(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0098172
(43) 공개일자 2018년09월03일

(51) 국제특허분류(Int. Cl.)
H04L 9/00 (2006.01) *G06F 7/76* (2006.01)
(52) CPC특허분류
H04L 9/003 (2013.01)
G06F 7/764 (2013.01)
(21) 출원번호 10-2018-0022242
(22) 출원일자 2018년02월23일
심사청구일자 2018년02월23일
(30) 우선권주장
17305202.8 2017년02월24일
유럽특허청(EPO)(EP)

(71) 출원인
시큐어-아이씨 에스에이에스
프랑스 쎄쏭-쎄비녜 35510, 작 데 샹 블랑, 뤼 끌로드 샤쁘 15
(72) 발명자
뉴엔 필리프
프랑스 35700 렌 뤼 뒤 뻬흐 부흐동 19
귀예 실뱅
프랑스 75013 파리 불러바흐 오귀스트 블랑키 99
(74) 대리인
특허법인아주김장리

전체 청구항 수 : 총 12 항

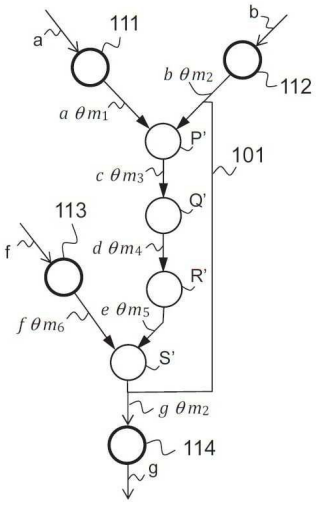
(54) 발명의 명칭 **알고리즘으로의 마스킹의 자동 삽입**

(57) 요 약

알고리즘의 호출 그래프 표현을 상기 알고리즘의 보안 호출 그래프 표현으로 변환하기 위한, 컴퓨터 구현 방법, 프로그램 제품, 및 상기 방법을 구현하는 시스템. 호출 그래프는, 입력(a, b, f), 그래프의 에지(c, d, e)인 내부 변수, 그래프의 노드인 기본 함수, 및 출력(g)을 포함하고, 상기 함수는 선형이거나 또는 비선형이며, 그 방법은 하기 단계들을 포함한다:

- 호출 그래프의 각각의 입력을 마스킹하는 단계,
- 호출 그래프의 각각의 마스킹되지 않은 내부 변수를 마스킹된 변수로 대체하는 단계,
- 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수로 대체하는 단계,
- 호출 그래프의 각각의 출력을 마스킹 해제하는 단계.

대 표 도 - 도1b



명 세 서

청구범위

청구항 1

알고리즘의 호출 그래프 표현(call graph representation)을 상기 알고리즘의 보안 호출 그래프 표현(secured call graph representation)으로 변환하기 위한 컴퓨터 구현 방법으로서,

상기 호출 그래프는 적어도 하나의 입력(a, b, f), 적어도 하나의 에지(c, d, e), 적어도 하나의 노드(P, Q, R, S) 및 적어도 하나의 출력(g)을 포함하고, 상기 호출 그래프의 상기 에지는 상기 알고리즘의 내부 변수를 나타내고, 상기 호출 그래프의 상기 노드는 상기 알고리즘의 선형 또는 비선형 기본 함수를 나타내고, 상기 컴퓨터 구현 방법은,

- 상기 호출 그래프의 각각의 입력을 마스킹하는 단계(601),

- 상기 호출 그래프의 각각의 마스킹되지 않은(unmasked) 내부 변수를 마스킹된 변수로 대체하는 단계(602),

- 상기 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수로 대체하는 단계(603),

- 상기 호출 그래프의 각각의 출력을 마스킹 해제하는(unmasking) 단계(604)를 포함하는, 컴퓨터 구현 방법.

청구항 2

제1항에 있어서, 상기 호출 그래프는 반복적으로 프로세싱되는 부분을 포함하고, 상기 마스킹되지 않은 내부 변수를 마스킹된 내부 변수에 의해 대체하는 단계(602)는, 반복적으로 프로세싱되는 상기 호출 그래프의 부분의 입력 및 출력 둘 다로서 사용되는 내부 변수(b, g)를 식별하는 단계, 및 상기 호출 그래프의 상기 부분의 입력 및 출력에서 이들 변수에 대해 동일한 마스크(m_2)를 사용하는 단계를 포함하는, 컴퓨터 구현 방법.

청구항 3

제1항에 있어서, 상기 호출 그래프는 반복적으로 프로세싱되는 부분을 포함하고, 상기 마스킹되지 않은 내부 변수를 마스킹된 내부 변수에 의해 대체하는 단계(602)는, 반복적으로 프로세싱되는 상기 호출 그래프의 부분의 입력 및 출력 둘 다로서 사용되는 내부 변수(b, g)를 식별하는 단계, 및 상기 내부 변수의 마스크를 수정하기 위한 추가 노드(120)를 상기 반복 부분의 피드백 에지에 삽입하는 단계를 포함하는, 컴퓨터 구현 방법.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서, 반복적으로 프로세싱되는 상기 호출 그래프의 부분에서의 내부 변수의 상기 마스크는 규칙적인 간격으로 변경되고, 관련된 함수가 상응하게 수정되는, 컴퓨터 구현 방법.

청구항 5

제4항에 있어서, 상기 반복 부분의 상기 내부 변수의 상기 마스크를 리프레쉬하기 위한 추가 노드(402)를 상기 호출 그래프에 삽입하는 단계를 더 포함하는, 컴퓨터 구현 방법.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서, 상기 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수로 대체하는 단계(603)에서 계산되는 등가 함수는 매치 테이블(match table)을 사용하여 구현되는, 컴퓨터 구현 방법.

청구항 7

제1항 내지 제6항 중 어느 한 항에 있어서, 상기 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수로 대체하는 단계(603)는, 상기 호출 그래프의 각각의 선형 함수를, 상기 입력 및 출력 내부 변수의 마스크를 고려하는 등가 함수에 의해 대체하는 단계를 더 포함하는, 컴퓨터 구현 방법.

청구항 8

제1항 내지 제7항 중 어느 한 항에 있어서, 모든 마스크 값은 랜덤하게 결정되는, 컴퓨터 구현 방법.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서, 보호된 실행 가능 코드를 생성하도록 상기 호출 그래프를 컴파일하는 추가 단계를 더 포함하는, 컴퓨터 구현 방법.

청구항 10

컴퓨터 시스템으로 하여금 제1항 내지 제9항 중 어느 한 항에 따른 방법을 수행하게 하기 위한 컴퓨터 실행 가능 명령어를 포함하는, 비휘발성 컴퓨터 판독 가능 데이터 저장 매체 상에 저장되는 컴퓨터 프로그램 제품.

청구항 11

컴퓨터 시스템으로 하여금 제1항 내지 제9항 중 어느 한 항에 따른 방법을 수행하게 하기 위한 컴퓨터 실행 가능 명령어를 포함하는 비휘발성 컴퓨터 판독 가능 데이터 저장 매체.

청구항 12

메모리에 커플링되는 프로세서를 포함하는 시스템으로서,

상기 메모리는, 상기 시스템으로 하여금 알고리즘의 호출 그래프 표현을 상기 알고리즘의 보안 호출 그래프 표현으로 변환하기 위한 컴퓨터 구현 방법을 수행하게 하기 위한 컴퓨터 실행가능 명령어를 저장하고, 상기 호출 그래프는 적어도 하나의 입력, 적어도 하나의 에지(c, d, e), 적어도 하나의 노드(P, Q, R, S) 및 적어도 하나의 출력(g)을 포함하며, 상기 호출 그래프의 상기 에지는 상기 알고리즘의 내부 변수를 나타내고, 상기 호출 그래프의 상기 노드는 상기 알고리즘의 선형 또는 비선형 기본 함수를 나타내며, 상기 시스템은 상기 프로세싱 디바이스를 포함하고, 상기 프로세싱 디바이스는,

- 상기 호출 그래프의 각각의 입력을 마스킹하도록,

- 상기 호출 그래프의 각각의 마스킹되지 않은 내부 변수를 마스킹된 변수로 대체하도록,

- 상기 호출 그래프의 적어도 각각의 비선형 함수를, 상기 마스킹된 변수에 적용되는 등가 함수에 의해 대체하도록, 그리고

- 상기 호출 그래프의 각각의 출력을 마스킹 해제하도록

구성되는, 메모리에 커플링되는 프로세서를 포함하는 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 암호학의 분야에 적용되며, 특히, 부채널 공격(side-channel attack)에 대한 알고리즘의 구현을 보호하기 위한 방법에 관한 것이다.

배 경 기 술

[0002] 민감한 데이터를 암호화하는 것을 수반하는 현존하는 암호화 알고리즘은, 암호 해독법 및 콘텐츠 복구 공격에 대해 효과적인 강건성을 제공한다. 공격자가 암호화 알고리즘의 입력 및 출력의 지식만을 가지고 있기 때문에, 이들 기술은 "블랙 박스 기술"로 칭해진다. 대부분의 암호화 알고리즘은 표준화되어 있으며, 알려져 있는 가장 많이 사용되는 것 중 하나는 고급 암호 표준(Advanced Encryption Standard; AES)이다. 암호화의 기밀성은 공유 비밀 암호 키에 기초한다. 공격자가 비밀 키를 무시하는 최상의 옵션은 모든 가능한 조합을 시도하는 것이다(무차별 대입(brute force) 디코딩). 키가 128 비트 또는 256 비트 길이인 경우, 필요로 되는 반복의 횟수는 무차별 대입 디코딩을 계산적으로 관리하기 매우 어렵게 만든다.

[0003] 그러나, 부채널 공격(Side-Channel Attack; SCA)으로 칭해지는 몇몇 공격은, 타이밍 정보, 전력 소비, 전자기 유출, 등등과 같은, 암호화 알고리즘의 물리적 구현으로부터 유출되는 정보에 기초하여 암호화 알고리즘에서 실

행되는 비밀 정보를 검색할 기회를 공격자에게 제공한다.

