On the Optimization of Pippenger's Bucket Method with Precomputation

Guiwen Luo, Guang Gong

{g27luo, ggong}@uwaterloo.ca

Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L3G1, CANADA

Problem

- Multi-scalar Multiplication (MSM) over fixed points:
$$S_{n,r} = a_1P_1 + a_2P_2 + \dots + a_nP_n, \quad (1)$$
where $0 \le a_i < r$ and P_i are fixed points in elliptic curve group.
- How can we compute it efficiently for large n : $n \ge 2^{16}$?

Motivation

- MSM dominates the time consumption in pairing-based zero-knowledge succinct non-interactive argument of knowledge (zkSNARK) schemes.
- Circuit size in Zcash: for single SHA-256 hash, the number of multiplication gates is about 23 thousands; for nested hash, several millions.

credit: `https://en.wikipedia.org/wiki/Zcash`

Pippenger's bucket method example

- Example:
$$\begin{aligned} S_{13,8} = &2P_1 + 3P_2 + 7P_3 + 6P_4 + 5P_5 + 1P_6 + 3P_7 \\ &+6P_8 + 2P_9 + 7P_{10} + 1P_{11} + 4P_{12} + 5P_{13}. \end{aligned}$$
- All points are sorted into 7 buckets with respect to the scalars $\{1, 2, \cdots, 7\}$:
$$\begin{aligned} S_{13,8} =&1(P_6 + P_{11}) + 2(P_1 + P_9) + 3(P_2 + P_7) + 4P_{12} \\ &+ 5(P_5 + P_{13}) + 6(P_4 + P_8) + 7(P_3 + P_{10}) \\ :=&1S_1 + 2S_2 + \dots + 7S_7. \end{aligned}$$
- The accumulated sum $\Sigma_{i=1}^{7} iS_i$ can be computed via
$$\begin{aligned} &S_7 \\ &+ (S_7 + S_6) \\ &+ (S_7 + S_6 + S_5) \\ &\cdots \\ &+ (S_7 + S_6 + S_5 + \cdots + S_1). \end{aligned}$$
- $\{S_i\}$: $13 - 7 = 6$ additions,
$\Sigma_{i=1}^{7} iS_i$: $2 \times 6 = 12$ additions.
In total, 18 additions.

Pippenger's bucket method variant

- If r is small enough, all n points are sorted into $r-1$ buckets according to the scalars.
- Then MSM
$$\begin{aligned} S_{n,r} =&1S_1 + 2S_2 + \dots + (r-1)S_{r-1} \\ =&S_{r-1} + (S_{r-1} + S_{r-2}) + \cdots \\ &+ (S_{r-1} + S_{r-2} + \cdots + S_1). \end{aligned}$$
- S_i's: $n - r + 1$ additions,
$\Sigma_{i=1}^{r-1} iS_i$: $2 \times (r-2)$ additions.
In total, $n + r - 3$ additions.
- If r is big (over BLS12-381 curve, $r \approx 2^{256}$), every scalar is decomposed into its q-ary form,
$$\begin{aligned} a_i &= a_{i,0} + a_{i,1}q + \cdots + a_{i,h-1}q^{h-1}, \\ S_{n,r} &= a_1P_1 + a_2P_2 + \cdots + a_nP_n \\ &= \Sigma_{i=1}^{n}\Sigma_{j=0}^{h-1} a_{ij} \cdot (q^jP_i), 0 \le a_{ij} < q, \\ &:= S_{nh,q}. \end{aligned}$$
- Precomputation (nh points):
$$\{q^jP_i \mid 1 \le i \le n,\ 0 \le j \le h-1\}.$$
- By aforementioned method, all points are sorted into $q-1$ buckets. In total, $S_{n,r}$ can be computed using
$$nh + q - 3$$
additions.

General framework

- Let us summarize the framework of computing MSM,
$$S_{n,r} = S_{hn,q} = \Sigma_{i=1}^{n}\Sigma_{j=0}^{h-1} a_{ij}q^jP_i,\ 0 \le a_i \le q.$$
- If $a_{ij} = m_{ij}b_{ij}$, $m_{ij} \in M$ (multiplier set), $b_{ij} \in B$ (bucket set), then
$$S_{hn,q} = \Sigma_{i=1}^{n}\Sigma_{j=0}^{h-1} b_{ij} \cdot (m_{ij}q^jP_i).$$
- Precompute ($nh|M|$ points)
$$\{mq^jP_i \mid 1 \le i \le n, 0 \le j \le h-1, m \in M\},$$
then it takes $\approx nh + |B|$ additions to compute $S_{n,r}$.
- Pippenger's bucket method,
$$M = \{1\},\ B = \{0, 1, 2, ..., q-1\}.$$
$S_{n,r}$ takes $\approx nh + q$ additions.
- Pippenger's bucket method variant (notice that $-P$ can be easily computed given P),
$$M = \{1, -1\},\ B = \{0, 1, 2, ..., \lceil q/2 \rceil\}.$$
$S_{n,r}$ takes $\approx nh + q/2$ additions.

