(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0061405
(43) 공개일자 2012년06월13일

(51) 국제특허분류(Int. Cl.)
G06F 21/24 (2006.01) *G06F 21/22* (2006.01)
(21) 출원번호 10-2010-0122719
(22) 출원일자 2010년12월03일
심사청구일자 2010년12월03일

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
김승주
서울특별시 송파구 중대로 24, 올림픽훼밀리APT 234동 403호 (문정동)
원동호
서울특별시 서초구 잠원동 한일17차아파트 334동 903호
(뒷면에 계속)
(74) 대리인
특허법인정직과특허

전체 청구항 수 : 총 16 항

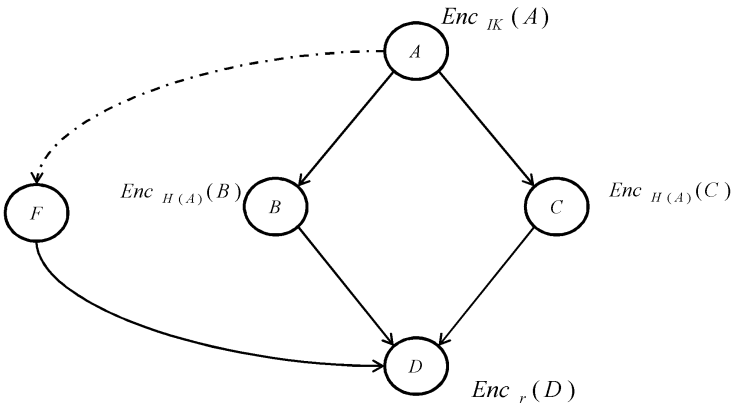
(54) 발명의 명칭 **인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 그 방법**

(57) 요 약

컴퓨터 프로그램의 실행파일을 암호화하거나, 암호화된 실행파일을 복호화하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법에 있어서, 실행파일의 실행코드들을 호출코드에 의해 코드블록으로 구분하고, 각 코드블록이 호출되는 회수 및 시작주소를 인덱스 테이블에 저장하는 인덱스 생성부; 및, 코드블록을 암호키로 암호화하되, 1회 호출되는 코드블록(이하 제1유형 코드블록)의 암호키는 제1유형 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성하고, 2회 이상 호출되는 코드블록(이하 제2유형 코드블록)의 암호키는 난수로 생성하고 실행파일에 저장하는, 블록 암호화부를 포함하는 구성을 마련한다.

상기와 같은 장치 및 방법에 의하여, 인덱스 테이블을 이용하여 코드블록의 호출회수에 따라 암호키를 달리 생성함으로써, 안전한 코드 암호화가 가능할 뿐만 아니라, 암호화 또는 복호화의 수행시간을 단축하고 저장하는 데이터의 양을 줄일 수 있다.

대 표 도 - 도6



(72) 발명자

조성규

경기도 수원시 장안구 율전로101번길 43, 성원빌라 203호 (율전동)

신동휘

서울특별시 송파구 중대로 109 (가락동)

조혜숙

경기도 수원시 장안구 천천동 성균관대학교자연과학캠퍼스 81207호

최동현

경기도 수원시 팔달구 정자천로32번길 20, 꽃뫼버들마을 엘지아파트 165동202호 (화서동)

류재철

대전광역시 유성구 대학로 99, 인터넷침해 대응기술센터 (궁동, 충남대학교)

특허청구의 범위

청구항 1

컴퓨터 프로그램의 실행파일을 암호화하는 인덱스 테이블 기반 코드 암호화 장치에 있어서,

상기 실행파일의 실행코드들을 호출코드에 의해 코드블록으로 구분하고, 각 코드블록이 호출되는 회수 및 시작주소를 인덱스 테이블에 저장하는 인덱스 생성부; 및,

상기 코드블록을 암호키로 암호화하되, 1회 호출되는 코드블록(이하 제1유형 코드블록)의 암호키는 상기 제1유형 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성하고, 2회 이상 호출되는 코드블록(이하 제2유형 코드블록)의 암호키는 난수로 생성하고 상기 실행파일에 저장하는, 블록 암호화부를 포함하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 2

제1항에 있어서,

상기 실행코드는 바이너리 코드 또는 어셈블리 코드이고, 상기 호출코드는 바이너리 코드 또는 어셈블리 코드의 분기 코드 또는 점프 코드인 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 3

제1항에 있어서,

상기 인덱스 생성부는 마지막 실행코드를 호출코드로 포함하는 일련의 실행코드를 하나의 코드블록으로 구분하되, 상기 호출코드는 상기 호출코드가 포함된 코드블록 이외의 다른 코드블록을 호출하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 4

제1항에 있어서,

상기 인덱스 생성부는 코드블록의 호출 회수를 인덱스 테이블에 저장하되, 각 코드블록의 호출코드 각각에 대하여, 상기 호출코드가 호출하는 코드블록의 호출 회수를 하나씩 증가시켜 저장하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 5

제1항에 있어서,

상기 블록 암호화부는 상기 제1유형 코드블록의 호출 블록을 해쉬하여 상기 제1 유형 코드블록의 암호키를 생성하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 6

제1항에 있어서,

상기 블록 암호화부는 최초 코드블록의 암호키를 초기키로 정하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 7

제1항에 있어서,

상기 블록 암호화부는 상기 제2유형 코드블록의 암호키를 초기키로 암호화하여 상기 실행파일의 데이터 영역에 저장하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 8

제1항에 있어서,

상기 블록 암호화부는 상기 인덱스 테이블을 상기 실행파일의 데이터 영역에 저장하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 9

제1항에 있어서,

상기 인덱스 생성부는 상기 코드블록의 크기를 상기 인덱스 테이블에 저장하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 장치.

청구항 10

제1항 내지 제9항 중 어느 한 항의 장치에 의해 암호화된 실행파일을 복호화하는 인덱스 테이블 기반 코드 복호화 장치에 있어서,

상기 실행파일의 암호화된 실행코드를 코드블록 단위로 코드블록의 암호키를 이용하여 복호화하되, 상기 인덱스 테이블을 참조하여, 상기 코드블록이 제1유형 코드블록이면 상기 코드블록의 호출 블록으로 암호키를 생성하여 이용하고, 제2유형 코드블록이면 상기 코드블록의 저장된 암호키를 이용하는 블록 복호화부; 및,

복호화된 코드블록의 마지막 호출코드가 실행되면 상기 코드블록을 다시 암화화하는 블록 재암호부를 포함하는 것을 특징으로 하는 인덱스 테이블 기반 코드 복호화 장치.

청구항 11

제10항에 있어서,

상기 블록 재암호부는 상기 코드블록이 반복실행되는 경우, 반복회수를 상기 인덱스 테이블에 저장하고 상기 코드블록이 호출될 때마다 상기 반복회수를 감소시켜, 반복회수가 0이 되면 상기 코드블록을 재암호화하는 것을 특징으로 하는 인덱스 테이블 기반 코드 복호화 장치.

청구항 12

컴퓨터 프로그램의 실행파일을 암호화하는 인덱스 테이블 기반 코드 암호화 방법에 있어서,

(b) 상기 실행파일의 실행코드들을 호출코드에 의해 코드블록으로 구분하고, 각 코드블록이 호출되는 회수 및 시작주소를 인덱스 테이블에 저장하고, 2회 이상 호출되는 코드블록(이하 제2유형 코드블록)에 대하여 암호키를 난수로 생성하여 저장하는 단계; 및,

(c) 상기 코드블록을 상기 코드블록의 암호키로 암호화하되, 1회 호출되는 코드블록(이하 제1유형 코드블록)의 암호키는 상기 제1유형 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성하는 단계를 포함하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 방법.

청구항 13

제12항에 있어서,

상기 실행코드는 어셈블리 코드이고, 상기 호출코드는 어셈블리 코드의 분기 코드 또는 점프 코드인 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 방법.

청구항 14

제12항에 있어서, 상기 (b)단계는,

(b1) 상기 실행코드들을 순차적으로 검사하는 단계;

(b2) 검사한 실행코드가 호출코드이면 호출코드의 피연산자로부터 주소를 추출하는 단계;

(b3) 상기 주소가 상기 인덱스 테이블에 존재하지 않으면, 상기 주소를 상기 인덱스 테이블에 코드블록의 시작주소로 삽입하는 단계;

(b4) 상기 주소가 상기 인덱스 테이블에 존재하면 상기 주소에 해당하는 코드블록의 호출 회수를 하나 증가시키는 단계 및,

(b5) 상기 코드블록의 호출 회수가 1회 이상이면, 상기 코드블록의 암호키를 난수로 생성하여 저장하는 단계를 포함하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 방법.

청구항 15

제12항에 있어서, 상기 (b)단계는,

(b6) 상기 실행파일의 마지막 실행코드를 검사하면, 각 코드블록의 크기를 계산하여 상기 인덱스 테이블에 저장하는 단계를 더 포함하는 것을 특징으로 하는 인덱스 테이블 기반 코드 암호화 방법.

청구항 16

제12항 내지 제15항 중 어느 한 항의 방법에 의해 암호화된 실행파일을 복호화하는 인덱스 테이블 기반 코드 복호화 방법에 있어서,

(d) 상기 실행파일의 암호화된 실행코드를 코드블록 단위로 코드블록의 암호키를 이용하여 복호화하되, 상기 인덱스 테이블을 참조하여, 상기 코드블록이 제1유형 코드블록이면 상기 코드블록의 호출 블록으로 암호키를 생성하여 이용하고, 제2유형 코드블록이면 상기 코드블록의 저장된 암호키를 이용하는 단계; 및,

(f) 복호화된 코드블록의 마지막 실행코드가 실행되면 상기 코드블록을 다시 암호화하는 단계를 포함하는 것을 특징으로 하는 인덱스 테이블 기반 코드 복호화 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 컴퓨터 프로그램의 실행파일을 암호화하거나, 암호화된 실행파일을 복호화하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법에 관한 것이다.

