Homomorphic Encryption

2024.12.26. 이지은

Basic concept

- **암호화 체계**
  - **키 생성**
  - **암호화**
  - **복호화**
    - **대칭 키 암호화 체계: 암호화와 복호화에 동일한 비밀 키를 사용**
      - 공개 키를 아는 사람은 누구나 데이터를 암호화할 수 있지만 비밀 키를 아는 사람만 데이터를 복호화하고 읽을 수 있음
    - **공개 키 암호화 체계: 암호화에 공개 키를 사용하고 복호화에 비밀 키를 별도로 사용**
      - 매우 많은 양의 데이터를 효율적으로 암호화하는 데 사용할 수 있으며 안전한 아웃소싱 클라우드 스토리지*를 가능하게 함
        - 아웃소싱 클라우드 스토리지* : 사용자 데이터를 CSP에 저장하는 방식
          (사용자가 직접 데이터를 관리하지 않고, 외부의 클라우드 서비스를 통해 저장과 관리를 위임)
      - 오늘날 안전한 온라인 통신을 가능하게 하는 기본 개념
      - 보안성이 높지만 속도가 느림

Basic concept

- **암호화 체계**
  - **키 생성=> 기존 대칭/공개키 암호화 사용 시, 아웃소싱된 계산은 계산 수행 전 암호화 계층이 제거되어야 함**
    - 따라서 아웃소싱된 계산 기능을 제공하는 클라우드 서비스는 비밀 키에 엑세스할 수 있어야 함
      - 이때 암호화를 이용하여 CSP가 데이터를 훔치거나 보는 등의 보안상의 문제를 방지할 수 있음 (데이터 처리를 위해 CSP가 비밀키를 알아야 할 경우 보안 위협 초래)

Preliminaries

- **Mathemetical Induction 수학적 귀납법**
  - **Well-ordering principle (axiom, postulate) 자연수의 대표적인 성질**
    - **공집합이 아닌 부분집합에서 항상 최소 원소가 존재함**
      - S : nonempty subset of {0, 1, 2, … , }
      - $\Longrightarrow$ S has th smallest element in S
      - $\Leftrightarrow$ There exists $a \in S$ such that $a \leq b$ for all $b \in S$
      - $\Leftrightarrow \exists\ a \in S$ s.t. $a \leq b$, $\forall\ b \in S$

Preliminaries

- **Mathemetical Induction 수학적 귀납법**
  - **First principle of induction**
    - Assume that $S \subseteq \mathbb{N}$ satisfies

      (a) $1 \in S$,

      (b) if $k \in S$, then $k+1 \in S$.

      Then $S = \mathbb{N}$.
    - Let $T := \mathbb{N} \setminus S$. Suppose that T is nonempty.

      Since $T \subset \mathbb{N}$, by well-ordering principle, T mush have the smallest element $a \in T$.
    - By (a), $1 \notin T \Longrightarrow a \geq 2 \Longrightarrow a-1 \in S$
    - Since $a \in T$, $a \notin S \Longrightarrow$ by (b), $a-1 \notin S \Longrightarrow a-1 \in T$, which is a contradiction to the assumption : $a \in T$ is the smallest element in T

      $\therefore$ T is empty $\Longrightarrow$ element $S = \mathbb{N}$

Preliminaries

- **Mathemetical Induction 수학적 귀납법**
  - **Example**
    - Show that $1+2+\cdots+n=\frac{n(n+1)}{2}$ for all $n\in\mathbb{N}$.
    - P(n) : $1+2+\cdots+n=\frac{n(n+1)}{2}$
    - Define S := {$n\in\mathbb{N}$: P(n) is true}
    - $1\in S \Leftrightarrow$ P(1) is true
    - If $k\in S$, then $k+1\in S \Leftrightarrow$ if P(k) is true, then P(k+1) is true

      $\because 1+2+\cdots+k+(k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$

Preliminaries

- **The Binomial Theorem 이항 전개**
  - **Definition**
    - Let n∈ℕ and 0≤k≤n. Then we define

      $\binom{n}{k} := \frac{n!}{k!(n-k)!}$
  - **Theorem**
    - $(a+b)^n = \sum_{k=0}^{n}\binom{n}{k}a^{n-k}b^k$Let n∈ℕ and 0≤k≤n. Then we define

