



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0081908
(43) 공개일자 2021년07월02일

(51) 국제특허분류(Int. Cl.)

H04L 9/06 (2006.01)

(52) CPC특허분류

H04L 9/0631 (2013.01)

H04L 2209/24 (2013.01)

(21) 출원번호 10-2019-0174322

(22) 출원일자 2019년12월24일

심사청구일자 2019년12월24일

(71) 출원인

주식회사 텔레칩스

서울특별시 송파구 올림픽로35다길 42(신천동)

(72) 발명자

이동수

경기도 용인시 수지구 풍덕천로 17, 621동 1701호(풍덕천동)

(74) 대리인

유미특허법인

전체 청구항 수 : 총 15 항

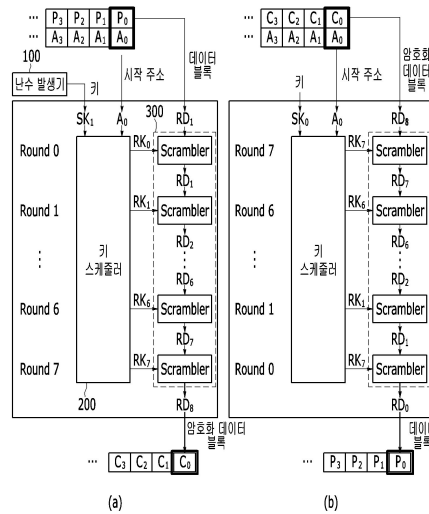
(54) 발명의 명칭 암호화 알고리즘이 적용된 시스템에서의 데이터 처리 방법

(57) 요약

적어도 하나의 프로세서에 의해 동작하는 암호화 알고리즘이 적용된 시스템이 비트들로 구성된 데이터 블록을 처리하기 위하여, 복수의 비트들로 구성된 스램블러 키와 데이터 블록의 시작 주소를 토대로, 데이터 블록을 암호화할 라운드 횟수만큼의 라운드 키들을 생성한다. 데이터 블록을 구성하는 복수의 비트들을 바이트 단위로 분리하여 서브 입력 데이터들로 생성하고, 암/복호화를 처리하는 복수의 각 바이트 레인으로 바이트 순서에 따라 각 서브 입력 데이터들을 입력한다.

암호화할 횟수만큼 생성된 라운드 키들을 바이트 단위로 분리하여 서브 라운드 키들로 생성하고, 각 암호화 라운드에 따라 서브 라운드 키들을 복수의 바이트 레인들로 각각 입력하여, 서브 라운드 키를 이용하여 서브 데이터를 암호화한다.

대표도 - 도4



명세서

청구범위

청구항 1

적어도 하나의 프로세서에 의해 동작하는 암호화 알고리즘이 적용된 시스템이 비트들로 구성된 데이터 블록을 처리하는 방법으로서,

복수의 비트들로 구성된 스캔블러 키와 상기 데이터 블록의 시작 주소를 토대로, 상기 데이터 블록을 암호화할 라운드 횟수만큼의 라운드 키들을 생성하는 단계,

상기 데이터 블록을 구성하는 복수의 비트들을 바이트 단위로 분리하여 서브 입력 데이터들로 생성하고, 암호화를 처리하는 복수의 각 바이트 레인으로 바이트 순서에 따라 각 서브 입력 데이터들을 입력하는 단계, 그리고

상기 암호화할 횟수만큼 생성된 라운드 키들을 바이트 단위로 분리하여 서브 라운드 키들로 생성하고, 각 암호화 라운드에 따라 상기 서브 라운드 키들을 상기 복수의 바이트 레인들로 각각 입력하여, 상기 서브 라운드 키를 이용하여 상기 서브 데이터를 암호화하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 2

제1항에 있어서,

상기 라운드 키들을 생성하는 단계는 각각,

제1 길이 비트인 제1 번째 스캔블러 키에서 미리 설정된 위치의 비트들을 선택하여 제2 길이 비트의 선택 비트 키를 생성하는 단계,

상기 제2 길이 비트로 구성된 시작 주소를 토대로 제2 길이 비트의 변환된 주소를 생성하는 단계, 그리고

상기 제2 길이 비트의 선택 비트 키와 상기 제2 길이 비트의 변환된 주소를 토대로 제1 번째 라운드 키를 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 3

제2항에 있어서,

상기 제2 길이 비트의 변환된 주소를 생성하는 단계는,

상기 제1 번째 스캔블러 키의 비트 길이가 128 비트이면, 상기 제2 길이 비트로 구성된 시작 주소의 비트들을 미리 설정된 위치로 변경하여 상기 제2 길이 비트의 변환된 주소로 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 4

제3항에 있어서,

상기 제2 길이 비트의 변환된 주소를 생성하는 단계는,

상기 제1 번째 스캔블러 키의 비트 길이가 64 비트이면, 상기 시작 주소를 형성하는 비트들에서 특정 비트에 해당하는 값을 한 비트로 더 생성하고, 상기 특정 비트에 해당하는 값의 역 서브바이트 연산한 값을 두 비트의 값으로 생성하는 단계, 그리고

상기 특정 비트를 기준으로 우측의 하위 1 비트 씩 세 개의 비트들은 삭제하고, 상기 특정 비트, 상기 특정 비트를 기준으로 좌측의 상위 비트들과 상기 생성한 세 비트들을 포함하여 상기 제2 비트의 변환된 주소로 생성하

는 단계

를 포함하고,

상기 특정 비트는 상기 시작 주소를 형성하는 비트들에서 가장 우측에서 4번째 비트에 해당하는, 데이터 블록 처리 방법.

청구항 5

제2항에 있어서,

상기 제1 번째 라운드 키를 생성하는 단계 이후에,

상기 제1 번째 라운드가 상기 암호화할 라운드 횟수의 마지막 횟수에 해당하지 않으면, 상기 제1 번째 스크램블러 키로부터 제2 번째 스크램블러 키를 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 6

제5항에 있어서,

상기 제2 번째 스크램블러 키를 생성하는 단계는,

상기 제1 번째 스크램블러 키의 상위 8비트는 역 서브바이트 연산하는 단계,

역 서브바이트 연산된 상위 8비트와 상기 상위 8비트를 제외한 하위 비트들로 구성된 제1 번째 스크램블러 키에서, 하위 3비트를 제1 방향으로 순환 시프트하는 단계, 그리고

상기 하위 3비트가 제1 방향으로 순환 시프트하여 생성된 키를 상기 제2 번째 스크램블러 키로 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 7

제2항에 있어서,

상기 제1 번째 라운드 키를 생성하는 단계는,

상기 제2 길이 비트의 선택 비트 기와 상기 제2 길이 비트의 변환된 주소를 배타적 논리 합 하여 제2 길이 비트의 결합 키로 생성하는 단계,

상기 결합 키의 각 비트들을 역 서브바이트 연산하여 제2 길이 비트의 인버스 키로 생성하는 단계,

상기 제2 길이 비트의 인버스 키의 비트들의 위치를 미리 결정한 위치로 변경하여 제2 길이 비트의 비트 위치 변경 키로 생성하는 단계,

상기 제2 길이 비트의 비트 위치 변경 키의 각 비트들을 역 서브바이트 연산하여 제2 길이 비트의 제2 인버스 키로 생성하는 단계,

상기 제2 인버스 키의 제2 길이 비트를 제1 길이 비트로 변경하여, 제1 길이 비트의 길이 변경 키로 생성하는 단계, 그리고

상기 제1 길이 비트의 제1 번째 스크램블러 키와 상기 제1 길이 비트의 길이 변경 키를 배타적 논리 합 하여, 상기 제1 번째 라운드 키로 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 8

제7항에 있어서,

상기 인버스 키로 생성하는 단계와 상기 비트 위치 변경 키로 생성하는 단계는 복수의 횟수로 생성되는, 데이터 블록 처리 방법.

청구항 9

제1항에 있어서,

상기 서브 데이터를 암호화하는 단계는,

상기 서브 데이터를 역 서브바이트 연산하여 역 서브바이트 연산된 서브 데이터를 생성하는 단계,

j-0번째 서브 라운드 키와 상기 역 서브바이트 연산된 서브 데이터를 배타적 논리 합하여 제1 서브 바이트를 출력하는 단계,

j-1번째 서브 라운드 키를 기초로 비트 순환 회전 결정 값을 계산하고, 계산한 비트 순환 회전 결정 값을 토대로 상기 제1 서브 바이트의 한 비트가 순환 회전되거나 상기 제1 서브 바이트의 비트 순서 그대로 하여 순환 처리 서브 바이트를 생성하는 단계,

j-2 번째 서브 라운드 키를 구성하는 각 비트들로 계산된 제1 값과 제2 값을 이용하여, 상기 순환 처리 서브 바이트의 비트 위치를 변경하여 비트 재배치 서브 바이트를 생성하는 단계,

j-3번째 서브 라운드 키와 상기 비트 재배치 서브 바이트를 배타적 논리 합하여 제2 서브 바이트를 해당 라운드의 라운드 바이트로 출력하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 10

제9항에 있어서,

상기 j는 상기 서브 데이터가 상기 데이터 블록에서 몇 번째 바이트에 해당하는지 나타내는 정수이고,

상기 j-0, j-1, j-2, j-3의 계산 값이 0 이하인 경우, 계산된 각 값을 16으로 나눈 나머지 값을 가지는 서브 라운드 키를 사용하는, 데이터 블록 처리 방법.

청구항 11

제9항에 있어서,

상기 순환 처리 서브 바이트를 생성하는 단계는,

상기 j-1번째 서브 라운드 키를 구성하는 비트들 중 미리 설정된 제1 비트와 제2 비트의 배타적 논리 합의 값이 1이면 상기 제1 서브 바이트를 한 비트씩 좌측으로 순환 회전하여 상기 순환 처리 서브 바이트로 생성하고,

상기 제1 비트와 제2 비트의 배타적 논리 합의 값이 0이면 상기 제1 서브 바이트를 상기 순환 처리 서브 바이트로 생성하는, 데이터 블록 처리 방법.

청구항 12

제11항에 있어서,

상기 비트 재배치 서브 바이트를 생성하는 단계는,

상기 순환 처리 서브 바이트와 상기 j-2번째 서브 라운드 키를 입력받는 단계,

상기 j-2번째 서브 라운드 키의 제1, 제2, 제3 비트 값의 배타적 논리 합하여 상기 제1 값을 계산하는 단계,

상기 j-2번째 서브 라운드 키의 제4, 제5, 제6 비트 값의 배타적 논리 합하여 상기 제2 값을 계산하는 단계, 그리고

상기 제1 값과 제2 값들에 따라, 상기 순환 처리 서브 바이트의 비트들의 위치를 변경하여 상기 비트 재배치 서브 바이트로 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 13

제11항에 있어서,

상기 서버 데이터를 암호화하는 단계 이후에,

암호화된 데이터 블록을 복호화하는 단계

를 더 포함하고,

상기 복호화하는 단계는,

암호화된 데이터 블록과 상기 암호화된 데이터 블록의 시작 주소, 그리고 상기 데이터 블록을 암호화할 때 생성한 스캐블러 키를 입력받는 단계,

상기 암호화된 데이터 블록과 상기 스캐블러 키를 기초로 상기 암호화할 횟수만큼 생성된 라운드 키들을 생성하는 단계, 그리고

암호화 라운드의 역순으로 라운드 키들을 상기 복수의 바이트 레인들로 입력하여 상기 암호화된 데이터 블록을 복호화하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 14

제13항에 있어서,

상기 복호화하는 단계는,

상기 j-3번째 서버 라운드 키와 암호화된 서버 데이터를 배타적 논리 합하여, 상기 비트 재배치 서버 바이트를 생성하는 단계,

상기 j-2 번째 서버 라운드 키를 구성하는 각 비트들로 계산된 제1 값과 제2 값을 이용하여, 상기 비트 재배치 서버 바이트의 비트 위치를 변경하여, 상기 순환 처리 서버 바이트를 생성하는 단계,

상기 j-1번째 서버 라운드 키를 기초로 비트 순환 회전 결정 값을 계산하고, 계산한 비트 순환 회전 결정 값을 토대로 상기 순환 처리 서버 바이트의 한 비트가 순환 회전되거나 상기 순환 처리 서버 바이트의 비트 순서 그대로 하여 상기 제1 서버 바이트를 생성하는 단계,

상기 j-0번째 서버 라운드 키와 상기 제1 서버 바이트를 배타적 논리 합하여 상기 역 서버바이트 연산된 서버 데이터를 제1 서버 바이트를 출력하는 단계,

상기 비트 재배치 서버 바이트와 상기 비트 재배치 서버 바이트를 배타적 논리 합하여, 상기 역 서버바이트 연산된 서버 데이터를 출력하는 단계, 그리고

상기 인버스된 서버 데이터를 서버바이트 연산하여 상기 서버 데이터를 생성하는 단계

를 포함하는, 데이터 블록 처리 방법.

청구항 15

제14항에 있어서,

상기 순환 처리 서버 바이트를 생성하는 단계는,

상기 j-1번째 서버 라운드 키를 구성하는 비트들 중 미리 설정된 제1 비트와 제2 비트의 배타적 논리 합의 값이 1이면 상기 제1 서버 바이트를 한 비트씩 우측으로 순환 회전하여 상기 순환 처리 서버 바이트로 생성하고,

상기 제1 비트와 제2 비트의 배타적 논리 합의 값이 0이면 상기 제1 서버 바이트를 상기 순환 처리 서버 바이트로 생성하는, 데이터 블록 처리 방법.

