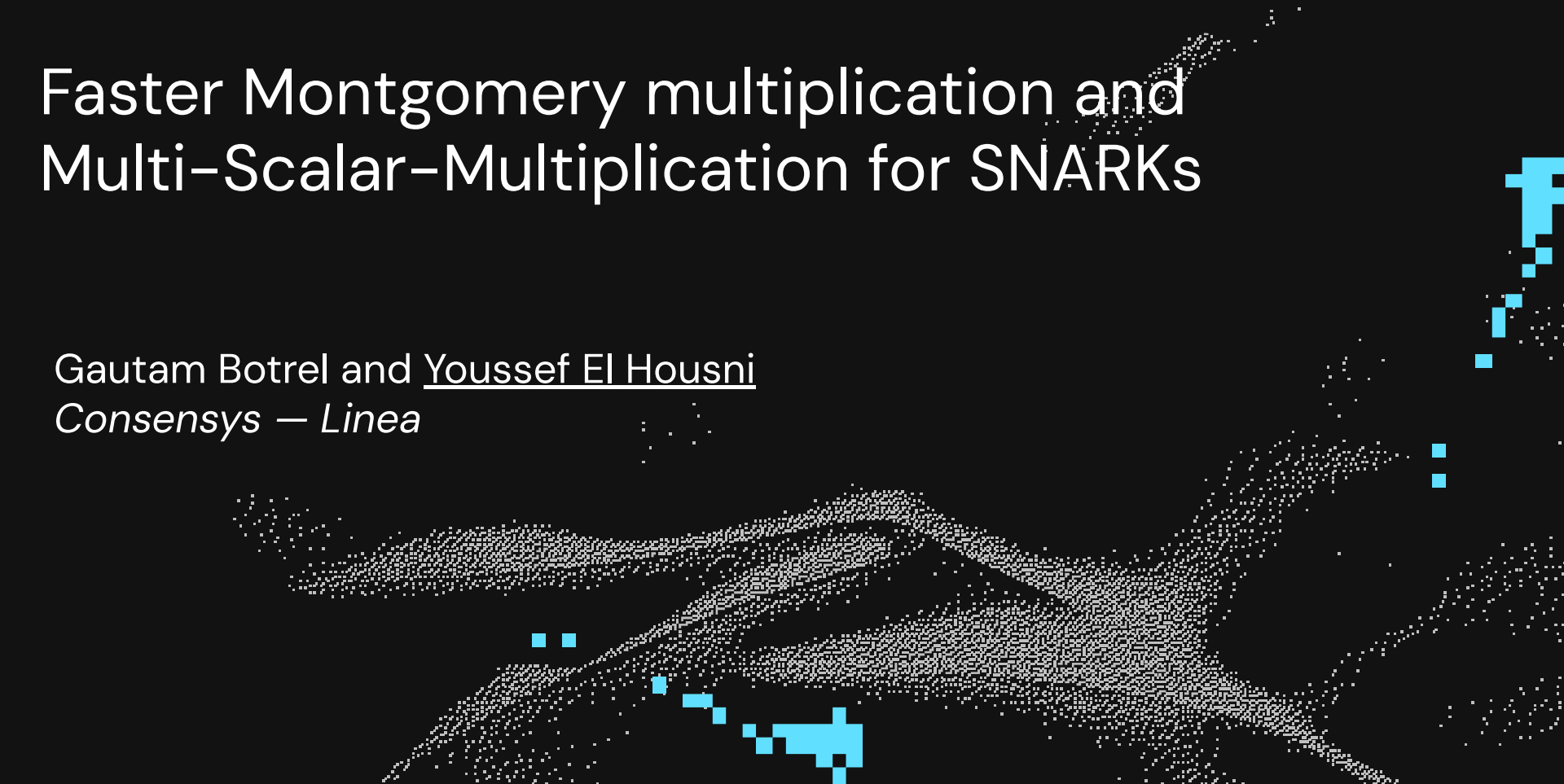
Faster Montgomery multiplication and Multi-Scalar-Multiplication for SNARKs

Gautam Botrel and Youssef El Housni
Consensys — Linea



Faster Montgomery multiplication and Multi-Scalar-Multiplication for SNARKs

Gautam Botrel and Youssef El Housni
Consensys — Linea

Agenda

01 SNARKs
02 Multi-Scalar-Multiplication
03 Faster Montgomery multiplication

SNARK

Succinct Non-Interactive ARgument of Knowledge

"I have a sound, complete and zero-knowledge proof that a statement is true". [GMR85]

- **Sound**

  False statement ⇒ cheating prover cannot convince honest verifier.

- **Complete**

  True statement ⇒ honest prover convinces honest verifier.

- **Zero-knowledge**

  True statement ⇒ verifier learns nothing other than statement is true. 1

"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.

