(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2012년04월05일
(11) 등록번호 10-1133988
(24) 등록일자 2012년03월30일

(51) 국제특허분류(Int. Cl.)
H04L 9/14 (2006.01) *H04L 9/28* (2006.01)
(21) 출원번호 10-2010-0037773
(22) 출원일자 2010년04월23일
심사청구일자 2010년04월23일
(65) 공개번호 10-2011-0118273
(43) 공개일자 2011년10월31일
(56) 선행기술조사문헌
제목: MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences, 저자: B.Briscoe, 발행처: Springer - Lecture Note in Computer Science, 발행일: 1999*
KR1020010007044 A
KR1020020009464 A
논문(TIS LABS-TIS REPORT0755,1998)
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
한국과학기술원
대전 유성구 구성동 373-1
(72) 발명자
박규호
대전광역시 유성구 대학로 291, 한국과학기술원 6-3208호 (구성동)
박기웅
서울특별시 노원구 광운로15길 48 (월계동)
김철민
대전광역시 유성구 대학로 291, 한국과학기술원 전자동 3243호 (구성동)
(74) 대리인
김성호

전체 청구항 수 : 총 11 항 심사관 : 이형일

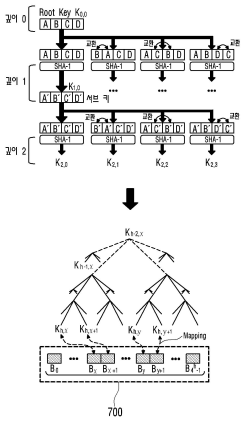
(54) 발명의 명칭 **해쉬 트리 기반의 스트림 암호화 및 복호화 방법과 암호 파일 시스템**

(57) 요 약

실시예는 스트림 암호화 방법에 관한 것이다.

실시예에 따른 스트림 암호화 방법은, 파일 시스템 암호화에 사용할 루트 키를 생성하는 단계, 해쉬 함수를 통해 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 단계, 키 트리에 구성된 복수의 서브 키와 암호 파일 시스템의 스토리지에 구성된 데이터 블록들을 매핑하는 단계, 및 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 단계를 포함한다.

대 표 도 - 도5



특허청구의 범위

청구항 1

암호 파일 시스템에서의 스트림 암호화 방법으로서,

파일 시스템 암호화에 사용할 루트 키를 생성하는 단계;

해쉬 함수를 통해 상기 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 단계;

상기 키 트리에 구성된 복수의 서브 키와, 상기 암호 파일 시스템의 스토리지에 구성된 데이터 블록들을 매핑하는 단계; 및

상기 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 단계를 포함하되,

상기 해쉬 함수의 입력은 상기 루트 키 또는 상기 복수의 서브 키를 교환 연산(Swapping Operation)함으로써 생성되는,

해쉬 트리 기반의 스트림 암호화 방법.

청구항 2

제1항에 있어서,

상기 키 트리를 구성하는 단계는,

a. 상기 루트 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키로 생성하는 단계;

b. 상기 a 단계를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하는 단계;

c. 상기 b 단계를 통해 생성된 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키로 생성하는 단계; 및

d. 상기 c 단계를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하는 단계를 포함하며,

상기 a 단계 내지 상기 d 단계를 통해 서브 키를 확장하면서 키 트리를 구성하되, 하기의 수식을 만족하는 상기 키 트리의 깊이(h)로 키 트리를 구성하며,

$$m^{h-1} < n < m^{h}$$

m은, 상기 서브섹션의 개수이고,

n은, 상기 데이터 블록의 개수인, 해쉬 트리 기반의 스트림 암호화 방법.

청구항 3

제1항의 방법에 따라 생성된 상기 암호문 스트림을 복호화하는 방법으로서,

상기 스토리지의 데이터 블록에 매핑된 복수의 서브 키를 해쉬 함수를 통하여 유도하는 단계; 및

해쉬 함수를 통해 유도된 복수의 서브 키와 상기 암호문 스트림을 XOR 연산하여 복호문 스트림을 생성하는 단계를 포함하는 해쉬 트리 기반의 스트림 복호화 방법.

청구항 4

제3항에 있어서,

상기 스토리지에 저장된 암호문 스트림은 데이터 블록 단위로 접근 가능한, 해쉬 트리 기반의 스트림 복호화 방법.

청구항 5

스트림 암호화를 위한 시스템으로서,

파일 시스템 암호화에 사용할 루트 키를 생성하는 루트 키 생성부;

해쉬 함수를 통해 상기 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 키 트리 구성부;

암호문 스트림을 저장하기 위한 복수의 데이터 블록을 포함하는 스토리지;

상기 키 트리에 구성된 복수의 서브 키와 상기 스토리지의 데이터 블록들을 매핑하는 매핑부; 및

상기 스토리지의 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 암호화 연산부를 포함하되,

상기 해쉬 함수의 입력은 상기 루트 키 또는 상기 복수의 서브 키를 교환 연산함으로써 생성되는,

암호 파일 시스템.

청구항 6

제5항에 있어서,

상기 키 트리 구성부는,

교환 연산부 및 키 생성부를 포함하고,

상기 교환 연산부는,

상기 루트 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키를 생성하고,

상기 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키를 각각 생성하며,

상기 키 생성부는,

상기 교환 연산부를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하고,

상기 교환 연산부를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하며,

상기 키 트리 구성부는,

상기 교환 연산부와 상기 키 생성부를 통해 서브 키를 확장하면서 키 트리를 구성하되, 하기의 수식을 만족하는 키 트리의 깊이(h)로 키 트리를 구성하며,

$$m^{h-1} < n < m^{h}$$

m은 상기 서브섹션의 개수이고,

n은 상기 데이터 블록의 개수인, 암호 파일 시스템.

청구항 7

제5항의 암호 파일 시스템에 따라 생성된 상기 암호문 스트림을 복호화하는 시스템으로서,

상기 스토리지의 데이터 블록에 매핑된 복수의 서브 키를 해쉬 함수를 이용하여 유도하는 키 유도부; 및

상기 키 유도부로부터 유도된 복수의 서브 키와, 상기 암호문 스트림을 XOR 연산하여 복호문 스트림을 생성하는 복호화 연산부

를 포함하는 암호 파일 시스템.

청구항 8

제7항에 있어서,

상기 스토리지에 저장된 암호문 스트림은 데이터 블록 단위로 접근 가능한, 암호 파일 시스템.

청구항 9

스트림 암호화 및 복호화를 위한 시스템으로서,

파일 시스템 암호화에 사용할 루트 키를 생성하는 루트 키 생성부;

해쉬 함수를 통해 상기 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 키 트리 구성부;

암호문 스트림을 저장하기 위한 복수의 데이터 블록을 포함하는 스토리지;

상기 키 트리에 구성된 복수의 서브 키와 상기 스토리지의 데이터 블록들을 매핑하는 매핑부; 및

상기 스토리지의 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 암호화 연산부;

상기 스토리지의 데이터 블록에 매핑된 복수의 서브 키를 해쉬 함수를 이용하여 유도하는 키 유도부; 및

상기 키 유도부로부터 유도된 복수의 서브 키와, 상기 암호문 스트림을 XOR 연산하여 복호문 스트림을 생성하는 복호화 연산부를 포함하되,

상기 해쉬 함수의 입력은 상기 루트 키 또는 상기 복수의 서브 키를 교환 연산함으로써 생성되는,

암호 파일 시스템.

청구항 10

제9항에 있어서,

상기 키 트리 구성부는,

교환 연산부 및 키 생성부를 포함하고,

상기 교환 연산부는,

상기 루트 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키를 생성하고,

상기 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키를 각각 생성하며,

상기 키 생성부는,

상기 교환 연산부를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하고,

상기 교환 연산부를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하며,

상기 키 트리 구성부는,

상기 교환 연산부와 상기 키 생성부를 통해 서브 키를 확장하면서 키 트리를 구성하되, 하기의 수식을 만족하는 키 트리의 깊이(h)로 키 트리를 구성하며,

$$m^{h-1} < n < m^{h}$$

m은 상기 서브섹션의 개수이고,

n은 상기 데이터 블록의 개수인, 암호 파일 시스템.

청구항 11

제9항에 있어서,

상기 스토리지에 저장된 암호문 스트림은 데이터 블록 단위로 접근 가능한, 암호 파일 시스템.

명 세 서

기 술 분 야

[0001] 스트림 암호화/복호화 방법 및 암호 파일 시스템에 관한 것이다.

