



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0067963
(43) 공개일자 2018년06월21일

(51) 국제특허분류(Int. Cl.)
H04L 9/00 (2006.01) H04L 9/06 (2006.01)
(52) CPC특허분류
H04L 9/003 (2013.01)
H04L 9/065 (2013.01)
(21) 출원번호 10-2016-0169432
(22) 출원일자 2016년12월13일
심사청구일자 없음

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암
동5가)
(72) 발명자
고가영
서울특별시 성북구 장위로51길 17, 나동 102호
(74) 대리인
심경식, 홍성욱

전체 청구항 수 : 총 1 항

(54) 발명의 명칭 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법

(57) 요약

복수의 입력 상수 스트림 각각의 좌측 절반 값 및 우측 절반 값 각각에 대응되는 복수의 좌측 절반 상수 스트림 및 우측 절반 상수 스트림에 기초하여, 복수의 제1 중간 상수 스트림을 생성하는, 제1 중간 상수 스트림 생성부, 복수의 제1 중간 상수 스트림에 기초하여, 복수의 제2 중간 상수 스트림을 생성하는, 제2 중간 상수 스트림 생성부 및 복수의 제2 중간 상수 스트림에 기초하여, 인코딩된 최종 인코딩 상수 스트림을 생성하는 인코딩 상수 스트림 생성부를 포함하는, 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법에 관한 것이다.

대표도 - 도1



(52) CPC특허분류

H04L 2209/34 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 1711026432

부처명 미래창조과학부

연구관리전문기관 정보통신기술진흥센터

연구사업명 대학 ICT연구센터 육성지원사업

연구과제명 무결점 보안을 위한 지상/위성 양자통신 기술개발

기 여 율 1/1

주관기관 고려대학교(산학협력단)

연구기간 2015.06.01 ~ 2018.12.31

명세서

청구범위

청구항 1

복수의 입력 상수 스트림 각각의 좌측 절반 값 및 우측 절반 값 각각에 대응되는 복수의 좌측 절반 상수 스트림 및 우측 절반 상수 스트림에 기초하여, 복수의 제1 중간 상수 스트림을 생성하는, 제1 중간 상수 스트림 생성부;

상기 복수의 제1 중간 상수 스트림에 기초하여, 복수의 제2 중간 상수 스트림을 생성하는, 제2 중간 상수 스트림 생성부; 및

상기 복수의 제2 중간 상수 스트림에 기초하여, 인코딩된 최종 인코딩 상수 스트림을 생성하는 인코딩 상수 스트림 생성부를 포함하는, 부채널 분석에 안전한 밸런스 인코딩 장치.

발명의 설명

기술 분야

[0001] 본 발명은 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법에 관한 것이다.

배경 기술

[0002] 암호학적으로 안전한 암호 알고리즘을 이용하여 중요 정보를 암호화 하더라도 실제 운용 과정에서 발생하는 부가적인 정보(전력, 전자기파 등)들로 인해 비밀 정보가 노출될 수 있다.

[0003] 부채널 분석에 대한 개념이 제시된 이후로, 이러한 비밀 정보 노출을 방지하기 위하여, 부채널 분석을 방어하기 위한 대응기법(countermeasure)에 대한 연구가 활발히 진행되고 있으며, 대표적인 알고리즘 적 부채널 분석 대응기법으로는 마스킹(Masking) 과 하이딩(Hiding)이 있다.

[0004] 마스킹은 암호 알고리즘이 연산되는 동안 비밀정보를 포함하고 있는 중간 값에 랜덤 값을 이용하여 공격자가 중간 값을 예측할 수 없도록 하는 방법이며, 하이딩은 디바이스 내에서 암호 알고리즘이 연산되면서 발생하는 leakage(전력, 전자기파 등)가 비밀정보를 포함하고 있는 데이터와 관련성이 없는 것처럼 보이도록 만드는 방법이다.

[0005] 한편, 이러한 부채널 분석에 대응하기 위한 하드웨어적 부채널 분석 대응기법은, 부채널 대응기법으로 사용되고 있는 예로 Dual-Rail with Precharge Logic(이하 DPL)이 있으며, 이러한 DPL을 사용하여 암호 알고리즘을 구현 하면 데이터가 연산되는 과정에서 데이터의 해밍 웨이트와 해밍 디스턴스가 항상 일정하여 부채널 분석 위협을 완화시킬 수 있다.

[0006] 그러나, 종래의 방법은 제한적인 암호 알고리즘에 대해서만 적용이 가능한 문제가 있어, 다양한 구조 기반의 대칭키 암호 알고리즘에 적용이 어렵다는 문제가 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 공개특허공보 제10-2006-0068177호 (2006.06.21.)