[0002] 특히, 본 발명은 1회 호출되는 코드블록은 자신을 호출하는 코드블록을 이용하여 암/복호화하고, 2회 이상 호출되는 코드블록은 난수로 생성한 암호키로 암/복호화하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법에 관한 것이다.

[0003] 또한, 본 발명은 코드블록의 시작주소, 호출회수, 블록의 크기 등을 저장하는 인덱스 테이블을 이용하여 실행코드를 블록으로 구분하거나 호출 회수를 계산하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법

에 관한 것이다.

배 경 기 술

[0004] 일반적으로, 소프트웨어의 저작권 침해로 인한 피해는 상당한 것으로 추산되고 있다. 특히, 역공학이 널리 알려진 이후로 이러한 저작권 침해는 매우 심각하다. 이것은 소프트웨어 특성상, 한번 배포된 이후에는 역공학에 의해 공격자에게 완전히 노출되기 때문이다.

[0005] 따라서 소프트웨어를 보호하기 위해서는 역공학에 대비한 기술이 필요하다. 코드 암호화 스킴(code encyption scheme)은 바이너리 실행 파일을 암호화하는 기술로써, 컴파일된 시점 이후의 프로그램을 암호화함으로써 이루어진다. 하지만, 능숙한 리버서(reverser)들은 이러한 스킴의 비밀키를 쉽게 찾아낼 수 있다. 이러한 문제를 해결하기 위해 코드 암호화 스킴은 안전한 키 관리 방법을 요구한다.

[0006] 이를 위해, Cappaert와 Jung은 런타임(runtime)에 바이너리 파일과 관련이 있는 비밀키를 생성하는 스킴을 제안하였다. 먼저 Cappaert는 올바른 비밀키를 생성하지 못하는 문제점이 있다. 만약 코드 암호화 스킴에서 비밀키가 올바르게 생성되지 못한다면 이것은 프로그램 충돌이나 다른 의도되지 않은 문제를 유발할 수 있다. 두 번째로 Jung의 스킴은 코드의 길이를 지나치게 크게 만든다. 이것은 효율성과 관련된 문제를 야기한다.

발명의 내용

해결하려는 과제

[0007] 본 발명의 목적은 상술한 바와 같은 문제점을 해결하기 위한 것으로, 1회 호출되는 코드블록은 자신을 호출하는 코드블록을 이용하여 암/복호화하고, 2회 이상 호출되는 코드블록은 난수로 생성한 암호키로 암/복호화하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법을 제공하는 것이다.

[0008] 또한, 본 발명의 목적은 코드블록의 시작주소, 호출회수, 블록의 크기 등을 저장하는 인덱스 테이블을 이용하여 실행코드를 블록으로 구분하거나 호출 회수를 계산하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법을 제공하는 것이다.

과제의 해결 수단

[0009] 상기 목적을 달성하기 위해 본 발명은 컴퓨터 프로그램의 실행파일을 암호화하는 인덱스 테이블 기반 코드 암호화 장치에 관한 것으로서, 상기 실행파일의 실행코드들을 호출코드에 의해 코드블록으로 구분하고, 각 코드블록이 호출되는 회수 및 시작주소를 인덱스 테이블에 저장하는 인덱스 생성부; 및, 상기 코드블록을 암호키로 암호화하되, 1회 호출되는 코드블록(이하 제1유형 코드블록)의 암호키는 상기 제1유형 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성하고, 2회 이상 호출되는 코드블록(이하 제2유형 코드블록)의 암호키는 난수로 생성하고 상기 실행파일에 저장하는, 블록 암호화부를 포함하는 것을 특징으로 한다.

[0010] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 실행코드는 바이너리 코드 또는 어셈블리 코드이고, 상기 호출코드는 바이너리 코드 또는 어셈블리 코드의 분기 코드 또는 점프 코드인 것을 특징으로 한다.

[0011] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 인덱스 생성부는 마지막 실행코드를 호출코드로 포함하는 일련의 실행코드를 하나의 코드블록으로 구분하되, 상기 호출코드는 상기 호출코드가 포함된 코드블록 이외의 다른 코드블록을 호출하는 것을 특징으로 한다.

[0012] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 인덱스 생성부는 코드블록의 호출 회수를 인덱스 테이블에 저장하되, 각 코드블록의 호출코드 각각에 대하여, 상기 호출코드가 호출하는 코드블록의 호출 회수를 하나씩 증가시켜 저장하는 것을 특징으로 한다.

[0013] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 블록 암호화부는 상기 제1유형 코드블록의 호출 블록을 해쉬하여 상기 제1 유형 코드블록의 암호키를 생성하는 것을 특징으로 한다.

[0014] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 블록 암호화부는 최초 코드블록의 암호키를 초기키로 정하는 것을 특징으로 한다.

[0015] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 블록 암호화부는 상기 제2유형 코드블록의 암호키를 초기키로 암호화하여 상기 실행파일의 데이터 영역에 저장하는 것을 특징으로 한다.

[0016] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 블록 암호화부는 상기 인덱스 테이블을 상기 실행파일의 데이터 영역에 저장하는 것을 특징으로 한다.

[0017] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 있어서, 상기 인덱스 생성부는 상기 코드블록의 크기를 상기 인덱스 테이블에 저장하는 것을 특징으로 한다.

[0018] 또한, 본 발명은 인덱스 테이블 기반 코드 암호화 장치에 의해 암호화된 실행파일을 복호화하는 인덱스 테이블 기반 코드 복호화 장치에 관한 것으로서, 상기 실행파일의 암호화된 실행코드를 코드블록 단위로 코드블록의 암호키를 이용하여 복호화하되, 상기 인덱스 테이블을 참조하여, 상기 코드블록이 제1유형 코드블록이면 상기 코드블록의 호출 블록으로 암호키를 생성하여 이용하고, 제2유형 코드블록이면 상기 코드블록의 저장된 암호키를 이용하는 블록 복호화부; 및, 복호화된 코드블록의 마지막 호출코드가 실행되면 상기 코드블록을 다시 암화화하는 블록 재암호부를 포함하는 것을 특징으로 한다.

[0019] 또, 본 발명은 인덱스 테이블 기반 코드 복호화 장치에 있어서, 상기 블록 재암호부는 상기 코드블록이 반복 실행되는 경우, 반복회수를 상기 인덱스 테이블에 저장하고 상기 코드블록이 호출될 때마다 상기 반복회수를 감소시켜, 반복회수가 0이 되면 상기 코드블록을 재암호화하는 것을 특징으로 한다.

[0020] 또한, 본 발명은 컴퓨터 프로그램의 실행파일을 암호화하는 인덱스 테이블 기반 코드 암호화 방법에 관한 것으로서, (b) 상기 실행파일의 실행코드들을 호출코드에 의해 코드블록으로 구분하고, 각 코드블록이 호출되는 회수 및 시작주소를 인덱스 테이블에 저장하고, 2회 이상 호출되는 코드블록(이하 제2유형 코드블록)에 대하여 암호키를 난수로 생성하여 저장하는 단계; 및, (c) 상기 코드블록을 상기 코드블록의 암호키로 암호화하되, 1회 호출되는 코드블록(이하 제1유형 코드블록)의 암호키는 상기 제1유형 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성하는 단계를 포함하는 것을 특징으로 한다.

[0021] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 방법에 있어서, 상기 실행코드는 어셈블리 코드이고, 상기 호출코드는 어셈블리 코드의 분기 코드 또는 점프 코드인 것을 특징으로 한다.

[0022] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 방법에 있어서, 상기 (b)단계는, (b1) 상기 실행코드들을 순차적으로 검사하는 단계; (b2) 검사한 실행코드가 호출코드이면 호출코드의 피연산자로부터 주소를 추출하는 단계; (b3) 상기 주소가 상기 인덱스 테이블에 존재하지 않으면, 상기 주소를 상기 인덱스 테이블에 코드블록의 시작주소로 삽입하는 단계; (b4) 상기 주소가 상기 인덱스 테이블에 존재하면 상기 주소에 해당하는 코드블록의 호출 회수를 하나 증가시키는 단계 및, (b5) 상기 코드블록의 호출 회수가 1회 이상이면, 상기 코드블록의 암호키를 난수로 생성하여 저장하는 단계를 포함하는 것을 특징으로 한다.

[0023] 또, 본 발명은 인덱스 테이블 기반 코드 암호화 방법에 있어서, 상기 (b)단계는, (b6) 상기 실행파일의 마지막 실행코드를 검사하면, 각 코드블록의 크기를 계산하여 상기 인덱스 테이블에 저장하는 단계를 더 포함하는 것을 특징으로 한다.

[0024] 또한, 본 발명은 상기 인덱스 테이블 기반 코드 암호화 방법에 의해 암호화된 실행파일을 복호화하는 인덱스 테이블 기반 코드 복호화 방법에 관한 것으로서, (d) 상기 실행파일의 암호화된 실행코드를 코드블록 단위로 코드블록의 암호키를 이용하여 복호화하되, 상기 인덱스 테이블을 참조하여, 상기 코드블록이 제1유형 코드블록이면 상기 코드블록의 호출 블록으로 암호키를 생성하여 이용하고, 제2유형 코드블록이면 상기 코드블록의 저장된 암호키를 이용하는 단계; 및, (f) 복호화된 코드블록의 마지막 실행코드가 실행되면 상기 코드블록을 다시 암호화하는 단계를 포함하는 것을 특징으로 한다.

발명의 효과

[0025] 상술한 바와 같이, 본 발명에 따른 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법에 의하면, 인덱스 테이블을 이용하여 코드블록의 호출회수에 따라 암호키를 달리 생성함으로써, 안전한 코드 암호화가 가능할 뿐만 아니라, 암호화 또는 복호화의 수행시간을 단축하고 저장하는 데이터의 양을 줄일 수 있는 효과가 얻어진다.