배 경 기 술

[0002] “Amazon Simple Storage Service(S3)” 와 같은 서비스가 활성화되면서, 사용자의 정보가 자신의 컴퓨터 장치에 저장되는 것이 아닌, 중앙 서버에 저장되는 스토리지 아웃소싱 서비스 시장이 커지고 있다. 이러한 서비스에 의해 스토리지 장치의 확장 없이, 필요한 만큼 저장장치를 임대하여 사용할 수 있게 되어, 그에 따른 편리성이 증대되었다.

[0003] 그러나 이러한 서비스의 활성화는 새로운 보안 문제를 야기하고 있다. 즉, 사용자의 정보가 자신의 PC 등이 아닌 인터넷 상의 다른 서버에 저장됨으로써, 기밀성에 대한 문제가 발생하게 된 것이다. 자신의 PC에는 아무런 암호화 없이 저장되어도 아무런 상관이 없었던 것들이 사용자 정보의 저장위치가 바뀜에 따라 기밀성이 중요한 문제로 대두되고 있다.

[0004] 이러한 문제를 해결하기 위해 암호 파일 시스템이란 것이 개발되었다. 암호 파일 시스템이란, 사용자의 데이터를 그대로 저장하는 것이 아니라, 암호화하여 저장하는 시스템을 의미한다.

[0005] 데이터 암호화(encryption)란 가치가 있는 정보(평문, plain text)를 제 3자가 의미를 알 수 없는 형식(암호문, cipher text)으로 변환하는 것을 의미하며, 평문은 암호 키를 사용하여 암호문으로 변환될 수 있다. 이와 반대로, 데이터 복호화(decryption)란 암호문을 평문으로 변환하는 것을 의미한다. 기억 장치에 저장되어 있거나 통신 회선을 통해 전송된 암호문은 복호 키를 사용하여 원래의 정보로 복원될 수 있다. 복호 키를 갖고 있지 않은 사람은 원래의 정보로 올바르게 복원할 수 없으므로, 복호 키가 제3자에게 알려지지 않으면 정보는 보호된다.

[0006] 일반적으로, 데이터를 암호화하는 방법에는 블록 암호화 알고리즘과 스트림 암호화 알고리즘으로 분류된다.

[0007] 도 1은 블록 암호화 알고리즘에 대한 개념을 나타낸 도면이다. 도 2는 스트림 암호화 알고리즘에 대한 개념을 나타낸 도면이다.

[0008] 블록 암호화 알고리즘은, 데이터를 블록 단위로 암호화하는 알고리즘으로서, 통상 128비트 단위로 암호화시키는 알고리즘이다.

[0009] 스트림 암호화 알고리즘은, 사용자의 키를 씨드 키(seed key) 값으로 하여 무작위 수를 발생시켜, 평문과 XOR 연산을 통해 암호화시키는 알고리즘이다.

[0010] 이러한 두 가지 알고리즘은 각각 다음과 같은 장단점을 가지고 있다.

[0011] 블록 암호화 알고리즘은, 블록 단위로 암호화를 시키기 때문에 블록 단위의 접근이 가능하다는 장점을 갖는다. 그러나, 스트림 암호화 알고리즘이 비해 매우 복잡한 연산 과정이 수반되어야 한다는 단점을 가지고 있다.

[0012] 스트림 암호화 알고리즘은, 난수열과 평문의 XOR 연산을 통해 암호화 시키기 때문에, 연산 속도가 매우 빠르다는 장점을 갖는다. 그러나, 암호화된 데이터 중 특정 블록을 해독하기 위해서는, 해당 블록을 암호화하는데 사용되었던 난수열을 생성해야 하고, 또한 그러기 위해 씨드 키 값을 이용하여 해당 상태가 나올 때까지 난수열을 생성해야 해독할 수 있으므로, 무작위 접근성이 매우 낮다는 단점을 갖는다.

[0013] 따라서, 일반적인 블록 암호화 알고리즘은 블록 단위의 접근성을 제공하지만, 연산 복잡도가 스트림 암호화 알고리즘에 비해 높다는 단점이 있고, 스트림 암호화 알고리즘은 블록 암호화 알고리즘에 비해 연산 속도가 매우 빠르다는 장점이 있지만, 블록 단위의 접근성이 제공되지 않는다는 단점이 있다.

발명의 내용

해결하려는 과제

[0014] 실시예는, 스트림 암호화 알고리즘의 빠른 연산 속도와, 블록 암호화 알고리즘의 블록 단위의 접근성을 제공할 수 있는 암호화 및 복호화 방법을 제공하는 것에 목적이 있다.

[0015] 실시예는, 스트림 암호화 알고리즘의 빠른 연산 속도와, 블록 암호화 알고리즘의 블록 단위의 접근성을 제공할 수 있는 암호 파일 시스템을 제공하는 것에 목적이 있다.

과제의 해결 수단

[0016] 청구항 1에 따른 스트림 암호화 방법은, 암호 파일 시스템에서의 스트림 암호화 방법으로서, 파일 시스템 암호화에 사용할 루트 키를 생성하는 단계, 해쉬 함수를 통해 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 단계, 키 트리에 구성된 복수의 서브 키와 암호 파일 시스템의 스토리지에 구성된 데이터 블록들을 매핑하는 단계, 및 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 단계를 포함한다.

[0017] 청구항 2에 따른 키 트리를 구성하는 단계는,

[0018] a. 루트 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키로 생성하는 단계, b. a 단계를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하는 단계, c. b 단계를 통해 생성된 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키로 생성하는 단계, 및 d. c 단계를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하는 단계를 포함하며,

[0019] a 단계 내지 d 단계를 통해 서브 키를 확장하면서 키 트리를 구성하되, 하기의 수식을 만족하는 키 트리의 깊이(h)로 키 트리를 구성하며,

[0020] $$m^{h-1} < n < m^{h}$$

[0021] m은 서브섹션의 개수이고, n은 데이터 블록의 개수인 것이 바람직하다.

[0022] 실시예에 따른 스트림 복호화 방법은, 청구항 1에 따라 생성된 암호문 스트림을 복호화하는 방법으로서, 스토리지의 데이터 블록에 매핑된 복수의 서브 키를 해쉬 함수를 통하여 유도하는 단계, 및 해쉬 함수를 통해 유도된 복수의 서브 키와 암호문 스트림을 XOR 연산하여 복호문 스트림을 생성하는 단계를 포함한다.

[0023] 스토리지에 저장된 암호문 스트림은 데이터 블록 단위로 접근 가능한 것이 바람직하다.

[0024] 청구항 5에 따른 암호 파일 시스템은, 스트림 암호화를 위한 시스템으로서, 파일 시스템 암호화에 사용할 루트 키를 생성하는 루트 키 생성부, 해쉬 함수를 통해 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 키 트리 구성부, 암호문 스트림을 저장하기 위한 복수의 데이터 블록을 포함하는 스토리지, 키 트리에 구성된 복수의 서브 키와 스토리지의 데이터 블록들을 매핑하는 매핑부, 및 스토리지의 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 암호화 연산부를 포함한다.

[0025] 청구항 6에 따른 키 트리 구성부는,

[0026] 교환 연산부 및 키 생성부를 포함하고,

[0027] 교환 연산부는,

[0028] 루트 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키를 생성하고, 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키를 각각 생성하며,

[0029] 키 생성부는,

[0030] 교환 연산부를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하고, 교환 연산부를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하며,

[0031] 키 트리 구성부는,

[0032] 교환 연산부와 키 생성부를 통해 서브 키를 확장하면서 키 트리를 구성하되, 하기의 수식을 만족하는 키 트리의 깊이(h)로 키 트리를 구성하며,

[0033] $$m^{h-1} < n < m^{h}$$

[0034] m은 서브섹션의 개수이고, n은 데이터 블록의 개수인 것이 바람직하다.

[0035] 청구항 7에 따른 암호 파일 시스템은, 청구항 5의 암호 파일 시스템에 따라 생성된 암호문 스트림을 복호화하는 시스템으로서, 스토리지의 데이터 블록에 매핑된 복수의 서브 키를 해쉬 함수를 이용하여 유도하는 키 유도부, 및 키 유도부로부터 유도된 복수의 서브 키와, 암호문 스트림을 XOR 연산하여 복호문 스트림을 생성하는 복호화 연산부를 포함한다.

[0036] 스토리지에 저장된 암호문 스트림은 데이터 블록 단위로 접근 가능한 것이 바람직하다.

