(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0094052
(43) 공개일자 2022년07월05일

(51) 국제특허분류(Int. Cl.)
H04L 9/14 (2006.01) *G16Y 30/10* (2020.01)
H04L 65/40 (2022.01) *H04L 9/00* (2022.01)
H04L 9/06 (2006.01) *H04L 9/40* (2022.01)
(52) CPC특허분류
H04L 9/14 (2013.01)
G16Y 30/10 (2020.01)
(21) 출원번호 10-2020-0185327
(22) 출원일자 2020년12월28일
심사청구일자 2020년12월28일

(71) 출원인
연세대학교 산학협력단
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
(72) 발명자
김한준
서울특별시 서대문구 연세로 50, 연세대학교 (신촌동)
김봉준
서울특별시 서대문구 연세로 50, 연세대학교 (신촌동)
(뒷면에 계속)
(74) 대리인
특허법인(유한)아이시스

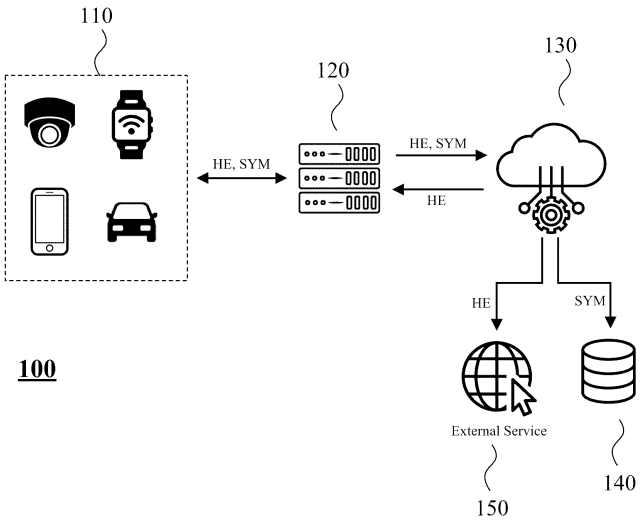
전체 청구항 수 : 총 14 항

(54) 발명의 명칭 **적응적 암호화를 이용한 IoT 서비스 방법 및 IoT 장치**

(57) 요 약

적응적 암호화를 이용한 IoT 서비스 방법은 IoT 장치가 수집 데이터를 이용한 서비스 요청을 수신하는 단계, 상기 IoT 장치가 상기 수집 데이터를 처리하는 함수 코드를 분석하여 데이터의 유형에 따라 동형 암호화 방식으로 암호화되는 수집 데이터를 처리할 수 있도록 상기 함수 코드를 변환하는 단계, 상기 IoT 장치가 상기 변환한 함수 코드를 클라우드에 전송하는 단계 및 상기 IoT 장치가 암호화된 수집 데이터 및 상기 변환한 함수 코드를 이용하여 처리된 결과를 상기 클라우드로부터 수신하는 단계를 포함한다.

대 표 도 - 도2



(52) CPC특허분류
H04L 63/0281 (2013.01)
H04L 63/045 (2013.01)
H04L 67/12 (2022.05)
H04L 9/008 (2013.01)
H04L 9/0631 (2013.01)
H04L 2209/046 (2013.01)

(72) 발명자
허선영
서울특별시 서대문구 연세로 50, 연세대학교 (신촌동)
이재호
서울특별시 서대문구 연세로 50, 연세대학교 (신촌동)
정신녕
서울특별시 서대문구 연세로 50, 연세대학교 (신촌동)
이용우
서울특별시 서대문구 연세로 50, 연세대학교 (신촌동)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711103119
과제번호	2018-0-01392-003
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보보호핵심원천기술개발사업
연구과제명	내재적 기능 기반의 랜섬웨어 공격 피해 복원
기 여 율	1/1
과제수행기관명	포항공과대학교 산학협력단
연구기간	2020.01.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

IoT 장치가 수집 데이터를 이용한 서비스 요청을 수신하는 단계;

상기 IoT 장치가 상기 수집 데이터를 처리하는 함수 코드를 분석하여 데이터의 유형에 따라 동형 암호화 방식으로 암호화되는 수집 데이터를 처리할 수 있도록 상기 함수 코드를 변환하는 단계;

상기 IoT 장치가 상기 변환한 함수 코드를 클라우드에 전송하는 단계; 및

상기 IoT 장치가 암호화된 수집 데이터 및 상기 변환한 함수 코드를 이용하여 처리된 결과를 상기 클라우드로부터 수신하는 단계를 포함하는 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 2

제1항에 있어서,

상기 함수 코드를 변환하는 단계는

상기 IoT 장치가 상기 함수 코드에서 함수 호출 관계에 따른 종속성을 기준으로 적어도 하나의 그래프를 생성하는 단계;

상기 IoT 장치가 상기 적어도 하나의 그래프별로 상기 수집 데이터에 해당하는 데이터 소스, 데이터 연산 항목 및 상기 수집 데이터의 연산 결과인 데이터 싱크를 결정하는 단계;

상기 IoT 장치가 상기 적어도 하나의 그래프 중 데이터 연산 항목이 있는 타깃 그래프에서 상기 데이터 소스 및 상기 데이터 싱크가 모두 존재하는 경우, 상기 타깃 그래프에 포함되는 데이터를 동형 암호화 대상으로 결정하는 단계; 및

상기 IoT 장치가 상기 함수 코드에서 동형 암호화 대상으로 결정된 항목들이 처리될 수 있도록 코드를 수정하는 단계를 포함하는 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 3

제1항에 있어서,

상기 IoT 장치는

상기 함수 코드를 변환하는 단계에서

상기 함수 코드에서 사용되는 상기 수집 데이터 중 연산이 필요한 제1 데이터인 경우 동형 암호화 방식으로 처리될 데이터로 매핑하고,

상기 함수 코드에서 사용되는 상기 수집 데이터 중 연산이 필요 없는 제2 데이터인 경우 대칭키 암호화 방식을 처리할 데이터로 매핑하는 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 4

제3항에 있어서,

상기 IoT 장치는 상기 제1 데이터를 동형 암호화 방식으로 암호화하여 상기 암호화된 수집 데이터를 생성하고, 상기 제2 데이터를 대칭키 암호화 방식으로 암호화하여 상기 클라우드에 전송하는 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 5

제3항에 있어서,

상기 IoT 장치는 상기 제1 데이터에 대한 매핑 정보 및 상기 제2 데이터에 대한 매핑 정보를 프록시 서버에 전

달하고.