도면의 간단한 설명

[0026] 도 1은 본 발명을 실시하기 위한 전체 시스템 구성의 일례를 도시한 도면이다.

도 2a는 본 발명의 일실시예에 따른 실행파일의 일례이다.

도 2b는 본 발명의 일례에 따라 실행파일이 메모리에 상주된 상태를 표시한 것이다.

도 3은 본 발명에 사용된 용어 및 약어 표이다.

도 4는 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 암호화 방법을 설명하는 흐름도이다.

도 5는 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 암호화 방법의 의사코드(Pseudo-code)이다.

도 6은 본 발명의 일실시예에 따른 키체인의 일례를 도시한 것이다.

도 7은 본 발명의 일실시예에 따른 프로그래밍 언어와 어셈블리 코드를 예시한 것이다.

도 8과 9는 본 발명의 일실시예에 따른 실행코드와 그에 따른 인덱스 테이블 생성의 일례를 표시한 것이다.

도 10은 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 복호화 방법을 설명하는 흐름도이다.

도 11은 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 복호화 방법의 의사코드(Pseudo-code)이다.

도 12는 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 암호화 장치의 구성에 대한 블록도이다.

도 13은 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 복호화 장치의 구성에 대한 블록도이다.

* 도면의 주요 부분에 대한 부호의 설명 *

11,12 : 컴퓨터 단말	20 : 네트워크
30 : 암호화 장치	31 : 인덱스 생성부
32 : 블록 암호화부	
40 : 복호화 장치	41 : 블록 복호화부
42 : 블록 재암호부	
60 : 실행파일	60a : 암호화된 실행파일
61 : 실행코드 영역	61a : 실행코드
62 : 데이터 영역	63 : 코드블록
65 : 동적 데이터 영역	

발명을 실시하기 위한 구체적인 내용

[0027] 이하, 본 발명의 실시를 위한 구체적인 내용을 도면에 따라서 설명한다.

[0028] 또한, 본 발명을 설명하는데 있어서 동일 부분은 동일 부호를 붙이고, 그 반복 설명은 생략한다.

[0029] 먼저, 본 발명을 실시하기 위한 전체 시스템 구성의 예들을 도 1을 참조하여 설명한다.

[0030] 도 1에서 보는 바와 같이, 본 발명을 실시하기 위한 전체 시스템은 암호화 장치(30)와 복호화 장치(40)로 구

성된다.

[0031] 암호화 장치(30) 또는 복호화 장치(40)는 프로그램으로 구성되어 컴퓨터 단말(11, 12)에 설치되어 실행될 수 있다. 컴퓨터 단말(11, 12)에 설치된 프로그램은 하나의 시스템(30, 40)과 같이 동작할 수 있다. 한편, 다른 실시예로서, 암호화 장치(30) 또는 복호화 장치(40)는 ASIC(주문형 반도체) 등 하나의 전자회로로 구성되어 실시될 수 있다. 즉, 소프트웨어 형태, FPGA 칩이나 여러 개의 회로소자로 구성된 전자회로의 형태로 구성될 수도 있다. 그 외 가능한 다른 형태도 실시될 수 있다. 그러나 이하에서 설명의 편의를 위해 컴퓨터 단말(11, 12)에 구현된 암호화 장치(30) 또는 복호화 장치(40)로 설명하기로 한다.

[0032] 실행파일(60)은 컴퓨터 단말(11)의 저장매체에 미리 저장되고, 저장된 실행파일(60)을 암호화 장치(30)에 의해 읽어 입력될 수 있다. 암호화 장치(30)는 실행파일(60)의 실행코드를 암호화하여 암호화된 실행파일(60a)을 생성한다.

[0033] 도 2a에서 보는 바와 같이, 실행파일(60)은 실행코드 영역(61)과 데이터 영역(62)으로 구분된다. 실행코드(61)는 코드블록(62)으로 구분된다. 바람직하게는, 실행파일(60)은 바이너리 코드, 어셈블리 코드 등의 실행코드(61)로 구성된다. 실행코드(61)가 중앙처리장치(CPU)에 의해 직접 실행되는 코드이면 기계어 또는 바이너리 코드이어야 하나, 인터프리터 등에 의해 해석되어 실행되는 경우는 그 인터프리터에 맞는 어셈블리 코드 등으로 생성되어야 한다.

[0034] 실행코드 영역(61)은 어셈블리 코드 등 실행코드를 저장하기 위한 영역이고, 데이터 영역(62)은 데이터를 저장하기 위한 영역이다.

[0035] 이때, 암호화 장치(30)는 실행코드(61)를 코드블록(63) 단위로 암호화한다. 따라서 코드블록 단위로 복호화도 가능하다. 예를 들어, "코드블록 3"만 별도로 복호화할 수 있다.

[0036] 코드블록(63)은 일련의 실행코드(61a)로 구성된다. "실행코드"라는 용어는 실행코드들 전체를 의미하는 용어나, 하나의 라인인 실행코드를 의미하는 용어로 혼용한다.

[0037] 코드블록(63) 내의 실행코드(61a)들은 순차적으로 실행되고, 코드블록(63) 내의 마지막 실행코드(61a)는 호출코드로서, 호출코드가 실행되면 다른 코드블록(63)으로 이동되어 실행된다. 호출코드는 어셈블리 코드에서 점프 코드(jump) 또는 분기 코드(branch) 등을 의미한다. 즉, 코드블록(63)의 호출코드에 의해 순차적으로 다음 실행코드가 실행되지 않고, 다른 위치에 있는 코드블록(63)이 새롭게 시작된다.

[0038] 또한, 상기 실행코드의 코드블록에 대한 정보는 인덱스 테이블(70)에 저장된다. 인덱스 테이블(70)은 데이터 영역(62)에 저장된다. 인덱스 테이블(70)은 각 코드블록의 주소(또는 시작주소), 블록의 크기, 호출 회수 등을 저장하는 테이블로 구성된다. 바람직하게는, 인덱스 테이블(70)은 초기키로 암호화되어 저장될 수 있다. 또, 데이터 영역 전체도 초기키에 의해 암호화되어 저장될 수 있다.

[0039] 상기와 같이, 암호화된 실행파일(60a)은 네트워크(20)를 통해 배포되거나 오프라인 상으로 배포된다. 배포된 실행파일(60a)은 컴퓨터 단말(12)에 설치되어 실행된다.

[0040] 도 2b에서 보는 바와 같이, 실행파일(60a)의 실행코드(61) 및 데이터 영역(62)은 컴퓨터 단말(12)의 메모리(12b)에 로딩되어 상주한다. 각 실행코드(61a)들은 한 라인씩(하나의 커맨드씩) 컴퓨터 단말(12)의 CPU(12a)에 의해 읽혀져 실행된다. 이때, 실행코드(61)는 전체가 메모리(12b)에 로딩될 수도 있고, 일부만 로딩될 수도 있다. 이때, 하나의 라인의 실행코드는 주소에 의해 식별되고, 하나의 코드블록은 다수의 실행코드(다수의 라인 커맨드)로 구성된다.

[0041] 한편, 실행파일(60a)이 로딩되어 실행될 때(런타임 run-time 인 경우), 데이터 영역은 실행파일(60a)에도 생성된 고정된 데이터 영역(62)과, 스택(stack) 또는 유동적인 데이터를 위한 유동적인 데이터 영역(65)도 생성된다. 이하에서 구별없이 데이터 영역(62)으로만 기재하기로 한다.

[0042] 앞서 본 바와 같이, 하나의 코드블록(63)은 순차적으로 수행되고, 코드블록(63)의 마지막 실행코드인 호출코드가 실행되면 다른 코드블록으로 점프(jump)되어 실행된다. 동일한 코드블록 내에서 반복되어 순차적으로 수행되는 경우도 있을 수 있다.

[0043] 복호화 장치(40)는 암호화된 실행파일(60 또는 60a)의 실행코드를 복호화하여 실행코드가 실행될 수 있도록 한다. 이때, 복호화 장치(40)는 실행될 코드블록(63)만 복호화하고, 복호화된 코드블록(63)을 모두 수행하면 복호화한 코드블록(63)을 다시 암호화한다. 그리고 다음에 실행될 코드블록(63)을 복호화하여 복호화된 코드

블록(63)을 수행한다.

[0044] 도 2b의 예에서, 코드블록 1 -> 코드블록 3 -> 코드블록 2를 수행한다고 가정하면, 먼저 코드블록 1을 복호화한다. 코드블록 1을 모두 수행하면, 코드블록 1을 다시 암호화하고, 다음 실행할 코드블록 3을 복호화한다. 그리고 코드블록 3을 모두 수행하면, 코드블록 3을 암호화하고 코드블록 2를 복호화한다.

[0045] 다음으로, 본 발명을 설명하기 위한 용어 및 약어를 도 3을 참조하여 설명한다.

[0046] IK는 초기키로써, 난수와 인덱스 테이블(indexed table)을 동시에 보호한다. 난수는 다중 블록에 의해 호출되는 기본 블록(또는 코드블록)을 암호화 하며, 또한 초기키(IK)는 난수를 암호화한다. 초기키(IK)를 보호하고 또한 제공하는 것은 전적으로 어플리케이션에 의존하며, 초기키(IK)는 외장하드나 TPM(Trusted Platform Module) 등의 외부 저장매체에 저장할 수 있다. 따라서 초기키(IK)가 오프라인 상에서 안전하게 분배되고 저장한다고 가정한다.

[0047] 난수는 2개 이상의 코드블록으로부터 호출되는 코드블록을 암호화 하며, 난수는 초기키(IK)를 암호화 한다. 보호키(PK)는 초기키(IK)에 의해 암호화된 난수를 의미한다. 보호키(PK)는 바이너리 코드(또는 실행파일)의 데이터 섹션(또는 데이터 영역)에 저장된다. 또한 암호화 또는 복호화 알고리즘은 종래의 암복호화 기법을 이용하므로, 구체적 설명은 생략한다.