[0004] 따라서, 먼저, 이러한 공격에 대해 민감한 알고리즘의 구현을 보호하기 위한 방법을 제공할 필요가 있다.

[0005] 임의의 알고리즘은, 각각의 노드가 함수이고 각각의 에지가 중간 변수(내부 변수로도 또한 칭해짐)인 방향성 그래프(directed graph)인 호출 그래프(call graph), 또는 연산의 그래프로서 표현될 수 있다. 이 호출 그래프는 또한 데이터 플로우 그래프 또는 제어 플로우 그래프로 칭해질 수 있다.

[0006] 함수는 단일의 연산일 수 있거나, 또는, 선형의 또는 선형이 아닌 연산의 조합일 수 있다. 그것은 어떠한 점프도 없는 직선 조각의 코드이다. 함수가 복수의 피연산자를 포함하는 경우, 그것은 복수의 1진(unary) 또는 2진(binary) 피연산자로 분해될 수 있다.

[0007] 통상적인 연산은, 주어진 기술에서 구현될 수 있는 연산이다. 예를 들면, 소프트웨어 프로그램은, 덧셈('+') 또는 배타적 불린(exclusive boolean) OR('XOR')와 같은 산술 및 논리 연산을 계산할 수 있다. 디지털 신호 프로세서(Digital Signal Processor; DSP) 또는 필드 프로그래머블 게이트 어레이(field programmable gate array; FPGA)는 MAC(Multiply-ACcumulate) 유닛을 사용하여 룩업 테이블(look-up-table; LUT) 또는 산술 연산에서 구현되는 임의의 함수를 계산할 수 있다. 주문형 반도체(Application Specific Integrated Circuit; ASIC)는 표준 셀 라이브러리를 활용하여 임의의 타입의 연산을 계산할 수 있다.

[0008] 함수는 하이 레벨 언어로 표현될 수 있지만, 그러나 일련의 연산으로 또한 매핑될 수 있다. 하이 레벨 언어로 잠재적으로 기술되는 이러한 함수를, 프로세싱 시간 및 리소스 소비를 최적화하는 기계 언어로 변환하는 것은 컴파일러의 역할이다.

[0009] 알고리즘을 나타내는 그래프는 방향성 그래프이다: 각각의 노드 또는 함수는, 입력 인수만큼 많은 진입 에지(entering edge) 및 출력 결과만큼 많은 진출 에지(outgoing edge)를 갖는다. 예를 들면, 함수가 (두 개의 인수 및 하나의 결과를 갖는) 단순한 2진 연산인 경우, 두 개의 입력 및 하나의 출력이 있다.

[0010] 에지는, 노드에서 노드로 전달되는 타입을 갖는 변수(typed variable)를 전달한다. 타입은 바이트, 32 비트 워드, 등등일 수 있다.

[0011] 호출 그래프로서 설명되는, 암호 알고리즘과 같은 민감한 알고리즘을 고려하는 것 및, 알고리즘의 타입 또는 암호 알고리즘의 구현에 관한 임의의 고려 사항에 무관하게, 부채널 공격에 대해 암호 알고리즘을 보호하도록 알고리즘을 변환하는 것이 본 발명의 목적이다.

[0012] 부채널 공격에 대한 알고리즘의 강건성을 증가시키기 위해, 알고리즘의 중요한 데이터를 마스킹하는 것이 알려져 있다. 마스킹의 하나의 예는, 새로운 변수의 합이 초기 변수를 산출하도록 초기 변수를 복수의 새로운 변수로 분할하는 비밀 공유에 기초한다. 합계는 변수의 기저의 타입에 따라 이해되어야만 한다. 예를 들면, 변수가 한 바이트인 경우, 합은 비트 단위의 XOR 또는 모듈로 256 가산(addition modulo 256)일 수 있다.

[0013] 마스킹된 데이터에 영향을 주는 연산이 선형 함수인 경우, 함수의 마스킹된 출력의 값은 마스킹된 입력으로부터 계산될 수 있다. 그러나, 함수가 (예를 들면, 멱 함수(power function), 암호 알고리즘의 치환 박스(substitution box), ...와 같이) 비선형인 경우, 마스크 계산은 불가능할 수도 있을 것이다. 마스크는 함수의 입력에서 제거되어야만 하고, 새로운 마스크가 함수의 출력에 삽입되어야만 한다.

[0014] 다양한 마스킹 기술이 알려져 있으며, 그들 중 일부는 입증되어 있다. 그들은, 선형 호출 그래프인 직선 프로그램에 적용된다. 마스킹은, 적절할 때마다, 연산 사이의 마스크의 랜덤 재공유(또는 리프레쉬)를 갖는 체인화 연산(chaining operation)이다. 그러나, 그래프가 직선이 아니면, 몇몇 취약점이 나타날 수도 있다.

[0015] CHES 2010, 페이지 413-427의 M. Rivain 및 E. Proof에 의한 Provably secure higher-order masking of AES에는, 완전한 마스킹된 AES 알고리즘이 제시된다. 이 논문에서는, 특정한 선형 및 비선형 함수의 마스킹에 대해 설명되며, 완전한 AES 알고리즘을 설명하기 위해 함수는 체인화된다. 그러나, FSE 2013 페이지 11-13의 JE . Coron, E. Proof, M. Rivain 및 T. Roche에 의한 High-Order Side Channel Security and Mask Refreshing에서 나타내어지는 바와 같이, 심지어 AES 알고리즘에 특별히 전용되는 접근법에서도, 몇몇 구현 문제가 발생할 수 있다. 이들 구현 문제는 몇몇 변수의 재사용으로부터 발생하고, 그러므로, 달성된 보안 레벨은 감소된다.

[0016] 따라서, 오늘날, 대부분의 마스킹 구현은 수동으로 수행되는데, 이것은 구현 에러가 발생하기 쉽다(예를 들면, 민감한 변수는 마스킹되지 않음). 단지 몇몇 연구만이 자동 마스킹을 고려하고 있다.

[0017] 이들 연구 중에는, CHES 2012, 페이지 58-75의 A. Moss, E. Oswald에 의한 Compiler assisted masking,

Cryptographic Hardware and Embedded Systems의 논문이 있다. 이 논문에서는, 민감한 데이터는 프로그래머에 의해 주석이 달리고, 그들의 비밀성은, 컴파일러가 비밀성 정보를 프로그램을 통해 전달하는 것을 허용하는 격자에서 값으로서 처리된다. 일단 컴파일되면, 비밀 데이터는 프로그램 실행 동안 일반 텍스트로 절대 나타나지 않으며, 그에 의해, 특히 부채널 공격에 대해, 마스킹된 데이터의 비밀성을 보장하게 된다. 그 다음, 알고리즘은 프로그램의 모든 값에서, 특히, 표현식을 변환할 때 도입되는 임시 변수에서 민감한 정보 유출을 검색하는 단계를 수행하고, 유출이 발생하면, 프로그램 변환의 세트를 사용하여 유출을 방지하려고 시도한다.

[0018] 이 논문에서 밝혀지는 솔루션의 단점은, 그것이 오로지 1 차 불린(Boolean) 마스킹 스킴 및 직선 코드에만 적용된다는 것이다. 더욱이, 유출을 탐색하고 그러한 유출을 방지하기 위해 프로그램을 변환하는 단계는 수렴하도록 구속되지는 않는다.

[0019] 2014년 1월의 Springer International Publishing의 Eldib H., Wang C.에 의한 Synthesis of masking countermeasures against side channel attacks, Computer aided verification(페이지 114-130)에서는, 모든 중간 값을 포함하여, 전체 알고리즘을 마스킹하는 것이 제안된다. 비선형 함수에 대처하기 위해, 함수적으로 등가인 선형 함수를 결정하는 것, 그 함수가 모든 가능한 입력에 대해 동등하고 완벽하게 마스킹된다는 것을 확인하는 것이 제안된다.

[0020] 이 방법은, 불린 타입의 호출 그래프에 대해서만 적용되기 때문에, Moss 등등의 방법과 유사한 단점을 갖는다. 또한, 그 방법은, 실행 시간이 보장되지 않는 시행 착오 방법론을 따른다.

[0021] 따라서, 보호되지 않은 알고리즘을 상기 알고리즘의 보안 버전으로 변환하기 위한 완전히 자동적이고 강건한 방법에 대한 보다 엄밀한 요구가 있다.

발명의 내용

[0022] 이러한 문제점 및 다른 문제점을 해결하기 위해, 알고리즘의 호출 그래프 표현(call graph representation)을 상기 알고리즘의 보안 호출 그래프 표현(secured call graph representation)으로 변환하기 위한 컴퓨터 구현 방법이 제공된다. 호출 그래프는, 하나 이상의 입력, 하나 이상의 에지(c, d, e), 하나 이상의 노드(P, Q, R, S) 및 하나 이상의 출력(g)을 포함한다. 호출 그래프의 에지는 상기 알고리즘의 내부 변수(또는 중간 변수)를 나타내며, 호출 그래프의 노드는 알고리즘의 선형 또는 비선형 기본 함수를 나타낸다. 본 발명에 따른 컴퓨터 구현 방법은 다음을 포함한다:

[0023] • 호출 그래프의 각각의 입력을 마스킹하는 단계,

[0024] • 호출 그래프의 각각의 마스킹되지 않은 내부 변수를 마스킹된 변수로 대체하는 단계,

[0025] • 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수로 대체하는 단계,

[0026] • 호출 그래프의 각각의 출력을 마스킹 해제하는(unmasking) 단계.

[0027] 그 방법은 유익하게는 제한된 그리고 결정론적(deterministic) 실행 시간을 가지고 선형 및 비선형 호출 그래프 양자에 적용되고, 불린 타입의 연산으로 제한되지 않는다.