발명의 설명

기술 분야

본 발명은 대용량 데이터 처리를 위한 암호화 알고리즘이 적용된 시스템에서의 데이터 처리 방법에 관한 것이다.

[0001]

배경 기술

- [0002] 데이터 보호를 위해 사용되는 블록 암호화 알고리즘 중 상용 알고리즘으로 DES(Data Encryption Standard), AES(Advanced Encryption Standard) 등이 있다. 이들 알고리즘은 데이터를 블록 단위로 처리한다.
- [0003] DES의 경우 한 블록의 크기는 8 바이트이며, AES의 경우 한 블록의 크기가 16 바이트이다. 블록 암호화 알고리즘은 암호화 품질을 향상시키기 위해 데이터 블록 안에서의 데이터의 위치를 변경하는 작업을 수행한다. 이는 현재 처리하는 데이터와 이전 데이터, 이후 데이터 사이의 종속관계를 지우는 역할을 하여 암호화 품질을 높이는 역할을 한다.
- [0004] 이와 같이 블록 암호 알고리즘은 데이터 블록 안에서 데이터 위치를 바꾸어 암호화 품질은 높이지만, 데이터를 블록 단위로만 처리해야하는 제약을 가진다. 예를 들어, 블록 암호 알고리즘을 사용하여 데이터를 암호화하여 메모리에 저장하는 시스템이 있다고 가정하면, 블록 안의 특정 부분을 변경하려고 할 때 해당 시스템에서는 ‘읽기-수정-쓰기’라는 작업을 수행해야 한다.
- [0005] 따라서, 변경하고자 했던 데이터는 임의의 주소에 저장된 1 바이트 데이터이나, 해당 데이터가 포함된 블록 단위로 처리해야 하는 제약 때문에 1 블록을 모두 읽어 복호화 한 뒤, 데이터를 변경한다. 그리고 변경된 데이터를 포함하는 블록을 암호화하여 메모리에 저장해야 한다.
- [0006] 암호/복호화 알고리즘으로 블록 암호 알고리즘을 사용하는 경우, 1 블록 길이보다 짧은 길이의 데이터를 처리할 때는 반드시 발생하는 문제로, 암호/복호화에 시간이 많이 걸릴 수 있다. 또한, 메모리 접근 시간이 길어지는 경우 전체 시스템 성능이 크게 하락할 수 있다.

발명의 내용

해결하려는 과제

- [0007] 따라서, 본 발명은 메모리에 데이터를 암호화하여 저장할 수 있으면서도, '읽기-수정-쓰기'와 달리 '쓰기'만으로 메모리 내용을 변경할 수 있는 대용량 데이터 처리를 위한 암호화 알고리즘이 적용된 시스템에서의 데이터 처리 방법을 제공한다.

과제의 해결 수단

- [0008] 상기 본 발명의 기술적 과제를 달성하기 위한 본 발명의 하나의 특징인 적어도 하나의 프로세서에 의해 동작하는 암호화 알고리즘이 적용된 시스템이 비트들로 구성된 데이터 블록을 처리하는 방법으로서,
- [0009] 복수의 비트들로 구성된 스캔블러 키와 상기 데이터 블록의 시작 주소를 토대로, 상기 데이터 블록을 암호화할 라운드 횟수만큼의 라운드 키들을 생성하는 단계, 상기 데이터 블록을 구성하는 복수의 비트들을 바이트 단위로 분리하여 서브 입력 데이터들로 생성하고, 암호/복호화를 처리하는 복수의 각 바이트 레인으로 바이트 순서에 따라 각 서브 입력 데이터들을 입력하는 단계, 그리고 상기 암호화할 횟수만큼 생성된 라운드 키들을 바이트 단위로 분리하여 서브 라운드 키들로 생성하고, 각 암호화 라운드에 따라 상기 서브 라운드 키들을 상기 복수의 바이트 레인들로 각각 입력하여, 상기 서브 라운드 키를 이용하여 상기 서브 데이터를 암호화하는 단계를 포함한다.
- [0010] 상기 라운드 키들을 생성하는 단계는 각각, 제1 길이 비트인 제1 번째 스캔블러 키에서 미리 설정된 위치의 비트들을 선택하여 제2 길이 비트의 선택 비트 키를 생성하는 단계, 상기 제2 길이 비트로 구성된 시작 주소를 토대로 제2 길이 비트의 변환된 주소를 생성하는 단계, 그리고 상기 제2 길이 비트의 선택 비트 키와 상기 제2 길이 비트의 변환된 주소를 토대로 제1 번째 라운드 키를 생성하는 단계를 포함할 수 있다.
- [0011] 상기 제2 길이 비트의 변환된 주소를 생성하는 단계는, 상기 제1 번째 스캔블러 키의 비트 길이가 128 비트이면, 상기 제2 길이 비트로 구성된 시작 주소의 비트들을 미리 설정된 위치로 변경하여 상기 제2 길이 비트의 변환된 주소로 생성하는 단계를 포함할 수 있다.
- [0012] 상기 제2 길이 비트의 변환된 주소를 생성하는 단계는, 상기 제1 번째 스캔블러 키의 비트 길이가 64 비트이면, 상기 시작 주소를 형성하는 비트들에서 특정 비트에 해당하는 값을 한 비트로 더 생성하고, 상기 특정 비트에 해당하는 값의 역 서브바이트 연산한 값을 두 비트의 값으로 생성하는 단계, 그리고 상기 특정 비트를 기준으로 우측의 하위 1 비트 씩 세 개의 비트들은 삭제하고, 상기 특정 비트, 상기 특정 비트를 기준으로 좌측의

상위 비트들과 상기 생성한 세 비트들을 포함하여 상기 제2 비트의 변환된 주소로 생성하는 단계를 포함하고, 상기 특정 비트는 상기 시작 주소를 형성하는 비트들에서 가장 우측에서 4번째 비트에 해당한다.

- [0013] 상기 제1 번째 라운드 키를 생성하는 단계 이후에, 상기 제1 번째 라운드가 상기 암호화할 라운드 횟수의 마지막 횟수에 해당하지 않으면, 상기 제1 번째 스캔블러 키로부터 제2 번째 스캔블러 키를 생성하는 단계를 포함할 수 있다.
- [0014] 상기 제2 번째 스캔블러 키를 생성하는 단계는, 상기 제1 번째 스캔블러 키의 상위 8비트는 역 서브바이트 연산하는 단계, 역 서브바이트 연산된 상위 8비트와 상기 상위 8비트를 제외한 하위 비트들로 구성된 제1 번째 스캔블러 키에서, 하위 3비트를 제1 방향으로 순환 시프트하는 단계, 그리고 상기 하위 3비트가 제1 방향으로 순환 시프트하여 생성된 키를 상기 제2 번째 스캔블러 키로 생성하는 단계를 포함할 수 있다.
- [0015] 상기 제1 번째 라운드 키를 생성하는 단계는, 상기 제2 길이 비트의 선택 비트 기와 상기 제2 길이 비트의 변환된 주소를 배타적 논리 합 하여 제2 길이 비트의 결합 키로 생성하는 단계, 상기 결합 키의 각 비트들을 역 서브바이트 연산하여 제2 길이 비트의 인버스 키로 생성하는 단계, 상기 제2 길이 비트의 인버스 키의 비트들의 위치를 미리 결정된 위치로 변경하여 제2 길이 비트의 비트 위치 변경 키로 생성하는 단계, 상기 제2 길이 비트의 비트 위치 변경 키의 각 비트들을 역 서브바이트 연산하여 제2 길이 비트의 제2 인버스 키로 생성하는 단계, 상기 제2 인버스 키의 제2 길이 비트를 제1 길이 비트로 변경하여, 제1 길이 비트의 길이 변경 키로 생성하는 단계, 그리고 상기 제1 길이 비트의 제1 번째 스캔블러 키와 상기 제1 길이 비트의 길이 변경 키를 배타적 논리 합 하여, 상기 제1 번째 라운드 키로 생성하는 단계를 포함할 수 있다.
- [0016] 상기 인버스 키로 생성하는 단계와 상기 비트 위치 변경 키로 생성하는 단계는 복수의 횟수로 생성될 수 있다.
- [0017] 상기 서브 데이터를 암호화하는 단계는, 상기 서브 데이터를 역 서브바이트 연산하여 역 서브바이트 연산된 서브 데이터를 생성하는 단계, j-0번째 서브 라운드 키와 상기 역 서브바이트 연산된 서브 데이터를 배타적 논리 합하여 제1 서브 바이트를 출력하는 단계, j-1번째 서브 라운드 키를 기초로 비트 순환 회전 결정 값을 계산하고, 계산한 비트 순환 회전 결정 값을 토대로 상기 제1 서브 바이트의 한 비트가 순환 회전되거나 상기 제1 서브 바이트의 비트 순서 그대로 하여 순환 처리 서브 바이트를 생성하는 단계, j-2 번째 서브 라운드 키를 구성하는 각 비트들로 계산된 제1 값과 제2 값을 이용하여, 상기 순환 처리 서브 바이트의 비트 위치를 변경하여 비트 재배치 서브 바이트를 생성하는 단계, j-3번째 서브 라운드 키와 상기 비트 재배치 서브 바이트를 배타적 논리 합하여 제2 서브 바이트를 해당 라운드의 라운드 바이트로 출력하는 단계를 포함할 수 있다.
- [0018] 상기 j는 상기 서브 데이터가 상기 데이터 블록에서 몇 번째 바이트에 해당하는지 나타내는 정수이고, 상기 j-0, j-1, j-2, j-3의 계산 값이 0 이하인 경우, 계산된 각 값을 16으로 나눈 나머지 값을 가지는 서브 라운드 키를 사용할 수 있다.
- [0019] 상기 순환 처리 서브 바이트를 생성하는 단계는, 상기 j-1번째 서브 라운드 키를 구성하는 비트들 중 미리 설정된 제1 비트와 제2 비트의 배타적 논리 합의 값이 1이면 상기 제1 서브 바이트를 한 비트씩 좌측으로 순환 회전하여 상기 순환 처리 서브 바이트로 생성하고, 상기 제1 비트와 제2 비트의 배타적 논리 합의 값이 0이면 상기 제1 서브 바이트를 상기 순환 처리 서브 바이트로 생성할 수 있다.
- [0020] 상기 비트 재배치 서브 바이트를 생성하는 단계는, 상기 순환 처리 서브 바이트와 상기 j-2번째 서브 라운드 키를 입력받는 단계, 상기 j-2번째 서브 라운드 키의 제1, 제2, 제3 비트 값의 배타적 논리 합하여 상기 제1 값을 계산하는 단계, 상기 j-2번째 서브 라운드 키의 제4, 제5, 제6 비트 값의 배타적 논리 합하여 상기 제2 값을 계산하는 단계, 그리고 상기 제1 값과 제2 값들에 따라, 상기 순환 처리 서브 바이트의 비트들의 위치를 변경하여 상기 비트 재배치 서브 바이트로 생성하는 단계를 포함할 수 있다.
- [0021] 상기 서브 데이터를 암호화하는 단계 이후에, 암호화된 데이터 블록을 복호화하는 단계를 더 포함하고, 상기 복호화하는 단계는, 암호화된 데이터 블록과 상기 암호화된 데이터 블록의 시작 주소, 그리고 상기 데이터 블록을 암호화할 때 생성한 스캔블러 키를 입력받는 단계, 상기 암호화된 데이터 블록과 상기 스캔블러 키를 기초로 상기 암호화할 횟수만큼 생성된 라운드 키들을 생성하는 단계, 그리고 암호화 라운드의 역순으로 라운드 키들을 상기 복수의 바이트 레인들로 입력하여 상기 암호화된 데이터 블록을 복호화하는 단계를 포함할 수 있다.
- [0022] 상기 복호화하는 단계는, 상기 j-3번째 서브 라운드 키와 암호화된 서브 데이터를 배타적 논리 합하여, 상기 비트 재배치 서브 바이트를 생성하는 단계, 상기 j-2 번째 서브 라운드 키를 구성하는 각 비트들로 계산된 제1 값과 제2 값을 이용하여, 상기 비트 재배치 서브 바이트의 비트 위치를 변경하여, 상기 순환 처리 서브 바이트를 생성하는 단계, 상기 j-1번째 서브 라운드 키를 기초로 비트 순환 회전 결정 값을 계산하고, 계산한 비트 순환

회전 결정 값을 토대로 상기 순환 처리 서브 바이트의 한 비트가 순환 회전되거나 상기 순환 처리 서브 바이트의 비트 순서 그대로 하여 상기 제1 서브 바이트를 생성하는 단계, 상기 j-0번째 서브 라운드 키와 상기 제1 서브 바이트를 배타적 논리 합하여 상기 역 서브바이트 연산된 서브 데이터를 제1 서브 바이트를 출력하는 단계, 상기 비트 재배치 서브 바이트와 상기 비트 재배치 서브 바이트를 배타적 논리 합하여, 상기 역 서브바이트 연산된 서브 데이터를 출력하는 단계, 그리고 상기 인버스된 서브 데이터를 서브바이트 연산하여 상기 서브 데이터를 생성하는 단계를 포함할 수 있다.

