**Lecture 9: Protecting Master Key via Threshold Scheme over F*p***

**LEARNING OUTCOME**

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

1. understand a concept of a threshold scheme in protecting a master key
2. compute a set of shadow key over a polynomial over Fp.
3. interpolate a polynomial from a set of shadow key over Fp.

Ultimately, the keys are stored in the system and the entire system may depend on a single master key. This master key may be exposed, lost or destroyed. Having copies of the same master key to more people will increase the vulnerability of the system from betrayal. Master key is the most supreme key in a cryptosystem.

Threshold scheme provides a solution by breaking the master key into *n* shadows of keys to *n* highest board of directors or generals in central command controls. The master key can be of valid use by having some *m* keys or more together. Anything less than *m* shadow keys will not be a valid master key. Key recovery can always be done with *m* shadow keys. Fewer than *n− m* shadow keys lost will not endanger the system.

These are the very important key in a cryptosystem.

1. Shadow Keys of the Threshold Scheme,
2. The Master key of the Cryptosystem,
3. The Public and Private Key of the Public Key Infrastructure and then
4. The Session Key of the Symmetric Encryption

The higher keys shall be capable of overruling or opening the lower layer keys. In this topic, we propose to present and deliver a master key generation process. There will be the master key generating ceremony and the supreme master key shall be distributed into shadow keys. This supreme master key shall be the highest key to overrule the cryptosystems such the PKI at the application layer and Key Exchange mechanism at network layer of communication.

Threshold Scheme

Newton Polynomial

Chinese Remainder Theorem

Figure 0. A theoretical background on a threshold scheme.

There are several threshold schemes or secret sharing schemes. In general, we want protect the highest secret, such as one Master Key. Master Key belongs to an owner of a cryptosystem. We will break up a master key into *n* shadow keys. We need *m* < *n* shadow keys in order to recover or open the master key. There are 2 classic domains, namely, prime domain and irreducible polynomial domain.

In this class we will do newton polynomial over prime field. We want to show that any 5 points can be used to generate the polynomial. Given 5 points, we want to interpolate and generate the polynomial.

Compute the polynomial A(*x*) = *a*0 + *a*1⋅*x* + *a*2⋅*x*2 +...+ *am*−1 ⋅ *xm*−1

Generate *x*0, *x*1, *x*2, ..., *xn*−1.

Take *xi*from a small irreducible polynomial as listed in Table 2.

First let us evaluate a polynomial given by its coefficient *a*0, *a*1, *a*2, …., *am*−1.

Algorithm 1: An efficient mode to evaluate a polynomial in O(*m*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

An efficient mode to evaluate a polynomial

A(*x*) = *am*−1 ⋅ *xm*−1 + ... + *a*2⋅*x*2 + *a*1⋅*x* + *a*0.

given by coefficient *am*−1, …, *a*2, *a*1, *a*0 in little endian.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function *y* = Evaluate(*x*; *a*0, *a*1, *a*2, …., *am*−1)

*y* = 0

for *i* from *m*−1 down to 0,

*y* = *ai* + *x*⋅*y* (mod *p*)

end

return *y.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Specifically in this case,

A(*x*) = *x* (*a*4 ⋅ *x*3 + *a*3 ⋅ *x*2 + *a*2⋅*x*1 + *a*1) + *a*0.

Compute *y*0, *y*1, *y*2, ..., *yn*−1 by *yi* =*A*(*xi*) via an Algorithm 1.

Shadow keys are (*x*0, *y*0), (*x*1, *y*1), ..., (*xn*−1, *yn*−1) to interpolate an original polynomial.

Having *m*=5 shadow key points, an original polynomial can be regenerated

and evaluated at *x*=0 so that *A*(0)= *a*0 = K.

Threshold Scheme using Newton Polynomial mod a prime P = 257. Today, the prime modulo is expected to be 2048-bit.

The policy is *m* = 5 of *n* = 8 shadow keys.

Given the master key K =200 as *a*0.

Then generate random coefficients *a*1, *a*2, ..., *am*−1 < P. Take a polynomial of degree *m*−1.

