**Lecture 6: 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

Let us list down irreducible polynomials of degree 8 as shown in Table 6.1 below.

Table 6.1. Irreducible Polynomials of degree 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i* | *Pi*(*t*) | Binary | Hexa | Decimal |
| 0 | *t*8 + *t*4 + *t*3 + *t*2 + 1 | 100011011 | 11B | 283 |
| 1 | *t*8 + *t*5 + *t*3 + *t*+ 1 | 100101011 | 12B | 299 |
| 2 | *t*8 + *t*6 + *t*4 + *t*3 + *t*2 + *t*+ 1 | 101011111 | 15F | 351 |
| 3 | *t*8 + *t*6 + *t*5 + *t*+ 1 | 101100011 | 163 | 355 |
| 4 | *t*8 + *t*6 + *t*5 + *t*2 + 1 | 101100101 | 165 | 357 |
| 5 | *t*8 + *t*6 + *t*5 + *t*3 + 1 | 101101001 | 169 | 361 |
| 6 | *t*8 + *t*7 + *t*6 + *t* + 1 | 111000011 | 1C3 | 451 |
| 7 | *t*8 + *t*7 + *t*6 + *t*5 + *t*2 + *t* + 1 | 111100111 | 1E7 | 487 |

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.

What will a question look like in an exam?

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 

Figure 2. Projection points as a digital signature

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)?

Table 6.2 A list of points on an elliptic curve *a*= 3, *b*= 7, *n*=137 over M=487.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | *xi* | *yi* | *i* | *xi* | *yi* | *i* | *xi* | *yi* |
| 1 | 2 | 3 | 51 | 32 | 76 | 101 | 17 | 57 |
| 2 | 124 | 194 | 52 | 15 | 26 | 102 | 215 | 100 |
| 3 | 245 | 200 | 53 | 100 | 38 | 103 | 3 | 105 |
| 4 | 183 | 193 | 54 | 236 | 27 | 104 | 209 | 182 |
| 5 | 4 | 158 | 55 | 44 | 159 | 105 | 218 | 4 |
| 6 | 214 | 148 | 56 | 85 | 102 | 106 | 196 | 185 |
| 7 | 254 | 213 | 57 | 186 | 98 | 107 | 88 | 37 |
| 8 | 231 | 126 | 58 | 62 | 19 | 108 | 125 | 85 |
| 9 | 63 | 216 | 59 | 177 | 66 | 109 | 23 | 134 |
| 10 | 14 | 44 | 60 | 52 | 9 | 110 | 242 | 25 |
| 11 | 39 | 199 | 61 | 159 | 226 | 111 | 164 | 253 |
| 12 | 95 | 220 | 62 | 149 | 223 | 112 | 110 | 49 |
| 13 | 175 | 76 | 63 | 194 | 61 | 113 | 56 | 31 |
| 14 | 255 | 232 | 64 | 140 | 192 | 114 | 152 | 136 |
| 15 | 153 | 159 | 65 | 111 | 71 | 115 | 26 | 152 |
| 16 | 16 | 249 | 66 | 64 | 113 | 116 | 219 | 32 |
| 17 | 225 | 105 | 67 | 182 | 159 | 117 | 248 | 75 |
| 18 | 123 | 162 | 68 | 101 | 32 | 118 | 146 | 119 |
| 19 | 146 | 229 | 69 | 101 | 69 | 119 | 123 | 217 |
| 20 | 248 | 179 | 70 | 182 | 41 | 120 | 225 | 136 |
| 21 | 219 | 251 | 71 | 64 | 49 | 121 | 16 | 233 |
| 22 | 26 | 130 | 72 | 111 | 40 | 122 | 153 | 6 |
| 23 | 152 | 16 | 73 | 140 | 76 | 123 | 255 | 23 |
| 24 | 56 | 39 | 74 | 194 | 255 | 124 | 175 | 227 |
| 25 | 110 | 95 | 75 | 149 | 74 | 125 | 95 | 131 |
| 26 | 164 | 89 | 76 | 159 | 125 | 126 | 39 | 224 |
| 27 | 242 | 235 | 77 | 52 | 61 | 127 | 14 | 34 |
| 28 | 23 | 145 | 78 | 177 | 243 | 128 | 63 | 231 |
| 29 | 125 | 40 | 79 | 62 | 45 | 129 | 231 | 153 |
| 30 | 88 | 125 | 80 | 186 | 216 | 130 | 254 | 43 |
| 31 | 196 | 125 | 81 | 85 | 51 | 131 | 214 | 66 |
| 32 | 218 | 222 | 82 | 44 | 179 | 132 | 4 | 154 |
| 33 | 209 | 103 | 83 | 236 | 247 | 133 | 183 | 118 |
| 34 | 3 | 106 | 84 | 100 | 66 | 134 | 245 | 61 |
| 35 | 215 | 179 | 85 | 15 | 21 | 135 | 124 | 190 |
| 36 | 17 | 40 | 86 | 32 | 108 | 136 | 2 | 1 |
| 37 | 83 | 135 | 87 | 208 | 115 | 137 | -1 | -1 |
| 38 | 77 | 173 | 88 | 105 | 137 |  |  |  |
| 39 | 206 | 141 | 89 | 134 | 94 |  |  |  |
| 40 | 122 | 136 | 90 | 189 | 32 |  |  |  |
| 41 | 98 | 96 | 91 | 45 | 49 |  |  |  |
| 42 | 207 | 83 | 92 | 163 | 137 |  |  |  |
| 43 | 201 | 64 | 93 | 84 | 238 |  |  |  |
| 44 | 84 | 186 | 94 | 201 | 137 |  |  |  |
| 45 | 163 | 42 | 95 | 207 | 156 |  |  |  |
| 46 | 45 | 28 | 96 | 98 | 2 |  |  |  |
| 47 | 189 | 157 | 97 | 122 | 242 |  |  |  |
| 48 | 134 | 216 | 98 | 206 | 67 |  |  |  |
| 49 | 105 | 224 | 99 | 77 | 224 |  |  |  |
| 50 | 208 | 163 | 100 | 83 | 212 |  |  |  |

Let  = 99. From an ID, take *i* = ID mod 100. Take a private key  = 30 + *i*.

Take j = MyKAD mod 100. Choose a signing key *k* = 20 + j.

Point Projection using ECC

random signing key *k*=122.

Base Point

(*x*0, *y*0) = (2, 3)

Projection Point

(*x*, *y*) = (77, 224)

Private key 

First Point

(*x*1, *y*1) = (153, 6)

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*), *e* =SHA(m) = 88. Generate session signing key *k*=122.

3. Compute  (mod *n*) = 126.

4. The digital signature is (*x*1, *s*) = (153, 126).

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) = (2, 3) and the projection point P(*x*, *y*) = (77, 224), supposedly belong to a sender Alice.

Let's check it by first verifying that the values or (*x*1, *s*) = (153, 126) 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*) = 88.

2. Compute *w* ≡ *s*−1 (mod *n*) = 126−1 (mod *n*) = 112.

3. Compute *u* ≡ *e*⋅*w* (mod *n*) = 129 and *v* ≡ *x*1⋅*w* (mod *n*) =11.

4. Compute (*xr*, *yr*) = *u*⊗(*x*0, *y*0) ⊕ *v*⊗(*x*, *y*) = (231, 153)⊕(254, 43)=(153, 6)

5. Check *x*1 = *xr* ? 153=153?