상기 프록시 서버가 상기 제1 데이터를 동형 암호화 방식으로 데이터를 암호화하고, 상기 제2 데이터를 대칭키 암호화 방식으로 데이터를 암호화하여 상기 클라우드에 전송하는 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 6

제1항에 있어서,

상기 IoT 장치는 상기 처리된 결과가 동형 암호화 방식으로 암호화된 데이터의 연산 결과인 경우, 상기 처리된 결과를 동형 암호화 방식에 대응되게 복호하는 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 7

제1항에 있어서,

상기 IoT 장치는 네트워크에 연결된 다른 IoT 장치로부터 암호화를 위한 공유키를 수신하고, 상기 다른 IoT 장치는 상기 처리된 결과에 따라 동작이 제어되는 장치인 적응적 암호화를 이용한 IoT 서비스 방법.

청구항 8

수집 데이터를 생성하는 센싱장치;

원본 함수 코드를 분석하여 이용하는 데이터 중 동형 암호화 방식으로 암호화되는 데이터를 처리할 수 있도록 상기 원본 함수 코드를 변환하는 프로그램을 저장하는 저장장치;

상기 프로그램을 이용하여 상기 수집 데이터를 처리하는 함수 코드를 변환하는 연산장치; 및

상기 변환된 함수 코드를 클라우드에 전송하고, 암호화된 수집 데이터 및 상기 변환한 함수 코드를 이용하여 처리된 결과를 상기 클라우드로부터 수신하는 통신장치를 포함하는 적응적 암호화를 이용한 IoT 장치.

청구항 9

제8항에 있어서,

상기 연산장치는

상기 함수 코드에서 함수 호출 관계에 따른 종속성을 기준으로 생성되는 적어도 하나의 그래프별 중 데이터 연산 항목이 있는 타깃 그래프에서 데이터 소스 및 데이터 싱크가 모두 존재하는 경우, 상기 타깃 그래프에 포함되는 데이터를 동형 암호화 대상으로 결정하고, 상기 함수 코드에서 동형 암호화 대상으로 결정된 항목들이 처리될 수 있도록 코드를 수정하여 상기 함수 코드를 변환하고,

상기 데이터 소스는 상기 수집 데이터 중 적어도 하나의 데이터이고, 상기 데이터 싱크는 상기 처리된 결과 중 적어도 하나인 적응적 암호화를 이용한 IoT 장치.

청구항 10

제8항에 있어서,

상기 연산장치는

상기 함수 코드에서 사용되는 상기 수집 데이터 중 연산이 필요한 제1 데이터인 경우 동형 암호화 방식으로 처리될 데이터로 매핑하고,

상기 함수 코드에서 사용되는 상기 수집 데이터 중 연산이 필요 없는 제2 데이터인 경우 대칭키 암호화 방식을 처리할 데이터로 매핑하는 적응적 암호화를 이용한 IoT 장치.

청구항 11

제10항에 있어서,

상기 연산장치는

상기 제1 데이터를 동형 암호화 방식으로 암호화하여 상기 암호화된 수집 데이터를 생성하고, 상기 제2 데이터

를 대칭키 암호화 방식으로 암호화하는 적응적 암호화를 이용한 IoT 장치.

청구항 12

제8항에 있어서,

상기 연산장치는

상기 처리된 결과가 동형 암호화 방식으로 암호화된 데이터의 연산 결과인 경우, 상기 처리된 결과를 동형 암호화 방식에 대응되게 복호하는 적응적 암호화를 이용한 IoT 장치.

청구항 13

제8항에 있어서,

상기 통신장치는

다른 IoT 장치로부터 암호화를 위한 공유키를 수신하고, 상기 다른 IoT 장치는 상기 처리된 결과에 따라 동작이 제어되는 장치인 적응적 암호화를 이용한 IoT 장치.

청구항 14

제8항에 있어서,

상기 함수 코드에서 사용되는 상기 수집 데이터 중 연산이 필요한 제1 데이터인 경우 동형 암호화 방식으로 처리될 데이터로 매핑되고,

상기 함수 코드에서 사용되는 상기 수집 데이터 중 연산이 필요 없는 제2 데이터인 경우 대칭키 암호화 방식을 처리할 데이터로 매핑되고,

상기 통신장치는 제1 데이터에 대한 매핑 정보 및 상기 제2 데이터에 대한 매핑 정보를 프록시 서버 또는 다른 IoT 장치에 전송하는 적응적 암호화를 이용한 IoT 장치.

발명의 설명

기 술 분 야

[0001] 이하 설명하는 기술은 동형 암호화 기반한 IoT 서비스의 보안 기법에 관한 것이다.

배 경 기 술

[0002] FaaS(Function-as-a-Service)는 전용 서버 없이 IoT(Internet of Things) 서비스를 구축하는데 유용하다. 사용자는 IoT 장치를 통해 자신의 개인정보를 클라우드에 전송할 수 있다. 예컨대, 사용자는 얼굴 영상을 클라우드에 전달할 수 있다. 개인 정보 보호를 위하여 서비스 사업자는 개인 정보를 암호화하여 관리할 수 있다.

[0003] 한편, 암호화 기법 중 동형 암호화(homomorphic encryption)는 개인 정보를 보호하면서도 클라우드에서 암호화된 정보를 기준으로 데이터 처리를 가능하게 한다.

선행기술문헌

비특허문헌

[0004] (비특허문헌 0001) On the use of Homomorphic Encryption to Secure Applications, Services, and Routing Protocols, European Journal of Scientific Research ISSN 1450-216X Vol. 88 No 3 October, 2012, pp.416-438

발명의 내용

해결하려는 과제

[0005] 동형 암호화는 데이터 처리의 연산 및 통신에 많은 부하가 걸린다. 따라서, 동형 암호화를 사용하는 IoT 서비스는 시스템에 많은 자원을 요구하는 문제점이 있다.

[0006] 이하 설명하는 기술은 적응적으로 동형 암호화 및 대칭키 암호화(symmetric-key encryption)를 사용하는 IoT 서비스를 제공하고자 한다.

과제의 해결 수단

[0007] 적응적 암호화를 이용한 IoT 서비스 방법은 IoT 장치가 수집 데이터를 이용한 서비스 요청을 수신하는 단계, 상기 IoT 장치가 상기 수집 데이터를 처리하는 함수 코드를 분석하여 데이터의 유형에 따라 동형 암호화 방식으로 암호화되는 수집 데이터를 처리할 수 있도록 상기 함수 코드를 변환하는 단계, 상기 IoT 장치가 상기 변환한 함수 코드를 클라우드에 전송하는 단계 및 상기 IoT 장치가 암호화된 수집 데이터 및 상기 변환한 함수 코드를 이용하여 처리된 결과를 상기 클라우드로부터 수신하는 단계를 포함한다.