Table 1. Coefficients of a Threshold Polynomial

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 |
| *ai* | 200 | 11 | 13 | 17 | 19 |
| *ai* | 200 | -246 | 13 | -240 | 19 |

Compute the polynomial A(*x*) = *a*0 + *a*1⋅*x* + *a*2⋅*x*2 +...+ *am*−1 ⋅ *xm*−1 (mod P)

Generate *x*0, *x*1, *x*2, ..., *xn*−1.

Compute *y*0, *y*1, *y*2, ..., *yn*−1 by *yi* =*A*(*xi*)

Table 2. Points of *n* shadow keys

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| *xi* | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 |
| *yi* | 35 | 188 | 193 | 138 | 211 | 186 | 194 | 209 |

Shadow keys are *m*=5 points out of *n*=8 points (*x*0, *y*0), (*x*1, *y*1), ..., (*xn*−1, *yn*−1) to interpolate an original polynomial.

Let us refresh on an interpolating polynomial.

If we have 2 points, we can generate a line, a linear polynomial P1(*x*) = b0 + b1 ⋅ *x*.

If we have 3 points, we can generate a quadratic polynomial P2(*x*) = b0 + b1 ⋅ *x* + b2 ⋅ *x*2 of degree 2.

If we have 4 points, we can generate a cubic polynomial P3(*x*) = b0 + b1 ⋅ *x* + b2 ⋅ *x*2 + b3 ⋅ *x*3 of degree 3.

Finally, if we have 5 points, we can generate

a quartic polynomial P4(*x*) = b0 + b1 ⋅ *x* + b2 ⋅ *x*2 + b3 ⋅ *x*3 + b4 ⋅ *x*4 of degree 4.

Having *m*=5 shadow key points (*x*0, *y*0), (*x*1, *y*1), ..., (*xm*−1, *ym*−1), an original polynomial can be regenerated and evaluated at *x*=0 so that *A*(0)= *a*0 = K.

Table 3. The first *m* Points of *n* shadow keys

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 |
| *xi* | 3 | 5 | 7 | 9 | 11 |
| *yi* | 35 | 188 | 193 | 138 | 211 |

To interpolate the original polynomial on these *m* shadow key points.

*y*12 = 

*y*23 = 

*y*34 = 

*y*02 = 

*y*13 = 

*y*24 = 

*y*03 = 

*y*14 = 

*y*04 = 

*x*0

*x*1

*x*2

*x*3

*x*4

*y*0

*y*1

*y*2

*y*3

*x*4

*y*01= 

Figure 1: Divided Difference Table to generate coefficients of interpolating newton polynomial

An interpolating newton polynomial is given by

P(*x*) = *y*0

+ *y*01⋅(*x−* *x*0)

+ *y*02⋅(*x−* *x*0)⋅(*x−* *x*1)

+ *y*03⋅(*x−* *x*0)⋅(*x−* *x*1)⋅(*x−* *x*2)

+ *y*04⋅(*x−* *x*0)⋅(*x−* *x*1)⋅(*x−* *x*2)⋅(*x−* *x*3)

Generating a **divided difference table** for newton interpolation

3

5

7

9

11

35

188

193

138

211

*y*01==153⋅129=205

*y*12==5⋅129=131

*y*23= 101

*y*34= 165

*y*02==110

*y*13= 121

*y*24= 16

*y*03= 216

*y*14=111

*y*04= 19

*y*01 = = ≡ 153⋅2−1 ≡ 153⋅129 ≡ 205 (mod 257)

*y*12 = = ≡ 5⋅2−1 ≡ 5⋅129 ≡ 131 (mod 257)

*y*02 = = ≡ 183⋅4−1 ≡ 183⋅193 ≡ 110 (mod 257)

*y*24 = = ≡ 64⋅4−1 ≡ 64⋅193 ≡ 16 (mod 257)

*y*03 = = ≡ 11⋅6−1 ≡ 11⋅43 ≡ 473 = 216 (mod 257)

*y*14 = = ≡ 152⋅6−1 ≡ 152⋅43 ≡ 111 (mod 257)

*y*04 = = ≡ 152⋅8−1 ≡ 152⋅225 ≡ 19 (mod 257)

An interpolating polynomial is given by

P(*x*) = 35

+ 205⋅(*x−* 3)

+ 110⋅(*x−* 3)⋅(*x−* 5)

+ 216⋅(*x−* 3)⋅(*x−* 5)⋅(*x−* 7)

+ 19⋅(*x−* 3)⋅(*x−* 5)⋅(*x−* 7)⋅(*x−* 9)

A target regenerated master is an intercept value of P(*x*) evaluated at *x* = 0.

