(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년10월10일
(11) 등록번호 10-1906363
(24) 등록일자 2018년10월02일

(51) 국제특허분류(Int. Cl.)
H04L 9/06 (2006.01)
(52) CPC특허분류
H04L 9/06 (2013.01)
H04L 2209/122 (2013.01)
(21) 출원번호 10-2016-0155667
(22) 출원일자 2016년11월22일
심사청구일자 2016년11월22일
(65) 공개번호 10-2018-0057255
(43) 공개일자 2018년05월30일
(56) 선행기술조사문헌
Cheon, Jung Hee, et al. "The polynomial approximate common divisor problem and its application to the fully homomorphic encryption." Information Sciences 326 (2016): 41-58.
Fouque, Pierre-Alain, et al. "Cryptanalysis of the Co-ACD Assumption." Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2015.
Cheon, Jung Hee, Hyung Tae Lee, and Jae Hong Seo. "A New Additive Homomorphic Encryption based on the co-ACD Problem." Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Secu
KR1020160017226 A

(73) 특허권자
명지대학교 산학협력단
경기도 용인시 처인구 명지로 116 (남동, 명지대학교)
(72) 발명자
서재홍
서울특별시 서초구 잠원로 150, 302동 706호 (잠원동, 잠원한신아파트)
김창진
경기도 용인시 처인구 명지로116번길 40, B04호 (남동, 대흥주택)
(74) 대리인
이은철, 이우영

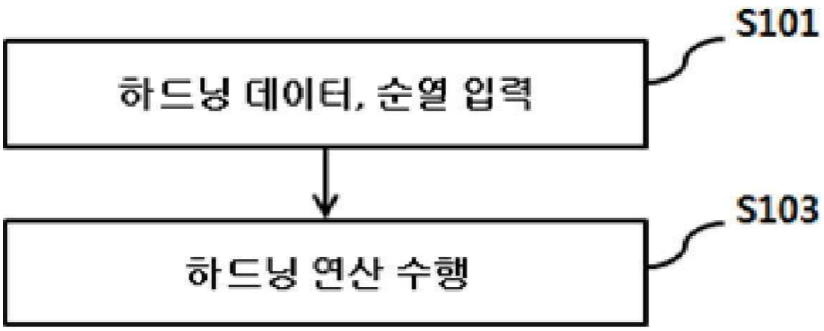
전체 청구항 수 : 총 2 항 심사관 : 양종필

(54) 발명의 명칭 **ACD 인스턴스 하드닝 방법 및 하드닝 복구 방법과 이를 이용하는 장치.**

(57) 요 약

본 기술은 ACD 인스턴스 하드닝 방법 및 하드닝 복구 방법과 이를 이용하는 장치에 관한 것이다. 본 기술의 구현예에 따르면, ACD 인스턴스의 링 구조를 무너뜨려 보안성을 유지하면서 ACD 인스턴스의 길이를 줄일 수 있으며 암호문 또한 짧게 할 수 있어 암호시스템의 효율성을 높일 수 있는 이점이 있다.

대 표 도 - 도1



명 세 서

청구범위

청구항 1

하드닝하고자 하는 데이터와 랜덤하게 선택된 순열을 입력 받는 단계; 및

입력 받은 상기 하드닝하고자 하는 데이터에 대하여 상기 순열에 따라 하드닝 연산을 수행하여 ACD 인스턴스의 링 구조를 은닉하도록 하는 단계를 포함하는 것을 특징으로 하는 ACD 인스턴스 하드닝 방법.

청구항 2

ACD 인스턴스의 링 구조를 은닉하도록 하드닝된 데이터와 상기 데이터의 하드닝 연산에 이용된 순열의 역순열을 입력 받는 단계; 및

상기 링 구조를 은닉하도록 하드닝된 데이터에 대하여 입력 받은 상기 역순열에 따라 하드닝 복구 연산을 수행하는 단계를 포함하는 것을 특징으로 하는 ACD 인스턴스 하드닝 복구 방법.

발명의 설명

기 술 분 야

[0001] ACD 인스턴스 하드닝 방법 및 하드닝 복구 방법과 이를 이용하는 장치에 관한 것으로서, 더욱 상세하게는 ACD 문제를 더욱 어렵게 만들어 보다 짧은 암호문을 갖는 대칭키 기반 동형암호를 설계하도록 하는 ACD 인스턴스 하드닝 방법 및 하드닝 복구 방법과 이를 이용하는 장치에 관한 것이다.

배 경 기 술

[0002] ACD 문제는 Howgrave-Graham 에 의해 도입된 방법으로, 작은 노이즈 r_i에 대해 $x_i = pq_i + r_i$인 p의 근접 곱들(Approximate multiples) 사이의 공통 정수(common integer) p를 찾기 위해 2001년경 도입되었다.

[0003] ACD 문제의 근접 곱들은 $\{x_i = pq_i + r_i\}$인 샘플들을 말하고, 위와 같은 샘플의 합은 작은 노이즈 $(r_j + r'_j)$에 대해 $x_j + x'_j = p(q_j + q'_j) + (r_j + r'_j)$ 과 같이 샘플과 같은 형태를 보존하는 유용한 특성이 있다.