      $= a^n + na^{n-1}b + \frac{n(n+1)}{2}a^{n-2}b^2 + \cdots + nab^{n-1} + b^n$

동형암호의 개요

- **동형암호**
  - **암호기술의 분류**
    - 1세대 암호: 패스워드(인증기술)
    - 2세대 암호: 대칭키암호(데이터 암호화) - 블록암호(AES)
    - 3세대 암호: 공개키 암호(키 암호화) – RSA 암호화
    - 4세대 암호: 동형/함수암호(키보호 암호)
      - 암호화된 상태에서 계산이 가능 -> 키가 덜 사용된다는 측면에서 안전성이 높음

동형암호의 개요

- **동형암호 – 미래 컴퓨팅 환경**
  - **'완벽한 하인'**
    - 일을 빠르게 대신 수행, 비밀을 알지 못함
    - ex 1) 개인 클라우드
      - 스토리지 클라우드: Dropbox, Google email 등
      - 계산 클라우드: DNA 계산, 헬스케어 등

        -> 기존의 대칭키 동형암호: 암호화해서 정보를 클라우드에 저장하고, 클라우드는 키를 갖지 않음

        이메일 검색 쿼리 날리면 일반 대칭키에서는 암호화 되어있기에 검색을 못해줌
    - ex 2) 다수 사용자 통계분석 -> 제한적으로 활용
      - 개인정보기반 마케팅(구글, 페이스북, 네이버 등)

        -> 동형암호로 마케팅에 정보를 사용하되, 허용한 경우에만 이용하는 등의 권한 관리 가능

동형암호의 개요

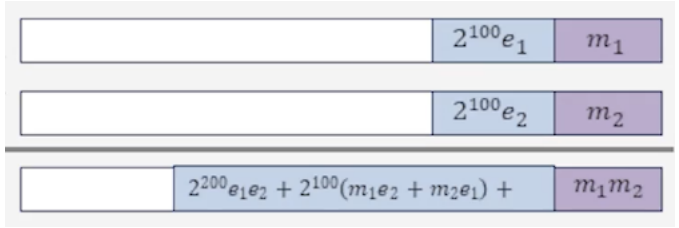
- **동형암호**
  - **Craig Gentry**
    - 암호화된 데이터 연산
      - 사용자가 1+2 하고자 함
      - 각각 33, 54로 암호화되어 클라우드에서 33+54=87 연산 수행
    - 결과 복호화
      - 연산 결과 87을 클라우드에서 다운 후 복호화 시 최종 결과 3 반환
  - **zk-SNARG (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge)**
    - 영지식 증명: 비밀 정보를 누설하지 않고, 특정 명제가 참임을 증명
    - 동형암호: 비밀을 보호하면서 계산
    - 영지식증명 : 비밀을 보호하면서 인증 zk-SNARG
      - 두가지가 양 축임(HE+zk-SNARG)

동형암호의 개요

- **정수 기반 동형암호 (Integer-based HE scheme)**
  - **PAD PH**
    - (**비밀키 p**(큰 소수), **공개키 n**(두 소수의 곱 pq_0))
    - **암호화:** $\boldsymbol{Enc(m) = m + pq\ (mod\ n)}$
    - **복호화:** $\boldsymbol{Enc(m)(mod\ n) = m}$ -> n이 p의 배수임을 활용해 복호화
    - $Enc(m_1)+Enc(m_2) = (m_1+ pq_1) + (m_2+ pq_2) = (m_1+m_2) + (pq_1+ pq_2) = (m_1+m_2) + p(q_1+q_2) = Enc(m_1+m_2)$
  - **DGHV HE scheme (on $\mathbb{Z}_2$) (van Dijk-Gentry-Halevi-Vaikuntanathan)** -> {0,1} 이진수 기반의 수학적 구조
    - $\boldsymbol{Enc(m) = m + 2e + pq}$ -> 랜덤한 노이즈값 e 추가, mod 2 수행 시 항상 m만 남음
    - Secure against quantum computing
    - Use a polynomial ring $R_q = \mathbb{Z}_q[x]/(xn+1)$ ->$\mathbb{Z}_q[x]$는 계수가 $\mathbb{Z}_q$에 속하는 다항식임