(zk) SNARK

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 : $(pk, vk) \leftarrow S(F, \lambda)$
- Prove : $\pi \leftarrow P(x, z, w, pk)$
- Verify : false/true $\leftarrow V(x, z, \pi, vk)$

SNARK

Pairing-friendly elliptic curves

- E : y^2 = x^3 + ax + b elliptic curve defined over Fq, q a prime.
- r prime divisor of #E(Fq) = q + 1 - t,
- t Frobenius trace.
- small embedding degree k: smallest integer k ∈ N s.t. r | q^(k - 1).
- G1 ⊂ E(Fq) and G2 ⊂ E(Fq^k) two groups of order r.
- GT ⊂ Fq^k group of r-th roots of unity.
- pairing e : G1 × G2 → GT .

Curves of interest:

y^2 = x^3 + 1

SNARK

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

Multi-Scalar-Multiplication

S = [a1]P1 + [a2]P2 + · · · + [an]Pn with Pi ∈ G1 (or G2) and ai ∈ Fr (|r| =b−bit)

- Step 1: reduce the b-bit MSM to several c-bit MSMs for some chosen fixed c ≤ 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

S = [a1]P1 + [a2]P2 + · · · + [an]Pn with Pi ∈ G1 (or G2) and ai ∈ Fr (|r| =b−bit)

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

Overall cost is: **b/c(n + 2^c) + (b − c − b/c − 1)**

- Mixed re-additions: to accumulate Pi in the c-bit MSM buckets with cost **b/c(n − 2^c + 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

Multi-Scalar-Multiplication

- Curves of interest have always a twisted Edwards form −y^2 + x^2 = 1 + dx^2y^2
- We introduce a custom coordinates system (y − x : y + x : 2dxy) → (7m per addition)
- We use 2-NAF buckets, Parallelism, software optimizations...

40-47% speedup compared to artworks (Rust)

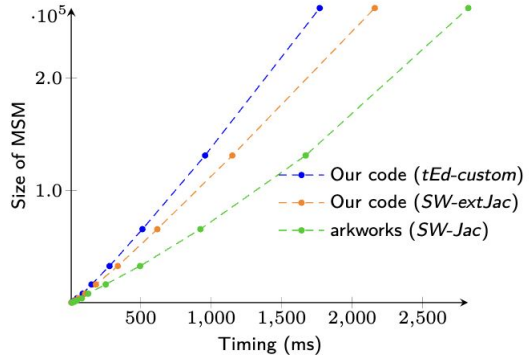


Figure 3: Comparison of our MSM code and the `arkworks` one for different instances on the BLS12-377 $\mathbb{G}_1$ group on the Samsung Galaxy A13.

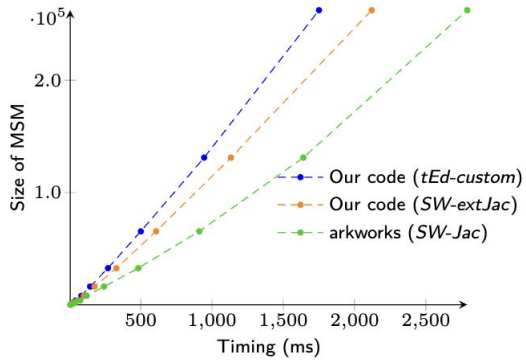


Figure 4: Comparison of our MSM code and the `arkworks` one for different instances on the BLS12-377 $\mathbb{G}_1$ group on the `x86` AWS machine.

Faster Montgomery multiplication

Given integers a, b and q the modular multiplication problem is to compute the remainder of the product

a x b mod q

- SNARKs invoke modular multiplication billions of times in a single execution.
- Other protocols (such as ECDSA or ECDH) are themselves invoked billions of times each second around the world.

In both cases, even a tiny improvement in modular multiplication yields a very significant computational saving—every nanosecond counts.

Faster Montgomery multiplication

Modular multiplication without division

- Do not compute **ab mod q**
- Instead compute **ab / R mod q** for some carefully chosen number R
- For example, if |p|=381 bits, R=2^(6x64)=2^384 (on a 64-bit architecture)
- Store a and b in the Montgomery form:

$$a' = aR \bmod q$$
$$b' = bR \bmod q$$

- Multiplication is

$$(aR)(bR) / R = abR \bmod q$$
$$a'b' / R = (ab)' \bmod q$$

Faster Montgomery multiplication

CIOS variant

- N the number of machine words to store q
- D the word size e.g. 2^64
- a'[i], b'[i], q[i] are the i-th words of a', b' and q
- q[0] is the lowest is lowest word of 1/q mod R
- t an array of size N+2
- C, S are machine words e.g. (C, S)=(high, low)

Cost: **4N^2+4N+2** integer additions and **2N^2+N** integer multiplications

Faster Montgomery multiplication

Our optimization

Proposition 1.
If the highest word of p is at most (D-1)/2 - 1, then the variables t[N] and t[N+1] always store the value 0 at the beginning of each iteration of the outer loop.

Cost: **4N^2** - **N**, a saving of **5N + 2** additions.

This optimization can be used whenever the highest bit of the modulus is zero (and not all of the remaining bits are set).

5-10% speedup depending on N. For N=6 we measure 8%.

Thank you

- Paper: https://tches.iacr.org/index.php/TCHES/article/view/10972/10279
- Code: https://github.com/gbotrel/zprize-mobile-harness
- Winner of the https://www.zprize.io/ mobile competition