[0004] 비슷하게, 두 샘플들 x_1과 x_2의 곱은 또한 충분히 작은 $r_1 r_2$ 라면 $p(q_1 x_2 + r_1 q_2) + r_1 r_2$ 과 같이 샘플과 같은 형태를 가진다. 이러한 동형성은 정수에 대한 FHE(Fully Homomorphic Encryptions)의 설계에 매우 유용한 성질이다.

[0005] 처음 ACD 문제 기반으로 설계된 Van Dijk, Gentry, Halevi, Vaikuntanathan이 제안한 동형암호 이후로 동형암호의 효율성을 향상시키기 위한 많은 연구가 있었다. 예를 들어, 공개키의 크기를 짧게 만들거나, 배치 버전(batch version)으로 확장하는 등의 방법으로 효율의 개선을 시도하는 연구들이 있었다.

[0006] 그러나, 기존 ACD 문제 기반의 암호는 ACD 인스턴스가 갖는 링 구조 때문에 유용한 정수론적 도구들을 적용할 수 있었고, 이를 이용한 몇몇 알려진 공격 및 직교격자공격(Orthogonal Lattice Attack)등에 의해 암호문의 크기가 $\omega((\eta-\rho)^2 \log\lambda)$ 보다 작아질 수 없는 문제가 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 1. 한국 공개특허공보 10-2016-0017226

발명의 내용

해결하려는 과제

[0008] 본 발명은 상기와 같은 문제를 해결하기 위한 것으로서, ACD 문제의 하드닝을 통해 크기가 줄어든 암호문을 이용하여 암호 연산의 효율을 높일 수 있는 ACD 인스턴스 하드닝 방법 및 하드닝 복구 방법과 이를 이용하는 장치를 제공하는 것이다.

과제의 해결 수단

[0009] 상기와 같은 목적을 달성하기 위한 본 발명의 일 실시예는 하드닝하고자 하는 데이터와 랜덤하게 선택된 순열을 입력 받는 단계; 및 입력 받은 상기 하드닝하고자 하는 데이터에 대하여 상기 순열에 따라 하드닝 연산을 수행하여 ACD 인스턴스의 링 구조를 무너뜨리는 단계를 포함하는 것을 특징으로 한다.

[0010] 본 발명의 다른 실시예는 ACD 인스턴스의 링 구조를 은닉하도록 하드닝된 데이터와 상기 데이터의 하드닝 연산에 이용된 순열의 역순열을 입력 받는 단계; 및 상기 하드닝된 데이터에 대하여 입력 받은 상기 역순열에 따라 하드닝 복구 연산을 수행하는 단계를 포함하는 것을 특징으로 한다.

발명의 효과

[0011] 상기된 바에 따르면, ACD 문제를 어렵게 하는 하드닝 연산을 통해 보다 짧은 길이의 암호문으로도 높은 보안성을 제공할 수 있으며, 이에 따라 암호화, 복호화 연산 속도를 개선할 수 있는 이점이 있다.

도면의 간단한 설명

[0012] 도 1은 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 방법의 흐름도이다.

도 2는 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 복구 방법의 흐름도이다.

도 3은 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 장치의 블록도이다.

도 4는 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 복구 장치의 블록도이다.

도 5는 본 발명의 일 실시예에 따른 하드닝 ACD 인스턴스 기반 동형 암호 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0013] 이상의 본 발명의 목적들, 다른 목적들, 특징들 및 이점들은 첨부된 도면과 관련된 이하의 바람직한 실시예들을 통해서 쉽게 이해될 것이다. 그러나 본 발명은 여기서 설명되는 실시예들에 한정되지 않고 다른 형태로 구체화될 수도 있다. 오히려, 여기서 소개되는 실시예들은 개시된 내용이 철저하고 완전해질 수 있도록 그리고 당업자에게 본 발명의 사상이 충분히 전달될 수 있도록 하기 위해 제공되는 것이다. 본 명세서에서, 어떤 구성요소가 다른 구성요소 상에 있다고 언급되는 경우에 그것은 다른 구성요소 상에 직접 형성될 수 있거나 또는 그들 사이에 제 3의 구성요소가 게재될 수도 있다는 것을 의미한다.

[0014] 어떤 엘리먼트, 구성요소, 장치, 또는 시스템이 프로그램 또는 소프트웨어로 이루어진 구성요소를 포함한다고 언급되는 경우, 명시적인 언급이 없더라도, 그 엘리먼트, 구성요소, 장치, 또는 시스템은 그 프로그램 또는 소프트웨어가 실행 또는 동작하는데 필요한 하드웨어(예를 들면, 메모리, CPU 등)나 다른 프로그램 또는 소프트웨어(예를 들면 운영체제나 하드웨어를 구동하는데 필요한 드라이버 등)를 포함하는 것으로 이해되어야 할 것이다.

[0015] 또한 어떤 엘리먼트(또는 구성요소)가 구현됨에 있어서 특별한 언급이 없다면, 그 엘리먼트(또는 구성요소)는 소프트웨어, 하드웨어, 또는 소프트웨어 및 하드웨어 어떤 형태로도 구현될 수 있는 것으로 이해되어야 할 것이다.