[0028] 보호되지 않은 알고리즘을 상기 알고리즘의 보안 버전으로의 변환하는 것은 두 가지 요구되는 특성을 충족시킨다:

[0029] • 의미론을 보존함, 즉, 원래의 프로그램과 비교하여, 결정된 입력의 세트로부터 동일한 결과가 획득, 및

[0030] • 모든 중간 변수(즉, 정점에 의해 전달되는 변수)를 마스킹.

[0031] 호출 그래프가 반복적으로 프로세싱되는 부분을 포함하는 경우, 마스킹되지 않은 내부 변수를 마스킹된 내부 변수에 의해 대체하는 단계는, 반복적으로 프로세싱되는 호출 그래프의 부분의 입력 및 출력 둘 다로서 사용되는 내부 변수를 식별하는 것, 및 호출 그래프의 상기 부분의 입력에서 그리고 출력에서 이들 변수에 대해 동일한 마스크를 사용하는 것에 의해 보장될 수도 있다.

[0032] 대안적으로, 마스킹되지 않은 내부 변수를 마스킹된 내부 변수에 의해 대체하는 단계는, 반복적으로 프로세싱되는 호출 그래프의 부분의 입력 및 출력 둘 다로서 사용되는 내부 변수를 식별하는 것, 및 상기 반복 부분의 피

드백 에지에 상기 내부 변수의 마스크를 수정하기 위한 추가적인 노드를 삽입하는 것에 의해 보장될 수도 있다.

[0033] 유익하게는, 반복적으로 프로세싱되는 호출 그래프의 부분에서의 내부 변수의 마스크는 규칙적인 간격으로 변경될 수도 있고, 관련 함수는 상응하게 수정된다. 이 메커니즘은 상기 반복 부분의 내부 변수의 마스크를 리프레쉬하기 위한 추가 노드를 호출 그래프에 삽입하는 것에 의해 달성될 수 있다.

[0034] 본 발명의 소정의 실시형태에 따른 컴퓨터 구현 방법에서, 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수로 대체하는 단계에서 계산되는 등가 함수는 매치 테이블(match table)을 사용하여 구현될 수도 있다. 본 발명의 하나의 실시형태에 따르면, 호출 그래프의 선형 함수는, 입력 및 출력 내부 변수의 마스크를 고려하는 등가 함수에 의해 대체될 수도 있다.

[0035] 본 발명의 하나의 실시형태에 따르면, 보안 호출 그래프의 마스크 값 중 일부 또는 전체는 랜덤하게 결정될 수도 있다.

[0036] 본 발명의 하나의 실시형태에 따르면, 컴퓨터 구현 방법은 보호된 실행 가능 코드를 생성하도록 상기 호출 그래프를 컴파일하는 추가 단계를 더 포함할 수도 있다.

[0037] 본 발명은 또한, 컴퓨터 시스템으로 하여금 본 발명의 임의의 실시형태에 따른 컴퓨터 구현 방법을 수행하게 하는 컴퓨터 실행가능 명령어를 포함하는 비휘발성 컴퓨터 판독가능 데이터 저장 매체 상에 저장되는 컴퓨터 프로그램 제품, 및 컴퓨터 시스템으로 하여금 상기 컴퓨터 구현 방법을 수행하게 하는 컴퓨터 실행가능 명령어를 포함하는 비휘발성 컴퓨터 판독가능 데이터 저장 매체에 관한 것이다.

[0038] 본 발명은 또한, 메모리에 커플링되는 프로세서를 포함하는 시스템에 관한 것으로, 메모리는 시스템으로 하여금, 알고리즘의 호출 그래프 표현을 상기 알고리즘의 보안 호출 그래프 표현으로 변환하기 위한 컴퓨터 구현 방법을 수행하게 하는 컴퓨터 실행 가능 명령어를 저장한다. 호출 그래프는, 하나 이상의 입력, 하나 이상의 에지(c, d, e), 하나 이상의 노드(P, Q, R, S) 및 하나 이상의 출력(g)을 포함한다. 호출 그래프의 에지는 상기 알고리즘의 내부 변수를 나타내고, 호출 그래프의 노드는 알고리즘의 선형 또는 비선형 기본 함수를 나타낸다. 시스템은 프로세싱 디바이스를 포함하는데, 그 프로세싱 디바이스는:

[0039] • 호출 그래프의 각각의 입력을 마스킹하도록,

[0040] • 호출 그래프의 각각의 마스킹되지 않은 내부 변수를 마스킹된 변수로 대체하도록,

[0041] • 호출 그래프의 적어도 각각의 비선형 함수를, 마스킹된 변수에 적용되는 등가 함수에 의해 대체하도록, 그리고

[0042] • 호출 그래프의 각각의 출력을 마스킹 해제하도록

[0043] 구성된다.

도면의 간단한 설명

[0044] 본 발명은 더 잘 이해될 것이며, 그것의 다양한 피쳐 및 이점은, 예시적인 목적만을 위해 제공되는 다수의 예시적인 실시형태의 다음의 설명 및 그것의 첨부된 도면으로부터 드러날 것인데, 첨부된 도면에서:

- 도 1a는 종래 기술로부터 알려져 있는 바와 같은 보호되지 않은 호출 그래프를 나타낸 도면,

- 도 1b 및 도 1c는, 본 발명의 두 실시형태에 따라 보호되어, 유익하게도, 그래프의 반복 부분의 입력 및 출력으로서 사용되는 변수에 적용되는 마스크의 일관성을 특히 보장하게 되는 호출 그래프를 나타낸 도면,

- 도 2a 및 도 2b는 선형 함수를 프로세싱하는 두 가지 방식을 예시한 도면,

- 도 3은, 마스킹된 입력 및 출력을 고려하여, 상기 비선형 함수와 등가의 매치 테이블을 구성하는 것에 의해, 비선형 함수를 프로세싱하는 하나의 방식을 예시한 도면,

- 도 4a는, 반복 부분이 예시의 목적을 위해 전개된 보호되지 않은 호출 그래프를 나타낸 도면,

- 도 4b 및 도 4c는 본 발명의 두 실시형태에 따라 보호되는 호출 그래프를 나타낸 도면으로, 호출 그래프의 반복 부분에서 사용되는 내부 변수의 마스크는 각각의 반복에서 수정되고, 반복 부분은 예시의 목적을 위해 전개되어 있음,

- 도 5a는 AES 알고리즘의 전형적인 호출 그래프 표현이고, 도 5b는 본 발명의 하나의 실시형태에 따라 생성되는 대응하는 보호된 호출 그래프,

- 도 6은 소정의 실시형태에 따른 컴퓨터 구현 방법을 묘사하는 플로우차트, 및

- 도 7은 본 발명을 프로세싱하기 위해 사용될 수 있는 시스템을 예시한 도면.

본 명세서에서 개시되는 예는 단지 본 발명의 몇몇 실시형태의 예시에 불과하다. 그들은 첨부된 청구범위에 의해 정의되는 본 발명의 범위를 어떤 식으로든 제한하지는 않는다.

발명을 실시하기 위한 구체적인 내용

[0045] 도 1a는 종래 기술로부터 알려져 있는 바와 같은 보호되지 않은 호출 그래프를 나타낸다. 이 표현은 예시의 목적을 위한 표준 표현이다.

[0046] 도 1a의 보호되지 않은 호출 그래프는 컴퓨터 코드의 실행을 설명한다. 그것은 복수의 노드(P, Q, R, S)를 포함하는데, 각각의 노드는 코드에 의해 수행되는 함수에 관련된다. 앞서 나타낸 바와 같이, 노드로서 표현되는 각각의 함수는 단일의 연산, 또는 연산의 조합으로 이루어질 수 있다. 함수는 선형일 수 있거나 또는 선형이 아닐 수 있다.

[0047] 보호되지 않은 호출 그래프는 또한, 노드의 출력을, 그래프의 에지로 칭해지는 다른 노드의 입력에 연결하는 지향성 링크(oriented link)를 포함한다. 이들 에지는, 하나의 함수로부터 후속하는 함수로 전송되는 중간 변수(c, d, e)와 관련된다.

[0048] 보호되지 않은 호출 그래프는 하나 이상의 입력(a, b, f), 및 하나 이상의 출력(g)을 더 포함한다.

[0049] 보호되지 않은 호출 그래프는 또한, 함수(P, Q, R 및 S)가 복수 회 프로세싱되는 반복 부분(101)을 포함하는데, 후속하는 반복은 입력으로서 이전 반복의 출력을 취한다.

[0050] 호출 그래프는 프로그램의 실행 동안의 함수의 상호 의존 관계, 및 입력 변수(a, b, f)로부터 출력 변수(g)를 생성하기 위해 필요로 되는 상호 작용의 표현이다. 그것은 다양한 프로그래밍 언어, 예를 들면, Graphic, UML(Unified Modeling Language; 통합 모델링 언어) 또는 HTML(HyperText Markup Language; 하이퍼텍스트 마크업 언어)를 사용하여 설명될 수 있다. 그것은, Doxygen© 또는 Eclipse©와 같은 소프트웨어를 사용하여 소스 코드로부터, 또는 컴파일된 코드(어셈블러 언어, LLVM-IR(Low Level Virtual Machine - Intermediate Representation; 로 레벨 가상 머신 - 중간 표현), VHDL(VHSIC Hardware Description Language; VHSIC 하드웨어 기술 언어), Verilog©로부터 자동적으로 생성될 수 있거나, 또는 심지어 수동으로 생성될 수 있다.

[0051] 도 1b는 본 발명의 제1 실시형태에 따라 보호되는 호출 그래프를 나타낸다.

[0052] 암호 해독법 및 콘텐츠 복구 공격에 대한 강건성을 제공하기 위해, 본 발명은, 프로그램의 각각의 변수를 마스크로 마스킹하는 것, 및, 본 발명의 몇몇 유익한 실시형태에 따라, 프로그램 실행 동안 이 마스크를 변경하는 것에 기초한다.

[0053] 이를 위해, 입력 변수(a, b, f)는 마스킹된다(111, 112, 113). 마스크 값(m_1, m_2, m_6)은 임의로 선택될 수 있다. 마스킹된 변수는 이후 $a \Theta m_1$, $b \Theta m_2$ 및 $f \Theta m_6$으로 지정된다.