- [0023] 상기 순환 처리 서브 바이트를 생성하는 단계는, 상기 j-1번째 서브 라운드 키를 구성하는 비트들 중 미리 설정된 제1 비트와 제2 비트의 배타적 논리 합의 값이 1이면 상기 제1 서브 바이트를 한 비트씩 우측으로 순환 회전하여 상기 순환 처리 서브 바이트로 생성하고, 상기 제1 비트와 제2 비트의 배타적 논리 합의 값이 0이면 상기 제1 서브 바이트를 상기 순환 처리 서브 바이트로 생성할 수 있다.

발명의 효과

- [0024] 본 발명에 따르면 암호화 장치와 복호화 장치의 동작에서 입력 데이터 위치를 변경하는 일이 없기 때문에, 데이터 블록 안의 특정 데이터를 용이하게 수정할 수 있다.
- [0025] 또한, 상용 블록 암호 알고리즘을 사용하는 경우, '읽기-수정-쓰기'의 과정을 거쳐야 하나, 본 발명에 따르면 변경하려는 데이터만 수정할 수 있으므로 상용 알고리즘을 사용하는 경우 보다 더 빠르게 데이터를 수정할 수 있다.
- [0026] 또한, 데이터 버스의 규격(64/128 비트)에 따라 확장/축소가 용이하며, 수행할 라운드 숫자를 조정할 수 있어 성능 향상에 유리하다.

도면의 간단한 설명

- [0027] 도 1은 일반적인 블록 암호화 방법에 대한 예시도이다.
- 도 2는 일반적인 블록 암호화 방법이 적용된 시스템의 구조도이다.
- 도 3은 일반적인 블록 암호화 방법이 적용된 시스템에서 데이터 변경 과정을 나타낸 예시도이다.
- 도 4는 본 발명의 실시예에 따른 암호화 알고리즘이 적용된 시스템의 구조도이다.
- 도 5는 본 발명의 실시예에 따른 키 스케줄러의 예시도이다.
- 도 6은 본 발명의 실시예에 따른 키 생성 모듈의 예시도이다.
- 도 7은 본 발명의 실시예에 따른 비트 선택 모듈의 예시도이다.
- 도 8은 본 발명의 실시예에 따른 주소 비트 변환 모듈의 예시도이다.
- 도 9는 본 발명의 실시예에 따른 하위 키 변환 모듈의 예시도이다.
- 도 10은 본 발명의 실시예에 따른 비트 위치 변경 모듈의 예시도이다.
- 도 11은 본 발명의 실시예에 따른 키 길이 변경 모듈의 예시도이다.
- 도 12 및 도 13은 본 발명의 실시예에 따른 비트 이동 모듈의 예시도이다.
- 도 14는 본 발명의 실시예에 따른 암호/복호화기에서 데이터 블록을 암호화하는 예시도이다.
- 도 15는 본 발명의 실시예에 따른 암호/복호화기에서 암호화된 데이터 블록을 복호화하는 예시도이다.
- 도 16은 본 발명의 실시예에 따른 비트 순환 회전 함수의 연산 방법에 대한 예시도이다.
- 도 17은 본 발명의 실시예에 따른 DERV 함수의 연산 방법에 대한 예시도이다.
- 도 18은 본 발명의 실시예에 따른 암호화 알고리즘이 적용된 시스템의 데이터 암호화 동작 예시도이다.
- 도 19는 본 발명의 실시예에 따른 데이터 변경 예시도이다.

발명을 실시하기 위한 구체적인 내용

- [0028] 아래에서는 첨부한 도면을 참고로 하여 본 발명의 실시예에 대하여 본 발명이 속하는 기술 분야에서 통상의 지

식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 그리고 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다.

- [0029] 명세서 전체에서, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.
- [0030] 이하, 도면을 참조로 하여 내용량 데이터 처리를 위한 암호화 장치 및 방법에 대해 상세히 설명한다.
- [0031] 본 발명의 실시예에 대해 설명하기 앞서, 일반적인 블록 암호화 방법에 대해 도 1 내지 도 3을 참조로 설명한다. 본 발명의 실시예에서는 다양한 블록 암호화 방법 중 AES 암호화 방법을 예로 하여 설명한다.
- [0032] 도 1은 일반적인 블록 암호화 방법에 대한 예시도이다.
- [0033] 상용 블록 암호화 알고리즘은 데이터를 블록 단위로 처리한다. AES 암호화 방식의 경우 입력으로 받는 한 블록의 데이터 크기는 16 바이트(128비트)이다.
- [0034] 도 1에 도시된 바와 같이, 주소 0x0 ~ 0xF에 위치한 128 비트의 데이터와 128 비트의 키로 AES 암호화한 결과도 1의 우측에 나타낸 바와 같다. 예를 들어, 도 1의 128 비트 데이터인 4973207468697320746865207265616C와 128 비트의 키인 139050849F660D3D84E48ADF51672652로 AES 암호화한 경우 암호화된 결과는 128 비트의 31C45DA459EAE1C26B3CFAD6E5975822가 된다.
- [0035] 이때, 블록 암호화 알고리즘은 암호화 품질을 향상시키기 위해 데이터 블록 안에서의 데이터 위치를 변경하는 작업을 수행한다. 이는, 현재 처리하는 데이터와 이전 데이터, 이후 데이터 사이의 종속 관계를 지우는 역할을 하며, 암호화 품질을 높여준다.
- [0036] 이와 같은 블록 암호화 알고리즘이 적용된 시스템과, 해당 시스템에서 데이터를 변경하는 과정에 대해 도 2를 참조로 설명한다.
- [0037] 도 2는 일반적인 블록 암호화 방법이 적용된 시스템의 구조도이고, 도 3은 일반적인 블록 암호화 방법이 적용된 시스템에서 데이터 변경 과정을 나타낸 예시도이다.
- [0038] 도 2에 도시된 바와 같이, 시스템(10)은 암호화된 데이터가 저장되는 메모리(11), 데이터를 암호화하는 암호화 기능과 암호화된 데이터를 복호화하는 복호화 기능을 포함하는 메모리 컨트롤러(12)를 포함한다. 그리고 시스템(10) 사용자가 데이터를 입력하거나 수정할 수 있도록 원본 데이터를 디스플레이하거나 수정된 데이터를 입력으로 받을 수 있도록 하는 응용 소프트웨어(13)를 포함한다.
- [0039] 여기서 메모리(11)에 저장된 암호화된 데이터의 블록에서 특정 부분을 변경하려 한다고 가정한다. 본 발명의 실시예에서는 0x5번 주소에 해당하는 16진수 데이터 69를 16진수 데이터 00으로 변경하려고 한다고 가정하여 설명하며, 이는 도 3의 음영으로 표시된 부분에 해당한다.
- [0040] 그러면 시스템(10)은 메모리(11)에서 메모리 컨트롤러로 암호화된 데이터를 읽은 메모리 컨트롤러(12)의 복호화 기능을 통해 암호화된 데이터를 복호화하여 응용 소프트웨어(13)로 전달한다(①). 사용자가 응용 소프트웨어(13)를 통해 0x5번 주소의 16진수 데이터 69를 16진수 데이터 00으로 수정한다(②).
- [0041] 시스템(10)은 사용자에게 의해 수정된 데이터를 메모리 컨트롤러(12)의 암호화 기능을 이용하여 암호화하고, 메모리(11)에 수정 후 암호화된 데이터로 저장한다(③). 이때, 최초 메모리(11)에 저장되어 있던 암호화된 데이터와 데이터가 수정된 후 암호화된 데이터를 비교하면, 0x5번 주소의 16진수 데이터 변경에 따라 블록 내 모든 주소의 암호화된 데이터들이 변경된 것을 알 수 있다.
- [0042] 즉, 변경하고자 했던 데이터는 주소 0x5의 1 바이트 16진수 데이터 '69' 였다. 그러나, 블록 암호화 방식의 특성 상 데이터를 블록 단위로 처리해야 하는 제약 때문에 1 블록을 모두 읽어 복호화(읽기) 한 뒤, '69'를 변경하려는 16진수 데이터 '00'으로 변경하고, 이 블록을 암호화하여 메모리(11)에 저장(쓰기)해야 하는 번거로움이 있다.
- [0043] 이는 암호/복호화 알고리즘으로 블록 암호 알고리즘을 사용하는 경우 반드시 발생하는 문제로, 암호/복호화에 시간이 많이 걸리거나, 메모리 접근 시간이 길어지는 경우 전체 시스템 성능이 크게 하락할 수 있다. 따라서, 본 발명의 실시예에서는 블록 단위로 메모리에 데이터를 암호화하여 저장할 수 있으면서도, 기존 블록 암호화 방식에

서 발생하는 '읽기-수정-쓰기' 절차와 달리 '쓰기' 만으로 메모리 내용을 변경할 수 있는 암호화 장치 및 방법을 제안한다.

- [0044] 본 발명의 실시예에 대한 설명에 앞서, 설명의 편의를 위해 사용되는 비트의 표현 방식에 대해 다음과 같이 정의한다.
- [0045] - A[M]: A의 M번째 비트
- [0046] - A[M:N]: A의 M번째 비트부터 N 번째 비트
- [0047] 예를 들어, A[7:0] = 0x96일 때, A[7]=1, A[6]=0, A[5]=0, A[4]=1, A[3]=0, A[2]=1, A[1]=1, A[0]=0임.
- [0048] - 바이트 배열 D[j]: 바이트 배열 D의 j 번째 바이트
- [0049] 예를 들어, 8 바이트 배열 D의 값이 D=0x1112131415161718 일 때, D[0]=0x11, D[1]=0x12, D[2]=0x13, D[3]=0x14, D[4]=0x15, D[5]=0x16, D[6]=0x17, D[7]=0x18
- [0050] - 1 바이트는 1 주소를 차지함
- [0051] 예를 들어, D = 0x1112131415161718212223242526272831 일 때, D[0]의 값인 0x11이 메모리 주소 0x100에 저장되어 있고, D[1]의 값인 0x12가 메모리 주소 0x101에 저장되어 있다. 그러나 일반적으로 4바이트를 한꺼번에 표시하기 때문에 D[0:3]인 0x11121314가 메모리 주소 0x100에 저장되는 것으로 예를 들어 설명한다.
- [0052] 도 4는 본 발명의 실시예에 따른 암호화 알고리즘이 적용된 시스템의 구조도이다.
- [0053] 도 4에 도시된 바와 같이, 본 발명의 실시예에 따른 암호화 알고리즘이 적용된 시스템은 난수 발생기(100), 키 스케줄러(200), 암/복호화기(300)로 구현된다. 본 발명의 실시예에서는 설명의 편의를 위하여 도 4의 (a)에는 데이터 블록을 암호화하는 과정을 나타내었고, 도 4의 (b)에는 암호화된 데이터 블록을 복호화하는 과정을 나타내었다. 그러나 두 절차 모두 동일한 시스템으로 수행된다.
- [0054] 난수 발생기(100)는 64 비트 또는 128 비트의 스크램블러 키(SK0)를 생성한다. 난수 발생기(100)가 스크램블러 키를 생성하는 방법과 생성된 키의 형태는 다양하므로 본 발명의 실시예에서는 어느 하나의 방법으로 한정하지 않는다.
- [0055] 키 스케줄러(200)는 난수 발생기(100)에서 생성한 스크램블러 키와, 암호화하고자 하는 데이터들로 구성된 하나의 블록(이하, 설명의 편의를 위하여 데이터 블록이라 지칭함)의 시작 주소(A)를 입력으로 받는다. 이때, 데이터 블록에 포함된 데이터들의 길이는 난수 발생기(100)에서 생성한 스크램블러 키의 길이와 동일하다.
- [0056] 여기서, 난수 발생기(100)에서 생성한 스크램블러 키는 메모리(도면 미도시)에 저장된다. 이는 생성한 스크램블러 키를 이용하여 데이터를 암호화하거나 복호화 할 때 동일하게 사용하기 위함이다.
- [0057] 본 발명의 실시예에서는 데이터 블록을 암호화하거나 암호화된 데이터 블록을 복호화하는 데 전체 8 라운드에 걸쳐 데이터를 처리하는 것을 예로 하여 설명한다. 따라서, 키 스케줄러(200)는 각 라운드에서 암호화 또는 복호화하기 위해 사용할 라운드 키들(RK₀ ~ RK₇)을 생성한다. 이때, 키 스케줄러(200)는 임의의 라운드에서 사용할 라운드 키는 이전 라운드에서 생성한 라운드 키를 기초로 생성하며, 스크램블러 키와 함께 메모리에 저장된다. 이에 대해서 이후 상세히 설명한다.
- [0058] 암/복호화기(300)는 복수의 스크램블러들로 구성된다. 스크램블러의 수는 암호화 또는 복호화하기 위하여 데이터를 처리하는 라운드 수만큼 구비되어 있다. 본 발명의 실시예에서는 8 라운드에 걸쳐 데이터를 처리하기 때문에, 스크램블러의 수도 8개 구비되는 것을 예로 하여 설명한다.
- [0059] 암/복호화기(300)의 첫 번째 스크램블러는 데이터 블록 또는 암호화된 데이터 블록을 입력으로 받는다. 그리고 키 스케줄러(200)에서 생성한 제1 라운드 키 또는 제8 라운드 키를 입력으로 받는다.
- [0060] 암호화는 경우라면 암/복호화기(300)의 첫 번째 스크램블러는 제1 라운드 키와 데이터 블록으로 암호화 하여, 제1 라운드 데이터 블록(RD₁)을 생성한다. 생성한 제1 라운드 데이터 블록은 두 번째 스크램블러로 입력된다.
- [0061] 두 번째 스크램블러는 제2 라운드 키를 이용하여 제1 라운드 데이터 블록을 암호화하여 제2 라운드 데이터 블록(RD₂)을 생성한다. 이와 같은 방식으로 전체 8 라운드의 암호화 절차를 거쳐 생성된 제8 라운드 데이터 블록이 암호화 데이터 블록으로 메모리(10)에 저장된다.

