이진 Goppa 부호 구현에 관한 연구

전창열, 김동찬
국민대학교

{chjeon96, dckim}@kookmin.ac.kr

A Study on Implementation of Binary Goppa Code

Chang-Yeol Jeon, Dong-Chan Kim
Kookmin Univ.

요 약

이진 Goppa 부호는 NIST 주관 양자내성암호 표준제정사업의 최종 후보인 Classic McEliece에서 개인키로 사용하는 부호이다. 이진 Goppa 부호는 패리티 검사 행렬을 효율적으로 생성하고, Patterson 디코딩 알고리듬을 이용하여 빠른 디코딩이 가능하다. 본 논문에서는 이진 Goppa 부호의 정의와 인코딩, 디코딩 과정을 설명하고, Classic McEliece에서 제안한 파라미터에 대해 구현한 이진 Goppa 부호의 연산 시간 측정 결과를 소개한다.

Ⅰ. 서 론

Classic McEliece는 NIST가 진행 중인 양자내성암호 표준제정사업에서 최종후보로 선정된 유일한 부호기반암호이다[3]. 해당 암호에서는 V.D.Goppa가 제안한 이진 Goppa 부호를 개인키로 사용한다[1]. 패리티 검사 행렬 생성시 연립 선형 동차방정식을 이용하는 일반적인 부호와 달리 이진 Goppa 부호는 정의에서 사용하는 인스턴스를 이용하여 효율적으로 패리티 검사 행렬을 생성할 수 있다[2]. 또한 Patterson 디코딩 알고리듬을 이용하여 빠른 디코딩 연산이 가능하다[4].

본 논문에서는 이진 Goppa 부호의 생성, 인코딩, 디코딩 과정을 소개한다. 그리고 SAGE로 구현한 각 과정의 시간 측정 결과를 소개한다. 구현은 Classic McEliece에서 제안한 5개의 파라미터에 대해 이루어졌다. 측정 결과 파라미터 m, n, t에 영향을 받아 연산 시간이 증가하였고, 디코딩은 t에 더 많은 영향을 받았다.

본 논문의 구성은 다음과 같다. Ⅱ절에서는 사용하는 기호를 정의한다. Ⅲ절에서는 이진 Goppa 부호의 정의와 생성 과정을 소개한다. Ⅳ절에서는 이진 Goppa 부호의 인코딩, 디코딩 과정에 대해 설명한다. Ⅴ절에서는 SAGE를 이용하여 이진 Goppa 부호를 구현한 결과를 소개한다.

Ⅱ. 기호

본 논문에서는 다음의 기호를 사용한다.

- F_{2^m} 2^m개의 원소를 갖는 유한체
- F_2^n F_2에서 정의한 n차원 벡터공간
- $F_{2^m}[X]$ F_{2^m}으로 정의한 다항식 환
- $F_{2^m}^{k\times n}$ F_{2^m}의 원소로 구성된 $k\times n$ 행렬의 집합
- $A_{m\times n}$ $m\times n$ 행렬
- I_k $k\times k$ 항등 행렬
- $\sqrt{a}$ a가 속한 유한체 상에서의 제곱근
- $\lfloor b \rfloor$ 실수 b보다 크지 않은 최대 정수

Ⅲ. 이진 Goppa 부호

이진 Goppa 부호는 정의를 위해 다음의 두 인스턴스를 사용한다.

- Support 집합 $L = \{\alpha_1, \dots, \alpha_n\} \subseteq F_{2^m}$: F_{2^m}상의 서로 다른 $n(\leq 2^m)$개의 원소를 갖는 집합.
- Goppa 다항식 $g(X) = \sum_{i=0}^{t} g_i X^i \in F_{2^m}[X]$: L의 모든 원소를 근으로 갖지 않는 분해 가능한 t차 다항식

두 인스턴스의 정의에 의해 $g(\alpha_j) \neq 0$ 이다. 따라서 $(X-\alpha_j)(X-\alpha_j)^{-1} \equiv 1 \,(mod\ g(X))$를 만족하는 $(X-\alpha_j)^{-1}$는 $t-1$차 다항식으로 항상 존재한다.

