MPHELL library

Marie-Angela Cornelie ¹ *

¹Supervisor : Philippe Elbaz-Vincent
Univ. Grenoble Alpes, CNRS, Institut Fourier, UMR 5582, F-38402 St Martin d’Hères
{marie-angela.cornelie}@ujf-grenoble.fr

Ateliers PARI-GP 2016
MPHELL

- C multi-precision high performance library for the arithmetic of (hyper-)elliptic curves used in cryptography.
- Different coordinates systems are used and other libraries can be plugged into MPHELL (like PARI).
- Current development only concerns elliptic curves.
- Industrial perspectives.
- Release is expected for q1 of 2016.
Introduction

Different families of elliptic curves
- Equations
- Computation costs

MPHELL
- Overall architecture
- Functionalities
- Interaction with PARI

Conclusion
Different families of elliptic curves

We work with prime fields $\mathbb{F}_p$, $p \neq 2, 3$.

**Curves supported**

- **Weierstrass curves** (projective and jacobian coordinates)
  
  $E_{a,b} : y^2 = x^3 + ax + b$, $a, b \in \mathbb{F}_p$, $4a^3 + 27b^2 \neq 0$

- **Jacobi Quartic curves** (extended projective coordinates)
  
  $JQ_a : y^2 = z^2 + 2ax^2 + t^2$, $x^2 = zt$, $a \in \mathbb{F}_p$, $a \neq 1$.

- **Twisted Edwards curves** (extended projective coordinates)
  
  $Ed_{a,d} : ax^2 + y^2 = dt^2 + z^2$, $xy = zt$, $a, d \in \mathbb{F}_p$, $ad(a - d) \neq 0$. 
## Computation costs for prime curves

<table>
<thead>
<tr>
<th>Coordinates system</th>
<th>Doubling</th>
<th>General addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affine</td>
<td>1I,2M,2S</td>
<td>1I,2M,1S</td>
</tr>
<tr>
<td>Projective</td>
<td>7M,3S</td>
<td>12M,2S</td>
</tr>
<tr>
<td>Jacobian</td>
<td>4M,4S</td>
<td>12M,4S</td>
</tr>
<tr>
<td>Edwards</td>
<td>3M,4S</td>
<td>9M,1S,1D</td>
</tr>
<tr>
<td>Jacobi Quartic</td>
<td>2M,5S,1D</td>
<td>7M,3S,1D</td>
</tr>
</tbody>
</table>
Overall architecture

MPHELL

Random number generation
- Hash DRBG
- CTR DRBG
- Entropy
- Random number generation

Entropy

BigUint
- GMP
- PARI

Multi-precision arithmetic

Field arithmetic
- Fp
- Quadratic extension
- Cubic extension
- Weierstrass
- Jacobi Quartic
- Edwards
- Curve arithmetic

Edwards

MPHELL

Overall architecture

M-A. Cornelie (IF)
Functionalities (1)

Random number generation

- DRBG mechanisms as specified in SP800 90-A.
- Possibility to use others generators (including hardware generators)

Multi-precision arithmetic

- Multi-precision integers are represented as table of block.
- The base type used is uint64_t but the library can be configured with another block size like uint32_t.
- Some functions are implemented in assembly language (only supported for x86-64 architecture).
- Implementation on ARM V7 architecture is in progress.
Functionalities (2)

Field arithmetic
- Montgomery representation is used to implement modular function.
- Quadratic and cubic extensions of prime field are available.

Curve arithmetic
- Support for Weierstrass (projective and jacobian coordinates), Jacobi Quartic and Edwards curves (extended projective coordinates).
- Addition (+ CO-Z addition for Weierstrass curves), Doubling, different versions of scalar multiplication (Left-to-Right, Windowed,...)
Plug others libraries

- Our implementation with `big_uint`
- Other library types and functions

- `number` and `field_elt`

Wrapper
interaction with pari

- big_uint and t_INT types are very similar.
- Use the wrappers associated to the types number and field_elt.
- For all functions concerning multi-precision and modular arithmetic, use the PARI equivalent:
  ex: Fp_add for modular addition.
- The arithmetic of curves is completely independent of the library which is used for field arithmetic.
Exemple (1)

```c
#include "big_uint.h"
typedef big_uint number;
typedef big_uint_t number_t;
#define LIMB(x) x->limb
#define SIZ(x) x->size
#define SIGN(x) x->sign

#include <pari/pari.h>
typedef GEN number[1];
typedef GEN number_t;
#define LIMB(x) (*((x) + 2)
#define SIZ(x) (lg(*x) - 2)
#define SIGN(x) signe(*x) == 0 ? signe(*x) : 1
```

#endif

M-A. Cornelie (IF)
Exemple (2)

```c
int8_t
number_cmp (const number src1, const number src2)
{
#if USE_BIG_UINT == 1
    return u_cmp(src1, src2);
#elif USE_GMP == 1
    return mpz_cmp(src1, src2);
#else
    return cmpii(*src1, *src2);
#endif
}
```
Performances

Processor : Intel core i7-4790 3.4 Ghz
Compiler : gcc-5.3.0 bootstrapped
PARI 2.7.5 and GMP 6.1.0
Tests with NIST and Brainpool curves.

Arithmetic functions

- Slowdown of about 30% using GEN version instead of the big_uint one.
- The management of the stack can be responsible for this slowdown.
- Work is in progress to improve the performances.
Comparison with \texttt{ell*} functions

- using \texttt{big\_uint}: between 40 and 50% better.
- using \texttt{GEN}: between 20 and 30% better.

Comparison with \texttt{FpE*_} functions

- using \texttt{big\_uint}: between 20 and 30% better.
- using \texttt{GEN}: between 0 and 10% better.

Remarks

- The improvements are better for addition and doubling than for scalar multiplication because only the naive implementation has been tested.
- We expected an increase with the other scalar multiplication implemented.
- The best results are obtained on large curves like NIST-384 and NIST-521.
Conclusion

High performance library which deals with different coordinates systems and families of curves.

The modularity of MPHELL allows to plug other libraries.

It can be seen as a test platform for the arithmetic of unsupported curves.

Work in progress

- ARM V7 architecture.
- Finalisation of the integration on PARI library.
Thank You!