(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년07월20일
(11) 등록번호 10-1880075
(24) 등록일자 2018년07월13일

(51) 국제특허분류(Int. Cl.)
G06F 21/78 (2013.01) *G06F 17/30* (2006.01)
G06F 21/62 (2013.01) *G06F 3/06* (2006.01)
(52) CPC특허분류
G06F 21/78 (2013.01)
G06F 17/30 (2013.01)
(21) 출원번호 10-2016-7015977
(22) 출원일자(국제) 2015년01월14일
심사청구일자 2016년06월15일
(85) 번역문제출일자 2016년06월15일
(65) 공개번호 10-2016-0085884
(43) 공개일자 2016년07월18일
(86) 국제출원번호 PCT/US2015/011341
(87) 국제공개번호 WO 2015/108931
국제공개일자 2015년07월23일
(30) 우선권주장
61/927,914 2014년01월15일 미국(US)
14/596,113 2015년01월13일 미국(US)
(56) 선행기술조사문헌
US20100313040 A1*
US20090268903 A1*
US08199911 B1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인텔 코포레이션
미합중국 캘리포니아 95054 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
체리톤, 데이비드, 알.
미국 94025 캘리포니아주 멘로 파크 스위트 110 미들필드 로드 200
(74) 대리인
양영준, 김연송, 백만기

전체 청구항 수 : 총 24 항 심사관 : 구대성

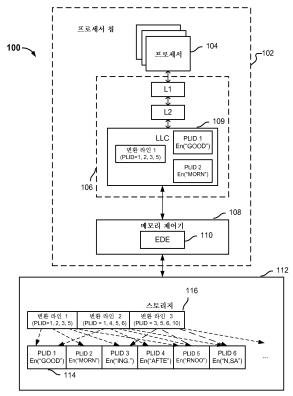
(54) 발명의 명칭 **중복 제거 기반 데이터 보안**

(57) 요 약

데이터 보안의 제공은, 스토리지에 데이터 컨텐츠를 기입하라는 요구에 응답하여, 데이터 컨텐츠에 기초하여 암호화된 데이터 컨텐츠를 생성하고; 스토리지 내의 암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도하고; 암호화된 데이터 컨텐츠에 대한 참조가 획득되는 경우에, 스토리지 내의 암호화된 데이터 컨텐츠에 대한 참조를

(뒷면에 계속)

대 표 도 - 도1



참고하기 위해 변환 라인을 수정하고; 암호화된 데이터 컨텐츠에 대한 참조가 획득되지 않은 경우에, 암호화된 데이터 컨텐츠를 새로운 로케이션에 저장하고, 새로운 로케이션에 저장된 암호화된 데이터 컨텐츠에 대한 참조를 획득하고, 새로운 로케이션에 저장된 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 변환 라인을 수정하는 것을 포함한다.

(52) CPC특허분류
G06F 21/6218 (2013.01)
G06F 3/0608 (2013.01)
G06F 2211/007 (2013.01)

명 세 서

청구범위

청구항 1

보안 시스템으로서,

스토리지; 및

상기 스토리지에 결합된 메모리 제어기

를 포함하고,

상기 메모리 제어기는,

상기 스토리지에 데이터 컨텐츠를 기입하라는 요구에 응답하여, 상기 데이터 컨텐츠에 적어도 부분적으로 기초하여 암호화된 데이터 컨텐츠를 생성하고 - 상기 데이터 컨텐츠는 복수의 도메인 중 하나의 도메인과 관련되고, 상기 암호화된 데이터 컨텐츠 생성은 상기 복수의 도메인 중 하나의 도메인과 관련된 암호화 키에 따른 것임 -;

상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도하고;

상기 암호화된 데이터 컨텐츠에 대한 참조가 획득되는 경우에, 상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 변환 라인(translation line)을 수정하고;

상기 암호화된 데이터 컨텐츠에 대한 참조가 획득되지 않은 경우에: 새로운 로케이션에 상기 암호화된 데이터 컨텐츠를 저장하고; 상기 새로운 로케이션에 저장된 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하고; 상기 새로운 로케이션에 저장된 상기 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 상기 변환 라인을 수정하는, 시스템.

청구항 2

제1항에 있어서, 상기 스토리지는 메인 메모리, 보조 스토리지, 또는 양자 모두를 포함하는 시스템.

청구항 3

제1항에 있어서, 상기 메모리 제어기는 상기 암호화된 데이터 컨텐츠를 생성하기 위해 결정론적 암호화 기능(deterministic encryption function)을 수행하는 시스템.

청구항 4

제1항에 있어서, 상기 메모리 제어기는 고급 암호화 표준(Advanced Encryption Standard; AES), ECB-Mix-ECB(EME), XEX-TCB-CTS(XTS), 및 CBC-Mask-CBC(CMC) 중 하나 이상을 구현하는 시스템.

청구항 5

제1항에 있어서, 상기 메모리 제어기에 결합된 캐시를 더 포함하고, 상기 캐시는 암호화되지 않은 데이터를 저장하는 시스템.

청구항 6

제1항에 있어서, 상기 메모리 제어기에 결합된 캐시를 더 포함하고, 상기 캐시는 중복 제거된 데이터(deduplicated data)를 저장하는 시스템.

청구항 7

제1항에 있어서, 상기 메모리 제어기에 결합된 캐시를 더 포함하고, 상기 메모리 제어기는 또한 캐시 미스(cache miss) 시에 보안 패드를 생성하고 상기 보안 패드를 사용하여 상기 스토리지로부터 페치된 암호화된 데이터를 복호화하는 시스템.

청구항 8

제1항에 있어서, 상기 메모리 제어기에 결합된 캐시를 더 포함하고, 상기 메모리 제어기는 또한

판독 로케이션에 있는 데이터 컨텐츠에 액세스하라는 요구에 응답하여, 상기 판독 로케이션에 대응하는 변환 라인 및 데이터 라인이 상기 캐시에서 이용가능한 지 여부를 결정하고;

상기 데이터 라인이 상기 캐시에서 이용가능하지 않은 경우에:

상기 스토리지로부터 상기 데이터 라인을 로드하고;

상기 데이터 라인을 복호화하고;

상기 복호화된 결과를 상기 캐시에 저장하는 시스템.

청구항 9

제1항에 있어서, 상기 메모리 제어기에 결합된 캐시를 더 포함하고,

상기 데이터 컨텐츠를 기입하라는 요구는 상기 데이터 컨텐츠를 저장한 캐시 라인을 축출(evict)하라는 요구에 의해 발생되고, 상기 캐시 라인은 스토리지에 최종 기입된 이후에 수정된 시스템.

청구항 10

제1항에 있어서, 상기 데이터 컨텐츠를 기입하라는 요구를 생성하는 프로세서를 더 포함하고;

상기 프로세서의 어드레스 공간은 다수의 중복 제거 도메인들을 포함하고;

각각의 중복 제거 도메인은 상기 각각의 중복 제거 도메인 내의 데이터 컨텐츠를 암호화하는 데 사용되는 대응하는 키(key)를 갖는 시스템.

청구항 11

제1항에 있어서, 상기 변환 라인은 암호화되는 시스템.

청구항 12

제1항에 있어서,

상기 변환 라인은 암호화되고;

상기 데이터 컨텐츠 및 상기 변환 라인은 상이한 키들을 사용하여 암호화되는 시스템.

청구항 13

제1항에 있어서, 상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도하는 것은 컨텐츠 디렉토리 내의 상기 암호화된 데이터 컨텐츠를 룩업(look up)하는 것을 포함하는 시스템.

청구항 14

제1항에 있어서, 상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도하는 것은 컨텐츠 디렉토리 내의 상기 암호화된 데이터 컨텐츠를 룩업하는 것을 포함하고;

상기 컨텐츠 디렉토리는 암호화된 메타데이터를 포함하는 시스템.

청구항 15

제14항에 있어서, 상기 암호화된 메타데이터는 상기 컨텐츠 디렉토리를 공유하는 복수의 보호 도메인들에 걸쳐 공유되는 공통 키를 사용하여 암호화되는 시스템.

청구항 16

제1항에 있어서, 기입 동작을 수행하는 데 사용되는 표시된 시간량을 변화시키기 위해 기입 동작의 완료 표시가 수정되는 시스템.

청구항 17

제1항에 있어서, 데이터 라인에의 판독 액세스 시에, 상기 메모리 제어기는 또한 상기 데이터 라인의 데이터 컨텐츠가 상기 데이터 라인과 관련된 메타데이터와 일치하는지 여부를 결정하는 것을 포함하는 무결성 체크(integrity check)를 수행하는 시스템.

청구항 18

제17항에 있어서, 상기 데이터 라인의 데이터 컨텐츠가 상기 데이터 라인과 관련된 메타데이터와 일치하지 않는 경우에 잠재적 보안 침해(potential security violation)가 보고되는 시스템.

청구항 19

제17항에 있어서, 상기 데이터 라인과 관련된 메타데이터는 보안 키 해시 함수(secure keyed hash function)를 사용하여 생성되는 시스템.

청구항 20

제1항에 있어서, 상기 스토리지는 상이한 레이턴시들을 갖는 복수의 메모리들을 포함하는 하이브리드 메모리를 포함하는 시스템.

청구항 21

제1항에 있어서, 상기 스토리지는 상이한 레이턴시들을 갖는 복수의 메모리들을 포함하는 하이브리드 메모리, 및 더 높은 레이턴시 메모리를 위한 암호화의 단위와는 상이한 크기인 더 낮은 레이턴시 메모리를 위한 암호화의 단위를 포함하는 시스템.

청구항 22

제1항에 있어서, 상기 스토리지는 동일한 데이터 컨텐츠를 갖는 데이터 라인들이 상이한 암호화 결과들을 갖는 오버플로우 영역(overflow area)을 포함하는 시스템.

청구항 23

데이터 보안을 제공하기 위한 방법으로서,

스토리지에 데이터 컨텐츠를 기입하라는 요구에 응답하여, 상기 데이터 컨텐츠에 기초하여 암호화된 데이터 컨텐츠를 생성하는 단계 - 상기 데이터 컨텐츠는 복수의 도메인 중 하나의 도메인과 관련되고, 상기 암호화된 데이터 컨텐츠 생성은 상기 복수의 도메인 중 하나의 도메인과 관련된 암호화 키에 따른 것임 -;

상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도하는 단계;

상기 암호화된 데이터 컨텐츠에 대한 참조가 획득되는 경우에, 상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 변환 라인을 수정하는 단계; 및

상기 암호화된 데이터 컨텐츠에 대한 참조가 획득되지 않은 경우에:

새로운 로케이션에 상기 암호화된 데이터 컨텐츠를 저장하고;

상기 새로운 로케이션에 저장된 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하고;

상기 새로운 로케이션에 저장된 상기 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 상기 변환 라인을 수정하는 단계

를 포함하는 방법.

청구항 24

데이터 보안을 제공하기 위한 비일시적인 유형의(tangible) 컴퓨터 판독가능 저장 매체로서, 상기 비일시적인 유형의 컴퓨터 판독가능 저장 매체는 컴퓨터 명령어들을 포함하고, 상기 컴퓨터 명령어들은 하나 이상의 프로세서에 의해 실행되는 것에 응답하여,

스토리지에 데이터 컨텐츠를 기입하라는 요구에 응답하여, 상기 데이터 컨텐츠에 기초하여 암호화된 데이터 컨텐츠를 생성하고 - 상기 데이터 컨텐츠는 복수의 도메인 중 하나의 도메인과 관련되고, 상기 암호화된 데이터 컨텐츠 생성은 상기 복수의 도메인 중 하나의 도메인과 관련된 암호화 키에 따른 것임 -;

상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도하고;

상기 암호화된 데이터 컨텐츠에 대한 참조가 획득되는 경우에, 상기 스토리지 내의 상기 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 변환 라인을 수정하고;

상기 암호화된 데이터 컨텐츠에 대한 참조가 획득되지 않은 경우에:

새로운 로케이션에 상기 암호화된 데이터 컨텐츠를 저장하고;

상기 새로운 로케이션에 저장된 상기 암호화된 데이터 컨텐츠에 대한 참조를 획득하고;

상기 새로운 로케이션에 저장된 상기 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 상기 변환 라인을 수정하기 위한 것인, 비일시적인 유형의 컴퓨터 판독가능 저장 매체.

발명의 설명

기 술 분 야

[0001] **다른 출원들에 대한 상호 참조**

[0002] 본원은 모든 목적들을 위해 본원에 참조로 포함된 2014년 1월 15일자 출원된 DEDUPLICATION-BASED MEMORY ENCRYPTION이라고 하는 미국 가 특허 출원 번호 61/927,914를 우선권 주장한다.