- [0062] 이상에서 설명한 암호화 알고리즘이 적용된 시스템의 데이터 처리 과정을 간단하게 요약하면, 데이터 암호화를 위해 사용할 라운드 수 즉, 데이터를 암호화할 횟수만큼의 라운드 키들을 생성한다. 이때, 라운드 키들은 난수 발생기에서 최초 생성한 스캐램블러 키와 데이터 블록의 시작 주소를 이용하여 생성된다.
- [0063] 라운드 키들이 생성되면, 암/복호화기는 평문의 데이터 블록을 입력으로 받아, 데이터 블록에 포함된 비트들을 바이트 단위로 분리하여 바이트 레인(byte lane)으로 입력한다. 이때, 바이트들의 위치를 바꿔 바이트 레인으로 입력하는 것이 아니라, 순서대로 입력한다. 바이트 레인은 스캐램블러 키의 길이에 따라 16개 바이트 레인 또는 8개의 바이트 레인으로 변경될 수 있으나, 어느 하나의 형태로 한정하지 않는다.
- [0064] 암/복호화기는 바이트 레인 각각에 키 스케줄러(200)에서 생성한 라운드 키를 바이트 단위로 입력하여 바이트 단위의 데이터를 암호화 한다. 전체 라운드를 거쳐 암호화된 데이터 블록은 메모리에 저장된다. 암/복호화기가 암호화된 데이터 블록을 복호화하는 방법은 암호화 방식의 역순으로 진행되며, 이미 라운드 키들이 생성되어 있기 때문에 라운드 키가 생성된 역순으로 암/복호화기에 입력하여 복호화된다.
- [0065] 여기서, 난수 발생기(100)에서 생성된 스캐램블러 키와 데이터 블록의 시작 주소를 이용하여 라운드 키를 생성하는 키 스케줄러(200)에 대해 도 5를 참조로 설명한다.
- [0066] 도 5는 본 발명의 실시예에 따른 키 스케줄러의 예시도이다.
- [0067] 도 5에 도시된 바와 같이, 키 스케줄러(200)는 암호화 또는 복호화하는 라운드 수와 동일한 수의 키 생성 모듈(210)들로 구성된다. 본 발명의 실시예에서는 시스템에서 데이터 블록이나 암호화된 데이터 블록을 8 라운드로 암호화/복호화를 수행하는 것을 예로 하여 설명하므로, 키 생성 모듈(210) 또한 8개 구비되는 것을 예로 하여 설명한다.
- [0068] 첫 번째 키 생성 모듈은 난수 발생기(100)에서 생성한 스캐램블러 키를 그대로 수신한다. 그리고 첫 번째 키 생성 모듈은 데이터 블록의 시작 주소를 입력으로 받는다.
- [0069] 첫 번째 키 생성 모듈은 첫 번째 스캐램블러 키(SK_0)와 시작 주소를 이용하여 제1 라운드 키(RK_0)를 생성한다. 그리고 첫 번째 키 생성 모듈은 첫 번째 스캐램블러 키(SK_0)를 이용하여 두 번째 스캐램블러 키(SK_1)를 생성한다.
- [0070] 두 번째 키 생성 모듈은 첫 번째 키 생성 모듈에서 생성한 두 번째 스캐램블러 키(SK_1)와 시작 주소를 이용하여 제2 라운드 키(RK_1)를 생성한다. 이와 같은 방법으로 스캐램블러 키와 시작 주소를 이용하여 각 라운드의 라운드 키를 생성하는 키 생성 모듈에 대해 도 6을 참조로 설명한다.
- [0071] 도 6은 본 발명의 실시예에 따른 키 생성 모듈의 예시도이다.
- [0072] 도 6에 도시된 바와 같이, 키 생성 모듈(210)은 비트 선택 모듈(211), 주소 비트 변환 모듈(212), 제1 비트 결합 모듈(213), 제1 하위 키 변환 모듈(214), 제1 비트 위치 변경 모듈(215), 제2 하위 키 변환 모듈(216), 제2 비트 위치 변경 모듈(217), 키 길이 변경 모듈(218), 제2 비트 결합 모듈(219), 그리고 비트 이동 모듈(220)을 포함한다.
- [0073] 본 발명의 실시예에서는 설명의 편의를 위하여 각 구성들을 "모듈"로 설명하나, 소프트웨어로 구현된 함수들이 각 기능들을 수행할 수 있다. 그리고, 각 모듈에서 출력된 키들의 명칭을 구분의 편의를 위하여 각각 상이한 키 이름을 부여하였으나, 반드시 이와 같이 한정되는 것은 아니다.
- [0074] 키 생성 모듈(210)이 P번째 라운드의 키 생성 모듈로부터 P번째 스캐램블러 키(SK_P)를 수신하면 비트 선택 모듈(211)은 수신한 P번째 스캐램블러 키를 구성하는 비트들 중 사전에 결정된 위치의 비트들을 선택하여 선택된 선택 비트 키(SK)로 생성한다. 여기서 선택 비트 키(SK)는 주소 비트 결합 모듈(212)의 출력인 비트 변환 주소(TA)와 같이 제1 비트 결합 모듈의 입력으로 전달되어, 배타적 논리 합(XOR)으로 결합된다.
- [0075] 본 발명의 실시예에서는 P번째 스캐램블러 키(SK_P)가 64 비트 또는 128 비트로 구성되어 있는 것을 예로 하여 설명한다. 그리고 비트 선택 모듈(211)에서 출력된 선택 비트 키(SK)는 32 비트의 데이터들로 구성되어 있는 것을 예로 하여 설명한다.
- [0076] 여기서, 비트 선택 모듈(211)은 64 비트 또는 128 비트로 구성된 P번째 스캐램블러 키에서, 사전에 미리 결정된 비트 위치의 데이터들이 선택되는 것을 예로 하여 설명한다.

- [0077] 또한, 비트 선택 모듈(211)은 데이터를 암호화할 때나 복호화할 때 동일하게 사용된다. 데이터를 암호화할 때 임의의 라운드에서 선택한 비트 위치를 알고 있어야 복호화할 때도 역으로 사용할 수 있으므로, 본 발명의 실시예에서는 키 생성 모듈(210)이 생성하는 키로 데이터를 암호화하거나 복호화할 때 선택되는 라운드별 각 비트 위치를 미리 알고 있다고 가정한다.
- [0078] 주소 비트 변환 모듈(212)은 32 비트의 시작 주소를 입력으로 받아, 32 비트의 변환된 주소(TA)를 출력한다. 이때, 주소 비트 변환 모듈(212)은 난수 발생기(100)가 생성한 스캐램블러 키의 비트 수에 따라 32 비트의 시작 주소를 미리 정해진 규칙에 따라 변환하거나 무작위로 변환하여 변환된 주소로 출력한다. 이에 대해서는 이후 상세히 설명한다.
- [0079] 제1 비트 결합 모듈(213)은 비트 선택 모듈(211)에서 출력된 32 비트의 선택 비트(SK)와 주소 비트 변환 모듈(212)에서 출력된 변환된 32 비트의 변환된 주소를 입력으로 각각 받아 배타적 논리 합을 계산하여 32 비트의 결합 키로 생성한다. 본 발명의 실시예에서는 제1 비트 결합 모듈(213)이 선택 비트 키(SK)와 변환된 주소를 배타적 논리 합 하여 결합 키로 생성하는 것을 예로 하여 설명하나, 반드시 이와 같이 한정되는 것은 아니다.
- [0080] 제1 하위 키 변환 모듈(214)은 제1 비트 결합 모듈(213)로부터 출력된 32 비트의 결합 키를 입력으로 받아, 1 바이트씩 묶어 4개의 하위 키들로 생성한다. 그리고 1 바이트씩 묶인 4개의 하위 키들을 역 서브바이트(InverseSubByte) 연산하여 32 비트의 제1 인버스 키로 생성한다.
- [0081] 제1 비트 위치 변경 모듈(215)은 제1 하위 키 변환 모듈(214)에서 출력된 32 비트의 제1 인버스 키를 입력으로 받아, 32 비트들의 위치를 변경하여 제1 비트 위치 변경 키로 생성한다.
- [0082] 제2 하위 키 변환 모듈(216)은 제1 비트 위치 변경 모듈(215)에서 비트 위치가 변경된 제1 비트 위치 변경 키를 입력으로 받아, 4 바이트씩 묶어 4개의 하위 키들로 생성한다. 그리고 4개의 하위 키들을 역 서브바이트 연산하여 32 비트의 제2 인버스 키로 생성한다.
- [0083] 제2 비트 위치 변경 모듈(217)은 제2 하위 키 변환 모듈(216)에서 출력된 32 비트의 제2 인버스 키를 입력으로 받아, 32 비트들의 위치를 변경하여 제2 비트 위치 변경 키로 생성한다.
- [0084] 키 길이 변경 모듈(218)은 제2 비트 위치 변경 모듈(217)에서 생성한 32 비트의 제2 비트 위치 변경 키를 입력으로 받아, 제2 비트 위치 변경 키의 길이를 변경하여 길이 변경 키로 생성한다. 이때, 키 길이 변경 모듈(218)은 난수 발생기(100)에서 최초 생성한 스캐램블러 키의 키 길이에 따라 제2 비트 위치 변경 키의 길이를 64 비트 또는 128 비트 중 어느 하나의 길이로 변경하여 길이 변경 키로 생성한다. 이에 대해서는 이후 상세히 설명한다.
- [0085] 제2 비트 결합 모듈(219)은 키 길이 변경 모듈(218)에서 출력한 길이 변경 키와 키 생성 모듈(210)이 수신한 P 번째 스캐램블러 키를 이용하여 배타적 논리 합을 계산하여, 스캐램블러 키 길이를 가지는 라운드 키(RK_P)로 생성한다. 제2 비트 결합 모듈(219)이 생성한 라운드 키(RK_P)는 P 번째 라운드의 스캐램블러로 입력된다.
- [0086] 비트 이동 모듈(220)은 키 생성 모듈(210)이 수신한 P 번째 스캐램블러 키를 구성하는 비트들의 위치를 변경하여, P+1 번째 스캐램블러 키(SK_{P+1})로 생성한다. 이에 대해서는 이후 상세히 설명한다.
- [0087] 이상에서 설명한 키 생성 모듈(210)을 구성하는 각 모듈(또는 각 함수)이 키를 생성하는 예에 대해 도 7 내지 도 11을 참조로 설명한다.
- [0088] 도 7은 본 발명의 실시예에 따른 비트 선택 모듈의 예시도이다.
- [0089] 도 7에 도시된 바와 같이, 비트 선택 모듈(211)은 스캐램블러 키의 길이에 따라 도 7의 (a)와 같이 128 비트의 스캐램블러 키를 32 비트의 선택 비트 키로 생성하거나, 도 7의 (b)와 같이 64 비트의 스캐램블러 키를 32 비트의 선택 비트 키로 생성할 수 있다.
- [0090] 비트 선택 모듈(211)은 128 비트들 각각에 삽입된 데이터들 중 미리 설정된 위치의 32 개의 비트들을 선택하여 선택 비트 키로 생성한다. 여기서 비트 선택 모듈(211)이 선택한 비트들을 어느 하나로 한정하지 않는다.
- [0091] 그리고, 0번 비트부터 127번 비트, 또는 0번 비트부터 63번 비트를 내림차순 또는 오름차순으로 순서대로 선택할 수도 있고, 무작위 순서로 선택할 수도 있다. 이때, 비트 선택 모듈(211)은 이후 암호화된 데이터를 복호화할 때 선택 비트 키를 생성하기 위하여 어떤 비트들이 선택되었는지 사전에 정의되었다고 가정한다.
- [0092] 예를 들어 설명하면, 64 비트 스캐램블러 키를 사용한다고 가정하면, 비트 선택 모듈(211)은 다음 수학적 식 1에

나타낸 바와 같은 비트들로 선택 비트 키를 생성할 수 있다.

수학식 1

$$SK[31:0] = \{K[20], K[19], K[28], K[43], K[4], K[8], K[7], K[14], K[27], K[12], K[63], K[5], K[59], K[39], K[38], K[21], K[55], K[41], K[57], K[16], K[54], K[6], K[50], K[13], K[53], K[58], K[37], K[34], K[31], K[29], K[42], K[47]\}$$

[0093]

[0094]

수학식 1을 설명하면, 64 비트 스크램블러 키의 20번째 비트를 선택 비트 키의 31번째 비트에 위치시킨다는 것을 의미한다. 수학식 1에 나타낸 선택 비트 키 생성 방식은 하나의 예로, 반드시 이와 같이 한정되는 것은 아니다.

[0095]

도 8은 본 발명의 실시예에 따른 주소 비트 변환 모듈의 예시도이다.

[0096]

도 8에 도시된 바와 같이 주소 비트 변환 모듈(212)은, 32 비트의 시작 주소를 입력으로 받아, 32 비트의 변환된 주소(TA)를 출력한다. 이때, 주소 비트 변환 모듈(212)은 난수 발생기(100)가 생성한 스크램블러 키가 128 비트이지 또는 64 비트인지에 따라 미리 정해진 규칙에 따라 32 비트의 변환된 주소를 출력한다.