[0048] $E_K()$ 와 $D_K()$는 각각 암호키 K를 이용하여 암호화 또는 복호화 연산을 하는 것을 표시한다. H()는 일방향 해쉬 함수를 의미한다.

[0049] 다음으로, 본 발명에 따른 인덱스 테이블 기반 코드 암호화 및 복호화 장치 및 방법이 충족해야 하는 요구사항, 즉, 안전한 코드 암호 기법의 요구사항을 설명한다.

[0050] 먼저, 기밀성을 갖추어야 한다. 원본 바이너리 코드(실행코드 또는 어셈블리 코드)는 기밀성 유지를 통해 정적 분석으로부터 보호되어야 한다. 바이너리 코드를 흐름이나 제어 이동 등의 동적 분석으로부터 보호하기 위해 메모리 상에는 최소 개수의 코드 블록이 있어야 한다. 메모리 상에 코드가 암호화되어 있는 한 프로그램은 정적 및 동적 분석으로부터 보호된다.

[0051] 또, 메모리 덤프 방지가 되어야 한다. 만약 단일 루틴을 통해 하나의 프로그램을 통째로 암호화 한다면, 복호화 루틴이 프로그램 전체를 복호화한 후, 시작지점을 해당 지정해준다면 이것은 메모리 덤프 공격에 취약해진다. 따라서 최소한의 프로그램이 복호화 되어야 하며, 나머지 부분은 암호화된 상태를 유지해야 한다.

[0052] 또, 올바른 키 체인을 갖추어야 한다. 코드 암호화가 프로그램에 적용되면 올바른 키(또는 암호키)가 필요해진다. 만약 다중 경로가 존재할 때 올바른 키 체인을 갖지 못한다면 시스템 크래쉬나 의도하지 않은 실행이 가능하다.

[0053] 또, 템퍼링(tampering) 방지를 만족해야 한다. 템퍼링으로부터 보호되기 위해서는 무결성 유지가 필요하다. 이것은 아래와 같은 특성이 필요하다.

[0054] 암호화 과정에서, 한 비트가 하나의 기본 블록을 변경하면, 그 효과는 모든 암호화된 블록에 영향을 미쳐야 한다.

[0055] 복호화 과정에서, 만약 암호화된 블록에서 하나 이상의 비트가 변경된다면 복호화 결과는 하나 이상의 비트가 변경되어야 한다.

[0056] 다음으로, 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 암호화 방법을 도 4를 참조하여 보다 구체적으로 설명한다.

[0057] 도 4a에서 보는 바와 같이, 본 발명에 따른 인덱스 테이블 기반 코드 암호화 방법은 (a) 컴파일을 하는 단계(S10), (b) 인덱스 테이블을 생성하는 단계(S20), 및, (c) 코드블록을 암호화하는 단계(S30)로 구분된다.

[0058] 상기 (a)단계(또는 컴파일 하는 단계)는 프로그램 소스 코드를 컴파일하여 실행코드(어셈블리 코드 또는 바이너리 코드)를 생성하는 단계이다.

[0059] 상기 (b)단계는 실행코드들을 호출코드에 의해 코드블록으로 구분하고 각 코드블록의 호출 회수 및 시작주소 등의 정보를 인덱스 테이블로 생성한다(S20). 이때, 호출회수가 2회 이상인 코드블록(또는 제2유형 코드블록)에 대하여 난수로 암호키를 생성하여 저장한다.

[0060] 이때, 호출 회수가 2회 이상인 코드블록을 제2유형 코드블록이라 부르기로 하고, 그 외 코드블록을 제1유형 코드블록이라 칭하기로 한다. 제1유형 코드블록은 호출 회수가 1회이다. 한편, 최초 코드블록은 처음 프로그램 시작에 의해 1회 호출되는 것으로 가정한다.

[0061] 다음으로, (b)단계를 보다 구체적으로 설명하기에 앞서 (c)단계를 구체적으로 먼저 설명한다. 상기 (c)단계는 도 4c에 도시한 바와 같다.

[0062] 도 4c에서 보는 바와 같이, 상기 (c)단계는 코드블록을 암호키로 암호화한다. 이때, 1회 호출되는 코드블록의 암호키는 이 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성한다. 즉, 호출 블록을 이용하여 암호화한다. 또, 2회 이상 호출되는 코드블록은 난수로 생성된 암호키를 통해 암호화된다. 예를 들어, 난수 자체를 암호키로 사용하여 코드블록을 암호화한다.

[0063] 코드블록(또는 제1유형 코드블록)은 다음 [수학식 1]에 의해 암호화된다.

[0064] [수학식 1]

[0065] 

[0066] i는 인덱스 테이블(indexed table) 안에 있는 코드블록의 순서이다. 만약 플래그(또는 다중경로 플래그)가 0이 아니면, 블록(또는 코드블록)은 난수에 의해 암호화된 것이다. 초기키(IK)는 난수를 암호화 하고 실행파일 내에 저장된다. 만약 난수가 노출되면 해당 블록은 난수에 의해 복호화될 수 있고, 따라서 블록들은 공격자에게 분석 가능하기 때문에 이러한 암호화 단계가 필요하다.

[0067] 인덱스 테이블(indexed table)은 올바른 키 체인을 형성하기 위해 사용된다. 테이블 구성은 다음과 같다. 우선, 기본 블록(또는 코드블록)의 현재 주소를 저장한 후, 호출코드(jump나 branch 계열의 명령어)를 포인터를 이동하며 조사한다. 그러한 호출코드의 명령어들은 다음에 실행될 코드블록의 주소를 뒤에 (피연산자로서) 담고 있다.

[0068] 만약 다음 주소(호출코드의 피연산자)가 현재의 기본 블록을 가리키고 있다면 이것은 루프(또는 반복문)나 재귀를 의미한다. 루프나 재귀가 발생하면 cmp 명령어 등을 통해 호출 횟수를 정할 수가 있게 되며, 이러한 호출 횟수를 테이블에 저장한다. 이와 마찬가지로 만약 현재의 블록이 이미 테이블에 저장되어 있다면, 이것은 해당 코드블록의 다중 호출(2회 이상인 호출)을 의미한다. 보호키(PK)가 이 때 생성되며 바이너리 이미지(또는 실행파일의 바이너리 이미지)의 데이터 영역(data section)에 저장된다. 보호키(PK)는 난수를 암호화한 것으로서, 난수가 코드블록을 암호화하는 암호키로서 사용된다.

[0069] 도 6에서 보는 바와 같이, 기본 블록(또는 코드블록) D는 B, C 그리고 F 등의 다중 블록에 의해 호출된다. 즉, 3개의 서로 다른 코드블록에 의해 호출되고 있다. B의 암호키는 A 블록의 해시값이며, C의 비밀키와 같다. F는 A에 의해 직접 호출되지는 않으며, D에 의해 호출된다. 그 때 난수 r이 D의 비밀키(또는 암호키)로 생성되며 그것은 다시 초기키(IK)로 암호화 된다. 그 결과는 보호키(PK)의 새로운 생성이며, 이것은 바이너리 파일(또는 실행파일) 내에 저장된다. 보호키(PK)는 난수 r을 초기키(IK)에 의해 암호화된 값을 의미한다.

[0070] 리눅스나 윈도우 등의 일반적인 컴퓨터 운영체제는 실행파일 이미지 내에 변수를 저장할 수 있는 데이터 영역(data section)을 지원하며 따라서 보호키(PK)는 이러한 실행파일 내에 존재하는 데이터 영역(data section) 내에 저장할 수 있다.

[0071] 인덱스 테이블(indexed table)은 반복 호출되는 횟수를 포함한다. 만약 이것이 고려되지 않으면, 루프 또는 재귀 등을 의미하는 기본 블록(또는 코드블록)이 여러 번 복호화될 수 있다. 따라서 루프나 재귀 횟수를 테이블 내에 표시해둠으로써 이러한 문제를 해결할 수 있다. 만약 기본 블록이 호출되면 테이블 내에 저장된 호출 횟수가 1 감소하고, 이러한 방식으로 호출 횟수가 0이 되면 해당 블록은 메모리 덤프 방지를 위해 재암호화된다.

[0072] 예를 들어, 루프의 경우 C 언어로 사용된 예시는 도 7과 같다. cmp 명령어에서 두 번째 피연산자의 값은 "0A"이다. 이것은 블록 "loc_401006"은 10번 수행된다는 의미이며, 이것은 블록의 루프 또는 재귀 횟수임을 알

수 있다.

[0073] 도 4c를 참조하여 상기 설명을 부연하면, 최초 코드블록(P_0)은 초기키(IK)로 암호화한다(S31). 인덱스 테이블에서 다음 코드블록의 주소(시작주소) 및 유형을 읽어와서(S32) 무슨 유형인지를 판단한다(S33). 인덱스 테이블에서 플래그(또는 다중경로 호출 플래그)가 "0"이면 제1유형 코드블록이고, 0이 아니면 제2유형 코드블록이다.

[0074] 현재 코드블록 P_i가 제1유형 코드블록이면, 직전 호출블록(또는 인덱스 테이블에서 직전 코드블록) P_{i-1} 를 해쉬하여 해쉬값 $H(P_{i-1})$로 현재 코드블록 P_i를 암호화한다(S34). 그리고 현재 코드블록 P_i가 제2유형 코드블록이면, 난수를 발생하여(S35) 난수 r로 현재 코드블록 을 암호화한다(S36). 이때, (b)단계에서 발생시킨 난수를 이용할 수도 있고, (b)단계에서 난수 발생을 하지 않으면 이 단계에서 난수를 발생한다.