동형암호의 개요

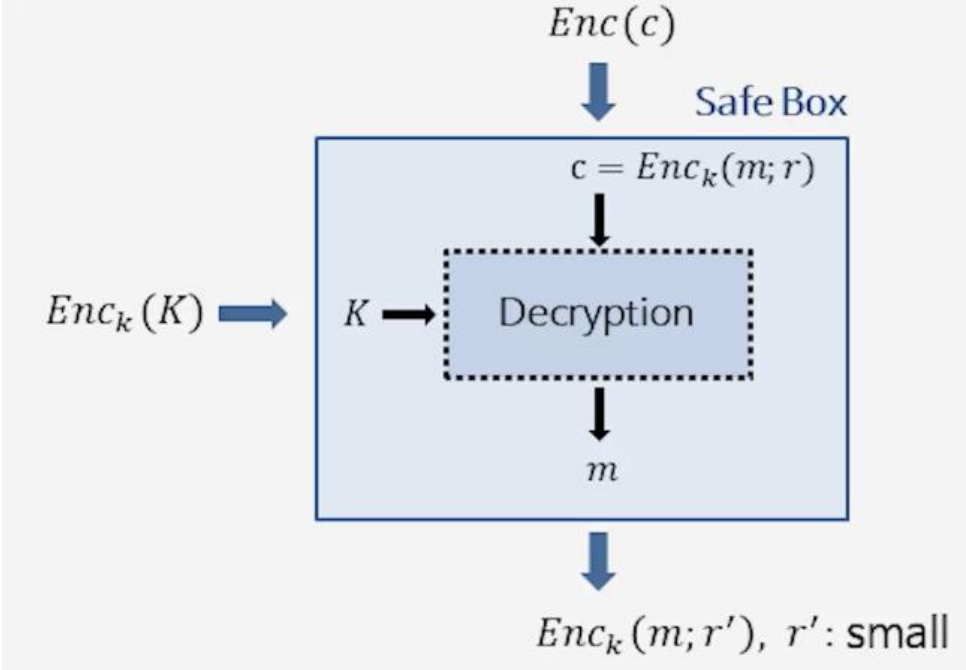
- **정수 기반 동형암호 (Integer-based HE scheme)**
  - **DGHV HE scheme (on $\mathbb{Z}_2$) (van Dijk-Gentry-Halevi-Vaikuntanathan)** -> {0,1} 이진수 기반의 수학적 구조
    - $Enc(m_1) = m_1 + 2^{100}e_1$, $Enc(m_2) = m_2 + 2^{100}e_2$ ->큰 노이즈를 추가해 m과 독립적으로 다룰 수 있게 함
    - ->노이즈가 증폭됨



    - ->0으로 이루어진 흰 공간 존재: 곱셈 횟수만큼 비워놓아야 함 => 동형암호 사이즈가 커지는 이유(암호화시 50배 커짐)
      - 연속적으로 50번 곱할 수 있음(Depth가 $log_2 50$) -> 연산 횟수에 제한이 생김
    - 노이즈를 재부팅해야 함: Bootstrapping (노이즈 리셋하여 암호문을 다시 원래 수준의 노이즈로 줄임)
      - 이를 통해 이론적으로 무한한 동형 연산 지원 가능