배 경 기 술

[0003] 보안 컴퓨터 시스템들은 컴퓨터의 메인 메모리에 액세스하는 침입자들에 의한 공격들을 방지하기 위한 보안 메모리를 필요로 한다. 예를 들어, 소위 콜드 부트(cold boot) 공격에서, 공격자는 메모리가 그 컨텐츠들을 손실하지 않고서 메인 메모리 내의 다이내믹 랜덤 액세스 메모리(DRAM)로부터 데이터 컨텐츠를 추출할 수 있다. 그러므로, 메모리 내에 저장된 민감한 평문 정보가 있으면 보안이 손상된다. 이중-포트된 DRAM은 공격자가 시스템 동작 중에 DRAM 내의 판독 및 기입 트래픽을 관찰할 수 있다는 것을 의미한다. 메인 메모리를 실현하기 위해 비휘발성 랜덤 액세스 메모리(NVRAM)를 사용한다는 것은 데이터가 메인 메모리 내에 보존되고 콜드-부트 공격보다 노력을 덜 필요로 하는 공격들을 받을 수 있다는 것을 의미한다.

[0004] 현재, 종래의 스트림 암호화는 보안 메모리 암호화에 적합하지 않은 것으로 간주된다. 이것은 종래의 스트림 암호화 기술이 동일한 데이터를 동일한 암호문으로 암호화하는 2개의 인스턴스들을 방지하기 위해 의사-랜덤 값으로 암호화를 시드(seed)하는 것을 요구하기 때문이다. 이 제공이 없다면, 공격자는 동일한 암호문의 발생들의 수를 관찰함으로써 빈도 분석 공격을 할 수 있다. 이 시드 또는 초기화 벡터(IV)를 저장하는 오버헤드뿐만 아니라 이 IV를 사용하기 위해 암호화 또는 복호화를 셋업하기 위한 오버헤드는 전형적으로 암호화의 단위들로서 비교적 많은 양의 데이터를 요구한다. 예를 들어, 16바이트 IV를 사용한다는 것은 4킬로바이트의 종래의 페이지 크기의 입상도에서의 암호화가 0.1퍼센트 만의 IV에 대한 공간 오버헤드를 부과한다는 것을 의미한다. 그러나, 암호화의 단위가 프로세서 캐시와 메인 메모리 간에 요구되는 것과 같이, 캐시 라인이면, 각각의 암호화 또는 복호화 조작을 위해 회수하고 셋업하기 위해 상당한 비용이 들 것이다. 예를 들어, 종래의 캐시 라인 크기는 64바이트이므로, 16바이트 IV는 25퍼센트 공간 오버헤드를 초래할 것이다.

[0005] 보안 데이터 무결성은 또한 유사한 이유들로 인해 큰 단위들을 필요로 한다. 특히, 종래의 방식들은 데이터 단위 당 128비트 메시지 인증 코드(MAC)를 필요로 한다. 이와 같이, 페이지와 같은 큰 단위를 사용하는 것은 이 MAC의 오버헤드를 분할 상환하지만, 캐시 라인 단위는 위에 확인된 IV 오버헤드에 부가하여 상당한 오버헤드를 발생시킬 것이다.

[0006] 컴퓨터들에 대한 표준 보안/위협 모델에서, 프로세서 칩 자체는 보안성으로 되고 공격자가 이 실리콘의 수정 및 프로세서 칩 내부에 상주하는 보호된 데이터에 액세스하는 것을 제한할 수 있다고 가정한다. (예를 들어, 프로세서 칩은 칩을 수정하거나 지원되지 않은 동작들을 수행하려는 시도가 있는 경우 그것의 컨텐츠들을 파괴 또는 삭제하도록 설계될 수 있다.) 특히, 최종-레벨 캐시까지 L1, L2 등 내에 저장된 데이터는 노출 또는 손상의 위험 없이 평문으로 저장될 수 있어서, 프로세서가 매 동작마다 데이터를 복호화할 필요없이 데이터에 대해 동작

하게 하게 한 다음에 그들이 생성된 직후에 결과들을 재암호화하게 한다. 또한, 프로세서 칩의 캐시 상에 상주하는 다른 메타데이터는 유사하게 이 칩의 물리적 보안뿐만 아니라 그것의 보안 설계에 의해 보호될 수 있다. 그러나, 이러한 데이터의 양은 전력 및 비용 제한들을 포함하는, 프로세서 칩에 대한 물리적 제약들에 의해 메인 메모리에서 이용가능한 것의 일부로 상당히 제한된다. 예를 들어, 현재 온-칩 캐시 상태는 전형적으로 수십 메가바이트로 제한되는 반면 메인 메모리는 쉽게 수십 기가바이트 이상, 1000 이상의 인자일 수 있다. 한편, 프로세서 칩으로부터 저장된 데이터는 앞서 설명된 바와 같이, 예를 들어, DRAM 자체를 제거하거나, DRAM에 제2 포트로 결합함으로써 공격자에 의해 액세스가능하게 되는 것으로 추정된다.

[0007] 그러므로, 데이터의 강력한 비밀 및 무결성을 제공하면서 메모리 라인 단위들을 효율적으로 암호화 및 복호화할 필요성이 있다.

도면의 간단한 설명

[0008] 본 발명의 다양한 실시예들은 다음의 상세한 설명 및 첨부 도면에 개시된다.

도 1은 중복 제거되고, 암호화된 데이터 컨텐츠를 지원하는 시스템의 실시예를 도시한 블록도.

도 2a 및 도 2b는 다수의 중복 제거 도메인들의 실시예들을 도시한 데이터 구조도들.

도 3은 데이터를 스토리지에 기입하는 프로세스의 실시예를 도시한 플로우차트.

도 4는 스토리지 내의 로케이션으로부터 데이터를 판독하는 프로세스의 실시예를 도시한 플로우차트.

도 5는 캐시로부터 데이터 라인을 축출하는 프로세스의 실시예를 도시한 플로우차트.

도 6은 초기화 벡터(IV)를 포함하는 예시적인 데이터 구조를 도시한 도면.

도 7a는 종래의 구현에서 구조화된 메모리를 지원하는 데 사용되는 메타데이터의 예를 도시한 데이터 구조도.

도 7b는 보안 구현에서 구조화된 메모리를 지원하는 데 사용되는 메타데이터의 예를 도시한 데이터 구조도.

발명을 실시하기 위한 구체적인 내용

[0009] 본 발명은 프로세스; 장치; 시스템; 물질의 조성물; 컴퓨터 판독가능 저장 매체 상에 구현된 컴퓨터 프로그램 제품; 및/또는 프로세서에 결합된 메모리에 저장되고/되거나 그에 의해 제공되는 명령어들을 실행하도록 구성된 프로세서와 같은 프로세서로서 포함하는 많은 방식으로 구현될 수 있다. 본 명세서에서, 이들 구현, 또는 본 발명이 취할 수 있는 기타 형태가 기술들로서 참조될 수 있다. 일반적으로, 개시된 프로세스들의 단계들의 순서는 본 발명의 범위 내에서 변경될 수 있다. 달리 표명하지 않는다면, 태스크를 수행하도록 구성되는 것으로 설명된 프로세서 또는 메모리와 같은 소자는 주어진 시간에 태스크를 수행하도록 임시로 구성된 일반적인 소자 또는 태스크를 수행하도록 제작된 특정한 소자로서 구현될 수 있다. 여기에 사용되는 바와 같이, '프로세서'라는 용어는 컴퓨터 프로그램 명령어들과 같은 데이터를 처리하도록 구성된 하나 이상의 디바이스, 회로, 및/또는 프로세싱 코어를 말한다.

[0010] 본 발명의 하나 이상의 실시예의 상세한 설명은 본 발명의 원리들을 도시한 첨부 도면과 함께 아래에 제공된다. 본 발명은 이러한 실시예들과 관련하여 설명되지만, 본 발명은 어떤 실시예로 제한되지 않는다. 본 발명의 범위는 청구범위에 의해서만 제한되고 본 발명은 많은 대체들, 수정들 및 균등물들을 포함한다. 많은 특정한 상세들이 본 발명의 철저한 이해를 제공하기 위해 다음의 설명에서 기술된다. 이들 상세는 예의 목적을 위해 제공되고 본 발명은 이들 특정한 상세들의 일부 또는 모두 없이 청구범위에 따라 실시될 수 있다. 명료성의 목적을 위해, 본 발명과 관련된 기술적 분야들에서 공지된 기술적 자료는 본 발명이 불필요하게 불명하게 되지 않도록 상세히 설명되지 않았다.

[0011] 중복 제거된 스토리지의 암호화 및 복호화를 사용하는 컴퓨터 보안이 개시된다. 일부 실시예들에서, 스토리지에 데이터 컨텐츠를 기입하라는 요구에 응답하여, 데이터 컨텐츠에 기초하여 암호화된 데이터 컨텐츠가 생성된다. 스토리지 내의 암호화된 데이터 컨텐츠에 대한 참조를 획득하려는 시도가 이루어진다. 암호화된 데이터 컨텐츠에 대한 참조가 획득되는 경우에, 스토리지 내의 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 변환 라인(translation line)이 수정된다. 암호화된 데이터 컨텐츠에 대한 참조가 획득되지 않은 경우에, 암호화된 데이터 컨텐츠는 새로운 로케이션에 저장되고; 새로운 로케이션에 저장된 암호화된 데이터 컨텐츠에 대한 참조가 획득되고; 변환 라인은 새로운 로케이션에 저장된 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해

수정된다.

[0012] 도 1은 중복 제거되고, 암호화된 데이터 컨텐츠를 지원하는 시스템의 실시예를 도시한 블록도이다. 이 예에서, 시스템(100)은 스토리지(112)에 접속된 프로세서 칩(102)을 포함한다. 프로세서 칩(102)은 계층적 캐시(106)에 접속된 하나 이상의 프로세서 코어(104)를 포함하는, 많은 회로 소자들을 갖는다. 3-레벨 캐시 계층이 이 예에서 예시되지만, 캐시 레벨이 상이한 다른 캐시 계층들이 사용될 수 있다. 캐시 내에는, 데이터 및 코드가 암호화되지 않은 평문으로 저장된다. 캐시(구체적으로, 최종 레벨 캐시(LLC)(109)는 하나 이상의 통신 인터페이스 및 암호화/복호화 엔진(EDE)(110)을 포함하는 메모리 제어기(108)에 접속된다.

[0013] 메모리 제어기는 스토리지(112)에 접속된다. 다양한 실시예들에서, 스토리지(112)는 다이내믹 랜덤 액세스 메모리(DRAM), 스태틱 랜덤 액세스 메모리(SRAM), 위상 변화 메모리(PCM), 랜덤 액세스 메모리(RAM)와 같은 하나 이상의 메모리 디바이스(또한 메인 메모리라고 함); 비휘발성 랜덤 액세스 메모리(NVRAM), 광학 및 자기 디스크, 플래시 드라이브와 같은 하나 이상의 보조 스토리지 디바이스; 또는 하나 이상의 메모리 디바이스와 보조 스토리지 디바이스의 조합을 포함한다. 예시의 목적들을 위해, 물리적 메모리를 사용하여 구현된 스토리지를 갖는 예들은 아래에 광범위하게 논의되지만, 기술들은 또한 보조 스토리지 디바이스들, 또는 메모리와 보조 스토리지의 조합을 사용하여 구현된 스토리지에 또한 적용가능하다.

[0014] 이 예에서, 메모리 제어기(108)와 스토리지(112)는 함께 캘리포니아주의 멘로 파크의 Hicamp Systems사에 의한 계층적 불변 컨텐츠 어드레스가능한 메모리 프로세서(Hierarchical Immutable Content Addressable Memory Processor)(HICAMP™)와 같은 구조화된 메모리를 구현한다. 다른 적절한 데이터 라인-기반 메모리 아키텍처들이 사용될 수 있다. 각각의 데이터 라인은 데이터 라인이 할당 해제되어 새로운 컨텐츠로 재할당될 때까지 변하지 않은 채로 남은 컨텐츠를 저장하도록 구성된다. HICAMP와 같은 구조화되고, 중복 제거된 메모리의 구현 및 동작들은 본 기술 분야의 통상의 기술자에게 공지되어 있다.

[0015] 도시한 바와 같이, 스토리지의 부분은 작은(예를 들어, 수 내지 수백 바이트 정도) 데이터 라인들(114)의 어레이를 포함한다. 각각의 데이터 라인은 데이터 라인 식별자(PLID)에 의해 어드레스된다. 일부 실시예들에서, 각각의 데이터 라인은 라인의 수명 동안 불변인 채로 남은 컨텐츠들을 갖는다. 바꾸어 말하면, 데이터 라인이 생성되어 데이터로 팝퓰레이트(populate)될 때, 그것의 컨텐츠들은 메모리가 할당 해제되어 상이한 컨텐츠들을 갖도록 재할당될 때까지 변하지 않는다. 일부 실시예들에서, 각각의 데이터 라인의 데이터 컨텐츠는 중복 제거된다. 바꾸어 말하면, 각각의 데이터 라인은 유일한 데이터 컨텐츠를 갖는다.