[0054] 그 다음, 호출 그래프의 각각의 마스킹되지 않은 내부 변수(c, d, e)는 마스킹된 변수($c \Theta m_3$, $d \Theta m_4$, $e \Theta m_5$)에 의해 대체된다. 마스크는 랜덤하게 선택될 수도 있거나, 또는 내부 변수가 선형 함수의 출력인 경우, 함수의 입력에 대해 사용되는 마스크로부터 상응하게 상속될 수도 있다.

[0055] 이러한 실시형태에서, 보호된 호출 그래프의 일관성을 보장하기 위해, 내부 변수가 (도 1의 g에 대해) 그래프의 반복 부분의 입력 및 출력 둘 다로서 사용되는 경우, 마스크 변수는 반복 부분의 입력 및 출력 둘 다에서 동일한 마스크를 사용할 수도 있다. 따라서, 변수(b 및 g)와 관련되는 마스크는 동일하다(m_2).

[0056] 따라서, 함수(P')의 입력은 각각의 반복에서 동등하게 마스킹된다.

[0057] 각각의 입력에 마스크를 할당하고 각각의 마스킹되지 않은 각각의 내부 변수를 마스킹된 변수로 대체한 이후, 호출 그래프의 노드에 관련되는 함수(P, Q, R, S)는, 마스크 값을 준수하기 위해, 수정된다(P', Q', R', S').

[0058] 함수가 선형이고 마스킹이 불린인 경우, 출력 마스크는 입력 마스크로부터 상속될 수도 있다. 따라서, 함수는

수정될 필요가 없다. 그렇지 않으면, 함수는, 입력으로서 마스킹된 변수를 취하는 동안, 동일한 결과에 도달하는 등가 함수에 의해 대체되어야 하고, 함수의 출력을 마스킹한다.

[0059] 함수가 비선형인 경우, 출력 마스크와 입력(들) 마스크(들) 사이에 링크를 만드는 것은 일반적으로 가능하지 않다. 보호되지 않은 정보가 나타날 것이고, 정보의 유출로 간주될 수 있기 때문에, 입력 데이터를 마스킹 해제하는 것, 함수를 프로세싱하는 것, 및 결과를 마스킹하는 것은 고려될 수 없다. 따라서, 함수는, 함수의 모든 가능한 결과를 제공하는 매치 테이블에 의해 대체될 수도 있는데, 매치 테이블은 입력 및 출력 마스크를 고려하여 구성된다. 이러한 방식에서는, 선행 기술과는 대조적으로, 알고리즘에 의해 프로세싱되는 모든 변수는 보호되고, 비선형 함수(들)의 근사는 이루어지지 않을 것이다.

[0060] 마지막으로, 호출 그래프를 출력하는 변수(g)는 마스킹 해제될 수도 있다(114).

[0061] 제1 실시형태에서, 모든 내부 변수는 상응하게 보호되고, 도 1b의 보호된 호출 그래프의 결과는, 도 1a의 보호되지 않은 호출 그래프의 결과와 정확히 동일하다. 입력을 마스킹하고 프로그램의 출력을 마스킹 해제하는 것을 제외하면, 이 제1 실시형태는 프로그램의 노드 수에 영향을 끼치지 않는다. 그것의 최종적인 복잡성 및 프로세싱 시간은 추가 보호 레이어에 의해 영향을 받지 않을 것이다. 일단 컴파일되면, 내부 변수 중 어느 것도 보호되지 않은 상태로는 나타나지 않을 것이다. 따라서, 프로그램의 실행은 암호 해독법 및 콘텐츠 복구 공격에 대해 완벽하게 보호된다.

[0062] 도 1c는 본 발명의 제2 실시형태를 나타낸다. 이러한 실시형태에서는, 도 1b와 마찬가지로, 입력(a, b 및 f)은 마스킹되고, 보호되지 않은 내부 변수(c, d, e)는 보호된 변수에 의해 대체되고, 함수는 상응하게 수정되고, 호출 그래프의 출력(g)은 마스킹 해제된다.

[0063] 그러나, 그래프의 반복 부분에 관한 마스킹의 일관성은, 반복 부분의 피드백 루프(101)에, 반복 부분의 입력 및 출력 둘 다로서 사용되는 내부 변수의 마스크 값을 수정하도록 구성되는 추가 노드(120)를 삽입하는 것에 의해 보장된다.

[0064] 도 1c에서, 반복 부분의 출력은 마스킹된 변수($g \Theta m_7$)에 의해 대체되고, 노드(120)는 $g \Theta m_7$을 $g \Theta m_2$로 변환하도록 이 변수의 마스크를 수정한다. 정보가 보호되지 않은 상태로 나타나지 않는 것을 보장하기 위해, 출력 마스크(m_7)가 제거되기 이전에 입력 마스크(m_2)가 적용된다. 도 1b에서와 같이, 함수(P')의 입력은 각각의 반복에서 동일한 마스크를 갖는다.

[0065] 제1 실시형태와 비교한 제2 실시형태의 이점은, 모든 마스크가 랜덤하게 선택될 수도 있다는 것이다.

[0066] 데이터를 마스킹하기 위해 다양한 방법이 사용될 수도 있다. 마스킹은, 예를 들면, 변수를 비밀 공유 값과 합산하는 것과 같은 간단한 1차 불린 마스킹, 고차 불린 마스킹, 또는 임의의 다른 더욱 정교한 마스킹 기술일 수 있다. 본 발명의 이점 중 하나는 그것이 어떤 마스킹 기술과도 호환 가능하다는 것이다.

[0067] 그림 2a는 선형 함수를 프로세싱하는 방식을 예시한다. 이러한 함수에서, 마스킹이 불린인 경우, 출력 변수(들)(203)에 적용되는 마스크는 입력 변수(들)(201 및 202)에 적용되는 마스크에 의존한다. 따라서, 출력 마스크는 선형 함수를 입력 마스크에 적용하는 것에 의해 주어진다. 이 예에서, 입력 변수(a 및 b)는 값(m_1 및 m_2)에 의해 각각 마스킹된다. 함수의 출력은, 마스크($f(m_1, m_2)$)에 적용되는 함수의 결과인 값에 의해 마스킹된다. 단순화된 예를 고려하면, 함수(211)가 곱셈 함수인 경우, $f(m_1, m_2)$는 $m_1 * m_2$와 동일하다.

[0068] 도 2b는, 함수의 결과로서 나타나는 보호된 값이 특정 마스크 값을 가질 때 선형 함수를 프로세싱하기 위한 실시형태를 예시한다.

[0069] 이러한 실시형태에서, 두 가지 가능성이 구현될 수도 있다:

[0070] • 호출 그래프에 추가 노드(212)를 삽입함, 추가 노드는 선형 함수의 결과에 영향을 끼치는 마스크$f(m_1, m_2)$를 필요로 되는 마스크(m_3)로 변환하는 것에 대응함. 이를 위해, 함수(212)는 마스크($f(m_1, m_2)$)를 제거하기 이전에 선형 함수의 결과에 마스크(m_3)를 추가할 수도 있음, 또는

[0071] • 마스크의 값을 고려하는 선형 함수에 등가인 매치 테이블(221)을 계산함, 매치 테이블은 메모리에 저장됨. 이 테이블은 $a \Theta m_1$ 및 $b \Theta m_2$의 모든 가능한 값을 입력으로서 포함하고, 이들 값의 각각을 출력($c \Theta m_3$)의 대응하

는 값과 관련시킨다. 매치 테이블은 모든 가능한 입력에 대해 함수(211 및 212)를 실행하는 것에 의해 계산될 수도 있다. 이 테이블은 암호화되어 저장될 수 있지만, 그 함수에 의해 프로세싱되는 원래의 보호되지 않은 데이터에 관한 임의의 정보를 결정하는 것을, 테이블에 포함되는 데이터가 허용하지 않기 때문에, 그것은 필요하지 않을 수도 있다.

[0072] 도 2b에 나타내어지는 바와 같은 선형 함수의 프로세싱은, 호출 그래프의 각각의 내부 변수에 대해 랜덤 마스크를 사용하는 것을 허용한다.

[0073] 도 3은 비선형 함수를 프로세싱하기 위한 실시형태를 예시한다. 이러한 실시형태에서, 비선형 함수의 프로세싱은, 마스킹된 입력 및 출력을 고려하여, 상기 함수와 등가인 매치 테이블을 구성하는 것에 기초한다. 이러한 프로세싱은 선형 함수를 프로세싱하도록 또한 적용될 수 있다는 것을 유의해야 한다.

[0074] 도 2a 및 2b에서 나타내어지는 선형 함수와는 대조적으로, 출력 값의 마스크는 입력 값의 마스크로부터 결정될 수 없다. 이러한 이유 때문에, 비선형 함수는, 입력의 모든 가능한 세트에 대한 함수의 결과를 제공하는 등가의 매치 테이블(310)에 의해 대체되어야만 한다. 매치 테이블은, 입력 변수($a \Theta m_1$ 및 $b \Theta m_2$)를 마스킹 해제하는 것(301 및 302), 비선형 함수를 적용하는 것(303), 및 비선형 함수의 결과를 마스킹하는 것(304)에 의해 계산될 수도 있다.

[0075] 테이블을 구성하기 위해, 모든 가능한 입력 변수가 브라우징될 수도 있다. 따라서, 테이블은 입력의 수, 출력의 수 및/또는 데이터 사이즈에 비례할 수도 있다. 예를 들면, 입력(a와 b)이 8 비트에 걸쳐 코딩된다는 것을 고려하면, 관련된 매치 테이블은, 2^8(a에 대한 가능성의 수) * 2^8(b에 대한 가능성의 수) * 8 비트(c의 사이즈)의 크기에 이르는 테이블이다.

