MPHELL: Multi-Precision (Hyper) ELliptic curves Library

Titouan Coladon, Philippe Elbaz-Vincent, Cyril Hugounenq

Univ. Grenoble Alpes, IF, mphell@univ-grenoble-alpes.fr

This work is supported by ANR ARRAND (ANR-15-CE39-0002) and SECURIOT-2-AAP FUI 23.

Atelier PARI-GP, Grenoble, 24 January, 2020

financed by
IDEX Université Grenoble Alpes

This work is supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02)

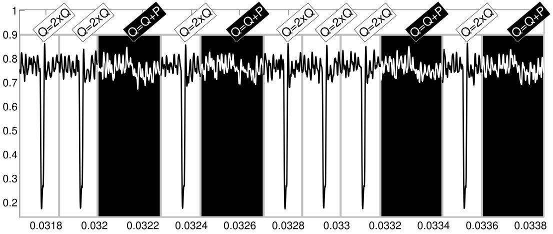
Requirements of our library

We want to address the need of a **fast** arithmetic library and possibly secured for Elliptic Curve Cryptography:

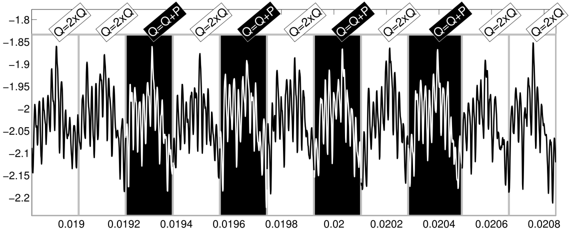
- Secure against **Simple Power Analysis (SPA)**;
- Easy to customize:
  - Customizable arithmetics (GMP, Intel IPPCP, MBedTLS);
  - Customizable curves;
  - Integration possible (PARIGP, PARITWINE, Demo. FIC 2020 de MbedTLS).
- Usable in industrial context:
  - microcontrollers (e.g., STM32);
  - ARM (32 bits and 64 bits);
  - Linux OS (32 bits and 64 bits).
- Competitive against other Elliptic Curve Cryptographic libraries.

The library has been designed with GNU/Linux systems as main targets (frequent on embedded systems) and for curves over prime fields.

SPA over a Weierstrass curve without protection against SPA



SPA over a Jacobi Quartic curve with protection against SPA



Existing libraries for ECC

There already exists libraries implementing elliptic curve arithmetics for cryptography such as:

- Intel IPPCP (fast, Intel architectures only, non resistant to SPA)
- MbedTLS, OpenSSL, LibreSSL, Libgcrypt, MIRACL, WolfSSL, libECC.
- NACL, libSodium usable only with hardcoded Edwards Elliptic Curves.

ELLIPTIC CURVE ARITHMETIC:

WEIERSTRASS: Y^2 = x^3 + a.X + b, using PROJECTIVE, JACOBIAN and COZ coordinates

TWISTED EDWARDS: a.X^2 + Y^2 = 1 + d.X^2.Y^2, using PROJECTIVE or EXTENDED coordinates

JACOBI QUARTIC: Y^2 = X^4 + 2.a.X^2 + 1, using EXTENDED coordinates

UNIFIED ADDITION is available for all these curves

FIELD ARITHMETIC:

Montgomery **Classic**

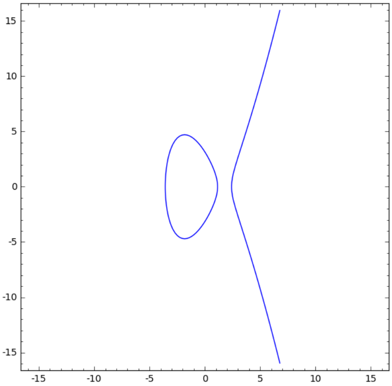
BIG NUMBER:

IPPCP **GMP** **MbedTLS**

Elliptic Curve formulae

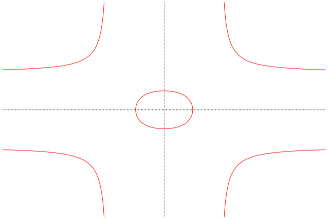
We propose for each type of elliptic curves implemented (Weierstrass, Jacobi Quartic, Edwards) two types of formulas:

- Dedicated arithmetic operations, and sliding windows multiplication to be fast when security **is not** required
- Unified arithmetic operations, to be protected against Simple Power Analysis when security **is** required.



