Classic McEliece 규격 및 파라미터별 성능에 관한 연구

최장혁, 박민진, 김동찬
국민대학교 정보보안암호수학과

{jhdh0813, minjin_99, dckim}@kookmin.ac.kr

A Study on Classic McEliece specification and performance by parameter

Janghyuck Choi, Minjin Park, Dong-Chan Kim

Kookmin University

요 약

1978년 McEliece는 부호기반 공개키 암호 시스템을 제안했다. 제안된 암호는 OW-CPA를 만족하며, 이진 Goppa 부호의 생성행렬에 가역행렬과 치환행렬을 곱하여 공개키를 생성한다. 지난 40년간 부호기반 공개키 암호 시스템에 대해 다양한 공격 기법이 제안되었으며, McEliece는 안전성을 유지하기 위해 키 공간을 확장했다. Classic McEliece는 Niederreiter 암호시스템과 같이 이진 Goppa 부호의 패리티 검사 행렬을 공개키로 사용한다. 해당 암호는 IND-CCA2를 만족하며, NIST 양자내성 암호 공모전 3라운드 공개키암호화/키체결 알고리듬 분야에서 최종 후보 중 하나로 선정되었다. 본 논문에서는 Classic McEliece의 규격과 파라미터별 성능에 대해 소개한다.

Ⅰ. 서론

1978년 McEliece는 부호기반 공개키 암호 시스템을 제안했다. 제안된 암호는 OW-CPA를 만족하며 신드롬 디코딩 문제의 어려움에 기반한다. McEliece는 이진 Goppa 부호의 생성행렬에 가역행렬과 치환행렬을 곱하여 공개키를 생성한다[5].

지난 40년간 McEliece를 비롯한 부호기반 공개키 암호 시스템에 대해 다양한 공격 기법이 등장했다. 그 중 가장 효과적으로 알려진 정보집합 디코딩은 현재까지 27종 이상 제안되었다. McEliece는 기존 및 양자 컴퓨터 환경에서 충분한 안전성을 유지하기 위해 키 공간을 확장했다[1].

Classic McEliece는 Niederreiter 암호시스템과 같이 이진 Goppa 부호의 패리티 검사 행렬로 공개키를 생성한다. 제안된 암호는 IND-CCA2를 만족하는 KEM(Key Encapsulation Mechanism)이며, NIST가 개최한 양자 내성 암호 공모전 3라운드에서 공개키암호화/키체결 알고리듬 최종 후보 중 하나로 선정되었다.

본 논문의 구성은 다음과 같다. Ⅱ절에서 기호를 정의한다. Ⅲ절에서는 Classic McEliece 알고리듬을 설명한다. Ⅳ절에서는 Classic McEliece에서 제안하는 파라미터별 성능을 소개한다.

Ⅱ. 기호 및 정의

본 논문은 다음의 기호를 사용한다.

- q $2^m\ (m = 12, 13)$
- $\mathbb{F}_2$ 원소의 개수가 2인 유한체
- $\mathbb{F}_q$ 원소의 개수가 q인 유한체
- $\mathbb{F}_2^n$ $\mathbb{F}_2$에서 정의된 길이 n의 열벡터 집합
- $\mathbb{F}_2^{(n-k)\times k}$ $\mathbb{F}_2$에서 정의된 $n-k$개의 행, k개의 열로 이루어진 행렬 집합
- c_i $\mathcal{H}\left(\in \mathbb{F}_2^{(n-k)\times n}\right)$의 i번째 행에서 처음으로 0이 아닌 원소의 열 인덱스 값
- $\mathrm{I_r}$ $r \times r$ 항등행렬
- $w_H(e)$ 벡터 e의 해밍 무게
- $\lceil x \rceil$ 실수 x 보다 크거나 같은 최소 정수
- α_i, α_i' $\mathbb{F}_q$의 원소

Ⅲ. Classic McEliece

Classic McEliece는 시드값 $\delta \in \mathbb{F}_2^{256}$로 생성한 길이 n, 차원 $k(=n-mt)$, 최대 t개의 오류를 정정할 수 있는 이진 Goppa 부호 Γ를 개인키로 사용한다. 공개키는 표준형 행렬로 변환한 패리티 검사 행렬 $\mathcal{H} = [I_{n-k} \mid T] \in \mathbb{F}_2^{(n-k)\times n}$ 의 부분행렬 $T \in \mathbb{F}_2^{(n-k)\times k}$이다. Classic McEliece는 다음의 함수를 사용한다.

