With my co-authors Aurore Guillevic and Diego F. Aranha, we published on ePrint mid-May a survey of elliptic curves for proof systems (ePrint 2022/586). Recently, there have been many tailored constructions of these curves that aim at efficiently implementing different kinds of proof systems. In that survey we provide the reader with a comprehensive view on existing work and revisit the contributions in terms of efficiency and security. We present an overview at three stages of the process:

1. curves to instantiate a SNARK,
2. curves to instantiate a recursive SNARK, and
3. curves to express an elliptic-curve related statement.

In this post, we are only interested in the first case. For pairing-based SNARKs, such as the circuit-specific Groth16 [1], bilinear pairings (e:\mathbb{G}_1\times \mathbb{G}_2 \rightarrow \mathbb{G}_T

) are needed for the proof verification. For universal SNARKs, such as PLONK [2] or Marlin [3], a polynomial commitment scheme is needed. An example of an efficient one is the Kate-Zaverucha-Goldberg (KZG) scheme [4]. The verifier needs to verify some polynomial openings using bilinear pairings.

So far, for both circuit-specific and universal constructions, the verification algorithms boil down mainly to pairing computations. However, the prover work is not the same. For Groth16, it is mainly multi-scalar-multiplications (MSM) in both

\mathbb{G}_1

and \mathbb{G}_2

while for KZG-based schemes it is mainly MSMs in \mathbb{G}_1

only

. In section 4 of the survey paper, we look at a particular family of pairing-friendly elliptic curves suitable for KZG. We derive algorithm 6 to tailor this family to the SNARK context. Note that KZG is also useful for other compelling applications such as Verkle trees for blockchain stateless clients.

Pairing-friendly curves

A pairing is a non-degenrate bilinear map

e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T

(P,Q) \mapsto e(P,Q)

where

- \mathbb{G}_1

and \mathbb{G}_2

are groups of order r

defined over elliptic curves (with generators G_1

and resp. G_2

) and

- \mathbb{G}_T

is also a group of order r

defined over a finite field extension.

We usually deal with type-3 pairing where

- \mathbb{G}_1

is defined over a curve E(\mathbb{F}_p)

, \mathbb{G}_2

over E(\mathbb{F}_{p^k})

and

- \mathbb{G}_T

over \mathbb{F}_{p^k}

.

The integer k

is defined as the smallest integer such that r \mid p^k-1

and \mathbb{G}_2

coordinates can be sometimes compressed into a smaller field because \mathbb{G}_2

can be isomorphic to the r

-torsion on a different curve E'(\mathbb{F}_{p^{k/d}})

for some d

(the twist degree) that characterizes the curve E

.

[

pairing-fig2

3381×1252 135 KB

](https://ethresear.ch/uploads/default/original/2X/1/15745c5ea6afc3faa216fc1d277b14c2a609ee25.jpeg)

A pairing-friendly curve construction aims at generating a curve with k

not too big (for efficient arithmetic in \mathbb{G}_2

and \mathbb{G}_T

) and not too small (for hard discrete logarithm DLP in \mathbb{G}_T

). The widely used BLS12-381 curve [5], is defined over a field \mathbb{F}_p

of size 381 bits and has a prime subgroup order r

of 255 bits. It has k=12

so \mathbb{G}_T

is over \mathbb{F}_{p^{12}}

and d=6

so \mathbb{G}_2

is over \mathbb{F}_{p^2}

. This curve is derived from the Barreto-Lynn-Scott of embedding degree 12 (BLS12) family and aims at optimizing the computations in all fronts:

- \mathbb{F}_r

: size is 255 bits (1 spare-bit and hard DL)

- \mathbb{G}_1

: coordinates over a 381-bit field

- \mathbb{G}_2

: coordinates over a 762-bit field (381\times2

) and \mathbb{F}_{p^2}[u]

is constructed using the irreducible polynomial u^2+1

(p \equiv 3 \mod 4

)

- \mathbb{G}_T

and pairing: elements over a 4572-bit field and the Hamming weight of the 64-bit Miller loop size (the curve seed for BLS) is just 6.

In some applications as we will see in KZG, not all fronts should be optimized to the same extent. One can for example reduce the size of the \mathbb{G}_1

at the cost of the \mathbb{G}_2

size.

KZG polynomial commitment

A polynomial commitment scheme allows to commit to a polynomial and then open it at any point (showing that the value of the polynomial at a point is equal to a claimed value, i.e.

p(z)=y

).

The protocol

- (vk,pk)\leftarrow\texttt{setup}(\tau, 1^\lambda):

For some security parameter \lambda

, sample randomly \tau \leftarrow \mathbb{F}_r

and then compute pk=\tau^iG_1

for i\in{1,\dots,m}

and vk=\tau G_2

.

