(Applied) Cryptography

Tutorial #8

Bernardo Portela (bernardo.portela@fc.up.pt) Rogério Reis (rogerio.reis@fc.up.pt)

November 24, 2023

1. The Discrete Logarithm Problem (DLP) and the cryptographic systems based on DPL designed for integer finite fields translate, very easily, to the Elliptic Curves realm. An example of such system is ElGamal.

   Alice and Bob agree to use a particular prime p, elliptic curve E, and point $P \in E(\mathbb{F}_p)$. Alice chooses a secret multiplier n_A and publishes the point $Q_A = n_A P$ as her public key. Bob's plaintext is a point $M \in E(\mathbb{F}_p)$. He chooses an integer k to be his ephemeral key and computes

   $$C_1 = kP \text{ and } C_2 = M + kQ_A.$$

   He sends the two points (C_1, C_2) to Alice, who computes

   $$C_2 - n_A C_1 = (M + kQ_A) - n_A(kP) = M + k(n_A P) - n_A(kP) = M$$

   to recover the plaintext. The problem remains how to transform an arbitrary token in a point of the chosen elliptic curve. A solution is given by a variant of this system: the Menezes–Vanstone variant for ECC ElGamal as it follows.

   - A trusted party chooses and publishes a (large) prime p, an elliptic curve E over $\mathbb{F}_p$, and a point $P \in E(\mathbb{F}_p)$.
   - Alice chooses a secret multiplier n_A, computes $Q_A = n_A P$ and publishes the public key Q_A.
   - Bob has his plaintext data coded as two integers, $m_1, m_2 \in \mathbb{Z}_p$ and generates a random number $k \in \mathbb{Z}_p$ that will play the role of an ephemeral key.
     Computes $R = kP$.
     Computes
     $$S = (x_S, y_S) = kQ_A.$$
     Makes
     $$c_1 \equiv x_S m_1 \pmod{p} \quad \text{and} \quad c_2 \equiv y_S m_2 \pmod{p}.$$
     Sends the ciphertext (R, c_1, c_2) to Alice.
   - Alice computes
     $$T = (x_T, y_T) = n_A R, \quad m_1' \equiv x_T^{-1} c_1 \pmod{p}, \text{ and } m_2' = y_T^{-1} c_2 \pmod{p}.$$

   a) Show that $(m_1, m_2) = (m_1', m_2')$ i.e. that Alice can recover the original plaintext.

   b) Write three **python/SAGE** methods that for a Elliptic Curve

   $$E : y^2 = x^3 + Ax + B, p \text{ prime, and } P = (x_P, y_P) \in E(\mathbb{F}_p)$$

   - `GenPubKey`(A, B, p, x_P, y_P) generates the public key Q_A and the private key n_A.
   - `Encript`$(A, B, p, x_P, y_P, Q_A, m_1, m_2)$ gererates the ciphertext (R, c_1, c_2).
   - `Decript`$(A, B, p, x_P, y_P, n_A, R, c_1, c_2)$ recovers the value of m_1, m_2 back.