Classic McEliece

장경배

https://youtu.be/mYZEZpmU4gQ

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

코드 기반 양자내성 암호

Classic McEliece

BIKE

CryptoCraft LAB

Classic McEliece

1. 1981 Clark–Cain [18], crediting Omura.
2. 1988 Lee–Brickell [33].
3. 1988 Leon [34].
4. 1989 Krouk [32].
5. 1989 Stern [52].
6. 1989 Dumer [24].
7. 1990 Coffey–Goodman [19].
8. 1990 van Tilburg [55].
9. 1991 Dumer [25].
10. 1991 Coffey–Goodman–Farrell [20].
11. 1993 Chabanne–Courteau [15].
12. 1993 Chabaud [16].
13. 1994 van Tilburg [56].
14. 1994 Canteaut–Chabanne [11].
15. 1998 Canteaut–Chabaud [12].
16. 1998 Canteaut–Sendrier [13].
17. 2008 Bernstein–Lange–Peters [8].
18. 2009 Bernstein–Lange–Peters–van Tilborg [10].
19. 2009 Finiasz–Sendrier [27].
20. 2011 Bernstein–Lange–Peters [9].
21. 2011 May–Meurer–Thomae [37].
22. 2012 Becker–Joux–May–Meurer [3].
23. 2013 Hamdaoui–Sendrier [29].
24. 2015 May–Ozerov [38].
25. 2016 Canto Torres–Sendrier [54].

- 최초의 코드기반 암호 McEliece 와 Niederreiter 의 듀얼버전
- KEM(Key Encapsulation Mechanism) 으로 설계 되었음
- 1978년 이후, 코드기반암호를 연구한 점점 더 정교한 공격이 발표되었음

  Effect → 동일한 키 사이즈로 동일한 보안성을 달성한다
- Classic McEliece 팀의 제출에 대한 주요쟁점은 보안

  40년동안 안전성을 지켜온 McEliece의 Goppa code 를 사용하며

  자신들의 보안성을 훼손시키지 않는 선에서 효율성을 향상 시켰음

Classic McEliece – Key Generation Scheme

Key generation

Given a set of CM parameters, a user generates a *CM key pair* as follows:

1. Generate a uniform random monic irreducible polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t.
2. Select a uniform random sequence $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ of n distinct elements of $\mathbb{F}_q$.
3. Compute the $t \times n$ matrix $\tilde{H} = \{h_{i,j}\}$ over $\mathbb{F}_q$, where $h_{i,j} = \alpha_j^{i-1}/g(\alpha_j)$ for $i = 1, \ldots, t$ and $j = 1, \ldots, n$.
4. Form an $mt \times n$ matrix $\hat{H}$ over $\mathbb{F}_2$ by replacing each entry $c_0 + c_1 z + \cdots + c_{m-1} z^{m-1}$ of $\tilde{H}$ with a column of t bits $c_0, c_1, \ldots, c_{m-1}$.
5. Reduce $\hat{H}$ to systematic form $(I_{n-k} \mid T)$, where I_{n-k} is an $(n-k) \times (n-k)$ identity matrix. If this fails, go back to Step 1.
6. Generate a uniform random n-bit string s.
7. Put $\Gamma = (g, \alpha_1, \alpha_2, \ldots, \alpha_n)$ and output (s, Γ) as private key and T as public key.

Classic McEliece - Encapsulation

Encapsulation

The sender generates a session key K and its ciphertext C as follows:

1. Generate a uniform random vector $e \in \mathbb{F}_2^n$ of weight t.
2. Use the encoding subroutine on e and public key T to compute C_0.
3. Compute $C_1 = \mathsf{H}(2, e)$; 2.9 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.9 for H input encodings.
5. Output session key K and ciphertext C.

