**Lecture 5.2: Elliptic Curve Cryptosystem** 15 October 2019

**LEARNING OUTCOME**

**By the end of the lesson the student will be able to:**

1. to understand an ECC projection point
2. to understand a difficult problem in ECC
3. to understand an ECC encryption mode
4. to relate to sign and verify a digital signature in ECDSA

In ECC, Projection Point P(*x*, *y*) = ⊗P(*x*, *y*), in order to compute the factor as the private key  is the discrete logarithm problem over elliptic curve field.

What is the difficult problem in ECC?

ECC Encryption and Decryption

Given a base point (*x*0, *y*0) and the projection point (*x*, *y*) as public key, describe the process of digital signing using an Elliptic Curve Cryptosystem.

Encryption using ECC

random Encrypt key *E*

Base Point

(*x*0, *y*0)

Projection Point

(*x*, *y*)

Private key 

First Point

(*x*1, *y*1)

Second Point

(*x*2, *y*2)

random Encrypt key *E*

Private key 

Figure 1. An encryption mode on ECC

Suppose Alice want to encrypt a message M to Bob using AES via secret key *K*.

1. Generate random Encrypt key *E*. In RSA an encrypt key is in fact the public exponent *e*.
2. A sender Alice project to the first point (*x*1, *y*1) via *E* from the base point, computing vertically downwards.
3. A sender Alice project to the second point (*x*2, *y*2) via *E* from the Bob’s public key, computing vertically downwards.
4. Alice will encrypt the message M using AES with key *K* = *x*2 and get the ciphertext C= EncryptAES*K*(M).
5. Send the first point (*x*1, *y*1), preferably just *x*1 and +1 to represent y1 together with the ciphertext C.

In the decryption process done horizontally; Once,

1. Bob received the first point (*x*1, *y*1) and the ciphertext C;
2. Bob will use his private key  and project from first point (*x*1, *y*1) to second point (*x*2, *y*2)
3. Bob will take the session key *K* = *x*2 and decrypt the message M = DecryptAES*K*(C).

Note: Going down can be publicly done by a sender. Computing from the left to the right can only be done by a receiver/owner of the public key who must have the private key.

Question 1a) Given Bob’s public key, describe step-by-step encryption process to be done by Alice to send an encrypted message to Bob.

Question 1b) Write THREE(3) steps Bob need to do in order to decrypt a message from Alice.

**Digital Signing using ECDSA**

Generating and verifying signatures via ECDSA is a little bit more complicated that encrypting data, but it's not impossible to understand. Again, let us assume that Alice has a public key pair P0(*x*0, *y*0) and P(*x*, *y*) and want to sign a message *m* with her private key . In ECC, Projection Point : P0(*x*0, *y*0) → P(*x*, *y*). From a base point to a projection point can be computed as P(*x*, *y*) =  P0(*x*, *y*), in order to compute the factor as the private key  is the discrete logarithm problem over elliptic curve field.

What is the difficult problem in ECC?

**Digital Signing**

1. First, Alice will calculate a hash value of the message *m* that we're about to sign. For this for example SHA can be used:

*e* = SHA(*m*)

2. Generate a random signing key *k*, 2 < *k* < *n* where p is the finite field modulo.

Given a base point (*x*0, *y*0) and the projection point (*x*, *y*), describe the process of digital signing using an Elliptic Curve Cryptosystem.

Point Projection using ECC

random signing key *k*

Base Point

(*x*0, *y*0)

Projection Point

(*x*, *y*)

Private key 

First Point

(*x*1, *y*1)

Second Point

(*x*2, *y*2)

random signing key *k*

Private key 

From Base Point (*x*0, *y*0), anyone can generate the First Point P1(*x*1, *y*1) = *k* P0(*x*0, *y*0),

Let the number of points on the elliptic curve be #*n*.

Take *x*1 (mod *n*)

3. Compute  (mod *n*)

4. The digital signature is (*x*1, *s*).

5. Alice sent the message *m* and the digital signature (*x*1, *s*).

**Signature Verification**

Bob receives a message *m*, a signature (*x*1, *s*) and Bob has Alice’s public key

base point P0(*x*0, *y*0) and the projection point P(*x*, *y*), supposedly belong to a sender Alice.