이진 Goppa 부호는 정의 1로 정의한다.

정의 1. 이진 Goppa 부호는 다음의 조건을 만족하는 F_2^n의 부분공간이다.

$$\left\{c=(c_1, c_2, \dots, c_n) \in F_2^n : \sum_{j=1}^{n} c_j (X-\alpha_j)^{-1} \equiv 0 \,(mod\ g(X))\right\}.$$

$\sum_{j=1}^{n} c_j (X-\alpha_j)^{-1}$는 $F_{2^m}[X]$의 원소이다. 하지만 이진 Goppa 부호는 F_2^n의 부분 공간이다. 즉, F_{2^m}의 부분체인 F_2의 원소만을 고려하여 부호를 구성한다. 이러한 특성을 가지는 부호를 부분체부분부호(subfield subcode)라 한다.

이진 Goppa 부호는 n개의 연립 선형 동차방정식으로 패리티 검사 행렬 $H'(\in F_{2^m}^{t\times n})$을 생성하는 것이 가능하다. 하지만 $L, g(X)$를 이용하여 효율적으로 패리티 검사 행렬을 구할 수 있고, H'은 다음과 같다.

$$H'_{t\times n} = ABC.$$

$$A_{t\times t} = \begin{pmatrix} g_t & g_{t-1} & \cdots & g_1 \\ 0 & g_t & & g_2 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & g_t \end{pmatrix}, B_{t\times n} = \begin{pmatrix} \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \\ \alpha_1^{t-2} & \alpha_2^{t-2} & & \alpha_n^{t-2} \\ & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix},$$

$$C_{n\times n} = \begin{pmatrix} -g(\alpha_1)^{-1} & 0 & \cdots & 0 \\ 0 & -g(\alpha_2)^{-1} & & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -g(\alpha_n)^{-1} \end{pmatrix}.$$

이진 Goppa 부호의 원소 c는 F_2^n의 원소이다. 따라서 이후에는 F_{2^m}의 원소로 구성된 행렬 H'을 F_2의 원소로 구성된 행렬 $H(\in F_2^{mt\times n})$로 변형하여 연산을 처리한다.

H를 패리티 검사 행렬로 갖는 선형 부호의 생성 행렬이 G라 할 때, $GH^T = 0$을 만족한다. 이때 H가 표준형 행렬인 경우 정리 1을 이용하여 G를 구할 수 있다.

정리 1. $H = [I_{mt} | M_{mt\times(n-mt)}]$이면, $G = [-M^T | I_{n-mt}]$.

증명 $GH^T = [-M^T | I_{n-mt}][I_{mt} | M]^T = [-M^T | I_{n-mt}] \begin{bmatrix} I_{mt} \\ M^T \end{bmatrix}$
$= -M^T I_{mt} + I_{n-mt} M^T = 0.$ ■

H가 표준형 행렬이 아닌 경우 SHP가 표준형 행렬이 되도록 하는 가역 행렬 $S \in F_2^{mt\times mt}$와 $P \in F_2^{n\times n}$가 존재한다. $SHP =: \widehat{H}(=[I_{mt}|M'])$인 경우, $\widehat{H}$은 어떠한 선형 부호의 패리티 검사 행렬이다. $\widehat{H}$을 패리티 검사 행렬로 갖는 선형 부호의 생성 행렬을 $\widehat{G}$이라 할 때, $\widehat{G}\,\widehat{H}^T = 0$을 만족한다. 이를 통해 다음이 유도된다.

$$0 = \widehat{G}\,\widehat{H}^T = \widehat{G}\,(SHP)^T = \widehat{G}\,P^T H^T S^T.$$

S^T는 가역 행렬이므로 양변에 $(S^T)^{-1}$를 곱하면 다음이 성립한다.

$$\left(\widehat{G}\,P^T\right)H^T = 0.$$

따라서 H의 생성 행렬은 $\widehat{G}\,P^T$이다. 상기 과정을 통해 이진 Goppa 부호의 생성이 가능하다.

