Code-Based PQC : McEliece

부호기반 양자내성암호 : 맥엘리스

https://youtu.be/u4y3YehFivA

장경배

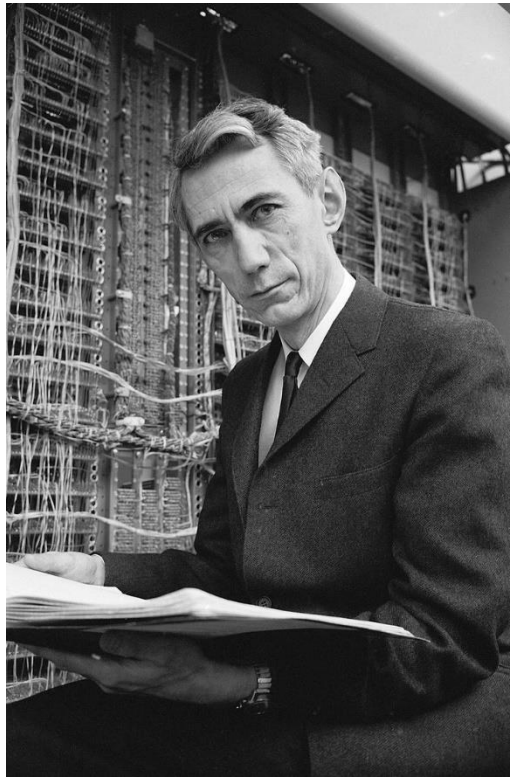
한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Contents

CryptoCraft LAB

1 부호이론



Claude Shannon

핵심은 덧붙임 !

1. 부호단어에 오류벡터를 추가하여 전송한다.

2. 수신자는 오류를 포함한 부호를 받게된다.

3. 수신자는 복호 알고리즘을 가지고있는 오류수정부호를 이용하여 정상부호를 획득하게 된다.

CryptoCraft LAB

2 McEliece

- 1978년 Robert J. McEliece 가 제안한 부호이론을 활용한 공개키 암호
- 최초의 부호기반 암호
- 양자 컴퓨터에 내성을 가짐
- NIST PQC 표준화 공모전 2Round 26개의 후보군 중 하나(Classic McEliece)
- 복호화 시 Syndrome Decoding 이 적용되는데, 이 부분이 NP-Complete 라는 점의 안전성에 기반함
- 암복호화 속도가 빠르지만 공개키의 크기가 매우 크다.

CryptoCraft LAB

3 알고리즘

개인키

T개의 오류수정 능력을 가진 선형부호 행렬 **G**(k x n) 생성 -> Goppa 부호를 사용

역행렬이 존재하는 가역행렬 **S**(k x k) 생성 ->랜덤하게

순열행렬 **P**(n x n) 생성 -> 랜덤하게

공개키

개인키 G, S, P 를 사용하여 공개키 생성

$$G' = S \cdot G \cdot P$$

CryptoCraft LAB

3 알고리즘

암호화

공개키와 메시지를 조합한다.

1의 개수가 T개 이하인 0과 1로 구성된 길이 n 오류벡터를 XOR ex) 0 0 0 0 1 0 0

$$c = G' \cdot m \oplus e$$

CryptoCraft LAB

3 알고리즘

복호화

P의 역행렬을 우측에 곱해준다

$$cP^{-1} = (mS)\,G \oplus eP^{-1}$$

그 후, 복호 알고리즘을 사용 -> 신드롬 복호 문제

참고 *

$$c = G' \cdot m \oplus e$$

$$G' = S \cdot G \cdot P$$

신드롬이란 홀짝검사행렬 즉 패리티 검사행렬 H를 이용해 임의의 벡터 x에 따라 Hx^T를 계산한 값

이 경우에는 암호문의 신드롬 값을 구하고 오류가 없다면 부호의 신드롬 값이 0임을 이용

$$Hx^T = 0$$

CryptoCraft LAB

3 알고리즘

복호화

$c' = mg' \oplus e$ 의 신드롬 값을 계산하면

$$Hc^T = H(g' + e)^T = Hg'^T + He^T = He^T$$

를 만족하므로 여기서 오류벡터를 찾은 후
오류벡터를 포함하지 않는 벡터들로 구성하면 (mS)
를 획득하게 된다.