[0037] 청구항 9에 따른 암호 파일 시스템은, 스트림 암호화 및 복호화를 위한 시스템으로서, 파일 시스템 암호화에 사용할 루트 키를 생성하는 루트 키 생성부, 해쉬 함수를 통해 루트 키로부터 복수의 서브 키를 생성하여 키 트리를 구성하는 키 트리 구성부, 암호문 스트림을 저장하기 위한 복수의 데이터 블록을 포함하는 스토리지, 키 트리에 구성된 복수의 서브 키와 스토리지의 데이터 블록들을 매핑하는 매핑부, 및 스토리지의 데이터 블록들에 매핑된 복수의 서브 키와, 평문 스트림을 XOR 연산하여 암호문 스트림을 생성하는 암호화 연산부, 스토리지의 데이터 블록에 매핑된 복수의 서브 키를 해쉬 함수를 이용하여 유도하는 키 유도부, 및 키 유도부로부터 유도된 복수의 서브 키와, 암호문 스트림을 XOR 연산하여 복호문 스트림을 생성하는 복호화 연산부를 포함한다.

[0038] 청구항 10에 따른 키 트리 구성부는,

[0039] 교환 연산부 및 키 생성부를 포함하고,

[0040] 교환 연산부는,

[0041] 루트 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키를 생성하고, 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 서브섹션들 간의 위치 교환을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키를 각각 생성하며,

[0042] 키 생성부는,

[0043] 교환 연산부를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하고, 교환 연산부를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성하며,

[0044] 키 트리 구성부는,

[0045] 교환 연산부와 키 생성부를 통해 서브 키를 확장하면서 키 트리를 구성하되, 하기의 수식을 만족하는 키 트리의 깊이(h)로 키 트리를 구성하며,

[0046] $$m^{h-1} < n < m^{h}$$

[0047] m은 서브섹션의 개수이고, n은 데이터 블록의 개수인 것이 바람직하다.

[0048] 청구항 11에 따른 스토리지에 저장된 암호문 스트림은 블록 단위로 접근 가능한 것이 바람직하다.

발명의 효과

[0049] 실시예에 따르면, 스트림 암호화 알고리즘의 빠른 연산 속도와, 블록 암호화 알고리즘의 블록 단위의 접근성을 제공할 수 있는 암호화/복호화 방법을 제공할 수 있다.

[0050] 실시예에 따르면, 스트림 암호화 알고리즘의 빠른 연산 속도와, 블록 암호화 알고리즘의 블록 단위의 접근성을 제공할 수 있는 암호 파일 시스템을 제공할 수 있다.

도면의 간단한 설명

[0051] 도 1은 블록 암호화 알고리즘에 대한 개념을 나타낸 도면.

도 2는 스트림 암호화 알고리즘에 대한 개념을 나타낸 도면.

도 3은 본 발명에 대한 개념을 나타낸 도면.

도 4는 실시예에 따른 스트림 암호화/복호화 방법을 나타낸 흐름도.

도 5는 실시예에 따른 키 트리 구성 방법 및 서브 키와 데이터 블록을 매핑하는 방법을 도시한 도면.

도 6은 실시예에 따라 암호문 스트림이 생성되어 스토리지 데이터 블록에 저장되는 과정을 도시한 도면.

도 7 은 실시예에 따른 스트림 암호화 알고리즘을 이용하여 파일을 저장하고 읽어오는 전반적인 과정을 도시한 도면.

도 8은 실시예에 따른 암호 파일 시스템의 전체적인 구성을 나타낸 블록도.

도 9는 실시예에 따른 스트림 암호화 알고리즘(BA-RC4), 종래의 블록 암호화 알고리즘(AES), 및 스트림 암호화 알고리즘(RC4)의 성능 비교를 나타낸 그래프.

발명을 실시하기 위한 구체적인 내용

[0052] 도 3은 본 발명에 대한 개념을 나타낸 도면이다.

[0053] 도 3을 참조하면, 본 발명은 블록 단위의 암호화를 위해 사용될 키(key)를 확장시키고, 확장된 키를 이용하여

스토리지의 데이터 블록을 스트림 암호화 알고리즘으로 암호화시킴으로써, 블록 단위의 접근(access)을 제공한다. 키의 확장은, 단 방향 해쉬 함수(one-way hash function)를 이용한 키 트리 구축을 통해 이루어진다. 스트림 암호화/복호화 방법 및 암호 파일 시스템에 대한 실시예를 통하여 보다 상세히 후술하도록 한다.

[0054] 이하, 실시예에 대하여 첨부한 도면을 참조하여 상세하게 설명하기로 한다. 단, 첨부된 도면은 실시예의 내용을 보다 쉽게 개시하기 위하여 설명되는 것일 뿐, 본 발명의 범위가 첨부된 도면의 범위로 한정되는 것이 아님은 이 기술분야의 통상의 지식을 가진 자라면 용이하게 알 수 있을 것이다.

[0055] 이하에서 설명하는 실시예에 따른 암호화 방식은 BA-RC4로 명명한다. 여기서 'BA'는 블록 접근(Block Access)를 의미하며, 'RC4'는 실시예에서 적용되는 스트림 암호화 알고리즘을 의미한다.

[0056] **1. 스트림 암호화/복호화 방법**

[0057] **1) 스트림 암호화 방법**

[0058] 도 4는 실시예에 따른 스트림 암호화/복호화 방법을 나타낸 흐름도이다. 도 5는 실시예에 따른 키 트리 구성 방법 및 서브 키와 데이터 블록을 매핑하는 방법을 도시한 도면이다.

[0059] 도 4를 참조하면, 실시예에 따른 스트림 암호화 방법은, 루트 키 생성 단계(S410), 키 트리 구성 단계(S420), 서브 키와 데이터 블록 매핑 단계(S430), 및 암호문 스트림 생성 단계(S440)를 포함한다.

[0060] 루트 키 생성 단계(S410)

[0061] S410 단계에서는, 평문(plain text)을 암호화하는데 필요한 키 트리를 구성하기 위하여 루트 키(Root Key, $K_{0,0}$)를 생성한다. 여기서 루트 키(Root Key, $K_{0,0}$)는 사용자만이 알고 있는 키로서, 키 트리의 최상위 노드에 있는 키에 해당된다. 단 방향 해쉬 함수는 SHA-1(Secure Hash Algorithm-1)과 같이 보안적으로 안전한 해쉬 함수를 사용하는 것이 바람직하다.

[0062] 키 트리 구성 단계(S420)

[0063] 도 5에 도시된 바와 같이, 먼저 루트 키(Root Key, $K_{0,0}$) 값을 복수의 서브섹션으로 나눈다. 서브섹션의 개수(m)는 임의로 설정할 수 있으며, 실시예에서는 4개로 설정하여 설명하도록 한다.

[0064] 다음, 4개로 나누어진 서브섹션(A B C D)을 교환 연산(Swapping Operation)을 통해 교환하여, 해쉬 함수(SHA-1)에 입력될 4개의 루트 키를 생성한다. 4개의 루트 키는 서로 다른 값을 가지며, 한 번의 교환 연산을 통해 생성된다. 예를 들어, 루트 키(Root Key, $K_{0,0}$)의 서브섹션 값이 A B C D 라고 가정하면, 이 서브섹션 값들 간의 위치 교환을 통해 SHA(A B C D), SHA(B A C D), SHA(A C B D), SHA(A B D C)와 같은 해쉬 함수(SHA-1)의 입력 값을 얻어낼 수 있다.

[0065] 다음, 교환 연산(Swapping Operation)을 통해 얻은 키 값들은 해쉬 함수(SHA-1)에 각각 입력하고, 서로 다른 값을 갖는 새로운 서브 키들($K_{1,0}$, $K_{1,1}$, $K_{1,2}$, $K_{1,3}$)을 생성한다. 이에 따라, 하나의 부모 노드에서 자식 노드로 분화됨으로써, 깊이 1의 키 트리가 구성될 수 있다.

[0066] 다음, 해쉬 함수(SHA-1)를 통해 얻어진 각 서브 키들($K_{1,0}$, $K_{1,1}$, $K_{1,2}$, $K_{1,3}$)의 값을 다시 4개의 서브섹션으로 나눈다. 각 서브섹션 값을 교환 연산(Swapping Operation)을 통해 서로 다른 값을 갖도록 하여, 해쉬 함수(SHA-1)에 입력될 4개의 서브 키를 생성한다. 예를 들어, SHA(B A C D)를 입력 값으로 하여 해쉬 함수(SHA-1)를 통해 얻어진 서브 키가 A', B', C', D'의 서브섹션 값으로 이루어졌다고 가정하면, 상술한 교환 연산(Swapping Operation)을 통해 'SHA(A' B' C' D'), SHA(B' A' C' D'), SHA(A' C' B' D'), SHA(A' B' D' C')와 같은 값을 얻어 낼 수 있다. 이러한 값은 다시 해쉬 함수(SHA-1)로 입력되어 새로운 서브 키들($K_{2,0}$, $K_{2,1}$, $K_{2,2}$, $K_{2,3}$)이 생성될 수 있다. 이에 따라, 어느 하나의 자식 노드에서 또 다른 자식 노드로 분화됨으로써, 깊이 2의 키 트리가 구성될 수 있다.