*** Encoding**

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Classic McEliece - Decapsulation

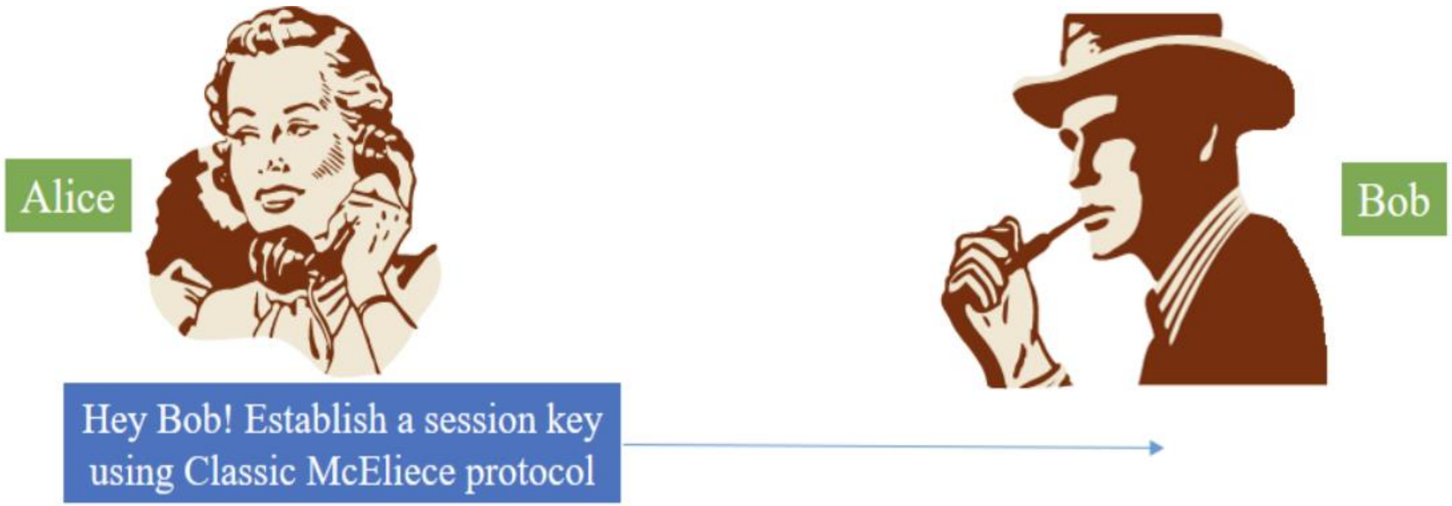
2.8 Decapsulation

The receiver decapsulates the session key K from ciphertext C as follows:

1. Split the ciphertext C as (C_0, C_1) with $C_0 \in \mathbb{F}_2^{n-k}$ and $C_1 \in \mathbb{F}_2^{\ell}$.
2. Set $b \leftarrow 1$.
3. Use the decoding subroutine on C_0 and private key Γ to compute e. If the subroutine returns $\perp$, set $e \leftarrow s$ and $b \leftarrow 0$.
4. Compute $C_1' = \mathsf{H}(2, e)$; see Section 2.9 for H input encodings.
5. If $C_1' \neq C_1$, set $e \leftarrow s$ and $b \leftarrow 0$.
6. Compute $K = \mathsf{H}(b, e, C)$; see Section 2.9 for H input encodings.
7. Output session key K.

Classic McEliece – Key Generation (1 / 10)

- Alice 는 Bob에게 Classic McEliece Protocol 을 사용하여 session key 성립을 요청



extension field $GF(2^m)$ / $k(x)$

- 이 때 extension field $GF(2^m)$ 와 유한체의 원소를 결정할 root polynomial $k(x)$ 는 공개정보

Classic McEliece – Key Generation (2 / 10)

- $GF(2^4)$ / $k(x)$

2^4 개의 유한개의 원소를 찾기위해 $X^{15} = 1$ 을 만족하는 irreducible polynomial 을 찾아야 함

$$X^{15} - 1 = (X+1)(X^2+X+1)(X^4+X+1)(X^4+X^3+1)(X^4+X^3+X^2+X+1).$$

ex) root polynomial : $k(x) = (X^4+X^3+1)$

$$\mathbb{F}_{2^4} = \frac{\mathbb{F}_2[x]}{\langle x^4 + x^3 + 1 \rangle} = \boxed{\mathbb{F}_2(\beta)}$$

Classic McEliece – Key Generation (3 / 10)

$k(x)$ → (X^4+X^3+1)

