FrodoKEM

정보컴퓨터공학과 권혁동

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

Learning With Errors

- 양자 환경에서도 안전하게 사용될 수 있는 기법
- 모듈러 정수군 Z에 속한 비밀 값 s를 찾아내는 풀이

$$\mathbf{s} \in \mathbb{Z}_q^n$$

- 에러(오차)가 없을 경우, 공격자는 n개의 식으로 s를 획득 가능
  - 가우시안 소거법 사용
  - 계산 복잡도 O(n)
- 에러를 포함하게 될 경우, s 내적 값에 에러 e를 포함
  - 계산 복잡도 O(n^2B+1)

Learning With Errors

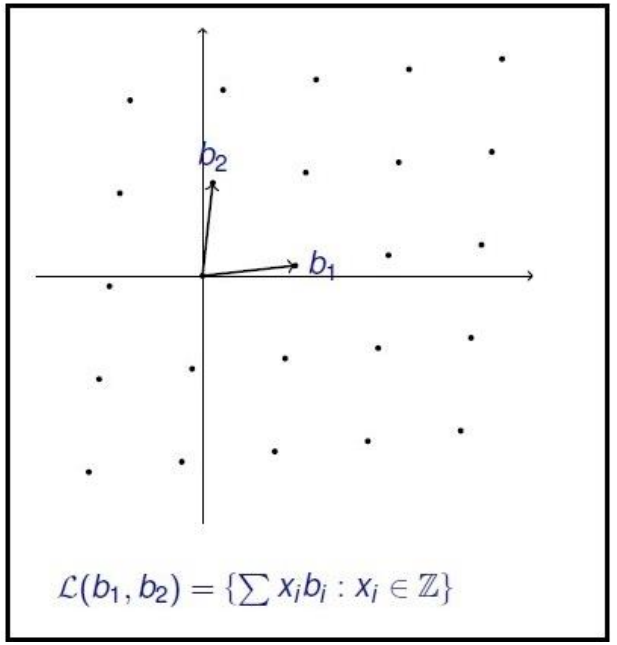
- 정수 배열 하나 선택 A[]
- 비밀 키 선택 s
- 에러 선택 e
- B[] = A[] * s + e
- 공개키 배열 B[]

$$b_A = \begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix} \times \begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \\ -1 \end{bmatrix} \pmod{13} = \begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix}$$

Lattice Based Cryptography

- Lattice(격자) 상의 수학적 난제를 기반하는 암호 알고리즘
  - **Shortest Vector Problem (SVP)**
  - Closest Vector Problem (CVP)
- SVP: 임의 차원에서 가장 가까운 Lattice Point를 찾기 어려움에 기반

- **FrodoKEM은 SVP에 기반하는 알고리즘**



FrodoKEM

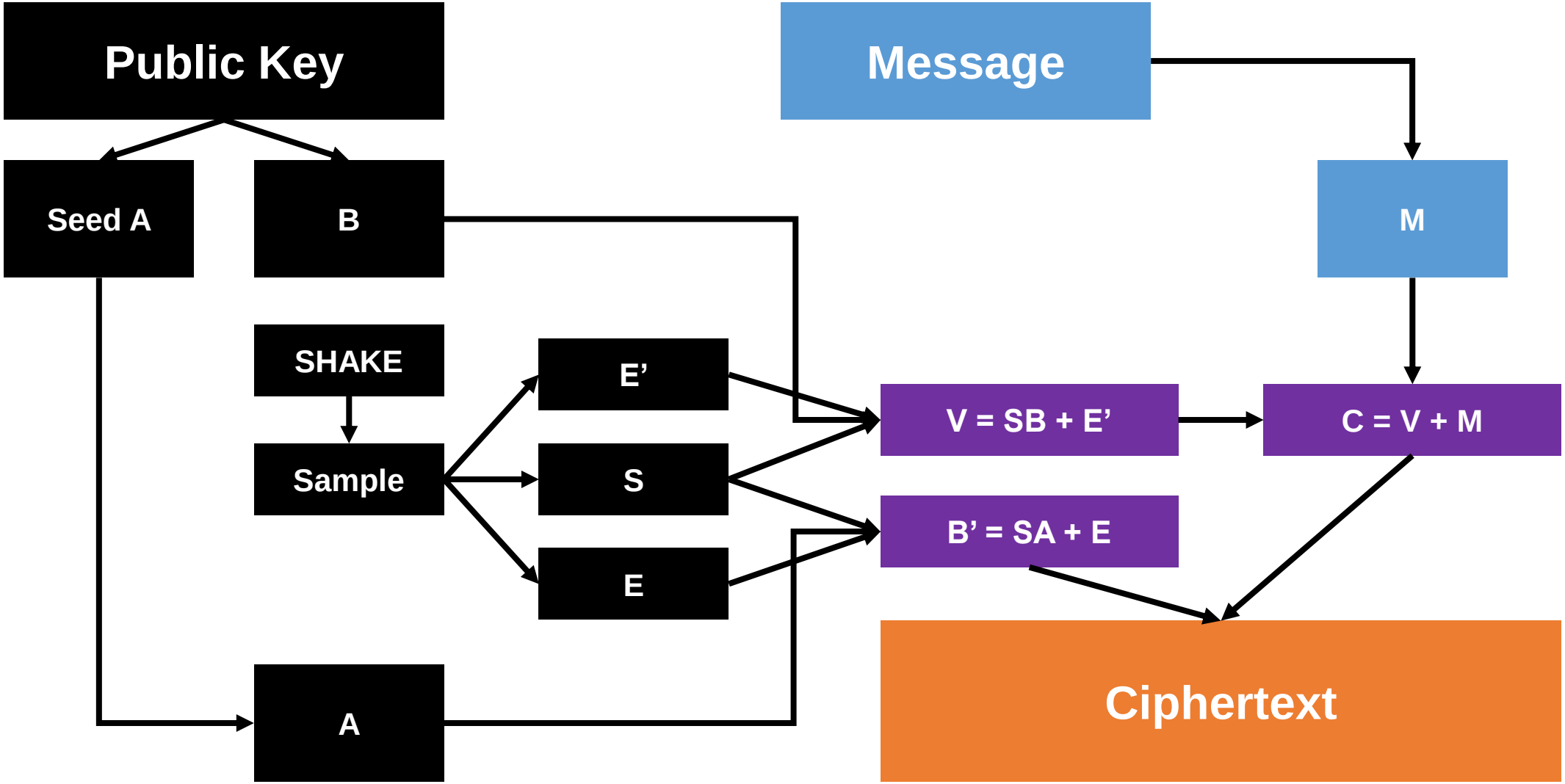
- LWE를 사용한 Lattice Based Cryptography
- NIST PQC Standardization project Round 3 alternate candidate
- NIST 요구 보안 수준 달성
  - FrodoKEM-640: NIST 요구 1레벨 (AES-128)
  - FrodoKEM-976: NIST 요구 3레벨 (AES-192)
  - FrodoKEM-1344: NIST 요구 5레벨 (AES-256)

