be public inputs, and w be a private input such that

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

.

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

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

Preprocessing ZK-SNARK of NP language

Let F be a public $\texttt{NP}$ program, x and z be public inputs, and w be a private input such that

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

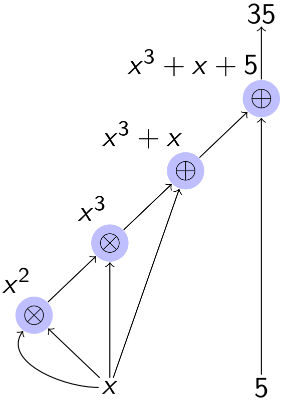
.

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

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

SNARKs of arithmetic circuits

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



SNARKs examples: Groth16 and PLONK

- m = number of wires
- n = number of multiplications gates
- a = number of additions gates
- ℓ = number of public inputs
- $\mathtt{M}_{\mathbb{G}}$ = multiplication in $\mathbb{G}$
- P=pairing

	Setup	Prove	Verify
Groth16 [Gro16]	$3n\ \mathtt{M}_{\mathbb{G}_1}$ $m\ \mathtt{M}_{\mathbb{G}_2}$	$(3n+m-\ell)\ \mathtt{M}_{\mathbb{G}_1}$ $n\ \mathtt{M}_{\mathbb{G}_2}$ 7 FFT	3P $\ell\ \mathtt{M}_{\mathbb{G}_1}$
PLONK (KZG) [GWC]	$d_{\geq n+a}\ \mathtt{M}_{\mathbb{G}_1}$ $1\ \mathtt{M}_{\mathbb{G}_2}$ 8 FFT	$9(n+a)\ \mathtt{M}_{\mathbb{G}_1}$ 8 FFT	2P $18\ \mathtt{M}_{\mathbb{G}_1}$

Bilinear pairings

- $E\colon y^2 = x^3 + ax + b$ 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.
- k embedding degree, smallest integer $k \in \mathbb{N}^*$ s.t. $r \mid q^k - 1$.
- a bilinear pairing

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

  - $\mathbb{G}_1 \subset E(\mathbb{F}_q)$ a group of order r
  - $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$ a group of order r
  - $\mathbb{G}_T \subset \mathbb{F}_{q^k}^*$ group of r-th roots of unity

Overview

- **Proof aggregation or**
- **Private computation (ZEXE)**
  e.g. G16 proof $\pi = (A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$
  and $vk = (vk_1, vk_2, vk_3, vk_4) \in \mathbb{G}_T \times \mathbb{G}_1^{\ell+1} \times \mathbb{G}_2 \times \mathbb{G}_2$

$$\mathtt{V}: \quad e(A, B) \stackrel{?}{=} vk_1 \cdot e(vk_2', vk_3) \cdot e(C, vk_4) \qquad (O_\lambda(\ell)) \qquad (1)$$

and $vk_2' = \sum_{i=0}^{\ell} [x_i] vk_2$.

- **BLS signatures**

$$\mathtt{V}: \quad e(\sigma, \mathbb{G}_2) \stackrel{?}{=} e(H(m), Q_{pk}) \tag{2}$$

where $\sigma \in \mathbb{G}_1$ is the signature, $H(m)$ the message hashed into $\mathbb{G}_1$ and Q_{pk} the public key of the sender.

- **Proof of KZG verification (zkEVM)**
  Proof of $P(z) = y$ ($P \in \mathbb{F}_r[X]$)

$$\mathtt{V}: \quad e(\pi, vk - [z]\mathbb{G}_2) \stackrel{?}{=} e(C - [y]\mathbb{G}_1, \mathbb{G}_2) \tag{3}$$

where $C \in \mathbb{G}_1$ is the commitment and $vk \in \mathbb{G}_1$ the verification key.