- G 해시함수 SHAKE256의 첫 $n+32q+16t+256$ bits 출력값.
- H 해시함수 SHAKE256의 첫 256 bits 출력값.
- FieldOrder 서로 다른 q개 원소 $(\alpha_1, \ldots, \alpha_q)$ 생성 알고리듬.
- GenIrr t차 일계수 기약다항식 $g \in \mathbb{F}_q[x]$생성 알고리듬.
- GenMat $(T, c_{n-k-\mu+1}, \ldots, c_{n-k}, \Gamma')$생성 알고리듬 $(\Gamma' = (g, \alpha_1', \ldots, \alpha_n'))$.

Ⅲ-1. 키 생성 알고리듬

Classic McEliece 키 생성 함수는 알고리듬 1이다.

알고리듬1: Classic McEliece 키 생성

입력: $\delta \in \mathbb{F}_2^l,\ m, n, t, q, k, \mu, \nu \in \mathbb{Z}$ ▷ $(\mu,\nu) = (0,0)$ or $(\mu,\nu) = (32,64)$
출력: 공개키 $T \in \mathbb{F}_2^{(n-k)\times k}$, 개인키 $(\delta, c, g, \alpha, s)$

1: $E \leftarrow G(\delta)$ ▷ $E \in \mathbb{F}_2^{n+32q+16t+256}$
2: $\delta', s \leftarrow E_{[n+32q+16t:]}$, $E_{[:n]}$
3: $(\alpha_1, \ldots, \alpha_q) \leftarrow$ FieldOrder $(E_{[n:n+32q]}, m, q)$
4: **if** line 3 fails **then**
5: Set $\delta \leftarrow \delta'$ and go to line 1.
6: $g \leftarrow$ GenIrr $(E_{[n+32q:n+32q+16t]}, m, q, t)$
7: **if** line 6 fails **then**
8: Set $\delta \leftarrow \delta'$ and go to line 1.
9: $\Gamma \leftarrow (g, \alpha_1, \alpha_2, \ldots, \alpha_n)$
10: $(T, c_{n-k-\mu+1}, \ldots, c_{n-k}, \Gamma') \leftarrow$ GenMat (Γ, m, n, k, t)
11: **if** line 10 fails **then**
12: Set $\delta \leftarrow \delta'$ and go to line 1.
13: $c \leftarrow (c_{n-k-\mu+1}, \ldots, c_{n-k})$
14: $\alpha \leftarrow (\alpha_1', \ldots, \alpha_n', \alpha_{n+1}, \ldots, \alpha_q)$
15: **return** 공개키 T, 개인키 $(\delta, c, g, \alpha, s)$

Classic McEliece는 $\mathbf{in-place\ Beneš\ network}$ 알고리듬에 사용되는 control bits만 저장하여 $(\alpha_1, \ldots, \alpha_q)$를 표현한다[3].

Ⅲ-2. Encapsulation 및 Decapsulation

Classic McEliece는 동일한 세션키의 공유를 위해 Encapsulation과 Decapsulation을 사용한다. 송신자는 Encapsulation 함수에서 오류 벡터와 공개키로 암호문을 생성하며, 오류 벡터와 암호문으로 세션키를 계산한다. 수신자는 Decapsulation 함수에서 암호문을 개인키로 복호화해 오류

벡터를 복구하며, 오류 벡터와 암호문으로 송신자의 세션키와 동일한 세션키를 계산한다. Classic McEliece에서 사용하는 Encapsulation은 알고리듬 2이다.