범례	640	976	1344
Dimension n	640	976	1344
Modulus q	2^15	2^16	2^16
공개키 크기	9616-byte	15632-byte	21520-byte
개인키 크기	19888-byte	31296-byte	43088-byte
해시함수	SHAKE-128	SHAKE-256	SHAKE-256
암호문 길이	9720-byte	15744-byte	21632-byte

FrodoKEM

- 디자인 기법
- 모듈러 연산: q는 2^16보다 작거나 같은 2의 배수,
  효과적인 비트 마스킹
- 에러 샘플링: 가우시안 분포에 유사한 분포를 사용
  Look-up Table을 사용하여 빠르고 단순한 구현
  부채널 공격 방지
- 행렬 연산: 고속 **polynomial multiplication** 구현
- Reconciliation 제외: 추가한 에러를 보정하는 부분이 없음
- 짧은 코드: 250의 C 코드로 구성

FrodoKEM


Public Key
Message
Seed A
B
M
SHAKE
E'
V = SB + E'
C = V + M
Sample
S
B' = SA + E
E
Ciphertext
A

FrodoKEM

Algorithm 13 FrodoKEM.Encaps.

Input: Public key $pk = \mathsf{seed}_{\mathbf{A}} \| \mathbf{b} \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_{\mathbf{A}}} + D \cdot n \cdot \overline{n}}$.
Output: Ciphertext $\mathbf{c}_1 \| \mathbf{c}_2 \in \{0,1\}^{(\overline{m} \cdot n + \overline{m} \cdot \overline{n}) D}$ and shared secret $\mathbf{ss} \in \{0,1\}^{\mathsf{len}_{\mathsf{ss}}}$.

1: Choose a uniformly random key $\mu \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len}_{\mu}})$
2: Compute $\mathbf{pkh} \leftarrow \mathrm{SHAKE}(pk, \mathsf{len}_{\mathbf{pkh}})$
3: Generate pseudorandom values $\mathsf{seed}_{\mathbf{SE}} \| \mathbf{k} \leftarrow \mathrm{SHAKE}(\mathbf{pkh} \| \mu, \mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}} + \mathsf{len}_{\mathbf{k}})$
4: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2\overline{m}n + \overline{m}\overline{n} - 1)}) \leftarrow \mathrm{SHAKE}(0\mathrm{x}96 \| \mathsf{seed}_{\mathbf{SE}}, (2\overline{m}n + \overline{m}\overline{n}) \cdot \mathsf{len}_{\chi})$
5: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\overline{m}n - 1)}), \overline{m}, n, T_{\chi})$
6: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n - 1)}), \overline{m}, n, T_{\chi})$
7: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_{\mathbf{A}})$
8: Compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
9: Compute $\mathbf{c}_1 \leftarrow \mathsf{Frodo.Pack}(\mathbf{B}')$
10: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n + \overline{m}\overline{n} - 1)}), \overline{m}, \overline{n}, T_{\chi})$
11: Compute $\mathbf{B} \leftarrow \mathsf{Frodo.Unpack}(\mathbf{b}, n, \overline{n})$
12: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
13: Compute $\mathbf{C} \leftarrow \mathbf{V} + \mathsf{Frodo.Encode}(\mu)$
14: Compute $\mathbf{c}_2 \leftarrow \mathsf{Frodo.Pack}(\mathbf{C})$
15: Compute $\mathbf{ss} \leftarrow \mathrm{SHAKE}(\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{k}, \mathsf{len}_{\mathbf{ss}})$
16: **return** ciphertext $\mathbf{c}_1 \| \mathbf{c}_2$ and shared secret $\mathbf{ss}$

Future Work

- 레퍼런스 코드 분석
- 코드를 사용한 ARM 기본 구현물 작성 (O)
- 최적화 기법 분석 및 적용
- 논문 작성

Q & A