[0075] 인덱스 테이블에서 읽어온 현재 코드블록 P_i가 마지막 코드블록인지를 확인하여(S37), 마지막 이면 종료하고, 그러하지 않으면, 다음 코드블록을 읽어와 상기 과정을 반복한다.

[0076] 다음으로, 상기 (b)단계를 도 4b와 도 8 및 도 9를 참조하여 보다 구체적으로 설명한다. 도 4b는 인덱스 테이블을 생성하는 단계를 설명하는 흐름도이고, 도 9는 도 8의 실행코드 예로부터 인덱스 테이블을 생성한 일례를 도시한 것이다.

[0077] 도 8에서 보는 바와 같이, 예시 코드(또는 실행코드)는 크게 5개의 코드블록으로 나누어질 수 있다. 기본 블록(또는 코드블록)들은 jump나 branch 계열의 명령어(또는 호출코드의 명령어)에 의해 나누어진다. 먼저 초기화가 수행된다. 0x0040103E는 프로그램의 시작지점으로 세팅된다. 이후 jump나 branch 계열의 명령어가 탐색된다.

[0078] 도 9에서 보는 바와 같이, 만약 명령어가 jump나 branch라면, 피연산자를 테이블에 저장하며 이것은 다른 블록의 첫 번째 주소가 되기 때문이다. 예를 들어 0x0040105A는 주소 0x0040104F에 존재하는 "jne 0x0040105A" 코드에 의해 테이블에 저장된다. 명령어의 다음주소는 다른 블록의 첫 번째 주소가 된다. 즉, 0x00401051은 테이블에 저장된다. 이러한 방법으로 0x0040106C와 0x00401060이 순서대로 저장된다. 주소 0x0040106A에서 명령어는 "jmp 0x00401051"이 식별된다. 0x00401051은 이미 테이블에 저장되어 있으므로, 이것은 다중 호출되는 블록이라고 할 수 있다. 따라서 해당 블록의 정보가 갱신되고, 난수가 생성된다. 이러한 방법으로 모든 블록들이 식별되고 구분된다.

[0079] 도 9의 인덱스 테이블에서 주소는 코드블록의 시작주소이며, 코드블록을 식별하는 식별자 기능을 한다. 또, 호출회수는 다른 코드블록이 해당 코드블록을 호출하는 회수를 기록한 것이다. 플래그는 다중경로 호출 플래그로서, 1회 호출되는 경우 "0"을 기록하고, 2회 이상 호출되어 다중경로의 호출이 형성되는 경우 "1"을 기록한다.

[0080] 상기 (b) 단계 및 (c) 단계에 대한 의사 코드(Pseudo-code)는 각각 도 5a와 도 5b와 같다.

[0081] 다음으로, 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 복호화 방법을 도 10을 참조하여 보다 구체적으로 설명한다.

[0082] 도 10에서 보는 바와 같이, 본 발명에 따른 인덱스 테이블 기반 코드 복호화 방법은 크게 (d) 코드블록을 복화화하는 단계(S40), (e) 코드블록의 실행코드를 실행하는 단계(S50), 및, (e) 복화화된 코드블록을 재 암호화하는 단계(S60)로 구분된다.

[0083] 먼저, 최초 코드블록은 초기키(IK)로 복호화한다(S41). 최초 코드블록의 실행이 완료되면, 다음 코드블록을 읽어온다. 이때, 인덱스 테이블에서 다음 코드블록의 주소를 검색한다(S42). 다음 코드블록이 어느 유형(제1유형 또는 제2유형)인지를 판단하여(S43), 제1유형이면 직전 코드블록(또는 호출블록)을 해쉬하여 해쉬값을 암호키로 하여 코드블록을 복호화한다(S44). 제2 유형이면 저장된 보호키를 가져와서 초기키(IK)로 복호화하여 난수를 추출하고(S45), 추출된 난수를 암호키로 하여 코드블록을 복호화한다(S46). 그리고 코드블록을 실행하고, 실행된 코드블록을 다시 암호화한다(S60). 모든 코드블록이 완료되어 마지막 코드블록이면 종료한다(S47).

[0084] 즉, 상기 (d)단계는, 실행파일의 암호화된 실행코드를 코드블록 단위로 코드블록의 암호키를 이용하여 복호화하되, 상기 인덱스 테이블을 참조하여, 상기 코드블록이 제1유형 코드블록이면 상기 코드블록의 호출 블록으

로 암호키를 생성하여 이용하고, 제2유형 코드블록이면 상기 코드블록의 저장된 암호키(또는 보호키를 복호화한 난수)를 이용한다.

[0085] 먼저, 프로그램이 시작되기 전에, 인덱스 테이블을 참조하여 프로그램의 시작점(entry point)을 찾는다. 그리고 암호화된 코드블록 C_i를 실행코드 또는 코드블록 P_i로 복호화한다. 암호화된 코드블록은 직전 블록의 해쉬값을 이용하여 복호화되어, 복호화된 코드 P_i 가 실행된다. 만약 직전 블록 P_{i-1} 가 P'_{i-1} 으로 변경(tampering)되었다면, 암호키는 $H(P_{i-1}) \neq H(P'_{i-1})$이다. 따라서 코드블록 P_i 는 올바로 복호화되지 않을 것이다.

[0086] 한편, 인덱스 테이블은 플래그를 포함한다. 플래그는 코드블록이 난수를 이용하는지 안하는지를 가리키는 플래그이다. 만약 플래그가 1이면, 암호화된 코드블록은 난수로 복호화해야 한다. 복호화 코드 작업이 완료되면, 복호화된 코드블록은 다시 암호화되어 메모리에 저장된다.

[0087] 다음으로, 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 암호화 장치(30)의 구성을 도 12를 참조하여 보다 구체적으로 설명한다.

[0088] 도 12에서 보는 바와 같이, 본 발명에 따른 암호화 장치(30)는 인덱스 생성부(31) 및, 블록 암호화부(32)로 구성된다.

[0089] 인덱스 생성부(31)는 실행파일의 실행코드들을 호출코드에 의해 코드블록으로 구분하고, 각 코드블록이 호출되는 회수 및 시작주소를 인덱스 테이블에 저장한다.

[0090] 블록 암호화부(32)는 코드블록을 암호키로 암호화하되, 1회 호출되는 코드블록(이하 제1유형 코드블록)의 암호키는 상기 제1유형 코드블록을 호출하는 코드블록(이하 호출 블록)을 이용하여 생성하고, 2회 이상 호출되는 코드블록(이하 제2유형 코드블록)의 암호키는 난수로 생성하고 상기 실행파일에 저장한다.

[0091] 이때, 실행코드는 바이너리 코드 또는 어셈블리 코드이고, 상기 호출코드는 바이너리 코드 또는 어셈블리 코드의 분기 코드 또는 점프 코드이다.

[0092] 한편, 인덱스 생성부(31)는 마지막 실행코드를 호출코드로 포함하는 일련의 실행코드를 하나의 코드블록으로 구분하되, 상기 호출코드는 상기 호출코드가 포함된 코드블록 이외의 다른 코드블록을 호출한다. 또, 인덱스 생성부(31)는 코드블록의 호출 회수를 인덱스 테이블에 저장하되, 각 코드블록의 호출코드 각각에 대하여, 상기 호출코드가 호출하는 코드블록의 호출 회수를 하나씩 증가시켜 저장한다. 또, 인덱스 생성부(31)는 상기 코드블록의 크기를 상기 인덱스 테이블에 저장한다.

[0093] 블록 암호화부(32)는 상기 제1유형 코드블록의 호출 블록을 해쉬하여 상기 제1 유형 코드블록의 암호키를 생성한다. 또, 블록 암호화부(32)는 최초 코드블록의 암호키를 초기키로 정한다. 또, 블록 암호화부(32)는 상기 제2유형 코드블록의 암호키를 초기키로 암호화하여 상기 실행파일의 데이터 영역에 저장한다. 바람직하게는, 블록 암호화부(32)는 상기 인덱스 테이블을 상기 실행파일의 데이터 영역에 저장한다.

[0094] 다음으로, 본 발명의 일실시예에 따른 인덱스 테이블 기반 코드 복호화 장치(40)의 구성을 도 13을 참조하여 보다 구체적으로 설명한다.

[0095] 도 13에서 보는 바와 같이, 본 발명에 따른 복호화 장치(40)는 블록 복호화부(41) 및, 블록 재암호부(42)로 구성된다.

[0096] 블록 복호화부(41)는 상기 실행파일의 암호화된 실행코드를 코드블록 단위로 코드블록의 암호키를 이용하여 복호화하되, 상기 인덱스 테이블을 참조하여, 상기 코드블록이 제1유형 코드블록이면 상기 코드블록의 호출 블록으로 암호키를 생성하여 이용하고, 제2유형 코드블록이면 상기 코드블록의 저장된 암호키를 이용한다.

[0097] 블록 재암호부(42)는 복호화된 코드블록의 마지막 호출코드가 실행되면 상기 코드블록을 다시 암화화한다. 특히, 블록 재암호부(42)는 상기 코드블록이 반복실행되는 경우, 반복회수를 상기 인덱스 테이블에 저장하고 상기 코드블록이 호출될 때마다 상기 반복회수를 감소시켜, 반복회수가 0이 되면 상기 코드블록을 재암호화한다.

[0098] 상기 암호화 또는 복호화 장치에 대한 설명 중 생략한 부분은 앞서 설명한 암호화 또는 복호화 방법의 기재를 참조한다.

[0099] 이상, 본 발명자에 의해서 이루어진 발명을 실시 예에 따라 구체적으로 설명하였지만, 본 발명은 실시 예에 한정되는 것은 아니고, 그 요지를 이탈하지 않는 범위에서 여러 가지로 변경 가능한 것은 물론이다.

