(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0131738
(43) 공개일자 2012년12월05일

(51) 국제특허분류(Int. Cl.)
H04L 9/06 (2006.01) *H04L 9/28* (2006.01)
(21) 출원번호 10-2011-0050122
(22) 출원일자 2011년05월26일
심사청구일자 2011년05월26일

(71) 출원인
고려대학교 산학협력단
서울 성북구 안암동5가 1
(72) 발명자
이동훈
서울특별시 종로구 사직로8길 34, 경희궁의아침3단지 아파트 1404호 (내수동)
박진형
서울특별시 송파구 올림픽로34길 27-20 (방이동)
백정하
서울특별시 송파구 송파대로32길 15, 105동 802호 (가락동, 가락금호아파트)
(74) 대리인
특허법인충현

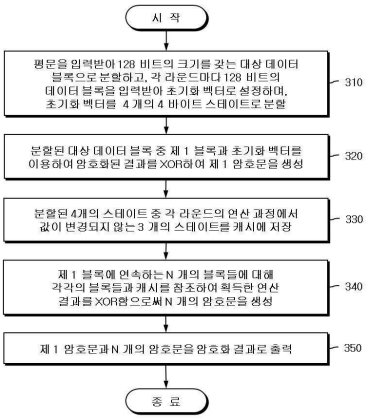
전체 청구항 수 : 총 11 항

(54) 발명의 명칭 **AES의 CTR 모드에 따른 암호화 장치 및 방법**

(57) 요 약

본 발명은 AES의 CTR 모드에 따른 암호화 장치 및 방법에 관한 것으로, 본 발명에 따른 암호화 방법은 평문을 입력받아 128 비트의 크기를 갖는 대상 데이터 블록으로 분할하고, 각 라운드마다 128 비트의 데이터 블록을 입력받아 초기화 벡터로 설정하고, 초기화 벡터를 4 개의 4 바이트 스테이트로 분할하고, 분할된 대상 데이터 블록 중 제 1 블록과 초기화 벡터를 이용하여 암호화된 결과를 XOR하여 제 1 암호문을 생성하고, 분할된 4 개의 스테이트 중 각 라운드의 연산 과정에서 값이 변경되지 않는 스테이트들을 캐시에 저장하고, 제 1 블록에 연속하는 복수 개의 블록들에 대해 각각의 블록들과 캐시를 참조하여 획득한 연산 결과를 XOR함으로써 복수 개의 암호문을 생성하며, 생성된 제 1 암호문과 복수 개의 암호문을 암호화 결과로 출력한다.

대 표 도 - 도3



이 발명을 지원한 국가연구개발사업
과제고유번호 20100020726
부처명 한국연구재단
연구사업명 원천기술개발사업
연구과제명 개인 프라이버시 보장 원격 접근기술(총괄:super mobile 구현을 위한 시스템 SW 원천기술 연구)
주관기관 성균관대학교
연구기간 2010.07.01 ~ 2011.06.30

특허청구의 범위

청구항 1

AES(Advanced Encryption Standard)의 CTR 모드(counter mode)에 따른 암호화 방법에 있어서,

평문을 입력받아 128 비트(bit)의 크기를 갖는 대상 데이터 블록으로 분할하고, 각 라운드(round)마다 128 비트의 데이터 블록을 입력받아 초기화 벡터(initialization vector)로 설정하며, 상기 설정된 초기화 벡터를 4 개의 4 바이트(byte) 스테이트(state)로 분할하는 단계;

상기 분할된 대상 데이터 블록 중 제 1 블록과 상기 설정된 초기화 벡터를 이용하여 암호화된 결과를 XOR하여 제 1 암호문을 생성하는 단계;

상기 분할된 4 개의 스테이트 중 상기 각 라운드의 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시(cache)에 저장하는 단계;

상기 제 1 블록에 연속하는 N 개(N은 양의 정수)의 블록들에 대해 각각의 블록들과 상기 캐시를 참조하여 획득한 연산 결과를 XOR함으로써 N 개의 암호문을 생성하는 단계; 및

상기 생성된 제 1 암호문과 상기 생성된 N 개의 암호문을 암호화 결과로 출력하는 단계를 포함하는 방법.

청구항 2

제 1 항에 있어서,

상기 라운드가 AES 초기화 라운드(initial round)인 경우,

상기 3 개의 스테이트를 캐시에 저장하는 단계는, 상기 초기화 라운드의 연산 결과인 16 바이트 중, 마지막 4 바이트를 제외한 12 바이트를 상기 캐시에 저장하는 것을 특징으로 하는 방법.

청구항 3

제 2 항에 있어서,

상기 N 개의 암호문을 생성하는 단계는,

상기 제 1 블록에 연속하는 2^{32}-1 개의 블록들에 대해 상기 캐시에 저장된 12 바이트를 독출하는 단계;

2^{32}-1 개의 블록들에 대해 각각의 카운터 값의 변화 규칙을 고려하여 XOR 연산 결과의 4 바이트를 산출하는 단계;

상기 독출된 12 바이트와 상기 산출된 4 바이트를 결합하여 해당 데이터 블록에 대한 연산 결과를 생성하는 단계; 및

상기 생성된 연산 결과에 기초하여 2^{32}-1 개의 암호문을 생성하는 단계를 포함하는 방법.

청구항 4

제 1 항에 있어서,

상기 라운드가 AES 라운드 1(round 1)인 경우,

상기 3 개의 스테이트를 캐시에 저장하는 단계는, 상기 라운드 1의 연산 결과인 16 바이트 중, 처음 4 바이트를 제외한 12 바이트를 상기 캐시에 저장하는 것을 특징으로 하는 방법.

청구항 5

제 4 항에 있어서,

상기 N 개의 암호문을 생성하는 단계는,

상기 제 1 블록에 연속하는 2^8-1 개의 블록들에 대해 상기 캐시에 저장된 12 바이트를 독출하는 단계;

상기 라운드 1의 연산 과정을 고려하여 연산 결과의 4 바이트를 산출하는 단계;

상기 독출된 12 바이트와 상기 산출된 4 바이트를 결합하여 해당 데이터 블록에 대한 연산 결과를 생성하는 단계; 및

상기 생성된 연산 결과에 기초하여 2^8-1 개의 암호문을 생성하는 단계를 포함하는 방법.

청구항 6

제 4 항에 있어서,

상기 캐시에 저장되지 않는 처음 4 바이트에 대하여 상기 AES 라운드 1에 선행하는 AES 초기화 라운드에서 소정 연산에 대한 결과를 미리 산출하여 치환 테이블에 저장하는 단계를 더 포함하고,

상기 AES 라운드 1에서는 상기 저장된 4 바이트의 치환 테이블을 독출하여 상기 카운터 값을 산출하는 것을 특징으로 하는 방법.

청구항 7

제 6 항에 있어서,

상기 소정 연산은 대상 블록의 컬럼(column)이 소정 다항식에 의해 치환되는 MixColumns() 연산 및 상기 MixColumns() 연산의 출력과 라운드 키(round key)의 XOR 값을 산출하는 AddRoundKey() 연산인 것을 특징으로 하는 방법.

청구항 8

제 6 항에 있어서,

상기 카운터 값에서 캐리(carry)가 발생하는 경우 새롭게 산출된 연산 결과에 따라 상기 저장된 치환 테이블을 갱신하고,

상기 갱신된 치환 테이블은 갱신 이후, 연속하는 $2^{40}\times16$ 바이트의 입력 데이터에 대해 상기 치환 테이블로부터 독출된 4 바이트의 값을 이용하여 상기 소정 연산 없이 상기 카운터 값을 산출하는 것을 특징으로 하는 방법.

청구항 9

제 1 항에 있어서,

상기 라운드가 AES 라운드 2(round 2)인 경우,

상기 3 개의 스테이트를 캐시에 저장하는 단계는, 상기 라운드 2의 연산 중, 대상 블록의 컬럼이 소정 다항식에 의해 치환되는 MixColumns() 연산 과정에서 인접한 블록 간에 값이 변경되지 않는 12 바이트의 XOR 결과를 상기 캐시에 저장하는 것을 특징으로 하는 방법.

청구항 10

제 9 항에 있어서,

상기 N 개의 암호문을 생성하는 단계는,

상기 제 1 블록에 연속하는 2^8-1 개의 블록들에 대해 상기 캐시에 저장된 12 바이트를 독출하는 단계;

상기 라운드 1의 연산 과정을 고려하여 상기 MixColumns() 연산 과정에서 인접한 블록 간에 값이 변경되는 4 바이트의 XOR 결과를 산출하는 단계;

상기 독출된 12 바이트와 상기 산출된 4 바이트를 결합하여 상기 MixColumns() 연산의 결과값으로 설정하는 단계;

상기 설정된 결과값에 기초하여 해당 데이터 블록에 대한 연산 결과를 생성하는 단계; 및

상기 생성된 연산 결과에 기초하여 2^8-1 개의 암호문을 생성하는 단계를 포함하는 방법.