이제 $\beta^4 = \beta^3 + 1$ 을 사용하여 다음 $GF(2^4)^*$ 를 찾아낼 수 있음

Classic McEliece – Key Generation (4 / 10)

$$
\begin{array}{lcccccccccl}
1 & = & 1 & & & & & & & = & (1,0,0,0)^T; \\
\beta & = & & & \beta & & & & & = & (0,1,0,0)^T; \\
\beta^2 & = & & & & & \beta^2 & & & = & (0,0,1,0)^T; \\
\beta^3 & = & & & & & & & \beta^3 & = & (0,0,0,1)^T; \\
\beta^4 & = & 1 & & & & & + & \beta^3 & = & (1,0,0,1)^T; \\
\beta^5 & = & 1 & + & \beta & & & + & \beta^3 & = & (1,1,0,1)^T; \\
\beta^6 & = & 1 & + & \beta & + & \beta^2 & + & \beta^3 & = & (1,1,1,1)^T; \\
\beta^7 & = & 1 & + & \beta & + & \beta^2 & & & = & (1,1,1,0)^T; \\
\beta^8 & = & & & \beta & + & \beta^2 & + & \beta^3 & = & (0,1,1,1)^T; \\
\beta^9 & = & 1 & & & + & \beta^2 & & & = & (1,0,1,0)^T; \\
\beta^{10} & = & & & \beta & & & + & \beta^3 & = & (0,1,0,1)^T; \\
\beta^{11} & = & 1 & & & + & \beta^2 & + & \beta^3 & = & (1,0,1,1)^T; \\
\beta^{12} & = & 1 & + & \beta & & & & & = & (1,1,0,0)^T; \\
\beta^{13} & = & & & \beta & + & \beta^2 & & & = & (0,1,1,0)^T; \\
\beta^{14} & = & & & & & \beta^2 & + & \beta^3 & = & (0,0,1,1)^T.
\end{array}
$$

$^*\beta^4 = \beta^3 + 1$

$$
\begin{aligned}
\beta^5 &= \beta^4 \cdot \beta \\
&= (\beta^3 + 1) \cdot \beta \\
&= \beta^4 + \beta \\
&= 1 + \beta + \beta^3
\end{aligned}
$$

- 위와 같이 순환 구조의 유한체 원소 형성

Classic McEliece – Key Generation (5 / 10)

$$\mathbb{F}_2(\beta) = \{0, 1, \beta, \beta^2, \ldots, \beta^{14}\}$$

- Goppa code $\Gamma(L, g(z))$ 를 정의할 수 있음

1. Bob은 개인키로 monic & irreducible 한 t 차 다항식 $g(z)$ 를 생성,이 때 t 는 최대로 수정할 수 있는 오류의 개수

   $g(z) = z^2 + z + \beta$ → 최대 2개의 오류 수정 가능

   * message x G = codeword

   parity-check H

2. 위와 같은 유한체 $\mathbb{F}_2(\beta)$의 원소에서 랜덤하게 n개의 원소를 순서대로 선택하여 subset $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$을 구성

   $L = \mathbb{F}_2(\beta) = \{0, 1, \beta, \beta^2, \ldots, \beta^{14}\}$ $(n = 16)$

Classic McEliece – Key Generation (6 / 10)

3. t $\times$ n (2×16)의 $parity - check$ 행렬 $H = \{ h_{i,j} \}$를 계산,

$$\begin{matrix} (g_2\alpha_i + g_1)/g(\alpha_j) \\ (g_2)/g(\alpha_j) \end{matrix} \Rightarrow H = \begin{pmatrix} \boxed{(0+1)/g(\alpha_1)} & (1+1)/g(\alpha_2) & \cdots & (1+1)/g(\alpha_{16}) \\ 1/g(\alpha_1) & 1/g(\alpha_2) & \cdots & 1/g(\alpha_{16}) \end{pmatrix}$$

$$\boxed{h_{1,1}} = g(0)^{-1} = (\beta)^{-1} = \beta^{14}$$

* $g(z) = z^2 + z + \beta$

$g_1 = 1, \; g_2 = 1$

subset $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} = \{0, 1, \beta, \beta^2, \ldots, \beta^{14}\}$