Pairings out-circuit

ate pairing

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

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

Pairings out-circuit: Miller algorithm

Algorithm 1: 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 1: 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 1: 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

$\mathbb{G}_2$: - Coordinates compressed in $\mathbb{F}_{q^{k/d}}$ instead of $\mathbb{F}_{q^k}$ (where d is the twist degree) [BN06]
- Homogeneous projective coordinates (X, Y, Z) [AKL+11, ABLR14]
- Sharing computation between Double/Add and lines evaluation [AKL+11, ABLR14]

Finite fields: - $\mathbb{F}_p \to \cdots \to \mathbb{F}_{p^{k/d}} \to \cdots \to \mathbb{F}_{p^k}$
- efficient representation of line (multiplying the line evaluation by a factor $\to$ wiped out later) [ABLR14]
- efficient sparse multiplications in $\mathbb{F}_{p^k}$ [Sco19]

Pairings out-circuit: Final exponentiation

$$\frac{p^k - 1}{r} = \underbrace{\frac{p^k - 1}{\Phi_k(p)}}_{\text{easy part}} \cdot \underbrace{\frac{\Phi_k(p)}{r}}_{\text{hard part}}$$

easy part: a polynomial in p with small coefficients (Frobenius maps)
e.g. (BLS12): `1F2 + 1Conj + 1Inv + 1Mul` in $\mathbb{F}_{p^{12}}$

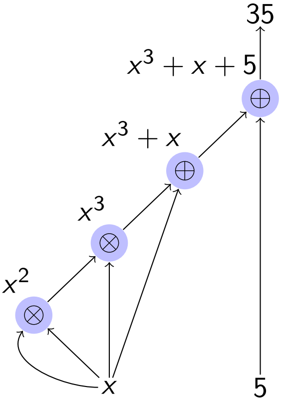
hard part: More expensive. Vectorial or lattice-based
Optimizations [HHT, AFK+13, GF16]
dominating cost: `CycloSqr` [GS10, Kar13] + `Mul` in $\mathbb{F}_{p^k}$

Overview

1. Preliminaries
   - SNARKs
   - Bilinear pairings
2. Motivations
   - Applications
3. Pairings in-circuit
   - R1CS
   - Optimizations

Rank-1 Constraint System

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



constraints:

$$\color{blue}{o = l \cdot r}$$

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

witness:

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

Pairings in-circuit

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

This work: 80 000 $\rightarrow$ 11 500

R1CS is about writing $o = l \cdot r$

- Over $\mathbb{F}_p$:
  - Square = Mul ($o = l \cdot l$)
  - Inv = Mul + 1C ($1/l = o \rightarrow 1 \stackrel{?}{=} l \cdot o$ with o an input hint)
  - Div = Mul + 1C ($r/l = o \rightarrow r \stackrel{?}{=} l \cdot o$ with o an input hint)
  - Inv+Mul $\rightarrow$ Div
- Over $\mathbb{F}_{p^e}$:
  - Square $\neq$ Mul (e.g. $\mathbb{F}_{p^2}$ 2C vs 3C)
  - Inv = Mul + eC ($1/l = o \rightarrow 1 \stackrel{?}{=} l \cdot o$ with o an input hint)
  - Div = Mul + eC ($r/l = o \rightarrow r \stackrel{?}{=} l \cdot o$ with o an input hint)
  - Inv+Mul $\rightarrow$ Div

$\mathbb{G}_2$ Double: $[2](x_1, y_1) = (x_3, y_3)$

$$\lambda = 3x_1^2/2y_1$$
$$x_3 = \lambda^2 - 2x_1$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

$\mathbb{G}_2$ Add: $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$

$$\lambda = (y_1 - y_2)/(x_1 - x_2)$$
$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_2 - x_3) - y_2$$

	Div (5C)	Sq (2C)	Mul (3C)	total
Double	1	2	1	12C
Add	1	1	1	10C

In the Miller loop, when $b = 1 \implies [2]R + Q \to$ **22C**
Instead: $[2]R + Q = (R + Q) + R \to$ **20C**
Better: omit y_{R+Q} computation in $(R + Q) + R \to$ **17C** [ELM03]
$\mathbb{G}_2$ Double-and-Add: $[2](x_1, y_1) + (x_2, y_2) = (x_4, y_4)$

$$\lambda_1 = (y_1 - y_2)/(x_1 - x_2)$$
$$x_3 = \lambda_1^2 - x_1 - x_2$$
$$\lambda_2 = -\lambda_1 - 2y_1/(x_3 - x_1)$$
$$x_4 = \lambda_2^2 - x_1 - x_3$$
$$y_4 = \lambda_2(x_1 - x_4) - y_1$$