[0076] 도 4a는, 알고리즘의 반복 부분이 전개된 실시형태에서의 도 1a의 보호되지 않은 호출 그래프를 나타낸다. 도 4a의 다음의 설명에서, 반복 부분은, 단지 예시적인 목적만을 위해, 단지 두 번만 실행되는 것이 고려된다.

[0077] 함수(P, Q, R 및 S)는 두 번 실행되는데, 두 번째 반복 동안의 내부 변수의 값(c', d', e' 및 g')은 첫 번째 반복 동안의 동일한 변수의 값(c, d, e 및 g)과는 상이하다.

[0078] 도 4b 및 도 4c는 본 발명의 다른 실시형태에 따라 보호되는 호출 그래프를 나타내는데, 호출 그래프의 반복 부분에서 사용되는 내부 변수의 마스크는 각각의 반복에서 수정된다. 도 4b 및 도 4c의 다음의 설명에서, 반복 부분은 도 4a와 마찬가지로 전개되어 있는 것이 고려된다.

[0079] 각각의 반복의 끝에서 마스크 값을 리프레쉬하는 것에 의해(즉, 마스크 값을 변경하는 것에 의해), 특히, 적용되는 마스크의 일정한 양태를 사용하는 부채널 공격에 대해서, 높은 레벨의 보호가 획득된다.

[0080] 도 4b에서, 첫 번째 반복은, 각각의 입력(a, b 및 f)을 마스킹하는 것, 각각의 내부 변수(c, d, e 및 g)를 마스킹된 변수($c \Theta m_3$, $d \Theta m_4$, $e \Theta m_5$ 및 $g \Theta m_7$)에 의해 대체하는 것, 및 마스킹된 변수를 고려하도록 함수를 수정하는 것(P', Q', R' 및 S')에 의해, 도 1b 또는 도 1c에서와 같이 수행된다.

[0081] 후속하는 반복에 대해서, 내부 변수(c, d, e 및 g)에 대해 사용되는 마스크는 수정되고, 관련된 함수는 상응하게 수정된다. 반복의 입력으로서 사용되는 변수에 적용되는 마스크도 또한 수정될 수도 있다. 그 예에서, 입력(a 및 f)에 마스크(m_1 및 m_6)를 적용하는 노드(111 및 113)는 새로운 마스크(m_8 및 m_{12})를 적용하는 노드(401 및 403)로 변경된다. 마스킹된 변수($c \Theta m_3$, $d \Theta m_4$ 및 $e \Theta m_5$)는 새로운 마스킹된 변수($c \Theta m_9$, $d \Theta m_{10}$ 및 $e \Theta m_{11}$)로 변경된다. 마스크(m_1 및 m_2)에 의해 마스킹되는 변수를 입력으로서 취하는 함수(P')는 등가 함수(P")로 수정되는데, 함수(P")는 m_8 및 m_7에 의해 마스킹되는 변수를 입력으로 취한다. 함수(P')가 선형 함수이고 함수의 출력이 입력 마스크로부터 상속되는 마스크에 의해 마스킹되는 경우, P'는 수정되지 않은 채로 유지될 수도 있다. 함수(Q', R' 및 S')는 함수(Q", R" 및 S")로 상응하게 수정된다.

[0082] 대안적인 실시형태(표현되지 않음)에서, 입력(a 및 f)의 마스크는 반복마다 수정되지 않을 수도 있다.

[0083] 호출 그래프는 예를 들면, 도 1c에서, 반복에서 수반되는 모든 변수의 마스크를 리프레쉬하는 단계에 대응하는 추가 노드, 예컨대 반복 부분의 피드백 루프에서의 노드(120)를 사용하여 표현될 수 있다.

[0084] 도 4c는, 노드(402)가 반복의 피드백 루프에 삽입되는 다른 실시형태를 나타낸다. 이 노드(402)는 반복에서 수반되는 입력 및 변수에 적용되는 모든 마스크를 리프레쉬하는 단계에 관련된다. 또한, 도 4c에 나타내어지는 바

와 같이, 이 노드는 이전 반복의 출력에 영향을 끼치는 마스크를 추가로 변경하고, 그에 의해, 마스크(m_7)를 마스크(m_9)로 변환할 수 있다.

[0085] 도 5a는 암호화되지 않은 AES 알고리즘의 호출 그래프 표현이다. 이러한 예시적인 표현은 본 발명에 따른 방법을 예시하기 위해 사용된다. 이러한 표현은 바이트 0(x_0), 및 반복적으로 프로세싱되는, "라운드 스텝"으로 칭해지는 AES 알고리즘의 단계에만 집중된다. AES 알고리즘의 표준 실행에서, 첫 번째 라운드(사전 프로세싱) 및 마지막 라운드(사후 프로세싱)는 특별하다. 이들 라운드는 여기서는 표현되지 않으며, 도 5a는 데이터 경로의 중간 라운드에만 집중한다(AES 키(k_i)에 대해 수행되는 프로세싱은 나타내지 않음).

[0086] 각각의 라운드는 반복 프로세스인데, 라운드의 한 번의 반복의 출력인 y_0 바이트가, 라운드의 입력인 x_0에 대해 루프백된다는 것을 의미한다.

[0087] 호출 그래프 표현에서, 원은 상이한 변수에 대해 수행되는 연산을 나타내지만, 그래프의 에지는 내부 변수를 나타낸다. 도 5에서는, 입력 또는 출력이 없는데, 그들이 라운드의 첫 번째 반복과 마지막 반복(표현되지 않음)에서 프로세싱되기 때문이다.

[0088] 라운드에서 x_0에 적용되는 첫 번째 함수는 치환 박스(S 박스(S-box)로 알려짐)로 칭해진다. 이 치환은 알고리즘의 주요 요소이며 x_0에 대해 수행되는 전단사의(bijective) 비선형 연산이다. 첫 번째 치환 박스의 출력은 도 5a의 중간 변수(a)이다.

[0089] a에 적용되는 다음 함수는 갈루아 체(Galois field)에서 수행되는 세 개의 선형 연산: 1 배(times 1), 2 배(times 2) 및 3 배(times 3)의 세트이다. 이러한 연산은 선형적이다.

[0090] b, b' 및 b"로 칭해지는 이들 연산의 결과는, 바이트 5, 10 및 15(x_5, x_A, x_F)에 대해 수행되는 대응하는 연산의 결과와 혼합된다. 이러한 연산은 "믹스칼럼(Mixcolumn)"으로 칭해진다. 혼합은 네 개의 엔트리에 대해 수행되는 XOR 연산이다. 이 함수는 세 개의 연속하는 XOR 연산과 등가이다. "믹스칼럼" 연산의 출력은 중간 변수(c)이다.

[0091] c에 적용되는 다음 함수는 "애드라운드키(Addroundkey)"로 칭해지는 단계인데, XOR 연산을 통해 c를 키(또는 키로부터 프로세싱되는 특정한 바이트)(k_0)와 혼합하여, 후속하는 반복의 입력으로서 사용될 y_0를 생성한다.

[0092] 도 5b는 본 발명의 하나의 실시형태에 따라 보호된 이후의 도 5a의 AES 알고리즘의 호출 그래프 표현이다.

[0093] x_0가 중간 변수이므로, 그것은 마스크(Θm_{00})에 의해 마스킹된다. 이 마스크는 x_0과 알려진 랜덤 값 사이에서 수행되는 XOR일 수 있지만, 그러나 그것은 또한, 복수의 마스크 레이어가 적용된다는 것을 의미하는 다차원 공유일 수 있다. 후자의 경우, Θm_{00}는 반드시 바이트(8 비트) 공유일 필요는 없지만, 그러나 임의의 차원의 공유일 수 있다. 그것은 또한, 쌍 또는 삼중 항 또는 동일한 또는 상이한 사이즈를 갖는 마스크의 임의의 다른 관련화일 수 있다.

[0094] S 박스 단계에서 프로세싱된 이후, 중간 변수(a)는 마스크(Θm_{01})에 의해 마스킹된다. 치환 박스가 비선형 함수이기 때문에, 도 3에서 예시되는 바와 같이, 그것은 마스크(Θm_{00})를 모두 한 번에 제거하는 등가의 매치 테이블에 의해 대체되어야만 하고, 비선형 함수를 수행해야만 하고, 결과를 마스크(Θm_{01})를 사용하여 마스킹해야만 한다. 따라서, Θm_{01}는 반드시 Θm_{00}에 관련되는 것은 아니다.

[0095] 1 배, 2 배, 및 3 배 연산을 수행한 이후, 중간 변수(b)는 마스크(Θm_{02})에 의해 마스킹될 수도 있다. 몇몇 실시형태에서, 마스크는 Θm_{01}와 동일할 수도 있거나, 또는 함수는 등가의 매치 박스에 의해 대체될 수도 있으며, 따라서 Θm_{02}는 Θm_{01}과 완전히 독립적으로 선택될 수 있다.

[0096] 다른 실시형태에서, 치환 박스의 연산 및 1 배, 2 배, 및 3 배의 연산을 수행하는 등가의 매치 테이블(501)이 계산될 수도 있다. 매치 테이블은 하나의 입력($x_0 qm_{00}$) 및 세 개의 출력($b\Theta m_{02}$, $b'\Theta m_{02}$, $b"\Theta m_{02}$)을 가질 수도 있다. 대안적으로, 세 개의 매치 테이블이 계산될 수도 있는데, 이들의 각각은 하나의 입력 및 하나의 출력을 갖는다. 매치 테이블에서 1 배, 2 배 및 3 배 연산자로 재그룹화되는 치환 박스는 암호화되지 않은 경우 "T 박스(T-box)"(Table box; 테이블 박스)로 칭해지는 연산을 나타낸다.

[0097] 다른 실시형태에서, 상이한 마스크가 중간 변수(b, b' 및 b")의 각각에 할당될 수도 있다.