$$H = \begin{pmatrix} \beta^{14} & 0 & \beta^{10} & \beta^{3} & \beta^{10} & \beta^{9} & \beta^{13} & 1 & \beta^{9} & \beta^{13} & \beta^{11} & \beta^{8} & \beta^{11} & \beta^{14} & \beta^{3} & \beta^{8} \\ \beta^{14} & \beta^{14} & \beta^{13} & \beta^{9} & \beta^{6} & \beta^{6} & \beta^{3} & \beta^{7} & \beta^{11} & \beta^{7} & \beta^{9} & \beta^{3} & \beta^{12} & \beta^{13} & \beta^{11} & \beta^{12} \end{pmatrix}$$

Classic McEliece – Key Generation (7 / 10)

4. 앞서 생성한 parity-check 행렬 H 를 각 원소에 해당하는 bit 로 변환

$$\begin{pmatrix} \beta^{14} & 0 & \beta^{10} & \beta^{3} & \beta^{10} & \beta^{9} & \beta^{13} & 1 & \beta^{9} & \beta^{13} & \beta^{11} & \beta^{8} & \beta^{11} & \beta^{14} & \beta^{3} & \beta^{8} \\ \beta^{14} & \beta^{14} & \beta^{13} & \beta^{9} & \beta^{6} & \beta^{6} & \beta^{3} & \beta^{7} & \beta^{11} & \beta^{7} & \beta^{9} & \beta^{3} & \beta^{12} & \beta^{13} & \beta^{11} & \beta^{12} \end{pmatrix}$$

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Classic McEliece – Key Generation (8 / 10)

$$
\begin{array}{lclclclclcl}
1 & = & 1 & & & & & & & = & (1,0,0,0)^T; \\
\beta & = & & & \beta & & & & & = & (0,1,0,0)^T; \\
\beta^2 & = & & & & & \beta^2 & & & = & (0,0,1,0)^T; \\
\beta^3 & = & & & & & & & \beta^3 & = & (0,0,0,1)^T; \\
\beta^4 & = & 1 & & & & & + & \beta^3 & = & (1,0,0,1)^T; \\
\beta^5 & = & 1 & + & \beta & & & + & \beta^3 & = & (1,1,0,1)^T; \\
\beta^6 & = & 1 & + & \beta & + & \beta^2 & + & \beta^3 & = & (1,1,1,1)^T; \\
\beta^7 & = & 1 & + & \beta & + & \beta^2 & & & = & (1,1,1,0)^T; \\
\beta^8 & = & & & \beta & + & \beta^2 & + & \beta^3 & = & (0,1,1,1)^T; \\
\beta^9 & = & 1 & & & + & \beta^2 & & & = & (1,0,1,0)^T; \\
\beta^{10} & = & & & \beta & & & + & \beta^3 & = & (0,1,0,1)^T; \\
\beta^{11} & = & 1 & & & + & \beta^2 & + & \beta^3 & = & (1,0,1,1)^T; \\
\beta^{12} & = & 1 & + & \beta & & & & & = & (1,1,0,0)^T; \\
\beta^{13} & = & & & \beta & + & \beta^2 & & & = & (0,1,1,0)^T; \\
\beta^{14} & = & & & & & \beta^2 & + & \beta^3 & = & (0,0,1,1)^T.
\end{array}
$$

5. 앞서 생성한 parity-check 행렬 H 를 가우스 소거(Gaussian elimination) 를 수행하여 아래과 같이 systematic form 으로 변환 → 후에 Decapsulation 시 사용됨

