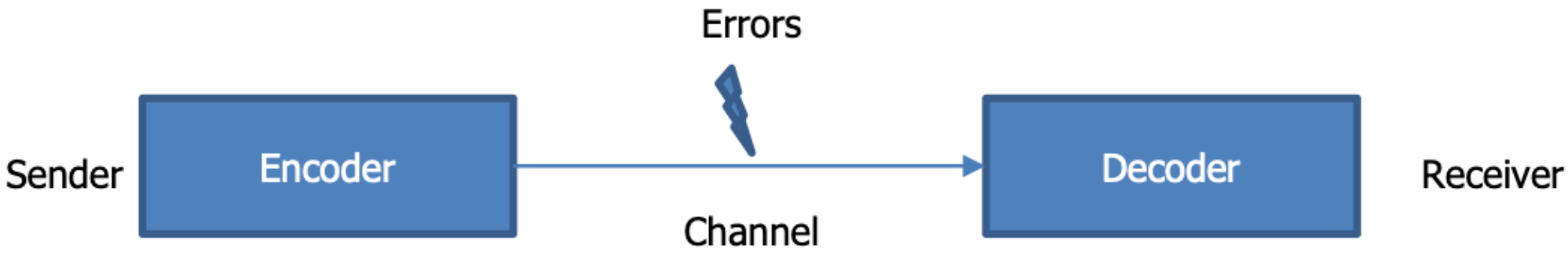
Coding Theory and Classic McEliece

https://youtu.be/uUNF3GUJLCE

Coding Theory

- 오류정정부호는 coding Theory(부호 이론)에서 많이 다루게 된다.

- 송신자가 메세지를 부호 이론에 의해 만들어진 코드로 인코딩
- 채널을 통해서 수신자에게 전달될 때 여러 채널의 특징으로 인해 Error가 발생
- 수신자는 디코더를 이용해 Error를 제거하고 원래의 메세지만 복원
  - Redundancy를 통해서 이런 기술을 구현 (원래 메세지보다 더 추가된 어떠한 메세지를 통해)

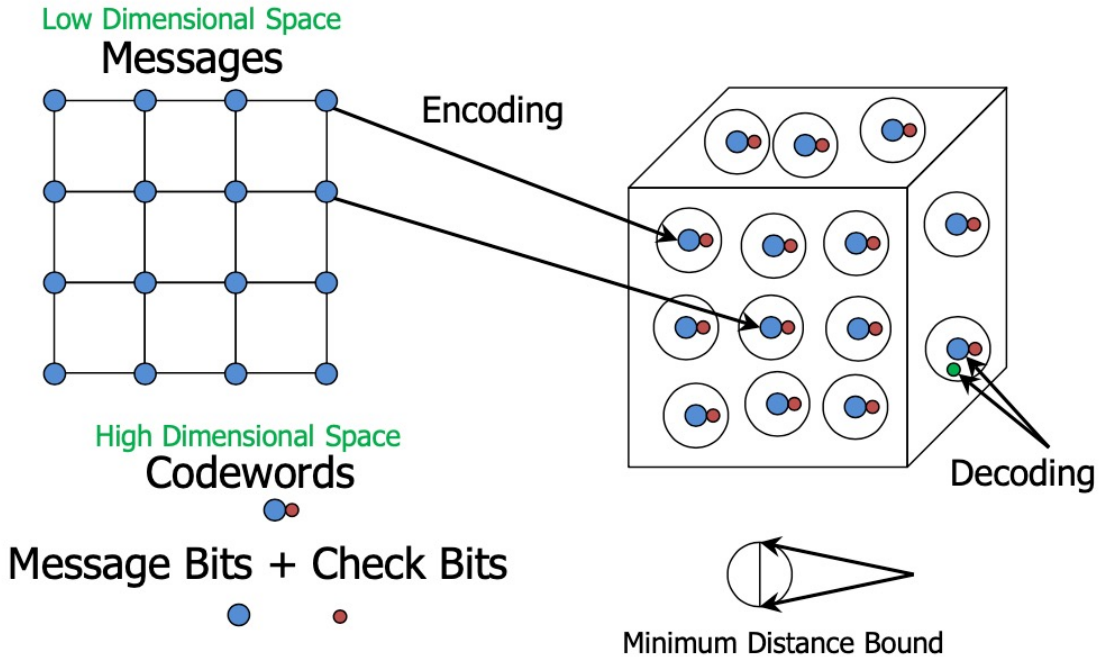


출처 : https://www.youtube.com/watch?v=y7VEis32Yck&list=PLdOq9g7U6PdscJE-2yRv37bGGBZptBG_j&index=4 (2021년 양자내성 암호교육/Introduction to Code 김영식 교수)