청구항 11

제 1 항 내지 제 9 항 중에 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

명 세 서

기 술 분 야

[0001] 본 발명은 AES 규격에 따른 암호화 장치 및 방법에 관한 것으로, 특히 평문 데이터의 양이 암호화를 위한 데이터의 크기보다 큰 경우 사용하는 블록 암호화 방법의 다양한 모드들 중, 인터넷 스트리밍 서비스 등에서 안전한 정보의 전송을 위해 사용되는 AES의 CTR 모드를 이용한 암호화 장치, 방법 및 그 방법을 기록한 기록매체에 관한 것이다.

배 경 기 술

[0002] 최근 네트워크 기술의 급속한 발전과 함께 인터넷 사용자 수가 급속히 증가하였다. 이에 따라 이러한 네트워크를 이용한 콘텐츠 전송 및 인터넷 전화(VoIP)등 과 같은 인터넷 서비스들이 증가하고 있다. 이러한 인터넷 서비스들이 유?무선망을 통해 유료 서비스 및 개인 통화 정보 등을 전송할 경우, 그 내용이 제 3자에게 쉽게 노출되지 않도록 전송되는 정보의 기밀성을 유지하고 실시간 처리를 지원할 수 있는 정보 보호 기술이 적용되어야 한다.

[0003] 이러한 정보 보호 기술과 관련하여 20년이 넘게 오랫동안 사용되어 온 암호화 알고리즘은 DES(Data Encryption Standard)는 1972년에 미국 상무성 산하 NIST (National Institute of Standards and Technology)의 전신인 NBS(National Bureau of Standards)에서 컴퓨터 데이터를 보호할 목적으로 표준 알고리즘을 공모하여 IBM사가 개발한 암호 알고리즘이다. DES는 연방 표준으로 제정된 후에 5년마다 안정성을 인정받으면서 표준으로 존속되어 왔으나, 1997년 이후 안정성에 대한 논란이 대두되자 NIST가 DES를 대체할 차세대 표준 암호 알고리즘 제정을 위한 프로젝트를 추진하였다.

[0004] 그 프로젝트 결과 새로운 차세대 표준 암호화 알고리즘으로서 제시된 것이 바로 AES(Advanced Encryption Standard)이다. AES는 로열티 없이도 사용할 수 있는 국제 표준 대칭키 암호화 알고리즘으로서, 알고리즘에서 사용하는 블록 크기는 128 비트(bit)이며, 128, 192, 256 비트의 키(key)를 사용할 수 있다. 이러한 키의 크기에 따라 AES-128, AES-192, AES-256으로 구별되며, 각각 10, 12, 14 라운드(round)로 동작한다. 이러한 각각의 라운드 일련의 연산들로 구성되는데, 그러한 일련의 연산은 때로는 복잡하고 수행에 시간이 많이 소요되므로, 인터넷 스트리밍 데이터를 처리해야 하는 기술 분야에 있어서는, 암호화 과정의 신속한 처리가 매우 중요하다.

[0005] 한편, OMA(Open Mobile Alliance)의 DRM(Digital Right Management) V2.0 은 모바일 기기에서의 콘텐츠 보호를 위해 음악이나 동영상과 같은 연속적인 미디어를 위한 PDCF 포맷에 암호화를 위한 AES-CTR 모드(mode)를 포함하고 있다. 또한 서비스의 특성상 고속의 암/복호화와 높은 안전성이 요구되는 IPTV 및 VoIP 서비스에는 실시간으로 전송되는 정보를 안전하게 보호하기 위해 Secure RTP(Real time Transport protocol) 기술이 적용되고 있다. Secure RTP는 보안이 고려되지 않은 기존의 RTP에서 정보전송의 안전성을 향상시킨 것으로 AES CTR Mode를 사용한다. 이와 같이 AES CTR Mode는 OMA DRM, IPTV 그리고 VoIP 등의 스트리밍 서비스에 사용되며 정보의 기밀성 확보와 효율적인 암/복호화 처리를 지원한다. 하지만 IPTV의 셋탑 박스나 DRM 서비스 혹은 VoIP 서비스 등을 이용하는 모바일 디바이스들은 제한된 자원을 가지고 있으므로, 이러한 환경에 사용되는 암호화 알고리즘은 효율성을 고려하여 구현되어야 한다.

발명의 내용

해결하려는 과제

[0006] 본 발명이 해결하고자 하는 기술적 과제는 인터넷 스트리밍 데이터를 암호화/복호화하는 과정에서 AES 알고리즘의 일련의 연산으로 인해 실시간 처리가 지연되는 한계를 극복하고, 제한된 자원을 갖는 환경에서 이들 연산을 처리하기 위해 많은 컴퓨팅 자원이 소모되는 문제점을 해결하고자 한다.

과제의 해결 수단

[0007] 상기 기술적 과제를 해결하기 위하여, 본 발명에 따른 AES(Advanced Encryption Standard)의 CTR 모드(counter mode)에 따른 암호화 방법은, 평문을 입력받아 128 비트(bit)의 크기를 갖는 대상 데이터 블록으로 분할하고, 각 라운드(round)마다 128 비트의 데이터 블록을 입력받아 초기화 벡터(initialization vector)로 설정하며, 상기 설정된 초기화 벡터를 4 개의 4 바이트(byte) 스테이트(state)로 분할하는 단계; 상기 분할된 대상 데이터 블록 중 제 1 블록과 상기 설정된 초기화 벡터를 이용하여 암호화된 결과를 XOR하여 제 1 암호문을 생성하는 단계; 상기 분할된 4 개의 스테이트 중 상기 각 라운드의 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시(cache)에 저장하는 단계; 상기 제 1 블록에 연속하는 N 개(N은 양의 정수)의 블록들에 대해 각각의 블록들과 상기 캐시를 참조하여 획득한 연산 결과를 XOR함으로써 N 개의 암호문을 생성하는 단계; 및 상기 생성된 제 1 암호문과 상기 생성된 N 개의 암호문을 암호화 결과로 출력하는 단계를 포함한다.

[0008] 상기된 암호화 방법에서 상기 라운드가 AES 초기화 라운드(initial round)인 경우, 상기 3 개의 스테이트를 캐시에 저장하는 단계는, 상기 초기화 라운드의 연산 결과인 16 바이트 중, 마지막 4 바이트를 제외한 12 바이트를 상기 캐시에 저장한다.

[0009] 상기된 암호화 방법에서 상기 라운드가 AES 라운드 1(round 1)인 경우, 상기 3 개의 스테이트를 캐시에 저장하는 단계는, 상기 라운드 1의 연산 결과인 16 바이트 중, 처음 4 바이트를 제외한 12 바이트를 상기 캐시에 저장한다.

[0010] 또한, 상기 라운드가 AES 라운드 1(round 1)인 경우의 암호화 방법은 상기 캐시에 저장되지 않는 처음 4 바이트에 대하여 상기 AES 라운드 1에 선행하는 AES 암호 초기화 라운드에서 소정 연산에 대한 결과를 미리 산출하여 치환 테이블에 저장하는 단계를 더 포함하고, 상기 AES 라운드 1에서는 상기 저장된 4 바이트의 치환 테이블을 독출하여 상기 카운터 값을 산출하는 것이 바람직하다.

[0011] 상기된 암호화 방법에서 상기 라운드가 AES 라운드 2(round 2)인 경우, 상기 3 개의 스테이트를 캐시에 저장하는 단계는, 상기 라운드 2의 연산 중, 대상 블록의 컬럼이 소정 다항식에 의해 치환되는 MixColumns() 연산 과정에서 인접한 블록 간에 값이 변경되지 않는 12 바이트의 XOR 결과를 상기 캐시에 저장한다.

[0012] 나아가, 또한, 이하에서는 상기 기재된 AES의 CTR 모드에 따른 암호화 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공한다.

발명의 효과

[0013] 본 발명은 각 라운드의 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시에 저장하여 참조함으로써 인터넷 스트리밍 데이터를 암호화/복호화하는 과정에서 AES 알고리즘의 일련의 연산을 단축하여 실시간 처리 속도를 향상시키고, 제한된 자원을 갖는 환경에서 이들 연산을 처리하기 위해 소모되는 컴퓨팅 자원을 절약함으로써 암호화 효율을 높일 수 있다.

도면의 간단한 설명

[0014] 도 1은 AES의 대상 데이터 블록을 4 개의 스테이트로 분할하는 과정을 설명하기 위한 도면이다.

도 2는 AES의 CTR 모드의 암호화 과정을 설명하기 위한 도면이다.

도 3은 본 발명의 일 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법을 도시한 흐름도이다.

도 4는 본 발명의 일 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 캐시를 이용하여 AES 초기화 라운드의 연산 과정을 단축하는 과정을 설명하기 위한 도면이다.