[0098] 다양한 바이트에 대해 수행되는 계산의 결과를 혼합하는 단계는 선형 함수이다. 결과적으로, 출력 마스크(Θm_{03})는 입력의 마스크(Θm_{01}, Θm_{51}, Θm_{A1}, 및 Θm_{F1})(Θm_{51}, Θm_{A1}, 및 Θm_{F1}는 변수(x_5, x_A 및 x_F)를 프로세싱하기 위한 T 박스 계산의 출력과 각각 관련되는 마스크임)로부터 검색될 수 있다. 그러나, 입력 마스크와는 완전히 독립적인 출력 마스크(Θm_{03})를 선택하는 것을 허용하는 등가의 매치 테이블이 계산될 수도 있다.

[0099] 다음 단계에서, 중간 변수($c\Theta m_{03}$)가 키(k_0)와 혼합된다. 키가 변수가 아니고 상수이기 때문에, 키는 마스킹될 필요가 없다. 혼합의 결과는 $y_0\Theta m_{04}$이다. 혼합 연산이 선형적이기 때문에, Θm_{04}는 Θm_{03}와 관련되거나, 또는 혼합 함수가 등가의 매치 테이블에 의해 대체되면 완전히 독립적일 수 있다.

[0100] 마지막으로, 리프레쉬 노드(502)가 삽입될 수도 있다. 리프레쉬 노드의 첫 번째 목적은, 변수(y_0/x_0)가 호출 그래프의 반복 부분의 입력/출력으로 사용될 때, Θm_{04}를 Θm_{00}로 변환하는 것에 의해 보호된 호출 그래프의 일관성을 보장하는 것이다. 몇몇 실시형태에서, 리프레쉬 마스크는, 반복 루프에 속하는 내부 변수 중 적어도 일부에 대한 마스크를 변경하는 단계와 추가로 관련될 수 있다(그 경우, 마스크(Θm_{00}, Θm_{01}, Θm_{02}, Θm_{03} 및 Θm_{04})임).

[0101] 선형 함수의 노드가 그들의 부모 노드로부터 상속될 때, 루프의 입력인 변수의 마스크(들)(도 5a에서의 Θm_{00}) 및 비선형 함수(들)의 출력(들)인 변수의 마스크(실시형태에 따라, 도 5b에서의 Θm_{01} 또는 Θm_{02})만을 리프레쉬하는 것이 가능하다. 이것은 호출 그래프 일관성을 보장하기 위해 관련된 등가의 테이블을 수정하는 것을 의미한다. 리프레쉬된 마스크는 선형 함수의 입력/출력으로 자동적으로 전파된다.

[0102] 리프레쉬 노드(512)는 옵션적이다는 것을 유의해야 한다. 보호된 호출 그래프의 일관성을 보장하는 다른 방식은, 예를 들면, Θm_{00}와 동일한 Θm_{04}를 선택하는 것일 수도 있다.

[0103] 도 6은 소정의 실시형태에 따른 컴퓨터 구현 방법을 묘사하는 플로우차트이다.

[0104] 그 방법은 다음을 포함한다:

[0105] - 호출 그래프의 입력을 마스킹하여 마스킹된 입력을 생성하는 단계(601);

[0106] - 그래프의 에지에 의해 표현되는 그래프의 보호되지 않은 변수를 마스킹된 변수에 의해 대체하는 단계(602). 마스킹된 변수의 마스크는 랜덤하게 선택될 수 있거나, 또는 마스킹된 변수에 대한 선형 함수의 사용의 결과일 수 있음;

[0107] - 함수의 입력/출력에 영향을 끼치는 마스크를 고려하면서, 초기 함수와 동일한 연산을 수행하도록, 그래프의 노드에 의해 표현되는 호출 그래프의 적어도 비선형 함수를, 등가 함수에 의해 대체하는 단계(603). 이 동작은 호출 그래프의 선형 함수에 대해 또한 수행될 수 있음. 하나의 가능한 구현은, 입력/출력에 영향을 끼치는 마스크를 고려하여 생성되는 매치 테이블에 의해 함수를 대체하고, 출력 값을 입력 값의 각각의 가능한 조합에 관련시키는 것임; 및

[0108] - 호출 그래프의 출력을 마스킹 해제하는 단계(604).

[0109] 본 발명에 따른 방법은, 마스킹된 입력 및 변수가, 규칙적으로 또는 랜덤하게, 루프의 각각의 반복에서 또는 더 느린 레이트로 리프레쉬되도록(마스크의 값이 수정되는 것을 의미함) 그래프의 반복 부분을 수정하는 추가적인 옵션적 단계(605)를 포함할 수도 있다. 따라서, 반복적으로 계산되는 변수는 동일한 마스크로 절대 보호되지 않는다. 비록 이러한 응용에 국한되지는 않지만, 본 실시형태에 따른 방법은, 작은 계산에 걸쳐 수행되는 아주 많은 수의 반복을 종종 포함하는 암호 알고리즘에 적용될 때 특별한 이점을 갖는다.

[0110] 본원에서 설명되는 방법은, 예를 들면, 마이크로프로세서, 마이크로컨트롤러, 또는 DSP와 같은 임의의 타입의 프로세서 또는 임의의 소프트웨어 프로그래머블 머신에 제공되어, 본원에서 명시되는 함수/액트(act)를 구현하기 위한 명령어를 실행하는 머신을 생성하는 컴퓨터 프로그램 명령어에 의해 구현될 수 있다. 이들 컴퓨터 프로그램 명령어는 또한, 특정한 방식으로 기능할 것을 컴퓨터에게 지시할 수 있는 컴퓨터 판독 가능 매체에 저장될 수도 있다. 이를 위해, 컴퓨터 프로그램 명령어는, 실행된 명령어가 본원에서 명시되는 기능을 구현하기 위한 프로세스를 제공하게끔, 일련의 동작 단계의 수행을 야기하도록 그리고 그에 의해 컴퓨터 구현 프로세스를 생성

하도록 컴퓨터 상으로 로딩될 수도 있다.

[0111] 그 방법은, 보호되지 않은 알고리즘으로부터 알고리즘의 보호된 표현을 생성하기 위해, 단독으로 사용될 수 있지만, 그러나 컴파일러와 짝을 이룰 수 있고, 그에 의해, 계산 머신에 의해 실행될 수 있는 보호되는 컴파일된 코드, 또는 하드웨어 코드를, 예를 들면, 필드 프로그래머블 게이트 어레이(FPGA), 또는 주문형 반도체(ASIC)와 같은 전용 계산 머신 상에서 구현되고 컴파일러에 의해 생성되는 넷리스트(netlist)의 형태로, 생성하게 된다.

[0112] 이를 위해, 그 방법은, 암호 해독법 및 콘텐츠 복구 공격에 강건한 실행 가능 코드를 생성하기 위해, 보호된 호출 그래프를 컴파일하는 추가 단계(606)를 포함할 수도 있다.

[0113] 도 7은 본 발명을 프로세싱하기 위해 사용될 수 있는, 예를 들면, 범용 컴퓨터 시스템과 같은 시스템을 예시한다. 그 시스템은, 컴퓨터 프로그램 제품이 저장되는 비휘발성 컴퓨터 판독 가능 메모리(M), 프로세서(CPU)에 의해 프로세싱되는 소스 코드 또는 실행 가능 코드와 같은 보호되지 않은 데이터, 또는 보호되지 않은 호출 그래프를 검색하여 호출 그래프를 생성하기 위한 입력/출력 인터페이스 I/O를 포함하는데, 프로세서는 데이터 버스를 통해 메모리 및 입력/출력 인터페이스에 연결된다.

[0114] 더 일반적으로는, 본원에서 설명되는 방법 및 디바이스는 다양한 수단에 의해 구현될 수도 있다. 예를 들면, 이들 기술은 하드웨어, 소프트웨어, 또는 이들의 조합으로 구현될 수도 있다.

[0115] 본 발명의 다양한 실시형태는 다음을 포함하는 여러 가지 이점을 제공한다:

[0116] - 그들은 구현과 관련이 없는 하이 레벨의 접근법을 제시함,

[0117] - 그들은, 현저하게도, 알고리즘에 의해 프로세싱되는 모든 변수가 마스킹되고 한편 알고리즘의 의미가 보존되기 때문에, 모든 종류의 부채널 공격에 대한 강건성을 알고리즘에게 제공함,

[0118] - 그들은 임의의 타입의 소프트웨어 프로그램에, 선형 함수에 그리고 비선형 함수에 적용 가능함,

[0119] - 그들은 구현 및 컴파일하기가 상당히 쉬움,

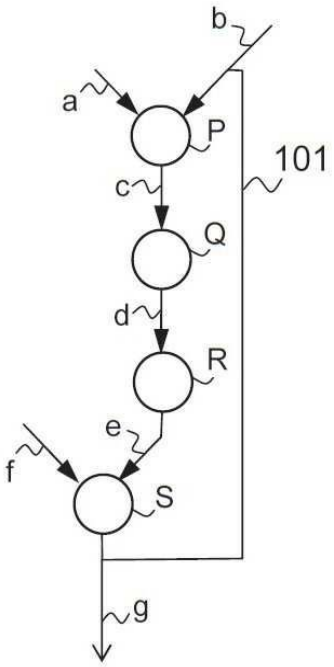
[0120] - 그들은, 모든 변수가 마스킹되기 때문에, 정보 유출을 나타내지 않음,

[0121] - 그들은, 인간 오퍼레이터를 수반하지 않고도, 자동적으로 실행되도록 프로그래밍될 수 있음.

[0122] 본 발명의 실시형태가 다양한 예에 대한 설명에 의해 예시되었지만, 그리고 이들 실시형태가 상당히 상세하게 설명되었지만, 첨부된 청구범위의 범위를 이러한 세부 사항으로 한정하거나 또는 어떤 식으로든 제한하는 것은 본 출원인의 의도가 아니다. 추가적인 이점 및 수정이 기술 분야의 숙련된 자에게는 쉽게 명백해질 것이다. 따라서, 보다 넓은 양태에서의 본 발명은, 도시되고 설명되는 특정 세부 사항, 대표적인 방법, 및 예시적인 실시형태로 제한되지 않는다.