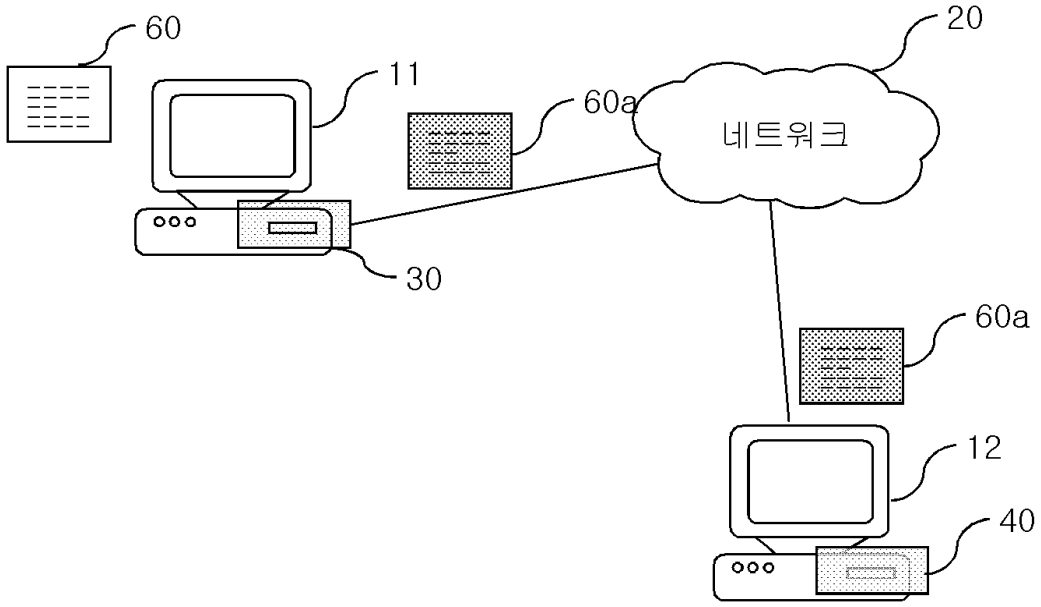
[0100]

산업상 이용가능성

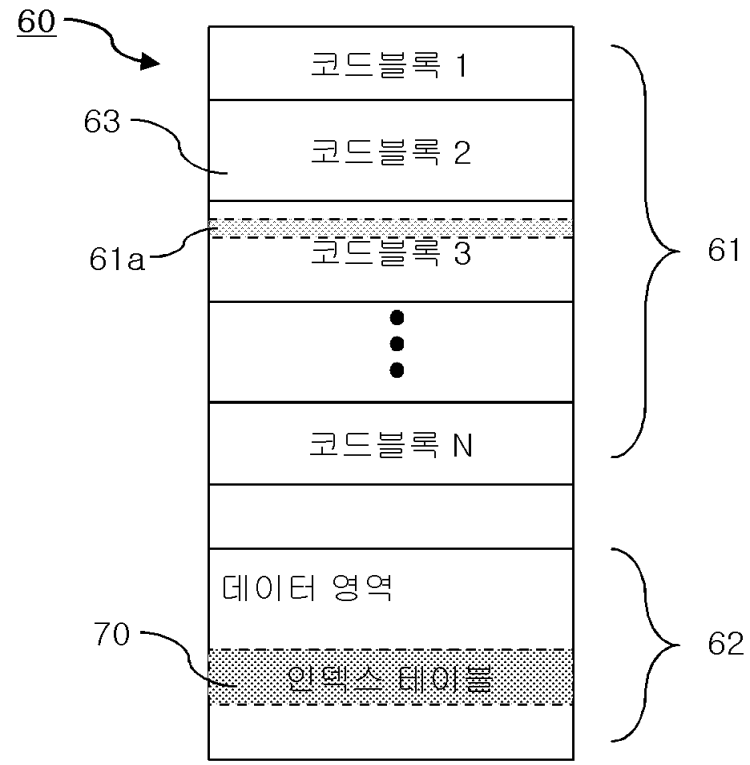
[0101] 본 발명은 1회 호출되는 코드블록을 직전 호출 코드블록으로 암/복호화하고, 2회 이상 호출되는 코드블록을 난수로 생성한 암호키로 암/복호화하는 인덱스 테이블 기반 코드 암호화 및 복호화 장치를 개발하는 데 유용하다.