도 5는 본 발명의 다른 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 캐시를 이용하여 AES 라운드 1의 연산 과정을 단축하는 과정을 설명하기 위한 도면이다.

도 6은 본 발명의 또 다른 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 치환 테이블을 이용하여 AES 라운드 1의 연산 과정을 단축하는 과정을 설명하기 위한 도면이다.

도 7은 본 발명의 또 다른 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 캐시를 이용하여 AES 라운드 2의 연산 과정을 단축하는 과정을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0015] 본 발명의 실시예들을 설명하기에 앞서 실시예들이 활용되는 AES 암호화 분야, 특히 CTR 모드에 대해 개괄적으로 소개하고, 실시예들이 공통적으로 채용하고 있는 기본 아이디어를 제시하고자 한다.

[0016] 통상적으로 암호화하고자 하는 대상 데이터의 양이 암호화를 위해 설정된 데이터의 크기보다 큰 경우 대상 데이터를 일정한 크기의 집합, 즉 블록으로 분할하여 암호화 처리를 하게 된다. AES 알고리즘은 이러한 블록 암호화 기법을 활용하고 있으며, 특히 인터넷 스트리밍 데이터와 같이 그 크기가 매우 큰 대상 데이터의 경우 이러한 블록 암호화 방식이 유용하게 활용될 수 있다.

[0017] AES 알고리즘의 라운드는 4 가지 변환, SubBytes(), ShiftRows(), MixColumns(), AddRoundKey()을 선택적으로 포함하며, 라운드를 반복할수록 안전성이 증가하는 특징을 갖는다. 이상의 4 가지 변환은 스테이트(state)라고 불리는 4×4 행렬로 구성된 16 바이트(byte) 블록을 사용하여 동작하며, 각 라운드에 사용되는 라운드 키(round key)는 S-box와 XOR, Rotation으로 구성된 일련의 프로세스(키 스케줄링을 의미한다.)에 의해 비밀키로부터 생성된다.

[0018] 도 1은 AES의 대상 데이터 블록을 4 개의 스테이트로 분할하는 과정을 설명하기 위한 도면으로서, 16 바이트의 크기를 갖는 블록(110)을 다시 4 바이트 단위의 4 개의 스테이트(120)로 분할하여 처리하는 변환 과정을 도시하고 있다. 보다 구체적으로, AES의 데이터 블록은 128 비트로 구성되며 16 바이트를 원소로 하는 1×16 행렬(110)로 표현할 수 있다. AES는 여러 횟수의 라운드를 사용하며, 각 라운드는 이상에서 소개한 4 가지 변환들로 구성된다. 데이터 블록은 각 단계에서 다른 단계로 이동함에 따라 데이터가 변형되는데, 각 단계 전후에 있는 데이터 블록을 스테이트(state)라고 정의한다. 데이터 블록과 마찬가지로 스테이트는 전체 16 바이트로 구성되며, 바이트를 성분으로 하는 4×4 행렬(120)로 나타낸다.

[0019] 알고리즘의 시작 부분에서 데이터 블록의 바이트들은 도 1과 같이 하나의 스테이트에 열 단위로 삽입되고, 각 열에 대해서는 위에서 아래로 삽입된다. 스테이트(120)는 4 열로 구성되어 있는데, 왼쪽에서부터 열 단위로 State[0], State[1], State[2], State[3]으로 표현된다. 이하에서 소개될 본 발명의 실시예들은 이렇게 분리된 각각의 스테이트들을 단위로 연산 과정 중, 일부 데이터를 캐시에 저장하여 처리한다.

[0020] 도 2는 AES의 CTR 모드의 암호화 과정을 설명하기 위한 도면으로서, 본 발명의 실시예들이 공통적으로 채택하고 있는 암호화 처리 방법의 일례를 제시하고 있다. 앞서 간단히 설명한 바와 같이 평문의 데이터 크기가 큰 경우 활용될 수 있는 블록 단위의 반복 암호화 기법에는 그 구체적인 방법에 따라 다양한 모드(mode)가 제시되어 있는데, 본 발명의 실시예들은 공통적으로 CTR 모드에 따라 암호화를 수행한다.

[0021] CTR 모드는 암호화/복호화가 동일한 구조이며, 카운터(Counter)를 사용함으로써 키 스트림(Key Stream)이 의사 난수성을 갖는다. 도 2에 나타난 것처럼 카운터라고 불리는 입력 값이 암호 알고리즘에 의해 암호화된 결과가 평문과 XOR되어 암호문을 생성한다. n-bit의 카운터는 미리 정해진 값(초기화 벡터(IV, initialization vector)를 의미한다.)으로 초기화되고, 미리 정해진 규칙에 따라 값을 증가시킨다. 일련의 카운터 값들은 암호 알고리즘의 입력으로 동일한 값이 사용되면 안 되며, 각각의 블록에 모두 다른 값이 사용되어야 한다. 동일한 키로 암호화되는 모든 메시지에 대해서 카운터 값은 반드시 서로 구별되어야 한다.

[0022] AES-CTR 모드는 최초 블록(210)에 입력된 IV(211)를 기점으로 매 블록마다 LSB를 1씩 증가시켜 다음 블록의 입력으로 사용한다. 최초의 블록(210)에 대해 입력된 IV(211)를 이용하여 AES 암호화 키 스트림(212)을 생성하고, 이를 128 비트의 평문(213)과 XOR(exclusive or, 배타적 논리합)함으로써 128 비트의 암호문(214)를 생성한다. 최초의 블록(210)과 마찬가지로 이에 연속하는 다음 블록들(220, 230)에 대해서도 동일한 연산을 수행한다.

[0023] 이 과정에서, AES 알고리즘의 첫 번째 연산은 AddRoundKey()로서 입력된 데이터 블록을 그대로 키와 XOR 연산에 적용한다. 단일 블록의 길이가 16 바이트인 AES 알고리즘이기 때문에 LSB로부터 1씩 증가한 카운터가 나머지 전체 블록에 영향을 주기 위해서는 상당한 양의 데이터 프로세싱을 거쳐야 한다. 따라서, 이하에서 소개될 본 발명의 다양한 실시예들은 알고리즘 내부적으로 매번 같은 입력 값을 AddRoundKey()를 비롯한 다양한 연산에 직접적으로 적용하지 않고, 이전의 블록에서 이미 계산된 결과를 재사용하여 그 연산의 효율을 향상시키고자 하는 노력에서 안출된 것이다. 또한, 이하의 실시예들을 통해 AddRoundKey() 뿐만 아니라, 첫 번째 라운드에 적용되는 SubBytes(), ShiftRows(), MixColumns()의 특성을 잘 활용한다면 추가적인 연산결과의 재사용이 가능하다는 점을 제시하고자 한다.

[0024] 도 3은 본 발명의 일 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법을 도시한 흐름도로서, 다음과 같은 단

계들을 포함한다.

[0025] 310 단계에서 평문을 입력받아 128 비트(bit)의 크기를 갖는 대상 데이터 블록으로 분할하고, 각 라운드(round)마다 128 비트의 데이터 블록을 입력받아 초기화 벡터(initialization vector)로 설정하며, 상기 설정된 초기화 벡터를 4 개의 4 바이트(byte) 스테이트(state)로 분할한다. 이 과정은 CTR 모드를 위한 대상 데이터 및 연산을 위한 기초 데이터를 가공하는 단계에 해당한다.

[0026] 이 과정은 데이터 처리(초기화 벡터 설정 및 스테이트 분할이 될 수 있다.)를 위해 적어도 하나 이상의 프로세서(processor)가 활용될 수 있으며, 처리된 데이터를 저장하는 적어도 하나 이상의 저장공간(memory)을 통해 구현될 수 있을 것이다.

[0027] 320 단계에서는 310 단계를 통해 분할된 대상 데이터 블록 중 제 1 블록과 310 단계를 통해 설정된 초기화 벡터를 이용하여 암호화된 결과를 XOR하여 제 1 암호문을 생성한다. 즉, 이 과정은 앞서 소개한 도 2의 최초의 블록을 암호화하는 과정에 대응한다. 이 과정 역시 암호화 과정에서 사용되는 4 개의 연산을 처리하기 위해 적어도 하나 이상의 프로세서와 저장공간을 통해 구현될 수 있다. 이상과 같이 최초의 블록을 암호화하는 과정은 통상적인 AES-CTR과 대부분 동일하나, 일단 최초의 블록에 대한 암호화 과정이 수행되고 나면, 최초의 블록에 연속하는 여타의 블록들을 암호화 과정은 통상적인 AES-CTR과 상이하다. 이를 위해 최초의 블록에 대한 암호화 과정에서 산출된 연산 결과를 저장하여 두었다가, 이후 재활용하는 과정이 수반된다.