[0016] 이 예에서, 각각의 데이터 라인은 암호화된 데이터 컨텐츠의 블록을 포함한다. 저장되고 암호화된 데이터 라인은 2개 이상의 별개의 어드레스들이 동일한 실제 데이터 라인을 참조하게 하는 인다이렉션(indirection)의 레벨로 액세스된다. 또한, 암호화된 데이터 라인들은 중복 제거된다. 인다이렉트 액세스 및 중복 제거의 상세들은 아래에 설명된다. 일부 실시예들에서, 엄격한 중복 제거가 요구되지 않고 상이한 데이터 라인들이 소정의 상황들 하에서 동일한 컨텐츠들을 저장하는 것이 허용된다. 잠재적으로 중복된 데이터 라인들은 중복 제거된 데이터 라인들과 별개로 유지된다.

[0017] 일부 실시예들에서, 데이터 라인들에 액세스하기 위한 인다이렉션의 레벨은 변환 라인들을 사용하여 제공된다. 도시한 도면에서, 스토리지(112)의 부분은 변환 라인들(116)을 저장한다. 변환 라인들(또한 인다이렉트 라인들이라고 함)은 특정한 컨텐츠(예를 들어, 페이지의 컨텐츠)를 포함하는 데이터 라인들의 순서화된 세트의 논리적 구성을 형성한다. 변환 라인은 데이터 라인들의 세트를 참조하는 것이라고 하고, 이는 인다이렉트 라인이 데이터 라인들의 어드레스들 또는 식별자들(예를 들어, 데이터 라인들의 PLID들)을 포함하거나, 그렇지 않으면 데이터 라인들의 그것의 대응하는 세트에 관련된다는 것을 의미한다. 도시한 예에서, 변환 라인은 대응하는 데이터 라인들에 대한 PLID 값들을 포함하는 다수의 엔트리들을 포함한다. 데이터 라인들은 중복 제거되기 때문에, 변환 라인들은 동일한 데이터 컨텐츠에 대한 동일한 데이터 라인을 참조한다. 예를 들어, 변환 라인들 1 및 2는 둘 다 1의 PLID를 갖는 데이터 라인을 참조한다.

[0018] 이 예 및 다른 예에서 도시한 변환 라인들 및 데이터 라인들의 형식, 크기, 및 수는 단지 예시의 목적들을 위한 것이고 다른 실시예들에서는 변화할 수 있다. 예를 들어, 일부 실시예들에서, 각각의 데이터 라인은 64바이트이고, 각각의 인다이렉트 라인은 (공유가능한 데이터 라인을 위한 식별자(또는 어드레스)를 각각 포함하는 64개의 4바이트 엔트리들을 포함하는) 256바이트이다. 고정된 크기들이 아래의 예들에서 논의되지만, 일부 실시예들에서, 가변 크기들이 허용된다(예를 들어, 시스템은 단위 크기들이 상이한 모드들에 대해 변화하는 다수의 모드들을 지원할 수 있다). 일부 실시예들에서, 추가의 메타데이터가 데이터 라인들 및/또는 변환 라인들 내에 포함된다. 일부 실시예들에서, 데이터 라인들은 데이터 라인들이 디렉토리 내의 그들의 컨텐츠들을 룩업함으로

써 어드레스가능한 컨텐츠 디렉토리 내에 배열된다.

[0019] 일부 실시예들에서, 스토리지는 세그먼트들의 수로서 액세스되고, 여기서 각각의 세그먼트는 데이터 라인들의 논리적으로 연속하는 시퀀스이고 방향성 비사이클 그래프("DAG")로서 구조화된다. 세그먼트 테이블은 각각의 세그먼트를 DAG의 루트(root)를 나타내는 PLID로 맵핑한다. 세그먼트들은 세그먼트 식별자들("SegID")에 의해 식별되고 액세스된다. 프로세서 내의 특별 목적 레지스터들(반복자 레지스터들이라고 함)은 DAG로부터의 데이터 로딩, 세그먼트 컨텐츠들의 반복, 프리페칭, 및 업데이트들을 포함하는, 세그먼트들 내에 저장된 데이터에의 효율적인 액세스를 가능하게 한다.

[0020] 도시한 예에서, 데이터 라인의 컨텐츠들은 En(x)로 나타내진 암호화된 데이터 컨텐츠를 포함하는데, 여기서 x는 암호화되지 않은 데이터 컨텐츠이다. 각각의 데이터 라인의 데이터 컨텐츠가 중복 제거되고 유일하기 때문에, 컨텐츠의 특정한 피스(예를 들어, "GOOD"의 암호화된 버전 또는 En("GOOD"))만이 데이터 라인들 중에서 한번 나타난다. 암호화된 블록들의 중복 제거로 인해, 블록 암호화에 대한 널리 공지된 공격들, 특히 빈도 분석 공격들은 데이터 컨텐츠의 주어진 피스의 기껏해야 하나의 카피가 저장되기 때문에 성공하지 못할 것이다. 이 중복 제거는 암호화된 미디어 확장들(EME)과 같은 블록 암호화 방식들이 EME가 "유일한 메시지들"을 위해 안전하다는 것을 나타내는 광범위한 연구에 의존하는, 데이터 블록들에 적용될 수 있다는 것을 의미한다. 이 응용에서, 중복 제거 메커니즘은 데이터 라인들이 컨텐츠들로서 유일한 메시지들을 포함하는 것을 보장한다.

[0021] 일부 실시예들에서, 변환 라인들은 또한 암호화된다. 데이터 라인 컨텐츠들 및/또는 변환 라인들을 어떻게 암호화, 복호화, 및 액세스하는지에 대한 상세들이 아래에 설명된다.

[0022] 메모리 제어기는 아래에 더 상세히 설명되는 컨텐츠 디렉토리-기반 기술을 사용하여 메모리 중복 제거를 지원한다. 변환 라인들 및 데이터 라인들은 필요한 대로 캐시(106) 내로 로드되거나, 그에 라이트백(write back)되거나, 또는 그로부터 축출된다. 메모리 제어기 내의 EDE는 캐시(106)로부터 획득되고 스토리지(112) 내에 저장될 데이터를 암호화하고, 스토리지(112)로부터 회수되고 캐시(106)로 전달될 데이터를 복호화한다.

[0023] 위에 설명된 메모리 제어기 및 EDE와 같은 모듈들은 하나 이상의 프로세서 상에서 실행하는 소프트웨어 소자들, 프로그래머블 논리 디바이스들 및/또는 소정의 기능을 수행하도록 설계된 주문형 집적 회로들과 같은 하드웨어 또는 이들의 조합으로서 구현될 수 있다. 일부 실시예들에서, 모듈들은 (퍼스널 컴퓨터들, 서버들, 네트워크 장비 등과 같은) 컴퓨터 디바이스가 본 출원의 실시예들에서 설명된 방법들을 구현하게 하기 위한 다수의 명령어들을 포함하는, (광학 디스크, 플래시 스토리지 디바이스, 모바일 하드 디스크 등과 같은) 비휘발성 저장 매체 내에 저장될 수 있는 소프트웨어 제품들의 형태로 실시될 수 있다. 모듈들은 단일 디바이스 상에 구현되거나 다수의 디바이스들에 걸쳐 분배될 수 있다. 모듈의 기능들은 서로 병합될 수 있거나 다수의 서브-모듈들로 더 나누어질 수 있다.

[0024] 도 2a 및 도 2b는 다수의 중복 제거 도메인들의 실시예들을 도시한 데이터 구조도이다. 도 2a 및 도 2b에 도시한 데이터 구조들의 상세들이 아래에 논의된다.

[0025] 도 3은 데이터를 스토리지에 기입하는 프로세스의 실시예를 도시한 플로우차트이다. 프로세스 300은 100과 같은 시스템 상에서, 및 특히 메모리 제어기(108)에 의해 수행될 수 있다.

[0026] 302에서, 스토리지에 소정의 데이터 컨텐츠를 기입하라는 요구가 수신된다. 일부 실시예들에서, 이 요구는 프로세서에 의해 발행되고 메모리 제어기에 의해 수신된다. 일부 실시예들에서, 이 요구는 메인 메모리 내의 물리 어드레스를 특정하고, 이 요구에 응답하여, 메모리 제어기는 물리 어드레스를 변환 함수, 또는 기타 적절한 기술들을 적용하는, 맵핑 테이블에서 룩업함으로써 변환 라인 엔트리에 대한 참조로 변환한다. 예를 들어, 이 요구는 데이터 컨텐츠 "hello"를 1100의 물리 어드레스에 기입하는 것을 포함할 수 있다. 메모리 제어기는 변환을 수행하고 110의 식별자를 갖는 변환 라인의 제1 엔트리가 이 물리 어드레스에 대응한다는 것을 결정할 것이다.

[0027] 304에서, 이 요구에 응답하여, 암호화된 컨텐츠는 기입된 데이터 컨텐츠에 기초하여 생성된다. 일부 실시예들에서, 암호화 함수 En(x)는 암호화된 출력을 생성하기 위해 데이터 컨텐츠 x에 적용된다. 이 예에서, 암호화된 컨텐츠는 En("hello")으로서 표현된다. En(x)는 필요에 따라 획득되는, 보안 키(security key) 및/또는 초기화 벡터(IV) 값과 같은 추가의 입력 파라미터들을 요구할 수 있다. 일부 실시예들에서, 보안 키는 특정한 레지스터와 같은 알려진 로케이션 내에 저장된다. 다수의 도메인들이 사용되는 실시예들에서, 적절한 키가 현재의 도메인을 위해 선택된다.

[0028] 다양한 암호화 기술들이 구현될 수 있다. 일부 실시예들에서, EDE는 암호화된 데이터 컨텐츠를 생성하기 위해

결정론적 암호화 기능을 수행한다. 결정론적 블록 암호화 방식(확률적 암호화 방식과 반대)은 항상 입력 데이터의 주어진 블록에 대해 동일한 암호문을 생성한다. 별개의 암호화 키 또는 IV는 데이터 라인들에 대해 요구되지 않는데 왜냐하면 데이터 라인들은 중복 제거되고 상이한 암호화된 결과들을 저장하므로, 빈도 분석 공격들을 받지 않기 때문이다. 일부 실시예들에서, 메모리 제어기 내의 EDE는 16바이트의 블록 크기를 위한 블록 암호화를 제공하기 위해 고급 암호화 표준(Advanced Encryption Standard)(AES) 기술들을 구현한다. 일부 실시예들에서, EDE는 종래의 프로세서의 64바이트 캐시 라인들을 처리하는 와이드-블록 보안 의사-랜덤 순열(PRP)을 구성하기 위해 ECB-Mix-ECB(EME) 기술들을 구현한다. EME의 강력한 병렬화는 다수의 독립적인 하드웨어 서브모듈들이 복호화를 병렬로 수행하게 함으로써, 캐시 미스에 대한 레이턴시를 최소화한다. 이것은 캐시 라인 축출 시간을 감소시키는, 암호화에 적용될 수 있다. EME는 예들에서 도시된 것보다 작은 캐시 라인 크기뿐만 아니라 큰 캐시 라인 크기를 지원하기 위해 적용될 수 있다. AES 및 EME와 같은 기술들이 예의 목적들을 위해 광범위하게 논의되지만, 다른 적절한 암호화/복호화 기술들이 사용될 수 있다. 예를 들어, XEX-TCB-CTS(XTS), CBC-Mask-CBC(CMC) 및 다른 길이-보존 암호화/복호화 기술들이 다른 실시예들에서 사용될 수 있다.

[0029] 305에서, 암호화된 데이터 컨텐츠에 대한 참조를 획득하려는 시도가 이루어진다. 다양한 실시예들에서, 이 참조는 소정의 컨텐츠에 액세스하기 위한 포인터, 어드레스, 핸들, 식별자, 또는 기타 적절한 표시일 수 있다. 일부 실시예들에서, 암호화된 데이터 컨텐츠에 대한 참조를 획득하려는 시도는 암호화된 데이터 컨텐츠를 고려하여 참조를 획득하기 위한 동작을 수행하는 것을 포함한다. 예를 들어, 암호화된 데이터 컨텐츠 En("hello")의 룩업 동작이 수행된다. 일부 실시예들에서, 이러한 동작은 구조화된 메모리 구현에 의해 지원된다.