Ⅳ. 이진 Goppa 부호의 인코딩, 디코딩

Ⅲ절의 방법을 이용하여 $L, g(X)$로 이진 Goppa 부호를 생성한다. 이후 생성 행렬 G를 이용해 인코딩을 수행한다.

이진 Goppa 부호의 인코딩은 $k(=n-mt)$에 대해 F_2^k의 원소 m과 생성 행렬 $G_{k\times n}$를 곱하는 것으로 처리한다. 알고리듬 1는 이 방식의 수행과정을 보인다.

알고리듬 1: 인코딩 알고리듬
입력: 메시지 $m \in F_2^k$, 생성 행렬 $G \in F_2^{k\times n}$ **출력:** 부호어 $c \in F_2^n$ 1. $c \leftarrow mG$ 2. **return** c

인코딩 이후 수신자에게 c를 보내는 과정에서 오류 벡터 e가 포함된다. 따라서 수신자는 $y(=c+e)$벡터를 수신 받는다. 이는 Patterson 디코딩 알고리듬을 이용하여 오류를 정정한다.

디코딩을 위해서는 다음의 정의를 사용한다.

- 오류 벡터 $e = (e_1, e_2, \dots, e_n)$ $\quad e \in F_2^n$
- 오류위치집합 E $\quad \{i: e_i \neq 0\}$
- 수신 벡터 $y = (y_1, y_2, \dots, y_n)$ $\quad y_i = c_i + e_i,\ y \in F_2^n$
- 신드롬 다항식 $s(X)$ $\quad \sum_{i=1}^{n} y_i (X-\alpha_i)^{-1}$
- 오류 위치 다항식 $\sigma(X)$ $\quad \prod_{i\in E}(X-\alpha_i)$

이때 $\sigma(X)s(X) \equiv \sigma'(X)\ (mod\ g(X))$를 만족한다.

$g(X)$가 $F_{2^m}[X]$상의 t차 기약다항식이므로 이진 Goppa 부호는 최대 t개의 오류를 정정 가능하다. 따라서 $\sigma(X) = \sum_{i=0}^{t}\sigma_i X^i\ (\sigma_i \in F_{2^m})$로 설정한다. 이때 $\sigma(X)$는 짝수차수항과 홀수차수항으로 나누어 다음의 표현이 가능하다.

$$\begin{aligned}\sigma(X) &= \sum_{i=0}^{\lfloor t/2\rfloor}\sigma_{2i}X^{2i} + X\sum_{i=0}^{\lfloor (t-1)/2\rfloor}\sigma_{2i+1}X^{2i} \\ &= \left(\sum_{i=0}^{\lfloor t/2\rfloor}\sqrt{\sigma_{2i}}X^i\right)^2 + X\left(\sum_{i=0}^{\lfloor (t-1)/2\rfloor}\sqrt{\sigma_{2i+1}}X^i\right)^2.\end{aligned}$$

결과적으로 $\sigma(X)$는 $\sigma(X) = a(X)^2 + b(X)^2X$로 표현 가능하다. 이때 $\sigma(X)s(X) \equiv \sigma'(X)\ (mod\ g(X))$를 만족하므로 다음과 같이 정리 가능하다.

$$\begin{aligned}&(s^{-1}(X)+X)b(X)^2 \equiv a(X)^2\ (mod\ g(X)) \\ &\Rightarrow \sqrt{(s^{-1}(X)+X)}b(X) \equiv a(X)\ (mod\ g(X)).\end{aligned}$$

따라서 $\sqrt{(s^{-1}(X)+X)}$와 $g(X)$의 확장 유클리드 알고리듬으로 $a(X), b(X)$를 계산하고, $\sigma(X)$를 생성하여 오류의 위치를 찾을 수 있다. 알고리듬 2은 이 방식의 수행과정을 보인다.