[0028] 330 단계에서는 310 단계를 통해 상기 분할된 4 개의 스테이트 중 상기 각 라운드의 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시(cache)에 저장한다. 이러한 구성은 대상 데이터 블록의 4 개의 스테이트 중 3 개의 스테이트가 AES-CTR 모드의 연산 과정에서 값이 변경되지 않으며, 나머지 1 개의 스테이트 중 일부 데이터의 값이 변경되는 것을 발견한데 따라 안출된 것이다. 즉, 캐시에 저장된 3 개의 스테이트는 이후 다음 대상 데이터 블록에 대한 연산 과정에서 재활용된다.

[0029] 비록 각 변환(연산)에 따라 실질적으로 그 값이 변경되는 바이트는 1 개의 스테이트의 바이트(4 바이트)보다 적어질 수 있으므로, 3 개의 스테이트 이상의 정보를 캐시에 저장하여 재활용하는 것이 가능하나, 이 경우 캐시의 지속적인 활용 가능성이 낮아지는 문제점이 발생하므로 본 발명의 실시예들은 캐시의 활용도를 높이기 위해 3 개의 스테이트만을 캐시에 저장하는 방법을 채택하였다. 캐시에 저장되는 데이터는 각 변환에 따라 달라질 수 있으며, 이후 도 4 내지 도 7을 통해 구체적으로 설명한다. 이상의 과정은 적어도 하나의 프로세서와 하나의 캐시를 통해 구현 가능하다.

[0030] 340 단계에서는 제 1 블록에 연속하는 N 개(N은 양의 정수)의 블록들에 대해 각각의 블록들과 상기 캐시를 참조하여 획득한 연산 결과를 XOR함으로써 N 개의 암호문을 생성한다. 이제, 최초의 블록(제 1 블록을 의미한다.)에 연속하는 복수 개의 블록들에 대해서도 일련의 변환을 수행하는데, 이러한 변환 과정에 앞서 저장한 캐시 데이터를 활용한다. 즉, 제 1 블록과 동일한 변환을 나머지 N 개의 블록들에 대해 동일하게 수행하는 것이 아니라, 이미 산출되어 캐시에 저장된 3 개의 스테이트를 활용하여 변환에 필요한 연산의 수를 줄일 수 있다. 이상의 과정은 적어도 하나의 프로세서와 하나의 캐시를 통해 구현 가능하다.

[0031] 마지막으로 350 단계에서는 320 단계를 통해 생성된 제 1 암호문과 340 단계를 통해 생성된 N 개의 암호문을 암호화 결과로 출력한다. 이렇게 생성된 암호화 결과는 실시간으로 수신자에게 전달될 수 있을 것이다.

[0032] 한편, 이상의 암호화 과정에 대응하여 암호문을 평문으로 복원하는 복호화 과정이 구현 가능하다. 그러나, 중복되는 설명을 피하기 위해 여기서는 암호화 과정만을 소개하도록 한다. 복호화 과정은 이상의 AES-CTR 암호화 과정의 역에 해당하는 것으로 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 이상의 실시예가 채택하고 있는 핵심적인 아이디어(캐시를 이용하여 각 라운드의 변환 연산의 연산 결과값을 재활용하는 구성)를 이미 알려진 AES-CTR 복호화 과정에 적용할 수 있다.

[0033] 이하에서는 도 3에서 제시한 캐시를 활용한 AES-CTR 암호화 방법에 대한 본 발명의 다양한 실시예들을 각각의 라운드마다 차별적으로 나타나는 특징을 중심으로 제시하도록 한다.

[0034] 도 4는 본 발명의 일 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 캐시를 이용하여 AES 초기화 라운드의 연산 과정을 단축하는 과정을 설명하기 위한 도면으로, 특히 AES 초기화 라운드에서 AddRoundKey() 연산의 결과를 캐시에 저장하고 재사용하는 방법에 대해 제시하고 있다.

[0035] 보다 구체적으로, AES-CTR 모드 알고리즘은 초기 라운드에서 암호 알고리즘의 입력 값과 라운드 키(Round Key, Cipher key)를 XOR한다. 첫 블록의 입력 값을 초기화 벡터(IV, Initialization Vector)로 설정하고, 미리 정해

진 규칙에 따라 초기화 벡터의 값을 증가시켜 다음 블록의 입력 값으로 사용한다. 초기화 벡터의 증가에 따라 캐리(Carry)가 발생하지 않는다면, 첫 블록의 입력 값인 초기화 벡터와 바로 다음 블록의 입력 값인 카운터 값은 증가한 값에 영향을 받는 마지막 1 바이트만 값만이 다르며 나머지 값들은 동일하다.

[0036] 예를 들어, 값을 1씩 증가시키는 카운터를 사용하며, IV_0(최초 블록의 입력값을 의미한다.)값을 1로 설정했다고 가정하면, IV_0값과 카운터에 의해 증가된 값 IV_1(다음 블록의 입력값을 의미한다.)은 다음과 같이 마지막 1 바이트만 다른 값을 갖는다.

[0037] - IV_0 : 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01

[0038] - IV_1 : 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

[0039] 이러한 차이는 도 4의 첫 번째 블록의 스테이트(411)와 이에 연속하는 두 번째 블록의 스테이트(412)의 연산 결과를 비교하면 명확히 알 수 있다. 결국 입력 값과 첫 번째 라운드 키가 XOR되는 초기 라운드에서 각각의 데이터 블록들은 같은 라운드 키 값과 XOR되기 때문에 그 결과도 IV0, IV1과 마찬가지로 마지막 1 바이트만 다른 값을 갖는다.

[0040] 이러한 특성에 기초하여 본 실시예는 초기 라운드의 결과 16 바이트 중 마지막 4 바이트를 제외한 나머지 12 바이트를 저장함으로써 2^{32}-1 블록을 연산하는 동안 재사용할 수 있는 방법을 제안한다. 즉, 초기화 벡터 IV를 { State[0], State[1], State[2], State[3] }이라 할 때, 초기 라운드의 결과로 저장된 스테이트에서 값이 변하지 않는 부분인 State[0], State[1], State[2] 부분을 캐시(420)에 저장하여 다음 블록의 연산 시에 재사용한다. 도 4에는 초기 라운드 과정과 결과 스테이트의 캐시 가능한 부분(즉, 변환 연산에 의해 값이 변경되지 않는 부분을 의미한다.)을 빗금을 쳐서 표시하고 있다.

[0041] 본 실시예에 따르면, 카운터 값의 증가에 따라 카운터 값(대상 데이터 블록)의 12번째 바이트 값이 변경되면서 캐리를 발생시켜 11번째 바이트에 영향을 미칠 때까지 캐시된 정보를 사용할 수 있다. 예를 들어, 카운터 증가 값이 1 이고, 초기화 벡터의 값이 0 인 경우 카운터 값의 11번째 바이트에서 캐리가 발생하는 블록은 4,294,967,297번째 블록이 된다. 따라서 이 경우 4,294,967,295개의 블록을 계산하는 동안은 초기 라운드에서 스테이트의 일부분에 대해 입력 값과 라운드 키의 XOR 연산을 수행하는 대신 저장된 정보를 사용할 수 있다. AES 알고리즘의 한 블록은 16 바이트이므로, 위와 같은 경우 약 65.5 GBytes 마다 한 번씩만 캐시된 내부 데이터를 갱신해주면 된다.

[0042] 요약하건대, AES-CTR 모드에서 라운드가 AES 초기화 라운드(initial round)인 경우, 값이 변경되지 않는 3 개의 스테이트를 캐시에 저장하는 단계는, 초기화 라운드의 연산 결과인 16 바이트 중, 마지막 4 바이트를 제외한 12 바이트를 캐시에 저장하는 것이 바람직하다.

[0043] 또한, 최초의 블록(제 1 블록을 의미한다.)에 연속하는 N 개의 블록에 대해 N 개의 암호문을 생성하는 단계는, 제 1 블록에 연속하는 2^{32}-1 개의 블록들에 대해 상기 캐시에 저장된 12 바이트를 독출하고, 2^{32}-1 개의 블록들에 대해 각각의 카운터 값의 변화 규칙을 고려하여 XOR 연산 결과의 4 바이트를 산출하고, 독출된 12 바이트와 산출된 4 바이트를 결합하여 해당 데이터 블록에 대한 연산 결과를 생성하며, 생성된 연산 결과에 기초하여 2^{32}-1 개의 암호문을 생성한다. 여기서, 2^{32}-1 개는 캐시에 저장된 3 개의 스테이트를 재활용할 수 있는 한계값을 의미하며, 2^{32}-1 개 이후의 암호화 대상 데이터 블록에 대해서는 캐리로 인해 캐시된 값을 갱신해 주어야 한다.

[0044] 도 5는 본 발명의 다른 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 캐시를 이용하여 AES 라운드 1의 연산 과정을 단축하는 과정을 설명하기 위한 도면으로서, AES 라운드 1에서 SubBytes(), ShiftRows(), MixColumns(), AddRoundKey()의 연산 결과 중 12 바이트 정보를 캐시하여 재사용하는 방법을 도시하고 있다.

