Courbes elliptiques — 4TMA902U

Responsables : G. Castagnos, D. Robert

ECDSA

The P − 256 elliptic curve

Several elliptic curves are standardized by the NIST for cryptographic purposes. See for instance `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`, appendix D.1. We are going to use the curve P − 256 defined over $\mathbf{F}_p$ where p is a prime number of 256 bits. The equation of this curve is $y^2 = x^3 - 3x + b$ where

$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$

$b = 41058363725152142129326129780047268409114441015993725554835256314039467401291$

1 Define this curve with Sage. Verify that the group of points of the curve over $\mathbf{F}_p$ is cyclic of prime order n with

$n = 115792089210356248762697446949407573529996955224135760342422259061068512044369.$

In the standard, the NIST gives as generator the point $P = (x, y)$ with

$x = 48439561293906451759052585252797914202762949526041747995844080717082404635286$

$y = 36134250956749795798585127919587881956611106672985015071877198253568414405109$

2 Verify that P is indeed on the curve. Compute nP using the double and add algorithm.

3 Simulate a Diffie-Hellman key exchange.

Elgamal Signature

Let $G = \langle g \rangle$ be a cyclic group of prime order n. A message m is mapped to $M = H(m) \in \mathbf{Z}/n\mathbf{Z}$ where H is an hash function $H : \{0,1\}^* \to \mathbf{Z}/n\mathbf{Z}$. We use the same setup as in the Elgamal encryption scheme: The public key is $h = g^x$ and the private key is $x \in \mathbf{Z}/n\mathbf{Z}$.

We try to build a signature of m from a Diffie-Hellman triplet that involves M. We thus consider an Elgamal encryption of g^M : $c = (g^r, h^r g^M)$ with r random in $\mathbf{Z}/n\mathbf{Z}$. Let $s := xr + M \in \mathbf{Z}/n\mathbf{Z}$ and $f := g^r$, we then have $c = (f, g^s)$. The signature of m could be $\sigma := (f, s)$, and the verification could consists in verifying if $(h, f, g^s/g^M) = (g^x, g^r, g^{xr})$ is a Diffie-Hellman triplet but from the DDH assumption this is an hard problem (Note that this will be possible with pairing friendly elliptic curves).

4 Show that if the signature is defined as $\sigma := (f, s, r)$ then it is possible to verify the signature but the scheme becomes insecure .

The solution used by Elgamal is to use $H(f)$ instead of using r in the computation of s and to define s such that $f^s := g^{xH(f)+M}$.

5 Give the complete Elgamal signature scheme and show that it is correct.

ECDSA

This NIST standard also described the ECDSA signature scheme. This is a variant of the Elgamal signature, with some changes and optimization to handle the representation of the group elements, *i.e.*, the points of an elliptic curve.

- **Global Parameters:**
  P a point of order n, $H : \{0,1\}^* \to \{1, \dots, n-1\}$ a cryptographic hash function
- **Key Generation:** $pk := Q := xP$ with x random $0 < x < n$, $sk := x$
- **Signing a message m with the key x:**
  r random, $0 < r < n$, $R := (x_R, y_R) := rP$, If $x_R \equiv 0 \pmod n$, restart with another r.
  $s :\equiv r^{-1}(x(x_R \mod n) + H(m)) \pmod n$. If $s \equiv 0 \pmod n$, restart with another r.
  The signature is $\sigma := (\sigma_1, \sigma_2) := (x_R \mod n, s)$.
- **Verifying a signature (σ_1, σ_2) of m with the key $pk = Q$**
  Verify that Q is on the curve, and that Q has order n and that $1 < \sigma_i < n$, for $i = 1, 2$.
  $u_1 :\equiv H(m)\sigma_2^{-1} \pmod n$; $u_2 :\equiv \sigma_1\sigma_2^{-1} \pmod n$; $(x_1, y_1) := u_1P + u_2Q$
  Signature is correct if $\sigma_1 \equiv x_1 \pmod n$

6 Show that the scheme is correct.

7 Show that if two signatures are issued with the same secret key x and the same randomness r, then one can deduce x (Real life attacks! Playstation 3 in 2010, an implementation of Bitcoin on Android in 2013).

8 Implement the ECDSA signature algorithm. For the hash function, use the following code

```
import hashlib
sha2 = hashlib.sha256

def H(m):
   hashHexa = sha2(str(m)).hexdigest()
   return ZZ(hashHexa,16)
```