ALGORITHMICS, graduate program 2025/2026
CRYPTOGRAPHY, 2025/2026, lab assignments list # 2, 13.11.2025
Regular deadline: 12.12 or 19.12.2025

In this list, we implement the basic building blocks for public-key cryptography.

1. (3pts) Implement a generic structure for arithmetic over finite fields of form $\mathbb{F}_{p^k}$ for a prime p. In particular, focus on extensions of $\mathbb{F}_p$ as polynomial rings over $\mathbb{F}_p$ modulo an irreducible polynomial of degree k.

   Your implementation should allow the user to *at least* perform the basic field operations, that is:

   - Addition: $a + b$
   - Negation: $-a$
   - Subtraction as addition with negation: $a + (-b) = a - b$
   - Multiplication: $a * b$
   - Inverse: a^{-1}
   - Division as multiplication by inverse: $a/b = a * b^{-1}$

   Additionally, implement exponentiation – repeated multiplication – using an efficient algorithm, that is, exponentiation a^{exp} should have complexity $O(log(exp))$ of multiplications.

   In cryptography, you may encounter finite fields of various sizes, but public-key cryptography typically requires at least 256 bits, so take "big integers" into account. Your structure does not have to be dynamically sized and may be compile-time configurable, but must support 256, 512, 1024 (and so on) -bit elements.

   Prepare the implementation in such a way that there are specialized interfaces for the "edge" cases of $k = 1$ and $p = 2$, where the $k = 1$ case is reduced to just an $\mathbb{F}_p$ and $\mathbb{F}_{2^k}$ element can be represented as bit strings. The case $\mathbb{F}_{2^k}$ is also commonly referred to as $\mathbb{F}_{2^m}$ or simply `F2m`.

   Note on the implementation: when dealing with bit strings we usually start counting from the first bit of the first byte (i.e. "Little Endian" byte order), while when dealing with large integers, we usually use a "Big Endian" notation with the most significant digits first. This is not a hard requirement for the internal representation, but a hint as to why we may encounter one or the other in the wild.

   Your structure should take p and an irreducible polynomial over $\mathbb{F}_p$ as input (when $k > 1$). Optionally, you may implement a separate procedure for generating such polynomials.

2. (3pts) Recall the concept of Elliptic Curves, in particular over finite fields. Using the structure $\mathbb{F}_{p^k}$, implement a structure for the elliptic curve groups over generic finite fields [1]. Start with the simple case where $p > 3$, which implies the field characteristic $K \notin \{2, 3\}$, which in turn allows the most common "Short Weierstrass" form:

$$y^2 = x^3 + ax + b$$

   As a reminder, elliptic curve groups are defined using the Chord-Tangent Law. It is based on a principle that a straight line intersecting the curve intersects it at one or three points (with tangent points counting as double), so every line that connects two points, also intersects the curve at a third one, with a minor exception of two points with opposite y coordinates that don't have an explicit

"third point" that can be represented using numerical coordinates but instead point to an additional "point at infinity", which represents a point where all the vertical lines cross. [1]

According to the Chord-Tangent Law, the group operation (addition) is defined as follows: for any two points P and Q, the points P, Q and $-(P+Q)$ are co-linear.

A decent summary of the operations, along with explicit formulas, is available, e.g., on Wikipedia [2].

3. (2 pts) Using the $\mathbb{F}_{2^m}$ structure, implement elliptic curve groups over binary fields. The major difference comes from the field characteristic $K = 2$, which means that the "Short Weierstrass" form is no longer the most general form. Instead, we must use the following form:

$$y^2 + xy = x^3 + ax^2 + b$$

(which n.b. is still not the most generalized Weierstrass form, but is general for all nonsupersingular curves of $K = 2$). A different representation unfortunately implies other formulas for point addition, and in turn requires a separate implementation[3, 4].

4. (2pts) In pairs or groups of three, test the interoperability of your structures. In particular, focus on serializing and de-serializing elements and field representations. For integers, ensure support for a "Base 10", "Base 16" (hexadecimal), and possibly "Base64" representation.

   This task applies to *all* the aforementioned structures, i.e., test the interoperability of everything that you implemented.

   If you have done all tasks 1-3, you should end up with: $\mathbb{F}_p$, $\mathbb{F}_{2^m}$, $\mathbb{F}_{p^k}$, $\mathbb{E}(\mathbb{F}_p)$, $\mathbb{E}(\mathbb{F}_{2^m})$, $\mathbb{E}(\mathbb{F}_{p^k})$

/-/ Marcin Słowik

References

[1] Ben Lynn. *Elliptic Curves*. URL: https://crypto.stanford.edu/pbc/notes/elliptic/.

[2] Wikipedia contributors. *Elliptic curve point multiplication*. URL: https://en.wikipedia.org/wiki/Elliptic_curve_point_multiplication.

[3] Thomas Pornin. *Efficient and Complete Formulas for Binary Curves*. Cryptology ePrint Archive, Paper 2022/1325. 2022. URL: https://eprint.iacr.org/2022/1325.

[4] Standards for Efficient Cryptography Group (SECG). *SEC 2: Recommended Elliptic Curve Domain Parameters*. URL: https://www.secg.org/sec2-v2.pdf.

[1] This point has a nice explanation using projective coordinates, but let us keep it simple here.