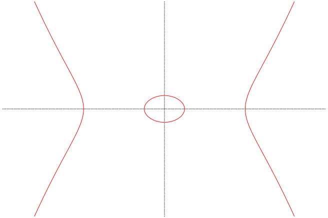
- PROJECTIVE: (X, Y, Z) matching the affine point (x, y) where $x = \frac{X}{Z}$, $y = \frac{Y}{Z}$, the neutral element is $(0, 1, 0)$.
- JACOBIAN: (X, Y, Z) matching the affine point (x, y) where $x = \frac{X}{Z^2}$, $y = \frac{Y}{Z^3}$, the neutral element is $(a^2, a^3, 0)$ with $a \neq 0$.

The Montgomery multiplication, using COZ arithmetic is used when unified arithmetic is required.



- PROJECTIVE: (X, Y, Z) matching the affine point (x, y) where $x = \frac{X}{Z}$, $y = \frac{Y}{Z}$, the neutral element is $(0, 1, 0)$.
- EXTENDED: (X, Y, T, Z) matching the affine point (x, y) where $x = \frac{X}{Z}$, $y = \frac{Y}{Z}$, and $T = \frac{XY}{Z}$ the neutral element is $(0, 1, 0, 1)$.

Jacobi's quartic



- EXTENDED: (X, Y, T, Z) satisfying $Y^2 = Z^2 + 2aX^2 + T^2$; $X^2 = ZT$; $a \neq 1$.
  Here $(X : Y : T : Z) = (sX : sY : T : sZ)$ for all nonzero s.

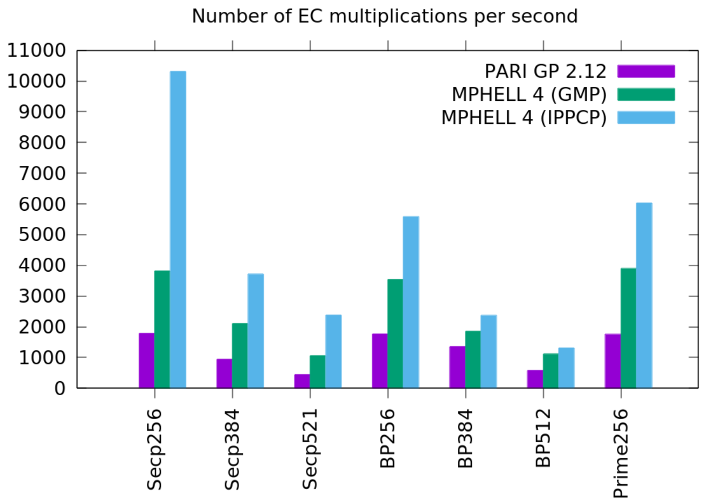
Curve conversions

- WEIERSTRASS $\leftrightarrow$ JACOBI QUARTIC, but you need θ such that $(\theta, 0)$ is a 2-torsion point on the Weierstrass elliptic curve
- WEIERSTRASS $\leftrightarrow$ TWISTED EDWARDS, but you need α, β such that $(\alpha, 0)$ is a 2-torsion point on the Weierstrass elliptic curve and that $3\alpha^2 + a_w = \beta^2$.