발명의 내용

해결하려는 과제

[0008] 본 발명의 목적은, 상기 문제점을 해결하기 위한 것으로, 복수의 입력 상수 스트림 각각의 좌측 절반 값 및 우측 절반 값 각각에 대응되는 복수의 좌측 절반 상수 스트림 및 우측 절반 상수 스트림에 기초하여, 복수의 제1

중간 상수 스트림을 생성하고, 복수의 제1 중간 상수 스트림에 기초하여, 복수의 제2 중간 상수 스트림을 생성하며, 복수의 제2 중간 상수 스트림에 기초하여, 인코딩된 최종 인코딩 상수 스트림을 생성하기 위함이다.

[0009] 본 발명이 해결하고자 하는 과제는 이상에서 언급한 과제(들)로 제한되지 않으며, 언급되지 않은 또 다른 과제(들)은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0010] 상기한 목적을 달성하기 위하여, 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 장치는 복수의 입력 상수 스트림 각각의 좌측 절반 값 및 우측 절반 값 각각에 대응되는 복수의 좌측 절반 상수 스트림 및 우측 절반 상수 스트림에 기초하여, 복수의 제1 중간 상수 스트림을 생성하는, 제1 중간 상수 스트림 생성부, 복수의 제1 중간 상수 스트림에 기초하여, 복수의 제2 중간 상수 스트림을 생성하는, 제2 중간 상수 스트림 생성부 및 복수의 제2 중간 상수 스트림에 기초하여, 인코딩된 최종 인코딩 상수 스트림을 생성하는 인코딩 상수 스트림 생성부를 포함한다.

발명의 효과

[0011] 본 발명의 일 실시예에 따르면, 복수의 입력 상수 스트림 각각의 좌측 절반 값 및 우측 절반 값 각각에 대응되는 복수의 좌측 절반 상수 스트림 및 우측 절반 상수 스트림에 기초하여, 복수의 제1 중간 상수 스트림을 생성하고, 복수의 제1 중간 상수 스트림에 기초하여, 복수의 제2 중간 상수 스트림을 생성하며, 복수의 제2 중간 상수 스트림에 기초하여, 인코딩된 최종 인코딩 상수 스트림을 생성하여, 추가적인 저장 공간이 없이도 다양한 대칭키 암호 알고리즘에 적용될 수 있다.

도면의 간단한 설명

[0012] 도 1은 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 장치를 설명하기 위한 구성도이다.
 도 2는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 1 실시예인 콘스탄트 앤드(Constant AND)를 설명하기 위한 도면이다.
 도 3은 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 2 실시예인 콘스탄트 쉬프트(Constant Shift)를 설명하기 위한 도면이다.
 도 4는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 2 실시예인 콘스탄트 쉬프트(Constant Shift)를 설명하기 위한 도면이다.
 도 5는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 3 실시예인 콘스탄트 애드(Constant ADD)를 설명하기 위한 도면이다.
 도 6, 도 7a, 도 7b, 도 7c, 도 7d 및 도 8은 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법 안정성 검증을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0013] 이하, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 정도로 상세히 설명하기 위하여, 본 발명의 가장 바람직한 실시예를 첨부 도면을 참조하여 설명하기로 한다. 우선 각 도면의 구성요소들에 참조부호를 부가함에 있어서, 동일한 구성요소들에 대해서는 비록 다른 도면상에 표시되더라도 가능한 한 동일한 부호를 가지도록 하고 있음에 유의해야 한다. 또한, 본 발명을 설명함에 있어, 관련된 공지 구성 또는 기능에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명은 생략한다.

[0014] 이하, 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법을 첨부된 도면을 참조하여 상세하게 설명하면 아래와 같다.

[0015] 도 1은 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 장치를 설명하기 위한 구성도이다. 도 2는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 1 실시예인 콘스탄트 앤드(Constant AND)를 설명하기 위한 도면이다. 도 3은 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 2 실시예인 콘스탄트 쉬프트(Constant Shift)를 설명하기 위한 도면이다. 도 4는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 2 실시예인 콘스탄트 쉬프트(Constant Shift)를

설명하기 위한 도면이다. 도 5는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 3 실시예인 콘스탄트 애드(Constant ADD)를 설명하기 위한 도면이다. 도 6, 도 7a, 도 7b, 도 7c, 도 7d 및 도 8은 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법 안정성 검증을 설명하기 위한 도면이다.