	Div (5C)	Sq (2C)	Mul (3C)	total
Double-and-Add	2	2	1	**17C**

Optimizations

lines evaluation

- ℓ is $ay + bx + c = 0 \in \mathbb{F}_{p^2}$
- $\ell_{\psi([2]R)}(P)$ and $\ell_{\psi(R+Q)}(P)$ are of the form $(a'y_P, 0, 0, b'x_P, c', 0) \in \mathbb{F}_{p^{12}}$ ($\psi : E'(\mathbb{F}_{p^{k/d}}) \to E(\mathbb{F}_{p^k})$) [ABLR14]
  $\to$ sparse multiplication (1) in $\mathbb{F}_{p^{12}}$
- precompute $1/y_P$ (1C) and x_P/y_P (1C) and $\ell(P)$ becomes $(1, 0, 0, b'x_P/y_P, c'/y_p, 0) \in \mathbb{F}_{p^{12}}$
  $\to$ better sparse multiplication (2) in $\mathbb{F}_{p^{12}}$

	total
Full Mul	54C
Sparse Mul (1)	39C
Sparse Mul (2)	30C

Easy part:

```
t.Conjugate(m)
m.Inverse(m)  // 66C
t.Mul(t, m)   // 54C
m.FrobeniusSquare(t)
m.Mul(m, t)   // 54C
```

Easy part:

```
t.Conjugate(m)
t.Div(t, m) // 66C
m.FrobeniusSquare(t)
m.Mul(m, t) // 54C
```

Easy part: (more on that later)

```
t.Div(-m[0], m[1]) // 18C
m.TorusFrobeniusSquare(t)
m.TorusMul(m, t) // 42C
r := Decompress(m) // 48C
```

	total
Old	174
New	120
New (Torus)	60 (or 108)

Hard part (Hayashida et al. [HHT])

```
t[0].CyclotomicSquare(m)
t[1].Expt(m) // m^x addchain (Mul + CycloSqr)
t[2].Conjugate(m)
t[1].Mul(t[1], t[2])
t[2].Expt(t[1])
t[1].Conjugate(t[1])
t[1].Mul(t[1], t[2])
t[2].Expt(t[1])
t[1].Frobenius(t[1])
t[1].Mul(t[1], t[2])
m.Mul(m, t[0])
t[0].Expt(t[1])
t[2].Expt(t[0])
t[0].FrobeniusSquare(t[1])
t[1].Conjugate(t[1])
t[1].Mul(t[1], t[2])
t[1].Mul(t[1], t[0])
m.Mul(m, t[1])
```

Optimizations

Arithmetic in cyclotomic groups

Table: Square in cyclotomic $\mathbb{F}_{p^{12}}$

	Compress	Square	Decompress
Normal	0	36	0
Granger-Scott [GS10]	0	18	0
Karabina [Kar13] SQR2345	0	12	19
Karabina [Kar13] SQR12345	0	15	8
Torus ($\mathbb{T}_2$)[RS03]	24	24	48

- 1 or 2 squarings $\implies$ Granger-Scott
- 3 squarings $\implies$ Karabina SQR12345
- $\geq$ 4 squarings $\implies$ Karabina SQR2345

Table: Mul in cyclotomic $\mathbb{F}_{p^{12}}$

	Compress	Multiply	Decompress
Normal	0	54	0
Torus ($\mathbb{T}_2$)[RS03]	24	42	48

- Compression/Decompression only once!
- Whole final exp. in compressed form over $\mathbb{F}_{p^6}$
- Better:
  - Absorb the compression in the easy part computation
  - Do we really need decompression?