이제 S^{-1} 을 계산해주면 원본메세지 m 획득

참고 *

$Hx^T = 0$ 을 이용.
(오류 없을 시)

CryptoCraft LAB

4 McEliece C코딩

Parameter.

***참고 : G**(k x n) , **S**(k x k), **P**(n x n)

4X7 행렬의 Goppa 부호 G를 사용

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include<math.h>
#include "inverse.h"

int main(void) {

    int Goppa[4][7] = { { 1,0,0,0,1,1,0 },{ 0,1,0,0,1,0,1 },{ 0,0,1,0,0,1,1 },{ 0,0,0,1,1,1,1 } };  //Goppa Code
    int PublicKey[4][7]; // PublicKey
    int Chiper[7]; //Chiper text
    int text[4]; // Plain text
    int(*P)[7] = MatrixPGeneration(); // Permutation Matrix as a Private Key
    int(*S)[4] = MatrixSGeneration(); // Invertible Matrix as a Private Key
    int(*ParrityCheck)[7] = MakeParrityCheckMatrix(Goppa); // ParrityCheck Matrix of Goppa Matrix for Decryption
    int(*TransPoseMatrix)[3] = Transpose(ParrityCheck);
```

CryptoCraft LAB

4 McEliece C코딩

Pivate Key.

***참고 : G**(k x n) , **S**(k x k), **P**(n x n)

순열행렬 P 생성

```
//Generate Random Permutation Matrix(7X7)
int MatrixPGeneration() {
    int Position[7];
    static int P[7][7] = { 0, };
    int check;
    for (int i = 0; i < 7; i++) {
        check = 1;
        int Randomnumber = rand() % 7;
        for (int j = 0; j < i; j++) {
            if (Randomnumber == Position[j])
                check = 0;
        }
        if (check == 1) {
            Position[i] = Randomnumber;
            P[i][Randomnumber] = 1;
        }
        else
            i--;
    }
    return P;
}
```

가역행렬 S 생성

```
//Generate Random Invertible Matrix(4X4)
int MatrixSGeneration() {
    srand(time(NULL));
    static int S[4][4];
    int check = 0;
    while (check == 0) {
        for (int i = 0; i < 16; i++) {
            S[0][i] = rand() % 2;
        }
        check = determinantOfMatrix(S, 4);
    }
    return S;
}
```

CryptoCraft LAB

4 McEliece C코딩

//공개키 생성함수
//입력받은 메시지를 암호화 하는 함수

```
PublicKeyGeneration(S, Goppa, P, PublicKey); //Generate Publickey using with P and S
Encryption(text, PublicKey, Chiper); //Encrypt Message with Publickey and intended Error

Encryption(int m[4], int PublicKey[4][7], int Chiper[7]) {
    int sum;
    int error[7] = { 0,0,0,0,1,0,0 };
    for (int i = 0; i < 7; i++) {
        sum = 0;
        for (int j = 0; j < 4; j++) {
            sum += m[j] * PublicKey[j][i];
        }
        Chiper[i] = sum % 2;
    }
    for (int i = 0; i < 7; i++) {
        Chiper[i] = Chiper[i] ^ error[i];
    }
}
```

공개키와 자신의 메시지를 조합한 뒤,

의도된 오류를 추가한 암호문 생성

CryptoCraft LAB

4 McEliece C코딩

복호화
P의 역행렬을 암호문 우측에 곱해줌

```
]Decryption(int Chiper[7], int InverseP[7][7], int InverseS[4][4], int TransPoseParrityCheck[7][3]) {
    int TempMessage[7];
    int ErrorMatrix[3];
    int ErrorPosition = 0;
    int sum = 0;
    int PlainText[4];

    //Multiplication InverseMatrix of P
    for (int i = 0; i < 7; i++) {
        for (int j = 0; j < 7; j++) {
            sum += Chiper[j] * InverseP[j][i];
        }
        TempMessage[i] = sum;
        sum = 0;
    }
```

$$cP^{-1} = (mS)\,G \oplus eP^{-1}$$