$$H$$

$$\begin{pmatrix}
0&0&0&0&0&1&0&1&1&0&1&0&1&0&0&0\\
0&0&1&0&1&0&1&0&0&1&0&1&0&0&0&1\\
1&0&0&0&0&1&1&0&1&1&1&1&1&1&0&1\\
1&0&1&1&1&0&0&0&0&0&1&1&1&1&1&1\\
0&0&0&1&1&1&0&1&1&1&1&0&1&0&1&1\\
0&0&1&0&1&1&0&1&0&1&0&0&1&1&0&1\\
1&1&1&1&1&1&0&1&1&1&1&0&0&1&1&0\\
1&1&0&0&1&1&1&0&1&0&0&1&0&0&1&0
\end{pmatrix}
\Rightarrow
\left(\begin{array}{cccccccc|cccccccc}
1&0&0&0&0&0&0&0&1&0&0&0&0&1&1&1\\
0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&1\\
0&0&1&0&0&0&0&0&1&1&1&1&1&0&1&1\\
0&0&0&1&0&0&0&0&0&1&0&1&1&1&0&1\\
0&0&0&0&1&0&0&0&0&0&0&1&1&1&1&0\\
0&0&0&0&0&1&0&0&1&1&0&0&1&1&1&0\\
0&0&0&0&0&0&1&0&1&0&1&1&0&1&0&0\\
0&0&0&0&0&0&0&1&0&1&1&0&0&1&1&0
\end{array}\right)$$

$$(\boxed{I_{n-k}} \mid \boxed{T})$$

Identity-matrix
$(n-k \times n-k)$

Classic McEliece – Key Generation (10 / 10)

6. T 를 공개키로 사용, Bob은 랜덤하게 n-bit 벡터 s 생성, s = (0000000000000000)

↳ 후에 Decapsulation 시 사용

7. $\Gamma = (g, \alpha_1, \alpha_2, \ldots, \alpha_n)$ 그리고 Bob 의 개인키는 (s, Γ)

- Private key

s = (0000000000000000)

$g(z) = z^2 + z + \beta$

$L = \mathbb{F}_2(\beta) = \{0, 1, \beta, \beta^2, \ldots, \beta^{14}\}$

- Public key

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Classic McEliece – Encoding (1 / 1)

- Alice는 인코딩 과정에서 두가지 입력 값이 필요 : weight – t 인 n-bit 벡터 e 그리고 공개키 T

1. $H = (I_{n-k} \mid T)$ 을 사용하여 인코딩
2. $C_0 = He^T$
3. return C_0 $(n - k)$ – bit

Challenge

$C_0 = He^T$ 라는 신드롬 계산 식에서 C_0 와 H 가 주어진다 해도

low – weight 벡터 e 를 찾아내기 매우 어려움→ Finding low-weight codeword problem

Classic McEliece – Decoding (1 / 1)

$C_0 = He^T$를 이용하여

Bob 은 수신한 C_0 를 디코딩하여 wt$(e) = t$ 인 벡터 e 를 복구해야 함

Decoding Subroutine

1. $(n - k)$ - bit 벡터 C_0 에 k – bit 만큼 zero 를 패딩하여 아래와 같은 n – bit 의 벡터 v 로 확장

$$C_0 \text{ to } v = (C_0, \underbrace{0, \ldots, 0}_{k\,-\,\text{bit}})$$

2. Bob 은 자신의 개인키 $\Gamma = (g, \alpha_1, \alpha_2, \ldots, \alpha_n)$ 를 가지고 wt$(e) = t$ 의 벡터 e 를 찾아낼 수 있음

Summary – Key Generation, Encoding, Decoding

Alice

Hey Bob! Establish a session key using Classic McEliece protocol

Bob

Generates Classic McEliece Key Pair

- Private key : $\Gamma = \{s, g(z), a_1, a_2, \ldots, a_n\}$
- Public key : T

Public key :

Random Plaintext : e

Using $H = (I_{n-k} \mid T)$,

Compute $C_0 = He^T$

Classic McEliece – Encapsulation (1 / 2)

1. Alice는 weight – t 인 n-bit 벡터 e 를 생성 ➡ $\mathbf{e} = (1100000000000000)$, length n = 16, weight t = 2

2. Public key T 를 사용하여 C_0 를 계산 ➡ $C_0 = \boldsymbol{He} = (11000000)$

3. $C_1 = \mathrm{H}(2, e)$ 를 계산, 그리고 Chipertext $C = (C_0, C_1)$ 구성

   이 때 사용하는 해시함수는 SHA-256, 해시 입력 값 2 는 Byte로 표현 (i.e. 00000010)

$C_1 = \mathrm{H}(2, \mathbf{e}) = 26fe36f811ac8fe9f19ba997a39d3682ef06b29509cca1903ffe4a0b247c833f$
00100110111111100011011011111000000100011010110010001111111010011111000110011011101
01001101011110100011100111010011011010000010111011110011010110010100101010100111001
10010100001100100011111111111110010010100101100100100011111001000001100111111.