알고리듬2: Classic McEliece Encapsulation

입력: 공개키 $T \in \mathbb{F}_2^{(n-k)\times k}$, $n,k,t \in \mathbb{Z}$
출력: 암호문 $C \in \mathbb{F}_2^{n-k+256}$, 세션키 $K \in \mathbb{F}_2^{256}$

1: Generate error-vector $e \in \mathbb{F}_2^n$ with $w_H(e) = t$.
2: $C_0 \leftarrow [I_{n-k} \mid T] \times e$ ▷ $C_0 \in \mathbb{F}_2^{n-k}$
3: $C_1 \leftarrow H(2 \,||\, e)$ ▷ $C_1 \in \mathbb{F}_2^{256}$
4: $C \leftarrow (C_0 \,||\, C_1)$
5: $K \leftarrow H(1 \,||\, e \,||\, C)$
6: **return** 암호문 C, 세션키 K

Classic McEliece에서 사용하는 Decapsulation은 알고리듬 3이다.

알고리듬3: Classic McEliece Decapsulation

입력: 암호문 $C \in \mathbb{F}_2^{n-k+256}$, 개인키$(s,\Gamma')$, $n,k,t \in \mathbb{Z}$
출력: 세션키 $K \in \mathbb{F}_2^{256}$,

1: $(C_0, C_1) \leftarrow C$
2: $b \leftarrow 1$
3: $e \leftarrow \text{Decode}(C_0, \Gamma')$ ▷ $e \in \mathbb{F}_2^n$, $w_H(e) = t$
4: **if** line 3 fails **then**
5: $e, b \leftarrow s, 0$
6: $C_1' = H(2||e)$ ▷ $C_1' \in \mathbb{F}_2^{256}$
7: **if** $C_1' \neq C_1$ **then**
8: $e, b \leftarrow s, 0$
9: $K \leftarrow H(b||e||C)$
10: **return** 세션키 K

Decode 함수는 길이 k의 영벡터 0_k로 $v = (C_0||0_k) \in \mathbb{F}_2^n$를 생성한다. C가 올바른 암호문일 경우 다음이 성립한다.

$$\begin{aligned} \mathcal{H}v &= [I_{n-k}|T] \times (C_0||0_k) \\ &= I_{n-k} \times C_0 + T \times 0_k \\ &= C_0 + 0_{n-k} \qquad = C_0 = \mathcal{H}e. \end{aligned}$$

따라서 $\mathcal{H}v = \mathcal{H}e$가 성립하며, $c = v + e \in \mathbb{F}_2^n$는 다음으로 인해 부호어가 된다.

$$\mathcal{H}v - \mathcal{H}e = \mathcal{H}(v-e) = \mathcal{H}(v+e) = 0.$$

Γ의 최소 거리는 $2t+1$ 이상이므로 v로부터 거리가 t이내인 부호어 c가 유일하게 존재한다. Line 3의 Decode 함수는 Berlekamp-Massey 알고리듬을 사용하여 c를 계산하며, $e = v + c$ 로 오류 벡터를 복구한다[2,4]. Encapsulation과 Decapsulation은 동일한 해시함수를 사용하므로 $H(2||e) = C_1' = C_1$ 이 성립한다. 따라서 line 9에선 $b = 1$ 이다. $H(b||e||C) = H(1||e||C) = K$이므로 수신자와 송신자는 동일한 세션키 K를 공유한다.

올바른 암호문이 아닌 경우 디코딩 함수는 실패를 반환한다. 이 때 연산을 종료하면 해시 함수가 작동하지 않아 timing attack과 같은 부채널 공격에 노출될 수 있다. 따라서 디코딩에 실패하여도 line 4, line 7과 같이 오류벡터 e와 상수 b에 시드값 s와 0을 저장하여 연산을 진행하며, 잘못된 세션키 K를 반환한다.

Ⅳ. Classic McEliece 파라미터별 성능

Classic McEliece는 $(\mu,\nu) = (0,0)$ 을 사용하는 5개의 systematic form 파라미터 집합, $(\mu,\nu) = (32,64)$를 사용하고 파라미터 이름에 f를 붙인 5개의 (μ,ν)-semi-systematic form 파라미터 집합을 제안한다. 제안된 파라미터 집합은 [표 1]과 같다.

[표 1] Classic McEliece 파라미터 집합

파라미터	m	n	t
mceliece348864, mceliece348864f	12	3488	64
mceliece460896, mceliece460896f	13	4608	96
mceliece6688128, mceliece6688128f	13	6688	128
mceliece6960119, mceliece6960119f	13	6960	119
mceliece8192128, mceliece8192128f	13	8192	128

각 파라미터 집합이 사용하는 공개키, 개인키, 암호문, 세션키의 크기는 [표 2]와 같다. 각 파라미터 크기를 bytes로 표현하는 방식은 다음과 같다.