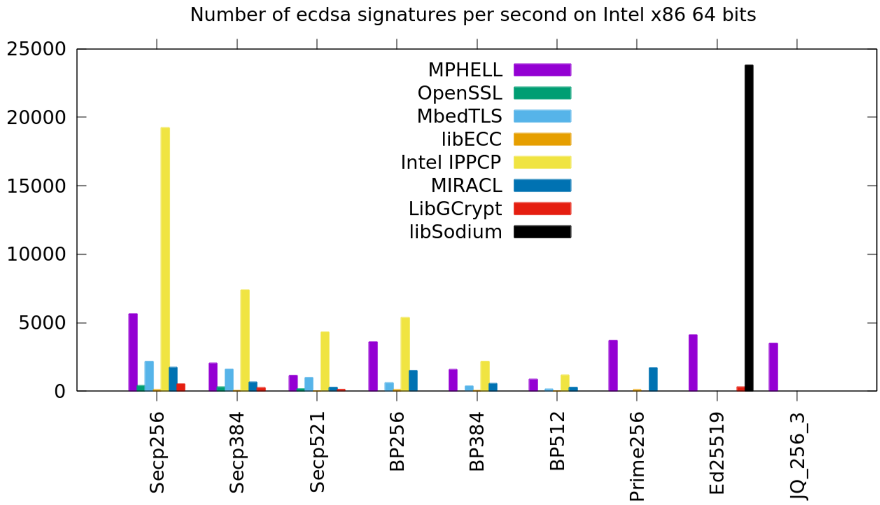
Hardcoded Curves

- Brainpool curves
- ANSSI (FR256v1)
- NIST Curves
- Ed25519
- Jq256 (Generated by us)

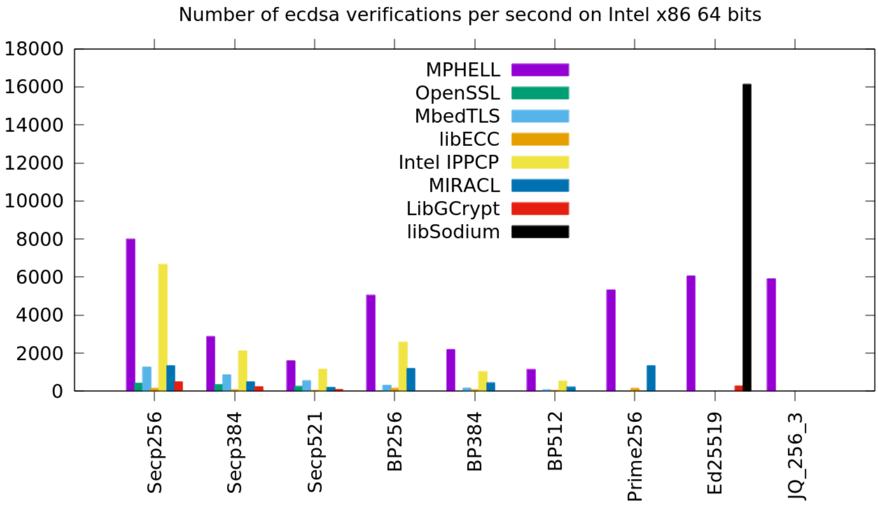
Performances (EC multiplications)



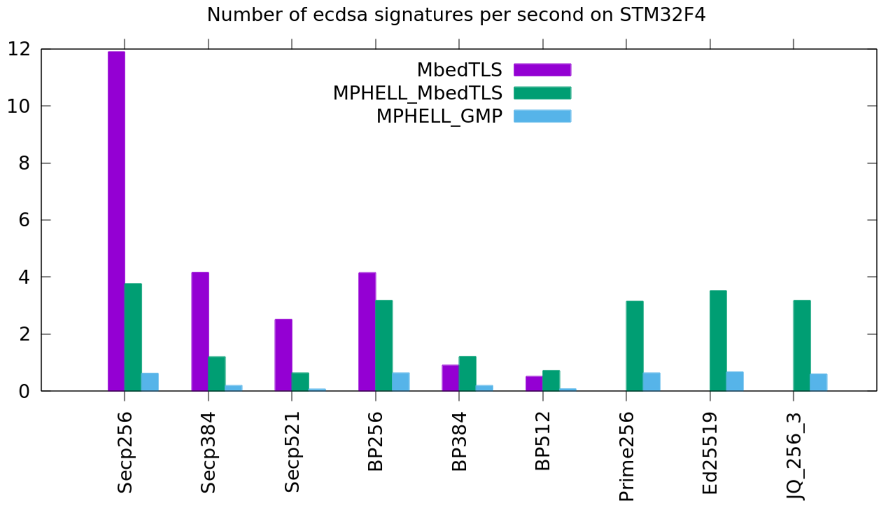
Performances (ECDSA Signatures)