[0016] 본 명세서에서 사용된 용어는 실시예들을 설명하기 위한 것이며 본 발명을 제한하고자 하는 것은 아니다. 본 명세서에서, 단수형은 문구에서 특별히 언급하지 않는 한 복수형도 포함한다. 명세서에서 사용되는 '포

함한다(comprises)' 및/또는 '포함하는(comprising)'은 언급된 구성요소는 하나 이상의 다른 구성요소의 존재 또는 추가를 배제하지 않는다.

[0017] 이하, 도면을 참조하여 본 발명을 상세히 설명하도록 한다. 아래의 특정 실시예들을 기술하는데 있어서, 여러 가지의 특정적인 내용들은 발명을 더 구체적으로 설명하고 이해를 돕기 위해 작성되었다. 하지만 본 발명을 이해할 수 있을 정도로 이 분야의 지식을 갖고 있는 독자는 이러한 여러 가지의 특정적인 내용들이 없어도 사용될 수 있다는 것을 인지할 수 있다. 어떤 경우에는, 발명을 기술하는 데 있어서 흔히 알려졌으면서 발명과 크게 관련 없는 부분들은 본 발명을 설명하는 데 있어 별 이유 없이 혼돈이 오는 것을 막기 위해 기술하지 않음을 미리 언급해 둔다.

[0018] 본 발명은 ACD 문제에 대하여 링 구조를 이용하는 공격들인 SDA(Simultaneous Diophantine Approximation), Coppersmith`s Technique, Chen-Nguyen Exhaustive Search for Noise 등의 알려진 공격 및 각종 직교격자공격에 안전하기 위한 ACD 인스턴스의 크기의 한계를 극복하고자 한 것이다. 본 발명의 하드닝 기술을 ACD 문제 기반 암호 시스템에 적용하면 ACD 인스턴스의 링 구조를 무너뜨림으로써 유용한 정수론적 도구들의 적용에 제한을 준다. 또한, 직교격자공격을 적용하는데 걸리는 시간은 주어진 격자의 차원(dimension)에 큰 영향을 받는데 본 발명의 하드닝 기술을 적용함으써 공격자는 높은 차원의 격자를 선택해야만 하는 제약을 받게 된다. 이로써, 보다 짧은 인스턴스의 크기로 같은 안전성을 얻을 수 있으며 짧은 암호문의 크기를 갖는 동형암호를 얻을 수 있다.

[0019] 본 발명에 따른 암호 시스템에서 γ 비트 정수를 비트단위로 쪼개서 섞는 과정을 하드닝 방법(Hardening Process), 이것의 역과정을 하드닝 복구 방법(Hardening Recovery Process)이라고 할 수 있다.

[0020] 기존의 ACD 문제 인스턴스에 대해 위의 하드닝 방법을 수행한 결과물을 이용하여 암호를 설계할 수 있다.

[0021] 본 발명의 이러한 기술에 따르면 ACD 문제를 보다 어려운 문제로 만들 수 있다. 기존 암호시스템에서는 기반문제보다 더 쉬운 문제를 이용하여 효율성을 향상시켰던 반면, 본 발명은 더 어려운 문제를 이용하여 효율성을 향상시킨다는 점이 기존의 방법에 비해 특이한 점이다.

[0022] 본 발명에 따른 ACD 인스턴스 하드닝 방법은 다음과 같은 알고리즘에 따라 구현될 수 있다.

[0023] 우선, Setup 알고리즘은 보안 파라미터 λ에 따라 순열 집합 S_γ로 부터 랜덤하게 선택된 순열 π_γ과 그 역순열 π_γ^{-1}을 생성한다. 보안 파라미터는 암호문제에 있어서 계산 문제의 입력 크기를 말한다.

[0024] 그리고, 하드닝 연산 알고리즘인 HL는 γ-비트 정수 x와 순열 π_γ을 입력 받아 하드닝 연산 결과인 $\sigma_{\pi_\gamma} \circ BD_\gamma(x)$를 ACD 인스턴스로서 생성한다. 또한, 이와 같이 하드닝 처리된 ACD 인스턴스를 동형 암호 설계에 이용할 수 있다.

[0025] BL는 비트 분해 함수로서 $BD_n(x) = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$과 같이 정의될 수 있으며 여기서, $x = \sum_{i=0}^{n-1} x_i 2^i$ 일 수 있다.

[0026] 또한 순열 $\pi_\gamma \in S_\gamma$ 이라고 할 때, 아래와 같이 정의되는 위치-치환(position-permuting) 함수 $\sigma_{\pi_\gamma}: \mathbb{Z}^\gamma \to \mathbb{Z}^\gamma$ 은 다음 수학식 1과같이 정의될 수 있다. 여기서 $\mathbb{Z}^\gamma$은 벡터 공간을 의미한다.