[0067] 키 트리는 상술한 방법들을 통해 키를 확장시켜 나가면서 구성될 수 있다. 키 확장을 위해 새로운 서브 키 값을 얻기 위해서는, 해쉬 함수(SHA-1)에 입력될 값이 서로 달라야 하며, 이를 위해 실시예에서는 서브섹션을 교환하

는 방법을 적용하였다. 그러나, 해쉬 함수(SHA-1)의 입력 값을 얻기 위한 방법이 반드시 교환 연산에 의한 방법에만 한정되는 것이 아니라, 다양한 방법들이 적용 가능하다.

[0068] 실시예에 따른 키(루트 키 또는 서브 키)는 하기의 수식과 같이 나타낼 수 있다.

수학식 1

[0069] $$K_{h,s}$$

[0070] 수학식 1의 h는 키 트리의 깊이를 의미하고, s는 같은 깊이(h)의 노드의 순번(sequence number)을 의미한다.

[0071] $K_{0,0}$는 루트 키를 의미하며, 하나의 부모 키(루트 키 또는 서브 키) $K_{h,s}$는 4 개의 자식 키 $K_{h+1,4s+0}, K_{h+1,4s+1}, K_{h+1,4s+2}, K_{h+1,4s+3}$로 분화될 수 있다.

[0072] 실시예에 따른 키 트리는 키 확장을 통해 구성하되, 하기의 수식으로 표현한 조건을 만족하는 깊이(h)로 구성되는 것이 바람직하다.

수학식 2

[0073] $$m^{h-1} < n < m^{h}$$

[0074] 수학식 2의 m은 서브섹션의 개수를 의미하며, n은 스토리지에 구성된 데이터 블록의 개수를 의미한다. 따라서, 수학식 2를 만족하는 깊이(h)의 키 트리가 구성될 때까지 상술한 방법을 반복적으로 수행하여 키 트리를 구축할 수 있다. 예를 들어, 데이터 블록 하나의 크기가 4KB이고, 전체 2MB 크기의 스토리지가 있다고 가정하면, 그 스토리지는 512개의 데이터 블록으로 이루어져 있을 것이다(n=512). 이때, 이를 만족하는 키 트리의 깊이(h)는 $4^{h-1} < 512 < 4^{h}$ 에 의해 5가 된다. 따라서, 깊이(h)가 5인 키 트리를 형성하면 된다.

[0075] 스토리지 시스템에 구성된 하나의 데이터 블록의 크기가 16KB이고, 하나의 부모 키에서 4개의 자식 키로 분화하면서, 깊이(h)가 12인 키 트리를 구성한다고 가정할 경우, 하기의 수식과 같은 방법에 의해 256GB의 스토리지를 커버할 수 있다.

수학식 3

[0076] $$2^{14}(=16KB) * 4^{12}(\text{자식 노드의 개수}) = 2^{38}(256GB)$$

[0077] 서브 키 및 데이터 블록 매핑 단계(S430)

[0078] S430 단계에서는, 도 5에 도시된 바와 같이, S420 단계를 통해 생성된 서브 키들과 스토리지(700)에 구성된 데이터 블록(B_0 … B_x, B_{x+1}, … B_y, B_{y+1}…B_{4h-1})을 각각 매핑한다. 예를 들어, 서브 키 "$K_{h,x}$, $K_{h,x+1}$… $K_{h,y}$, $K_{h,y+1}$"은 데이터 블록 "B_x, B_{x+1}, … B_y, B_{y+1}"과 각각 매핑할 수 있다.

[0079] 암호문 스트림 생성 단계(S440)

[0080] 도 6은 실시예에 따라 암호문 스트림이 생성되어 스토리지 데이터 블록에 저장되는 과정을 도시한 도면이다.

[0081] S440 단계에서는, S430을 통해 스토리지의 데이터 블록(B_0 … B_x, B_{x+1}, … B_y, B_{y+1}…B_{4h-1})에 매핑된 각각의 서브 키들을 이용하여, 평문(plain text) 스트림을 암호화한다. 스트림 암호화 알고리즘에 기초하여 해당 데이터 블록을 암호화한다. 보다 구체적으로, 도 6에 도시된 바와 같이, 해당 데이터 블록에 매핑된 서브 키(401, 402)와, 해당 데이터 블록에 저장될 평문 스트림(501, 502)을 각각 XOR 연산하여 블록 단위로 암호화 스트림(601, 602)을 생성한다.

[0082] 도 6에 도시된 바와 같이, 평문 스트림의 첫 번째 블록은, 키 트리 구성을 통해 생성된 깊이 h의 첫 번째 서브 키($K_{h,0}$)를 이용하여 암호화되어 스토리지 시스템(700)에 구성된 첫 번째 데이터 블록(B_0)에 저장될 수 있다. 또한, 평문 스트림의 두 번째 블록은, 깊이 h의 두 번째 서브 키($K_{h,1}$)를 이용하여 암호화되어 스토리지 시스템(700)에 구성된 두 번째 데이터 블록(B_1)에 저장될 수 있다. 암호화 방법은 스트림 암호화 알고리즘에 기초한다. 서브 키 K_h는 S420 단계를 통해 구성된 키 트리에서 깊이 h에 있는 자식 노드 중 가장 왼쪽에 위치한 첫 번째 자식 노드의 서브 키에 해당하며, 서브 키 K_h는 그 다음 왼쪽에 위치한 두 번째 자식 노드의 서브 키에 해당한다. 스토리지 시스템(700)의 블록 구조에 있어서도, 데이터 블록 B_0, B_1은 가장 왼쪽부터 첫 번째, 두 번째 위치한 블록이 된다.

[0083] 실시예에서는, 암호화를 수행한 후에 아웃소싱된 스토리지를 접근(access)하는데 있어서, 클라우드 서비스 제공자(Cloud Service Provider)와 사용자가 공유한 다른 하나의 키($K_{u,s}$)가 있다고 가정하며, 사용자는 자신만이 알고 있는 키(키 트리로부터 유도한 키)를 이용하여 암호화를 한 후, 클라우드 서비스 제공자와 사용자가 공유한 다른 하나의 키($K_{u,s}$)를 이용하여, 다시 한번 암호화를 수행할 수 있다. 이와 같이 이중으로 암호화하는 이유는, 클라우드 서비스 제공자로부터 업로드 또는 다운로드 명령이 있을 경우, 다른 외부의 사용자로부터 기밀성을 보장하기 위함이다.

[0084] **2) 스트림 복호화 방법**

[0085] 도 4를 참조하면, 실시예에 따른 스트림 복호화 방법은, 서브 키 유도 단계(S450) 및 복호문 스트림 생성 단계(S460)를 포함한다.

[0086] 서브 키 유도 단계(S450)

[0087] S450 단계에서는, 스토리지(700)의 데이터 블록에 매핑된 서브 키들을 해쉬 함수를 통해 유도한다. 또한, 특정 블록에 대한 복호화가 필요한 경우, 해당 블록에 매핑된 자식 노드의 서브 키를 해쉬 함수를 이용하여 유도할 수 있다.

[0088] 복호문 스트림 생성 단계(S460)

[0089] S460 단계에서는, S450 단계를 통해 유도된 해당 서브 키를 이용하여 복호화를 수행할 수 있다. 보다 구체적으로, 유도된 해당 서브 키와 해당 블록에 저장된 복호문 스트림을 XOR 연산하여 평문 스트림을 생성한다. 이때, 복호화 방법은 스트림 암호화 알고리즘에 기초하며, 이를 통해 블록 단위로 복호문 스트림이 생성될 수 있다. 이는, 복호화 과정뿐만 아니라, 암호화 과정에도 블록 단위의 스트림 암호화가 가능하다.

[0090] 해쉬 연산은 일반적인 암호화 알고리즘에 비하여 1/10 정도의 복잡도를 가지므로 수행 속도가 매우 빠르며, 특정 블록과 매핑되어 있는 자식 노드의 서브 키를 유도하는데 있어서도 매우 적은 연산 횟수를 통해 유도할 수 있으므로, 매우 빠르게 키를 유도할 수 있다. 이에 따라, 블록 접근성을 제공할 수 있게 될 뿐만 아니라, 각 블록의 암호화를 스트림 암호화 알고리즘에 적용함으로써, 매우 빠른 연산 효율성을 제공할 수 있게 된다.