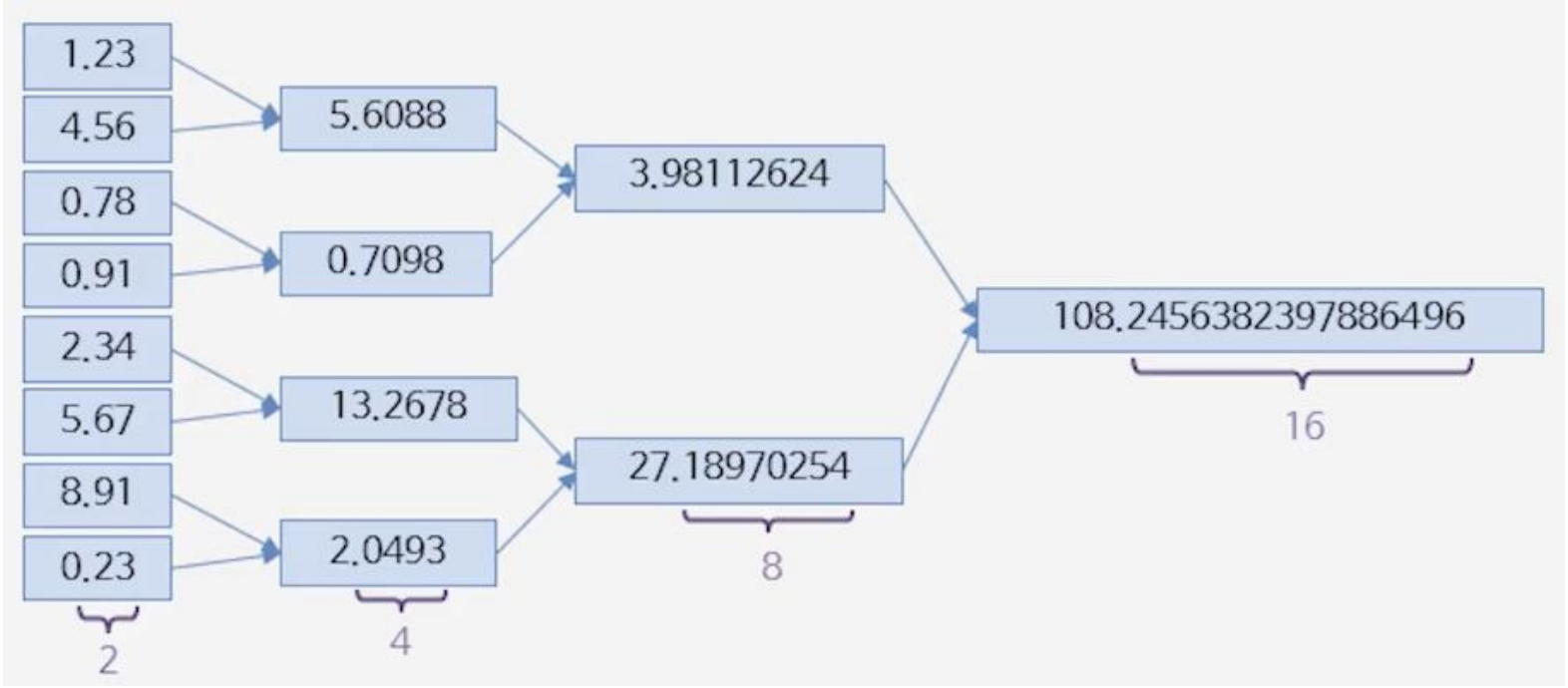
동형암호의 개요

- **정수 기반 동형암호 (Integer-based HE scheme)**
  - **재부팅 (Bootstrapping)**
    - Clearing 과정에서 비밀키가 필요함-> HE 안에서 진행(암호화된 비밀키 Enc(K) 그대로 이용: 암호화된 상태에서 복호화)
    - 다항식 기반 FHE에서는 연산이 반복되면서 다항식의 차수(Degree)가 증가하고, 이로 인해 노이즈도 커짐
    - Deergee가 50이 되면 재부팅 진행 -> public하게 가능