[0045] 보다 구체적으로, 초기화 라운드를 거친 이후에 처리되는 데이터는 첫 번째 라운드(라운드 1을 의미한다.)에 입력으로 활용된다. 이 때, 첫 번째 라운드에서도 역시 데이터를 저장하여 다음 블록에 활용할 수 있다. 다만 초기화 라운드보다 캐시된 데이터의 갱신이 자주 일어난다. 도 5에 도시된 바와 같이 라운드 1의 입력이 되는 초기 라운드의 스테이트는 첫 번째 블록과 두 번째 블록이 마지막 1 바이트만 서로 다른 값을 갖는다. 비록 라운드 1 과정을 통해 스테이트의 서로 다른 값을 갖는 마지막 1 바이트는 ShiftRows()와 MixColumns()의 단계에서 이동되고 확산되지만, 그 영향은 하나의 스테이트에 국한된다. 따라서 도 5와 같이 라운드 1의 결과로 출력된 스테이트들(511, 512)을 비교하여 보면, State[0]을 제외한 나머지 스테이트들을 캐시(520)에 저장하여 사용할

수 있다.

[0046] 이 때, 카운터 값의 증가에 따라 카운터 값의 15번째 바이트에서 캐리가 발생하여 초기 라운드 스테이트의 14번째 바이트 값이 바뀌면, 캐시 해두었던 스테이트 값들을 갱신해 주어야 한다. 한편, 캐리로 인해 초기 라운드의 결과 스테이트에서 14 번째 바이트 값이 함께 바뀐 경우에도 라운드 1 과정 후의 스테이트 결과 값 중 State[2]와 State[3]은 여전히 같은 값이 유지된다. 이러한 경우를 고려하여 14번째 바이트 값이 바뀐 경우 State[2], State[3]을 연산하지 않고 캐시된 데이터를 불러오도록 구현할 수도 있다. 하지만, 이 경우 연산의 횟수를 줄여서 얻을 수 있는 이득보다 캐리 검사(Carry Check)로 인한 오버헤드가 더 크기 때문에 전체 알고리즘의 수행 속도는 앞의 방법보다 느리다.

[0047] 결국 효율성을 고려하였을 때 초기화 라운드에서 캐시된 스테이트 정보가 사용될 수 있는 최대 블록 수는 $2^{32}-1$ 이며, 라운드 1에서 캐시된 스테이트 정보가 사용될 수 있는 최대 블록 수는 $2^{8}-1$ 이다.

[0048] 요약하건대, AES-CTR 모드의 라운드가 AES 라운드 1(round 1)인 경우, 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시에 저장하는 단계는, 라운드 1의 연산 결과인 16 바이트 중, 처음 4 바이트를 제외한 12 바이트를 캐시에 저장하는 것이 바람직하다.

[0049] 또한, 최초의 블록(제 1 블록을 의미한다.)에 연속하는 N 개의 블록에 대해 N 개의 암호문을 생성하는 단계는, 제 1 블록에 연속하는 $2^{8}-1$ 개의 블록들에 대해 캐시에 저장된 12 바이트를 독출하고, 라운드 1의 연산 과정을 고려하여 연산 결과의 4 바이트를 산출하고, 독출된 12 바이트와 산출된 4 바이트를 결합하여 해당 데이터 블록에 대한 연산 결과를 생성하며, 생성된 연산 결과에 기초하여 $2^{8}-1$ 개의 암호문을 생성한다. 여기서, $2^{8}-1$ 개는 캐시에 저장된 3 개의 스테이트를 재활용할 수 있는 한계값을 의미한다.

[0050] 나아가, 본 발명의 또 다른 실시예는 도 5를 통해 제시된 본 발명의 실시예에서 전체 16 바이트의 카운터 값에서 캐시되지 않은 4 바이트 정보를 미리 획득하는 방법을 제공한다. 이하에서 도 6을 참조하여 설명한다.

[0051] 도 6은 본 발명의 또 다른 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 치환 테이블을 이용하여 AES 라운드 1의 연산 과정을 단축하는 과정을 설명하기 위한 도면이다. 앞서 도 5를 통해 소개된 실시예는 AES 라운드 1에서 12 바이트의 정보를 미리 저장된 캐시로부터 독출하여 사용하고, 나머지 4 바이트 정보는 매 블록마다 새로 계산한다. 도 6의 실시예에서는 암호 인스턴스 초기화 단계에서 매번 계산되는 4 바이트 정보에 대해 미리 산출된 테이블(Pre-Computation Table)을 생성하여 효율성을 극대화하는 기법을 제안한다.

[0052] 일반적으로 암호 모듈이 구현될 경우 두 단계로 나뉘어 구성된다. 첫 번째 단계는 초기화 단계이다. 암호 초기화 단계에서는 사용하려는 암호 알고리즘의 인스턴스가 사용하게 될 암호화 키와 초기화 벡터가 결정된다. 이 단계에서 키는 정해진 AES 키 길이에 따라 라운드 키로 확장되며, CTR 모드에서는 초기화 벡터가 해당 암호 인스턴스의 카운터로 사용된다. 두 번째 단계는 암/복호화 단계로서 정해진 라운드 키와 초기화 벡터를 사용하여 입력되는 입력 데이터를 암/복호화 하는 과정이 진행된다. 특히, 이 단계는 제공되는 암호 함수의 성격에 따라 입력을 불연속적으로 처리할 수 있도록 여러 과정을 거쳐 나뉘어서 진행되기도 하므로, 같은 키와 초기화 벡터라도 여러 번 반복적으로 호출될 수도 있다.

[0053] 도 6의 실시예는 초기 1 회만 호출되어 인스턴스를 유지하는 초기화 단계에서 AES-CTR 모드의 라운드 1에서 빈번하게 사용되는 정보를 미리 계산하여, 최대 4 바이트 정보가 변경되기 전까지 재사용 가능한 기법을 도시하고 있다. 도 6은 AES-CTR 모드가 동작하는 동안 라운드 1에서의 데이터들 위치 변화와 상관관계가 미치는 영향을 나타낸다. AES 알고리즘의 입력은 다음과 같이 표현될 수 있다.

수학식 1

[0054] $$Input=\{State[0],State[1],State[2],State[3]\}=\{b_0,b_1,\ldots,b_{15}\}$$

[0055] 암호의 입력 16 바이트의 데이터가 0부터 15까지 순서를 가진다고 했을 때, AES-CTR 모드는 15 번째 바이트부터 1씩 증가하여 입력을 매 블록마다 달리한다. 이 때, 두 번째 블록에서 b_{15}는 첫 번째 블록과 다르기 때문에 MixColumns() 연산에서 b_{15}는 b_0, b_5, b_{10}에 영향을 미치게 된다. b_0, b_5, b_{10}가 이전 블록과 비교했을 때 값의 변

경이 없다는 점을 상기해보면, b_{15}의 변화가 캐리를 수반하지 않는다고 가정하면, b_{15}가 라운드 1의 MixColumns() 연산에서 b_0, b_5, b_{10}를 결정한다. 이는 도 6에서 (a) MixColumns() 연산과 (b) AddRoundKey() 연산이 매 라운드마다 반복되는 것을 의미하고, 따라서 미리 산출해 놓은 (a)와 (b)의 결과 값을 재사용하는 방법을 적용하면 연산 효율성을 향상시킬 수 있다. 도 6에는 이들 (a)와 (b)의 연산에 의해 변경되는 값과 변경되지 않는 값을 카운터 값(610, 620)을 통해 표시하고 있다.

[0056] 주어진 b_{15}에 대한 State[0]를 결정하는 테이블을 구성한다고 할 때, b_{15}는 0x00부터 0xFF까지 변화하고, State[0]는 4 바이트이기 때문에 총 4 × 256 = 1 KBytes의 메모리가 필요하다. 이 테이블은 또한 모든 암호 인스턴스마다 같지는 않지만, 오직 입력된 키와 초기화 벡터에 의존하기 때문에 초기화 단계에서 미리 계산하고 결정할 수 있다. 초기화 단계는 매 암호 인스턴스에서 단 한 번만 수행되기 때문에 실제 암/복호화 연산 단계에서 효율성을 확보할 수 있다.

[0057] 생성된 치환 테이블을 사용할 수 있는 것은 b_{15} 증가의 영향이 b_{10}에 영향을 미칠 때, 즉 [b_{11} - b_{15}]가 [0xFFFFFFFFFF]가 되어 b_{10}에 캐리가 발생할 경우이다. 이 때, 미리 계산된 치환 테이블에 업데이트 과정이 필요한데, 한번 업데이트가 이루어진 이후에는 2^{40} × 16 바이트의 입력은 추가적인 테이블 업데이트 과정 없이 치환 테이블을 사용할 수 있다. 즉, 16 TeraBytes의 데이터를 처리하는 동안에는 추가 연산 과정이 필요하지 않기 때문에 연산의 효율성을 높일 수 있다.