[0008] 적응적 암호화를 이용한 IoT 장치는 수집 데이터를 생성하는 센싱장치, 원본 함수 코드를 분석하여 이용하는 데이터 중 동형 암호화 방식으로 암호화되는 데이터를 처리할 수 있도록 상기 원본 함수 코드를 변환하는 프로그램을 저장하는 저장장치, 상기 프로그램을 이용하여 상기 수집 데이터를 처리하는 함수 코드를 변환하는 연산장치 및 상기 변환된 함수 코드를 클라우드에 전송하고, 암호화된 수집 데이터 및 상기 변환한 함수 코드를 이용하여 처리된 결과를 상기 클라우드로부터 수신하는 통신장치를 포함한다.

발명의 효과

[0009] 이하 설명하는 기술은 적응적으로 동형 암호화와 대칭키 암호화를 이용하여 개인 정보를 보호하면서도 클라우드에서 암호화된 데이터를 이용한 서비스 제공을 가능하게 한다. 대칭키 암호화는 동형 암호화에 비하여 복잡도가 낮아 빠른 처리가 가능하다. 따라서, 이하 설명하는 기술은 동형 암호화만을 사용하는 방식에 비하여 시스템의 부하가 낮다.

도면의 간단한 설명

[0010] 도 1은 대칭키 암호화 및 동형 암호화의 성능을 비교한 결과이다.

도 2는 적응적 암호화 기반 IoT 서비스 시스템의 예이다.

도 3은 컴파일러 동작에 대한 예이다.

도 4는 런타임 동작에 대한 예이다.

도 5는 IoT 장치의 구성에 대한 예이다.

발명을 실시하기 위한 구체적인 내용

[0011] 이하 설명하는 기술은 다양한 변경을 가할 수 있고 여러 가지 실시례를 가질 수 있는 바, 특정 실시례들을 도면에 예시하고 상세하게 설명하고자 한다. 그러나, 이는 이하 설명하는 기술을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 이하 설명하는 기술의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0012] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 해당 구성요소들은 상기 용어들에 의해 한정되지는 않으며, 단지 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 이하 설명하는 기술의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

[0013] 본 명세서에서 사용되는 용어에서 단수의 표현은 문맥상 명백하게 다르게 해석되지 않는 한 복수의 표현을 포함하는 것으로 이해되어야 하고, "포함한다" 등의 용어는 설명된 특징, 개수, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함을 의미하는 것이지, 하나 또는 그 이상의 다른 특징들이나 개수, 단계 동작 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 배제하지 않는 것으로 이해되어야 한다.

[0014] 도면에 대한 상세한 설명을 하기에 앞서, 본 명세서에서의 구성부들에 대한 구분은 각 구성부가 담당하는 주기능 별로 구분한 것에 불과함을 명확히 하고자 한다. 즉, 이하에서 설명할 2개 이상의 구성부가 하나의 구성부로 합쳐지거나 또는 하나의 구성부가 보다 세분화된 기능별로 2개 이상으로 분화되어 구비될 수도 있다. 그리고 이하에서 설명할 구성부 각각은 자신이 담당하는 주기능 이외에도 다른 구성부가 담당하는 기능 중 일부 또는 전부의 기능을 추가적으로 수행할 수도 있으며, 구성부 각각이 담당하는 주기능 중 일부 기능이 다른 구성부에 의해 전담되어 수행될 수도 있음은 물론이다.

[0015] 또, 방법 또는 동작 방법을 수행함에 있어서, 상기 방법을 이루는 각 과정들은 문맥상 명백하게 특정 순서를 기재하지 않은 이상 명기된 순서와 다르게 일어날 수 있다. 즉, 각 과정들은 명기된 순서와 동일하게 일어날 수도 있고 실질적으로 동시에 수행될 수도 있으며 반대의 순서대로 수행될 수도 있다.

[0016] 이하 설명에서 사용하는 용어 및 기술에 대하여 간략하게 설명한다.

[0017] 동형 암호화는 데이터를 암호화된 상태에서 연산할 수 있는 암호화 방법이다. 동형 암호화를 위한 다양한 구현 기술이 있다. 암호문에 대한 연산의 종류를 기준으로 동형 암호화는 AHE(additive homomorphic encryption, LHE(leveled homomorphic encryption) 및 FHE(fully homomorphic encryption)로 구분할 수 있다. 연구자는 FHE 중 BFV 스킴을 사용하여 시뮬레이션을 수행하였다. BFV는 RLWE(Ring Learning with Errors) 기반한 방식이다. 구체적인 암호화 과정에 대한 설명은 생략한다. 이하 설명하는 동형 암호화는 다양한 기법 중 어느 하나를 이용할 수 있다. 동형 암호화는 평문을 암호문으로 변환한다. 데이터 처리 장치가 암호문을 이용하여 연산한 결과는 새로운 암호문이 된다. 이후 새로운 암호문을 복호하여 얻은 평문은 암호화하기 전 원래 데이터의 연산 결과와 같다.

[0018] 대칭키 암호화는 암호화와 복호화에 같은 암호 키를 쓰는 암호화 방법이다. 대표적인 대칭키 암호화는 AES(Advanced Encryption Standard)가 있다. 구체적인 암호화 과정에 대한 설명은 생략한다.

[0019] 이하 설명하는 기술은 동형 암호화와 대칭키 암호화를 사용하여 정보를 보호한다. 전술한 바와 같이 동형 암호화는 연산 복잡도 및 데이터양이 많은 암호화 방식이며, 대칭키는 상대적으로 부하가 적은 암호화 방식이다. 도 1은 대칭키 암호화 및 동형 암호화의 성능을 비교한 결과이다. 도 1은 대칭키 암호화 방식 중 AES와 동형 암호화 방식 중 BFV를 비교한 예이다. 도 1은 두 개의 암호화 알고리즘을 Raspberry Pi 3B+에서 실행한 결과이다. 도 1(A)는 평문을 암호화한 암호문의 크기에 대한 결과이다. 도 1(B)는 평문의 암호화하는 시간에 대한 결과이다. 도1(C)는 암호문을 복호하는 시간에 대한 결과이다. 도 1을 살펴보면, 동형 암호화가 대칭키보다 데이터양도 많고, 데이터 처리 시간도 더 많이 걸리는 것을 알 수 있다. 평균적으로 대칭키 암호화가 동형 암호화에 비하여 약 4배 정도 암호문의 크기가 작았고, 암호화 시간은 약 58배 적었고, 복호 시간은 약 11배 적었다.

[0020] FaaS는 클라우드 컴퓨팅 서비스에서 주목받는 서비스이다. FaaS는 별도의 서버 구축이나 관리 없이 개발자가 개발한 코드를 수행하게 한다. FaaS를 이용하면 개발자는 특정 명령이나 함수를 설계하고 업로드하여 바로 실행할 수 있다. 따라서, FaaS는 다양한 IoT 장치가 생성하는 이벤트를 처리하는 서비스 제공을 편리하게 한다. FaaS를 지원하는 IoT 플랫폼은 AWS Lambda, Azure Functions, Google Cloud Functions, IBM Cloud Functions 등이 있다. 이하 설명하는 IoT 서비스도 FaaS에 기반한 서비스라고 전제한다.