New alogorithm

Goal: Construct bucket set B, *s.t.* $|B| \approx q/\ell$. It will yield an algorithm to compute $S_{n,r}$ using $\approx hn + q/\ell$ additions.

- Let q be a prime *s.t.*
2 is a primitive element in $\mathbb{F}_q$.
ℓ and h are small positive integers *s.t.*
$2^{\ell-1} < q$ and $q^{h-1} < r \le 2^{\ell-1}q^{h-1}$.
- The multiplier set is
$$M = \{2^i \mid 0 \le i \le \ell - 1\} \cup \{-1\},$$
- The corresponding bucket set ($|B| \approx q/\ell$) is
$$B = \{i \mid 0 \le i \le 2^{\ell-1}\} \cup \{2^{i \cdot \ell} \bmod q \mid 0 \le i \le \lfloor q/\ell \rfloor\}.$$
The idea behind the construction is that
$$\{i \mid 1 \le i \le q-1\} = \{2^i \bmod q \mid 0 \le i \le q-2\}.$$
- Note: Elements in the bucket set is no longer consecutive, a new accumulation algorithm is needed.

Further optimization

- Observation:
$$\begin{aligned} \{i \mid 1 \le i < q\} =&\{2^i \bmod q \mid 0 \le i \le q-2\} \\ =&\{2^i \bmod q \mid (3-q)/2 \le i \le (q-1)/2\}. \end{aligned}$$
Bucket set can be further reduced to ($|B| \approx q/(2\ell)$)
$$\begin{aligned} B =&\{i \mid 0 \le i \le 2^\ell\} \cup \\ &\{2^{i \cdot \ell} \bmod q \mid 0 \le i \le \lfloor (q-1)/2\ell \rfloor\}. \end{aligned}$$
- GLV endomorphism
$$\lambda P = \lambda \cdot (x, y) = (\xi x, y),\ \lambda^3 = 1 \in \mathbb{F}_r, \xi^3 = 1 \in \mathbb{F}_p,$$
where $\lambda \approx \sqrt{r}$, every scalar $a = a_0 + a_1\lambda$, so
$$S_{n,r} = S_{2n,\lambda}.$$
It further reduces the precomputation by a factor of 2.

Result

Conclusion: $S_{n,r}$ over fixed points can be computed using at most approximately
$$nh + q/(2\ell) \quad (2)$$
additions, with the help of ℓnh precomputed points
$$\{mq^jP_i \mid 1 \le i \le n, 0 \le j \le h-1, m \in \{1, 2, ..., 2^{\ell-1}\}\},$$
where $h = \lceil \log_q r \rceil$, and q is a prime selected to minimize the cost.

Instantiation

The instantiation is done over the elliptic curve group of order $r = 2^{256}$. Some comparisons against Pippenger's bucket method and its variant are presented in the following tables.

Comparison of methods that computes $S_{n,r}$

Method	Storage	Worst case cost
Pippenger [1]	$n \cdot P$	$h(n + q/2) \cdot A$
Pippenger variant [2]	$nh \cdot P$	$(nh + q/2) \cdot A$
Our algorithm	$\ell nh \cdot P$	$(nh + q/(2\ell)) \cdot A$

Radix q employed by different methods

n	Pippenger	Pippneger variant	Our method
2^{16}	2^{13}	2^{17}	$2^{18} - 5$
2^{17}	2^{14}	2^{18}	$2^{21} - 19$
2^{18}	2^{15}	2^{19}	$2^{21} - 19$
2^{19}	2^{17}	2^{20}	$2^{21} - 21$
2^{20}	2^{18}	2^{20}	$2^{23} - 21$

Number of additions taken to compute $S_{n,r}$

n	Pippenger	Pippenger variant	Our method
2^{16}	1.39×10^6	1.11×10^6	1.00×10^6
2^{17}	2.65×10^6	2.10×10^6	1.88×10^6
2^{18}	5.01×10^6	3.93×10^6	3.58×10^6
2^{19}	9.44×10^6	7.34×10^6	6.99×10^6
2^{20}	1.77×10^7	1.42×10^7	1.33×10^7

Acknowledgement

To be updated. This work is supported in part by xxx

References

[1] Daniel J Bernstein, Jeroen Doumen, Tanja Lange, and Jan-Jaap Oosterwijk. Faster batch forgery identification. In *International Conference on Cryptology in India*, pages 454–473. Springer, 2012.

[2] Ernest F Brickell, Daniel M Gordon, Kevin S McCurley, and David B Wilson. Fast exponentiation with precomputation: algorithms and lower bounds. *preprint, Mar*, 27, 1995.