[0058] 요약하건대, AES-CTR 모드에 따른 암호화 방법에서 캐시에 저장되지 않는 처음 4 바이트에 대하여 AES 라운드 1에 선행하는 AES 초기화 라운드에서 특정 연산에 대한 결과를 미리 산출하여 치환 테이블에 저장하는 단계를 더 포함하고, AES 라운드 1에서는 저장된 4 바이트의 치환 테이블을 독출하여 카운터 값을 산출하는 것이 바람직하다.

[0059] 이 때, 이상의 특정 연산은 대상 블록의 컬럼(column)이 다항식에 의해 치환되는 MixColumns() 연산 및 MixColumns() 연산의 출력과 라운드 키(round key)의 XOR 값을 산출하는 AddRoundKey() 연산을 의미한다. 각각의 연산 방법에 관한 구체적인 설명은 본 실시예의 본질을 흐리게 하므로 여기서는 생략한다.

[0060] 또한, 카운터 값에서 캐리(carry)가 발생하는 경우 새롭게 산출된 연산 결과에 따라 저장된 치환 테이블을 갱신하고, 갱신된 치환 테이블은 갱신 이후, 연속하는 2^{40}×16 바이트의 입력 데이터에 대해 상기 치환 테이블로부터 독출된 4 바이트의 값을 이용하여 상기된 특정 연산 없이 카운터 값을 산출하는 것이 가능하다.

[0061] 도 7은 본 발명의 또 다른 실시예에 따른 AES의 CTR 모드에 따른 암호화 방법에서 캐시를 이용하여 AES 라운드 2의 연산 과정을 단축하는 과정을 설명하기 위한 도면으로, AES 라운드 2에서 MixColumns()연산 과정 중 매 블록마다 12 바이트 정보가 반복되는 점을 이용하여 해당 데이터를 캐시하고 재사용하는 방법을 도시하고 있다.

[0062] 첫 번째 라운드를 거친 스테이트는 두 번째 라운드의 입력으로 사용된다. 도 7은 첫 번째 블록과 두 번째 블록의 라운드 2 과정을 나타낸 것으로, 특히 MixColumns()에 State[0]의 변환 과정이 나타나 있다. 도 7과 같이 첫 번째 블록과 두 번째 블록의 Round 1의 결과로 출력된 스테이트는 State[0]만 다른 값을 갖는다. 스테이트의 각 바이트를 S[i] (i = 0, 1, 2, ... , 15) 라 표현하면, 라운드 2 과정에서 ShiftRows() 변환을 거친 후의 State[0]는 S[0], S[5], S[10], S[15] 바이트들로 구성된다. 이 State[0] 는 MixColumns()에서 아래와 같은 변환이 일어난다.

수학식 2

[0063]
$$\begin{pmatrix} 2\,3\,1\,1 \\ 1\,2\,3\,1 \\ 1\,1\,2\,3 \\ 3\,1\,1\,2 \end{pmatrix} \cdot \begin{pmatrix} S[0] \\ S[5] \\ S[10] \\ S[15] \end{pmatrix} = \begin{pmatrix} 2S[0] \oplus 3S[5] \oplus 1S[10] \oplus 1S[15] \\ 1S[0] \oplus 2S[5] \oplus 3S[10] \oplus 1S[15] \\ 1S[0] \oplus 1S[5] \oplus 2S[10] \oplus 3S[15] \\ 3S[0] \oplus 1S[5] \oplus 1S[10] \oplus 2S[15] \end{pmatrix} = \begin{pmatrix} rlt[0] \\ rlt[1] \\ rlt[2] \\ rlt[3] \end{pmatrix}$$

[0064] 첫 번째 블록과 두 번째 블록에서 이 변환을 살펴보면, 두 블록의 각 State[0]에서 서로 다른 바이트는 S[0] 하나뿐이다. 따라서 위 연산에서 S[0]를 제외한 나머지 연산, 즉 rlt[0]의 3?S[5] $\oplus$ 1?S[10] $\oplus$ 1?S[15] 과

rlt[1]의 2?S[5] ⊕ 3?S[10] ⊕ 1?S[15], rlt[2]의 1?S[5] ⊕ 2?S[10] ⊕ 3?S[15], 그리고 rlt[3]의 1?S[5] ⊕ 1?S[10] ⊕ 2?S[15] 부분의 연산 결과(710)를 캐시(720)에 저장하여 활용할 수 있다. 이 부분의 연산 결과를 캐시하여 사용함으로써 3번의 XOR연산을 1번으로 줄일 수 있다. State[1], State[2], State[3] 에서도 역시 마찬가지로 각각 S[3], S[2], S[1]을 제외한 나머지 부분의 연산 결과를 캐시하여 사용할 수 있다.

[0065] 앞서 설명한 도 5의 실시예와 마찬가지로 라운드 2의 입력 스테이트의 State[1], State[2], State[3]의 값이 유지되는 동안에만 캐시된 정보가 사용될 수 있기 때문에 캐시된 정보가 사용될 수 있는 최대 블록의 수는 도 5의 라운드 1의 경우와 마찬가지로 2^8-1 이다.

[0066] 요약하건대, AES-CTR 모드의 라운드가 AES 라운드 2(round 2)인 경우, 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시에 저장하는 단계는, 라운드 2의 연산 중, 대상 블록의 컬럼이 특정 다항식에 의해 치환되는 MixColumns() 연산 과정에서 인접한 블록 간에 값이 변경되지 않는 12 바이트의 XOR 결과를 캐시에 저장하는 것이 바람직하다.

[0067] 또한, 최초의 블록(제 1 블록을 의미한다.)에 연속하는 N 개의 블록에 대해 N 개의 암호문을 생성하는 단계는, 제 1 블록에 연속하는 2^8-1 개의 블록들에 대해 캐시에 저장된 12 바이트를 독출하고, 라운드 1의 연산 과정을 고려하여 MixColumns() 연산 과정에서 인접한 블록 간에 값이 변경되는 4 바이트의 XOR 결과를 산출하고, 독출된 12 바이트와 산출된 4 바이트를 결합하여 MixColumns() 연산의 결과값으로 설정하고, 설정된 결과값에 기초하여 해당 데이터 블록에 대한 연산 결과를 생성하며, 생성된 연산 결과에 기초하여 2^8-1 개의 암호문을 생성한다.

[0068] 상기된 본 발명의 다양한 실시예들에 따르면, 각 라운드의 연산 과정에서 값이 변경되지 않는 스테이트들을 캐시에 저장하여 참조함으로써 인터넷 스트리밍 데이터를 암호화/복호화하는 과정에서 AES 알고리즘의 일련의 연산을 단축하여 실시간 처리 속도를 향상시키고, 제한된 자원을 갖는 환경에서 이들 연산을 처리하기 위해 소모되는 컴퓨팅 자원을 절약함으로써 암호화 효율을 높일 수 있다.

[0069] AES-CTR 모드는 병렬처리가 가능하다는 특성으로 인해 고속 처리를 필요로 하는 다양한 스트리밍 서비스 등의 암/복호화 연산에 사용되고 있다. IPTV, VoIP등의 서비스를 사용하는 모바일 환경은 자원이 제한되어 있기 때문에 이러한 환경에서 사용되는 암호화 알고리즘은 효율성을 고려하여 구현되어야 한다. 플랫폼에 특화된 알고리즘 성능 개선은 확장성이 많이 부족하지만, AES 알고리즘 자체의 구현 로직 개선은 알고리즘이 동작하는 아키텍처에 제한되지 않고 적용 가능하다.

[0070] 나아가, 본 발명은 컴퓨터로 읽을 수 있는 기록 매체에 컴퓨터가 읽을 수 있는 코드로 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다.

[0071] 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현하는 것을 포함한다. 또한, 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다. 그리고 본 발명을 구현하기 위한 기능적인(functional) 프로그램, 코드 및 코드 세그먼트들은 본 발명이 속하는 기술 분야의 프로그래머들에 의하여 용이하게 추론될 수 있다.

