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 \( \lambda \):

- Setup : \((pk, vk) \leftarrow S(F, \lambda )\)
- Prove : \(\pi \leftarrow P(x, z, w, pk)\)
- Verify : false/true \(\leftarrow V(x, z, \pi, vk)\)
Pairing-friendly elliptic curves

- $E : y^2 = x^3 + ax + b$ elliptic curve defined over $F_q$, $q$ a prime.
- $r$ prime divisor of $\#E(F_q) = q + 1 - t$,
- $t$ Frobenius trace.
- small embedding degree $k$: smallest integer $k \in \mathbb{N}$ s.t. $r | q^{(k - 1)}$.
- $G_1 \subset E(F_q)$ and $G_2 \subset E(F_q^k)$ two groups of order $r$.
- $GT \subset F_q^k$ group of $r$-th roots of unity.
- pairing $e : G_1 \times G_2 \rightarrow GT$.

Curves of interest:

$y^2 = x^3 + 1$
## SNARK

**Table:** Cost of $S$, $P$ and $V$ algorithms for Groth16 and Universal. $n =$ number of multiplication gates, $a =$ number of addition gates and $\ell =$ number of public inputs. $M_G =$ multiplication in $G$ and $P =$ pairing.

<table>
<thead>
<tr>
<th></th>
<th>Setup</th>
<th>Prove</th>
<th>Verify</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groth16</td>
<td>$3n \ M_{G_1}, n \ M_{G_2}$</td>
<td>$(4n - \ell) \ M_{G_1}, n \ M_{G_2}$</td>
<td>$3 \ P, \ell \ M_{G_1}$</td>
</tr>
<tr>
<td>Universal (PLONK-KZG)</td>
<td>$d_{\geq n+a} \ M_{G_1}, 1 \ M_{G_2}$</td>
<td>$9(n+a) \ M_{G_1}$</td>
<td>$2 \ P, 18 \ M_{G_1}$</td>
</tr>
</tbody>
</table>
Multi-Scalar-Multiplication

\[ S = [a_1]P_1 + [a_2]P_2 + \cdots + [a_n]P_n \text{ with } P_i \in G_1 \text{ (or } G_2) \text{ and } a_i \in F_r \ (|r| = b\text{-bit}) \]

- 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

\[ S = [a_1]P_1 + [a_2]P_2 + \cdots + [a_n]P_n \] with \( P_i \in G_1 \) (or \( G_2 \)) and \( a_i \in \mathbb{F}_r \) (\(|r| = b\)-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

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

- Mixed re-additions: to accumulate \( P_i \) in the \( c \)-bit MSM buckets with cost \( \frac{b}{c}(n - 2^c + 1) \)
- Additions: to combine the bucket sums with cost \( \frac{b}{c}(2^c - 3) \)
- Additions and doublings: to combine the \( c \)-bit MSMs into the \( b \)-bit MSM with cost \( b - c + \frac{b}{c} - 1 \)
  - \( \frac{b}{c} - 1 \) 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 $(y - x : y + x : 2dxy) \rightarrow (7m \text{ per addition})$
- 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 $G_1$ group on the Samsung Galaxy A13.

**Figure 4:** Comparison of our MSM code and the **arkworks** one for different instances on the BLS12-377 $G_1$ group on the x86 AWS machine.
Given integers $a$, $b$ and $q$ the modular multiplication problem is to compute the remainder of the product $a \times b \mod q$.

- 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, \(R=2^{(6\times64)}=2^{384}\) (on a 64-bit architecture)
- Store \(a\) and \(b\) in the Montgomery form:
  \[
  a' = aR \mod q \\
  b' = bR \mod q
  \]
- Multiplication is
  \[
  (aR)(bR) / R = abR \mod q \\
  a'b' / R = (ab)' \mod q
  \]
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

- Code: https://github.com/gbotrel/zprize-mobile-harness
- Winner of the https://www.zprize.io/ mobile competition