Number theoretic transform(NTT)

https://youtu.be/k33E4na8Lis

IT융합공학부 송경주

Number theoretic transform (NTT)

- **Number theoretic transform(NTT)**

- Fast Fourier transform의 도메인을 정수필드로 일반화 한 것
- **Lattice 기반 암호에서 많이 사용됨**
- 긴 다항식 곱을 효율적으로 계산할 수 있음
- 일반적으로 Karatsuba 곱셈 방법보다 더 효율적임
- 과정 : NTT변환 → pointwise 곱셈 → Inverse NTT

<n 길이의 두 다항식 곱의 계산 복잡도>

- Basic : $O(n^2)$
- Karatsuba : $O(n^{\log_2 3})$, $O(n^{1.585})$
- NTT : $O(\mathrm{nlog}\, n)$

다항식 곱셈 효율 : Basic < Karatsuba < NTT

Number theoretic transform (NTT)

- 방정식 : $a = a_0 \cdots a_m,\ b = b_0 \cdots b_m$
- 링$(x^n + 1)$ 상에서의 방정식 $a,\ b$의 곱을 구함 → **링의 차원을 낮춤**

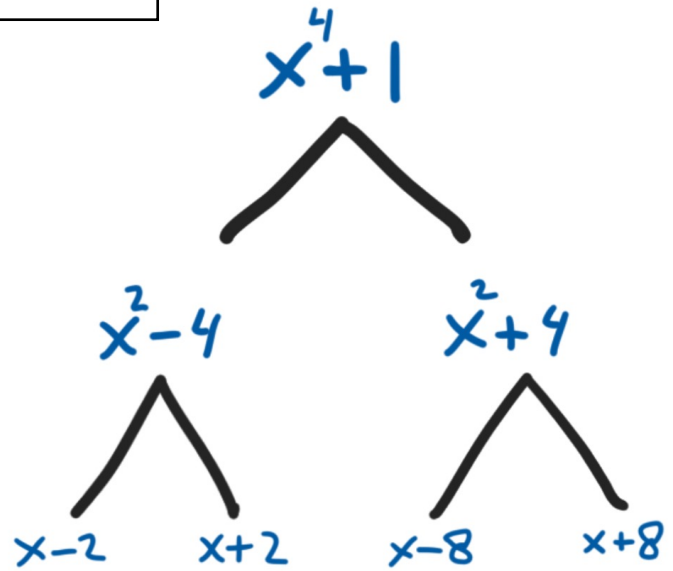
$$R_q = \mathbb{Z}_q[X]/(X^n + 1)$$ * q는 각 계수의 modulus 수

- $(x^n + 1)$의 링 차원을 낮춰 더 작은 서브링으로 표현

Ex) $(x^4 + 1) = (x^2 - 4)(x^2 + 4)$

$(x^2 - 4) = (x - 2)(x + 2)$

$(x^2 + 4) = (x - 8)(x + 8)$



- $(x^4 + 1)$차 상에서의 나머지를 $(x - 2)(x + 2)(x - 8)(x + 8)$차 상에서의 나머지로 표현 가능 → 1차 방정식 상에서의 나머지 이므로 나머지 : 1개의 항

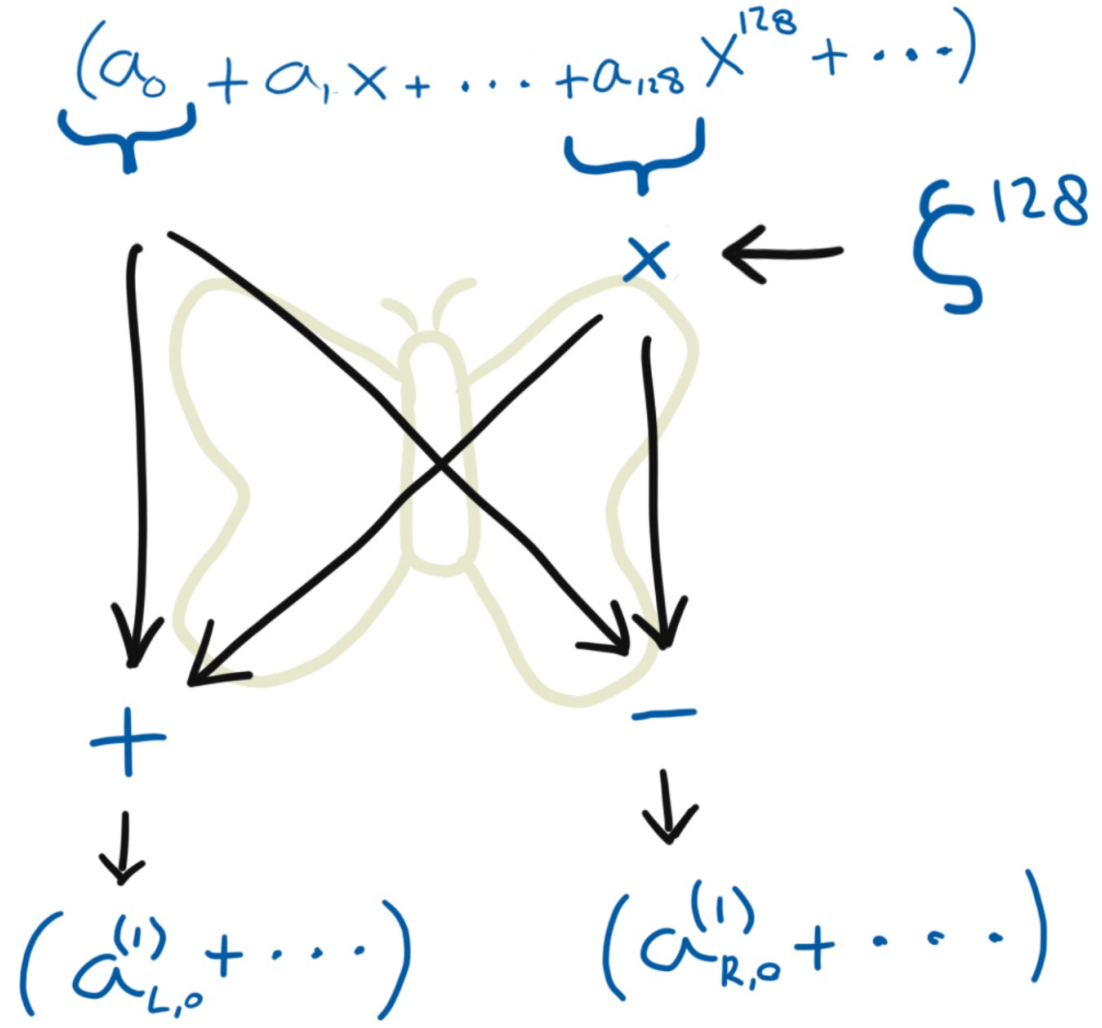
Number theoretic transform(NTT)