[0072] 이상에서 본 발명에 대하여 그 다양한 실시예들을 중심으로 살펴보았다. 본 발명에 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

부호의 설명

[0073] 110 : 대상 데이터 블록 120 : 4×4 스테이트 행렬

210, 220, 230 : 데이터 블록 211 : 초기화 벡터(카운터)

212 : AES 암호화 방법에 따른 암호화 값

213 : 128 비트 평문 214 : 128 비트 암호문

411, 412 : 초기화 라운드에서의 블록 스테이트

511, 512 : 라운드 1에서의 연산 결과에 따른 블록 스테이트

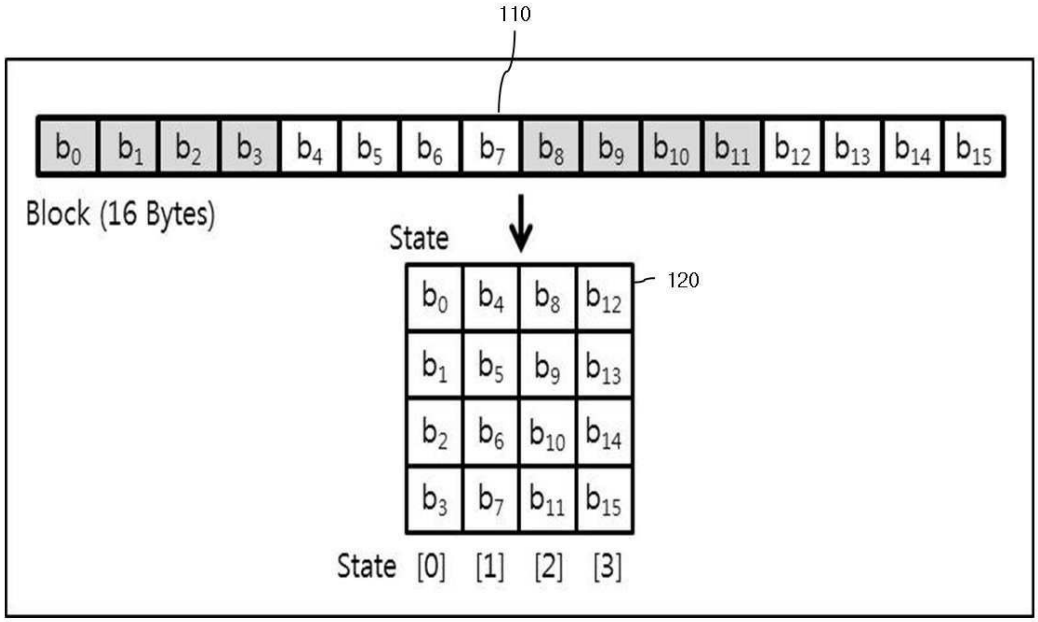
610, 620 : 라운드 1에서의 각 연산 결과에 따른 블록 스테이트 일부