[0030] 306에서, 암호화된 데이터 컨텐츠에 대한 참조가 성공적으로 획득되는지가 결정된다. 암호화된 데이터 컨텐츠가 이미 스토리지 내에 존재하면, 암호화된 데이터 컨텐츠에 대한 참조가 성공적으로 획득된다. 예를 들어, En("hello")이 스토리지 내에 이미 존재하고 이 컨텐츠를 저장한 데이터 라인이 14의 PLID를 가지면, 기존의 암호화된 컨텐츠 En("hello")을 참조하는 14의 PLID가 획득된다. 아래에 더 상세히 설명되는 바와 같이, 일부 실시예들에서, 암호화된 데이터 컨텐츠는 특정한 컨텐츠의 빠른 룩업을 허용하는 컨텐츠 디렉토리/해시 테이블 내에 저장된다.

[0031] 암호화된 데이터 컨텐츠에 대한 참조가 성공적으로 획득되면, 312에서, 변환 라인은 획득된 참조를 참고하기 위해 수정된다. 일부 실시예들에서, 변환 라인은 참조 자체를 저장하고; 일부 실시예들에서, 변환 라인은 참조와 관련된 어드레스, 포인터, 핸들 등을 저장한다. 이 예에서, (1100의 요구된 물리 어드레스에 대응하는) 변환 라인 110의 제1 엔트리는 PLID 14인 획득된 참조를 참고하기 위해 수정된다.

[0032] 그러나, 암호화된 컨텐츠가 스토리지 내에 이미 존재하지 않고 암호화된 데이터 컨텐츠에 대한 참조가 획득되지 않으면, 310에서, 암호화된 컨텐츠는 새로운 로케이션에 저장되고 이 새로운 로케이션에 저장된 암호화된 컨텐츠에 대한 참조가 획득된다. 예를 들어, En("hello")이 스토리지 내에 이미 존재하지 않는다고 결정되면, 19의 PLID를 갖는 새로운 데이터 라인이 En("hello")을 저장하도록 생성되고, PLID 19가 획득된다. 314에서, 변환 라인은 새로운 로케이션에 저장된 암호화된 컨텐츠에 대한 참조를 참고하기 위해 수정된다. 이 예에서, (물리 어드레스 1100에 대응하는) 변환 라인 110의 제1 엔트리가 PLID 19를 참조하기 위해 수정된다.

[0033] 도 4는 스토리지 내의 로케이션으로부터 데이터를 판독하는 프로세스의 실시예를 도시한 플로우차트이다. 프로세스 400은 100과 같은 시스템 상에서, 및 특히 메모리 제어기(108)에 의해 수행될 수 있다.

[0034] 402에서, 로케이션(또한 판독 로케이션이라고 함)에 있는 데이터 컨텐츠에 액세스하라는 요구가 수신된다. 일부 실시예들에서, 이 요구는 프로세서에 의해 이루어지고 이 요구에서 특정된 로케이션은 메인 메모리 내의 물리 어드레스에 대응한다.

[0035] 404에서, 물리 어드레스는 변환 함수 등을 적용하는, 맵핑 테이블에서 룩업함으로써 변환 라인 로케이션(예를 들어, 변환 라인 내의 엔트리의 어드레스)으로 변환된다. 예를 들어, 물리 어드레스 1200에서의 데이터 컨텐츠에 대한 요구는 19의 PLID를 저장한, 변환 라인 120의 엔트리 2에 대응할 수 있다.

[0036] 406에서, 이 요구에 응답하여, 데이터 컨텐츠가 캐시에서 이용가능한 지가 결정된다. 구체적으로, 캐시 내의 PLID를 갖는 변환 라인 및 대응하는 데이터 라인의 이용가능성들이 체크된다. 4개의 가능성: 미스/미스(즉, 변환 라인과 데이터 라인이 캐시에서 이용가능하지 않음); 히트/미스(즉, 변환 라인은 캐시에서 이용가능하지만 데이터 라인은 그렇지 않음); 히트/히트(변환 라인과 데이터 라인 둘 다가 캐시에서 이용가능함); 미스/히트(즉, 변환 라인은 캐시에서 이용가능하지 않지만 데이터 라인은 캐시에서 이용가능함)가 있다.

[0037] 408에서, 미스/미스 경우가 처리된다. 구체적으로, PLID를 갖는 변환 라인 및 데이터 라인은 둘 다 메인 메모

리로부터 로드되고, 필요한 대로 복호화된다. 예를 들어, 변환 라인 120과 PLID 19의 데이터 라인이 캐시에서 이용가능하지 않으면, (그것의 PLID 엔트리들을 포함하는) 변환 라인 120은 스토리지로부터 캐시 내로 로드되고, 그것의 대응하는 암호화된 컨텐츠들을 갖는 PLID 19의 데이터 라인은 또한 메인 메모리로부터 로드되어, 복호화되고, 복호화된 결과는 캐시 내로 저장된다. 일부 실시예들에서, 변환 라인 120 내의 모든 엔트리들의 컨텐츠들은 복호화되어 캐시 내로 로드되고; 일부 실시예들에서, 요구된(이 경우에, PLID 19) 특정한 데이터 라인 엔트리의 컨텐츠들 만이 복호화되어 캐시 내로 로드된다. 캐시 내의 데이터는 다음에 요구자에 제시된다. 일부 실시예들에서, 변환 라인들은 중복 제거되지 않으므로 빈도 분석 공격들로부터의 보호를 필요로 한다. IV들, MAC들, 또는 다른 보안 메모리 기술들을 사용하는 암호화/복호화 기술들이 변환 라인들을 보호하는 데 사용될 수 있다.

[0038] 410에서, 히트/미스 경우가 처리된다. 구체적으로, 변환 라인이 캐시 내에 이미 존재하고, 데이터 라인에 대한 암호화된 컨텐츠들이 메인 메모리로부터 로드되고, 복호화되어, 캐시에 저장된다. 예를 들어, 변환 라인 120이 캐시 내에 이미 존재하지만 PLID 19를 갖는 데이터 라인은 그렇지 않으면, 데이터 라인의 암호화된 컨텐츠들은 메인 메모리로부터 로드되고, 복호화되어, 캐시 내로 저장된다. 변환 라인 내의 모든 엔트리들의 컨텐츠들이 복호화되어 미스/미스 상황에서 캐시 내로 로드되는 실시예들에서, 변환 라인이 캐시 내에 이미 존재할 때, 그것의 대응하는 PLID 엔트리들도 그러하므로, 히트/미스 시나리오는 비교적 희박하게 생길 가능성이 크다는 점에 주목한다.

[0039] 414에서, 히트/히트 경우가 처리된다. 이러한 경우에, 변환 라인 및 데이터 라인 둘 다는 캐시 내에 존재하고 메인 메모리로부터의 추가의 회수가 요구되지 않는다. 캐시 내의 데이터 라인의 데이터 컨텐츠는 요구자에 제시된다. 캐시 내에 저장된 것은 평문으로 복호화되기 때문에 복호화는 요구되지 않는다는 점에 주목한다.

[0040] 412에서, 미스/히트 경우가 처리된다. 이러한 경우에, 변환 라인은 캐시 내로 로드되지만, 데이터 라인은 캐시 내에 이미 존재하기 때문에 판독 또는 변환될 필요가 없다.

[0041] 414에서, 캐시 내의 데이터 라인의 데이터 컨텐츠가 요구자에 제시된다. 예를 들어, 캐시 내의 PLID 19의 복호화된 데이터 컨텐츠가 요구자에 제시된다.

[0042] 캐시 액세스 프로세스는 i) 변환 라인 및 그것의 대응하는 데이터 라인들이 캐시에서 자주 발견되어, 히트/히트 시나리오가 꽤 일반적이게 하고; ii) 암호화 키가 사용 시마다 프로세서 코어 밖으로부터 회수될 필요가 없도록 프로세서 상에서 실행하는 프로세스에 걸쳐 일반적이고; iii) 프로세서 액세스는 데이터가 평문으로 저장된 프로세서 캐시로부터 만족되므로 메인 메모리에의 액세스는 대부분의 경우에서 피해지기 때문에 효율적이다. 그러므로, 이들 경우에서, 사용 전의 복호화 또는 메인 메모리 액세스가 요구되지 않는다.

[0043] 일부 실시예들에서, EDE는 암호화된 메모리 라인을 페칭하는 것과 동시에 캐시 라인 미스 시에 보안 패드를 생성(그리고 따라서 암호화된 메모리로부터 페치)하도록 구현된다. 여기에 사용되는 바와 같이, 보안 패드는 복호화 중에 마스크로서 기능하는 이진 값들의 시퀀스를 말한다. 보안 패드를 생성하기 위한 기술들은 본 기술분야의 통상의 기술자에게 공지되어 있다. 복호화는 다음에 암호화된 데이터 라인이 메인 메모리로부터 수신될 때 평문 데이터를 생성하기 위해 보안 패드로 암호화된 라인의 간단한 XOR을 요구함으로써, 복호화의 레이턴시 영향을 최소화한다.

[0044] 캐시 공간이 제한되기 때문에, 일부 캐시 라인들 내의 기존의 데이터는 캐시 라인들이 새로운 데이터에 대해 이용가능하게 되기 위해 가끔 축출될 필요가 있다. 축출될 캐시 라인들 내의 수정된 데이터는 스토리지에 라이트백 되어야 한다. 도 5는 캐시로부터 데이터 라인을 축출하는 프로세스의 실시예를 도시한 플로우차트이다. 프로세스 500은 100과 같은 시스템 상에서, 및 특히 메모리 제어기(108)에 의해 수행될 수 있다.

[0045] 502에서, 데이터 라인을 저장한 캐시 라인을 추출하라는 요구가 수신된다. 예를 들어, 21의 PLID를 갖는 데이터 라인 및 "hello"의 데이터 컨텐츠를 저장한 캐시 라인을 축출하라는 요구가 수신된다.

[0046] 504에서, 그것의 컨텐츠들이 스토리지(예를 들어, 메인 메모리)에 최종 기입된 이후 캐시 라인이 수정되었는지가 결정된다. 일부 실시예들에서, 데이터 라인이 수정될 때마다 설정되는 "더티(dirty)" 플래그가 있다. 이 결정은 이 플래그를 체크함으로써 이루어진다.

[0047] 데이터 라인을 저장한 캐시 라인이 그것의 컨텐츠들이 스토리지에 최종 기입된 이후 수정되지 않았다고 결정되면, 506에서, 데이터 라인은 캐시로부터 제거될 수 있고 캐시 라인은 비워지거나 이용가능한 것으로 표시된다. 축출 프로세스는 종료된다.

[0048] 그러나, 데이터 라인을 저장한 캐시 라인이 그것의 컨텐츠들이 스토리지에 최종 기입된 이후 수정되었다고 결정되면, 수정이 스토리지에 기입되어야 한다. 라이트백(writeback) 동작이 508에서 수행된다. 이 예에서, 라이트백 동작은 캐시 라인 내의 수정된 컨텐츠가 스토리지에 기입되는 도 3의 프로세스 300과 동일하다.

[0049] 상기 예들에서, 스토리지 내의 데이터 라인들은 암호화되고 중복 제거된다. 일부 실시예들에서, 캐시(예를 들어, LLC)는 특정한 데이터 컨텐츠가 이미 캐시 내에 존재하는지를 결정하는 동일한 컨텐츠 디렉토리-기반 룩업 메커니즘을 구현함으로써 중복 제거를 또한 지원하므로, 캐시 레벨에서의 중복들을 제거함으로써 캐시 리소스들(예를 들어, 캐시 공간)에 대한 요구를 감소시킨다.

[0050] 위에 설명된 기술들을 사용하여, 프로세서는 메모리 내의 데이터의 고도의 보안성인 암호화를 보장하고 다량의 메인 메모리로 스케일하면서, 캐시 미스 시의 메모리 라인들의 효율적인 낮은 레이턴시 복호화를 구현하고, 라이트백 시의 암호화를 위해서도 유사하게 구현할 수 있다.

[0051] **암호화된 변환 라인들**

[0052] 위에 논의된 예들에서, 변환 라인들은 암호화되지 않는다. 일부 실시예들에서, 변환 라인들은 암호화된다. 그러므로, LLC 내에 있지 않은 변환 라인에 액세스하는 것은 데이터 라인들에 대한 유사한 단계들을 수행함으로써, 즉 이 라인을 LLC 내로 로드할 때 변환 라인을 복호화함으로써 처리된다. 유사하게, 수정된 변환 라인을 라이트백하는 것은 변환 라인을 그것의 암호화된 형태로 암호화하는 것을 수반한다.