- **기본 NTT (Butterfly 연산)**

```
def ntt(f):
    """Compute the NTT of a polynomial.

    Args:
        f: a polynomial

    Format: input as coefficients, output as NTT
    """
    n = len(f)
    if (n > 2):
        f0, f1 = split(f)
        f0_ntt = ntt(f0)
        f1_ntt = ntt(f1)
        f_ntt = merge_ntt([f0_ntt, f1_ntt])
    elif (n == 2):
        f_ntt = [0] * n
        f_ntt[0] = (f[0] + sqr1 * f[1]) % q
        f_ntt[1] = (f[0] - sqr1 * f[1]) % q
    return f_ntt
```



Number theoretic transform(NTT)

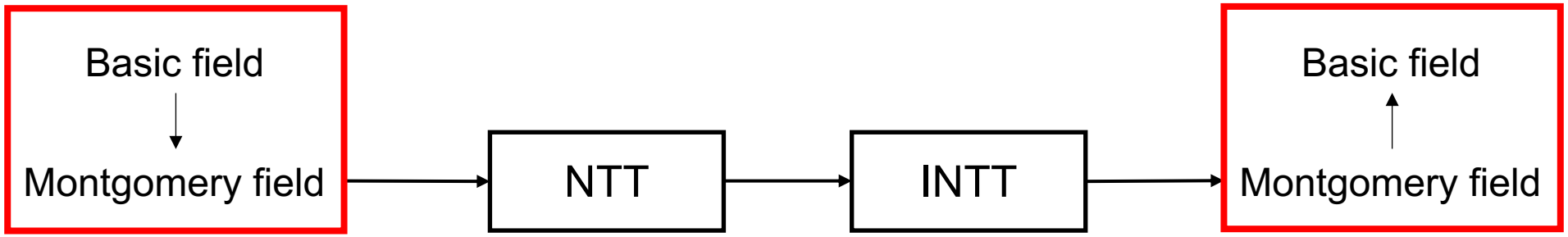
- **Montgomery을 활용한 NTT**

Montgomery 곱셈 : 모듈러 상에서의 곱셈 속도를 높이는 방법

- 나눗셈을 곱셈으로 대체 (나눗셈이 곱셈보다 느린 연산)

$$a \cdot b \equiv c \ (mod\ m), \qquad m < 2^n$$

- Montgomery를 사용해서 NTT에서 modulus 나눗셈을 곱셈으로 변환 하는 방식 활용
- 단점 : NTT 이전 필드를 Montgomery 표현으로 변환하고 INTT 이후 필드를 Montgomery 이전으로 변환해야 함
- 최적화 방법 : 1의 제곱근과 역함수 테이블을 미리 연산하여 사용



Number theoretic transform (NTT)

- **Falcon** : NIST에서 진행한 Post Quantum Conference에서 Finalists로 지정된 격자기반 전자서명 암호
- **Falcon에서 사용되는 NTT ($Z_{q[x]}/x^n + 1(phi)$)**
  - 다항식 $Z_{q[x]}$ (q: Integer modulus)
  - Integer modulus q : 12289
  - Polynomial modulus phi : $(x^n + 1)$, n은 $n \leq 1024$ 인 2의 거듭제곱

- 일반적으로 NTT에 사용하는 phi의 제곱근 값들을 미리 계산 해놓고 사용 (Falcon 뿐만 아니라 확인한 NTT관련 논문들은 모두 사전 연산된 제곱근 값을 사용함)

Number theoretic transform(NTT)

- 양자로 구현할 때 해결해야 하는 문제.. 어려움..
  - 정수필드 상에서의 덧셈, 곱셈
  - 암호에 맞는 prime modulus 연산

- 하고 싶은 것
  1. Butterfly 방식으로 NTT 구현
  2. 기존 Butterfly 방식에 비해 Montgomery 방식이 양자에서도 효율적인지

(처음에는 Montgomery 방식은 속도 측면으로 좋으니까 양자 자원에서 비효율적이라 예상 했는데 공부하다 보니 의외로 양자에서 Modulus 나눗셈을 곱셈으로 대체하는 과정에서 자원 절약이 될 수도 있을 것 같음)

Q & A