- [0016] 도 1에 도시된 바와 같이, 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 장치(100)는 제1 중간 상수 스트림 생성부(110), 제2 중간 상수 스트림 생성부(120) 및 인코딩 상수 스트림 생성부(130)를 포함한다.
- [0017] 제1 중간 상수 스트림 생성부(110)는 복수의 입력 상수 스트림 각각의 좌측 절반 값 및 우측 절반 값 각각에 대응되는 복수의 좌측 절반 상수 스트림 및 우측 절반 상수 스트림에 기초하여, 복수의 제1 중간 상수 스트림을 생성한다.
- [0018] 제2 중간 상수 스트림 생성부(120)는 복수의 제1 중간 상수 스트림에 기초하여, 복수의 제2 중간 상수 스트림을 생성한다.
- [0019] 인코딩 상수 스트림 생성부(130)는 복수의 제2 중간 상수 스트림에 기초하여, 인코딩된 최종 인코딩 상수 스트림을 생성한다.
- [0020] 예컨대, 하드웨어 기반의 Dual-rail Logic 스타일을 소프트웨어로 구현한 밸런스 인코딩 기법은 추가적인 저장 공간이 필요 없는 효과적인 부채널 분석 대응방법이다.
- [0021] 예를 들어, 본 발명의 실시예에 따른 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법을 통한, 밸런스 인코딩 기법을 이용하여 암호 알고리즘을 구현하면 암호 알고리즘이 연산되는 동안 입력 값에 상관없이 비밀 정보를 포함하고 있는 중간 값은 항상 일정한 해밍 웨이트 및 해밍 디스턴스를 유지하게 되어 부채널 분석을 어렵게 만드는 효과가 있다.
- [0022] 예를 들어, 본 발명의 실시예에 따른 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법은, ARX 구조 기반의 다양한 대칭키 암호 알고리즘에도 적용이 가능하고, 효율적인 메모리 관리를 위해 Look-up table을 사용하지 않는 새로운 Constant AND, Constant Shift 연산 알고리즘을 포함할 수 있다.
- [0023] 예를 들어, 본 발명의 실시예에 따른 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 장치 및 방법은 제 1 실시예인 콘스탄트 앤드(Constant AND), 제 2 실시예인 콘스탄트 쉬프트(Constant Shift) 및 제 3 실시예인 콘스탄트 애드(Constant ADD) 중 어느 하나를 포함할 수 있다.
- [0024] 예컨대, 제 1 실시예인 콘스탄트 앤드(Constant AND), 제 2 실시예인 콘스탄트 쉬프트(Constant Shift) 및 제 3 실시예인 콘스탄트 애드(Constant ADD) 각각은 8비트 레지스터를 사용하는 디바이스를 대상으로 한 Atmel AVR 8-bit instruction set을 활용할 수 있으나, 본 발명은 이에 한정되지 않는다.
- [0025] 예컨대, Atmel AVR 8-bit instruction set을 활용하는 경우, 해당되는 주요 명령어에 대한 기능은 종래에 공지된 바와 같으므로, 구체적인 설명은 생략한다.
- [0026] 제 1 실시예인 콘스탄트 앤드(Constant AND), 제 2 실시예인 콘스탄트 쉬프트(Constant Shift) 및 제 3 실시예인 콘스탄트 애드(Constant ADD) 각각의 INPUT 값은 nibble에 $\overline{b_3}b_3\overline{b_2}b_2\overline{b_1}b_1\overline{b_0}b_0$ 형태의 밸런스 인코딩을 적용할 수 있으며, 각각의 실시예에서는 임의 값이 레지스터에 저장되는 것을 방지하기 위해 레지스터를 사용하기 전 “CLR” 명령어로 초기화 하였으며, 중간 값의 해밍 웨이트 및 해밍 디스턴스가 입력 값과 무관하게 일정하도록 설계할 수 있으나, 본 발명은 이에 한정되지 않는다.
- [0027] 도 2에 도시된 바와 같이, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 1 실시예인 콘스탄트 앤드(Constant AND)는, 입력 값을 $x: \overline{x_3}x_3\overline{x_2}x_2\overline{x_1}x_1\overline{x_0}x_0$, $y: \overline{y_3}y_3\overline{y_2}y_2\overline{y_1}y_1\overline{y_0}y_0$ 라 하고, 이를 AND 연산한 결과 값을 $z: \overline{z_3}z_3\overline{z_2}z_2\overline{z_1}z_1\overline{z_0}z_0$ 로 정의하며, 도 2에 도시된 바와 같은 총 3가지 단계를 포함할 수 있다.
- [0028] 제 1 단계(STEP 1)는 입력 값 x를 절반으로 나눈 후 “AND”, “SWAP” 명령어를 비롯한 각종 명령어를 사용하여 $x_L: \overline{x_3}x_3\overline{x_2}x_2\overline{x_3}x_3\overline{x_2}x_2$ 를 생성하며, 상술한 과정과 동일한 과정을 활용하여, $x_R: \overline{x_1}x_1\overline{x_0}x_0\overline{x_1}x_1\overline{x_0}x_0$,

y_L, y_R 을 생성하는 단계를 의미할 수 있다.