[0028] <수학식 1>

[0029] $$(a_0, \ldots, a_{\gamma-1}) \mapsto (a_{\pi_\gamma(0)}, \ldots, a_{\pi_\gamma(\gamma-1)})$$

[0031] 하드닝 복구 알고리즘인 RC는 벡터 $a \in \mathbb{Z}^\gamma$ 과 역순열 π_γ^{-1}을 입력 받아 생성된 결과인

$CP_\gamma \circ \sigma_{\pi_\gamma}^{-1}(a)$ 을 반환한다. 벡터 $a \in \mathbb{Z}^\gamma$ 는 ACD 인스턴스의 링 구조를 은닉하도록 하드닝된 것이다.

[0032] 여기서, CP는 합성 함수로서 <,>를 유클리드 공간(Euclidean space)의 표준 내적 곱(standard inner product)라고 할 때 $\mathrm{CP}_n(y) = \langle y, (1,2,\ldots,2^{n-1}) \rangle \in \mathbb{Z}$ 이고, $y \in \mathbb{Z}^n$ 로 정의되는 것일 수 있다.

[0033] 또한 $\mathrm{CP}_n \circ \mathrm{BD}_n(x) = x$ 이고, 여기서 $\circ$ 은 함수의 합성을 뜻한다. 본 명세서에서는 n이 문맥상 자명한 경우에는 생략되었을 수 있다.

[0034] 이러한 하드닝 알고리즘과 하드닝 복구 알고리즘을 식으로 나타내면 다음 수학식 2와 같이 나타낼 수 있다.

[0036] <수학식 2>

[0037]
$$\mathrm{RC}\left(\sum_i HD(x_j, \pi_\gamma), \pi_\gamma^{-1}\right) = \sum_i RC\left(HD(x_j, \pi_\gamma), \pi_\gamma^{-1}\right) = \sum_i x_j$$

[0039] 위와 같이 정의될 수 있는 하드닝 연산을 거친 ACD 인스턴스는 덧셈구조는 여전히 유지하고 있지만 곱셈구조는 무너지게 된다.

[0040] 이로써 하드닝 연산을 적용한 ACD 인스턴스는 링 구조가 무너지게 된다.

[0041] 따라서 링 구조에서 적용할 수 있었던 유용한 정수론적 도구들을 하드닝 처리를 수행한 인스턴스에는 적용할 수 없게 되며 기존의 알려진 공격은 적용하기 어렵게 된다.

[0042] 그럼에도, 공격자는 여전히 직교격자공격을 적용할 수 있지만, 선택한 격자의 생성자들의 독립성을 보장하기 위해 적어도 γ개의 인스턴스를 선택해야 한다.

[0043] 이는 공격을 적용하는데 걸리는 시간이 선택한 격자의 차원에 크게 의존적인 직교격자공격을 적용하는데 큰 제약을 준다.

[0044] 도 1은 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 방법의 흐름도를 도시한다. 이하 순열 π_γ은 랜덤하게 선택된 순열이다.

[0045] 도 1에 따르면, 본 발명에 따른 ACD 인스턴스 하드닝 방법은 하드닝하고자 하는 데이터와 순열 π_γ을 입력 받는 단계(S101)와 입력 받은 상기 하드닝하고자 하는 데이터에 대하여 순열 π_γ에 따라 하드닝 연산하여 ACD 인스턴스의 링 구조를 무너뜨리는 단계(S103)를 포함한다.

[0046] 하드닝하고자 하는 데이터와 순열 π_γ을 입력 받는 단계(S101)에서, 하드닝하고자 하는 데이터는 정보처리 장치에 의하여 처리될 수 있는 γ-비트 정수 x 일 수 있고, 순열 π_γ는 상술한 Setup 알고리즘을 보안 파라미터 λ에 따라 수행하여 순열 집합 S_γ로 부터 랜덤하게 선택된 순열일 수 있다.

[0047] ACD 인스턴스의 링 구조를 은닉하도록 하는 단계(S103)는 상술한 HD 알고리즘에 따라 수행될 수 있다.

[0048] 도 2는 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 복구 방법의 흐름도를 도시한다.

[0049] 도 2에 따르면, 본 발명에 따른 ACD 인스턴스 하드닝 복구 방법은 ACD 인스턴스의 링 구조를 은닉하도록 하드닝된 데이터와 상기 데이터의 하드닝에 이용된 순열 π_γ의 역순열 π_γ^{-1}을 입력 받는 단계(S201)와 상기 하드닝된 데이터에 대하여 입력 받은 상기 역순열 π_γ^{-1}에 따라 하드닝 복구 연산을 수행하는 단계(S203)를 포함한다.

[0050] 하드닝된 데이터와 순열 π_γ의 역순열 π_γ^{-1}을 입력 받는 단계(S201)에서, 하드닝된 데이터는 상술한 하

드닝 연산을 거쳐 생성된 벡터 $a \in \mathbb{Z}^{\gamma}$ 일 수 있고, 순열 π_{γ}의 역순열 π_{γ}^{-1}은 Setup 알고리즘을 보안 파라미터 λ에 따라 수행하여 순열 집합 S_{γ}로 부터 랜덤하게 선택된 순열일 수 있다.

[0051] 하드닝 복구 연산을 수행하는 단계(S203)는 상술한 RC 알고리즘에 따라 수행될 수 있다.