[0053] 변환 라인들은 반드시 중복 제거되지 않는다는 점에 주목한다. 도 6은 초기화 벡터(IV)를 포함하는 예시적인 데이터 구조를 도시한 도면이다. IV 값은 암호화 중에 EDE에 의해 사용된 랜덤화된 파라미터를 포함한다. 동일한 데이터 컨텐츠가 상이한 IV 값들을 사용하여 암호화될 때, 결과적인 암호화된 결과들은 상이하다. 이 예에서, 각각의 변환 라인은 IV 필드를 포함하고, 중복 제거되지 않은 변환 라인들에는 상이한 IV 값들이 할당된다. 예를 들어, 변환 라인들 100 및 502는 동일한 컨텐츠들을 갖지만 상이한 IV 값들이 할당된다. 일부 실시예들에서, EDE는 결정론적 블록 암호화를 수행한다. 도시한 바와 같이, 동일한 데이터를 포함하는 변환 라인들은 상이한 IV 값들이 암호화 엔진에 의해 입력의 부분으로서 사용되기 때문에 상이한 암호화된 결과들을 산출하므로, 빈도 분석 공격들로부터 변환 라인들을 보호한다. 더구나, 변환 라인들은 전체 메모리의 작은 퍼센티지이기 때문에, 변환 라인 당 IV 값을 포함하는 것은 상당한 공간 오버헤드를 도입시키지 않는다.

[0054] 일부 실시예들에서, IV에 부가하여, 키는 암호화/복호화를 수행하는 것이 EDE에 의해 요구된다. 키는 공간 또는 액세스 오버헤드를 최소화하기 위해 레지스터 또는 다른 알려진 로케이션 내에 저장될 수 있다.

[0055] 일부 실시예들에서, 변환 라인들을 위해 사용된 키는 데이터 라인들을 위해 사용된 키와 상이하다.

[0056] 일부 실시예들에서, 다수 세트들의 변환 라인들이 있고, 별개의 키가 별개의 세트의 변환 라인들에 의해 사용된다. 일부 실시예들에서, 별개의 키들은 상이한 보호 도메인들 내의 상이한 변환 라인들을 위해 사용된다. 예를 들어, 다수 세트들의 변환 라인들이 머신 상의 상이한 프로세스들을 위해 할당되고, 별개의 키가 각각의 프로세스에 대응하는 별개의 세트의 변환 라인들을 위해 사용된다. 또 하나의 예로서, 프로세서의 어드레스 공간은 상이한 세그먼트들로 나누어지고, 여기서 일부 세그먼트들은 다수의 프로세스들에 의해 공유된다. 그러므로, 공유된 세그먼트는 공유된 세그먼트에 액세스할 수 있는 프로세스들 중에 공유되는 키를 사용하여 암호화되고 개인 세그먼트들은 개인의 공유되지 않은 키들을 사용하여 암호화된다. 다수 세트들의 변환 라인들의 다른 할당들이 가능하다.

[0057] **암호화된 컨텐츠 디렉토리 메타데이터**

[0058] 도 7a는 종래의 구현에서 구조화된 메모리를 지원하는 데 사용되는 메타데이터의 예를 도시한 데이터 구조도이다. 이 예에서, 구조화된 메모리는 데이터 컨텐츠 및 엔트리의 서명과 같은 대응하는 메타데이터를 저장한 메인 메모리 내의 해시 테이블인 컨텐츠 디렉토리(800)에 의해 지원된다. 특정한 데이터 컨텐츠가 이미 메모리에 존재하는 지를 결정하기 위해서, 데이터 컨텐츠는 해시 테이블에서 해시되고 룩업된다. 그러므로, 각각의 데이터 라인에 관련된 해시 인덱스가 유효하게 있다. 예를 들어, 암호화를 포함하지 않는 종래의 시스템에서, 어드레스 Ai에 할당된 라인은 Ai로부터 해시 테이블의 기본 어드레스(Bi)를 빼고, 그 결과를 버킷(S)의 크기로 나눔으로써 버킷과 관련되는 것으로 결정될 수 있다. 메타데이터는 해시 인덱스를 포함한다.

[0059] 이 예에서, 메타데이터는 해시 버킷 내의 특정한 엔트리를 룩업하기 위해 보조 인덱스로서 사용되는, 데이터 컨텐츠의 별개의 해시로서 계산된, 해시 버킷 내의 각각의 엔트리와 관련된 서명을 포함한다.

[0060] 이 예에서, 이 엔트리를 참조하는 변환 라인들의 수를 표시하는, 데이터 라인 엔트리 당 참조 계수가 있다. 참

조 계수가 0에 도달할 때, 데이터 라인은 수집된 가비지일 수 있고 그것의 메모리는 자유롭게 된다.

[0061] 도 7b는 보안 구현에서 구조화된 메모리를 지원하는 데 사용되는 메타데이터의 예를 도시한 데이터 구조도이다. 이 예에서, 메타데이터 값들은 허가되지 않은 수정이 검출될 수 있도록 암호화된다. 구체적으로, 해시 인덱스는 암호화된 데이터 컨텐츠(En("hello"))에 대한 보안 키 해시(secure keyed hash)(SecHash)를 사용하여 계산된다. 암호화된 데이터 컨텐츠는 해시 인덱스에 대응하는 테이블 내의 로케이션에 저장되고, 암호화된 컨텐츠의 룩업은 이 해시 인덱스를 사용하여 수행된다. 관련된 서명 라인은 유사하게 계산된다. 일부 실시예들에서, 해시 인덱스와 서명은 보안 키 해시의 상이한 절사된 부분들, 예를 들어, 해시-기반 메시지 인증 코드(HMAC)이다. 유사하게, 참조 계수들이 암호화될 수 있다. 데이터 컨텐츠와 메타데이터의 불일치는 허가되지 않은 수정과 같은 잠재적으로 보안 침해가 있다는 것을 표시한다.

[0062] 다른 실시예들에서, 상이한 메타데이터 구조 및 룩업 처리가 사용될 수 있다. 예를 들어, 일부 실시예들에서, 해시는 데이터 라인의 암호화되지 않은 데이터 컨텐츠에 대해 계산된다. 데이터 컨텐츠는 다음에 암호화되어 계산된 로케이션(즉, 암호화되지 않은 데이터 컨텐츠에 대응하는 해시 테이블 내의 로케이션)에 저장된다. 이러한 실시예들에서, 암호화된 컨텐츠를 룩업하기 위해 또는 데이터 라인의 암호화된 데이터 컨텐츠에 대한 참조를 획득하기 위해, 암호화된 데이터 컨텐츠가 복호화되고, 해시 엔트리 로케이션이 복호화된 결과에 기초하여 결정된다.

[0063] **데이터 키 도메인들**

[0064] 일부 실시예들에서, 별개의 보호 도메인들은 데이터 라인들을 암호화하기 위해 별개의 데이터 암호화 키들을 사용한다. 이것은 이들 도메인 사이의 증가된 보호를 제공한다. 여기에 사용되는 바와 같이, 보호 도메인은 어느 데이터 라인들이 선택적 프로세스들에만 액세스가능한 메모리의 영역을 말한다.

[0065] 별개의 도메인들 내의 데이터 라인들을 위해 별개의 키들을 사용하는 일부 실시예들에서, 데이터 라인들은 메인 메모리 내의 공통의 공유된 컨텐츠 디렉토리(예를 들어, 공통 해시 테이블) 내에 저장된다. 뚜렷한 컨텐츠의 2개의 라인들이 2개의 상이한 키들 하에서 동일한 암호화된 데이터 컨텐츠로 암호화하는 것이 가능하다. 그러므로, 상이한 보호 도메인들은 원칙적으로 컨텐츠 디렉토리 내의 동일한 엔트리를 결국 공유하게 될 수 있다. 예를 들어, 데이터 도메인 1에 대해, 데이터 컨텐츠 "hello"의 암호화 결과는 "xyz123"이고; 데이터 도메인 2에 대해, 데이터 컨텐츠 "world"의 암호화 결과는 또한 "xyz123"이다. "xyz123"의 단지 하나의 인스턴스 만이 메인 메모리 내에 저장된다.

[0066] 일부 실시예들에서, 공통 데이터 암호화 키가 사용되어, 데이터가 상이한 보호 도메인들 사이에서도 다시 복호화 및 재암호화하지 않고 카피되게 한다. 상이한 도메인들 내의 영향받은 변환 라인들만이 변환 라인들이 상이한 도메인들 내의 상이한 키들을 사용하여 보호됨에 따라 복호화 및 재암호화될 필요가 있다. 그러므로, 요구되는 복호화 및 재암호화 처리의 양이 감소된다.

[0067] 일부 실시예들에서, 소정의 도메인들이 공유되고 공유된 키를 사용하여 암호화된다. 예를 들어, 2개의 별개의 프로세스들은 둘 다 공통 도메인을 위한 공통 공유된 키를 제공함으로써 공유된 일부 보안성의 암호화된 데이터에 액세스할 수 있고, 둘 다의 프로세스들은 동일한 데이터를 복호화할 수 있다. 한편, 개인으로 유지된 데이터는 공유되지 않은 개인 키를 사용하여 암호화된다.

[0068] 일부 실시예들에서, 암호화된 컨텐츠들은 별개의 키들을 사용하여 암호화되지만, 컨텐츠 디렉토리와 관련된 메타데이터는 이 컨텐츠 디렉토리를 공유하는 보호 도메인들에 걸쳐 공유된 공통 키를 사용하여 암호화된다. 메타데이터를 위한 공통 키를 공유함으로써, 각각의 보호 도메인은 버킷, 참조 계수 등과 관련된 서명 라인과 같은, 중복된 엔트리와 관련된 메타데이터를 액세스하고 업데이트할 수 있다.

[0069] 공유된 메타데이터 키가 소정의 보호 도메인들 사이에서 허용되지 않으면, 실시예는 다음에 설명되는 바와 같이, 별개의 중복 제거 도메인들을 지원할 수 있다.

[0070] **중복 제거 도메인들**

[0071] 일부 실시예들에서, 프로세서의 어드레스 공간은 다수의 중복 제거 도메인들로 나누어진다. 여기에 사용되는 바와 같이, 중복 제거 도메인은 데이터 라인들이 중복 제거되고 유일한 컨텐츠들을 갖는 메모리의 영역을 말한다. 프로세서는 하나의 중복 제거 도메인 내의 주어진 데이터 컨텐츠의 기존의 카피를 찾는다.

[0072] 일부 경우들에서, 메모리 내에 주어진 데이터 블록의 하나보다 많은 인스턴스가 있을 수 있다. 그러나, 중복 제거 도메인들에 걸치는 주어진 데이터 블록의 빈도는 여전히 전체적인 응용 내의 주어진 데이터 블록의 빈도와

독립이므로, 빈도-기반 분석 및 공격을 방지한다. 또한, 별개의 중복 제거 도메인들이 별개의 키들을 사용할 수 있으므로, 각각의 도메인 내의 동일한 컨텐츠들은 동일한 값으로서 저장되지 않는다. 바꾸어 말하면, 주어진 암호문(예를 들어, 데이터 라인의 암호화된 컨텐츠들)의 빈도는 어느 특정한 평문의 발생의 빈도에 의존하지 않는다.

[0073] 일부 실시예들에서, 각각의 중복 제거 도메인은 중복 제거 도메인에서 사용된 데이터 라인들을 암호화하기 위해 별개의 비밀 키를 사용하므로, 중복 제거 도메인 내의 응용(또는 중복 제거 도메인 내의 메모리에 액세스할 수 있는 공격자)이 별개의 중복 제거 도메인 내의 데이터를 복호화할 수 없다. 일부 실시예들에서, 시스템은 각각의 별개의 보호 도메인에 대해 하나씩인, 다수의 중복 제거 도메인들로 구성된다.

[0074] 도 2a는 다수의 도메인들의 실시예를 도시한 데이터 구조도이다. 도시한 바와 같이, 별개의 비밀 키들은 데이터 라인들을 암호화하기 위해 상이한 도메인들을 위해 사용된다. 키들은 상이한 값들을 갖고 사전 특정된 레지스터들 내에 저장될 수 있다. 일부 실시예들에서, 키들은 상이한 형식들을 갖는다. 예를 들어, 128비트 키는 하나의 도메인을 위해 사용될 수 있고, 256비트 키는 상이한 도메인을 위해 사용될 수 있다. 도메인을 구성하는 것은 구현에 의존하고 상이한 실시예들에서 변화할 수 있다. 예를 들어, 별개의 사용자는 별개의 도메인에 대응할 수 있고, 별개의 키는 각각의 대응하는 도메인 내의 데이터 라인들을 암호화/복호화하는 데 사용될 수 있다. (별개의 프로세스들을 위한 별개의 도메인들과 같은) 보호 도메인들의 다른 할당들이 가능하다. 이 예에서, 데이터 라인들은 각각의 도메인에서 중복 제거되지만 반드시 도메인들에 거칠 필요는 없으므로; 별개의 도메인들 내에 중복된 데이터 라인들이 있을 수 있다. 메모리 제어기가 데이터를 암호화 또는 복호화할 필요가 있을 때, 현재의 도메인과 관련된 키는 액세스되고 암호화 또는 복호화를 위해 사용된다. 별개의 키들을 사용하여 별개의 중복 제거 도메인들을 암호화하는 것은 중복 제거 도메인들 간의 보호를 개선시키고 상이한 도메인들 내의 동일한 데이터 컨텐츠가 상이하게 암호화되는 것을 보장하므로, 빈도 분석 공격들에 대한 추가의 보호를 제공한다.