- [0029] 제 2 단계(STEP 2)는 제 1 단계에서 생성한 x_L, y_L 를 이용하여 z_L 을 생성하고, 동일 방법으로 x_R, y_R 를 이용하여 z_R 를 생성하는 단계를 의미할 수 있다.
- [0030] 도 2에 도시된, 6~9번째 줄은 중간 값의 일정한 해밍 웨이트와 해밍 디스턴스를 위해 추가적인 단계를 더 포함하게 되며 이에 대한 설명은 후술하도록 한다.
- [0031] 제 3 단계(STEP 3)는 입력 값 z_L, z_R 를 이용하여 $x \wedge y$ 연산의 결과 값인 z 를 최종적으로 구성하는 단계를 의미할 수 있다.
- [0032] 예컨대, 제 3 단계에서는 SWAP과 OR 명령어를 활용할 수 있으나, 본 발명은 이에 한정되지 않는다.
- [0033] 도 3 및 도 4에 도시된 바와 같이, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 2 실시예인 콘스턴트 쉬프트(Constant Shift)는 총 4가지의 알고리즘을 더 포함할 수 있다.
- [0034] 예컨대, 제 2 실시예에 따른 콘스턴트 쉬프트(Constant Shift) 중 제1 콘스턴트 쉬프트(ConstShift 1) 및 제 3 콘스턴트 쉬프트(ConstShift 3)는 “MUL” 명령어를 활용할 수 있다.
- [0035] 예컨대, MUL 명령어는 두 개의 레지스터에 저장되어 있는 값을 곱한 후 그 결과 값을 r0, r1에 저장하는 명령어를 의미할 수 있으며, 제1 콘스턴트 쉬프트(ConstShift 1) 및 제 3 콘스턴트 쉬프트(ConstShift 3)에서는 입력 값 $r4 : \overline{x_3x_3x_2x_2x_1x_1x_0x_0}$ 와 $r5 : \overline{x_7x_7x_6x_6x_5x_5x_4x_4}$ 가 어떤 값과 곱해지느냐에 따라 r0과 r1에 저장되는 값들이 얼마나 왼쪽으로 shift 되는지 결정될 수 있다.
- [0036] 예컨대, 이에, 제1 콘스턴트 쉬프트(ConstShift 1)에서는 r4와 r5의 값들을 왼쪽으로 두 칸 shift 하기 위해 4번째 줄에서 r6에 '00000100'이라는 값을 저장하여 r4, r5에 곱해주었으며, 제 3 콘스턴트 쉬프트(ConstShift 3)에서는 r6에 '01000000'이라는 값을 저장하여 r4와 곱한 후 ANDi, SWAP, LSL, OR 명령어를 비롯한 각종 명령어를 활용하여 r4와 r5의 값들을 왼쪽으로 여섯 칸 shift 할 수 있다.
- [0037] 제2 콘스턴트 쉬프트(ConstShift 2)와 제4 콘스턴트 쉬프트(ConstShift 4)는 AND와 SWAP, MOV 명령어를 비롯한 각종 명령어를 활용하여, 입력 값을 왼쪽으로 각각 네 칸, 여덟 칸shift 할 수 있으며, 왼쪽으로 shift 후 '0'으로 채워지는 오른쪽 부분도 '10'과 같은 밸런스 인코딩 형태를 유지하도록 하였는데, 이를 통해 오류 주입 공격(Fault Injection Attack)에 의해 '00', '11'과 같이 비트 플립이 발생하여 동일한 비트 값이 연속되는 경우도 식별할 수 있다.
- [0038] 도 5에 도시된 바와 같이, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 3 실시예인 콘스턴트 애드(Constant ADD)는 입력 값 x, y 에 대해 'carry(올림 수)'가 발생할 가능성이 있는 경우'와 'carry가 반드시 발생하는 경우'에 대해 값을 미리 생성한 후, 그 값을 이용하여 4비트를 동시에 병렬(parallel) 처리함으로써 효율적인 연산이 가능하도록 고안된 하드웨어 기반 알고리즘을 의미할 수 있다.
- [0039] 예컨대, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 3 실시예인 콘스턴트 애드(Constant ADD)는 종래의 AND, Shift, XOR와 같은 Boolean 연산만으로 구현할 수 있도록 한 알고리즘인, Kogge-Stone Adder에 종래의 Constant XOR 연산과 본 발명의 제 1 실시예에 따른 콘스턴트 앤드(Constant AND) 및 제 2 실시예에 따른 콘스턴트 쉬프트(Constant Shift)를 함께 적용한 알고리즘을 의미할 수 있다.
- [0040] 본 발명의 실시예에 의하면, ARX 구조 기반의 다양한 대칭키 암호 알고리즘에도 적용이 가능하고, Look-up table을 사용하지 않도록 하여, 효율적인 메모리 관리를 할 수 있으며, 나아가, 본 발명의 실시예에 의하면, 초소형 무인항공기나 초소형 IoT 디바이스처럼 제한된 내부 저장 공간으로 인해 하드웨어 기반 암호화 방법을 적용하기 어려운 환경에서는 소프트웨어 암호화 방법을 제공할 수 있으며, 이러한 경우에 본 발명의 실시예에 따른 부채널 분석에 안전한 밸런스 인코딩 방법을 적용한다면 부채널 분석 위협도 완화시킬 수 있다.
- [0041] 예컨대, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법의 제 1 실시예인 콘스턴트 앤드(Constant AND)의 제 2 단계에서 중간 값과 그에 대한 해밍 웨이트 및 해밍 디스턴스를 분석함으로써 중간 값이 입력 값과 무관하게 일정한 해밍 웨이트와 해밍 디스턴스를 유지할 수 있다.
- [0042] 이제, 도 6, 도 7a, 도 7b, 도 7c, 도 7d 및 도 8을 참조하여, 본 발명의 실시예에 따른, 부채널 분석에 안전한