[0091] 도 7 은 실시예에 따른 스트림 암호화 알고리즘을 이용하여 파일을 저장하고 읽어오는 전반적인 과정을 도시한 도면이다.

[0092] 스토리지(700)에 특정 정보를 저장할 경우, 그 정보를 저장할 위치가 정해지면, 키 트리를 이용하여 저장할 위치에 매핑이 된 키를 유도하여, 암호화를 시킨다. 이후, 통신의 기밀성을 위하여 스토리지 서비스 제공자와 사용자가 공유한 키를 이용하여 추가적인 암호화를 시킨 후 스토리지(700)에 저장하게 된다.

[0093] 스토리지(700)로부터 특정 정보를 읽어올 경우, 그 정보가 저장되어 있는 위치를 알아내고, 아웃소싱된 스토리지로부터 저장되어 있는 정보를 얻어온다. 이후, 스토리지 서비스 제공자와 사용자가 공유한 키를 이용하여 해독한 후, 상기 명시된 키 유도 방법을 이용하여, 키를 유도하고 해독하여 원래 저장했던 데이터를 읽어 올 수 있게 된다.

[0094] 실시예에 따르면, 해쉬 함수를 이용한 키 확장을 통해 키 트리를 구축하고, 스토리지의 각 블록을 키 트리로부터 유도된 각각 다른 키로 스트림 암호화를 시킴으로써, 블록 암호화 알고리즘의 블록 접근성과 함께, 스트림 암호화 알고리즘의 연산 효율성을 제공할 수 있다.

[0095] **2. 암호 파일 시스템**

[0096] 도 8은 실시예에 따른 암호 파일 시스템(800)의 전체적인 구성을 나타낸 블록도이다.

[0097] 도 8을 참조하면, 실시예에 따른 암호 파일 시스템(800)은 스트림 암호화 시스템(810), 스트림 복호화 시스템(820), 및 스토리지(700)를 포함한다. 실시예에 따른 암호 파일 시스템(800)은, 상술한 스트림 암호화/복호화를 위해 구현된 장치이며, 도 3 내지 도 7과 결부하여 설명하도록 한다.

[0098] **1) 스트림 암호화 시스템(810)**

[0099] 도 8을 참조하면, 실시예에 따른 스트림 암호화 시스템(810)은, 루트 키 생성부(811), 키 트리 구성부(813), 매핑부(815), 및 암호화 연산부(817)를 포함한다.

[0100] 루트 키 생성부(811)

[0101] 루트 키 생성부(811)는, 평문(plain text)을 암호화하는데 필요한 키 트리를 구성하기 위하여 루트 키(Root Key, $K_{0,0}$)를 생성한다. 여기서 루트 키(Root Key, $K_{0,0}$)는 사용자만이 알고 있는 키로서, 키 트리의 최상위 노드에 있는 키에 해당된다. 단 방향 해쉬 함수는 SHA(Secure Hash Algorithm)-1과 같이 보안적으로 안전한 해쉬 함수를 사용하는 것이 바람직하다.

[0102] 키 트리 구성부(813)

[0103] 키 트리 구성부(813)는 교환 연산부(미도시) 및 키 생성부(미도시)를 포함한다.

[0104] 교환 연산부(미도시)는, 루트 키 값을 복수의 서브섹션으로 나누고, 나누어진 각각의 서브섹션을 교환 연산을 통해 서로 다른 서브섹션 값을 갖는 복수의 루트 키를 생성할 수 있다. 또한, 서브 키의 값을 복수의 서브섹션으로 나누고, 나누어진 각각의 서브섹션을 교환 연산을 통해 서로 다른 서브섹션 값을 갖는 복수의 서브 키를 생성할 수 있다.

[0105] 키 생성부(미도시)는, 교환 연산부(미도시)를 통해 생성된 복수의 루트 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성할 수 있다. 또한, 교환 연산부(미도시)를 통해 생성된 복수의 서브 키를, 해쉬 함수를 이용하여 서로 다른 값을 갖는 서브 키로 생성할 수 있다.

[0106] 이하, 교환 연산부(미도시)와 키 생성부(미도시)를 통한 키 트리 구성 방법에 관하여 보다 상세히 설명한다.

[0107] 먼저, 교환 연산부(미도시)는, 루트 키(Root Key, $K_{0,0}$) 값을 복수의 서브섹션으로 나눌 수 있다. 여기서, 서브섹션의 개수는 임의로 설정할 수 있으며, 실시예에서는 4개로 설정하여 설명하도록 한다.

[0108] 다음, 교환 연산부(미도시)는, 4개로 나누어진 서브섹션(A B C D)을 교환 연산(Swapping Operation)을 통해 교환하여, 해쉬 함수(SHA-1)에 입력될 4개의 루트 키를 생성할 수 있다. 4개의 루트 키는 서로 다른 값을 가지며, 한번의 교환 연산을 통해 생성된다. 예를 들어, 루트 키(Root Key, $K_{0,0}$)의 서브섹션 값이 A B C D 라고 가정하면, 이 서브섹션 값들 간의 위치 교환을 통해 SHA(A B C D), SHA(B A C D), SHA(A C B D), SHA(A B D C)와 같은 해쉬 함수(SHA-1)의 입력 값을 얻어낼 수 있다.

[0109] 키 생성부(미도시)는, 교환 연산부(미도시)를 통해 얻은 키 값들을 해쉬 함수(SHA-1)에 각각 입력하고, 서로 다른 값을 갖는 새로운 서브 키들($K_{1,0}$, $K_{1,1}$, $K_{1,2}$, $K_{1,3}$)을 생성할 수 있다. 이에 따라, 하나의 부모 노드에서 자식 노드로 분화됨으로써, 깊이 1의 키 트리가 구성될 수 있다.

[0110] 다음, 키 생성부(미도시)는, 해쉬 함수(SHA-1)를 통해 얻어진 각 서브 키들($K_{1,0}$, $K_{1,1}$, $K_{1,2}$, $K_{1,3}$)의 값을 다시 4개의 서브섹션으로 나눌 수 있다. 또한, 각 서브섹션 값을 교환 연산(Swapping Operation)을 통해 서로 다른 값을 갖도록 하여, 해쉬 함수(SHA-1)에 입력될 4개의 서브 키를 생성한다. 예를 들어, SHA(B A C D)를 입력 값으로 하여 해쉬 함수(SHA-1)를 통해 얻어진 서브 키가 A', B', C', D'의 서브섹션 값으로 이루어졌다고 가정

하면, 상술한 교환 연산(Swapping Operation)을 통해 'SHA(A' B' C' D'), SHA(B' A' C' D'), SHA(A' C' B' D'), SHA(A' B' D' C')와 같은 값을 얻어 낼 수 있다. 이러한 값은 다시 해쉬 함수(SHA-1)로 입력되어 새로운 서브 키들($K_{2,0}$, $K_{2,1}$, $K_{2,2}$, $K_{2,3}$)이 생성될 수 있다. 이에 따라, 어느 하나의 자식 노드에서 또 다른 자식 노드로 분화됨으로써, 깊이 2의 키 트리가 구성될 수 있다.

[0111] 키 트리 구성부(813)는 상술한 방법들을 통해 키를 확장시켜 나가면서 키 트리를 구성할 수 있다. 키 확장을 위해 새로운 서브 키 값을 얻기 위해서는, 해쉬 함수(SHA-1)에 입력될 값이 서로 달라야 하며, 이를 위해 실시예에서는 서브섹션을 교환하는 방법을 적용하였다. 그러나, 해쉬 함수(SHA-1)의 입력 값을 얻기 위한 방법이 반드시 교환 연산에 의한 방법에만 한정되는 것이 아니라, 다양한 방법들이 적용 가능하다.

[0112] 실시예에 따른 키(루트 키 또는 서브 키)는 하기의 수식과 같이 나타낼 수 있다.

수학식 4

[0113] $$K_{h,s}$$

[0114] 수학식 4의 h는 키 트리의 깊이를 의미하고, s는 같은 깊이(h)의 노드의 순번(sequence number)을 의미한다.

[0115] $K_{0,0}$는 루트 키를 의미하며, 하나의 부모 키(루트 키 또는 서브 키) $K_{h,s}$는 4 개의 자식 키 $K_{h+1,4s+0}, K_{h+1,4s+1}, K_{h+1,4s+2}, K_{h+1,4s+3}$로 분화될 수 있다.

[0116] 실시예에 따른 키 트리는 키 확장을 통해 구성하되, 하기의 수식으로 표현한 조건을 만족하는 깊이(h)로 구성되는 것이 바람직하다.