```
//Find Inverse Matrix(maximum 25X25)
int GenerateInverse(int S[25][25], int size) {
    int TempInverse[25][25];
    int Inverse[25][25];
    cofactor(S, size, TempInverse);

    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++)
            Inverse[i][j] = ((TempInverse[i][j] + 2) % 2);
    return Inverse;
}
```

CryptoCraft LAB

4 McEliece C코딩

복호화

신드롬 복호 -> 오류위치를 찾은 뒤 수정해주는 과정 $Hc^T = H(g' + e)^T = Hg'^T + He^T = He^T$

(패리티 행렬 생성 부분 생략)

```
//Syndrome Decoding(Find ErrorPosition)
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 7; j++) {
        sum += TempMessage[j] * TransPoseParrityCheck[j][i];
    }
    ErrorMatrix[i] = sum % 2;
    sum = 0;
}
ErrorPosition = ErrorMatrix[0] + ErrorMatrix[1] * 2 + ErrorMatrix[2] * 4;
TempMessage[7 - ErrorPosition] = (TempMessage[7 - ErrorPosition] + 1) % 2; //Error Correction
```

CryptoCraft LAB

4 McEliece C코딩

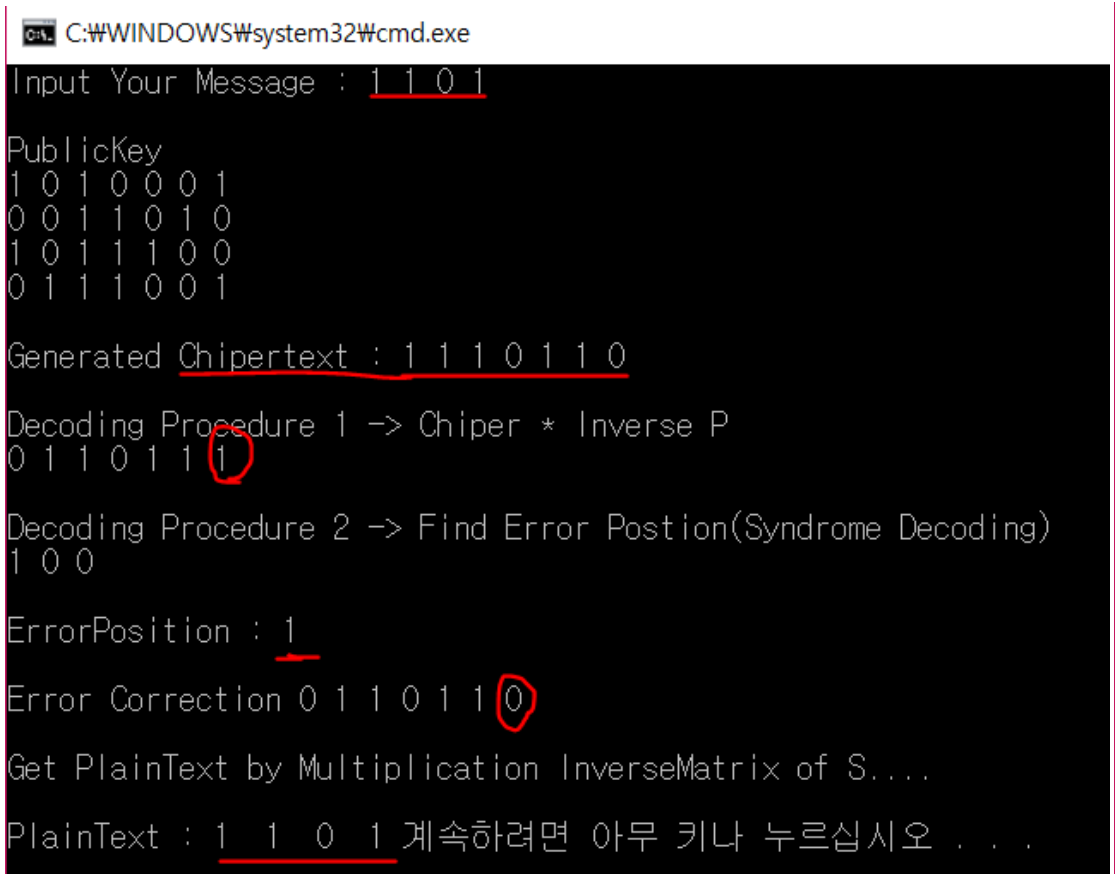
복호화

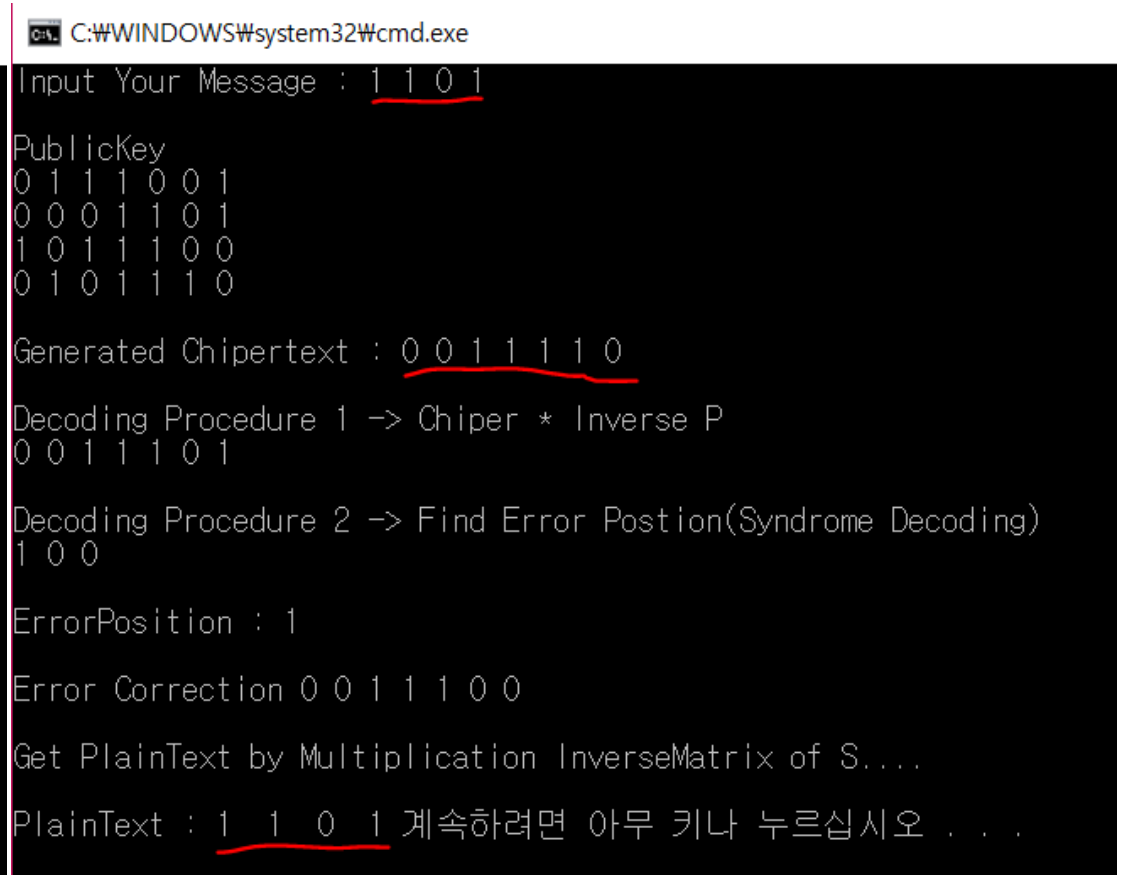
S의 역행렬을 곱해줌으로써 원본 메세지 복호 완료

```
//Multiplication InverseMatrix of S
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        sum += TempMessage[j] * InverseS[j][i];
    }
    PlainText[i] = sum % 2;
    sum = 0;
}
```

CryptoCraft LAB

4 McEliece C코딩





CryptoCraft LAB

Thank You