[0097]

예를 들어, 스크램블러 키가 128 비트라고 가정한다. 그러면 주소 비트 변환 모듈(212)은 시작 주소의 32 비트들의 비트 위치를 미리 설정된 규칙 또는 무작위로 바꿔 변환된 주소(TA)로 생성한다.

[0098]

만약 스크램블러 키가 64 비트라고 가정하면, 주소 비트 변환 모듈(212)은 시작 주소의 하위 3개의 비트인 A[2], A[1], 및 A[0]의 비트는 삭제하고, 하위 4번째 비트인 A[3] 비트를 동일하게 두 개 생성하고, A[3] 비트의 인버스된 비트인 $A^{-1}[3]$ 비트를 두 개 생성한다.

[0099]

그리고 하위 4개 비트를 제외한 나머지 비트 A[31:4]에, 생성된 두 개의 A[3] 비트와 인버스된 두 개의 비트인 $A^{-1}[3]$ 비트들을 이용하여, 32 비트의 주소 변환 비트를 생성한다. 이때, 주소 비트 변환 모듈(212)은 32 비트(A[31:4], A[3], A[3], A-1[3], A-1[3])들의 비트 위치를 미리 설정된 규칙 또는 랜덤하게 바꿔 변환된 주소(TA)로 생성한다.

[0100]

여기서, 스크램블러 키가 64 비트일 경우 하위 4번째 비트인 A[3] 비트를 사용하는 이유는, 스크램블러 키가 64 비트일 경우 시작 주소는 0x0, 0x8, 0x10, 0x18이 된다. 이를 2진수로 표현하면 00000000, 00001000, 00010000, 00011000 등으로 표현되며 하위 4번째 비트인 A[3]이 0과 1로 변경되며 중요한 정보로써 사용되기 때문이다.

[0101]

도 9는 본 발명의 실시예에 따른 하위 키 변환 모듈의 예시도이다.

[0102]

하위 키 변환 모듈은 제1 하위 키 변환 모듈(214)과 제2 하위 키 변환 모듈(216)이 동일한 형태로 구현된다.

[0103]

도 9에 도시된 바와 같이, 하위 키 변환 모듈은 입력으로 32 비트의 결합 키 또는 32 비트의 제1 비트 위치 변경 키를 수신하면, 32 비트의 키를 1 바이트의 하위 키가 포함되도록 4개의 하위 키들로 그룹핑한다. 그리고, 그룹핑한 1 바이트의 4개 하위 키들을 역 서브바이트 연산하여 제1 인버스 키, 또는 제2 인버스 키로 출력한다. 역 서브바이트 연산 방법은 이미 알려진 것으로, 본 발명의 실시예에서는 상세한 설명을 생략한다.

[0104]

도 10은 본 발명의 실시예에 따른 비트 위치 변경 모듈의 예시도이다.

[0105]

비트 위치 변경 모듈은 제1 비트 위치 변경 모듈(215)과 제2 비트 위치 변경 모듈(217)이 동일한 형태로 구현된다.

[0106]

도 10에 도시된 바와 같이, 비트 위치 변경 모듈이 32 비트의 제1 인버스 키 또는 32 비트의 제2 인버스 키를 입력으로 받으면, 32 비트에 각각 삽입된 데이터들의 위치를 변경한다. 이를 본 발명의 실시예에서는 비트 위치를 변경한다고 지칭한다.

[0107]

즉, 도 10에 나타낸 바와 같이, IN[31:0]의 제1 인버스 키 또는 제2 인버스 키가 입력되면, 비트 위치 변경 모듈은 해당 인버스 키의 비트들의 순서를 {IN[31], IN[27], IN[23], IN[19], IN[15], ..., IN[4], IN[0]}으로 바꾸어 출력한다. 예를 들어, 입력으로 0x81726354를 수신하면, 출력으로 0x802B3C66이 생성된다.

- [0108] 도 11은 본 발명의 실시예에 따른 키 길이 변경 모듈의 예시도이다.
- [0109] 도 11에 도시된 바와 같이 키 길이 변경 모듈(218)은 사용하는 스크램블러 키의 길이(64 비트 또는 128 비트)에 따라 도 11의 (a) 및 (b)와 같이 키 길이 변경 모듈을 분류하여 사용한다.
- [0110] 도 11의 (a)에 도시된 바와 같이, 스크램블러 키의 길이가 128 비트인 경우, 키 길이 변경 모듈은 32 비트의 제 2 비트 위치 변경 키를 4번 이어 붙여 128 비트의 길이 변경 키로 생성한다. 그리고 도 11의 (b)와 같이 스크램블러 키의 길이가 64 비트인 경우, 키 길이 변경 모듈은 32 비트의 제2 비트 위치 변경 키를 2번 이어 붙여 128 비트의 길이 변경 키로 생성한다.
- [0111] 예를 들어 스크램블러 키의 길이가 128 비트인 경우, 키 길이 변경 모듈이 0x81726354를 입력으로 수신하면, 출력인 길이 변경 키는 0x81726354817263548172635481726354가 된다. 마찬가지로 스크램블러 키의 길이가 64 비트인 경우, 키 길이 변경 모듈이 0x81726354를 입력으로 수신하면, 출력인 길이 변경 키는 0x8172635481726354가 된다.
- [0112] 다음은 비트 이동 모듈(220)에 대해 도 12 및 도 13을 참조로 설명한다.
- [0113] 도 12 및 도 13은 본 발명의 실시예에 따른 비트 이동 모듈의 예시도이다.
- [0114] 본 발명의 실시예에서는 P번째 스크램블러 키를 입력으로 수신하여 P+1번째 스크램블러 키로 생성하는 비트 이동 모듈(220)을 도 12 또는 도 13에 도시된 비트 이동 모듈 중 어느 하나의 비트 이동 모듈을 사용하는 것을 예로 하여 설명한다.
- [0115] P번째 스크램블러 키(SK_p)와 데이터 블록의 시작 주소(A)를 입력으로 받아 키 길이 변경 모듈까지 수행하여 얻은 결과, 그리고 P번째 스크램블러 키(SK_p)를 가지고 배타적 논리 합을 계산하여 P번째 라운드 키(RK_p)를 얻은 후에, 다음 라운드(P+1번째 라운드)를 위해 P번째 스크램블러 키(SK_p)를 다음 라운드로 넘겨줘야 한다. 이때, 인접 라운드 간의 연관성을 지우기 위해 비트 이동 모듈을 사용한다.
- [0116] 본 발명의 실시예에서는 도 12에 도시된 바와 같이 P번째 스크램블러 키(SK_p)의 상위 8 비트만 역 서브바이트 연산을 수행한다. 그 후, 왼쪽으로 3 비트 순환 회전(ROT_L3: ROTate Left 3 bit)하여, P+1번째 스크램블러 키(SK_{p+1})를 생성한다.
- [0117] 이때, 빠른 연산을 위하여, 도 13에 도시된 바와 같이 역 서브바이트 연산 없이 왼쪽으로 3 비트 순환 회전(ROT_L3)만 한 후 P+1번째 스크램블러 키(SK_{p+1})를 생성할 수도 있다.
- [0118] 다음은, 상기 도 4에서 키 스케줄러(200)에서 출력된 8개의 라운드 키들과 데이터 블록을 이용하여 암호/복호화기(300)에서 암호화하고 복호화하는 예에 대해 도 14 및 도 15를 참조로 설명한다.
- [0119] 도 14는 본 발명의 실시예에 따른 암호/복호화기에서 데이터 블록을 암호화하는 예시도이고, 도 15는 본 발명의 실시예에 따른 암호/복호화기에서 암호화된 데이터 블록을 복호화하는 예시도이다.
- [0120] 도 14 및 도 15에서는 128 비트의 입력 데이터와 라운드 키를 사용하는 경우의 데이터 블록의 암호화와 암호화된 데이터 블록의 복호화를 나타낸 것이다. 그리고, 본 발명의 실시예에서는 전체 8 라운드의 데이터 처리 과정 중 임의의 라운드인 P 라운드의 데이터 처리 과정을 실시예로 나타내었다.
- [0121] 도 14에 도시된 바와 같이, 128 비트의 데이터 블록(RD_p)과 라운드 키(RK_p)가 입력되면, 입력된 데이터 블록을 바이트 단위로 묶어 각각 바이트 레인(byte lane)에 입력한다.
- [0122] 여기서, 바이트 레인이라 함은, 블록 단위로 입력된 데이터들을 바이트 단위로 각각 암호화하거나 복호화하는 기능들의 집합을 의미한다. 만약 데이터 블록이 128 비트라면 16개의 바이트 레인이 형성된다. 그리고 64 비트의 데이터 블록이 입력되면, 8개의 바이트 레인이 형성된다.
- [0123] 기존 블록 단위의 암호/복호화 알고리즘을 사용하는 시스템들은 입력 데이터들이 8 바이트 또는 16 바이트가 모두 채워지지 않을 경우 암호/복호화 시 오류가 발생할 수 있기 때문에, 데이터가 입력되지 않은 비트에 특정 비트(예를 들어, 0)를 채워 넣는다거나 패딩 비트(padding bit)를 채워 넣은 데이터 블록을 입력으로 사용하였다. 이는 블록 암호화 방식의 특성 상 원문 데이터의 길이는 블록 길이의 배수가 되어야 하기 때문이다.
- [0124] 그러나, 본 발명의 실시예에서는 블록 단위로 입력 데이터를 수신하더라도, 이를 바이트 단위로 바이트 레인에

서 암호화 처리하기 때문에, 입력된 데이터의 위치와 동일한 위치로 암호화된 데이터가 출력된다. 따라서, 데이터가 입력되지 않은 바이트 레인의 경우, 해당 바이트 레인은 데이터의 암호화나 복호화를 수행하지 않기 때문에, 특정 비트나 패딩 비트로 채워 넣지 않아도 무방하다.

- [0125] 본 발명의 실시예에서는 하나의 바이트 레인에 하나의 역 서브바이트 연산 함수 처리 모듈, 제1 배타적 논리 합 계산 함수 처리 모듈, 비트 순환 함수(ROTL/ROTR) 처리 모듈, 서브 바이트 비트 재배치 모듈, 제2 배타적 논리 합 계산 함수 처리 모듈을 포함하는 것을 예로 하여 설명한다. 본 발명의 실시예에서는 암호/복호화기를 구성하는 바이트 레인 역시 설명의 편의를 위하여 모듈로 표현하였으나, 소프트웨어로 구현된 프로그램이 바이트 레인의 기능을 수행할 수 있다.
- [0126] 제1 배타적 논리 합 계산 함수 처리 모듈로 입력되는 바이트 단위로 구분된 라운드 키(이하, 설명의 편의를 위하여 '서브 라운드 키'라 지칭함)는 $j-0$ 번째 서브 라운드 키가 입력된다. 여기서 j 는 바이트 단위로 구분한 라운드 입력 데이터(이하, '서브 입력 데이터'라 지칭함)의 서브 입력 데이터 순서를 의미한다.
- [0127] 예를 들어, 바이트 레인에 $RD_b[0]$ 의 서브 입력 데이터가 입력되었다고 가정하면 j 는 0이 된다. 하나의 서브 입력 데이터는 8 바이트인 것을 예로 하여 설명하므로, j 는 0에서 7까지의 수 중 어느 하나의 수가 된다.
- [0128] 그리고, 비트 순환 함수 처리 모듈로는 $j-1$ 번째 서브 라운드 키가 입력되고, DREV 함수 처리 모듈로는 $j-2$ 번째 서브 라운드 키가 입력된다. 그리고 제2 배타적 논리 합 계산 함수 처리 모듈로는 $j-3$ 번째 서브 라운드 키가 입력된다.
- [0129] 여기서, $j-0$, $j-1$, $j-2$, $j-3$ 의 값이 0보다 작을 경우, 16으로 나눈 나머지 값을 사용한다. 예를 들어, $j=0$ 인 경우 $j-0$ 은 0으로 0과 같고, $j-1$ 은 -1, $j-2$ 는 -2, $j-3$ 은 -3으로 0보다 작기 때문에, 각 값인 0, -1, -2, -3을 16으로 나눈 나머지 값인 $0(0 \bmod 16)$, $15(-1 \bmod 16)$, $14(-2 \bmod 16)$, $13(-3 \bmod 16)$ 을 이용한다. 이때의 값 0은 0번째 서브 라운드 키를, 15는 15번째 서브 라운드 키를 의미하며, 도 14의 바이트 레인으로 표시된 부분의 나타낸 것과 같다.
- [0130] 도 14에 나타낸 복수의 바이트 레인들 중 하나의 바이트 레인을 예로 하여 이어 설명하면, 0번째 서브 입력 데이터 $RD_b[0]$ 를 입력으로 받은 바이트 레인은, 서브 입력 데이터에 역 서브바이트 연산을 수행하여 역 서브바이트 연산된 서브 입력 데이터를 생성한다.
- [0131] 0번째 서브 입력 데이터의 역 서브바이트 연산된 서브 입력 데이터와, 0번째 서브 라운드 키를 배타적 논리 합을 계산하여, 제1 서브 바이트를 출력한다.
- [0132] 15번째 서브 라운드 키와 제1 서브 바이트를 입력으로 받은 비트 순환 함수 처리 모듈은 입력으로 받은 15번째 서브 라운드 키에서 N 값을 계산하고, 계산된 N 값을 토대로 제1 서브 출력을 순환 회전하거나 순환 회전하지 않은 채, 순환 처리 서브 바이트를 생성한다. 여기서, N 은 비트 순환 회전 결정 값이라 지칭하며, 이에 대해서는 이후 설명한다.
- [0133] 비트 순환 함수 처리 모듈에서 출력된 순환 처리 서브 바이트와 14번째 서브 라운드 키를 입력으로 받은 서브 바이트 비트 재배치 모듈은, 14번째 서브 라운드 키의 각 비트로 계산된 제1 값과 제2 값을 이용하여 순환 처리 서브 바이트의 비트 위치를 변경하여, 비트 재배치 서브 바이트로 생성한다.
- [0134] 비트 재배치 서브 바이트와 13번째 서브 라운드 키($KP[13]$)를 입력으로 받은 제2 배타적 논리 합 함수 처리 모듈은 비트 재배치 서브 바이트와 13번째 서브 라운드 키의 배타적 논리 합을 계산하여, 제2 서브 바이트를 생성한다. 이렇게 생성된 제2 서브 바이트가 해당 라운드의 라운드 바이트 출력이 된다.
- [0135] 도 15에 나타낸 복호화는 상기 도 14의 암호화에서 수행하는 연산의 순서를 반대로 하여 데이터를 처리한다. 즉, 13번째 서브 라운드 키($KP[13]$)와 암호화된 서브 데이터를 배타적 논리 합하여, 비트 재배치 서브 바이트를 생성한다. 그리고, 14번째 서브 라운드 키를 구성하는 각 비트들로 계산된 제1 값과 제2 값을 이용하여, 비트 재배치 서브 바이트의 비트 위치를 변경하여, 순환 처리 서브 바이트를 생성한다.
- [0136] 15번째 서브 라운드 키를 기초로 비트 순환 회전 결정 값을 계산하고, 계산한 비트 순환 회전 결정 값을 토대로 순환 처리 서브 바이트의 한 비트가 순환 회전되거나, 순환 처리 서브 바이트의 비트 순서 그대로 하여 제1 서브 바이트를 생성한다. 0번째 서브 라운드 키와 제1 서브 바이트를 배타적 논리 합하여 역 서브바이트 연산된 서브 데이터를 생성한다. 그리고 가장 마지막으로 수행하는 SB(SubByte) 연산은, 기존 역 서브바이트 연산된 서브 입력 데이터를 다시 서브바이트 연산하여, 최초 입력된 서브 입력 데이터로 생성한다는 것을 의미한다. 서