[0021] 전술한 바와 같이 IoT 서비스는 다양한 개인 정보를 요구할 수 있다. 따라서, 암호화를 통한 정보 보호가 필요하다. FaaS 기반의 IoT 플랫폼에서 동형 암호화는 유효한 기법이다. 이는 동형 암호화가 개인 정보를 보호하면서도, 제3자가 제공하는 클라우드에서 암호문을 통한 데이터 처리를 가능하게 하기 때문이다.

[0022] FaaS에서 데이터 처리 함수(processing function)가 특정 사용자 데이터에 대한 연산 없이 클라우드에 저장하기만 한다면, 해당 데이터는 대칭키 암호화로 암호화하여도 충분하다. 이하 설명하는 기술은 IoT 장치가 수집 또는 생성한 데이터 중 클라우드에서 연산이 필요한 데이터만을 동형 암호화로 처리하고, 연산 대상이 아닌 데이터는 대칭키 암호화로 처리하고자 한다. 예컨대, 클라우드의 처리 함수가 안면 인식용 프로그램이라면, IoT 장치가 생성하는 데이터 중 얼굴 영상은 동형 암호화로 암호화하고, 연산처리에 사용하지 않는 비디오 클립은 대칭키로 암호화할 수 있다. 이와 같이 이하 설명하는 기술은 클라우드 서비스에서의 데이터 처리 여부에 따라 동형 암호화 및 대칭키 암호화를 적응적으로 사용한다.

[0024] 도 2는 적응적 암호화 기반 IoT 서비스 시스템(100)의 예이다. IoT 서비스 시스템(100)은 IoT 장치(110), 프록시 서버(120) 및 클라우드 장치(130)를 포함한다.

[0025] IoT 장치(110)는 다양한 장치들 중 적어도 하나일 수 있다. IoT 장치(110)는 IoT 서비스를 위한 사용자 정보를 수집 내지 생성하는 장치이다. 도 1은 예시적으로 스마트기기, 스마트 워치, 자동차, 감시 카메라와 같은 장치를 도시하였다.

[0026] IoT 장치(110)는 IoT 서비스를 위한 프로그램 내지 애플리케이션이 설치될 수 있다. IoT 장치(110)는 자신이 생성한 데이터 중 클라우드에서 연산 처리가 필요한지 여부에 따라 다른 암호화 방식으로 암호화할 수 있다. 즉, IoT 장치(110)는 자신이 생성한 데이터 중 클라우드에서 연산 처리되는 데이터는 동형 암호화 방식(HE)으로 암호화한다. 또한, IoT 장치(110)는 자신이 생성한 데이터 중 클라우드에서 연산 처리되지 않는 데이터는 대칭키 암호화 방식(SYM)으로 암호화한다.

[0027] IoT 장치(110)는 암호화된 데이터를 유선 또는 무선 네트워크를 통해 클라우드 측에 전달한다. 이때 전달되는 암호화된 데이터는 암호화 방식에 따라 두 가지 유형으로 구분된다. 두 가지 유형의 데이터는 다음과 같다. 제1 암호화 데이터는 동형 암호화 기반 데이터(HE)이고, 제2 암호화 데이터는 대칭키 암호화 기반 데이터(SYM)이다.

[0028] 통상적으로 클라우드 시스템은 방화벽을 갖는다. 따라서, 클라우드 시스템에서 프록시 서버(120)가 IoT 장치(110)로부터 암호화된 데이터를 수신할 수 있다.

[0029] 클라우드 장치 내지 클라우드 시스템(130)은 프록시 서버(120)를 경유하여 IoT 장치(110)가 전달한 암호화된 데이터를 수신할 수 있다.

[0030] 클라우드 장치(130)는 수신하는 데이터를 이용하여 일정한 연산이나 처리를 한다. 클라우드 장치(130)는 연산 처리가 필요 없는 제2 암호화 데이터는 별도의 저장 장치(140)에 저장하다. 클라우드 장치(130)는 연산이 필요한 제1 암호화 데이터를 이용하여 일정한 연산을 할 수 있다. 경우에 따라서, 클라우드 장치(130)는 연산이 필요한 제1 암호화 데이터를 다른 서버(150)에 전달하여 암호화 데이터를 처리하게 할 수 있다.

[0031] 클라우드 장치(130)는 제1 암호화 데이터를 이용하여 연산한 결과를 프록시 서버(120)를 경유하여 IoT 장치(110)에 전달할 수 있다. 동형 암호화의 원리에 따라 연산한 결과도 일정하게 암호화된 데이터이다. IoT 장치(110)는 수신한 연산 결과를 복호한다. 복호한 결과는 IoT 장치(110)가 본래 생성한 데이터(평문)를 연산한 결과와 같다. IoT 장치(110)는 복호한 결과를 이용하여 일정한 서비스를 제공할 수도 있다.

[0032] 상기와 같은 데이터 처리를 위하여 IoT 장치(110)는 소프트웨어적인 구성을 사용할 수 있다. IoT 장치(110)는 컴파일러(compiler)와 런타임(runtime)을 이용하여 암호화 데이터를 처리할 수 있다. 컴파일러는 데이터를 처리하는 코드(code)를 동형 암호화 대상과 대칭키 대상의 데이터를 별도로 처리할 수 있는 코드로 변환한다. 런타임은 프로그램 실행을 위한 운영 환경을 의미하며, 컴파일러가 전달하는 정보를 이용하여 데이터 암호화 및 복호화를 한다. 이하 구체적으로 설명한다.

[0033] 한편, IoT 장치(110)가 연산 능력이나 가용 전력이 적은 장치인 경우, 프록시 서버(120)가 전술한 컴파일러와 런타임을 갖추어 데이터를 처리할 수 있다. 이 경우 IoT 장치(110)는 단순히 데이터를 생성하여 전달하는 기능을 하고, 프록시 서버(120)가 데이터 종류에 따라 암호화 방식을 선별하여 처리하는 역할을 하게 된다. 이하, 설명의 편의를 위하여 IoT 장치(110)가 컴파일러 및 런타임을 내장한 경우를 중심으로 설명한다.

[0035] 컴파일러는 함수 코드를 분석하여 함수가 이용하는 데이터 항목별로 암호화 방식을 결정한다. 컴파일러는 각 데이터 항목별로 매핑한 암호화 방식을 테이블 형태로 저장할 수 있다. 컴파일러는 해당 매핑 정보를 런타임에게 전달한다.