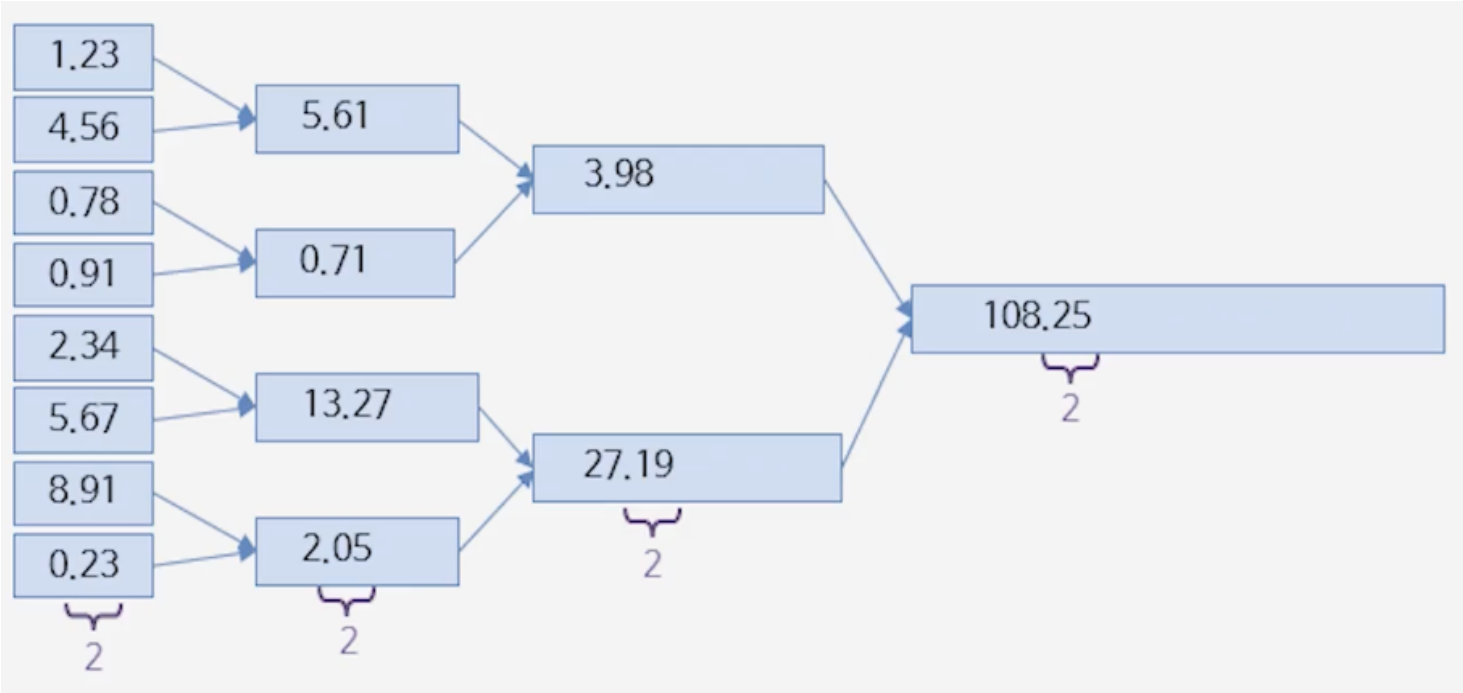
동형암호의 개요

- **근사동형암호**
  - 평문 크기가 연산마다 2배 증가 시 20번 후에는 백 만 비트 ; 쌀 한 톨의 비유
    - 곱하기 10번 하면 2^10 1000비트 공간 필요 => 평문 크기도 커짐
    - => 반올림(Rounding) 필요 : + * 만으로도 가능



동형암호의 개요

- **근사동형암호**
  - HEAAN [CKKS17]
    - 근사계산 지원(암호화된 정수값의 Re-scaling) -> 암호문 크기 관리
    - 항상 같은 크기의 자릿수 유지 가능, 대부분 응용분야에서는 실수계산을 필요로 함 -> 효율적인 연산 지원



동형암호의 개요

- **동형암호의 현재**
  - HEAAN [CKKS17]
    - 암호화된 복소수를 2^(-30) 이하의 Error로 Depth2의 연산이 가능한 파라미터 기준
    - -> 곱셈 연산이 반복되면 오차가 기하급수적으로 커질 수 있으므로, Depth로 연산의 깊이를 제한하여 오차를 관리
    - -> slot: 암호문이 동시에 처리할 수 있는 데이터 항목 수
    - => 평문이 복소수

	공개키 크기	암호문 크기	암호화	복호화	덧셈	곱셈	평문	Slot 개수
RSA-2048	2048bit	2048bit	6ms	200ms	-	-	-	-
ECC-193	193bit	80B	8ms	15ms	-	-	-	-
HEAAN	444kB	111kB	11ms	5ms	0.2ms	25ms	복소수	2048

동형암호의 개요

- **Fully Homomorphic Encryption**
  - **정수 동형암호: 정수론 (AGCD 기반)**
    - [DGHV10] FHE over the Integers. Eurocrypt 2010
    - CMNT11, CNT12, CCKLLTY13, CLT14, etc
  - **행렬 동형암호: 격자(Lattice)이론 (LWE 기반)** -> Learning With Errors
    - [BV11a] Efficient FHE from (Standard) LWE. FOCS11
    - Bra12, BGV12, GSW13
  - **다항식 동형암호: Ideal Lattice (RingLWE 기반)**
    - Ideal lattice: SV10, NTRU: LTV12
    - Ring-LWE: BV11b, GHS13, BLLN13, HEAAN, etc

