Software Engineering Department

Data Security and Cryptology 61767, Spring 2021

ORT Braude

Rabbit Cipher

McEliece Cryptosystem

ECDSA Signature

for secure access to database

Liad Vaksman

Elroye Cahana

Or Steiner

Lior Wunsch

1. Project Study

As we started our project, we searched for a proper cryptosystem that will enable us to access a passwords database securely with ECDSA digital signatures, McEliece cryptosystem and Rabbit encryption. We searched as a group for the papers for those ciphers and studied them.

2. ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) offers a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography. Elliptic curve cryptography [ECC] is a public-key cryptosystem where the public key is used for signature verification and the private key is used for signature generation.

Key Generation:

An elliptic curve is a plane curve defined by an equation of the form $y^2 = x^3 + ax + b$over a field Z_p where $p > 3$ must be prime and $(x, y) \in Z_p \, x \, Z_p$.

Suppose Alice wants to send a signed message m to Bob. Initially, they must agree on the curve parameters $(CURVE, G, n)$. Alice creates a key pair, consisting of a private key integer d_A, randomly selected in the interval $[1, n-1]$, and a public key curve point $Q_A = d_A * G$.

Signature Generation:

For Alice to sign a message m, she uses her own private key and follows these steps:

1. Select a cryptographically secure random integer k from $[1, n-1]$.
2. Calculate r as the curve point $(x_1, y_1) = k * G$.
3. Calculate $s = k^{-1} * (Hash(m) + r * d_A) \, mod \, n$.
4. The signature is the pair (r, s).

Signature Verification:

For Bob to authenticate Alice's signature, he must have a copy of Alice's public keyQ_A. which can be verified a valid curve point as follows:

1. Check that Q_A lies on the curve and verify that r and s are integers in $[1, n-1]$.
2. Calculate $u_1 = Hash(m) * s^{-1} \, (mod \, n)$ and $u_2 = r * s^{-1} \, (mod \, n)$.
3. Calculate the curve point $(x_1, y_1) = u_1 * G + u_2 * Q_A$and verify $r == (x_1, y_1)$.

3. McEliece Cryptosystem

McEliece cryptosystem is an asymmetric encryption algorithm, which means everyone has a public key and a secret key. Here Alice encrypts her message to Bob using his public key and only Bob's secret key can decode the message.

Key Generation:

1. Bob chooses his secret key (S, G, P) where $G \in {F_2}^{n\,x\,k}$ is a generator matrix for an appropriate linear code C that is efficiently decodable from up to t errors, $S \in {F_2}^{k\,x\,k}$ is a random invertible matrix, and $P \in {F_2}^{n\,x\,n}$ is a random permutation matrix.
2. Bob calculates his public key $\widehat{(G} = P\ x\ G\ x\ S, t)$.

Encryption:

Suppose Alice wishes to send Bob a cipher key $x \in {F_2}^{k}$ and Bob's public key is $(\hat{G}, t)$:

1. Alice chooses a random $e \in {F_2}^{n}$, of weight t.
2. Alice sends $c = x * \hat{G} + e$ to Bob.

Decryption:

Upon receipt of c, Bob performs the following steps to decrypt the message:

1. Bob computes $c * P^{-1} = (x * S) * G + e * P^{-1} = (xS) * G + e'$.
2. Bob uses the decoding algorithm A get xS.
3. Bob computes $x = (xS) * S^{-1}$.

4. Rabbit Cipher

Rabbit is a symmetric stream cipher. Rabbit uses a 128-bit key and a 64-bit initialization vector. The Rabbit Cipher uses an automated machine that runs on key and IV. After i iterations the machine produces a new cipher key k' which will be used for encryption and decryption on the input stream as: $cipher\ text\ =\ message\ \oplus\ k'$, $message\ =\ cipher\ text\ \oplus\ k'$. Each side knows the key and the IV and calculates k'.

5. Implementation Flow

Client sends a request to Server:

1. Generate McEliece keys, a public key and a private key, for each participant.
2. Generate ECDSA keys, a public key and a private key, for each participant.
3. Client generates a Rabbit key (up to 128 bits).
4. Client encrypts the request using Rabbit.
5. Client and Server share their public keys.
6. Client encrypts the Rabbit key via McEliece using Server's public McEliece key.
7. Client signs the request via ECDSA using his private ECDSA key.
8. Client sends to Server the cipher and the signature.
9. Server decrypts the Rabbit key using his private McEliece key.
10. Server decrypts the request using the Rabbit key.
11. Server verifies Client's signature on the request using Client's public ECDSA key.
12. Server processes Client's request.