[0036] 도 3은 컴파일러 동작(200)에 대한 예이다. 컴파일러는 원본 함수 코드를 입력받고(210), 원본 함수 코드를 변환하는 과정(220~250)을 수행한다.

[0037] 컴파일러는 원본 함수 코드(original function code)를 입력받는다(210). 또한, 컴파일러는 원본 함수 코드 처리에 필요한 설정 파일(configuration file)을 입력받을 수도 있다(210).

[0038] 원본 함수 코드는 IoT 서비스 내용에 따라 사전에 제공된다. 도 3은 아래의 표 1의 원본 함수 코드를 예시한다.

표 1

[0039]

```
1 image = get('/camera/image')
2 video = get('/camera/video')
3 result = recognize(image)
4 DB.save(video)
5 pub('/recog/result', result)
```

[0040] 상기 원본 함수 코드는 얼굴 인식 함수이다. 원본 함수 코드는 IoT 장치인 카메라로부터 정지 이미지(image) 및 동영상(video)을 입력받는다(라인 1~2). 원본 함수 코드는 이미지를 이용하여 얼굴 인식하는 함수(recognize(image))로부터 결과를 입력받는다(라인 3). 원본 함수 코드는 동영상을 DB에 저장한다(라인 4). 그리고, 원본 함수 코드는 얼굴 인식 결과를 출력한다(라인 5).

[0041] 컴파일러는 함수 코드를 분석하여 데이터 종속성(data dependence)을 파악한다(220). 함수 코드의 제1 명령(instruction)이 제2 명령의 결과를 사용한다면, 제1 명령은 제2 명령에 종속되는 관계이다. 상기 원본 함수 코드를 기준으로 설명하면 데이터 종속성은 {1→3→5} 및 {2→4}이다. {1→3→5}는 이미지를 이용한 얼굴 인식 과정이고, {2→4}는 동영상을 저장하는 과정이다. 데이터 종속성은 방향성 그래프인 데이터 종속 그래프(data dependence graph)로 표현 가능하다. 데이터 종속 그래프는 코드에서 서로 연결관계를 갖는 함수 또는 데이터 처리 과정이 하나의 그래프를 구성한다.

[0042] 컴파일러는 데이터 종속 그래프를 기준으로 데이터 소스(data source), 데이터 싱크(data sink) 및 연산 동작 여부를 결정한다(230). 도 3을 살펴보면, 데이터 종속 그래프에서 1번 및 2번은 데이터 소스이고, 5번은 데이터 싱크이다. 데이터 종속 그래프에서 3번은 일정한 연산 동작이 필요한 항목이고, 4번은 연산이 필요없는 항목에 해당한다.

[0043] 컴파일러는 데이터 종속 그래프의 각 항목에 대하여 적절한 암호화 방식을 결정한다(240). 도 3은 이 과정을 컬러링(coloring)이라고 명명하였다. 컴파일러는 데이터 종속 그래프 중 연산 동작이 포함된 그래프에서 데이터 소스 및 데이터 싱크를 확인하고, 해당 데이터 종속 그래프에 포함된 항목(노드)들에 대하여 동형 암호화 방식(HE)을 설정한다. 도 3에서 컴파일러는 데이터 종속 그래프에서 연산이 필요한 항목인 3번이 포함된 그래프에서 데이터 소스 1 번 및 데이터 싱크 5번을 찾아 동형 암호화 방식을 설정한다. 그리고, 컴파일러는 나머지 데이터 종속 그래프의 항목을 모두 대칭키 암호화 대상(STM)으로 설정한다. 최종적으로 컴파일러는 도 3의 하단에 표시한 암호 테이블(encryption table)과 같이 데이터 항목별로 암호화 방식을 매칭한 정보를 생성할 수 있다.

[0044] 마지막으로 컴파일러는 240 과정에서 설정한 방식의 암호화 데이터를 처리 가능하도록 원본 함수 코드를 변환한다(250). 변환한 코드는 아래 표 2와 같다. 수정되거나 추가된 내용은 밑줄로 표시하였다.

표 2

[0046]

```
1a image = Ciphertext()
1 image = get( '/camera/image' )
2 video = get( '/camera/video' )
3 result = recognize^HE (image)
4 DB.save(video)
5 pub( '/recog/result' , result)
```

[0047] 컴파일러는 240 과정에서 동형 암호화 방식으로 설정한 항목을 처리할 수 있도록 코드를 수정한다. 컴파일러는 동형 암호화로 입력 이미지를 암호화하는 과정을 추가한다(라인 1a). 컴파일러는 동형 암호화된 입력 이미지를 이용하여 얼굴 인식을 처리하도록 해당 함수를 수정한다(라인 3). 또한, 컴파일러는 도 3의 하단에 표시한 복호 테이블(decryption table)과 같이 데이터 항목별로 복호 방식을 매칭한 정보를 생성할 수 있다.

[0049] 아래 표 3은 얼굴 인식 함수를 변환한 예를 나타낸다.

표 3

[0050]

원본 코드(original code)	변환된 코드(transformed code)
1 def recognize(im,w1,w2): 2 c=conv(im,w1) 3 a1=act(c) 4 p=pool(a1) 5 a2=act(p) 6 r=fc(a2,w2) 7 return r 8 9 # Square Activation 10 def act(inp): 11 out=[] 12 for i in ₩ 13 range(0,len(inp)): 14 r=inp[i]*inp[i] 15 out.append(r) 16 return out 17	1 def recognize^{HE}(im,w1,w2): 2 c=conv^{HE}(im,w1) 3 a1=act^{HE}(c) 4 p=pool^{HE}(a1) 5 a2=act^{HE}(p) 6 r=fc^{HE}(a2,w2) 7 return r 8 9 # Square Activation 10 def act^{HE}(inp): 11 out=[] 12 for i in ₩ 13 range(0,len(inp)): 14 r=inp[i]*inp[i] 15 r=relinearize(r,reKey) 16 out.append(r) 17 return out

[0051] 표 3의 좌측은 원본 코드이고, 우측은 변환된 코드를 예시한다. 컴파일러는 상기 얼굴 인식 함수 recognize()에서 동형 암호화된 데이터를 처리하도록 서브 함수를 변경한다. 변경된 함수는 HE로 표시하였다. 컴파일러는 동형 암호화된 데이터 처리를 위하여 라인 15에 새로운 함수 relinearize(r,reKey)를 추가한다. 해당 함수는 동형 암호화의 곱셈 연산을 위해 추가한 함수이다.

[0053] 런타임은 컴파일러로부터 매핑 정보를 수신한다. 매핑 정보는 전술한 암호 테이블 및 복호 테이블을 포함할 수 있다. 런타임은 매핑 정보에 따라 데이터 항목별로 암호화 및 복호화를 수행한다.