밸런스 인코딩 방법을 적용한 경우, 발생하는 leakage의 정보량이 그렇지 않은 경우에 비해 낮다는 것을 상호정보량 분석을 통해 확인한다.

[0043] 일반적으로 하나의 연산을 수행하기 위해서는 1 clock이 소모되는데, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하면 하나의 연산을 수행하는데 여러 clock이 소모된다.

[0044] 이는 unprotected case에서 leakage는 한 포인트에서 발생하지만, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용을 적용하면 동일한 leakage가 여러 포인트에서 발생하기 때문에 공격자에게 오히려 많은 정보를 제공할 우려가 있다는 것을 의미한다.

[0045] 따라서 이하에서는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하지 않은 중간 값(unprotected)을 기준으로 하여, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용한 콘스탄트 앤드(Constant AND)와 콘스탄트 쉬프트(Constant Shift) 1 내지 4에서 해밍 웨이트가 4인 중간 값(protected)을 대조군으로 설정하여 시뮬레이션을 수행하며, 공격자는 PoI(Point of Interest) 검출을 통해 동일 leakage가 발생하는 여러 포인트를 찾아서 공격에 사용할 수 있다고 가정하여 상호정보량 분석을 설명한다.

[0046] 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 실험하기 위한 시뮬레이션은 아래와 같은 4가지 변수(X축)의 값들을 변화시켜가며 실시하였고, 각 실험을 100번 반복 후 상호 정보량(Y축)의 평균을 계산하였다.

[0047] 도 7a에 도시된 시뮬레이션 1은 과형의 개수(n), 도 7b에 도시된 시뮬레이션 2는 variance of noise(σ_a), 도 7c에 도시된 시뮬레이션 3은 bit leakage weights variance(σ_b), 도 7d에 도시된 시뮬레이션 4는 장비별 bit weights variance(σ_c)을 의미할 수 있다.

[0048] 예컨대, 시뮬레이션 결과에 대한 안전성 검증은 MATLAB R2015a를 이용하여 상호 정보량 분석을 통해 실시할 수 있으며 상호 정보량 분석이란 두 개의 서로 다른 random variable X, Y에 대해 X가 Y의 정보를 어느 정도 가지고 있는지 판단하는 척도로서 아래 수학적 식 1과 같이 산출될 수 있다.

[0049] 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 검증하기 위하여, X는 데이터(unprotected / protected)를, Y는 부채널 leakage를 의미하며, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하지 않은 경우 발생하는 부채널 leakage에 포함된 중간 값의 정보량과 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하였을 때 발생하는 부채널 leakage에 포함된 중간 값의 정보량을 비교하였다.

[0050] 이를 통해 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하면 동일 leakage가 여러 포인트에서 발생하더라도 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하지 않았을 때보다 leakage를 통해 누출되는 정보량이 적다는 것을 확인할 수 있다.

[0051] [수학적 식 1]

$$I(X; Y) = H(X) - H(X|Y)$$

[0052]

[0053] 시뮬레이션을 통해 상호 정보량을 측정한 결과 모든 경우에서 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하였을 때 상호 정보량이 더 낮게 측정되었다. 이는 데이터에 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용하면 leakage가 비밀정보를 포함하는 중간 값과 관련이 없는 것처럼 보이도록 하는 대응기법으로서의 목적을 달성했다고 할 수 있다.

[0054] 도 7a에 도시된, 시뮬레이션 1의 경우, 과형의 개수가 많아질수록 상호 정보량은 특정한 값으로 수렴하지만, 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용한 경우 상호 정보량이 그렇지 않은 경우에 비해 현저히 낮다.