[0052] 도 1 및 도 2를 참조하여 설명하고 있는 하드닝 및 하드닝 복구를 위한 동작들은 컴퓨터 장치에서 구현될 수 있으며 예를 들면 1개의 컴퓨터 장치 내에서 구현되거나 또는 2개 이상의 컴퓨터 장치로 분산되어 구현될 수 있고, 각 단계는 하드웨어(예를 들면, 칩) 및/또는 컴퓨터 프로그램 코드에 의해 구현될 수 있다.

[0053] 도 3은 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 방법을 이용하는 장치를 설명하기 위한 도면이다.

[0054] 도 3을 참조하면, 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 장치(100)는 하드닝 데이터 입력부(110), 하드닝 연산부(120)를 포함한다.

[0055] 하드닝 데이터 입력부(110)는 하드닝하고자 하는 데이터와 순열 π_{γ}을 입력 받는다.

[0056] 하드닝 연산부(120)는 입력 받은 상기 하드닝하고자 하는 데이터에 대하여 순열 π_{γ}에 따라 하드닝 연산을 수행하여 ACD 인스턴스의 링 구조를 은닉하도록 한다.

[0057] 하드닝 연산은 상술한 HD 알고리즘에 따라 수행될 수 있다.

[0058] 도 4는 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 복구 방법을 이용하는 장치를 설명하기 위한 도면이다.

[0059] 도 4를 참조하면, 본 발명의 일 실시예에 따른 ACD 인스턴스 하드닝 장치(200)는 하드닝 복구 데이터 입력부(210), 하드닝 복구 연산부(220)를 포함한다.

[0060] 하드닝 복구 데이터 입력부(210)는 ACD 인스턴스의 링 구조를 은닉하도록 하드닝된 데이터와 순열 π_{γ}의 역순열 π_{γ}^{-1}을 입력 받는다.

[0061] 하드닝 복구 연산부(220)는 상기 하드닝된 데이터에 대하여 입력 받은 상기 역순열 π_{γ}^{-1}에 따라 하드닝 복구 연산을 수행한다.

[0062] 하드닝 복구 연산은 상술한 RC 알고리즘에 따라 수행될 수 있다.

[0063] 본 발명에 따른 하드닝 방법은 암호시스템에 구체적으로 적용될 수 있다.

[0064] 우선, 이하에서는 본 발명에 따른 암호 시스템을 구현하기 위한 CS(Cheon-Stehle) 도식(scheme)을 설명한다.

[0065] 또한 이하에서, sk(secret key)는 비밀키(또는 대칭키라고도 함)이고, ek(evaluation key)는 연산키를 나타낸다.

[0066] CS 도식은 공지된 바와 같이 Setup, Enc, Add, Mult, ConOp, Dec와 같은 알고리즘을 포함하여 구성될 수 있다.

[0067] 여기서, Setup 알고리즘은 ACD 문제의 원본 인스턴스를 생성한다. Enc 알고리즘은 메세지 m를 암호화한 암호문을 생성한다. Dec 알고리즘은 암호문 c와 sk를 입력 받아 모듈러 연산에 따라 복호화 한다. Add, Mult, ConOp 알고리즘은 각각 CS 도식에 있어서 필요한 합 연산, 곱 연산, ConOp 연산을 정의한다.

[0069] 암호 시스템의 파라미터(parameter)들은 다음과 같이 정의될 수 있다.

[0070] λ : 보안 파라미터(security parameter)

[0071] ρ : 에러의 비트 길이(bit length of error)

[0072] η : 홀수인 비밀 정수의 비트 길이(bit length of secret odd integer)

[0073] γ : ACD 인스턴스의 비트 길이(bit length of ACD instances)

[0075] 또한, 함수 P_n은 아래 수학식 3과 같이 정의될 수 있다.

[0076] <수학식 3>

[0077] $$P_n : \mathbb{R} \longrightarrow \mathbb{R}^n \text{ as } y \mapsto (y, 2y, \ldots, 2^{n-1}y) \in \mathbb{R}^n$$

[0079] 그리고, P_n과 BD_n 사이의 관계는 $\langle BD_n(x), P_n(y)\rangle = xy$ 와 같이 나타낼 수 있다. 본 명세서에서 l비트 메세지에 대하여 변형된 CS 도식은 다음과 같다.

[0080] Setup(1^λ)는 $\mathbb{Z}_g$를 $\lfloor \log g \rfloor + 1 = l$ 이라고 할 때의 메세지 공간이라고 할 때, (ρ, η, γ)-ACD 문제의 원본 인스턴스를 생성하며, 랜덤의 η-비트 홀수(odd) 정수 p와 아래 수학식 4와 같은 샘플들을 생성한다.

[0082] <수학식 4>

[0083] $$y_0, \ldots, y_{l-1} \xleftarrow{s} D_{\gamma,\rho}(p) \text{ and for } i = 0, \ldots, l-1, \text{set } y_i' \leftarrow y_i + [\frac{p}{g}]2^i$$

[0085] 아래 수학식 5에 따라 산출되는 $z_{i,j}$에 따라 세팅되는 벡터 z를 아래 수학식 6에 따라 산출한다.