[0055] IoT 서비스는 복수의 장치가 관여할 수 있다. 예컨대, 현관문에 설치된 IP 카메라가 획득한 영상을 분석하고, 인증에 성공하면 해당 정보를 도어락에 전달하여 도어락을 제어할 수 있다. 도 4는 이와 같은 시나리오를 가정한 동작의 예이다. 도 4는 런타임 동작에 대한 예이다.

[0056] 도 4(A)는 데이터 암호화 및 복호화를 위한 정보를 전달하는 과정이다. λ는 원본 함수 코드이다. 컴파일러는 원본 함수 코드를 변환한 함수 λ', 암호 테이블 및 복호 테이블을 생성한다.

[0057] 배포 매니저는 IoT 장치 또는 프록시 서버에 내장된 객체일 수 있다. 배포 매니저(deploy manager)는 변환한 함수 λ'를 클라우드에 전달한다(①). 배포 매니저는 데이터 항목을 복호할 마스터 IoT 장치를 선택할 수 있다. 마스터 IoT 장치는 도어락이다. 마스터 IoT 장치는 암호화 및 복호화를 위한 키를 생성하고, 생성한 키를 배포 매니저에 전달한다(②, ③). 배포 매니저는 암호 테이블, 복호 테이블 및 암호/복호 위한 키를 필요한 다른 IoT 장치에 전달할 수 있다(④). 동형 암호화 및 대칭키 암호화를 위하여 서로 다른 키를 사용할 수 있다. 배포 매니저는 클라우드에도 필요한 공개키를 전달할 수 있다.

[0058] 도 4(B)는 데이터를 암호화하고 복호하는 과정의 예이다. 도 4(B)는 IoT 장치인 IP 카메라에 설치된 런타임 A와 프록시 서버에 설치된 런타임 B를 각각 도시한다.

[0059] 예컨대, 클라우드가 IP 카메라에 이미지 및 동영상 촬영을 요청하면, λ 함수가 트리거될 수 있다. 런타임 B가 λ 함수를 트리거할 수 있다(ⓐ).

[0060] 런타임 A는 암호 테이블을 기준으로 이미지 데이터(I_{plain})를 동형 암호화(Enc(I_{plain},HE))하고, 암호화한 데이터

(I_{cipher})를 출력한다(ⓑ). 또한, 런타임 A는 암호 테이블을 기준으로 동영상 데이터(V_{plain})를 대칭키 암호화(Enc(V_{plain},SYM))하고, 암호화한 데이터(V_{cipher})를 출력한다(ⓒ).

[0061] 클라우드는 암호화된 데이터 I_{cipher} 및 V_{cipher}를 수신한다. 클라우드는 수신한 I_{cipher}를 연산하여 산출된 얼굴 인식 결과 R_{cipher}를 프록시에 전달할 수 있다. 런타임 B는 복호 테이블을 이용하여 R_{cipher}를 복호한다(ⓓ). 복호한 결과는 도어락과 같은 IoT 장치에 전달할 수 있다. 도 4는 프록시 서버가 클라우드의 연산 결과를 복호하는 예를 도시한다. 나아가, 도 4와 달리 IoT 장치에 설치된 런타임이 클라우드의 연산 결과를 복호할 수도 있다.

[0063] 도 5는 IoT 장치(300)의 구성에 대한 예이다. IoT 장치(300)는 도 2의 IoT 장치(110)에 해당한다. IoT 장치(300)는 센서장치(310), 저장장치(320), 메모리(320), 통신장치(340), 연산장치(350) 및 출력장치(360)를 포함할 수 있다.

[0064] 센서장치(310)는 IoT 장치(300)의 주변 또는 사용자에 대한 정보를 센싱하는 장치이다. 예컨대, 센서장치(310)는 이미지 센서, 온도 센서, 위치 센서, 생체 정보 수집 센서 등을 포함할 수 있다.

[0065] 저장장치(320)는 센서장치가 생성한 데이터를 저장할 수 있다.

[0066] 저장장치(320)는 전술한 데이터 처리를 위한 프로그램을 저장할 수 있다. 예컨대, 저장장치(320)는 전술한 원본 함수 코드를 저장할 수 있다.

[0067] 저장장치(320)는 동형 암호화 및 대칭키 암호화를 수행하는 프로그램을 저장할 수 있다.

[0068] 저장장치(320)는 전술한 컴파일러 및/또는 런타임을 저장할 수 있다. 컴파일러 및 런타임은 소프트웨어 객체에 해당한다.

[0069] 저장 장치(320)는 평문을 암호화한 데이터, 암호문을 복호한 데이터 등을 저장할 수도 있다.

[0070] 저장 장치(320)는 원본 함수 코드를 변환한 함수 코드, 암호 테이블 및 복호 테이블을 저장할 수 있다.

[0071] 메모리(330)는 IoT 장치(300)가 데이터를 처리하는 과정에서 발생하는 임시 데이터를 저장할 수 있다.

[0072] 통신장치(340)는 네트워크를 통해 일정한 정보를 수신하고 전송하는 구성을 의미한다.

[0073] 통신장치(340)는 특정 기능 함수의 실행 요청(트리거)을 수신할 수 있다.

[0074] 통신장치(340)는 다른 IoT 장치로부터 암호화를 위한 공유키를 수신할 수 있다.

[0075] 통신장치(340)는 원본 함수 코드를 변환한 함수 코드, 암호 테이블 및 복호 테이블을 송신할 수 있다.

[0076] 통신장치(340)는 암호화한 데이터를 프록시 서버 또는 클라우드에 송신할 수 있다.

[0077] 통신장치(340)는 클라우드에서 동형 암호화된 데이터를 이용하여 연산한 결과를 수신할 수 있다.

[0078] 연산장치(330)는 컴파일러를 실행하여 원본 함수 코드를 암호화 데이터의 처리가 가능한 함수로 변환할 수 있다. 예컨대, 연산장치(330)는 컴파일러를 실행하여 원본 함수 코드를 분석하면서 데이터 종속성을 결정하여, 데이터 종속 그래프를 산출할 수 있다. 연산장치(330)는 데이터 종속 그래프를 기준으로 전술한 바와 같이 데이터 항목별로 암호화 방식을 결정할 수 있다. 연산장치(330)는 원본 함수 코드를 동형 암호화 데이터를 처리할 수 있는 코드로 변환할 수 있다.

[0079] 연산장치(330)는 데이터 항목별로 데이터 암호화 및 복호화에 필요한 매칭 정보를 생성할 수 있다. 연산장치(330)는 전술한 암호 테이블 및 복호 테이블을 생성할 수 있다.

[0080] 연산장치(330)는 런타임을 실행하여 암호 테이블 기준으로 데이터 항목별로 설정된 암호화 방식에 따라 데이터를 암호화할 수 있다. 연산장치(330)는 복호 테이블 기준으로 클라우드로부터 수신한 연산 결과(동형 암호문을 연산한 결과)를 복호할 수 있다.