수학식 5

[0117] $$m^{h-1} < n < m^{h}$$

[0118] 수학식 5의 m은 서브섹션의 개수를 의미하며, n은 스토리지(700)에 구성된 데이터 블록의 개수를 의미한다. 따라서, 수학식 5를 만족하는 깊이(h)의 키 트리가 구성될 때까지 상술한 방법을 반복적으로 수행하여 키 트리를 구축할 수 있다. 예를 들어, 데이터 블록 하나의 크기가 4KB이고, 전체 2MB 크기의 스토리지가 있다고 가정하면, 그 스토리지는 512개의 데이터 블록으로 이루어져 있을 것이다(n=512). 이때, 이를 만족하는 키 트리의 깊이(h)는 $4^{h-1} < 512 < 4^{h}$에 의해 5가 된다. 따라서, 깊이(h)가 5인 키 트리를 형성하면 된다.

[0119] 스토리지 시스템에 구성된 하나의 데이터 블록의 크기가 16KB이고, 하나의 부모 키에서 4개의 자식 키로 분화하면서, 깊이(h)가 12인 키 트리를 구성한다고 가정할 경우, 하기의 수식과 같은 방법에 의해 256GB의 스토리지를 커버할 수 있다.

수학식 6

[0120] $$2^{14}(=16KB) * 4^{12}(\text{자식 노드의 개수}) = 2^{38}(256GB)$$

[0121] 매핑부(815)

[0122] 매핑부(815)는 도 5에 도시된 바와 같이, 키 트리 구성부(813)를 통해 생성된 서브 키들과 스토리지(700)에 구성된 데이터 블록(B_0 … B_x, B_{x+1}, … B_y, B_{y+1}…B_{4h-1})을 각각 매핑한다. 예를 들어, 서브 키 "$K_{h,x}$, $K_{h,x+1}$… $K_{h,y}$, $K_{h,y+1}$"은 데이터 블록 "B_x, B_{x+1}, … B_y, B_{y+1}"과 각각 매핑할 수 있다.

[0123] 암호화 연산부(817)

[0124] 암호화 연산부(817)는, 스토리지의 데이터 블(B_0 … B_x, B_{x+1}, … B_y, B_{y+1}…B_{4h-1})에 매핑된 각각의 서브 키들을 이용하여, 평문(plain text) 스트림을 암호화할 수 있다. 스트림 암호화 알고리즘에 기초하여 해당 데이터 블록을 암호화한다. 보다 구체적으로, 도 6에 도시된 바와 같이, 해당 데이터 블록에 매핑된 서브 키(401, 402)와, 해당 데이터 블록에 저장될 평문 스트림(501, 502)을 각각 XOR 연산하여 블록 단위로 암호화 스트림(601, 602)을 생성할 수 있다.

[0125] **2) 스트림 복호화 시스템(820)**

[0126] 실시예에 따른 스트림 복호화 시스템(810)은, 키 유도부(821) 및 복호화 연산부(823)를 포함한다.

[0127] 키 유도부(821)

[0128] 키 유도부(821)는, 스토리지(700)의 데이터 블록에 매핑된 서브 키들을 해쉬 함수를 통해 유도한다. 또한, 특정 블록에 대한 복호화가 필요한 경우, 해당 블록에 매핑된 자식 노드의 서브 키를 해쉬 함수를 이용하여 유도할 수 있다.

[0129] 복호화 연산부(823)

[0130] 복호화 연산부(823)는, 키 유도부(821)를 통해 유도된 해당 서브 키를 이용하여 복호화를 수행할 수 있다. 보다 구체적으로, 유도된 해당 서브 키와 해당 블록에 저장된 복호문 스트림을 XOR 연산하여 평문 스트림을 생성한다. 이때, 스트림 암호화 알고리즘에 기초하여 복호화를 수행하며, 이를 통해 블록 단위로 복호문 스트림을 생성할 수 있다. 이는, 복호화 과정뿐만 아니라, 암호화 과정에도 블록 단위의 스트림 암호화가 가능하다.

[0131] 도 9는 실시예에 따른 스트림 암호화 알고리즘(BA-RC4), 종래의 블록 암호화 알고리즘(AES), 및 스트림 암호화 알고리즘(RC4)의 성능 비교를 나타낸 그래프이다.

[0132] 도 9의 그래프에서 가로축은 스토리지에 접근(access)하는 시작 주소를 의미하고, 세로축은 처리량(throughput)을 나타내는 것으로, 처리량(throughput)은 높으면 높을수록 좋다. 본 실험을 통해 얻은 그래프는, 1MB의 파일을 읽고, 스토리지의 한 블록을 16KB라고 가정할 경우, 접근하는 시작 주소를 변화시켜 가며 처리량(throughput)을 측정한 것이다.

[0133] 블록 암호화 알고리즘(AES)의 경우, 블록 단위로 암호화/복호화를 수행하므로, 접근하는 위치와 관계없이 일정한 처리량(throughput)을 보인다.

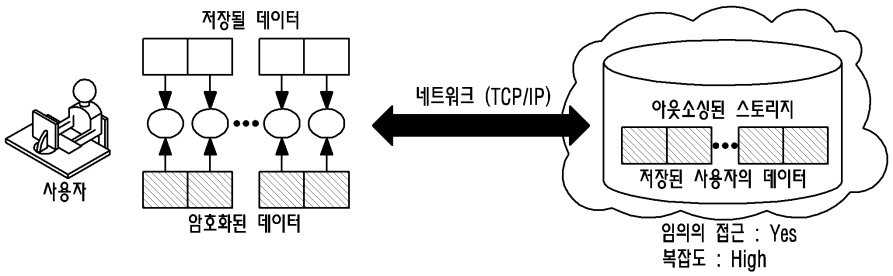
[0134] 반면, 스트림 암호화 알고리즘(RC4)의 경우, 접근하는 시작 주소 위치(수치)가 증가함에 따라 처리량(throughput)이 점차 감소하게 된다. 스트림 암호화 알고리즘(RC4)은, 특정 블록을 읽기 위해, 해당 블록을 암호화하는데 사용되었던 난수열을 생성해야 하고, 그러기 위해, 초기 키 값을 이용하여 해당되는 상태가 나올 때까지 난수열을 생성해야 해독할 수 있으므로, 무작위 접근성이 매우 낮다는 단점을 갖는다. 이에 따라, 접근 위치가 멀어질수록 키를 유도하는데 발생되는 시간이 점차 증가하기 때문에, 전체적인 처리량(throughput) 또한 감소되는 것을 알 수 있다.

[0135] 이상에서 보는 바와 같이, 본 발명이 속하는 기술 분야의 당업자는 본 발명이 그 기술적 사상이나 필수적 특징을 변경하지 않고서 다른 구체적인 형태로 실시 될 수 있다는 것을 이해할 수 있을 것이다.

[0136] 그러므로, 이상에서 기술한 실시 예는 모든 면에서 예시적인 것이며 한정적인 것이 아닌 것으로 이해해야만 하고, 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 등가개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