브ایت 연산은 이미 알려진 것으로, 본 발명의 실시예에서는 상세한 설명을 생략한다.

[0137] 여기서, 비트 순환 회전 함수가 수행되는 예에 대해 도 16을 참조로 설명한다.

[0138] 도 16은 본 발명의 실시예에 따른 비트 순환 회전 함수의 연산 방법에 대한 예시도이다.

[0139] 도 16의 (a)는 좌측으로 비트를 순환 회전시키는 함수(이하, 설명의 편의를 위하여 '레프트 쉬프트 함수'라 지칭함)를 이용한 연산 방법이고, 도 16의 (b)는 우측으로 비트를 순환 회전시키는 함수(이하, 설명의 편의를 위하여 '라이트 쉬프트 함수'라 지칭함)를 이용한 연산 방법의 예시도이다.

[0140] 도 16에 도시된 바와 같이, 좌측으로 비트를 순환 회전시키는 함수는 1바이트의 서브 라운드 키 $K_p[j-1]$ 를 입력으로 받아, 비트 순환 회전 결정 값 N 을 계산하는데, 다음 수학식 2와 같이 계산한다.

수학식 2

$$N = k_5 \oplus k_4$$

[0141]

[0142] 여기서, K_5 와 K_4 는 각각 1바이트의 서브 라운드 키 $K_p[j-1]$ 에 포함된 비트들 $\{K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0\}$ 중에서, 4번째 비트(K_4)와 5번째 비트(K_5)의 값을 의미한다.

[0143] 수학식 2를 기초로 계산한 비트 순환 회전 결정 값 N 을 토대로, 레프트 쉬프트 함수는 왼쪽으로 1 비트 순환 회전하거나, 회전하지 않는다. 즉, N 값이 1인 경우 왼쪽으로 1 비트씩 순환회전 하여, 서브 입력 데이터인 $IN[7:0]$ 을 $\{IN[6:0], IN[7]\}$ 로 순환 회전하여 순환 처리 서브 바이트로 출력한다.

[0144] 그러나, N 값이 0으로 계산되면 서브 입력 데이터인 $IN[7:0]$ 을 그대로 출력하여, 순환 처리 서브 바이트로 출력한다.

[0145] 도 16의 (b)에 도시한 라이트 쉬프트 함수에 의한 연산 방법은 레프트 쉬프트 함수의 연산 방법과 동작 방식이 동일하나, 1 비트 순환 회전이 오른쪽으로 이루어진다.

[0146] 도 17은 본 발명의 실시예에 따른 서브 바이트 비트 재배치 함수의 연산 방법에 대한 예시도이다.

[0147] 도 17에 도시된 바와 같이, 서브 바이트 비트 재배치 함수는 라운드 키인 $K_p[j-2]$ 를 입력으로 받아, 7, 3, 0번째 비트 값의 배타적 논리 합(A), 6, 2, 1번째 비트 값의 배타적 논리 합(B)을 계산한다. 그리고, A와 B의 값에 따라 입력 IN 의 각 비트를 도 17에 도시한 바와 같이 배열하여 입력 라운드 키의 각 비트의 위치를 변경한다. 본 발명의 실시예에서는 서브 바이트 비트 재배치 함수가 유리 함수(DERV 함수)인 것을 예로 하여 설명하나, 반드시 이와 같이 한정되는 것은 아니다.

[0148] 이상에서 설명한 암호화 알고리즘이 적용된 시스템에서 데이터를 암호화하는 예에 대해 도 18을 참조로 설명한다.

[0149] 도 18은 본 발명의 실시예에 따른 암호화 알고리즘이 적용된 시스템의 데이터 암호화 동작 예시도이다.

[0150] 도 18에 도시된 데이터 암호화 동작 실시예는 16 바이트의 스크램블러 키 ($0x139050849F660D3D84E48ADF51672652$), 16 바이트의 데이터 ($0x4973207468697320746865207265616C$), 그리고 데이터의 시작 주소는 00000000, 라운드 횟수는 8회인 것을 예로 하여 설명한다.

[0151] 도 18에 도시된 바와 같이, 한 라운드 안에서 입력된 데이터 블록에 포함된 데이터들을 바이트 단위로 나누어 각각의 바이트 레인에서 처리한다. 이때, 바이트 단위로 나뉜 데이터들의 위치는 바뀌지 않고 바이트 레인으로 입력된다.

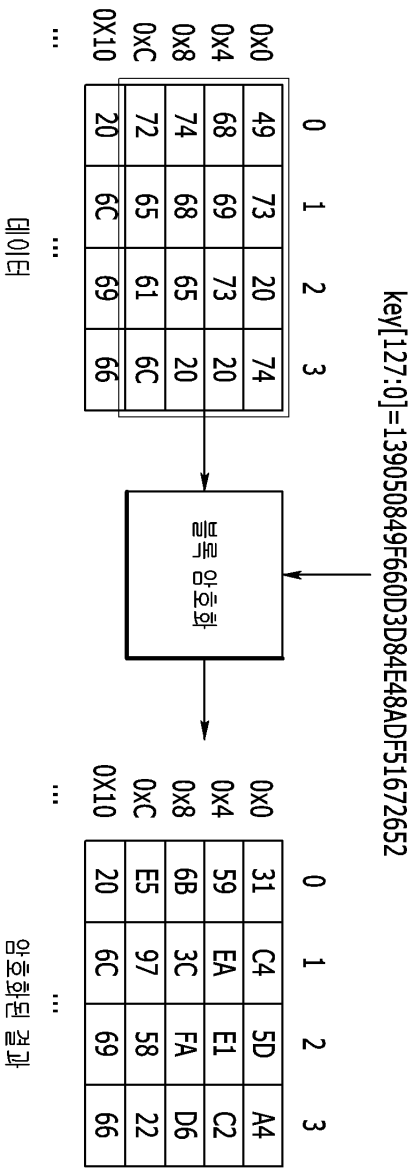
[0152] 예를 들어, 입력 데이터 RD_0 의 두 번째 바이트인 $RD_0[1]$ 은 라운드 0 수행 후 $RD_1[1]$ 에 위치하고, 라운드 1 수행 후 $RD_2[1]$ 에 위치한다. 마지막 라운드까지 이 위치는 유지된다.

[0153] 한 블록 안에서 연산 시작과 종료 시점의 데이터 바이트의 위치를 그대로 유지하는 구조 덕분에, 상기 도 3에서 설명한 바와 같이 '읽기-수정-쓰기' 없이 블록 안에서 다른 데이터를 변경하지 않으면서 특정 데이터를 변경하는 것이 가능하다.

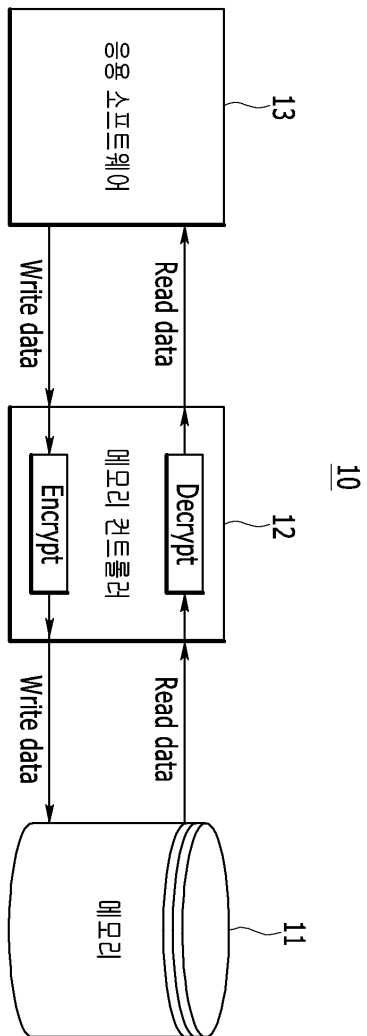
- [0154] 도 19는 본 발명의 실시예에 따른 데이터 변경 예시도이다.
- [0155] 상기 도 18에서 설명한 구조를 사용할 때, 도 19에 도시된 바와 같이 메모리에 저장되어 있는 값을 바꾸는 데이터 변경을 예로 하여 나타내었다. 응용 소프트웨어가 주소 0x0 ~ 0xF에 데이터(0x49732074...)를 썼을 때, 메모리 컨트롤러가 데이터를 암호화하여 메모리에 저장(0x704E4241...)한다.
- [0156] 이때, 사용자가 특정 주소의 데이터만을 변경하려고 시도한다고 가정하고, 변경하고자 하는 데이터가 주소 0x1의 값을 0x73에서 0xF1으로, 주소 0x9의 값을 0x68에서 0xF2로 변경하려 한다고 가정한다. 이 경우, 본 발명의 실시예에 따른 암호화 알고리즘을 사용하면, 주소 0x1과 0x9에 각각 0x16과 0x73을 쓰는 것으로 메모리에 저장된 데이터를 변경할 수 있다.
- [0157] 일반 블록 암호 알고리즘을 사용하는 경우 상기 도 3에서 설명한 바와 같이, 수정하려는 데이터가 포함된 블록을 전부 읽고 수정해야 하지만, 본 발명의 실시예에 따른 데이터 수정은 변경하고자 하는 위치의 데이터만 수정할 수 있기 때문에 메모리 내용을 빠르고 용이하게 변경할 수 있다.
- [0158] 이상에서 본 발명의 실시예에 대하여 상세하게 설명하였지만 본 발명의 권리범위는 이에 한정되는 것은 아니고 다음의 청구범위에서 정의하고 있는 본 발명의 기본 개념을 이용한 당업자의 여러 변형 및 개량 형태 또한 본 발명의 권리범위에 속하는 것이다.
- [0159] 이상에서 본 발명의 실시예에 대하여 상세하게 설명하였지만 본 발명의 권리범위는 이에 한정되는 것은 아니고 다음의 청구범위에서 정의하고 있는 본 발명의 기본 개념을 이용한 당업자의 여러 변형 및 개량 형태 또한 본 발명의 권리범위에 속하는 것이다.