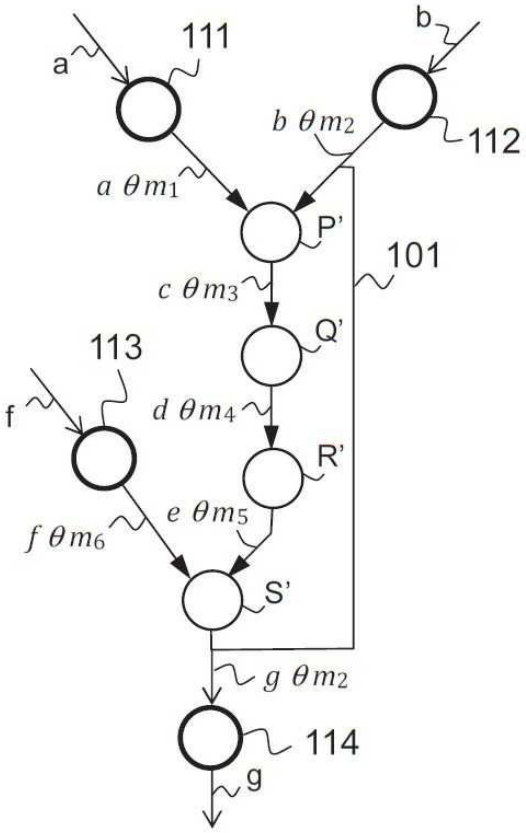
도면

도면1a

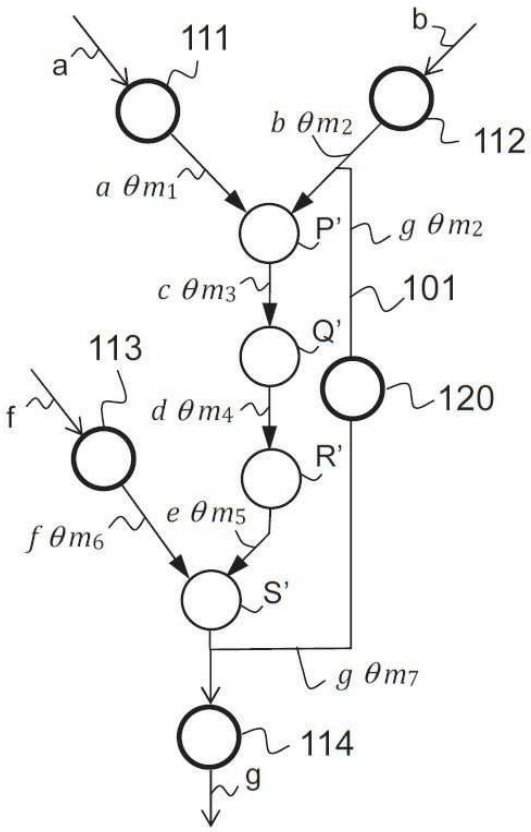


종래 기술

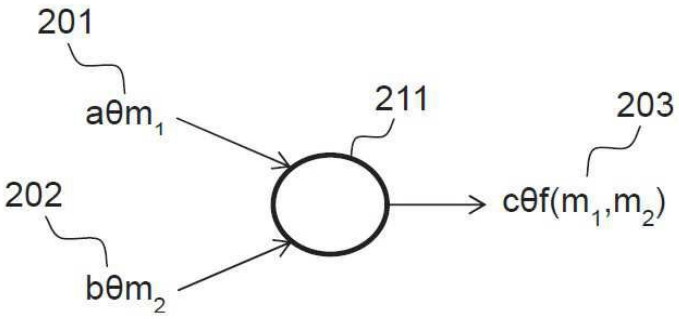
도면1b



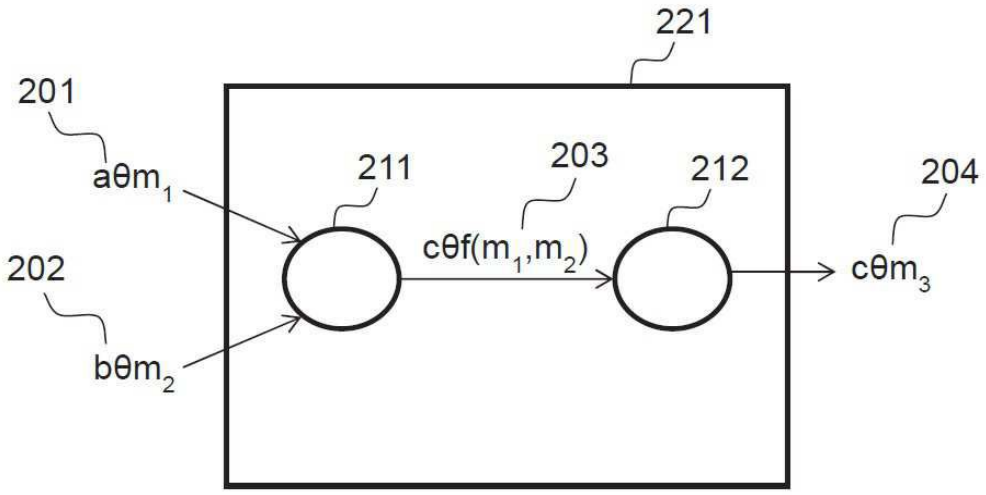
도면1c


a
111
b
b θm2
112
a θm1
P'
g θm2
c θm3
101
Q'
113
120
f
d θm4
R'
e θm5
f θm6
S'
g θm7
114
g