710 : 라운드 2에서의 연산 결과에 따른 블록 스테이트 일부

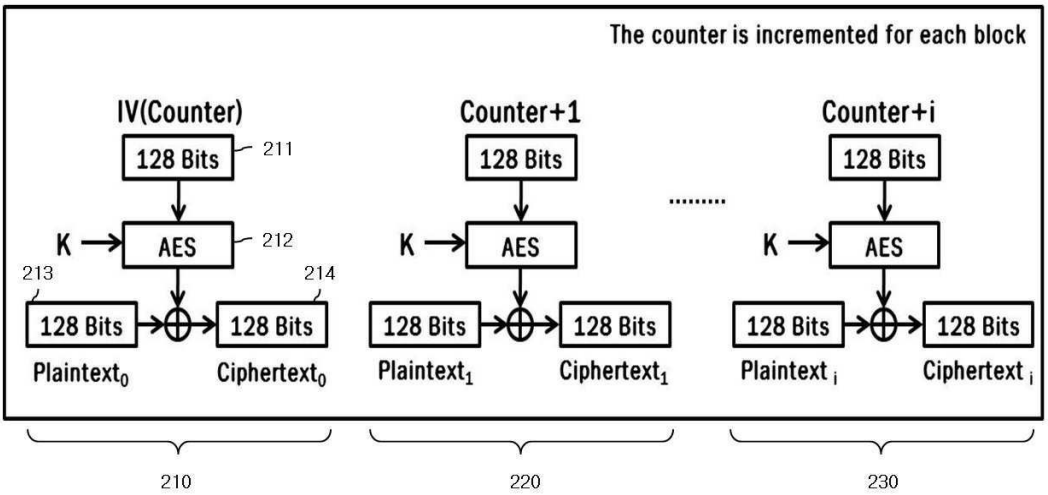
420, 520, 720 : 캐시

도면

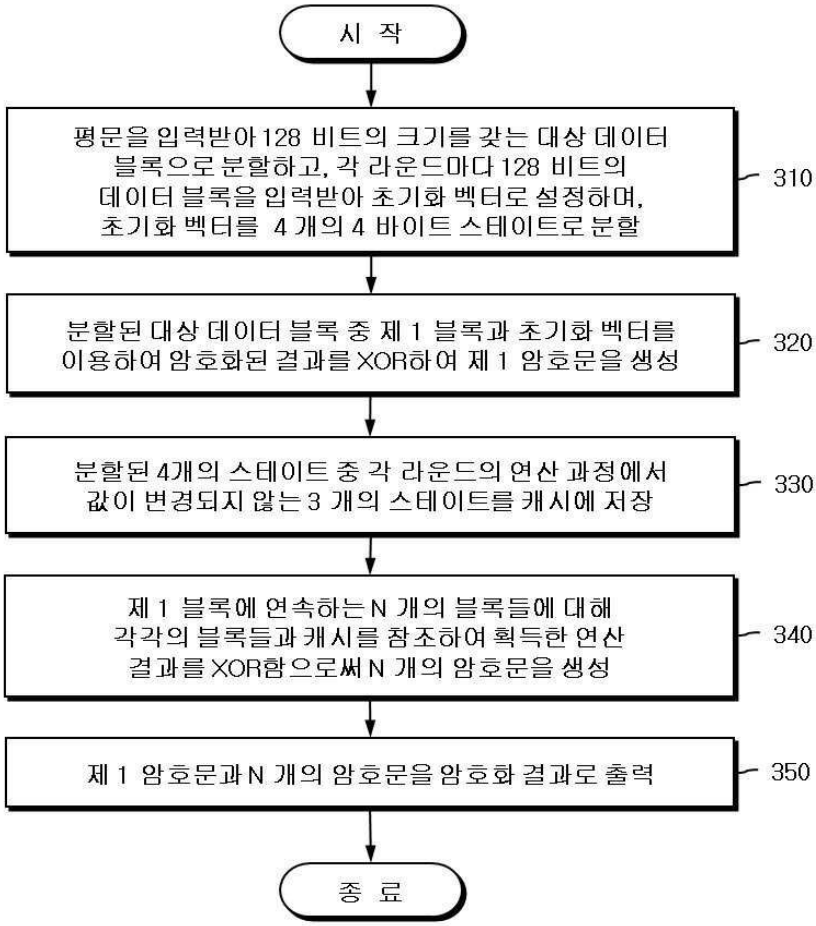
도면1



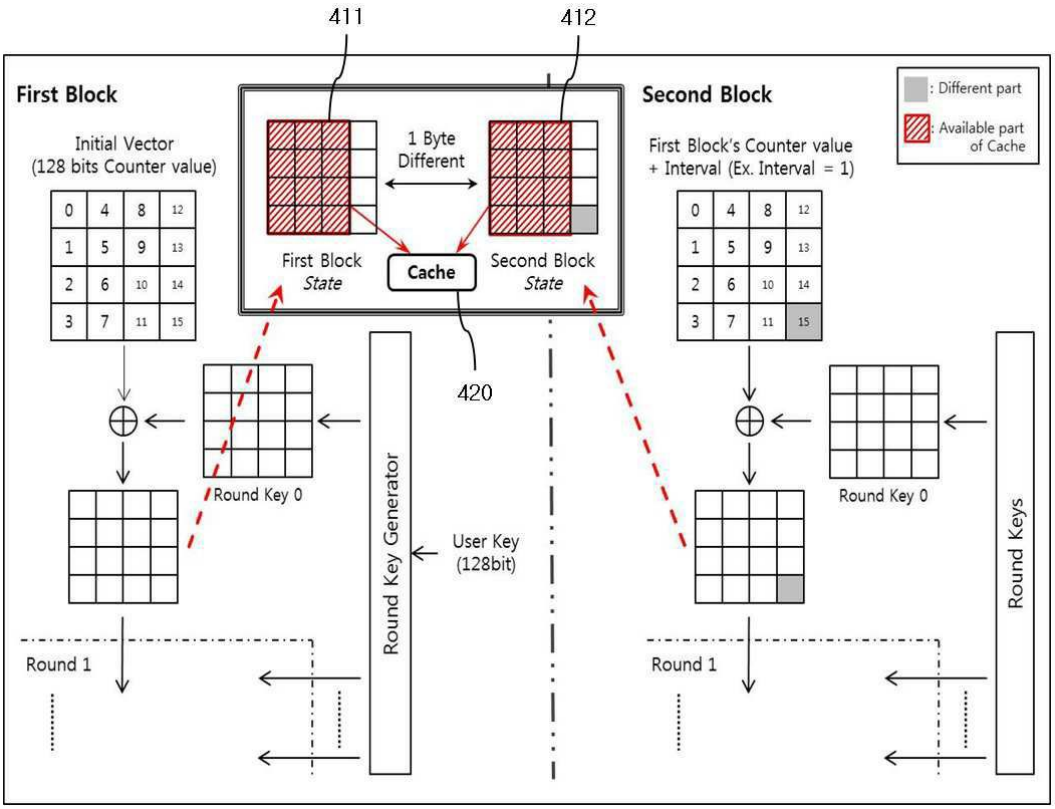
도면2



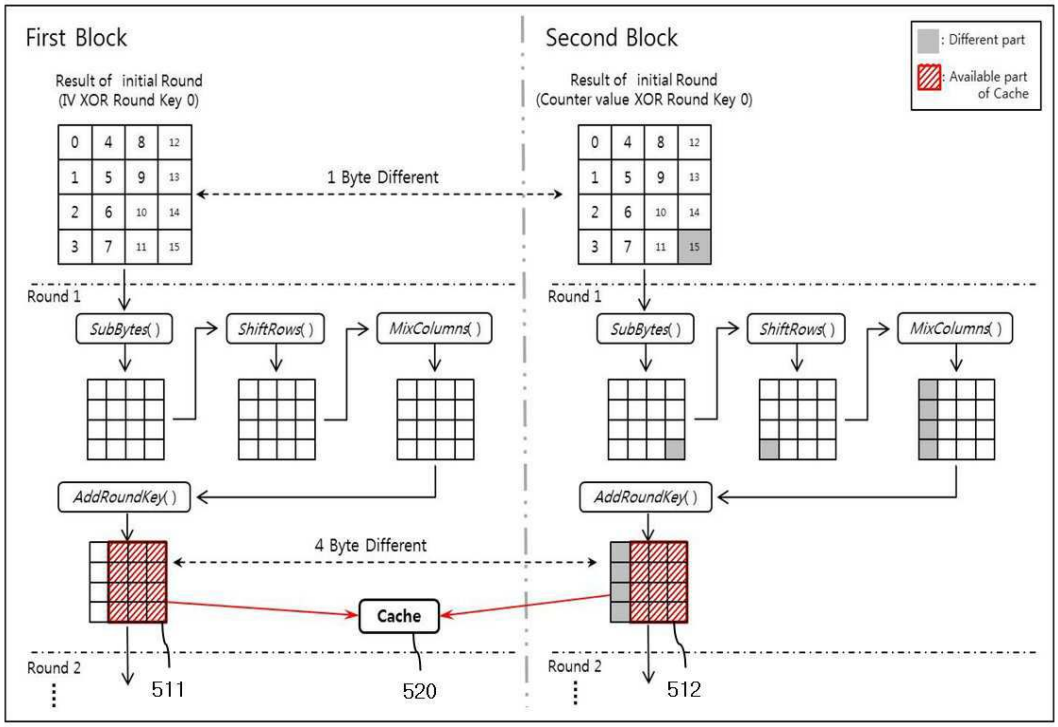
도면3


시 작
평문을 입력받아 128 비트의 크기를 갖는 대상 데이터 블록으로 분할하고, 각 라운드마다 128 비트의 데이터 블록을 입력받아 초기화 벡터로 설정하며, 초기화 벡터를 4 개의 4 바이트 스테이트로 분할
310
분할된 대상 데이터 블록 중 제 1 블록과 초기화 벡터를 이용하여 암호화된 결과를 XOR하여 제 1 암호문을 생성
320
분할된 4개의 스테이트 중 각 라운드의 연산 과정에서 값이 변경되지 않는 3 개의 스테이트를 캐시에 저장
330
제 1 블록에 연속하는 N 개의 블록들에 대해 각각의 블록들과 캐시를 참조하여 획득한 연산 결과를 XOR함으로써 N 개의 암호문을 생성
340
제 1 암호문과 N 개의 암호문을 암호화 결과로 출력
350
종 료

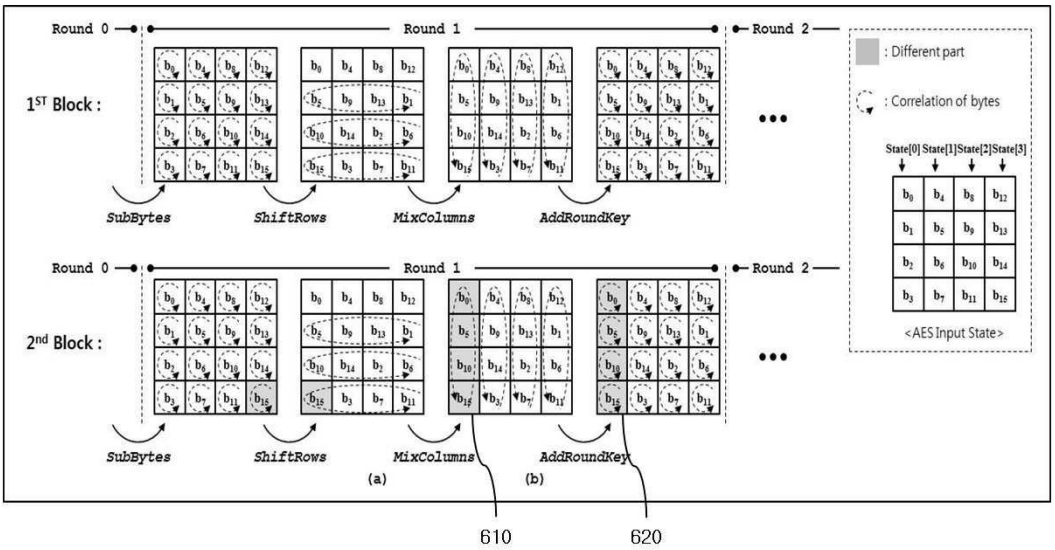
도면4


411
412
First Block
Second Block
: Different part
: Available part of Cache
Initial Vector
(128 bits Counter value)
1 Byte Different
First Block's Counter value
+ Interval (Ex. Interval = 1)
First Block State
Cache
Second Block State
420
Round Key 0
Round Key Generator
User Key
(128bit)
Round Keys
Round 1

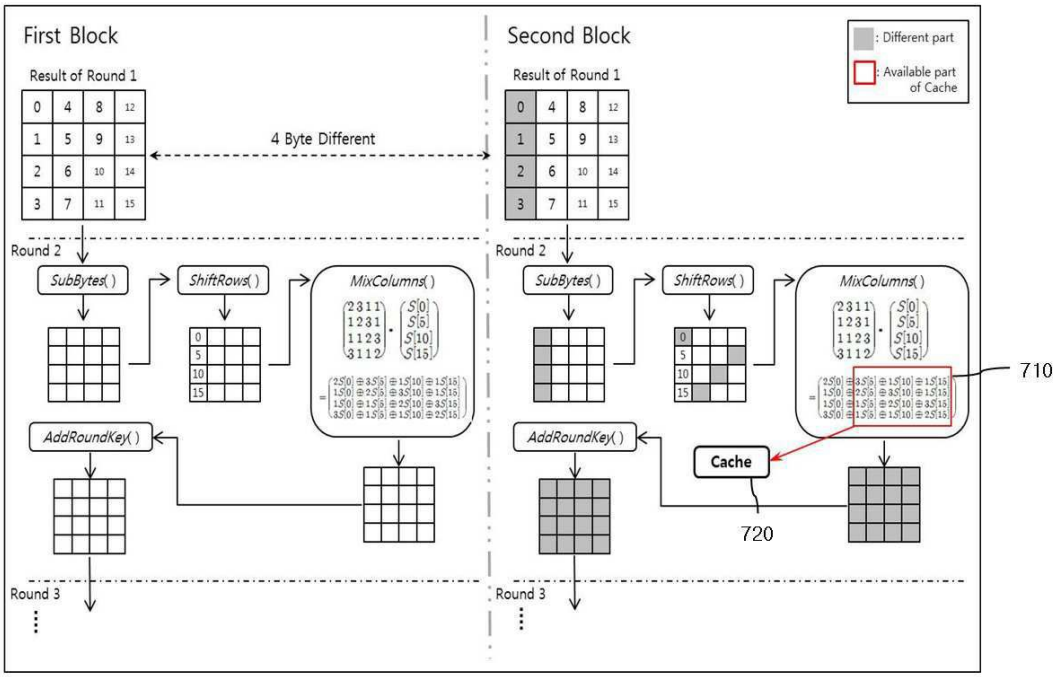
도면5


First Block
Second Block
: Different part
: Available part of Cache
Result of initial Round
(IV XOR Round Key 0)
Result of initial Round
(Counter value XOR Round Key 0)
1 Byte Different
Round 1
SubBytes( )
ShiftRows( )
MixColumns( )
AddRoundKey( )
4 Byte Different
Cache
Round 2
511
520
512

도면6


Round 0
Round 1
Round 2
1ST Block :
2nd Block :
SubBytes
ShiftRows
MixColumns
AddRoundKey
: Different part
: Correlation of bytes
State[0] State[1] State[2] State[3]
<AES Input State>
(a)
(b)
610
620

도면7


First Block
Second Block
Result of Round 1
4 Byte Different
: Different part
: Available part of Cache
Round 2
SubBytes( )
ShiftRows( )
MixColumns( )
AddRoundKey( )
Cache
710
720
Round 3