도면

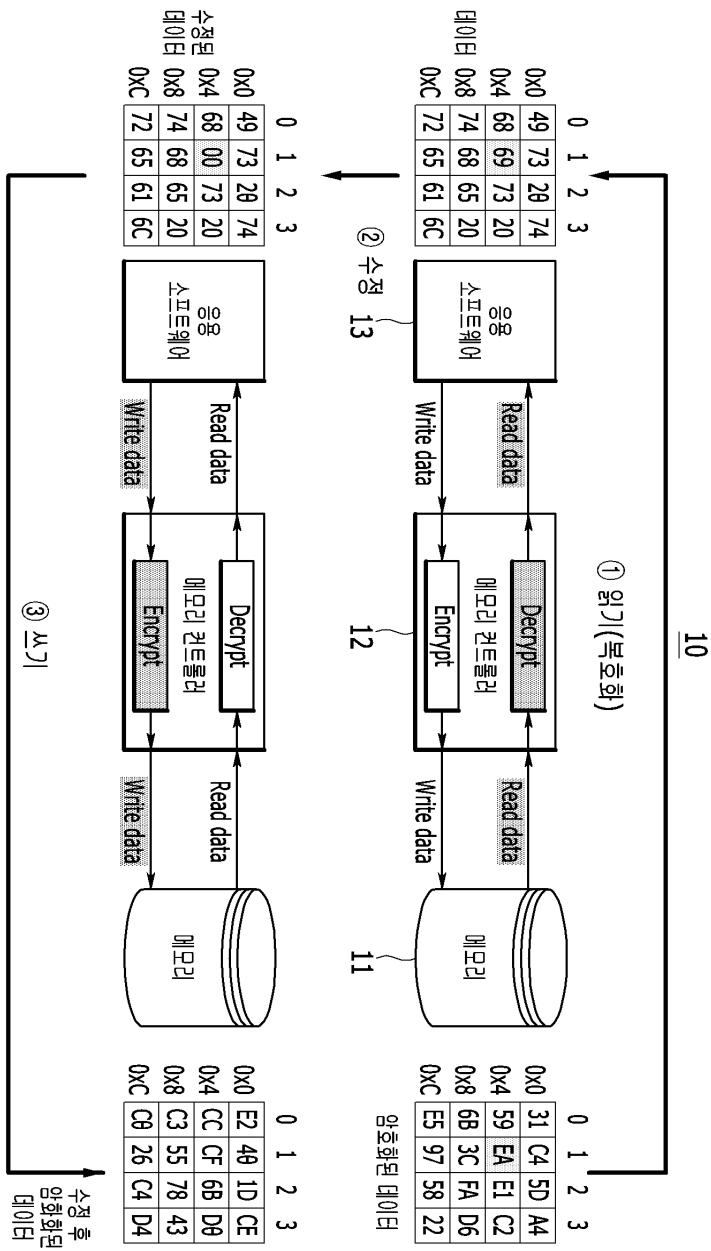
도면1



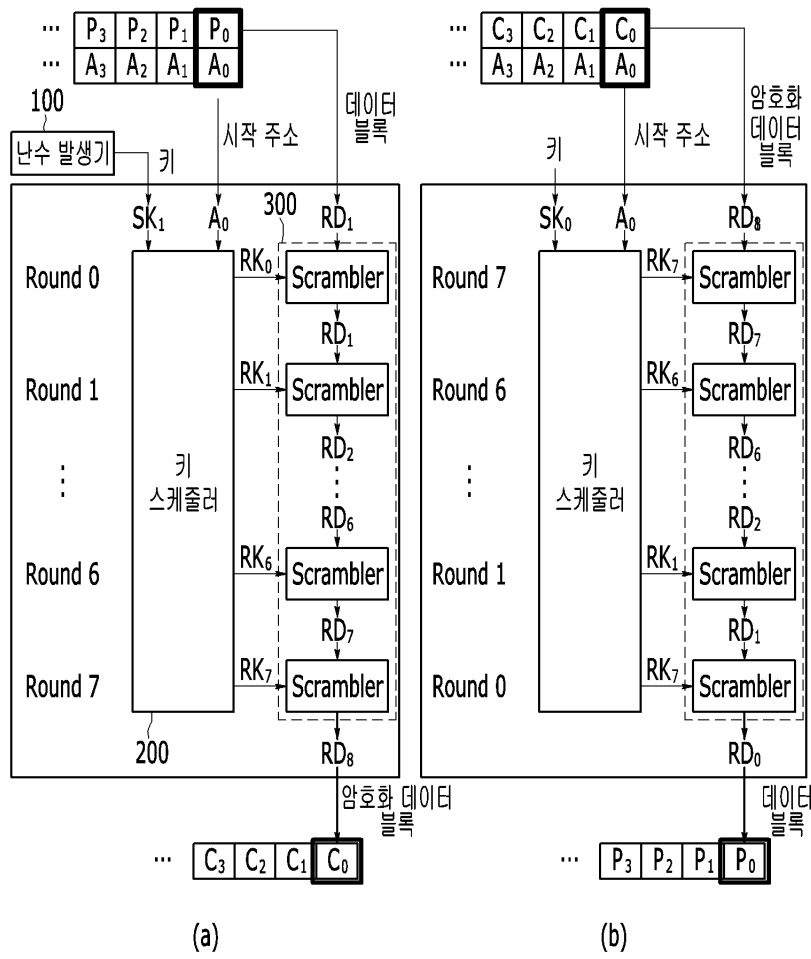
도면2



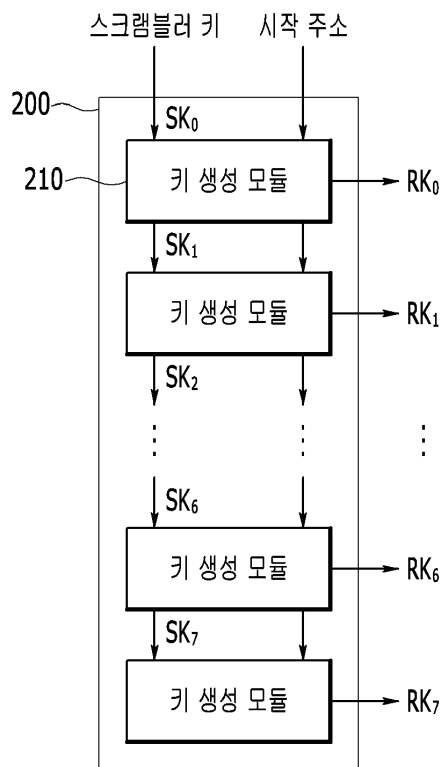
도면3



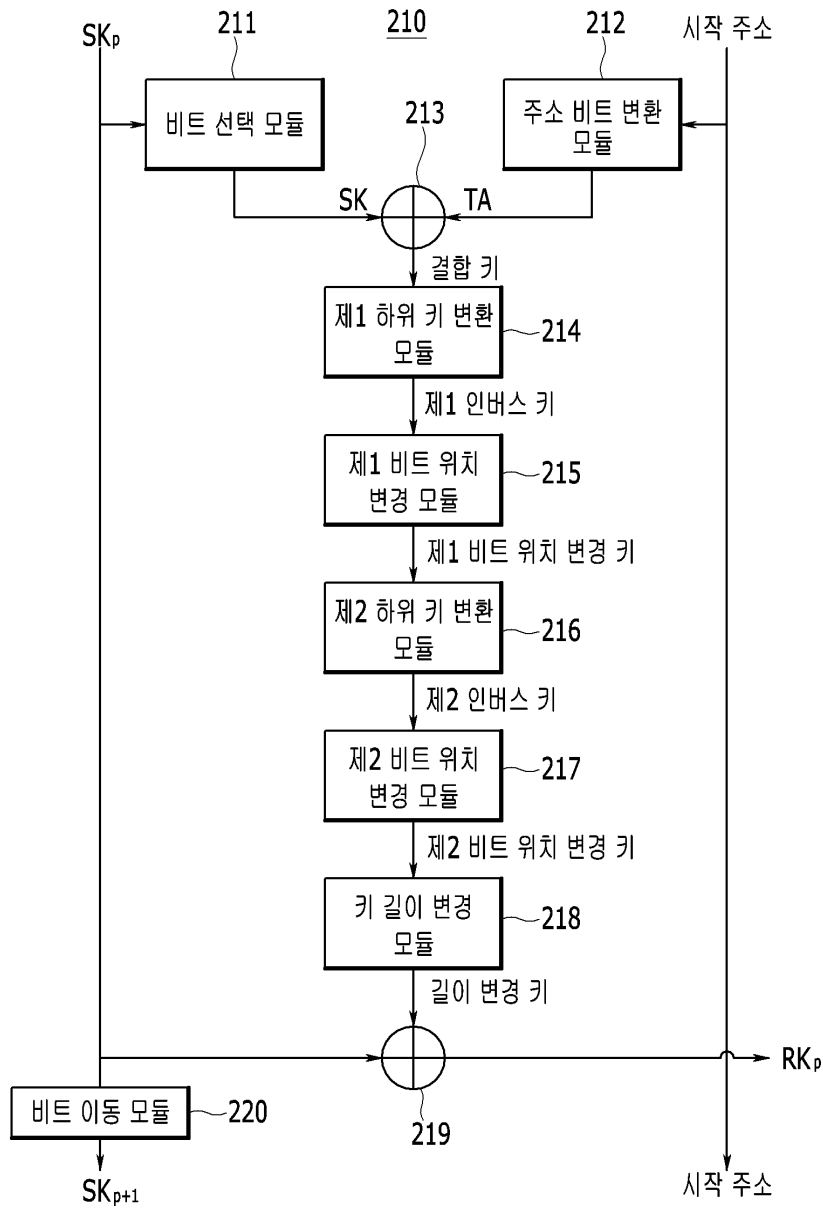
도면4



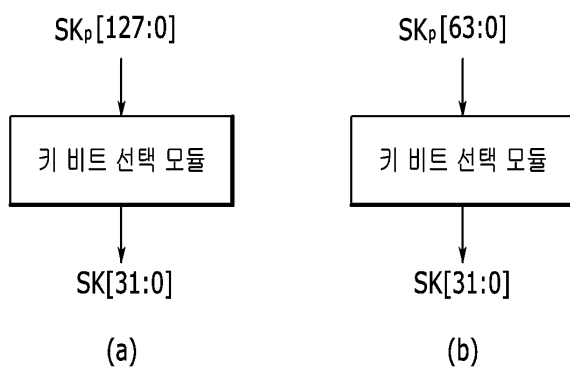
도면5



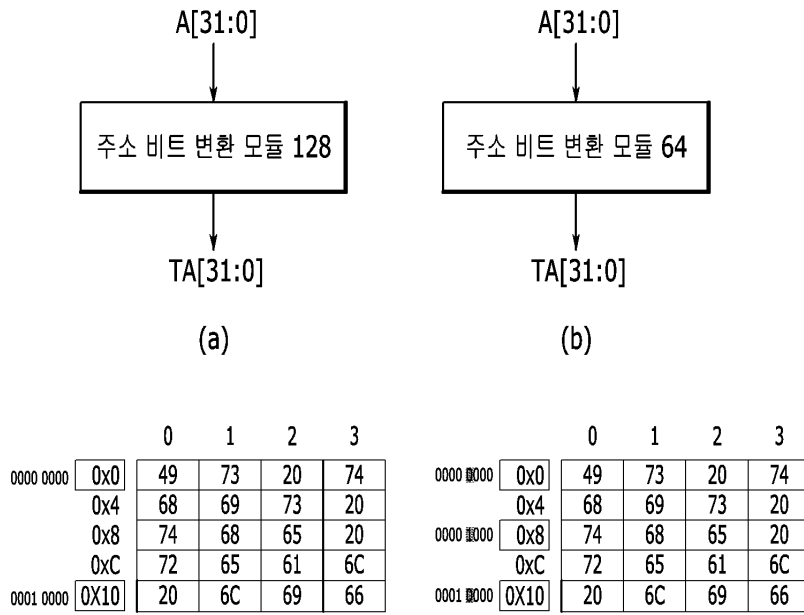
도면6



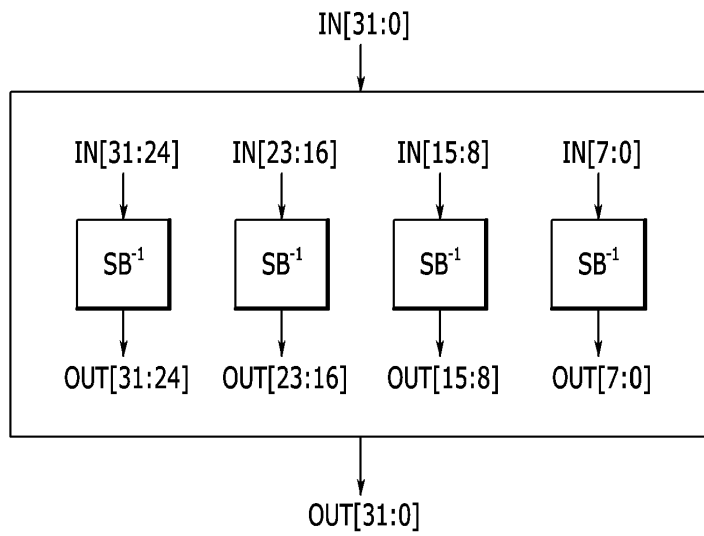
도면7



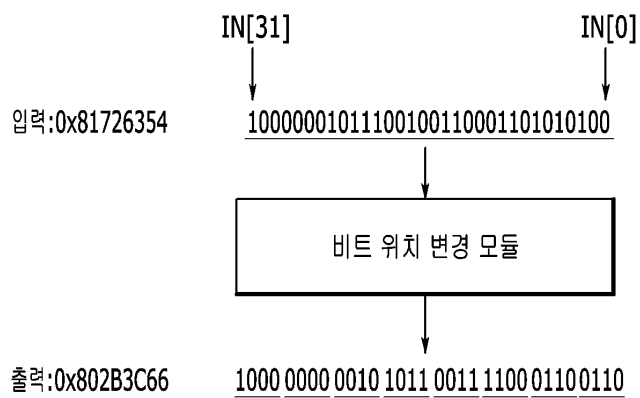
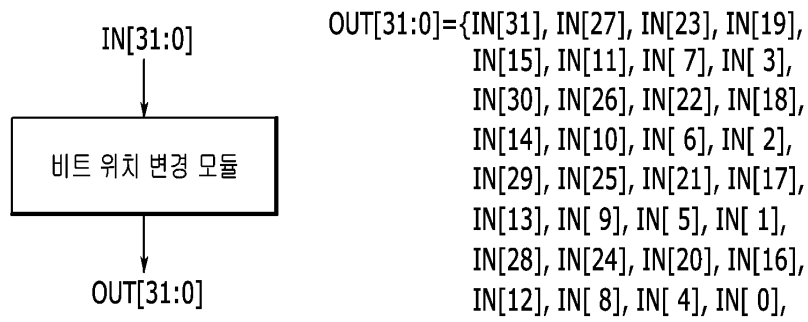
도면8