[0075] 도 2b의 실시예에서, 각각의 중복 제거 도메인은 변환 라인들을 위해 사용된 데이터를 암호화하기 위해 별개의 변환 라인 키를, 그리고, 컨텐츠 디렉토리에 관련된 메타데이터를 암호화하기 위해 별개의 메타데이터 키를 선택적으로 사용한다. 키들은 상이한 값들을 갖고 사전 특정된 레지스터들 내에 저장될 수 있다.

[0076] **타이밍 공격들을 방지**

[0077] 일부 실시예들에서, 라이트백 동작들의 타이밍은 기입 동작을 수행하는 데 사용된 표시된 시간량을 변화시키도록 인공적으로 수정되므로, 컨텐츠가 이미 메모리 내에 저장될 때, 컨텐츠가 이미 존재하지 않고 새로운 메모리 라인들이 할당될 필요가 있을 때, 또는 메모리 라인이 오버플로우 영역에 기입될 때, 중복 제거된 메모리에 기입하기 위한 표시된 시간량 사이의 분간할 수 있는 차이가 없다.

[0078] 한 방식은 이들 동작 중 어느 것에 의해 요구되는 최대 시간까지 각각의 기입의 완료 표시를 지연하는 것이다. 이 방식은 대부분의 라이트백들이 "포스트되고" (즉, 응용 실행에 대해 비동기적으로 일어나고) 추가의 지연은 전형적으로 메모리 라이트백 처리량을 감소시키지 않기 때문에 응용 수행에 상당한 부정적인 영향을 주지 않아야 한다.

[0079] **메모리 무결성**

[0080] 일부 보안 시스템들을 위한 추가의 요건은 시스템이 실행하는 동안 암호화된 데이터를 수정할 수 있는 것으로 가정된 공격자에 대해 데이터의 무결성을 보장하는 것이다.

[0081] 이 응용에서, 중복 제거를 지원하기 위해서, 컨텐츠 디렉토리는 위에 설명된 바와 같이, (컨텐츠 룩업의 일부로서) 컨텐츠에 의해 데이터의 로케이션을 결정하는 데 도움을 주는 메타데이터를 명시적으로 또는 암시적으로 포함한다.

[0082] 위에 설명된 바와 같이, 일부 실시예들에서, 메타데이터는 특정한 해시 버킷 내의 데이터 라인의 로케이션(즉, 해시 버킷 인덱스)이 계산되게 한다.

[0083] 일부 실시예들에서, 메모리 무결성 체크는 데이터 라인에의 판독 액세스 시에, 데이터 컨텐츠가 데이터 라인과 관련된 메타데이터와 일치하는 것을 보장하기 위해 데이터 라인의 컨텐츠들을 체크하는 것을 포함한다. PLID (또는 등가적으로, 액세스될 어드레스) 및 데이터 컨텐츠가 판독 액세스 시에 이미 이용가능하게 되어, 어떤 추가의 메모리 액세스 없이 무결성 체크가 수행되게 한다. 특히, 데이터 라인의 해시 인덱스는 재계산되어 데이터 라인이 위치하는 해시 버킷과 비교된다. 데이터 라인 컨텐츠들에 기초하여 계산되고 버킷 내의 엔트리를 결

정한 서명은 유사하세 재계산되고 엔트리에 대해 체크된다. 계산된 해시 인덱스가 해시 버킷 내의 데이터 라인 로케이션과 일치하지 않으면, 또는 계산된 서명이 해시 버킷 내의 엔트리의 서명과 일치하지 않으면, 그것은 메모리 무결성이 손상되었다는 것을 표시한다. 그러므로, 공격자는 수정된 데이터 컨텐츠가 관련된 메타데이터와 계속 일치하도록 데이터 컨텐츠를 수정하는 것이 가능하다면 검출 없이(바꾸어 말하면, 데이터 무결성을 침해하지 않고) 데이터를 단지 수정할 수 있다.

[0084] 또한, 일부 실시예들에서, 컨텐츠 룩업 또는 논리적 기입 시에, 데이터 라인 엔트리 내의 컨텐츠가 실제적으로 라이트백될 컨텐츠와 일치하는 것을 체크하기 위한 요건이 이미 있어서, 이 경우에 이미 무결성 체크가 있게 된다.

[0085] 또한, 일부 실시예들에서, 컨텐츠 룩업 시에, 그것의 메타데이터에 대한 데이터 라인 엔트리의 데이터 무결성이 또한 데이터 컨텐츠에 기초하여 메타데이터를 계산하고 그 결과들을 저장된 메타데이터와 비교함으로써 체크된다.

[0086] 일부 실시예들에서, 해시 인덱스 메타데이터 및 서명 메타데이터는 앞서 설명된 바와 같이, 보안 키 해시 함수(secure keyed hash function)에 의해 생성된다. 그러므로, 이러한 실시예에서, 암호화된 데이터 라인을 수정하는 공격자는 수정된 암호화된 데이터가 메타데이터 키를 알지 않고 이 메타데이터와 일치하는지를 결정할 수 없다. 대신에 공격자가 수정된 데이터 라인에 대한 참조를 위조할 수 있고 다음에 이 라인에 액세스하려고 시도한다면, 시스템은 메타데이터를 데이터 컨텐츠에 기초하여 계산하고 메타데이터와 데이터 컨텐츠가 일치하지 않는 것을 검출함으로써 무결성 침해를 검출할 것이다. 잠재적인 보안 침해가 보고된다. 해시 테이블 내의 보안 해시들의 사용으로 인해, 현재의 버킷 로케이션과 일치하는 값을 추측함으로써 라인에 대한 허가되지 않은 수정을 하려고 시도하는 침입자는 올바른 값을 추측하지 못할 가능성이 크므로, 보안 침해가 있다는 것을 표시하는 (로그 또는 메시지와 같은) 경보를 트리거한다.

[0087] 위에 논의된 바와 같이, 데이터 무결성 체크는 추가의 공간 비용 없이 제공된다.

[0088] **오버플로우 핸들링**

[0089] 일부 실시예들에서, 스토리지는 오버플로우 영역을 갖는다. 이러한 실시예들에서, 라이트백 시에, 컨텐츠가 맵핑되는 컨텐츠 디렉토리 해시 버킷 내에 자유로운 엔트리들이 없으면, 메모리 라인은 오버플로우 영역에 기입된다.

[0090] 일부 실시예들에서, 오버플로우 영역은 라인들을 중복 제거하지 않는다. 그러므로 동일한 데이터의 2개 이상의 인스턴스들이 중복 제거 도메인에 관련된 오버플로우 영역 내에 저장되는 것이 가능하다. 오버플로우 라인들을 안전하게 처리하기 위해서, 별개의 Ⅳ와 같은 여분의 메타데이터가 오버플로우 데이터 라인 마다 저장되고 데이터 라인의 데이터 컨텐츠를 암호화하는 데 사용된다. 상이한 Ⅳ들을 사용하여 암호화된 동일한 데이터 컨텐츠가 상이한 암호화된 결과들을 초래하기 때문에, 라인-레벨 암호화가 결정론적이 아닌 것으로 보장된다. 즉, 오버플로우 영역 내에 저장된 동일한 데이터의 2개의 인스턴스들은 동일한 암호문으로 암호화하지 않는다.

[0091] 또한, 일부 경우들에서, 다음의 시나리오가 발생할 수 있다: 먼저, 데이터 라인의 제1 라이트백 동안에 컨텐츠 디렉토리 내에 자유로운 엔트리가 없으므로, 데이터 라인은 오버플로우 영역 내에 저장되고; 나중에, 가비지 수집이 발생하고 컨텐츠 디렉토리 내의 하나 이상의 엔트리가 확보되고; 후속하여, 동일한 데이터 컨텐츠를 갖는 데이터 라인의 제2 라이트백이 발생하고, 데이터 라인은 정규 중복 제거된 컨텐츠 디렉토리 내에 저장된다. 이들 2개의 인스턴스들은 여전히 암호화를 위해 2개의 Ⅳ들을 효과적으로 사용하기 때문에 동일한 데이터의 이들 2개의 인스턴스들은 동일한 암호문으로 암호화하지 않고, 여기서 컨텐츠 디렉토리의 해시 버킷 내의 데이터와 관련된 Ⅳ는 고정되고 또는 널(null)이고, 오버플로우 영역 내의 엔트리의 Ⅳ는 랜덤하지만 해시 버킷과 관련된 Ⅳ과 상이하다.

[0092] 일부 실시예들에서, 프로세서/메모리 제어기는 요구된 어드레스가 어디에 상주하는지를 체크함으로써 판독 액세스가 컨텐츠 디렉토리의 오버플로우 영역에 대한 것인지 또는 컨텐츠 디렉토리의 비오버플로우 영역에 대한 것인지를 검출할 수 있다. 이것은 다음에 Ⅳ-기반 복호화를 적용할 수 있다. 유사하게, 컨텐츠 디렉토리를 구현하는 메모리 제어기는 컨텐츠에 대한 해시 버킷이 언제 가득 차게 되는지를 필수적으로 알게 되므로; 그것은 오버플로우 엔트리에 대응하는 Ⅳ를 사용하여 데이터 라인을 재암호화할 수 있다.

[0093] 다수의 중복 제거 도메인들을 갖는 일부 실시예들에서, 상이한 중복 제거 도메인들 사이의 오버플로우 영역들은 상이한 키들을 사용한다. 그러므로, 상이한 도메인 내의 동일한 컨텐츠들은 상이한 암호화 결과들을 산출한다.

[0094] 실제로, 오버플로우 영역은 오버플로우 메모리의 작은 퍼센티지라서 오버플로우 라인들에 딱 제한되는, IV 메타데이터를 위한 여분의 오버헤드가 공간 또는 액세스할 시간에 있어서 크지 않다.

[0095] 중복 제거는 중복 제거된 메모리의 사용 시의 조사들에 의해 보여진 바와 같이, 대부분의 응용들을 위해 요구되는 물리적 메모리 및 전력의 양을 감소시킨다. 일부 실시예들에서, 중복 제거는 또한 보안 메타데이터가 변환 라인들 및 오버플로우 라인들을 위해서만 요구되기 때문에 IV들과 같은 보안 메타데이터의 공간 오버헤드를 감소시킨다.

[0096] **하이브리드 메모리**

[0097] 일부 실시예들에서, 스토리지는 상이한 레이턴시들을 갖는 메모리들의 조합을 사용하여 구현된 하이브리드 메모리를 포함한다. 아래의 실시예들에서, 비휘발성 랜덤 액세스 메모리(NVRAM) 및 다이내믹 랜덤 액세스 메모리(DRAM)를 포함하는 하이브리드 메인 메모리가 예의 목적들을 위해 논의되고; 다른 유형들의 메모리들이 다른 실시예들에서 사용될 수 있다. NVRAM은 DRAM보다 높은 레이턴시를 갖고, 비용이 더 낮고, 더 적은 전력을 소비한다. 결과적인 하이브리드 메인 메모리 시스템은 DRAM 내의 자주 액세스되는 데이터를 캐싱함으로써 합당한 액세스 횟수들을 제공하면서 자본 지출에 있어서 더 낮은 비용 및 전력 소비로, DRAM 단독으로 가능한 것보다 상당히 더 큰 용량을 제공한다.

[0098] 하이브리드 메인 메모리를 사용하는 일부 실시예들에서, 데이터 라인들은 높은 레이턴시 메모리 및 낮은 레이턴시 메모리(예를 들어, NVRAM 및 DRAM) 둘 다에서 동일한 메모리 라인 크기 및 키를 사용하여, 위에 설명된 바와 같이 결정론적으로 암호화되고 중복 제거된다. 그러므로, 데이터 라인들은 데이터에 대한 암호화/복호화 조작 없이 DRAM과 NVRAM 사이에 전달될 수 있다. 예를 들어, DMA 엔진은 암호화 키들에 액세스할 필요 없이 데이터 라인들을 전달할 수 있다. DRAM에서와 같이, NVRAM 내의 중복 제거를 사용하는 것은 그렇지 않은 경우 중복들에 의해 소비되는 공간뿐만 아니라 그렇지 않은 경우 IV 스토리지에 의해 소비되는 여분의 공간을 절약한다.