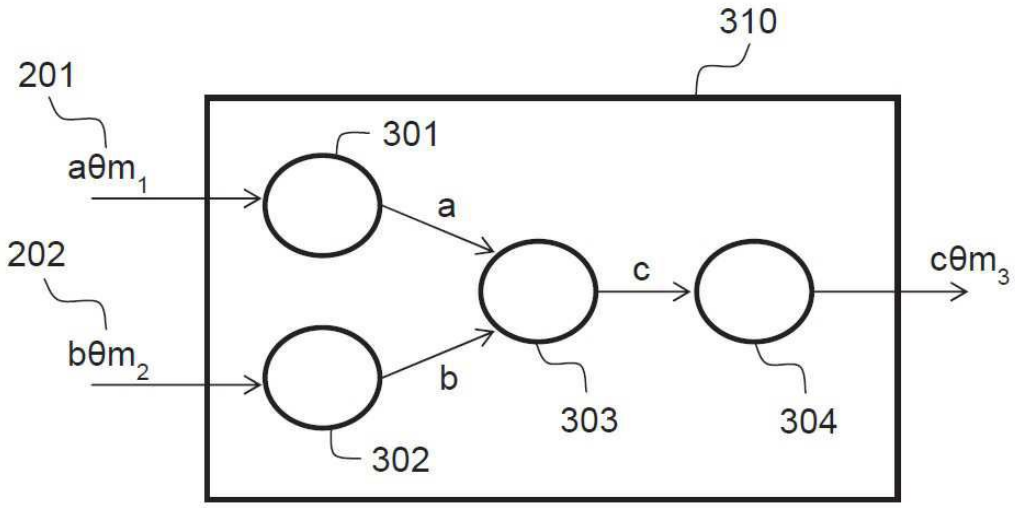
도면2a


201
aθm1
211
203
202
cθf(m1,m2)
bθm2

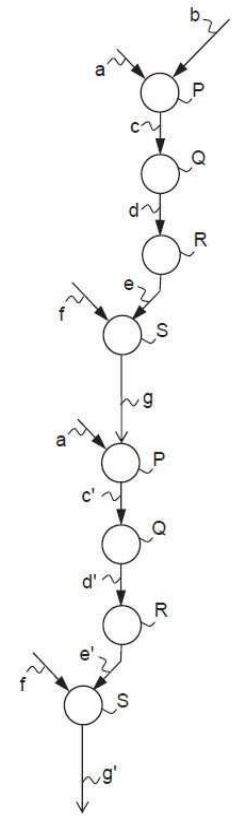
도면2b


221
201
aθm1
211
203
212
204
cθf(m1,m2)
202
cθm3
bθm2

도면3

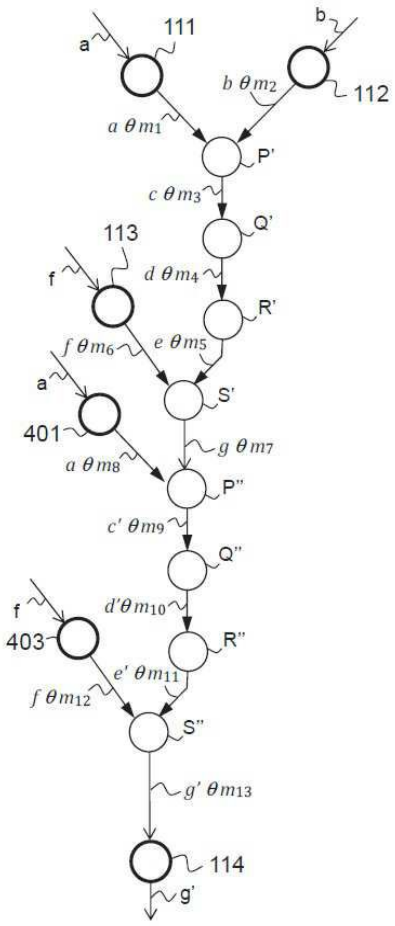

310
201
301
aθm1
a
202
c
cθm3
bθm2
b
303
304
302

도면4a

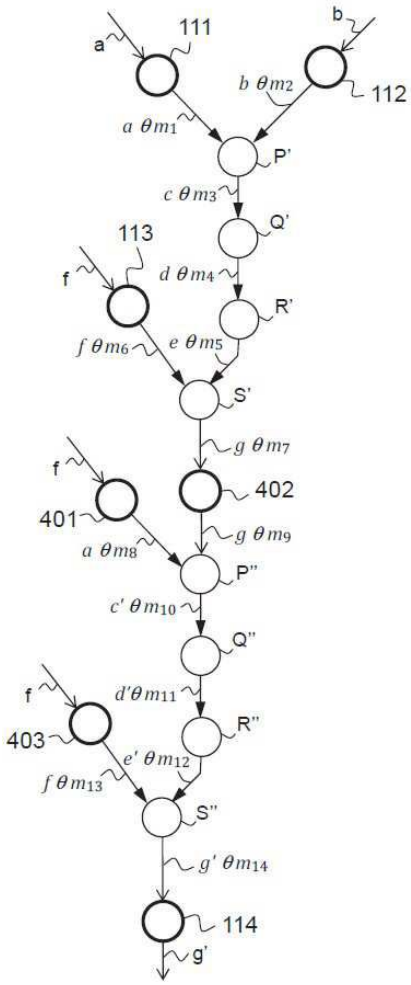


종래 기술

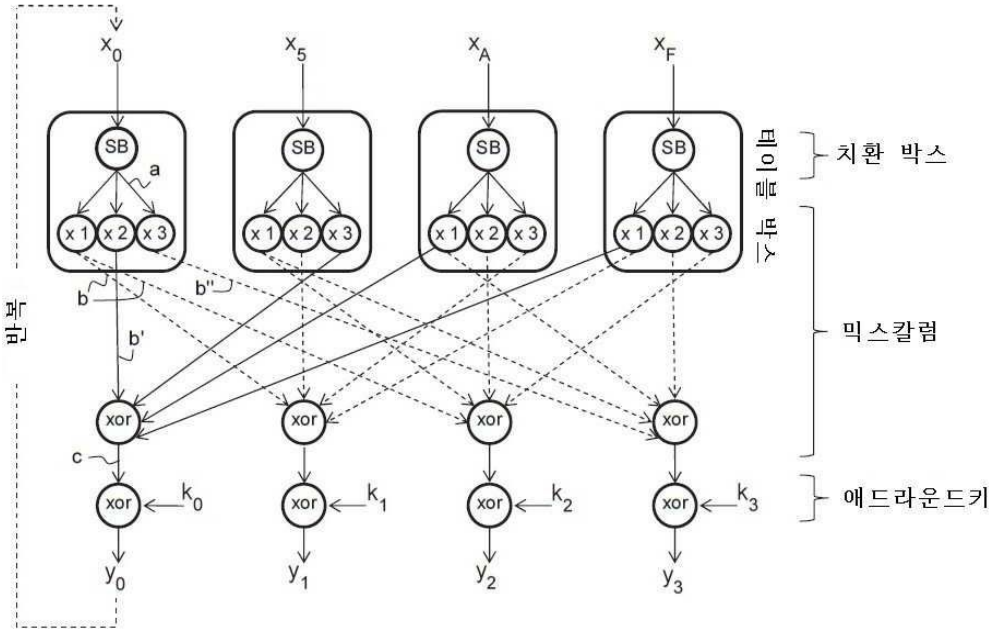
도면4b


a
111
b
112
b θm2
a θm1
P'
c θm3
Q'
113
f
d θm4
R'
f θm6
e θm5
a
S'
401
g θm7
a θm8
P''
c' θm9
Q''
f
d'θm10
403
R''
e' θm11
f θm12
S''
g' θm13
114
g'

도면4c


a
111
b
112
b θm2
a θm1
P'
c θm3
Q'
113
f
d θm4
R'
f θm6
e θm5
S'
g θm7
f
402
401
g θm9
a θm8
P"
c' θm10
Q"
f
d'θm11
R"
403
e' θm12
f θm13
S"
g' θm14
114
g'

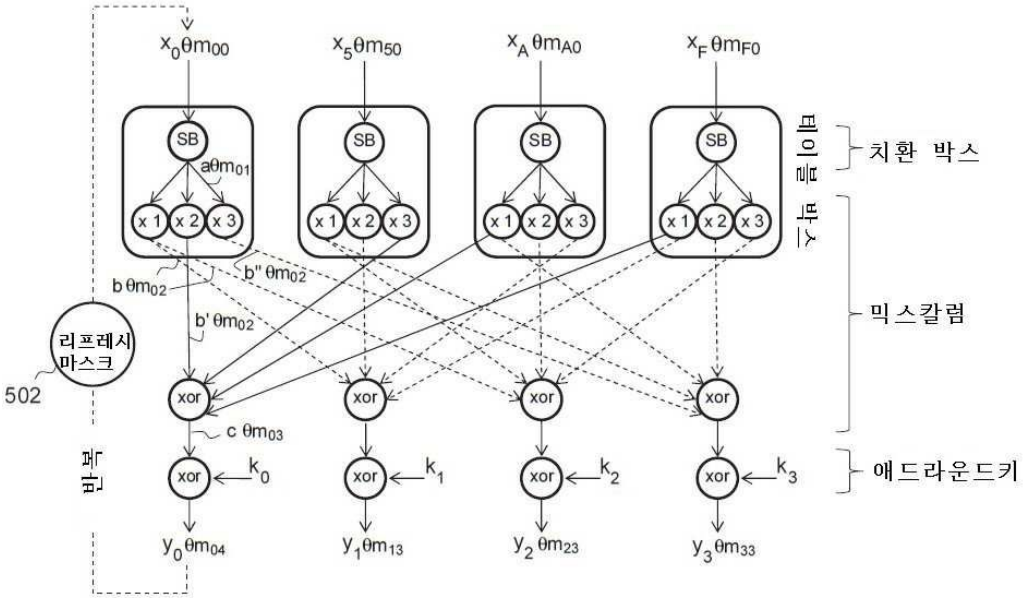
도면5a



AES 라운드의 그래프

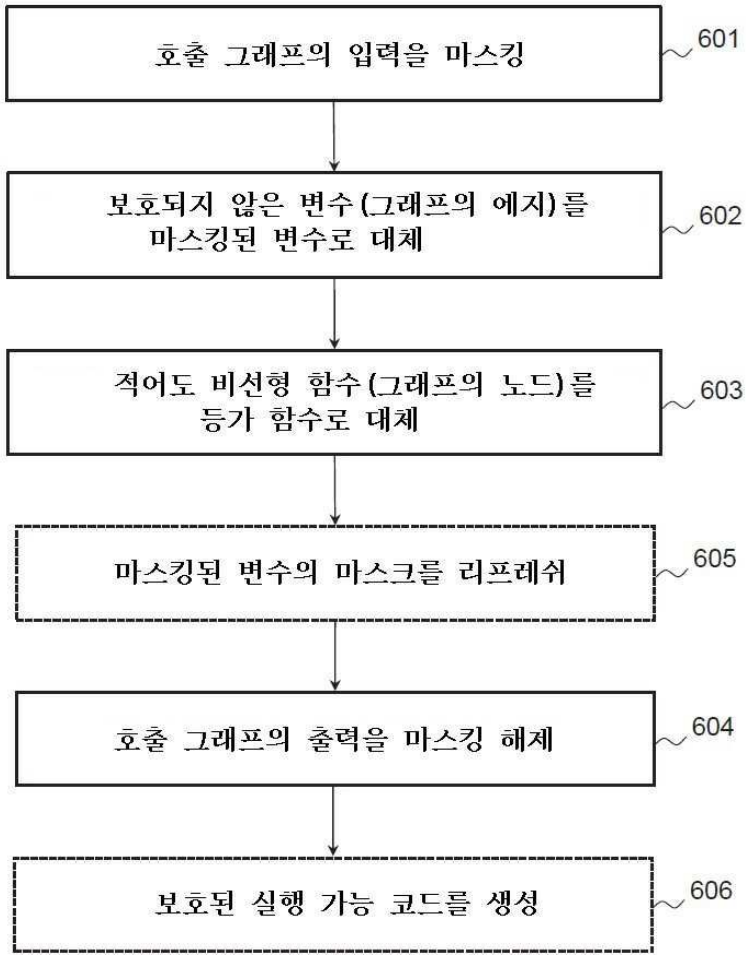
종래 기술

도면5b

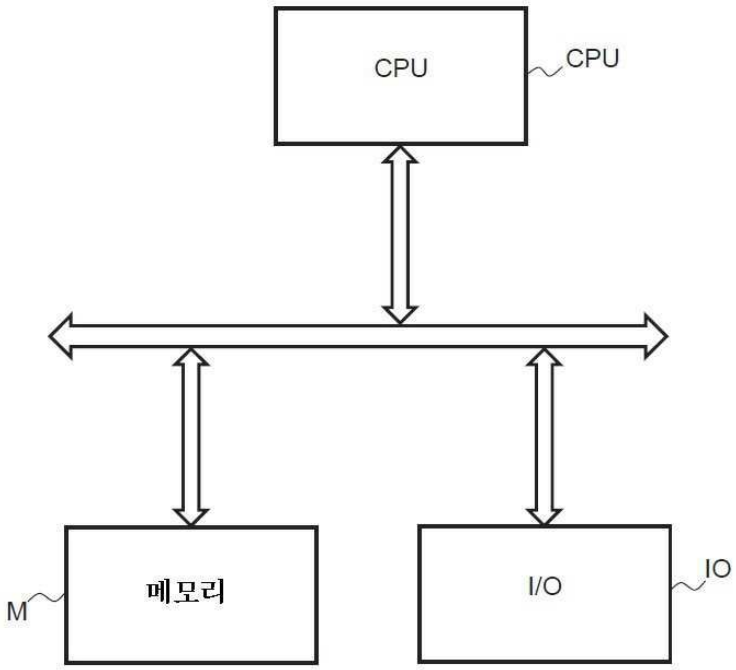


AES 라운드의 변환된 그래프

도면6


호출 그래프의 입력을 마스킹
601
보호되지 않은 변수(그래프의 에지)를 마스킹된 변수로 대체
602
적어도 비선형 함수(그래프의 노드)를 등가 함수로 대체
603
마스킹된 변수의 마스크를 리프레쉬
605
호출 그래프의 출력을 마스킹 해제
604
보호된 실행 가능 코드를 생성
606

도면7


CPU
CPU
메모리
M
I/O
IO