[0087] <수학식 5>

[0088] $$z_{i,j} \xleftarrow{s} D_{\gamma,\rho}(p)$$

[0090] <수학식 6>

[0091] $$z = (z_{i,j}')0 \le i,j < \gamma = (z_{i,j})0 \le i,j < \gamma + [\frac{p}{g}([P_\gamma(\frac{g}{p})]_g \otimes [P_\gamma(\frac{g}{p})]_g)]$$

[0093] 최종적으로, 연산키인 $ek = (y_0', \ldots, y_{l-1}', z)$에 따라 산출되고, 비밀키인 sk = p는 비밀로 유지된다.

[0094] Enc(m, sk)는 $x_0, \ldots, x_{l-1} \xleftarrow{s} D_{\gamma,\rho}(p)$ 를 선택하고, $x' \leftarrow \sum_{i=0}^{\ell-1} x_i + \lfloor \frac{p}{g} \rfloor m$.[8] 연산을 수행한다.

[0096] 암호문 c_1, c_2에 대하여,

[0097] Add(c_1, c_2)는 $c_1 + c_2$연산을 수행한다.

[0098] Mult(c_1, c_2,ek)는 $\langle BD_\gamma(c_1) \otimes BD_\gamma(c_2), z\rangle$ 연산을 수행한다.

[0099] ConOp(m, ek)는 $\sum_{i=0}^{\ell-1} m_i y_i'$ 연산을 수행한다.

[0100] Dec(c,sk)는 $\lfloor c \cdot \frac{g}{p} \rceil \pmod g$ 연산을 수행한다.

[0102] 본 발명의 하드닝 방법은 FHE 시스템에 매우 조화롭게 적용될 수 있으며, 짧은 암호문의 이점을 제공할 수 있다.

[0103] 상술한 CS 도식이 FHE 시스템에 구현된 CS FHE 도식을 CS = (CS.Setup, CS.Enc, CS.Add, CS.Mult, CS.ConOp, Cs.Dec)라고 나타낼 수 있다. 아래에서 HCS.Setup과 같이 앞의 H는 CS 도식이 구현된 FHE 시스템에

있어 본 발명의 하드닝 방법 및 장치가 적용되었음을 나타낸다.

[0104] 본 발명에 따른 하드닝 방법은 아래와 같이 일반적으로 CS FHE 도식이 구현되는 방법에 적용될 수 있다.

[0106] HCS.Setup(1^λ) :

[0107] 1. $\lfloor \log g \rfloor + 1 = l$ 이라고 할 때 $\mathbb{Z}_g$ 를 메세지 공간으로 한다. CS.Setup(1^λ)을 실행하고, 평가키와 비밀키인 $\mathsf{ek} = (y'_0, \ldots y'_{\ell-1}, z)$ 와 p를 얻는다.

[0108] 2. 랜덤 순열인 $\pi_\gamma \xleftarrow{s} S_\gamma$ 과 그 역순열인 π_γ^{-1} 를 선택한다.

[0109] 3. 연산 파라미터(evaluation parmeter)를 다음과 같이 하드닝 한다.

[0110] j= 0,...,l-1 이라고 할 때, $\mathsf{HD}(y'_j, \pi_\gamma) \rightarrow y_j \in \mathbb{Z}^\gamma$ 을 계산한다. 여기서 $z = (z'_{i,j})0 \le i,j < \gamma$ 이고, $w_{i,j} = z'_{\pi_\gamma(i),\pi_\gamma(j)}$ 로 정한다. 그리고 아래 수학식 7을 계산한다.

[0112] <수학식 7>

[0113] $$\mathsf{HD}(w_{i,j}, \pi_\gamma) \rightarrow w_{i,j} \in \mathbb{Z}^\gamma \text{ for } 0 \le i,j < \gamma$$

[0115] 4. 최종적으로, $\mathsf{ek} = ((y_i)_{0\le i<\ell}, (w_{i,j})_{0\le i,j<\gamma})$ 이고, 비밀로 유지되는 $\mathsf{sk} = (p, \pi_\gamma, \pi_\gamma^{-1})$ 이다.

[0116]

[0117] HCS.Enc(m,sk) :

[0118] 암호화 하고자 하는 데이터인 m을 l 비트의 m_i 로 파싱한다. 즉 $m = \sum_{i=0}^{\ell-1} m_i 2^i$ 로 나타낼 수 있다. I를 m_i=1 과 같이 나타낼 수 있는 i들의 집합이라고 한다.

[0119] 이제 $x_0, \ldots, x_{\ell-1} \xleftarrow{\$} \mathcal{D}_{\gamma,\rho}(p)$ 를 선택하고,

[0120] $\sum_{i\in I} \mathsf{HD}(x_i + \left\lfloor \frac{p}{g} \right\rceil 2^i, \pi_\gamma) + \sum_{i\in[0,\ell-1]\setminus I} \mathsf{HD}(x_i, \pi_\gamma)$ 를 출력한다.

[0122] HCS.Add(c_1,c_2) :

[0123] c_1+c_2 연산을 수행하고, 여기서 '+' 는 벡터공간 Z^r 위에서의 합을 뜻한다.