도면

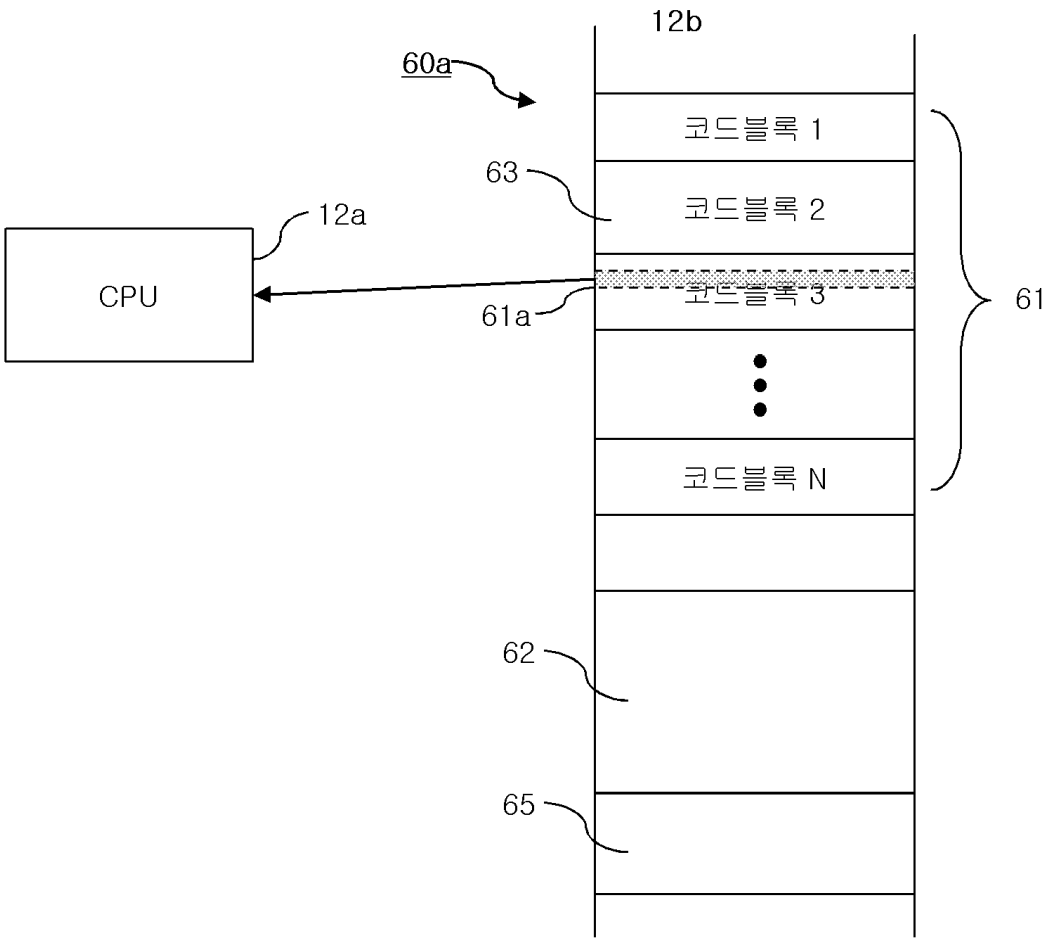
도면1



도면2a


60
63
61a
코드블록 1
코드블록 2
코드블록 3
코드블록 N
61
데이터 영역
70
인덱스 테이블
62

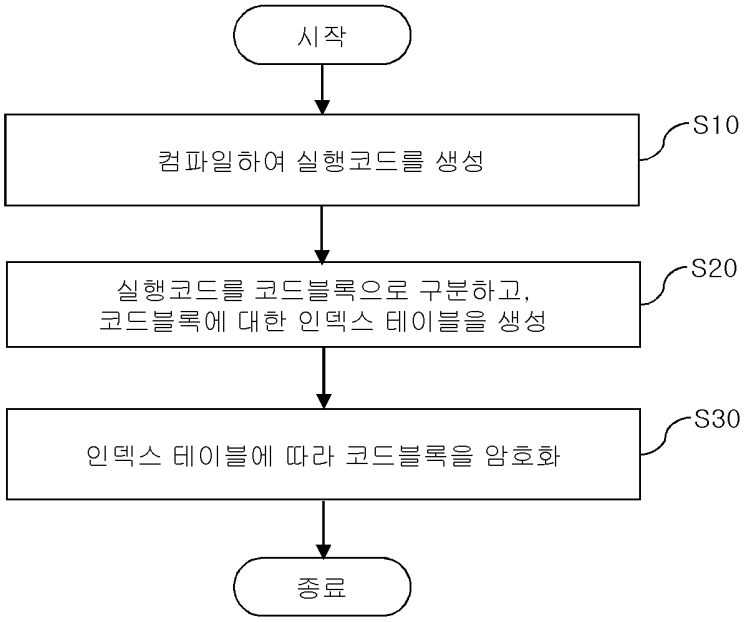
도면2b


12b
60a
63
12a
CPU
61a
코드블록 1
코드블록 2
코드블록 3
코드블록 N
61
62
65

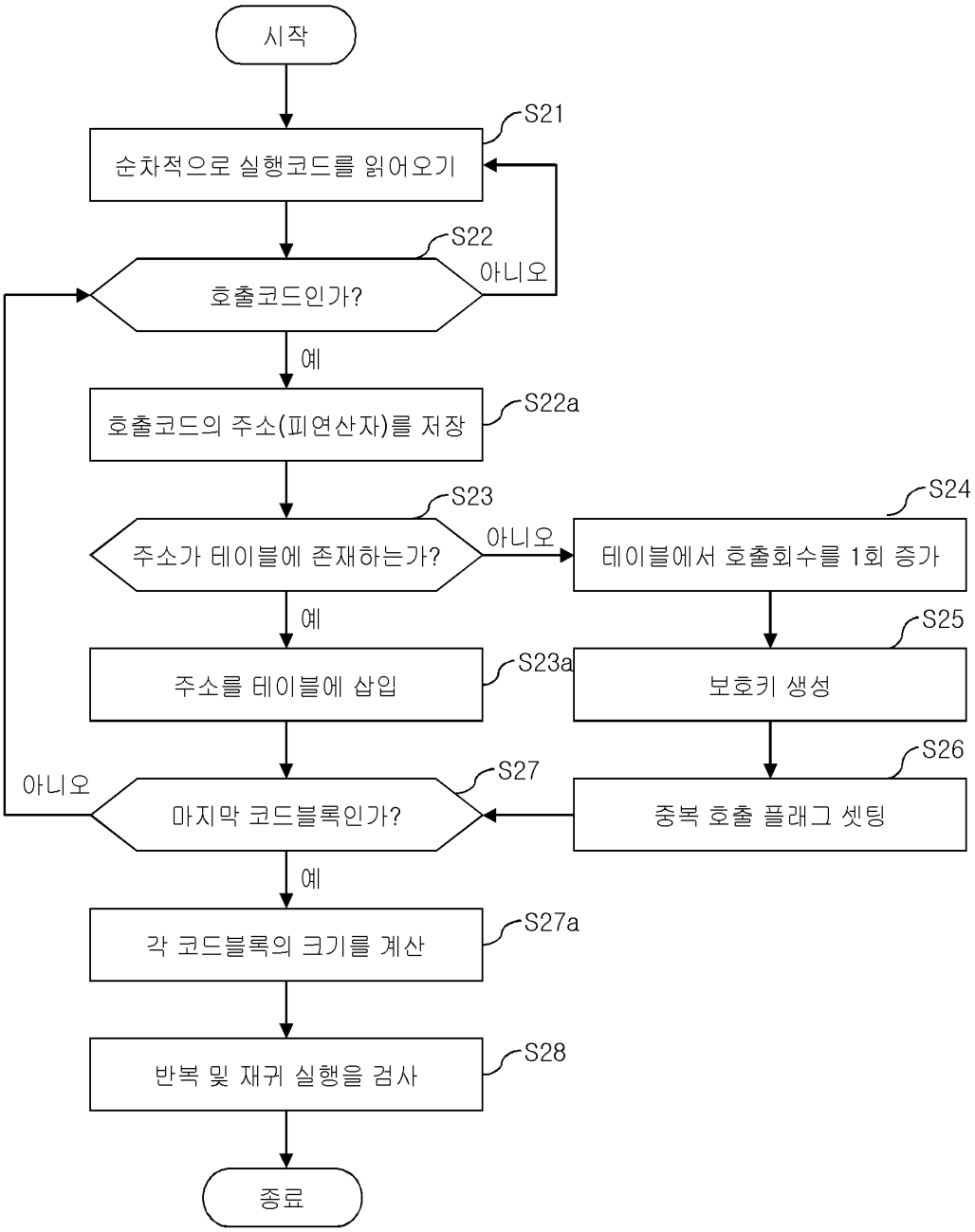
도면3

용어 및 약어	설명
IK	초기키(Initial Key)
PK	보호키(Protected Key)
$E_K(\cdot)$	키 K에 의한 암호화(Encrypt with key K)
$D_K(\cdot)$	키 K에 의한 복호화(Decrypt with key K)
$H(\cdot)$	일방향 해쉬함수(One-way hash function)
Rand	난수(Random number)
$A \sim Z$	바이너리 코드의 기본 블록(Basic blocks in binary code)
$a \sim z$	기본 블록 A~Z의 m 비트 (m bits of basic block A~Z)

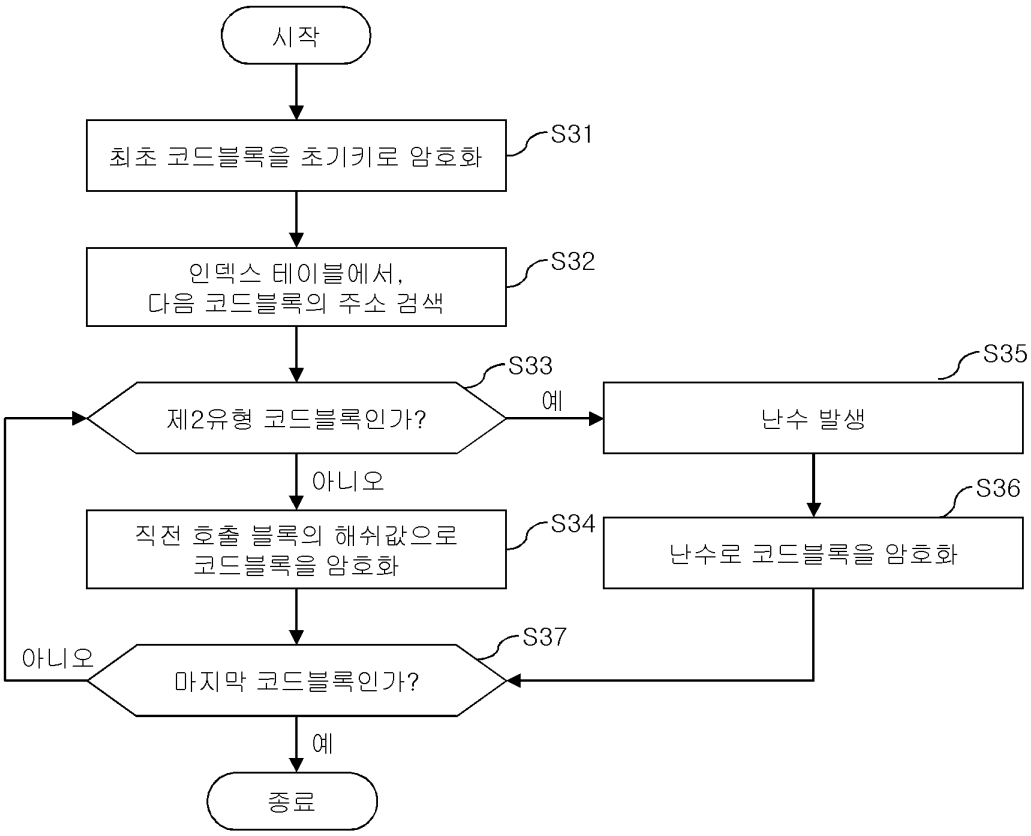
도면4a



도면4b


시작
S21
순차적으로 실행코드를 읽어오기
S22
호출코드인가?
아니오
예
S22a
호출코드의 주소(피연산자)를 저장
S23
주소가 테이블에 존재하는가?
아니오
S24
테이블에서 호출회수를 1회 증가
예
S25
주소를 테이블에 삽입
S23a
보호키 생성
S26
아니오
S27
마지막 코드블록인가?
중복 호출 플래그 셋팅
예
S27a
각 코드블록의 크기를 계산
S28
반복 및 재귀 실행을 검사
종료

도면4c


시작
최초 코드블록을 초기키로 암호화
S31
인덱스 테이블에서,
다음 코드블록의 주소 검색
S32
S33
제2유형 코드블록인가?
예
S35
난수 발생
아니오
S34
직전 호출 블록의 해쉬값으로
코드블록을 암호화
S36
난수로 코드블록을 암호화
S37
아니오
마지막 코드블록인가?
예
종료

도면5a

Procedure ConstructTable()

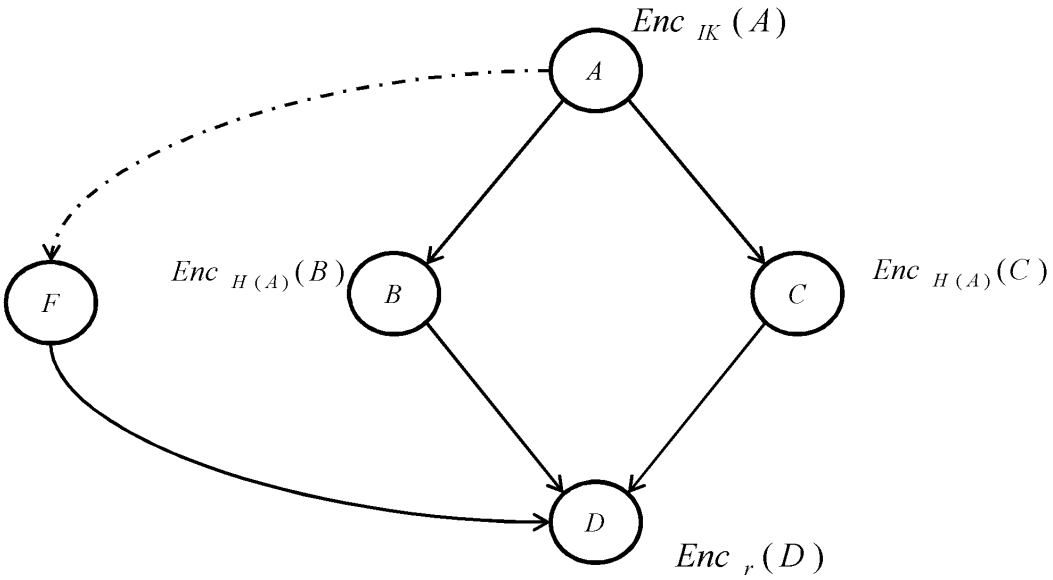
```
1     entrypoint ← Find_EntryPoint(); // store an address of entry point
2      currentPointer ← entrypoint;
3      nextPointer  ← currentPointer++;
3     index  ← 0;
4
5     while(File pointer is not end of file) {
6          if(Current_opcode == jump or Current_opcode == branch){
7          // branch or jump command is an unit of block
8                nextAddress ← operand;
9                // store an address of current address
10               if(currentAddress == nextAddress) {
11                     // loop, or recursion
12                     Tuple[index].Address ← currentAddress;
13                    Tuple[index].Size ← sizeofBlock;
14                     Tuple[index].Cnt ← prev_operand_2;
15                     Tuple[index].flag ← 0;
16                    // index, entry point address, size, number of calling, and no protected key
17                    StoreAttribute(Tuple[index]);
18               }
19                else{
20                     if(FindAddress(Tuple[index].currentAddress) {
21                       // repeated calling
22                            GenerateProtectedKey();
23                            StoretoDatasection();
24                            Tuple[index].flag ← 1
25                     }
26                }
27           nextBlock ← Get_NextBlock(currentAddress);
28           // Get next block's address
29           currentAddress ← nextAddress;
30           Sort(Tuple);
31           Compute_blocksize(Tuple);
32           CheckIterations();
33           index++;
34     }
```