[0055] 도 7b에 도시된, 시뮬레이션 2와 도 7c에 도시된 시뮬레이션 3의 경우도 noise가 커질수록 정확한 분석은 제한되지만, noise가 적은 상황에서는 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용했을 때 상호 정보량 값이 확연히 낮음을 확인할 수 있다.

[0056] 예컨대, 레지스터의 공정변이 개념을 적용한 도 7d에 도시된, 시뮬레이션 4는 장비별 비트 평균의 분산이 커질수록 본 발명의 실시예에 따른, 부채널 분석에 안전한 밸런스 인코딩 방법을 적용했을 때 상호 정보량값이 급격

하게 증가하였는데, 이는 반대로 비트마다 특성이 확연해 질수록, 즉 비트 값에 따른 평균 전력소모량이 0과 1에 가까울수록 부채널 분석이 용이하다는 것을 보여준다.

[0057] 이상에서 본 발명에 따른 바람직한 실시예에 대해 설명하였으나, 다양한 형태로 변형이 가능하며, 본 기술분야에서 통상의 지식을 가진 자라면 본 발명의 특허청구범위를 벗어남이 없이 다양한 변형예 및 수정예를 실시할 수 있을 것으로 이해된다.

부호의 설명

[0058] 100: 부채널 분석에 안전한 밸런스 인코딩 장치
110: 제1 중간 상수 스트림 생성부
120: 제2 중간 상수 스트림 생성부
130: 인코딩 상수 스트림 생성부

도면

도면1



도면2

Algorithm 1 ConstAND

STEP 1 <Generate $x_D x_R y_D y_R$ >

Input: $r4 = x(x : \overline{x_3 x_2 x_1 x_0})$

Output: $r5 = x_L(x_L : \overline{x_3 x_2 x_1 x_0})$

```

1  CLR    r5
2  CLR    r6
3  LDi    r5, b 11110000
4  LDi    r6, b 11110000
5  AND    r5, r4
6  AND    r6, r4
7  SWAP   r6
8  OR     r5, r6

```

STEP 2 <Generate $z_D z_R$ >

Input: $r4 = x_L(x_3 x_2 x_1 x_0)$,

$r5 = y_L(y_3 y_2 y_1 y_0)$

Output: $r6 = z_L(0000 z_3 z_2 z_1)$

```

1  CLR    r6
2  LDi    r6, b 11110000
3  EOR    r6, r4
4  CLR    r7
5  MOV    r7, r6
6  AND    r6, r5
7  EOR    r7, r5
8  ANDi   r7, b 10101010
9  EOR    r6, r7
10 ANDi   r6, b 00001111

```

Algorithm 1 ConstAND

STEP 3 <Construct z >

Input: $r4 = z_L(0000 z_3 z_2 z_1)$,

$r5 = z_R(0000 z_1 z_0 z_0)$

Output: $r4 = z(z_3 z_2 z_1 z_0)$

```

1  SWAP   r4
2  OR     r4, r5

```

도면3

Algorithm 2 ConstShift 1

Input: $r4 = \overline{x_3x_3x_2x_2x_1x_1x_0x_0}$,
 $r5 = \overline{x_7x_7x_6x_6x_5x_5x_4x_4}$
Output: $r4 = \overline{x_2x_2x_1x_1x_0x_010}$,
 $r5 = \overline{x_6x_6x_5x_5x_4x_4x_3x_3}$

```

1  CLR    r0
2  CLR    r1
3  CLR    r6
4  LDi    r6, b 00000100
5  MUL    r4, r6
6  ORi    r0, b 00000010
7  CLR    r4
8  MOV    r4, r0
9  CLR    r7
10 MOV    r7, r1
11 CLR    r0
12 CLR    r1
13 MUL    r5, r6
14 OR     r0, r7
15 CLR    r5
16 MOV    r5, r0

```

Algorithm 3 ConstShift 2

Input: $r4 = \overline{x_3x_3x_2x_2x_1x_1x_0x_0}$,
 $r5 = \overline{x_7x_7x_6x_6x_5x_5x_4x_4}$
Output: $r4 = \overline{x_1x_1x_0x_01010}$,
 $r5 = \overline{x_5x_5x_4x_4x_3x_3x_2x_2}$

```

1  CLR    r6
2  LDi    r6, b 11110000
3  AND    r6, r4
4  SWAP   r6
5  ANDi   r4, b 00001111
6  SWAP   r4
7  ORi    r4, b 00001010
8  ANDi   r5, b 00001111
9  SWAP   r5
10 OR     r5, r6