[0125] HCS.Mult(c_1,c_2,ek) :

[0126] $\langle c_1 \otimes c_2, (w_{i,j})_{0\le i,j<\gamma} \rangle$ 연산을 수행한다. 여기서 이진 연산자인 $\langle (a_{i,j})_{i,j}, (b_{i,j})_{i,j} \rangle$ 는 $\sum_{i,j} a_{i,j} b_{i,j} \in \mathbb{Z}^\gamma \text{ for } a_{i,j} \in \mathbb{Z} \text{ and } b_{i,j} \in \mathbb{Z}^\gamma$ 로서 정의된다.

[0128] HCS.ConOp(m,ek) :

[0129] $\sum_{i=0}^{\ell-1} m_i y'_i$ 연산을 수행하며 여기서 $m = \sum_{i=0}^{\ell-1} m_i 2^i$ 이다.

[0131] HCS.Dec(c,sk) :

[0132] $\left\lfloor \mathsf{RC}(c, \pi_\gamma^{-1}) \frac{g}{p} \right\rceil \pmod g$ 연산을 수행하며 이에 따라 암호문 c를 복호와 한다.

[0134] 위와 같이 본 발명에 따른 하드닝 방법을 포함하는 대칭키 기반 동형 암호 시스템은 하드닝 연산 과정을 통하여 곱셈구조가 무너졌음에도 scale-invariant technique을 이용하여 여전히 곱셈성질을 보존시킬 수 있으며 이에 따라 동형암호를 설계할 수가 있다.

[0135] 상술한 하드닝 방법이 적용되는 암호 방법은 컴퓨터 장치에서 구현될 수 있으며 예를 들면 1개의 컴퓨터 장치 내에서 구현되거나 또는 2개 이상의 컴퓨터 장치로 분산되어 구현될 수 있고, 각 단계는 하드웨어(예를 들면, 칩) 및/또는 컴퓨터 프로그램 코드에 의해 구현될 수 있다.

[0136] 본 발명에 따른 하드닝 방법을 이용하는 FHE 암호 시스템인 하드닝 ACD 인스턴스 기반 동형암호 시스템(300)은 셋업연산부(310), 암호화부(320), 복호화부(330)를 포함한다. 또한, CS연산부(340)를 더 포함할 수 있다.

[0137] 셋업연산부(310)는 HCS.Setup(1^λ) 알고리즘에 따른 연산을 수행할 수 있다. 암호화부(320)는 HCS.Enc(m,sk) 알고리즘에 따른 연산을 수행할 수 있다. 복호화부(330)는 HCS.Dec(c,sk) 알고리즘에 따른 연산을 수행할 수 있다. CS연산부(340)는 HCS.Add(c_1,c_2), HCS.Mult(c_1,c_2,ek), HCS.ConOp(m,ek) 알고리즘에 따른 연산을 수행할 수 있다. 여기서 셋업연산부(310), 암호화부(320)는 본 발명에 따른 ACD 인스턴스 하드닝 장치(100)를 이용한다. 복호화부(330)는 본 발명에 따른 ACD 인스턴스 하드닝 복구 장치(200)를 이용한다.

[0138] 본 발명에 따른 하드닝 연산 방법을 적용한 ACD 인스턴스를 이용하여 설계하는 Hardened CS 도식의 암호문은 $\omega((\eta-\rho)\log\lambda)$로 기존 CS 도식에서 $\omega((\eta-\rho)^2\log\lambda)$의 크기를 가졌던 것에 비해 암호문의 크기를 매우 짧게 할 수 있다. 아래 표 1은 다른 ACD 기반의 동형 암호와 암호문 크기를 비교한 표이다.

[0140] <표 1>

FHE Scheme	Evaluation Key Size	CT Size	Underlying Hard Problem
CLT14 [11]	$\mathcal{O}(L^3\lambda^4)$	$\mathcal{O}(L^2\lambda^3)$	ACD
CS15 [7]	$\mathcal{O}(L^6\lambda^7)$	$\mathcal{O}(L^2\lambda^3)$	ACD
Ours	$\mathcal{O}(L^3\lambda^4)$	$\mathcal{O}(L\lambda^2)$	Hardened ACD

λ : security parameter, L : multiplicative dapth

[0141]

[0143] 이 밖에도 ACD를 기반으로 한 동형암호에 하드닝 방법을 적용하여 암호문의 크기를 효율적으로 짧게 만들 수 있으며, 동형 암호를 설계하기 위한 도구로 쓰이는 LWE, RLWE 등에도 하드닝 방법을 적용할 수도 있을 것이다.

[0144] 전술한 본 발명의 설명은 예시를 위한 것이며, 본 발명이 속하는 기술분야의 통상의 지식을 가진 자는 본 발명의 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 쉽게 변형이 가능하다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시 예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 예를 들어, 단일형으로 설명되어 있는 각 구성 요소는 분산되어 실시될 수도 있으며, 마찬가지로 분산된 것으로 설명되어 있는 구성 요소들도 결합된 형태로 실시될 수 있다.