[0081] 연산장치(330)는 암호화 및 복호화를 위한 키를 생성할 수도 있다.

[0082] 연산 장치(330)는 데이터를 처리하고, 일정한 연산을 처리하는 프로세서, AP, 프로그램이 임베디드된 칩과 같은 장치일 수 있다.

[0083] 출력 장치(360)는 일정한 정보를 출력할 수 있다. 출력 장치(360)는 IoT 서비스를 위한 인터페이스 화면 및 클라우드에서 제공하는 서비스 결과를 출력할 수 있다.

[0084] 또한, 상술한 바와 같은 적응적 암호화 방법, IoT 서비스 제공 방법 및 클라우드의 데이터 처리 방법은 컴퓨터에서 실행될 수 있는 실행가능한 알고리즘을 포함하는 프로그램(또는 어플리케이션)으로 구현될 수 있다. 상기 프로그램은 일시적 또는 비일시적 판독 가능 매체(non-transitory computer readable medium)에 저장되어 제공될 수 있다.

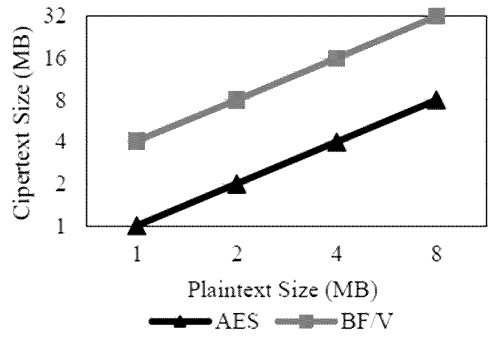
[0085] 비일시적 판독 가능 매체란 레지스터, 캐쉬, 메모리 등과 같이 짧은 순간 동안 데이터를 저장하는 매체가 아니라 반영구적으로 데이터를 저장하며, 기기에 의해 판독(reading)이 가능한 매체를 의미한다. 구체적으로는, 상술한 다양한 어플리케이션 또는 프로그램들은 CD, DVD, 하드 디스크, 블루레이 디스크, USB, 메모리카드, ROM(read-only memory), PROM (programmable read only memory), EPROM(Erasable PROM, EPROM) 또는 EEPROM(Electrically EPROM) 또는 플래시 메모리 등과 같은 비일시적 판독 가능 매체에 저장되어 제공될 수 있다.

[0086] 일시적 판독 가능 매체는 스태틱 램(Static RAM, SRAM), 다이내믹 램(Dynamic RAM, DRAM), 싱크로너스 디램(Synchronous DRAM, SDRAM), 2배속 SDRAM(Double Data Rate SDRAM, DDR SDRAM), 증강형 SDRAM(Enhanced SDRAM, ESDRAM), 동기화 DRAM(Synclink DRAM, SLDRAM) 및 직접 램버스 램(Direct Rambus RAM, DRRAM) 과 같은 다양한 RAM을 의미한다.

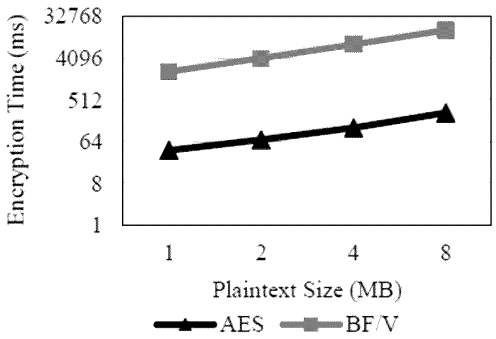
[0087] 본 실시례 및 본 명세서에 첨부된 도면은 전술한 기술에 포함되는 기술적 사상의 일부를 명확하게 나타내고 있는 것에 불과하며, 전술한 기술의 명세서 및 도면에 포함된 기술적 사상의 범위 내에서 당업자가 용이하게 유추할 수 있는 변형 예와 구체적인 실시례는 모두 전술한 기술의 권리범위에 포함되는 것이 자명하다고 할 것이다.

도면

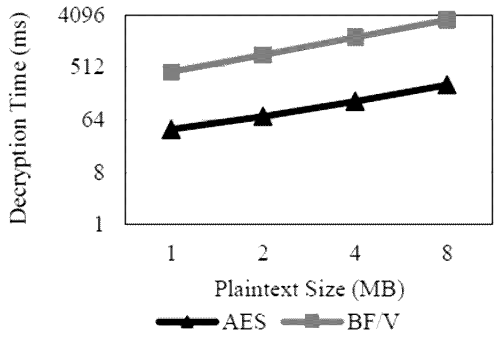
도면1


Cipertext Size (MB)
32
16
8
4
2
1
1
2
4
8
Plaintext Size (MB)
AES
BF/V

(A)

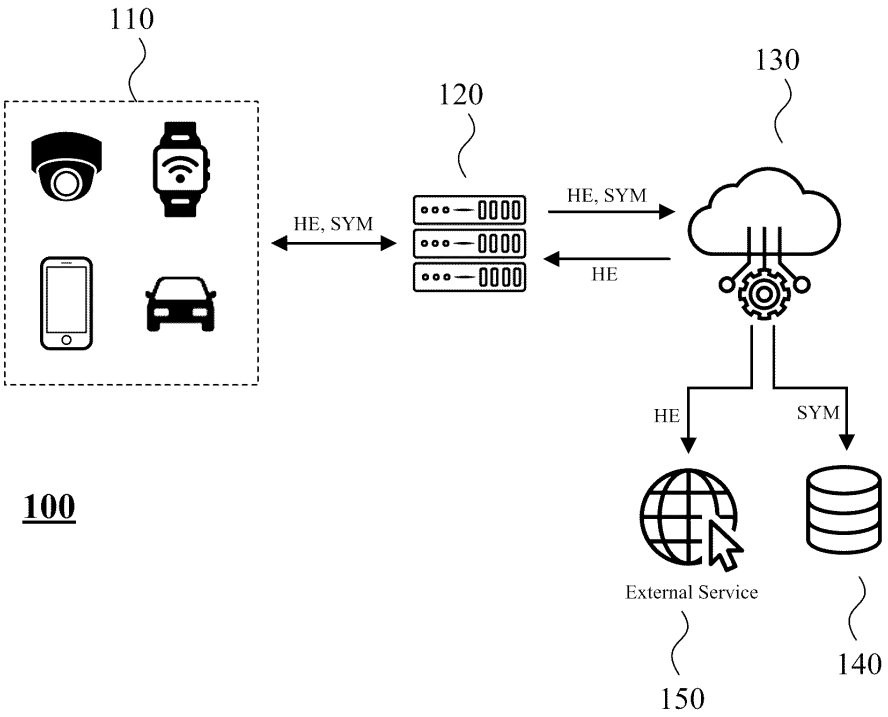

Encryption Time (ms)
32768
4096
512
64
8
1
1
2
4
8
Plaintext Size (MB)
AES
BF/V

(B)