P(0) = 35

+ 205⋅(0*−* 3)

+ 110⋅(0*−* 3)⋅(0*−* 5)

+ 216⋅(0*−* 3)⋅(0*−* 5)⋅(0*−* 7)

+ 19⋅(0*−* 3)⋅(0*−* 5)⋅(0*−* 7)⋅(0*−* 9)

= 35 – 615 + 1650 – 22680 + 17955 = –3655 ≡ 200 (mod 257).

Let us see another example:

The master key K will spread out into the w shadows of congruence classes. The policy is *m* = 5 out of *n* = 8 shadows.

Step 0: Given the secret master key K = 177.

Step 1: Pick a prime number p > K.

Step 2: Take *a*0 = K, generate random coefficients *a*1, *a*2, *a*3,…., *am*−1,

{*a*1, *a*2, *a*3,… , *a*4}= {73, 79, 83, 89}

Step 3: We generate the shadow key set

{*x*0, *x*1, *x*2,…., *xn−*1}= {101, 103, 105, 107, 109, 111, 113, 115}

Step 4: Decompose K into the *n* shadows by taking *yi* = P*m*−1(*xi*) mod p , for every *i* = 0, …., *n*−1.

*yi* = P4(*xi*) = *a*0 + *a*1⋅*x* + *a*2⋅*x*2 + *a*3⋅*x*3 + *a*4 ⋅*x*4

= 177 + 73⋅*x* + 79⋅*x*2 + 83⋅*x*3 + 89⋅*x*4

We shall have 8 shadow keys

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *x*1 | 101 | *y*1 | 128 |
|  | *x*2 | 103 | *y*2 | 94 |
|  | *x*3 | 105 | *y*3 | 244 |
|  | *x*4 | 107 | *y*4 | 190 |
|  | *x*5 | 109 | *y*5 | 53 |
|  | *x*6 | 111 | *y*6 | 206 |
|  | *x*7 | 113 | *y*7 | 246 |
|  | *x*8 | 115 | *y*8 | 22 |

Step 5: Pick the first 5 shadow keys

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *ki* |  |  |  |
| 101 | 128 |  |  |  |
|  |  | (94-128)/(103-101) = | -34⋅129 | = 240 |
| 103 | 94 |  |  |  |
|  |  | (244-94)/(105-103) = | 150⋅129 | = 75 |
| 105 | 244 |  |  |  |
|  |  | (190-244)/107-105) = | -54⋅129 | = 230 |
| 107 | 190 |  |  |  |
|  |  | (53-190)/(109-107) = | -137⋅129 | = 60 |
| 109 | 53 |  |  |  |

What is 2−1 ≡ 129 mod 257?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *xi* | *ki* |  |  |  |  | |  |  |
| 101 | 128 |  |  |  |  |
|  |  | 240 |  |  |  |
| 103 | 94 |  | (75-240)/(105-101) | -165⋅193 | 23 |
|  |  | 75 |  |  |  |
| 105 | 244 |  | (230-75)/(107-103) | 155⋅193 | 103 |
|  |  | 230 |  |  |  |
| 107 | 190 |  | (60-230)/(109-105) | -170⋅193 | 86 |
|  |  | 60 |  |  |  |
| 109 | 53 |  |  |  |  |

What is 4−1 ≡ 2−1 ⋅ 2−1 mod 257?

= 129 ⋅ 129 ≡ 193 (mod 257).

Step 6: Generate the Divided Difference Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *i* | *xi* | *f* [*xi*] | *f* [*xi*, *xi*+1] | *f* [*xi*, *xi*+1, *xi*+2] | *f* [*xi*, *..*, *xi*+3] | *f* [*xi*, *..*, *xi*+4] |
| 0 | 101 | 128 |  |  |  |  |
|  |  |  | 240 |  |  |  |
| 1 | 103 | 94 |  | 23 |  |  |
|  |  |  | 75 |  | 99 |  |
| 2 | 105 | 244 |  | 103 |  | 89 |
|  |  |  | 230 |  | 40 |  |
| 3 | 107 | 190 |  | 86 |  |  |
|  |  |  | 60 |  |  |  |
| 4 | 109 | 53 |  |  |  |  |

What is 6−1 ≡ 43 mod 257?