$C = (C_0, C_1) =$
110000000010011011111111000110110111110000001000110101100100011111110100111110001100
11011101010011001011110100011100111010011011010000010111011110011010110010100101010
10011100110010100001100100011111111111110010010100101100100100011111001000001100111
111.

Classic McEliece – Encapsulation (2 / 2)

4. $K = \mathsf{H}(1, e, C)$ 를 계산

   $K = \mathtt{H}(1, \mathbf{e}, C) =$ 90d7c9dccc4689f6894b1b6e58ee9b3832 8e4df9937536eb9b5715a38ee4e1be

5. Alice는 Session key K 출력, 그리고 Ciphertext C 를 Bob 에게 전송

Classic McEliece – Decapsulation (1 / 2)

- Bob은 수신한 Ciphertext C 로부터 Session Key K 를 decapsulate 해야함

1. Ciphertext C 를 (C_0, C_1)로 나눈다.

$C = (C_0, C_1) =$
$\boxed{11000000}$001001101111111000110110111110000001000110101100100011111110100111110001100
110111010100110010111101000111001110100110110100000010111011110011010110010100101010
10011100110010100001100100011111111111100100101001011001001000111110010000001100111
111.

$C_0 = H\mathbf{e} = (11000000)$

2. Set $b \leftarrow 1$, 해시함수의 input 값으로 사용 됨

*$C_0 = H\mathbf{e}$

3. C_0 에 대해 private key $\Gamma = \{s, \boxed{g(z), a_1, a_2, \ldots, a_n}\}$ 를 사용한 디코딩 알고리즘으로 plaintext e 를 복구

   만약 디코딩 알고리즘이 $\perp$ 를 return 한다면, set $e \leftarrow s, b \leftarrow 0$

Classic McEliece – Decoding Algorithm (1 / 5)

- $C_0 = H\mathbf{e} = (11000000)$

1. $\mathbf{v} = (C_0, 00000000) = (1100000000000000)$ 와 같이 k – bit 의 zero 벡터로 패딩하여 n - bit 벡터로 확

2. 벡터 $\mathbf{V}$ 에서 t 개의 오류를 수정

   해당 Goppa code 의 원본 codeword c 가 있었고, → $Hc^T = 0$

   c 의 두 자리 bit 에 오류가 생긴 벡터가 $\mathbf{V}$ 라 가정하고, 오류수정을 진행

Classic McEliece – Decoding Algorithm (2 / 5)

3. Goppa code 를 사용한 오류수정의 핵심 방정식

$S(\mathbf{z})\sigma(z) \equiv w(z) \mod g(z)$ 을 풂으로써 오류 수정

키 생성시 Goppa Code 로 생성한 H 를 사용하여 벡터 $\mathbf{V}$ 의 Syndrome 값 계산

$\mathbf{v} = (C_0, 00000000) = (\boxed{11}00000000000000)$

$$H = \begin{pmatrix} \boxed{\begin{matrix} \beta^{14} & 0 \\ \beta^{14} & \beta^{14} \end{matrix}} & \begin{matrix} \beta^{10} & \beta^{3} & \beta^{10} & \beta^{9} & \beta^{13} & 1 & \beta^{9} & \beta^{13} & \beta^{11} & \beta^{8} & \beta^{11} & \beta^{14} & \beta^{3} & \beta^{8} \\ \beta^{13} & \beta^{9} & \beta^{6} & \beta^{6} & \beta^{3} & \beta^{7} & \beta^{11} & \beta^{7} & \beta^{9} & \beta^{3} & \beta^{12} & \beta^{13} & \beta^{11} & \beta^{12} \end{matrix} \end{pmatrix}$$