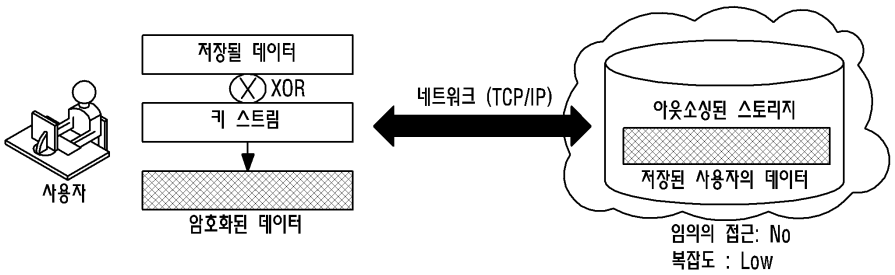
도면

도면1



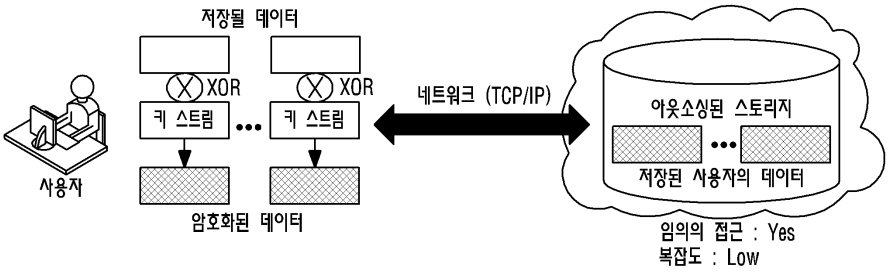
블록 암호화 기반의 보안 스토리지

도면2



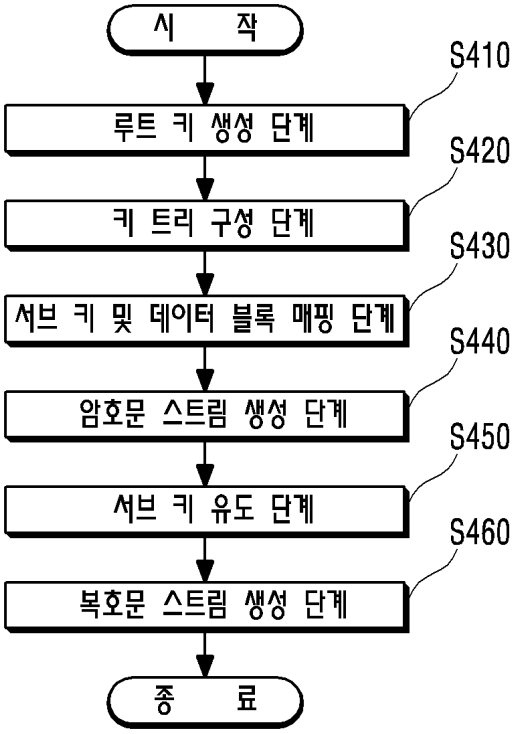
스트림 암호화 기반의 보안 스토리지

도면3

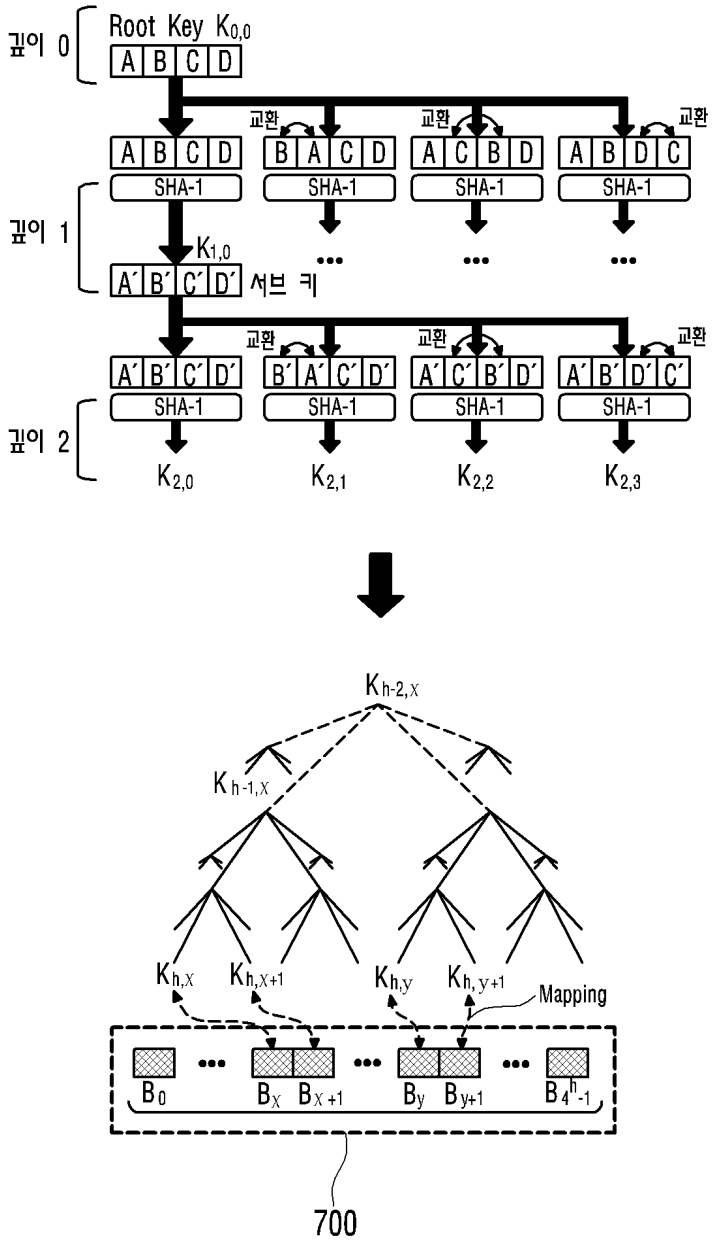


BA-RC4 기반의 보안 스토리지

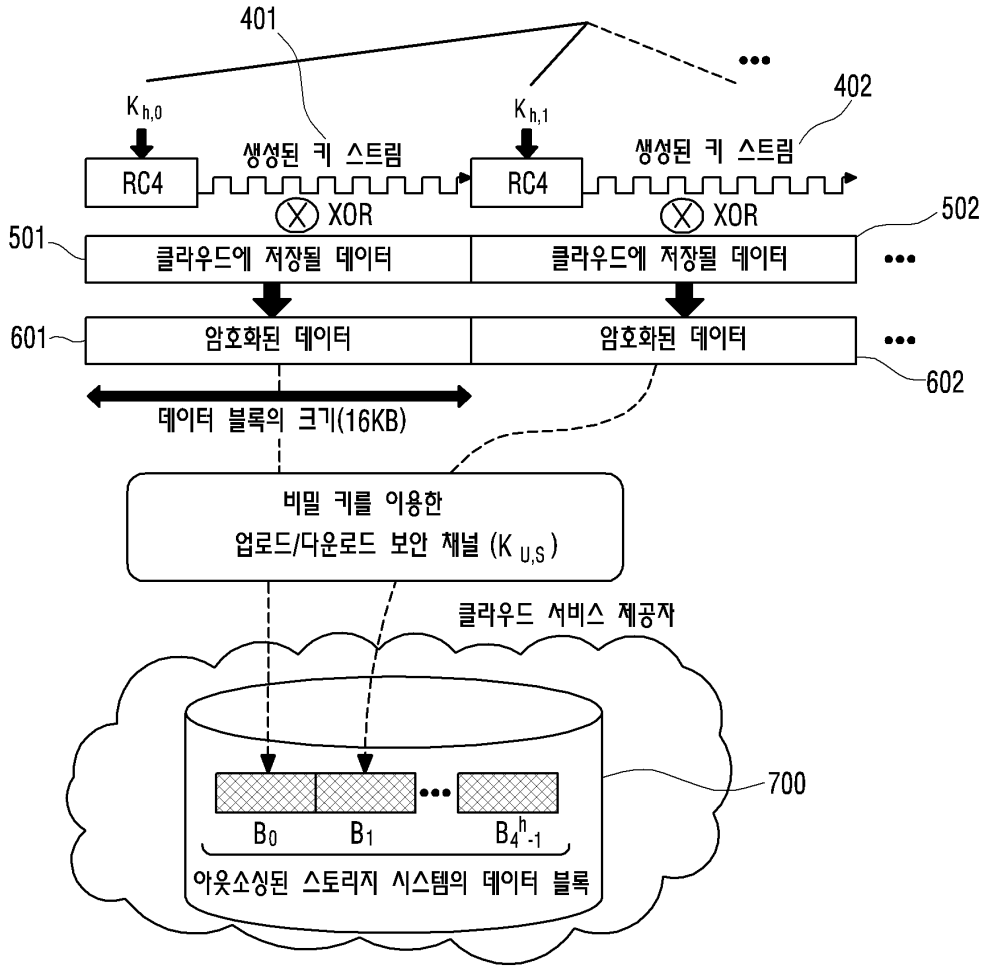
도면4


시 작
S410
루트 키 생성 단계
S420
키 트리 구성 단계
S430
서브 키 및 데이터 블록 매핑 단계
S440
암호문 스트림 생성 단계
S450
서브 키 유도 단계
S460
복호문 스트림 생성 단계
종 료