- C\leftarrow\texttt{commit}(pk,p):

given the polynomial p(x)\in \mathbb{F}_r[x]

and the proving key pk

, compute C=p(\tau)\cdot G_1=\sum_{i=0}^{n<m}p_i\cdot\tau^iG_1

- \pi\leftarrow\texttt{open}(p, y, z):

to prove that p(z)=y

, compute the polynomial q(x)=(p(x)-y)/(x-z)

and then the proof \pi=q(\tau)\cdot G_1

- 0/1\leftarrow\texttt{verify}(C, \pi, vk):

compute P_z=zG_2

and P_y=yG_1

, and then verify that e(\pi, vk-P_z)=e(C-P_y,G_2)

\rightarrow

correctnes:

\begin{align*} e(\pi, vk-P_z)&=e(C-P_y,G_2)\\ e(q(\tau)\cdot G_1, \tau G_2-z G_2)&=e(p(\tau)\cdot G_1-y G_1,G_2)\\ e(G_1,

G_2)^{q(\tau)(\tau -z)}&=e(G_1,G_2)^{p(\tau)-y} \ X_{\mathbb{G}T}^{q(\tau)(\tau -z)}&=X{\mathbb{G}_T}^{p(\tau)-y} \end{align}

KZG-friendly curves

So far, we have seen that the KZG commitment is a MSM computation in \mathbb{G}_1

only

. Thus, a KZG-friendly curve should be a pairing-friendly curve where \mathbb{G}_1

is the smallest possible, the pairing fast and \mathbb{G}_2

can be big though. The BLS24 family is suitable for these goals because one can reduce \mathbb{F}_p

size (w.r.t. DLP in \mathbb{F}_{p^{24}}

) and still have a fast pairing (curve seed is 32 bits, half the size of BLS12). However, for SNARKs, we need \mathbb{F}_r

to be highly 2-adic, i.e. 2^{large} \mid r-1

to be able to implement efficient polynomial arithmetic using radix-2 Fast Fourier Transforms (FFT).

However, as mentioned in this GitHub thread[6], there are only 2^{32}

possible seeds that fit lead to r < 256

bits and by enumerating these the best high 2-adicity achieved is 22. In the survey paper, we propose a different Hensel-like technique for lifting multiple roots and thus increasing the 2-adicity. With this technique (Alg. 6), we found a particular BLS24 curve of 2-adicity 60. It has similar properties to BLS12-381 but is tailored for KZG-based SNARKs.

BLS24-317

curve

seed x

2-adicity

r = #\mathbb{G}_1

p, \mathbb{G}_1

p^{k/d}, \mathbb{G}_2

p \equiv 3 \mod 4

security

BLS24-317

0xd9018000 (HW=6)

60

255

317

1268

Yes

127

BLS12-381

-0xd201000000010000 (HW=6)

32

255

381

762

Yes

126

Implementation

I implemented this curve in gnark-crypto and Diego implemented it in Relic. Hereafter, I benchmark KZG operations (gnark-crypto) over BLS24-317 and BLS12-381 curves on a AMD EPYC 7R32 AWS machine.

benchmark BLS12-381 ns/op BLS24-317 ns/op delta BenchmarkKZGCommit-32 30658007 23818500 -22.31% BenchmarkKZGOpen-32 32785660 25866237 -21.11% BenchmarkKZGVerify-32 1411429 3379792 +139.46% BenchmarkKZGBatchVerify10-32 1829747 3783824 +106.79%

We see that the commitments and openings are ~20% faster on the new BLS24-317 while the verification is way slower but still acceptable (3.3 ms vs. 1.4 ms). This is due to the cost of \mathbb{F}_{p^{24}}

in the pairing computation. However, this can be somewhat reduced following[7], which we have not implemented in gnark-crypto yet.

Conclusion

For KZG-based applications that want to speed up the commitment (and opening) parts of the computations (e.g. PLONK or Marlin prover), it seems that the BLS24-317 curve is faster than the well-optimized BLS12-381 curve. However, this comes at the cost of a slower verification although, in many applications, the timings are acceptable (e.g. 3.7 ms for a batch verification of 10 commitments). It was also believed that BLS24 are incompatible with high 2-adicity in \mathbb{F}_r

which we proved it is not.

References

[1] https://eprint.iacr.org/2016/260.pdf

[2] https://eprint.iacr.org/2019/953.pdf

[3] https://eprint.iacr.org/2019/1047.pdf

[4] https://www.iacr.org/archive/asiacrypt2010/6477178/6477178.pdf

[5] BLS12-381 For The Rest Of Us - HackMD

[6] Implement BLS24-319 Curve · Issue #10 · arkworks-rs/curves · GitHub

[7] https://eprint.iacr.org/2010/104.pdf

Author: Youssef El Housni.

I'm part of a research team at ConsenSys. If you are interested in our work (fast finite field arithmetic, elliptic curves, pairings, and zero-knowledge proofs), give us a shout.