Let's check it by first verifying that the values or (*x*1, *s*) are plausible. If they're not, the signature is invalid:

1. First, Bob calculates a hash value of the message *m* that he is about to verify. For this for example SHA can be used:

*e* = SHA(*m*)

2. Compute *w* ≡ *s*−1 (mod *n*)

3. Compute *u* ≡ *e*⋅*w* (mod *n*) and *v* ≡ *x*1⋅*w* (mod *n*)

4. Compute (*xr*, *yr*) = *u*⊗(*x*0, *y*0) + *v*⊗(*x*, *y*)

5. Check *x*1 = *xr* ?

**Validity of Signature**

From

(*xr*, *yr*) = *u*⊗(*x*0, *y*0) + *v*⊗(*x*, *y*)

= *e*⋅*w* ⊗(*x*0, *y*0) + *x*1⋅*w* ⊗(*x*, *y*)

= *e*⋅*w* ⊗(*x*0, *y*0) + *x*1⋅*w* ⋅⊗(*x*0, *y*0),

= [*e*⋅*w* + *x*1⋅*w*⋅⊗(*x*0, *y*0)

= *w*⋅[*e* + *x*1⋅⊗(*x*0, *y*0)

Remember

*w* = *s*−1 mod *n* and  (mod *n*)

(*xr*, *yr*) = **⋅ (*e* + ⋅*x*1⊗(*x*0, *y*0)

= **⋅(*e* + ⋅*x*1⊗(*x*0, *y*0)

= *k*⊗(*x*0, *y*0)

= (*x*1, *y*1)

From First Point (*x*1, *y*1), the owner can generate the Second Point P2(*x*2, *y*2) =  P1(*x*1, *y*1),

From Base Point (*x*0, *y*0), the owner has generated the First Point (*x*1, *y*1) =*k*(*x*0, *y*0),

And only the owner can generate the Projection Point P (*x*, *y*) =(*x*0, *y*0),

A difficult problem here is to determine , given P (*x*, *y*) =P0(*x*0, *y*0),

from Base Point P0(*x*0, *y*0) and Projection Point P (*x*, *y*).

How do we reach P(*x*, *y*)? How do we compute ⊗P0(*x*0, *y*0)?

Let P0(*x*0, *y*0) = (1, 3), *a*=1; *b*=7; *p*=257;

From a base point (1, 3), take a private key  = 199, the projection point as a public key shall be (70, 53).

Encryption using ECC

random Encrypt key *E*

Base Point

(*x*0, *y*0) =(1, 3)

Projection Point

(*x*, *y*) = (70, 53)

Private key 

First Point

(*x*1, *y*1) =(7, 247)

Second Point

(*x*2, *y*2) =(208, 234)

random Encrypt key *E*

Private key 

Figure 1. An encryption mode on ECC

Take a random encrypt key *E*=188.

From base point, Alice is going to project to First Point (7, 247).

From Bob’s public key, Alice is going to project to Second Point (208, 234).

Alice will encrypt C=AESk=208(M) then send(7, 247) and ciphertext C to Bob.

Signing using ECC

random Signing key *k*=233

Base Point

(*x*0, *y*0) =(1, 3)

Projection Point

(*x*, *y*) = (70, 53)

Private key =199

First Point

(*x*1, *y*1) =(39, 68)

Second Point

(*x*2, *y*2) =(?,?)

random Signing key *k*=233

Private key =199

Figure 2. Digital signing mode on ECC

3. Compute  (mod *n*) = 221.

4. The digital signature is (*x*1, *s*) = (39, 221) with *e*=SHA(*m*)=177.

5. Alice sent the message *m* and the digital signature (*x*1, *s*).

Verification Process

1. First, Bob calculates a hash value of the message *m* that he is about to verify. For this for example SHA can be used:

*e* = SHA(*m*)=177.

2. Compute *w* ≡ *s*−1 (mod *n*)=192.

3. Compute *u* ≡ *e*⋅*w* (mod *n*) = 264 and *v* ≡ *x*1⋅*w* (mod *n*) = 182

4. Compute (*xr*, *yr*) = *u*⊗(*x*0, *y*0) ⊕ *v*⊗(*x*, *y*) = (154, 202) ⊕ (207, 170) = (39, 68)

5. Check *x*1 = *xr* ? 39=39?