Definition

Let $\mathbb{F}_q$ be a finite field and $\mathbb{F}_{q^k}$ a field extension of $\mathbb{F}_q$. Then the norm of an element $\alpha \in \mathbb{F}_{q^k}$ with respect to $\mathbb{F}_q$ is defined as the product of all conjugates of α over $\mathbb{F}_q$, namely $N_{\mathbb{F}_{q^k}/\mathbb{F}_q} = \alpha\alpha^q \cdots \alpha^{q^{k-1}} = \alpha^{(q^k-1)/(q-1)}$

$$T_k(\mathbb{F}_q) = \bigcap_{\mathbb{F}_q \subset F \subset \mathbb{F}_{q^k}} ker(N_{\mathbb{F}_{q^k}/F})$$

Lemma

Let $\alpha \in \mathbb{F}_{q^k}$, then $\alpha^{(q^k-1)/\Phi_k(q)} \in T_k(\mathbb{F}_q)$

$\mathbb{T}_2$ cryptosystem introduced by Rubin and Silverberg [RS03].
Let $\alpha = c_0 + \omega c_1 \in \mathbb{F}_{q^k} - \{1, -1\}$ (cyclotomic subgroup), we have

compress $f(\alpha) = (1 + c_0)/c_1 = \beta \in \mathbb{F}_{q^{k/2}}$

decompress $f^{-1}(\beta) = (\beta + \omega)/(\beta - \omega) = \alpha$

Mul $\beta_1 \times \beta_2 = (\beta_1\beta_2 + \omega)/(\beta_1 + \beta_2)$

Square $\beta^2 = \frac{1}{2}(\beta + \omega/\beta)$

Inverse $1/\beta = -\beta$

$\mathbb{T}_2$ **arithmetic is R1CS-friendly!**

Easy part: $m^{(q^{12}-1)/\Phi_k(p)} = m^{(p^6-1)(p^2+1)}$

Let $\alpha = c_0 + \omega c_1 \in \mathbb{F}_{q^{12}} - \{1\}$ (cyclotomic subgroup),

$$\begin{aligned}
\alpha^{p^6-1} &= (c_0 + \omega c_1)^{p^6-1} \\
&= (c_0 + \omega c_1)^{p^6} / (c_0 + \omega c_1) \\
&= (c_0 - \omega c_1)/(c_0 + \omega c_1) \\
&= (-c_0/c_1 + \omega)/(-c_0/c_1 - \omega) \\
f(\alpha) &= (-c_0/c_1)^{p^2+1} \\
&= (-c_0/c_1)^{p^2} \times (-c_0/c_1)
\end{aligned}$$

$\rightarrow$ 60C

Conclusion

Implementation open-sourced (MIT/Apache-2.0) at
https://github.com/ConsenSys/gnark
e.g. For BLS12-377,

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

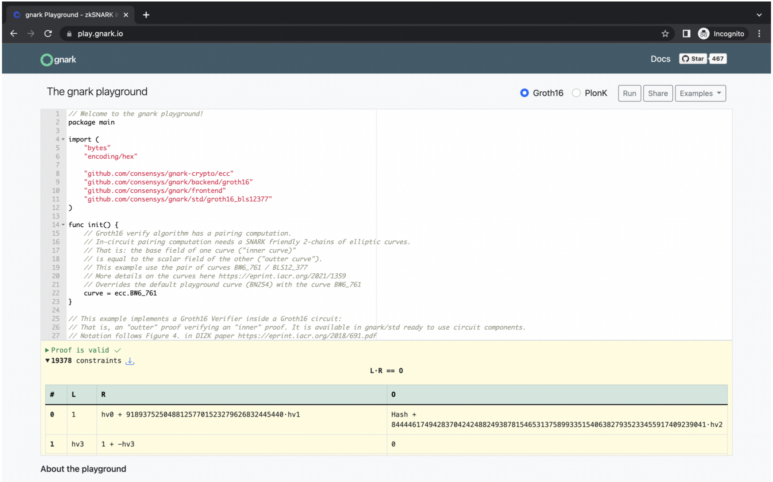
For BLS24-315, a pairing is **27608** contraints .
More optimizations in mind:

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

Conclusion

Let's play with gnark!

https://play.gnark.io/



References I

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.