$S(\mathbf{v}) = \beta^{14}$ → Syndrome 값

$\sigma(z) = (z + \sigma_1)(z + \sigma_2)$ → 오류 위치 다항식

$w(z) = 1$ → 오류 값 (Binary 이므로 1)

Classic McEliece – Decoding Algorithm (3 / 5)

- 즉, 다음 방정식을 풂으로 써 오류 위치를 찾을 수 있음

$$\beta^{14}(z+\sigma_1)(z+\sigma_2) \equiv 1 \; mod \; g(z)$$

$$\beta^{14}z^2 + (\sigma_1+\sigma_2)z\beta^{14} + \sigma_1\sigma_2 + \underline{\beta^{14}(z^2+z+\beta)} \equiv 1 \; mod \; g(z)$$

$^{*}\; g(z) = z^2 + z + \beta$

$$(\sigma_1+\sigma_2+1)z\beta^{14} + \sigma_1\sigma_2\,\beta^{14} + \underline{\beta^{15}} \equiv 1 \; mod \; g(z)$$

$$\sigma_1+\sigma_2 = 1, \quad \sigma_1\sigma_2 = 0$$

$$\therefore \; \sigma_1 = 0, \; \sigma_2 = 1$$

오류 위치에 따라 수신한 벡터 $\mathbf{v}$ 의 오류를 수정하여 Goppa code 의 원본 codeword 를 복구할 수 있음

$\mathbf{v}$ = (1100000000000000) → $\mathbf{c}$ = (0000000000000000)

Classic McEliece – Decoding Algorithm (4 / 5)

- Question. 다음과 같은 오류 수정을 하여 c 를 구하는 이유?

Bob은 수신한 C_0 로부터 plaintext $\mathbf{e}$ 를 복구해야 함

$$C_0 = H\mathbf{e}$$

디코딩 알고리즘을 모르는 수신자는 low-weight codeword $\mathbf{e}$ 를 찾아내기 매우 어려움

→ Finding low-weight codeword problem

하지만 디코딩 알고리즘을 사용하면 아래 식으로 간단하게 복구 가능

$$\mathbf{e} = \mathbf{v} + \mathbf{c} = (110000000000000)$$

Classic McEliece – Decoding Algorithm (5 / 5)

$\mathbf{e} = v + c = (1100000000000000)$ $C_0 = \boldsymbol{H}\mathbf{e} = (11000000)$

$$He = H(v+c) = Hv + \boxed{Hc} = Hv = C_0$$

↳ 원본 코드워드의 syndrome 값은 0

$Hv = C_0$?

∵ 앞서 $H = (I_{n-k} \mid T)$ 그리고 $\mathbf{v} = (C_0, 00000000) = (1100000000000000)$

마지막으로 c 는 v 에 대한 오류수정으로 t 개의 bit 가 수정 되었기 때문에 $v + c$ 의 weight 는 t

Classic McEliece – Decapsulation (2 / 2)

- Alice의 Encapsulation 과정과 동일하게 Sesseion key K를 Bob 또한 획득하여 키 교환이 완료

4. $C_1' = \mathsf{H}(2, e)$ 를 계산, 만약 $C_1' \neq C_1$ 라면, $\text{set } e \leftarrow s$, $b \leftarrow 0$
5. $K = \mathsf{H}(b, e, C)$ 를 계산
6. Session key K 설립

구현의 관점에서 보았을 때, 부채널 공격으로 인한 비밀정보의 노출을 피해야 함

연산과정에서 Success 와 failure 구분의 특징이 드러나지 않기 위해

failure ($\perp$) 를 반환해도 연산을 멈추지 않고 계속 진행하기를 권장

Classic McEliece – Conclusion

Alice

Hey Bob! Establish a session key using Classic McEliece protocol

Bob

Generates Classic McEliece Key Pair

- Private key : $\Gamma = \{s, g(z), a_1, a_2, \ldots, a_n\}$
- Public key : T

Public key :

Random Plaintext : e
Using $H = (I_{n-k} \mid T)$
Compute $C_0 = He^T$
Session Key : $K = H(1, e, C)$

Ciphertext: C

Decrypt C to obtain e
Session Key : $K = H(1, e, C)$

Q & A