[0099] 일부 실시예들에서, 낮은 레이턴시 메모리 및 높은 레이턴시 메모리 내의 데이터 라인들은 캐시 및 메인 메모리 내의 데이터 라인들과 유사한 방식으로 관리된다. 예를 들어, DRAM은 캐시와 유사하게 처리될 수 있고 NVRAM은 메인 메모리와 유사하게 처리될 수 있다. 데이터 라인들은 먼저 DRAM에 기입되고 다음에 NVRAM에 적절하게 라이트백된다. DRAM 내의 데이터 라인은 수정되는 대로 플래그될 수 있고, 중복 제거되지도 않고 NVRAM으로 맵핑되지도 않는다. DRAM 내의 수정된 데이터 라인이 축출될 때, 프로세스 500과 유사한 프로세스가 수행된다. 특히, 수정된 데이터 라인의 데이터 컨텐츠는 NVRAM 내의 컨텐츠 디렉토리에서 룩업된다. 데이터 컨텐츠가 NVRAM 내의 컨텐츠 디렉토리 내에 존재하면 기존의 라인이 위치하고; 그렇지 않으면, 새로운 데이터 라인이 컨텐츠 디렉토리의 적절한 버킷 내에 할당된다. 또한, 수정된 데이터 라인을 참조하는 변환 라인들이 위치하고, 수정된 데이터 라인에 대한 참조들이 NVRAM 내의 기존의 라인에 포인트하도록 변경된다.

[0100] 일부 실시예들에서, 하이브리드 메모리 시스템 내의 낮은 레이턴시 메모리로부터의 변환 라인 축출 및 데이터 라인 축출은 캐시로부터의 변환 라인 축출 및 데이터 라인 축출과 유사하게 처리된다. 변환 라인들이 DRAM으로부터 축출되기 전에, 변환 라인에 의해 참조된 각각의 수정된 데이터 라인은 데이터 라인 컨텐츠들에 대한 NVRAM 내의 PLID(NVRAM PLID라고 함)를 결정하기 위해 NVRAM에서 중복 제거되고, 변환 라인은 이 새로운 NVRAM PLID를 참조하기 위해 업데이트된다. 참조하는 변환 라인이 없는 어떤 데이터 라인은 축출될 때 폐기된다. 이러한 실시예들에서, NVRAM 중복 제거는 가능한 한 늦게까지 연기될 수 있고, 라인은 축출되기 전에 폐기되기 때문에 중복 제거가 일부 경우들에서 피해짐으로써, 중복 제거 오버헤드를 감소시킨다. 상이한 키들이 NVRAM 및 DRAM을 위해 사용된 실시예에서, NVRAM으로의 축출 시에, 수정된 라인이 먼저 판독되고 캐시 내로 복호화되고, NVRAM 키로 재암호화되고, 다음에 NVRAM 내로 중복 제거된다.

[0101] 일부 실시예들에서, NVRAM에 상주하는 변환 라인으로의 제1 데이터 액세스 시에, 요구된 라인을 복귀시킨 후에, 메모리 제어기는 관련된 변환 라인에 의해 참조된 다른 데이터 라인들을 프리페치하여 DRAM으로 전달하여, 그들이 생기는 경우 후속하는 참조들에 대한 레이턴시를 감소시킨다. 이 최적화는 NVRAM과 DRAM 사이의 전달의 더 큰 단위를 사용하는 이점들의 일부를 제공하는 것으로 기대된다.

[0102] 일부 실시예들에서, 낮은 레이턴시 메모리에 대한 암호화의 단위는 높은 레이턴시 메모리에 대한 암호화의 단위들과 상이하다. 여기에 사용되는 바와 같이, 암호화의 단위는 단일 암호화 결과를 생성하기 위해 암호화되는 데이터의 양을 말한다. 예를 들어, DRAM 상의 각각의 메모리 라인은 단위로서 암호화되고 NVRAM 상의 다수의 메모리 라인들은 단위로서 함께 암호화된다(예를 들어, 64개 라인들은 함께 암호화됨). 데이터가 NVRAM으로부터 DRAM으로 전달될 때, NVRAM으로부터의 다수의 메모리 라인들을 포함하는 단위는 복호화되고, 복호화된 라인

들은 개별적으로 암호화될 DRAM으로 전달된다. 일부 실시예들에서, 암호화/복호화 메커니즘들은 프로세서 칩 상에서 구현된다. 데이터 전달 단위(예를 들어, 64개 라인들)는 NVRAM으로부터 데이터 전달 단위를 복호화하고 캐시 내에 메모리 라인을 저장하는 프로세서로 전달된다. 후속하여, 메모리 라인들은 개별적으로 재암호화되고 DRAM에 라이트백 된다. 일부 구현들에서, NVRAM은 NVRAM이 DRAM과 상이한 데이터 단위 크기를 이용하더라도 중복 제거 및 결정론적 암호화로 관리된다. 일부 다른 구현들에서, DRAM 부분만이 중복 제거 및 결정론적 암호화를 구현하고, NVRAM은 종래의 확률적 암호화를 구현한다.

[0103] NVRAM이 DRAM보다 큰 단위에서 데이터를 암호화하고 전달하는 실시예들에서, NVRAM은 효과적으로 프로세서와 관련한 I/O 디바이스이다. 즉, 프로세서는 실제로 이 NVRAM에 직접 판독 또는 기입하지 않는다. 대신에, DRAM 및 NVRAM은 각각 캐시 및 메인 메모리와 유사한 방식으로 사용된다. 구체적으로, 보다 큰 데이터 블록들이 NVRAM으로부터 DRAM으로 판독되고 다음에 프로세서에 이용가능하게 된다. 유사한 메커니즘이 라이트백을 위해 구현된다. 효과적으로, 시스템은 DRAM과 NVRAM 간의 페이징의 형태를 구현한다.

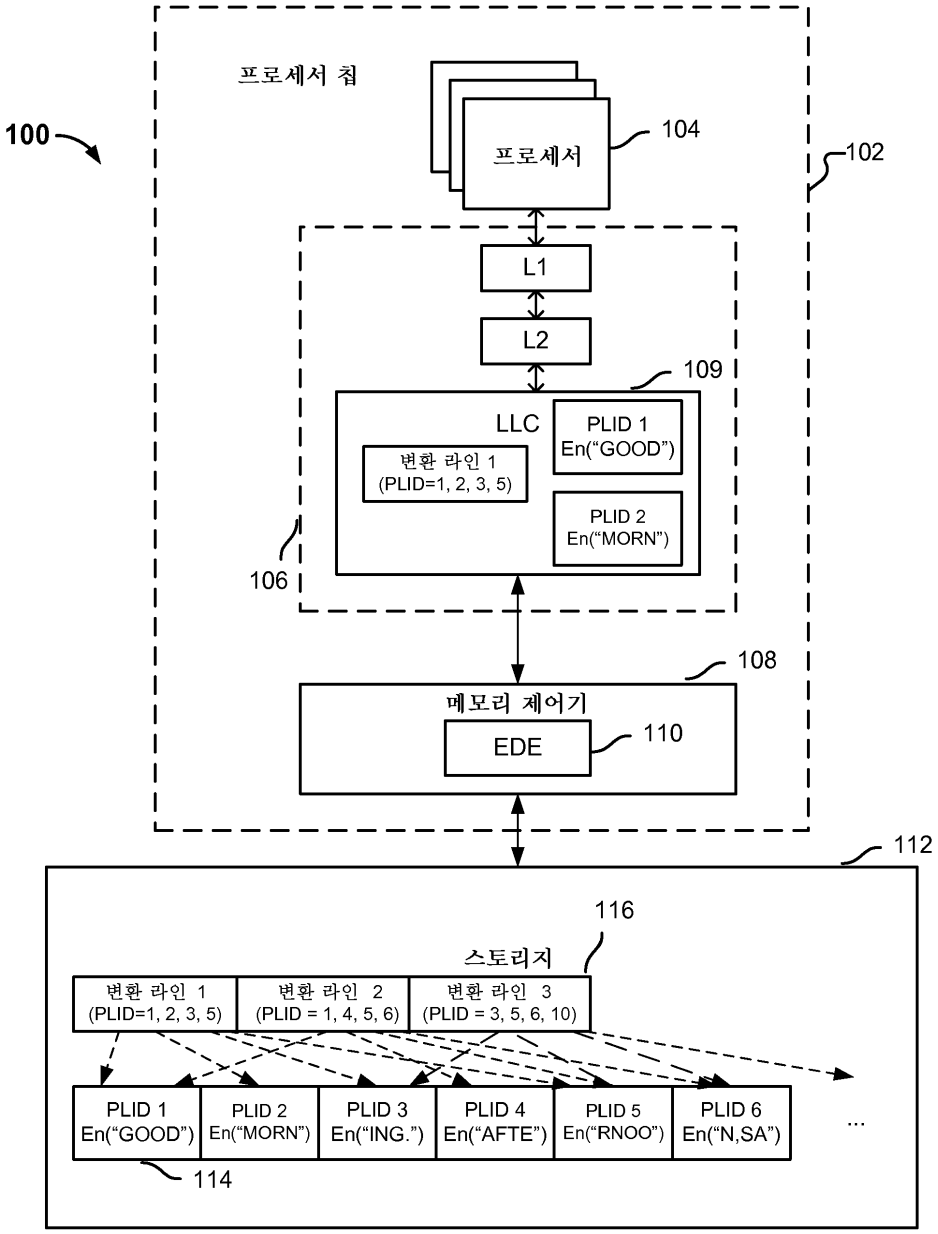
[0104] 중복 제거 기반 메모리 암호화가 설명되었다. 위에 설명된 기술들은 보조 스토리지 요소뿐만 아니라 메인 메모리에 적용가능하다. 여기에 사용되는 바와 같이, 보조 스토리지 요소는 자기 또는 광 디스크와 같은 소자, 메모리보다는 스토리지 요소로서 운영 체제 및 응용 소프트웨어에 의해 처리된 NVRAM-기반 드라이브 등과 같은 고상 드라이브를 말한다. 운영 체제 및 응용 소프트웨어는 전형적으로 메모리 액세스 호출들보다는 I/O 기능 호출들을 사용하여 보조 스토리지 시스템으로부터 판독 및 그에 기입한다. 보조 스토리지 시스템들의 일부 실시예들에서, 변환 라인들 및 메타데이터는 별개의 데이터의 액세스 당 비용이 DRAM보다 일반적으로 상당히 높기 때문에 캐시된다. 예를 들어, 데이터 라인들 상의 참조 계수들은 DRAM 내에 저장될 수 있고, 각각의 데이터 라인에 대한 참조들의 수를 계수하기 위해 보조 스토리지(예를 들어, NVRAM)를 스캔함으로써 리부트 시에 재계산됨으로써, 보조 스토리지 내에 이들 값을 저장하거나 업데이트할 필요성을 없애 준다.

[0105] 중복 제거된 스토리지의 암호화 및 복호화를 사용하는 컴퓨터 보안이 개시되었다. 위에 논의된 기술은 효율적인 낮은 레이턴시 암호화 및 복호화가 수행되게 하고, 높은 데이터 보안성 및 스케일가능성을 보장하고, 대부분들의 응용들에 요구되는 리소스들(예를 들어, 물리적 메모리, 전력, 공간 오버헤드)의 양을 감소시키고, 추가의 공간 비용이 없이 데이터 보안을 제공한다.

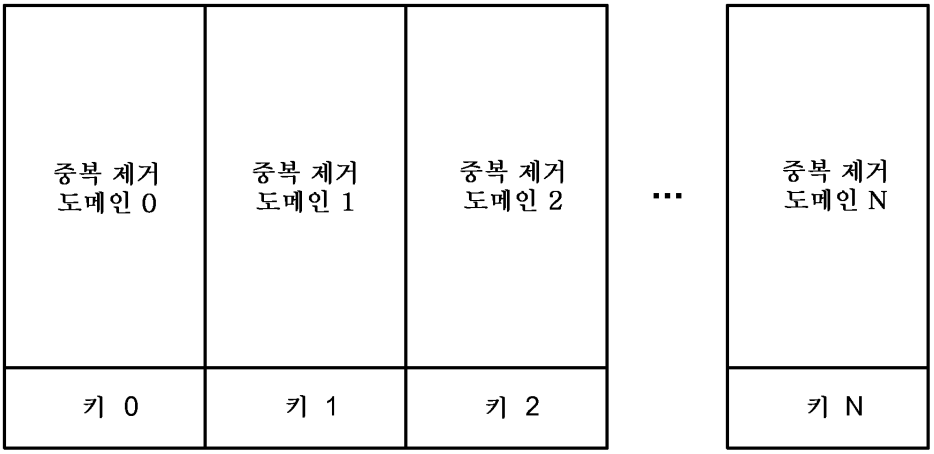
[0106] 상기 실시예들이 이해를 명확히 하는 목적들을 위해 어느 정도 상세히 설명되었지만, 본 발명은 제공된 상세들로 제한되지 않는다. 본 발명을 구현하는 많은 대안적 방식들이 있다. 개시된 실시예들은 예시적이고 제한적이지 않다.