동형암호의 개요

- **동형암호의 발전 과정**
  - **2009~2010: Plausibility**
    - [GH11]: 동형암호를 사용한 단일 비트 연산의 가능성을 입증, 단일 비트 연산에 30분 소요
  - **2011~2012: Scalable for Large Circuits**
    - [GHS12b]: AES-128 알고리즘의 120 블록을 36시간 내에 처리 / Bra12, BGV12, GSW13
  - **2013~2015: Depth-Linear Construction**
    - [HS14]: IBM의 오픈소스 라이브러리 Helib
    - Brakerski-Gentry-Vaikuntanathan(BGV) 체계 구현 / 동일한 30,000 게이트 회로를 4분 만에 처리
  - **2015~Today: Usability**
    - HEAAN, TFHF
    - 실제 응용: 빅데이터 분석, 기계학습 등
    - 동형암호 표준화 회의 / Private Genome Computation 대회 (iDash)

동형암호의 개요

- **Best Performing HE Schemes**

Type	Classical HE	Fast Bootstrapping	Approximate Encryption
Scheme	[BGV12] BGV [Bra12, FV12] B/FV	[DM15] FHEW [CGGI16] TFHE	[CKKS17] HEAAN
Plaintext	Finite Field Packing	Binary string	Real/Complex numbers Packing
Operation	Addition, Multiplication	Look-up table & bootstrapping	Fixed-point Arithmetic
Library	HElib (IBM) SEAL (Microsoft Research) Palisade (Duality inc.)	TFHE (inpher, gemalto, etc.)	**HEAAN (SNU)**

SEAL > CKKS

- **native/examples/5_ckks_basics.cpp**
  - Microsoft SEAL 라이브러리를 사용하여 CKKS(Chexly-Kim-Kim-Song) 암호화 방식을 통해 암호화된 데이터 상에서 다항식 계산을 수행하는 예제
    - CKKS는 실수나 복소수 데이터를 암호화하여 동형연산을 가능하게 하며, 주로 머신러닝이나 데이터 분석에서 사용됨
  - CKKS 암호화 스키마를 활용하여 암호화된 데이터 상에서 **다항식 $f(x) = \pi x^3 + 0.4x + 1$을 계산하는 방법을 시연**하는 코드

SEAL > CKKS

- **native/examples/5_ckks_basics.cpp**
  - **코드 흐름**
    - 1) CKKS 스키마 초기화: 암호화 매개변수(EncryptionParameters) 설정(다항식 차수, mod 크기 등 포함)
    - 2) 키 생성: 암호화를 위한 pk, sk 생성
      - 곱셈 후 암호문 크기를 줄이는 재선형화 키(RelinKeys)
      - 슬롯 데이터의 위치를 바꾸는 회전 키(GaloisKeys).
    - 3) 입력 데이터 준비 및 암호화: [0, 1] 구간의 4096개 점을 생성하여 암호화 함
    - 4) 다항식 계산: 연산 중 발생하는 스케일 문제를 해결하기 위해 재선형화와 모듈러스 체인 이동 수행
    - 5) 암호문 합산 및 복호화: 연산 중 발생하는 스케일 문제를 해결하기 위해 **재선형화**와 **모듈러스 체인 이동**을 수행.

SEAL > CKKS

- **native/examples/5_ckks_basics.cpp**
  - **관련 개념**

    1. 스케일 관리
       - CKKS는 소수점을 처리하기 위해 데이터를 스케일링(확대)하여 정수로 변환된다. 연산이 진행될수록 스케일이 증가하므로, Rescale을 사용해 스케일을 조정한다.

    2. 모듈러스 체인 이동
       - CKKS는 계수 모듈러스(coeff_modulus)라는 수학적 공간을 사용한다. 연산 중 모듈러 체인을 이동시켜 암호문의 크기를 줄이고, 연산 호환성을 유지할 수 있다.

    3. 재선형화
       - 암호문 곱셈 연산은 결과의 크기를 키우므로, 재선형화를 통해 암호문의 크기를 줄여 연산 효율성을 높인다.

    4. 암호문 간 호환성
       - 암호문끼리 더하거나 곱하려면 동일한 스케일과 모듈러스 체인을 가져야 한다. 이를 위해 스케일을 정규화하고 모듈러 체인을 맞춘다.

End

End