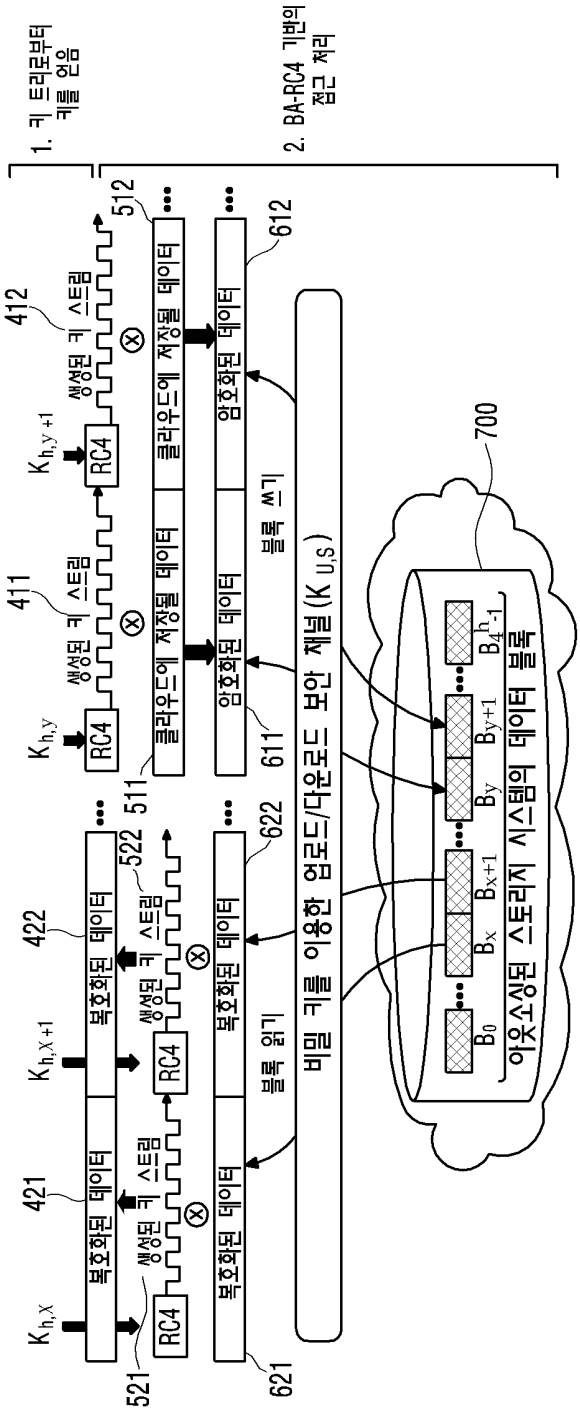
도면5


깊이 0
Root Key $K_{0,0}$
A B C D
교환
SHA-1
깊이 1
$K_{1,0}$
A´ B´ C´ D´ 서브 키
깊이 2
$K_{2,0}$
$K_{2,1}$
$K_{2,2}$
$K_{2,3}$
$K_{h-2,X}$
$K_{h-1,X}$
$K_{h,X}$
$K_{h,X+1}$
$K_{h,y}$
$K_{h,y+1}$
Mapping
B_0
B_X B_{X+1}
B_y B_{y+1}
B_{4^h-1}
700

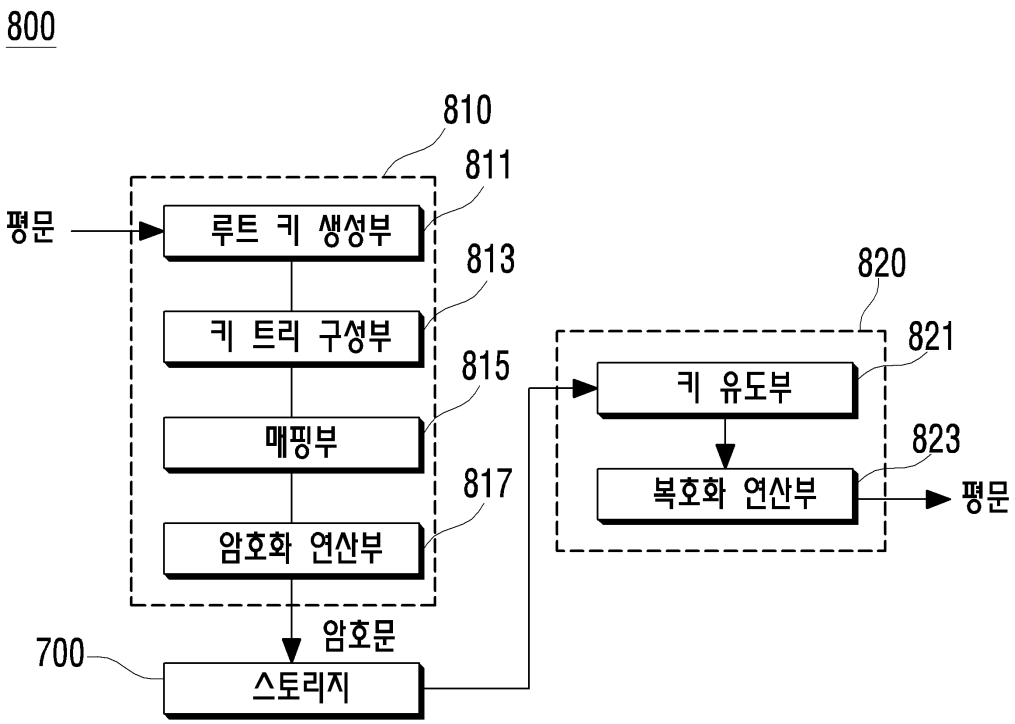
도면6


401
402
$K_{h,0}$
$K_{h,1}$
생성된 키 스트림
생성된 키 스트림
RC4
RC4
XOR
XOR
501
502
클라우드에 저장될 데이터
클라우드에 저장될 데이터
601
암호화된 데이터
암호화된 데이터
602
데이터 블록의 크기(16KB)
비밀 키를 이용한
업로드/다운로드 보안 채널 ($K_{U,S}$)
클라우드 서비스 제공자
700
B_0
B_1
B_4^h-1
아웃소싱된 스토리지 시스템의 데이터 블록

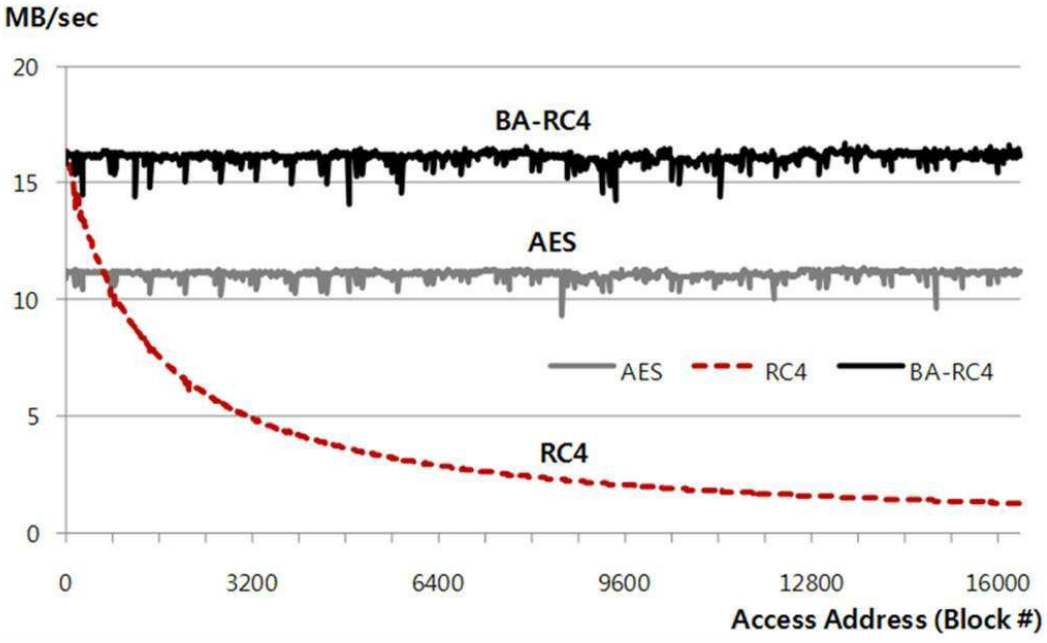
도면7


1. 키 트래로부터 키를 얻음
2. BA-RC4 기반의 접근 처리
$K_{h,X}$
$K_{h,X+1}$
$K_{h,y}$
$K_{h,y+1}$
RC4
생성된 키 스트림
복호화된 데이터
암호화된 데이터
클라우드에 저장될 데이터
421
422
411
412
511
512
521
522
611
612
621
622
블록 읽기
블록 쓰기
비밀 키를 이용한 업로드/다운로드 보안 채널 ($K_{U,S}$)
700
아웃소싱된 스토리지 시스템의 데이터 블록
B_0
B_x
B_{x+1}
B_y
B_{y+1}
B_{4^h-1}

도면8


800
810
811
평문
루트 키 생성부
813
키 트리 구성부
815
매핑부
817
암호화 연산부
암호문
700
스토리지
820
821
키 유도부
823
복호화 연산부
평문

도면9


MB/sec
20
15
10
5
0
BA-RC4
AES
RC4
AES
RC4
BA-RC4
0
3200
6400
9600
12800
16000
Access Address (Block #)