도면

도면1


100
프로세서 칩
프로세서
104
102
L1
L2
109
LLC
PLID 1
En("GOOD")
변환 라인 1
(PLID=1, 2, 3, 5)
PLID 2
En("MORN")
106
108
메모리 제어기
EDE
110
112
116
스토리지
변환 라인 1
(PLID=1, 2, 3, 5)
변환 라인 2
(PLID = 1, 4, 5, 6)
변환 라인 3
(PLID = 3, 5, 6, 10)
PLID 1
En("GOOD")
PLID 2
En("MORN")
PLID 3
En("ING.")
PLID 4
En("AFTE")
PLID 5
En("RNOO")
PLID 6
En("N,SA")
...
114

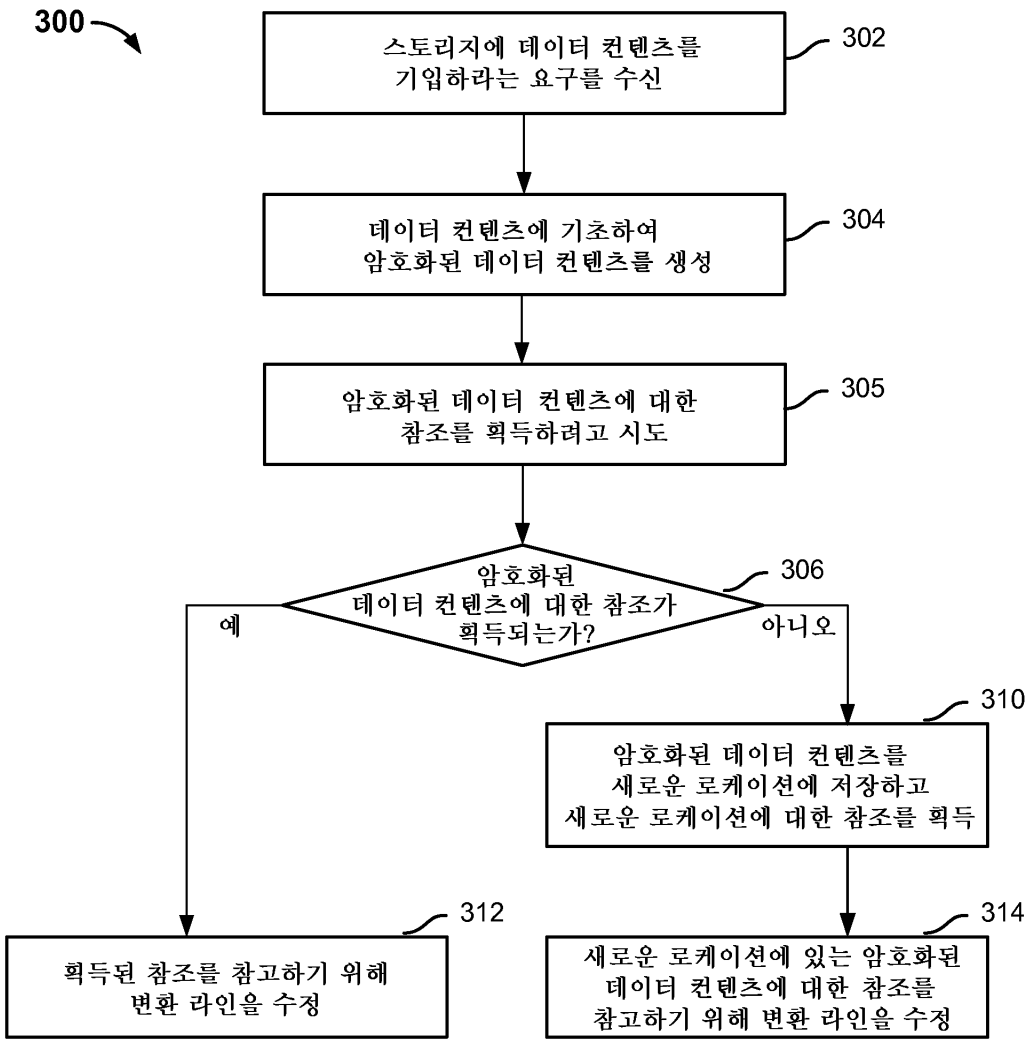
도면2a


중복 제거
도메인 0
중복 제거
도메인 1
중복 제거
도메인 2
...
중복 제거
도메인 N
키 0
키 1
키 2
키 N

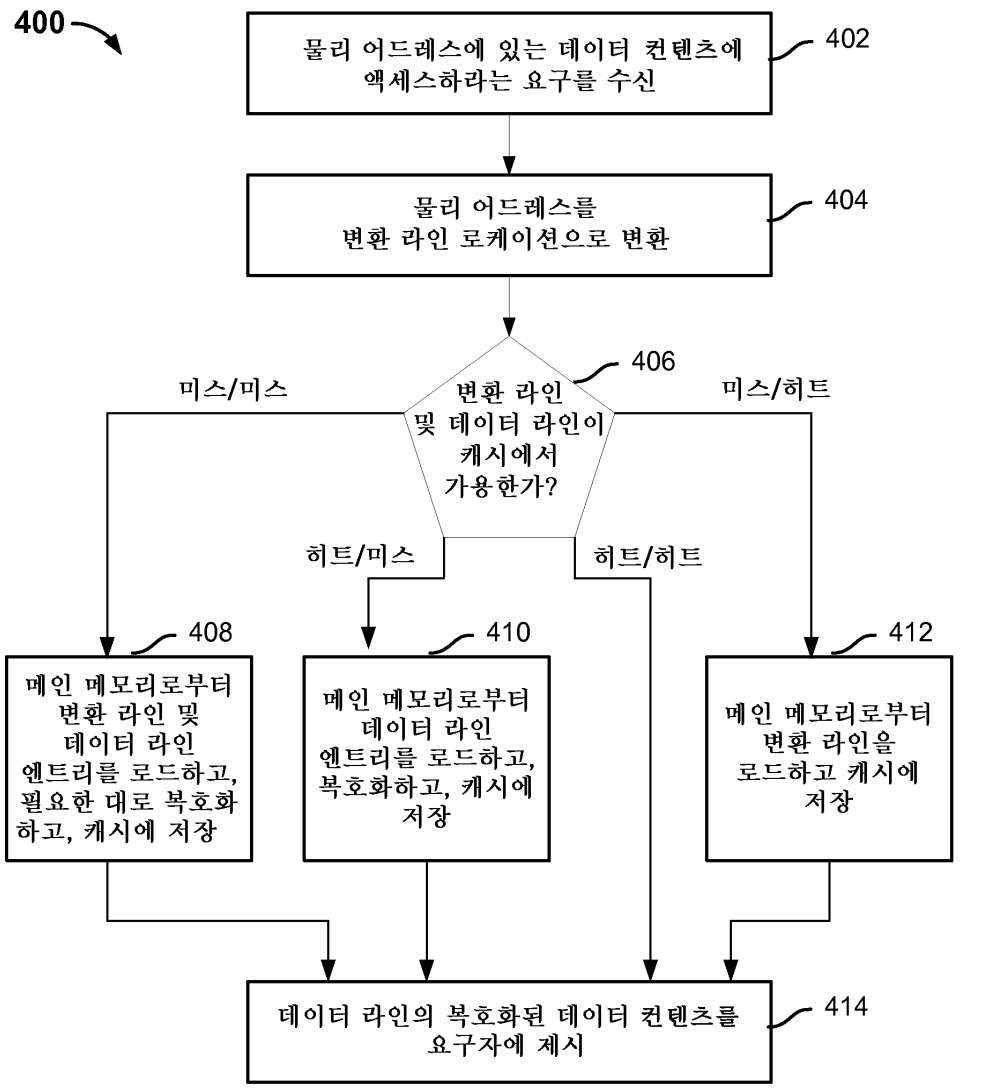
도면2b


중복 제거
도메인 0
중복 제거
도메인 1
중복 제거
도메인 2
...
중복 제거
도메인 N
데이터 라인
키 0
데이터 라인
키 1
데이터 라인
키 2
데이터 라인
키 N
변환 라인
키 0
변환 라인
키 1
변환 라인
키 2
변환 라인
키 N
메타데이터
키 0
메타데이터
키 1
메타데이터
키 2
메타데이터
키 N

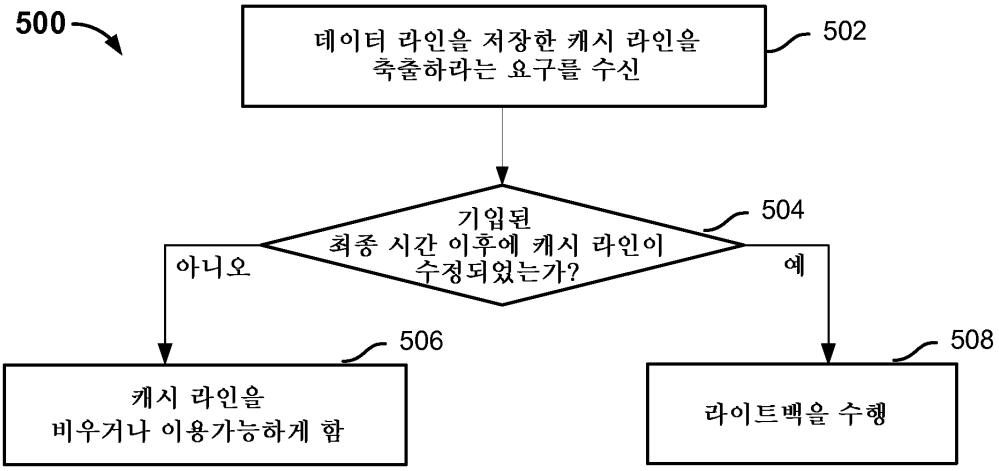
도면3


300
스토리지에 데이터 컨텐츠를 기입하라는 요구를 수신
302
데이터 컨텐츠에 기초하여 암호화된 데이터 컨텐츠를 생성
304
암호화된 데이터 컨텐츠에 대한 참조를 획득하려고 시도
305
암호화된 데이터 컨텐츠에 대한 참조가 획득되는가?
306
예
아니오
310
암호화된 데이터 컨텐츠를 새로운 로케이션에 저장하고 새로운 로케이션에 대한 참조를 획득
312
획득된 참조를 참고하기 위해 변환 라인을 수정
314
새로운 로케이션에 있는 암호화된 데이터 컨텐츠에 대한 참조를 참고하기 위해 변환 라인을 수정

도면4


400
물리 어드레스에 있는 데이터 컨텐츠에 액세스하라는 요구를 수신
402
물리 어드레스를 변환 라인 로케이션으로 변환
404
변환 라인 및 데이터 라인이 캐시에서 가용한가?
406
미스/미스
미스/히트
히트/미스
히트/히트
메인 메모리로부터 변환 라인 및 데이터 라인 엔트리를 로드하고, 필요한 대로 복호화하고, 캐시에 저장
408
메인 메모리로부터 데이터 라인 엔트리를 로드하고, 복호화하고, 캐시에 저장
410
메인 메모리로부터 변환 라인을 로드하고 캐시에 저장
412
데이터 라인의 복호화된 데이터 컨텐츠를 요구자에 제시
414

도면5


500
데이터 라인을 저장한 캐시 라인을 축출하라는 요구를 수신
502
기입된 최종 시간 이후에 캐시 라인이 수정되었는가?
504
아니오
예
캐시 라인을 비우거나 이용가능하게 함
506
라이트백을 수행
508

도면6

변환 라인	IV	PLID	PLID	...	PLID	암호화된 내용들
100	259	3	40	...	99	AZ95MKP0
...	...	...	...	...	...	...
502	802	3	40	...	99	MPLLR092
503	243	98	2	...	71	REW10832
...	...	...	...	...	...	...

도면7a

인덱스	서명	참조 계수	내용들
...	...	...	...
$(A_i-B_i)/S$	Hash(“hello”)	2	“hello”
$(A_{i+1}-B_i)/S$	Hash(“world”)	4	“world”
...	...	...	...

도면7b

인덱스	서명	참조 계수	내용들
...	...	...	...
$SecHash_1(En(“hello”))$	$SecHash_2(En(“hello”))$	En(2)	En(“hello”)
...	...	...	...