• 논문의 의미

↱ automorphism 으로 plain text도 랜덤, canonical & Norm 도입, §§ A.3 appendix

=> AES를 동형암호화 해보고 그과정 유명한 테크닉들 습자!

↳ 이는 BGV 성능을 매우 향상

automorphism, 갈루아군

↳ 자기자신으로 가는 사상 (함수, 곱셈) 들의 집합

ex) $f(x) = (x^2-2)(x^2-3)$ $\begin{pmatrix} \sqrt{2} & \times & = & \times & = \\ -\sqrt{2} & \times & \times & = & = \end{pmatrix}$ => 4개다.

↳ 이건 automorphism 으로 평문 아주 좋은 성질 가지게 된다.

↳ 평문에 관심

V_1 | m_1 | m_2 | m_3 | m_4 |

↓

V_2 | m_2 | m_3 | m_4 | m_1 |

⋮

V_3 | m_1+m_2 | m_2+m_3 | · | · |

=> 여러연산 병이

AES Shift 연산 바꿀

* 기존은 Cypher text에 관심있었지만 그건 AI도 신경쓰니!

미리 좋은 성질 만들어 둔다 (very Good technique)

$$\Phi(x) = \prod_{i \in (\mathbb{Z}/m\mathbb{Z})^*} (x - 3^i)$$

$F_x = F_1 \times F_2 \times F_3 \times F_4$ 인데 $\Rightarrow$ 공간을 나눠서 변관없이 효율적으로 쓸수있다.

$F_m = F$ 랑같은 걸 $F_1 = (x-3^1)$ $F_2 = (x-3^2)$ ...

성질을 만족하면 좋은데 이는 특수한 값들이 있는것이다

$(m = \bigcirc, \bigcirc = \bigcirc\!\!\bigcirc)$ = $T_m = \bigcirc$ 인경우 전부 계속 이고 + 이다.

- Cuvenlcan embeded 쓰임은

  식에 $(3^1, 3^2, 3^3, 3^4, \cdots)$ 꼭 대입해서

  이들 벡터를 가장 큰값을 의미하고 이는 noise 측정하는데 쓰인다.

$$\prod_{i=1}^{2} \mathbb{Z}[x]/f_i \simeq \mathbb{Z}_q[x]/\Phi_m(x) \longrightarrow \mathbb{Z}_q[x]/\Phi_m(x) = A_q$$

(각모건의 decomposition 확대)

plaintext space

ciphertext space

Contribution (Plane 공간도 polynomial (Lattice) 로 계산 X 대비 norm 증명

↳ poly 로 $E(a(x)+b(x))$, $E(a(x) \times b(x))$ 연산이 가능한데 우리가 증명하는 것은 아니다.

그래서 $\Phi_m(x) = \prod_{i=1}^{l} f_i \pmod p$ 로 공간나눔 $\mathbb{Z}/\Phi_m(x)$

그래서 Vector로 생성한다.

$\therefore A = (a_1(x), a_2(x), \cdots, a_2(x))$ ← slot ← slot ← slot

$B = (b_1(x), b_2(x), \cdots, b_2(x))$

그러면 $A+B \longrightarrow a(x)+b(x) \longrightarrow c_1(x)+c_2(x)$

$A \odot B \longrightarrow a(x) \times b(x) \longrightarrow \mod(c_1(x) \times c_2(x))$

(불가능했던 plane 에서 연산의 유의미 해결고, 시간 공간 잘 사용)

$Ct_{xt} \in A_q \quad \left(\mathbb{Z}_q[x]/\Phi_m(x)\right)$ ∴ Double CRT 의 의미?