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 III

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.

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.

References IV

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.

References V

Youssef El Housni and Aurore Guillevic.
Families of SNARK-friendly 2-chains of elliptic curves.
In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022*, volume 13276 of *LNCS*, pages 367–396. Springer, 2022.
ePrint 2021/1359.

Kirsten Eisenträger, Kristin Lauter, and Peter L. Montgomery.
Fast elliptic curve arithmetic and improved Weil pairing evaluation.
In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 343–354. Springer, Heidelberg, April 2003.

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.

References VI

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.

Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru.
PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge.
ePrint 2019/953.

References VII

Daiki Hayashida, Kenichiro Hayasaka, and Tadanori Teruya.
Efficient final exponentiation via cyclotomic structure for pairings over families of elliptic curves.
ePrint 2020/875.

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://ia.cr/2022/367.

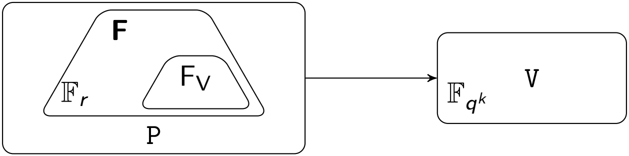
References VIII

Karl Rubin and Alice Silverberg.
Torus-based cryptography.
In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 349–365. Springer, Heidelberg, August 2003.

Michael Scott.
Pairing implementation revisited, 2019.

Pairings in SNARKs

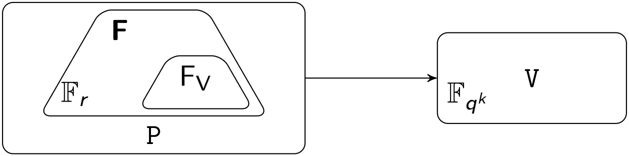
An arithmetic mismatch



- **F** any program is expressed in $\mathbb{F}_r$
- P proving is performed over $\mathbb{F}_r[X]$ and $\mathbb{G}_1$ (and $\mathbb{G}_2$)
- V verification (eq. 1, 2 and 3) is done in $\mathbb{F}^*_{q^k}$
- F_V programs of V are natively expressed in $\mathbb{F}^*_{q^k}$ not $\mathbb{F}_r$

Pairings in SNARKs

An arithmetic mismatch



- **F** any program is expressed in $\mathbb{F}_r$
- P proving is performed over $\mathbb{F}_r[X]$ and $\mathbb{G}_1$ (and $\mathbb{G}_2$)
- V verification (eq. 1, 2 and 3) is done in $\mathbb{F}^*_{q^k}$
- F_V programs of V are 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, BCTV14, BCG+20]

Pairings in SNARKs: solutions

A *cycle* of elliptic curves:

$$\begin{tikzcd} E_2(\mathbb{F}_{p_2}) \arrow[d, bend left, "\#E_1(\mathbb{F}_{p_1}) = p_2"] \\ E_1(\mathbb{F}_{p_1}) \arrow[u, bend left, "\#E_2(\mathbb{F}_{p_2}) = p_1"] \end{tikzcd}$$

A *2-chain* of elliptic curves:

$$\begin{tikzcd} E_2(\mathbb{F}_{p_2}) \\ E_1(\mathbb{F}_{p_1}) \arrow[u, "\#E_2(\mathbb{F}_{p_2}) = h \cdot p_1"'] \end{tikzcd}$$

Pairings in SNARKs: 2-chains

Eurocrypt 2022 [EG22]

Groth16

$$\begin{tikzcd} \textbf{BW6-761} \\ \textbf{BLS12-377} \arrow[u] \end{tikzcd}$$

KZG

$$\begin{tikzcd} \textbf{BW6-633 (or BW6-672)} \\ \textbf{BLS24-315} \arrow[u] \end{tikzcd}$$