```

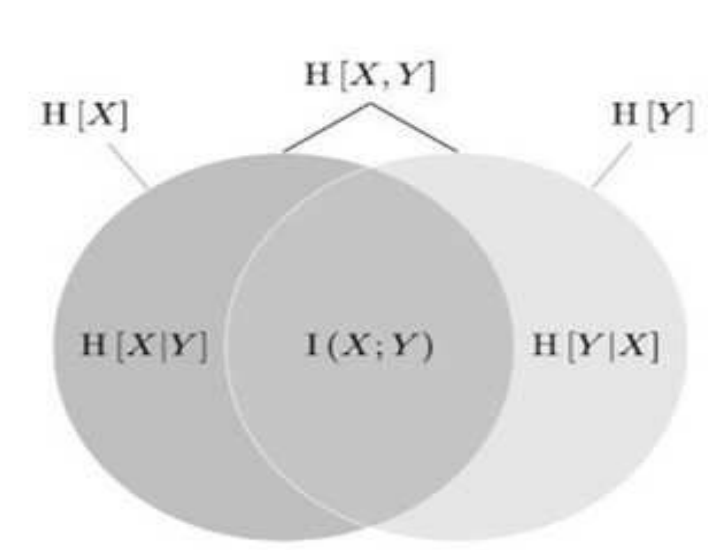
도면4

Algorithm 4 ConstShift 3		
Input: $r4 = \overline{x_3x_3x_2x_2x_1x_1x_0x_0}$, $r5 = \overline{x_7x_7x_6x_6x_5x_5x_4x_4}$		
Output: $r4 = \overline{x_0x_0}101010$, $r5 = \overline{x_4x_4x_3x_3x_2x_2x_1x_1}$		
1	CLR	r0
2	CLR	r1
3	CLR	r6
4	LDi	r6, b 01000000
5	MUL	r4, r6
6	ANDi	r5, b 00000011
7	SWAP	r5
8	LSL	r5, 2
9	OR	r5, r1
10	CLR	r4
11	MOV	r4, r0
12	ORi	r4, b 00101010
Algorithm 5 ConstShift 4		
Input: $r4 = \overline{x_3x_3x_2x_2x_1x_1x_0x_0}$, $r5 = \overline{x_7x_7x_6x_6x_5x_5x_4x_4}$		
Output: $r4 = 10101010$, $r5 = \overline{x_3x_3x_2x_2x_1x_1x_0x_0}$		
1	CLR	r5
2	MOV	r5, r4
3	CLR	r4
4	ORi	r4, b 10101010

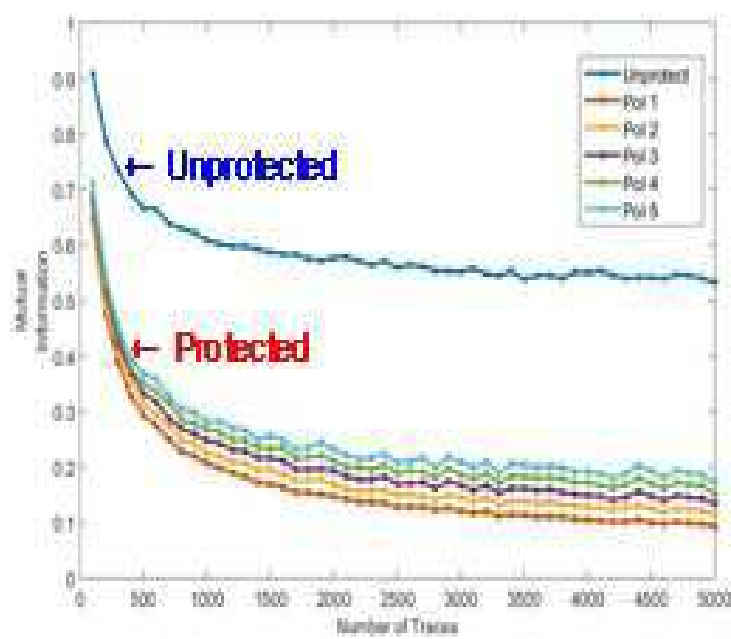
도면5

Algorithm 6 Kogge-Stone Adder	
Input: $x, y \in 0, 1^k$, and $n = \max(\lceil \log_2(k-1) \rceil, 1)$	
Output: $z = x + y \bmod 2^k$	
1	$P \leftarrow x \oplus y$
2	$G \leftarrow x \wedge y$
3	for $i := 1$ to $n-1$ do
4	$G \leftarrow (P \wedge (G \ll 2^{i-1})) \oplus G$
5	$P \leftarrow P \wedge (P \ll 2^{i-1})$
6	end for
7	$G \leftarrow (P \wedge (G \ll 2^{n-1})) \oplus G$
8	return $x \oplus y \oplus (2G)$