[0145] 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 균등 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

부호의 설명

[0148] 100 : ACD 인스턴스 하드닝 장치

110 : 하드닝 데이터 입력부

120 : 하드닝 연산부

200 : ACD 인스턴스 하드닝 복구 장치

210 : 하드닝된 데이터 입력부

220 : 하드닝 복구 연산부

300 : 하드닝 ACD 인스턴스 기반 동형 암호 시스템

310 : 셋업연산부

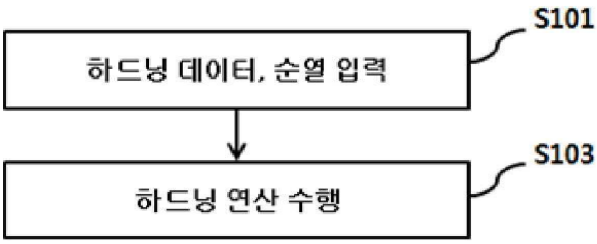
320 : 암호화부

330 : 복호화부

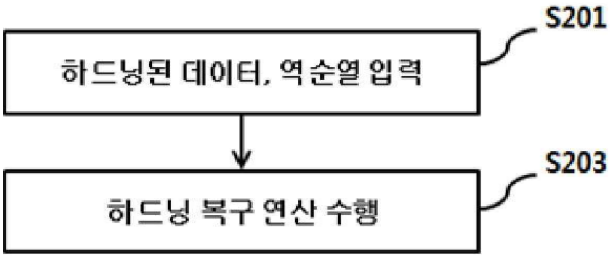
340 : CS연산부

도면

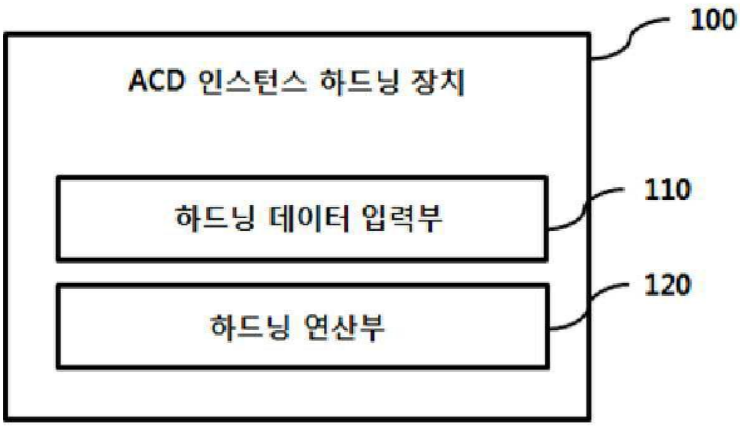
도면1



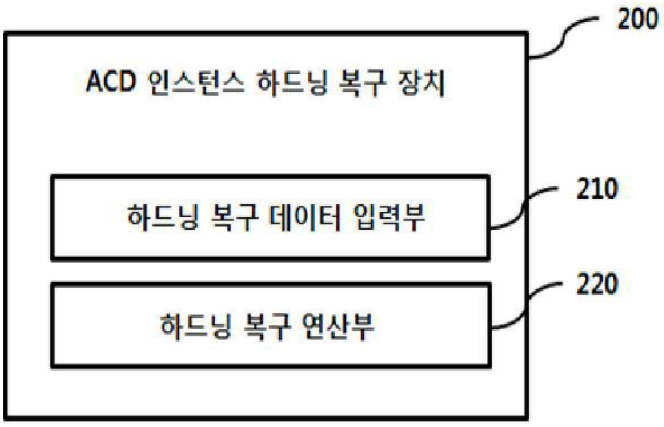
도면2



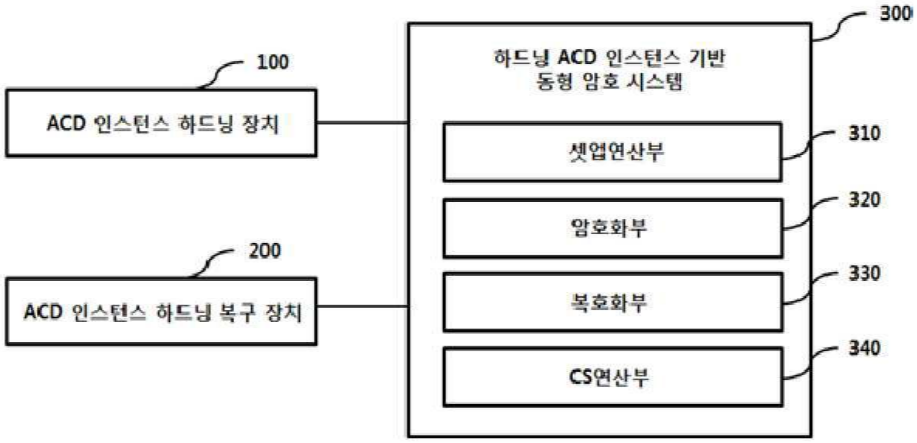
도면3



도면4


200
ACD 인스턴스 하드닝 복구 장치
210
하드닝 복구 데이터 입력부
220
하드닝 복구 연산부

도면5


300
하드닝 ACD 인스턴스 기반
동형 암호 시스템
100
ACD 인스턴스 하드닝 장치
200
ACD 인스턴스 하드닝 복구 장치
310
셋업연산부
320
암호화부
330
복호화부
340
CS연산부