*f* [*x*0, *..*, *x*3] = (103−23)⋅43 = 80⋅43 = 3440 = 99 (mod 257)

*f* [*x*1, *..*, *x*4] = (86−103)⋅43 = −17⋅43 = −731 = 40 (mod 257)

What is 8−1 ≡ 2−1 ⋅ 4−1 = 129⋅193 = 24897 ≡ 225 mod 257?

*f* [*x*0, *..*, *x*4] = (40−99)⋅225 = −59⋅225 ≡ 89 (mod 257)

Step 7: We will write a newton polynomial as

P4(*x*) = *f* [*xi*] + *f* [*xi*, *xi*+1](*x* − *x*0) + *f* [*xi*, *xi*+1, *xi*+2]

P1(*x*) = *f* [*x*0] + *f* [*x*0, *x*1](*x* − *x*0)

P2(*x*) = *f* [*x*0] + *f* [*x*0, *x*1](*x* − *x*0) + *f* [*x*0, *x*1, *x*2](*x* − *x*0)(*x* − *x*1)

P3(*x*) = *f* [*x*0] + *f* [*x*0, *x*1](*x* − *x*0) + *f* [*x*0, *x*1, *x*2](*x* − *x*0)(*x* − *x*1)

+ *f* [*x*0, *…*, *x*3](*x* − *x*0)(*x* − *x*1)(*x* − *x*2)

P4(*x*) = *f* [*x*0] + *f* [*x*0, *x*1](*x* − *x*0) + *f* [*x*0, *x*1, *x*2](*x* − *x*0)(*x* − *x*1)

+ *f* [*x*0, *…*, *x*3](*x* − *x*0)(*x* − *x*1)(*x* − *x*2)

+ *f* [*x*0, *…*, *x*4](*x* − *x*0)(*x* − *x*1)(*x* − *x*2)(*x* − *x*3)

P4(*x*) = 128 + 240(*x* − *x*0) + 23(*x* − *x*0)(*x* − *x*1)

+ 99(*x* − *x*0)(*x* − *x*1)(*x* − *x*2)

+ 89(*x* − *x*0)(*x* − *x*1)(*x* − *x*2)(*x* − *x*3)

P4(*x*) = 128 + 240(*x* − 101) + 23(*x* − 101)(*x* − 103)

+ 99(*x* − 101)(*x* − 103)(*x* − 105)

+ 89(*x* − 101)(*x* − 103)(*x* − 105)(*x* − 107)

Step 8: The master key is recovered when we evaluate the polynomial P4(*x*) at *x* =0.

P4(0) = 128 + 240(0 − 101) + 23(0 − 101)(0 − 103)

+ 99(0 − 101)(0 − 103)(0 − 105)

+ 89(0 − 101)(0 − 103)(0 − 105)(0 − 107)

P4(0) = 128 − 240⋅101 + 23(101)(103)

− 99(101)(103)(105)

+ 89(101)(103)(105)(107)

P4(0) = 128 − 24240 + 239269 − 108139185+ 10402115745

= 10294191717

≡ 177 (mod 257)

**Tutorial 9: Newton Threshold Scheme**

a) Take your master key as *K*=100 + (ID mod 100)

b) Threshold Scheme using Newton Polynomial mod a prime P = 257. The policy is *m* = 5 of *n* = 8 shadow keys. Given the master key K as *a*0. Set a polynomial of degree m-1 from coefficients

[*a*0, *a*1, *a*2, …, *am*−1] in Table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 |
| *ai* | *K* | 11 | 13 | 17 | 19 |

Generate *n*=8 shadow keys from polynomial A(*x*) = *a*0 + *a*1⋅*x* + *a*2⋅*x*2 +...+ *am*−1 ⋅ *xm*−1

From *x*0, *x*1, *x*2, ..., *xn*−1, compute *y*0, *y*1, *y*2, ..., *yn*−1 by *yi* =*A*(*xi*) mod P.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| *xi* | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 |
| *yi* |  |  |  |  |  |  |  |  |

c) Take the first 5 points of shadow key to generate a polynomial P*m*−1(*x*) via Newton interpolation.

d) Evaluate the polynomial at *x* = 0 to regenerate the master key K.