Ⅴ. 파라미터 별 연산 시간

SAGE로 구현한 이진 Goppa 부호의 연산 시간 측정 환경은 다음과 같다.

- 하드웨어 환경 1.4GHz 쿼드코어 Intel Core i5, 8GB RAM, macOS Big Sur 버전 11.0.1
- SAGE 버전 SageMath9.2 (Release: 2020.10.24)

알고리듬 2: 디코딩 알고리듬
입력: 수신벡터 y, Goppa다항식 $g(X)$, Support집합 L **출력:** 부호어 $c \in F_2^n$ 1. Generator zero vector $e \in F_2^n$ 2. $s(X) \leftarrow \sum_{i=1}^{n} y_i (X-\alpha_i)^{-1}$ 3. Compute $s(X)^{-1}\ (mod\ g(X))$ 4. Compute $\sqrt{s(X)^{-1}+X}$ 5. Compute $a(X), b(X)$ such that $\sqrt{s(X)^{-1}+X}\,b(X) \equiv a(X)\ (mod\ g(X))$ $\left(\deg(a(X)) \le \left\lfloor\frac{t}{2}\right\rfloor, \deg(b(X)) \le \left\lfloor\frac{t-1}{2}\right\rfloor\right)$ 6. $\sigma(X) \leftarrow a(X)^2 + b(X)^2X$ 7. **for** $j = 1$ **to** n 8. **if** $\sigma(\alpha_j) = 0$ 9. $e_j = 1$ 10. **return** $c \leftarrow y + e$

시간 측정 시 time 라이브러리의 time 함수를 이용하고, 50회 연산에 대한 평균 연산 시간을 구하였다. 또한 Classic McEliece에서 제안한 5개의 파라미터에 대해 시간 측정이 이루어졌다[1]. 해당 파라미터에 대한 이진 Goppa 부호의 생성, 인코딩, 디코딩 연산 시간을 측정하여 [표 1]에 나타내었다.

[표 1] 파라미터 별 알고리듬 연산 시간 (단위: 초)

파라미터 (n,m,t)	이진 Goppa 부호 생성	인코딩	디코딩
Mcecliece348864 (3488,12,64)	101.07	0.76	7.39
Mcecliece460896 (4608,13,96)	443.64	1.23	13.43
Mcecliece6688128 (6688,13,128)	1185.83	1.76	24.27
Mcecliece6960119 (6960,13,119)	1399.17	1.90	22.03
Mcecliece8192128 (8192,13,128)	1739.45	6.18	26.10

전체적으로 n, m, t의 증가에 따라 연산 시간이 증가했다. 이때 Mcecliece6960119는 Mcecliece6688128에 비해 디코딩 연산만 빠른 결과를 보였다. 이는 알고리듬 2의 Line4, 5가 비교적 연산량이 많은 지수승모듈러, 확장 유클리드 알고리듬을 사용하고, 해당 알고리듬은 t의 영향을 받기 때문이다.

Ⅵ. 결론

본 논문에서는 이진 Goppa 부호의 정의와 인코딩, 디코딩 과정을 소개하고, 해당 과정을 SAGE로 구현하여 속도 측정한 결과를 나타내었다. 측정 결과 n, m, t의 증가에 따라 연산 시간이 증가했고, 디코딩 연산은 t의 영향을 더 많이 받았다.

추후에는 알고리듬 별 최적 구현 및 Classic McEliece의 구현에 대해 연구 예정이다.

참 고 문 헌

[1] Daniel J. Bernstein, et al. Classic McEliece: conservative. code-based cryptography. NIST submissions, 2017.

[2] V. D. Goppa, “A new class of linear correcting codes,” Probl. Peredach. Inform., vol. 6, pp. 24-30, Sept. 1970.

[3] NIST, “Post Quantum Cryptography - Round 3 Submissions,” July 22, 2020.

[4] N. Patterson, “The algebraic decoding of Goppa codes,” IEEE Trans. Inf. Theor., 21(2): 203-207, Sept. 2006.