Coding Theory

- 선형 블록 코드
  - 코드 길이 : n GF(q)
  - 메세지의 길이 : k
  - 〔n, k〕 선형 블록 코드

  - Codewords : n차원으로 확장된 메세지



Coding Theory

- 〔n, k〕 binary linear block code → Γ.
  - Γ 는 길이가 n인 vector의 집합으로 구성
  - k개의 선형 독립인 벡터들을 사용하여 선형 결합으로 길이가 n인 codewords를 2^k개 생성
    - $g_1, g_2, \cdots, g_k$

$$c = m_0 g_1 + m_1 g_2 + \ldots + m_{k-1} g_k = \begin{pmatrix} m_0 & m_1 & \ldots & m_{k-1} \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \\ \ldots \\ g_k \end{pmatrix} = mG$$

Coding Theory

- G(Generator matrix)에 대응이 되는 H(Parity-check matrix)가 항상 결정 가능
- 두 matrix의 특징은 $GH^T = 0$ (or $HG^T = 0$)

Ex) For a linear block code,

$$G = [I_k; P] = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \Leftrightarrow H = [-P^T; I_r] = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Syndrome computation of a received vector r:

$$s^T = Hr^T = H\left(c^T + e^T\right) = He^T$$

Classic McEliece

- 기존 McEliec와 Niederreiter의 장점들을 합쳐서 설계
  - 기존 McEliec에서는 공개키를 Generator matrix를 사용함
  - 이를 Niederreiter에서 사용하는 공개키 parity check matrix로 변경

- IND-CCA KEM으로 한번 생성한 키를 여러번 반복 사용 가능
- Syndrome decoding problem에 기반을 두고 있는 암호

	McEliece 348864	McEliece 460896	McEliece 6688128	McEliece 6960119	McEliece 8192128
(n, m, t)	(3488,12,64)	(4608,13,96)	(6688,13,128)	(6960,13,119)	(8192,13,128)
Public Key (bytes)	261,120	524,160	1,044,992	1,047,319	1,357,824
Private Key (bytes)	6,452	13,568	13,892	13,908	14,080
Ciphertext (bytes)	128	188	240	226	240
KeyGen (cycles)	36,627,388	116,914,656	284,468,140	246,291,008	316,116,640
Encapsulation (cycles)	43,832	115,540	149,080	159,116	177,480
Decapsulation (cycles)	134,184	270,856	322,988	300,688	325,744

출처 : https://www.youtube.com/watch?v=y9LALjlNUp0&list=PLdOq9g7U6PdscJE-2yRv37bGGBZptBG_j&index=8 (2021년 양자내성 암호교육/Classic McEliece 김영식 교수)
출처 : https://www.youtube.com/watch?v=mYZEZpmU4gQ&list=PLdOq9g7U6Pdt5ZlWeffEU-ViDUS6j-jFS&index=371 (연구실 세미나/ Classic McEliece 장박사)

Classic McEliece

- Key Generation
  - Degree가 t인 g(x)를 생성
  - Support Set(L) $(a_1, a_2, \cdots, a_n)$ code의 길이 만큼 생성 //Goppa code 생성
  - t * n matrix H = $\{h_{i,j}\}$를 생성 할 수 있음 //parity check matrix
    - $h_{i,j} = a_j^{i-1}/g(a_j)$ i = 1, ⋯, t / j = 1, ⋯, n
  - 생성된 H를 binary 형태로 변경
    - t * n → mt * n
  - H를 Gaussian elimination을 수행하여 systematic form으로 변환
    - H를 $(I_{n-k}|T)$ 형태로 변환 Identity-matrix는 자르고 T 행렬을 공개키로 사용 → Key size 감소
  - 랜덤한 s를 생성
  - Γ = (g, L) → (s, Γ)를 개인키로 사용

Classic McEliece

- Encapsulation
  - Mceliece 암호지만 Niederreiter 타입
    - Niederreiter는 m(메세지) → e(error) 변화 시키는데 이 과정이 오래걸림
  - weight t 인 random vector e를 생성
  - e와 T(공개키)를 사용하여 C_0 (신드롬에 해당) 를 생성
  - C_1 = Hash(2, e)
  - C = (C_0 , C_1)
  - K = Hash(1, e, C)

- 암호문 C 를 전송

Classic McEliece

- Decapsulation
  - C = (C_0, C_1) 이며 C_0는 신드롬임.
    - goppa code의 Syndrome Decoding을 이용해 거꾸로 복호화 가능
  - C_0 와 Γ 를 활용하여 e를 찾을 수 있음.
    - C_1' 을 계산 C_1' = Hash(2, e)
  - C_1'과 C에서 분리한 C_1 을 비교
    - 서로 같지 않을 경우 b=0 같으면 b=1
  - K = H(b,e,C)

- 이를 통해 수신자와 송신자가 같은 Session Key K를 공유할 수 있다.

Classic McEliece

- 장점
  - 오래됨 → 오래된 만큼 연구가 많이 진행되었지만 깨지지 않았음
  - 매우 짧은 Ciphertext를 갖는다
  - Encapsulation / Decapsulation이 빠르다

- 단점
  - 공개키가 너무 길다
  - 키 생성이 상대적으로 느리다(하지만 여러번 반복 사용할 수 있기 때문에 상쇄가능)

감 사 합 니 다