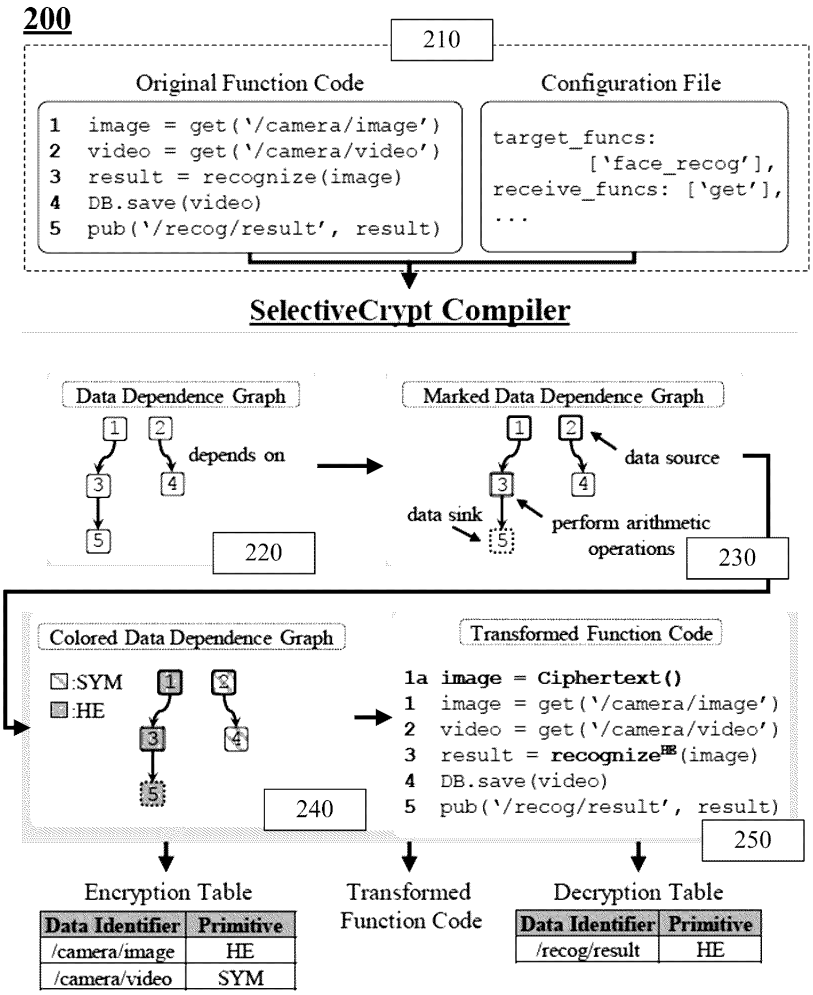

Decryption Time (ms)
4096
512
64
8
1
1
2
4
8
Plaintext Size (MB)
AES
BF/V

(C)

도면2


110
120
130
HE, SYM
HE, SYM
HE
HE
SYM
External Service
140
150
100

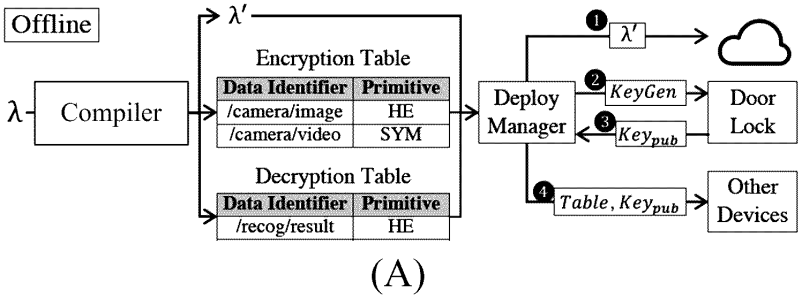
도면3



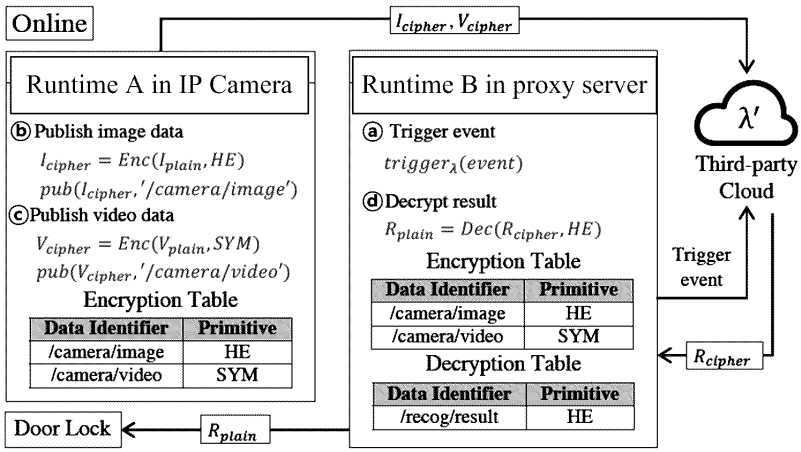
Data Identifier	Primitive
/camera/image	HE
/camera/video	SYM

Data Identifier	Primitive
/recog/result	HE

도면4


Offline
λ
Compiler
λ′
Encryption Table
Data Identifier
Primitive
/camera/image
HE
/camera/video
SYM
Decryption Table
Data Identifier
Primitive
/recog/result
HE
Deploy Manager
1 λ′
2 KeyGen
3 Key_{pub}
4 $Table, Key_{pub}$
Door Lock
Other Devices

(A)


Online
I_{cipher}, V_{cipher}
Runtime A in IP Camera
ⓑ Publish image data
$I_{cipher} = Enc(I_{plain}, HE)$
$pub(I_{cipher}, '/camera/image')$
ⓒ Publish video data
$V_{cipher} = Enc(V_{plain}, SYM)$
$pub(V_{cipher}, '/camera/video')$
Encryption Table
Data Identifier
Primitive
/camera/image
HE
/camera/video
SYM
Runtime B in proxy server
ⓐ Trigger event
$trigger_{\lambda}(event)$
ⓓ Decrypt result
$R_{plain} = Dec(R_{cipher}, HE)$
Encryption Table
Data Identifier
Primitive
/camera/image
HE
/camera/video
SYM
Decryption Table
Data Identifier
Primitive
/recog/result
HE
λ′
Third-party Cloud
Trigger event
R_{cipher}
Door Lock
R_{plain}

(B)

도면5