

## Cryptographic system

This assignment consists in combining the functions built in previous practicals in order to implement a communication system which is secure and guarantees privacy, integrity, authenticity and non-repudiation of the information exchanged among the users of the system.

We assume that each user has a pair of keys for the DH system and a pair of keys for the ECDSA system.

**Sending a message.** When a user (the sender) wishes to send a message  $M$  to another user (the receiver) (s)he acts according to the following procedure:

1. The plain-text  $M$  is signed with a private ECDSA key, adding the 64 bytes corresponding to the signature; let  $M\|F$  be the message together with the signature.
2. A list of 32 bytes  $KSE$  is randomly generated; this together with a private ECC key and a public ECC key is used to generate a 256-bit key  $KS$ , the session key. With this key the message  $M\|F$  is encrypted using AES with the mode of operation CBC. Let  $E(M\|F)$  be the encrypted message.
3. The concatenation of  $KSE$  and  $E(M\|F)$  is sent to the receiver.

**Receiving a message.** When the receiver gets the cryptogram  $KSE\|E(M\|F)$ , (s)he proceeds in reverse order to obtain the original plaintext and verify the signature:

1. First the received information is split into two parts, corresponding to  $KSE$  and  $E(M\|F)$ .
2. The session key  $KS$  is recovered using  $KSE$  and the corresponding ECC keys.
3. The message  $E(M\|F)$  is decrypted with the session key  $KS$  and the symmetric cipher AES, obtaining  $M\|F$ .
4. The message  $M$  is recovered, and the signature  $F$  is verified with the public ECDSA key of the sender. If the verification is correct, return  $M\|F$  concatenated with the byte 0x00 ( $M\|F\|0x00$ ), if the verification is incorrect, return  $M\|F$  concatenated with the byte 0xff ( $M\|F\|0xff$ ).

**Implementation: signatures.** Define the class `systemaCriptografic` with the following methods:

```
public static byte[] enviarMissatge(byte[] M, BigInteger clauDeFirma, BigInteger clauPrivadaECC,
                                   BigInteger[] clauPublicaECC, BigInteger[] parametresECC)
```

input:  $M$  is the message to be signed, given by a list of bytes,  
 $clauFirma$  is the private key of the sender,  
 $clauPrivadaECC$  is an integer,  
 $clauPublicaECC = \{P_x, P_y\}$  (different from the point at infinity)  
 $parametresECC = \{n, G_x, G_y, a, b, p\}$ ,  $G = (G_x, G_y)$  point of order  $n$  in the curve  $y^2 = x^3 + ax + b \pmod p$  (obviously,  $G$  is not the point at infinity);  
output: a list of bytes representing  $KSE\|E(M\|F)$ .

```
public static byte [] rebreMissatge(byte[] C, BigInteger[] clauDeVerificacioDeFirma,  
    BigInteger clauPrivadaECC, BigInteger[] clauPublicaECC,  
    BigInteger[] parametresECC)
```

input: C is the received cryptogram, given as a list of bytes,  
clauDeVerificacioDeFirma public key of the sender for signature verification,  
clauPrivadaECC private key corresponding to the public key used to encrypt the  
message,  
clauPublicaECC public key corresponding to the private key used to encrypt the  
message,  
parametresECC= $\{n, G_x, G_y, a, b, p\}$ ,  $G = (G_x, G_y)$  point of order  $n$  in the curve  $y^2 = x^3 + ax + b \pmod p$  (obviously,  $G$  is not the point at infinity);  
output: a list of bytes  $M||F||ver$  where  $M$  is the decrypted message,  $F$  is the signature and  $ver$   
is a byte with value 0x00 if the signature has been successfully verified and 0xff is the  
signature has not been verified.