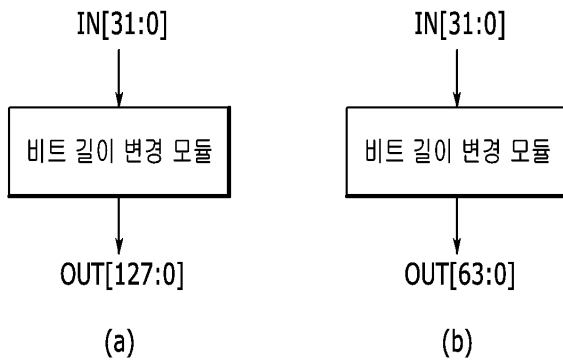
도면9



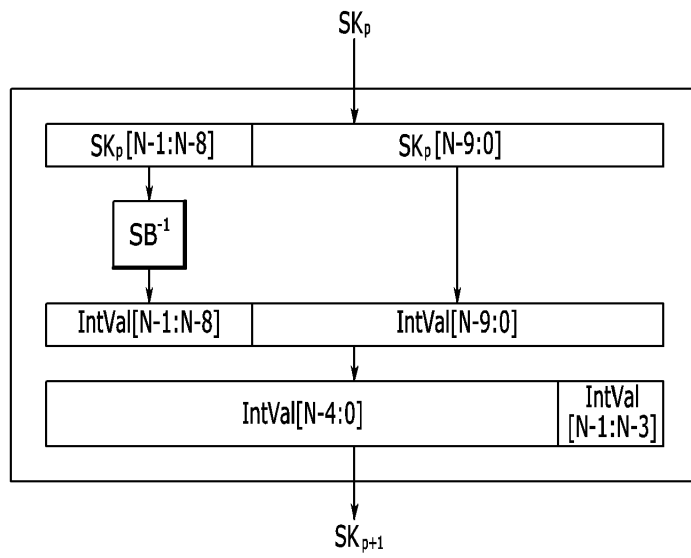
도면10



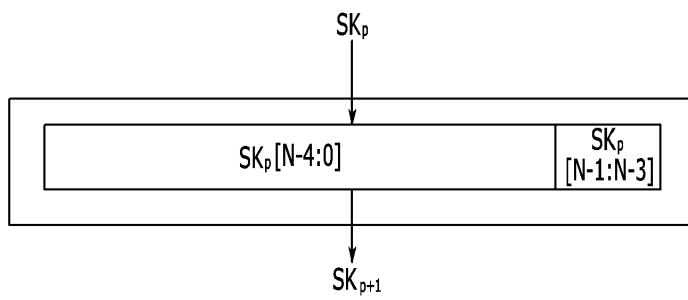
도면11



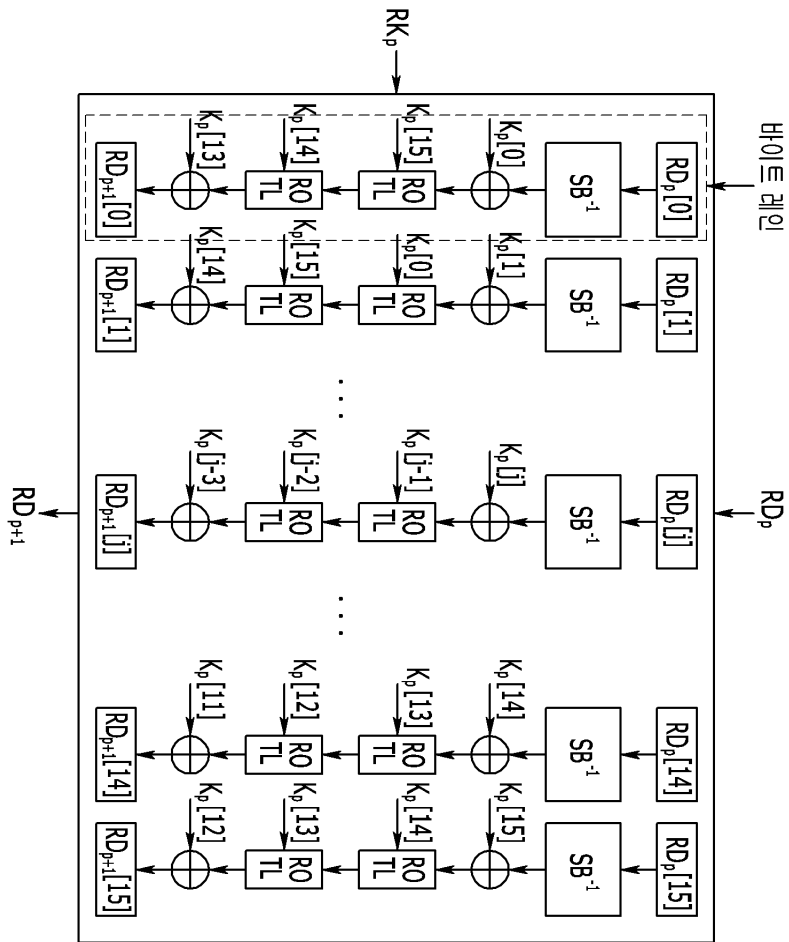
도면12



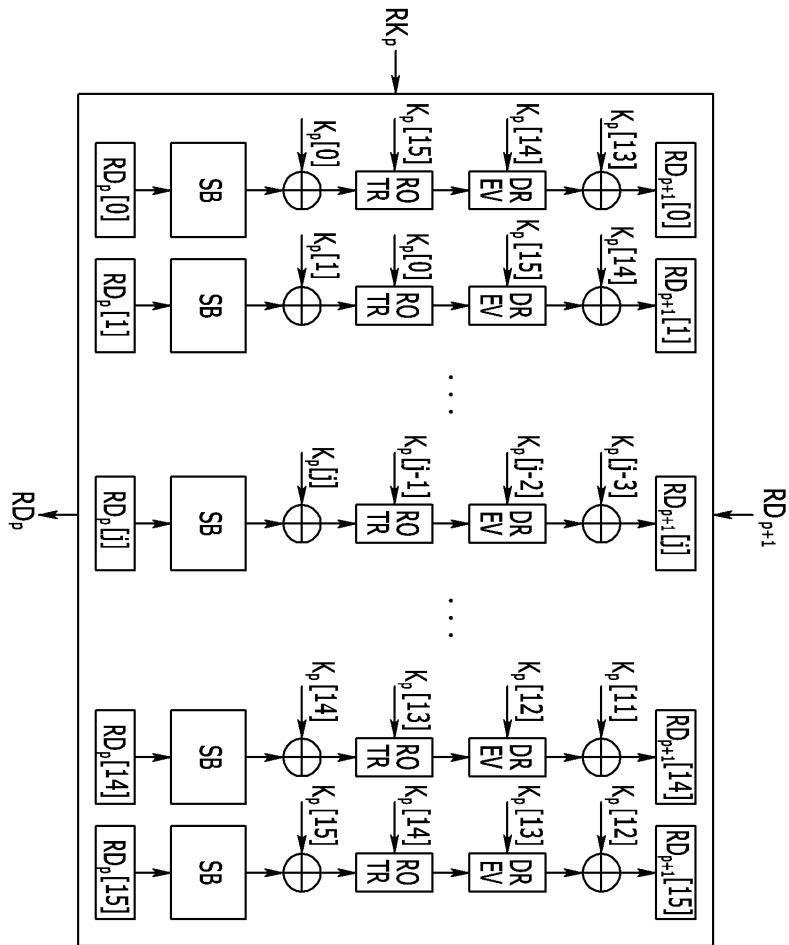
도면13



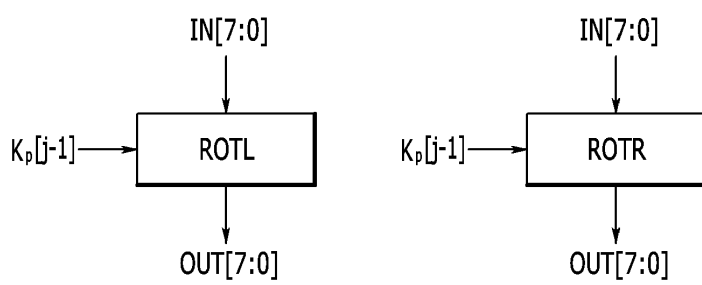
도면14



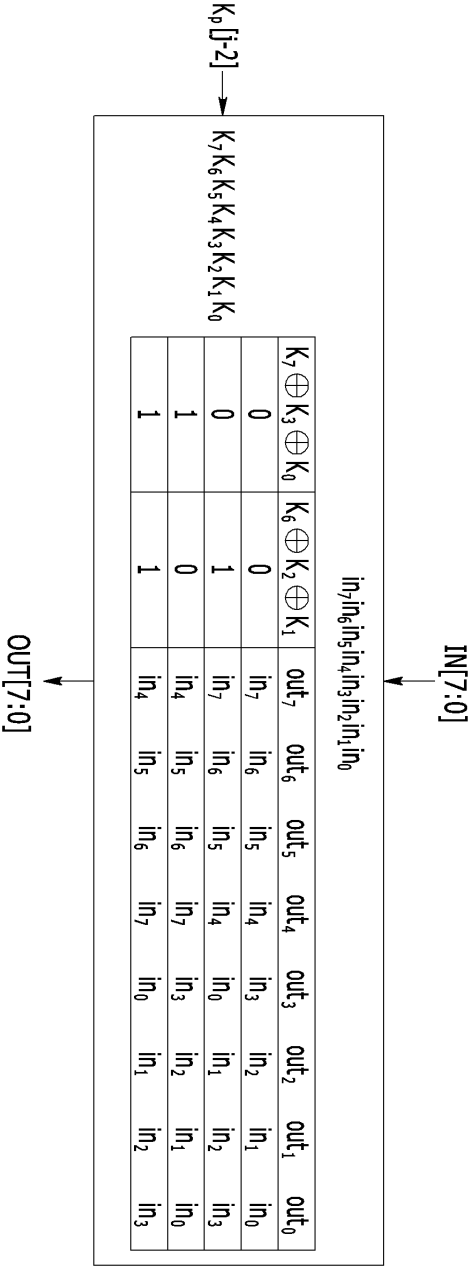
도면15



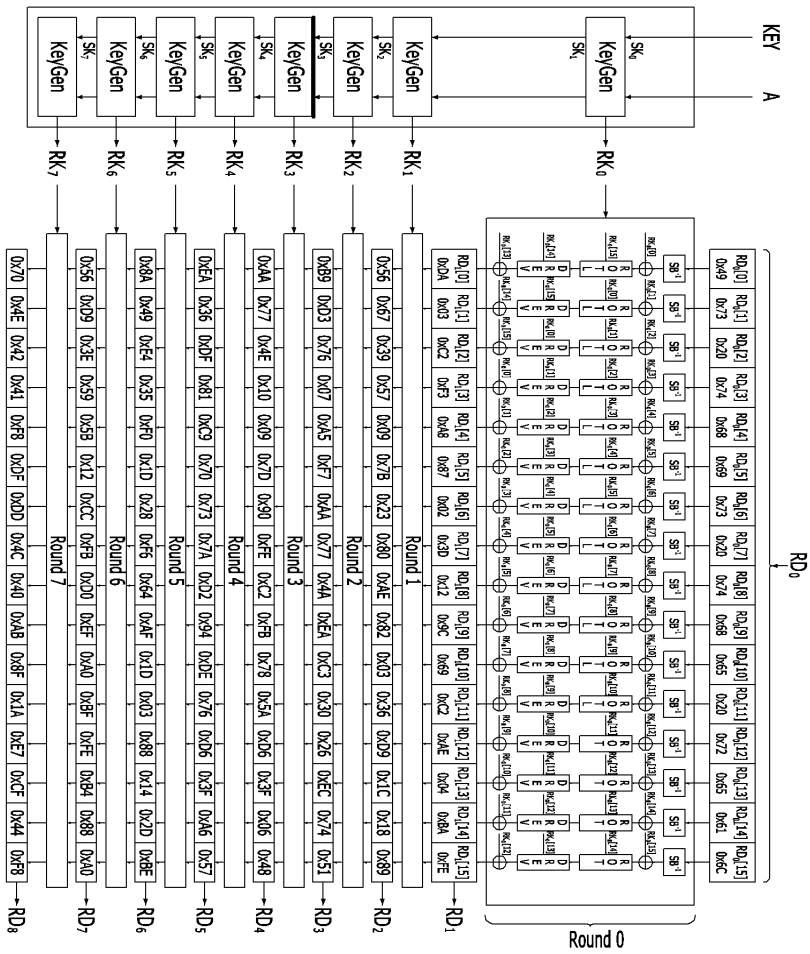
도면16



도면17



도면18



도면19