↳ $\Phi_m(x)$ 를 쪼갬

⇒ 쪼개서 차수↓한 곱셈 연산을 정의

$$\mathbb{Z}_q[x]/\Phi_m(x) \simeq \prod_{i \in \mathbb{Z}_m^*} \mathbb{Z}_q[x]/(x-3^i) = \mathbb{Z}_q[x]/(x-3^1) \times \mathbb{Z}_q[x]/(x-3^2) \times \cdots$$

$$f \longmapsto (f \bmod (x-3^1),\ f \bmod (x-3^2),\ \cdots)$$

$$\downarrow \qquad \downarrow$$

$$f(3^1),\ f(3^2),$$

polynomial 을 그자체로 보는게 아니라 특별한 점들에서 evaluation 하는데 ($q \equiv 1 \pmod m$ 되는 $3 \in \mathbb{Z}_q$ 에 들어갈) ↳ 유명한 성질

저런 성질 쓰면 $3 \in \mathbb{Z}_q$ 라서 예쁘게 되네, degree 낮출 (CRT, FFT)

$\left(q = \prod_{i=1}^{L} p_i\right)$

$\mathbb{Z}_q \simeq \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_L}$ (q를 작게함)

$a \longmapsto ([a]_{p_1}, [a]_{p_2}, \cdots)$

⇒ 2개 짬뽕 (CRT / FFT) (CRT) ⇒ Double CRT가 된다, 덧셈, 곱셈 가능

$$\therefore \mathbb{Z}_q[x]/\Phi_m(x) \simeq \prod_{i \in \mathbb{Z}_m^*} \prod_{j=1}^{L} \mathbb{Z}_{p_j^{(i)}}/(x-j^i)$$

(곱셈 증가) ⇒ 쪼개고 / 차수도 ↓

(조건 $p_j = 1 \bmod j$, $3 \in \mathbb{Z}_{p_j}$)

$[5(3^1)]_{p_1}$ $[5(3^1)]_{p_2}$ $[5(3^1)]_{p_3}$ ....

$[5(3^2)]_{p_1}$

$[5(3^3)]_{p_1}$

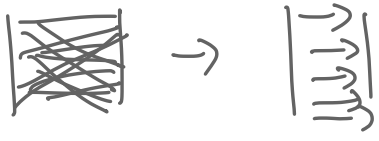
⋮

=> 이런 계산량

원래는 $(p_1 p_2 \cdots p_k)$ 를 mod 해야 했는데!

$p_1, p_2, p_3 \cdots$ 각각 소수로 해서 계산 빨라진다.

$O(n^2) \rightarrow O(n)$



=> 매우 빨라짐.

Modulus Switching

$q_L = \prod_{i=0}^{j} p_i$, $q_{L-1} = \prod_{i=0}^{j-1} p_i$ $\Rightarrow$ $\frac{q_{t+1}}{q_t} = p_L$

$c_0 + c_1 \cdot s = m + 2e \pmod{q_L}$

↳ 크기만 줄어드는가

↳ 계수들도 줄어든다.

$\frac{1}{p_L}(c_0 + c_1 s) = m + \frac{2e}{p_L} \pmod{q_{L-1}}$

↳ scale 이라는 항목도 m은 건드지 않는다 $\Rightarrow + B_{scale}$

보정값이 붙음

Key Switching

$c_0 + c_1 s = m + 2e \ [q]$

$c_0' + c_1' s = m' + 2e' \ [q]$

다항형

$c_0 c_0' + s(c_0 c_1' + c_1 c_0') + s^2(c_1 c_1') = mm' + 2\tilde{e} \ [q]$

↳ 얘가 생겨서 문제!

$Des = x(1, -s)$ 한다.

$d_0 + d_1 s + d_2 s^2 = mm' + 2\tilde{e} \approx \tilde{c_0} + \tilde{c_1} s \pmod{q}$

→ 얘를 구하고 싶다

$\tilde{c_0}$: $d_0 + \square$, $\tilde{c_1}$: $d_1 + \square$

* 러닝시간에 한 $d_2 s^2 \approx \square + \square s$ 에서 $\square$ 찾는것

relinearization (key switching)

$d_2 s^2 \approx d + d s$

↳ msg 를 생각해 s 로 Enc 한다 생각한다.