도면5b

Procedure Encryption()

```
1   Compile(); // compile the source code to generate object or executable file
2
3   entrypoint ← Find_EntryPoint(); // store an address of entry point
4   currentAddress = entrypoint;       // initialization
5   nextAddress = 0;
6
7   ConstructTable()  // this procedure is described in Fig. 10.
8
9   nextAddress = Find_next(entrypoint);  // find an address of next block in current block
10  Encrypt(IK, entrypoint);  // encrypt first block
11
12  while(File pointer is not end of file)
13  {
14        if(SearchTable(nextAddress))   // if next block's flag in the indexed table indicates 1
15        {
16            random = GenerateRandom(); // generate random number
17            Encrypt(random, nextAddress); // Encrypt with the random number
18            Encrypt(random, IK);  // Encrypt the random number with the IK
19        }
20        else                                  // next block's flag indicates 0
21            Encrypt(random, currentAddress);  // Encrypt with the current block
22  }
```

도면6



도면7

```
for(i=0;i<10;i++)
{
   a[i] = rand();
}
```

(a)

```
loc_401006
call    _rand
add     esi, 1
cmp     esi, 0Ah
jl      short loc_401006
```

(b)

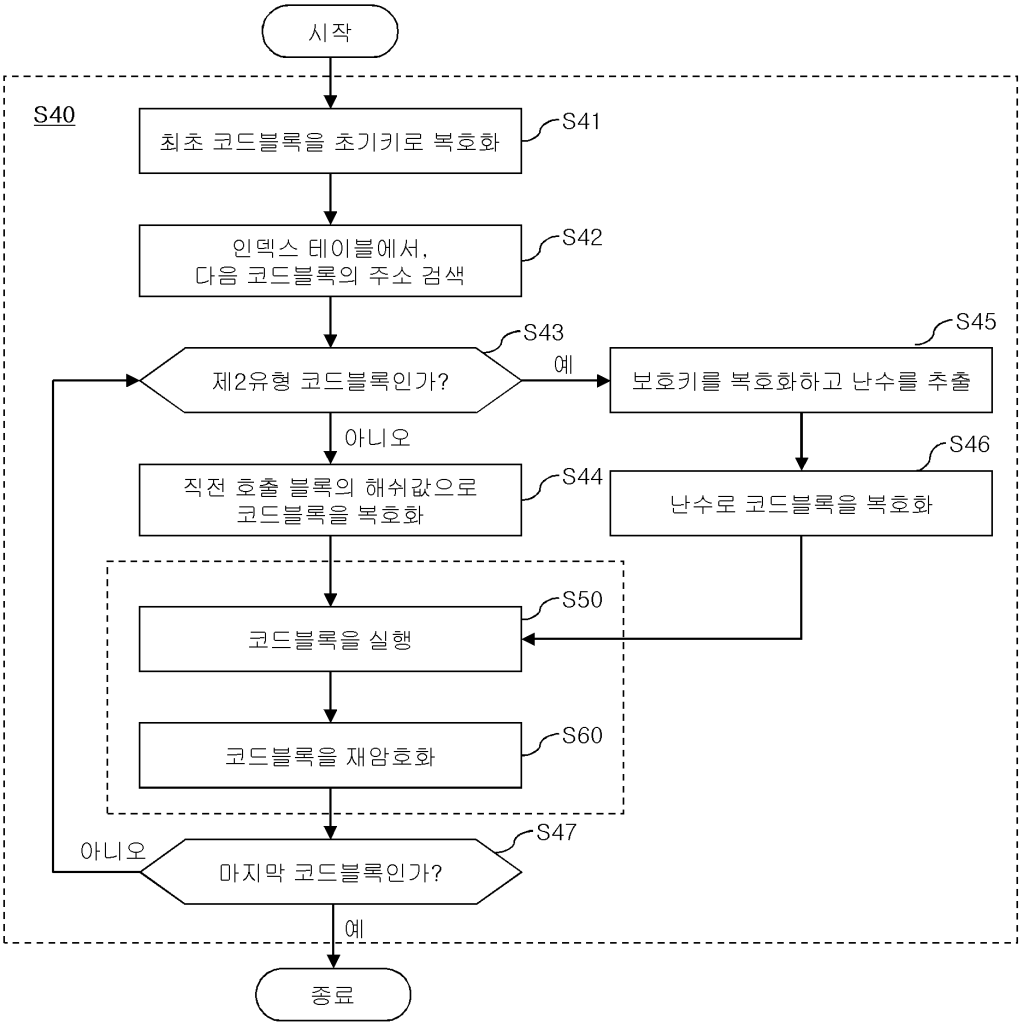
도면8

Basic block *A*	0040103E	mov	ecx,64h
	00401043	idiv	eax,ecx
	00401045	mov	dword ptr [ebp-0Ch],edx
	00401048	cmp	dword ptr [ebp-10h],8
	0040104F	jne	0040105a
B	00401051	mov	edx,dword ptr [ebp-10h]
	00401054	add	edx,1
	00401057	mov	dword ptr [ebp-10h],edx
C	0040105A	cmp	dword ptr [ebp-10h],5
	0040105E	jge	0040106c
D	00401060	mov	eax,dword ptr [ebp-4]
	00401063	imul	eax,dword ptr [ebp-10h]
	00401067	mov	dword ptr [ebp-4],eax
	0040106A	jmp	00401051
E	0040106C	mov	ecx,dword ptr [ebp-4]
	0040106F	cmp	ecx,dword ptr [ebp-8]
	00401072	jle	0040108a

도면9

주소(Offset)	블록크기	호출회수	플래그
0x0040103E	19	1	0
0x00401051	9	2	1
0x0040105A	6	2	1
0x00401060	12	1	0
0x0040106C	8	2	1
	…		

도면10

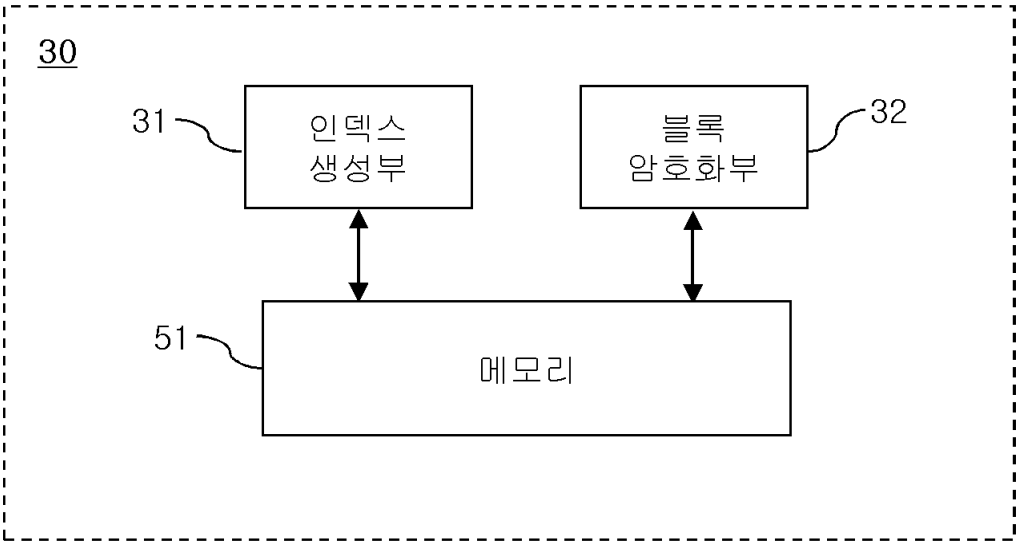

시작
S40
최초 코드블록을 초기키로 복호화
S41
인덱스 테이블에서,
다음 코드블록의 주소 검색
S42
S43
제2유형 코드블록인가?
예
S45
보호키를 복호화하고 난수를 추출
아니오
S46
직전 호출 블록의 해쉬값으로
코드블록을 복호화
S44
난수로 코드블록을 복호화
S50
코드블록을 실행
S60
코드블록을 재암호화
S47
아니오
마지막 코드블록인가?
예
종료

도면11

Procedure Decryption()

```
1   entrypoint ← Find_EntryPoint(); // store an address of entry point
2   currentAddress = entrypoint;        // initialization
3   nextAddress = 0;
4
5   nextAddress = Find_next(entrypoint);  // find an address of next block in current block
6   Decrypt(IK, entrypoint);  // decrypt first block
7   Execute(currentAddress); // execute the first block
8
9   while(File pointer is not end of file)
10  {
11        if(LookupTableTable(nextAddress))   // if next block's flag in the indexed table indicates 1
12        {
13            random = ExtractRandomNumber(nextAddress, IK);
14            // extract the random number with IK
15            Decrypt(random, nextAddress); // decrypt with the random number
16        }
17        else                                    // next block's flag indicates 0
18            Decrypt(random, currentAddress);  // encrypt with the current block
19
20  Execute(nextAddress); // execute the next block
21  ReEncrypt(currentAddress);
22
23  }
```

도면12



도면13