- 공개키 T $(n-k)\lfloor k/8 \rfloor$ bytes
- 개인키 δ: 32 bytes , c: $\lceil v/8 \rceil$ bytes, g: $t\lceil m/8 \rceil$ bytes, α: $\lceil (2m-1)2^{m-4} \rceil$ bytes, s: $\lceil n/8 \rceil$ bytes
- 암호문 C $\lceil (n-k+256)/8 \rceil$ bytes
- 세션키 K 32 bytes

μ, ν는 키 및 암호문 공간에 영향을 주지 않는다. 따라서 (μ,ν)-semi-systematic form과 systematic form의 키 및 암호문 공간은 동일하다.

[표 2] 파라미터별 키 및 암호문 공간 (단위: byte)

파라미터	공개키	개인키	암호문	세션키
mceliece348864, mceliece348864f	261,120	6,492	128	32
mceliece460896 mceliece460896f	524,160	13,608	188	32
mceliece6688128, mceliece6688128f	1,044,992	13,932	240	32
mceliece6960119, mceliece6960119f	1,047,319	13,948	226	32
mceliece8192128, mceliece8192128f	1,357,824	14,120	240	32

[표 3]은 Intel Haswell CPU core 상에서 구현한 Classic McEliece systematic form 파라미터 집합에 대한 키 생성 및 암호화, 복호화 중앙값 연산 시간을 나타낸다[1].

[표 3] 파라미터별 중앙값 연산 시간 (단위: CPU cycle 수)

파라미터	키 생성	암호화	복호화
mceliece348864	46,526,112	43,832	134,184
mceliece348864f	36,627,388		
mceliece460896	158,155,696	115,540	270,856
mceliece460896f	116,914,656		
mceliece6688128	458,561,448	149,080	322,988
mceliece6688128f	284,468,140		
mceliece6960119	330,214,944	159,116	300,688
mceliece6960119f	246,291,008		
mceliece8192128	409,854,088	177,480	325,744
mceliece8192128f	316,166,640		

Classic McEliece는 공개키 생성시 패리티 검사 행렬 $\mathcal{H}$를 reduced-row echelon form으로 변형한다. 이때 systematic form 파라미터 집합은 변형된 행렬이 표준형 행렬이 아닐 시 다른 부호를 선택해 다시 키를 생성한다. 반면 (μ,ν)-semi-systematic form 파라미터 집합은 표준형 행렬이 아니어도 경우에 따라 적절한 치환 행렬을 곱해 공개키로 사용한다. 따라서 (μ,ν)-semi-systematic form 파라미터 집합의 키 생성 실패 확률은 2^{-30}이하로, systematic form 파라미터 집합보다 낮다[1].

Ⅴ. 결론

본 논문에서는 Classic McEliece의 규격과 키 공간 및 성능을 소개하였다. 기존 McEliece는 이진 Goppa 부호를 효율적으로 디코딩 하는 Patterson 디코딩 알고리듬을 사용하였고, Classic McEliece는 Berlekamp-Massey 알고리듬을 사용한다. 향후 두 디코딩 알고리즘을 구현하여 속도 및 효율성을 비교할 예정이다.

참 고 문 헌

[1] Daniel J. Bernstein, et al. "Classic McEliece: conservative code-based cryptography, " NIST submission, Oct. 2020

[2] Daniel J. Bernstein, "McBits: Fast constant-time code-based cryptography, " International Conference on Cryptographic Hardware and Embedded Systems, pp.250-272, 2013

[3] Daniel J. Bernstein, et al. "Verified fast formulas for control bits for permutation networks," IACR Cryptol. ePrint Arch. 2020

[4] T. Chou, "McBits revisited," International Conference on Cryptographic Hardware and Embedded Systems, pp.213-231, Sep. 2017

[5] R. J. McEliece, "A Public-key cryptosystem based on algebraic," Coding Thv 4244, pp.114-116, Jan. 1978