$d_2 s^2 + 2e + as - as$

↓

$b - as$

s^2 을 공개 s 로 암호화

$(m+2e+as,\ a)$

⇓ $s^2 \to$ 메세지를 보낼 때만

$(b + 2e + as,\ a)$

↓

$b - a \cdot s$ (b, a 는 evaluaton key, relinearization key)

$d_2 \dfrac{(s^2 + 2e + as - as)}{b}$ $a \xleftarrow{R} A_q$

$\therefore\ d_2 b - d_2 as$

$\to\ d_0 + d_1 s + d_2 s^2 = c_0 1 + c_1 s \pmod q$

Var 1

$c_0 + d_2 b$ ⇓ $d_1 - d_2 a$

(공개한다) (binary 로)

Var 2 $d_2 s^2 = \sum_{i=0} d_2 2^i s^2 \simeq \sum_{i=0} d_{2i} \dfrac{(2^i s^2 + 2e_i + a_i s - a_i s)}{(b, a_i)}$

Var 3 (에넉 노믈)

$d_2 s^2 \approx \square + \square \cdot s \pmod q$

↳ 곱수

$P\, d_2 s^2 = d_2 (P_2 s^2) = \qquad \pmod{Pq}$

$\approx d_2 \cdot \dfrac{(P_2 s^2 + as + 2e - as)}{(b, a)} \pmod{Pq}$

Mod switch → $(d_2 b,\ b_2 a) \bmod Pq$

$(\text{Scale}) \bmod q$

⇓

고장 자체를 P로 올리고

error 샅꽉하고

다시 P로 내림

$$\Phi(x) = \prod_{i \in (\mathbb{Z}/m\mathbb{Z})^*} (x - 3^i)$$

$F_x = F_1 \times F_2 \times F_3 \times F_4$ 인데 $\Rightarrow$ 공간을 나눠서 변관없이 효율적으로 쓸수있다.

$F_m = F$ 랑같은 걸 $F_1 = (x-3^1)$ $F_2 = (x-3^2)$ ...

성질을 만족하면 좋은데 이는 특수한 값들이 있는것이다

$(m = \bigcirc, \bigcirc = \bigcirc\!\!\bigcirc)$ = $T_m = \bigcirc$ 인경우 전부 계속 이고 + 이다.

- Cuvenlcan embeded 쓰임은

  식에 $(3^1, 3^2, 3^3, 3^4, \cdots)$ 꼭 대입해서

  이들 벡터를 가장 큰값을 의미하고 이는 noise 측정하는데 쓰인다.

다시 리뷰

메시지 공간을 잘 나눠 줄것이 중요!

식표현 ⇔ 점 표현

Double CRT ⇒ $\Phi_m(x)$ 를 나눠서 ⇒ 3 들을 점으로 생각 → FFT

↳ ∴ 속도가 빨라진다. (Memory 는 많음)

$n^2 \to n$ (매우 빨라짐)

곱혹 계속 돌 mod 연산 필요할때는 $\Phi_m(x)$ 로 갔다가 다시 $T_i(x-3)$ 로 와온다.

($q \equiv 1 \pmod m$ 이면 3 들이 모두 정수가 되어 $q \equiv 1 \pmod n$ 으로 잡아준다)

⇒ FFT, CRT 사용

Key Switching

$1 + d_1 s + d_2 s^2 = c_1 + c_2 s$

↳ $1 \to 1 + 1 s$ 로 바꾸는 방법

•

1. $d_2\left(\frac{s^2+2e+as}{b} - \frac{as}{a}\right)$, b, a 공개함 . but error를 많이 안

2. $d_2 s^2 = \sum d_i \times 2^i s^2 \approx \sum d_i (2^i s^2 + 2e + as - a_i s)$

binary

(binary or +-base)

↳ 왜 binary 쓰나? 너무 키워서 d_2를 작게 표현한다.

(Coeffi가 0 or 1를 가지기 때문에

$q \to \log q$ error는 클어둠 (장), Key는 많이 만듦 (단),

잘못 error 많이 불음)

3. 연형로 p곱함 ⇒ 상대적 에러 감소

$$P d_2 s^2 = d_2 (Ps^2) \pmod{pq}$$

$$\approx d_2 (P_2 s^2 + as + 2e - as)$$

↳ 빠름

, (Key 1개 만듦 / error 작음 / 득따리 다 잡음)