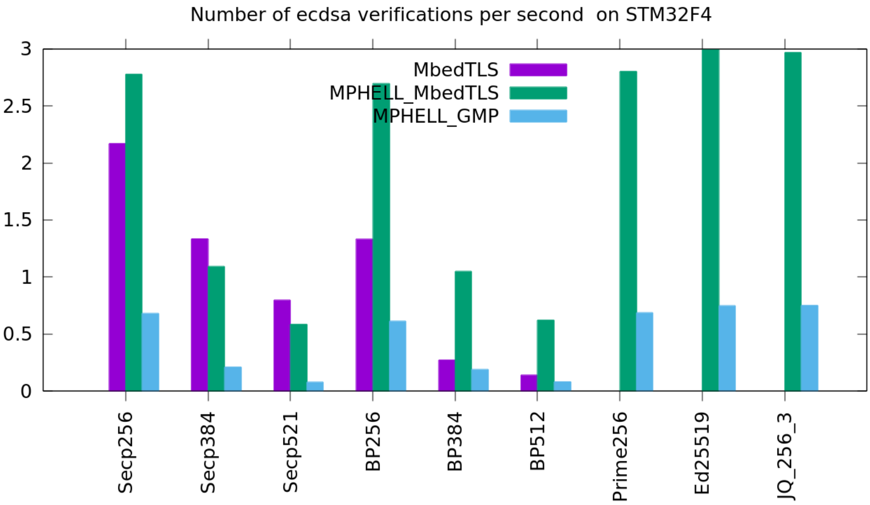
Performances (ECDSA Verifications)



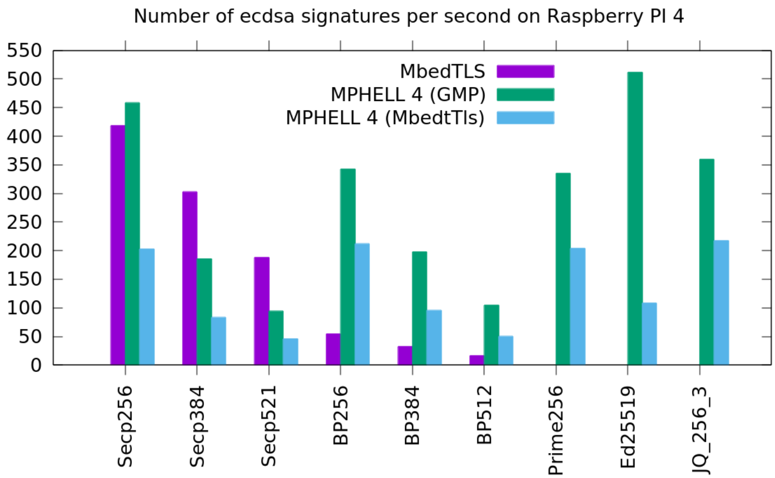
Performances (ECDSA Signatures)