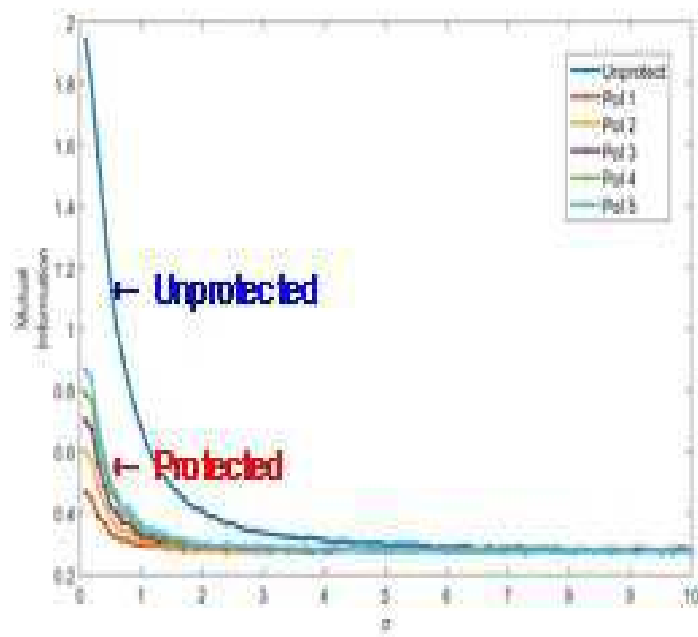
도면6



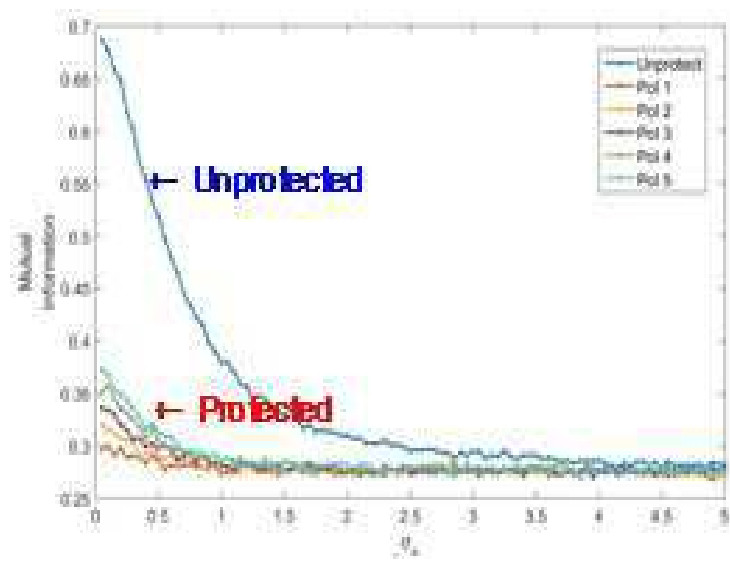
도면7a



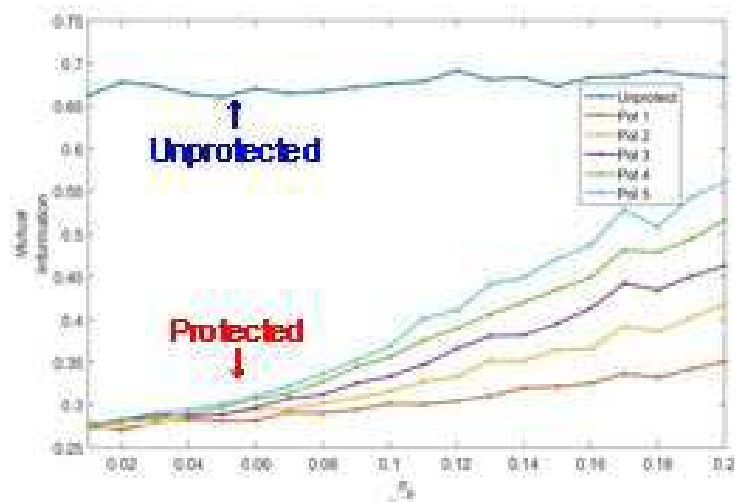
도면7b



도면7c



도면7d



도면8

Algorithm	Clock cycle
Constant AND	54 (STEP 1: 8×4, STEP 2: 10×2, STEP 3: 2)
Constant Shift 1	18
Constant Shift 2	10
Constant Shift 3	14
Constant Shift 4	4
Constant XOR [6]	47 (STEP 1: 8×4, STEP 2: 4×2, STEP 3: 7)
Constant ADD [10] (Kogge-Stone Adder)	637 (ConstAND: 6, ConstShift 1: 3, ConstShift 2: 2, ConstShift 4: 1, ConstXOR: 5)