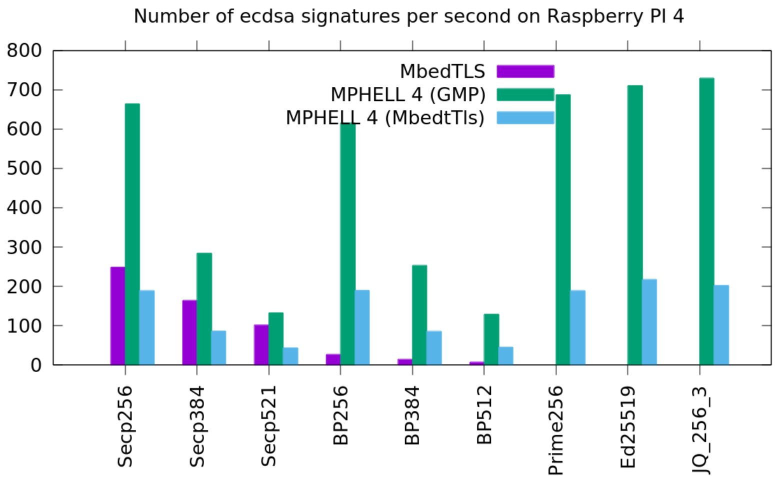


Performances (ECDSA Verifications)



Performances (ECDSA Signatures)




Number of ecdsa signatures per second on Raspberry PI 4
800
700
600
500
400
300
200
100
0
MbedTLS
MPHELL 4 (GMP)
MPHELL 4 (MbedtTls)
Secp256
Secp384
Secp521
BP256
BP384
BP512
Prime256
Ed25519
JQ_256_3

Conclusion

We present a new open source (LGPL3) elliptic curve library for cryptography

- suitable for embedded systems and industrial use (already tested with industrial partners);
- performant compared to other libraries;
- unified operations for providing SPA counter-measures.

Web site of MPHELL:

`https://www-fourier.univ-grenoble-alpes.fr/mphell/`

[BJL+15] Daniel J Bernstein, Simon Josefsson, Tanja Lange, Peter Schwabe, and Bo-Yin Yang.
Eddsa for more curves.
Cryptology ePrint Archive, 2015, 2015.

[BSI18] BSI.
Technical guideline BSI TR-03111 Elliptic Curve Cryptography of Bundesamt fur Sicherheit in der Informationstechnik, 2018.

[Cor16] Marie-Angela Cornélie.
Implantations et protections de mécanismes cryptographiques logiciels et matériels.
PhD thesis, Université Grenoble Alpes, 2016.

[Plû11] Jérôme Plût.
On various families of twisted jacobi quartics.
In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 373–383. Springer, 2011.

[Pon16] Simon Pontie.
Sécurisation matérielle pour la cryptographie à base de courbes elliptiques.
PhD thesis, Université Grenoble Alpes, 2016.

[Sch91] Claus-Peter Schnorr.
Efficient signature generation by smart cards.
Journal of cryptology, 4(3):161–174, 1991.

[Tea98] KCDSA Task Force Team.
The korean certificate-based digital signature algorithm.
IEEE P1363a, 1998.

ECDSA signing

The ECDSA signin of a message M with the hashing function H consists of:

- $k \xleftarrow{\$} \{1, n-1\}$,
- $Q = [k]B$,
- $r = x_Q \bmod n$ `if` $r == 0$ go to first step,
- $s = k^{-1}(r \cdot d + H(M)) \bmod n$ `if` $s == 0$ go to first step,
- return (r, s).

ECDSA Verifying

The ECDSA verification of a signed message $M, (r, s)$ with the hashing function H consists of:

- checking that $r, s \in \{1, n-1\}$,
- $u1 = s^{-1} \cdot H(M) \bmod n$,
- $u2 = s^{-1} \cdot r \bmod n$,
- $Q = [u1]B + [u2]P$,
- $v = x_Q$,
- return the boolean value of the test $v == r$.

Other standards of signing with elliptic curve do exist such as: `ECGDSA` [BSI18], `ECSDSA` [Sch91], `EdDSA` [BJL$^+$15], `ECKCDSA` [Tea98]. This list is non-exhaustive.