이 일 구 교수 지도
석사학위 청구논문

사물인터넷을 위한 화이트박스 암호 최적화 기법

2023

성신여자대학교 대학원
미래융합기술공학과
이 진 민

사물인터넷을 위한 화이트박스 암호 최적화 기법

이 일 구 교수 지도

이 논문을 석사학위논문으로 제출함

2022년 11월

성신여자대학교 대학원

미래융합기술공학과

이 진 민

인 준 서

이진민의 석사학위 논문으로 인준함

2022년 11월

심사위원장 김 경 진 (서명 또는 인)

심 사 위 원 이 일 구 (서명 또는 인)

심 사 위 원 임 연 섭 (서명 또는 인)

성신여자대학교 대학원

논 문 개 요

사물인터넷(IoT, Internet of Things) 기술이 생활에 밀접한 영역에 활용되고 있으나, 보안성에 대한 고려가 부족한 실정이다. 또한 사물인터넷 기기는 저메모리 저용량 초절전 모드로 작동하기 때문에 복잡한 방식의 보안 매커니즘을 적용하기 어렵다. 사물인터넷 기기는 계속해서 데이터를 축적하기 때문에 보안 취약점에 노출될 경우 피해 규모는 커질 것이다. 기기에 암호화를 적용하여 보안성을 강화한다고 하여도 기기를 탈취하여 암호 키를 알아낸다면 기기에 저장된 정보들은 악의적인 사용자들에게 노출될 수 있다. 대표적으로 드론과 폐쇄형 카메라는 해당 기기의 카메라를 이용해 거리와 사람 등을 촬영하는데, 이러한 기기가 탈취되면 기밀정보, 개인의 얼굴 데이터 등이 노출될 수 있다는 문제점이 제기된 바가 있다. 이러한 문제를 해결하고자 기기가 언제든지 공격자에게 탈취되어도 암호화된 데이터를 복호화할 수 없도록 키를 숨기는 화이트박스 암호에 대한 연구가 시작되었다. 화이트박스 암호는 암호 키를 공격자로부터 안전하게 보호하기 위해 암호 알고리즘인 룩업 테이블 안에 숨기는 암호화 방식이다. 하드웨어 기반의 암호화 방식은 한 번 만들어진 후 수정이 어렵다. 반면, 소프트웨어 기반의 화이트박스 암호화 방식은 룩업 테이블의 크기가 매우 크기 때문에 공격자는 암호 키를 쉽게 유추하기 어렵고, 새로 발견된 취약점의 패치를 비교적 쉽고 빠르게 반영할 수 있다. 그러나, 사물인터넷 기기, 드론 등은 제한된 메모리와 용량, 배터리를 갖고 있어 현재 적용되고 있는 암호화 방법 대비 룩업 테이블의 크기가 크고 암·복호화 속도가 느린 화이트박스 암호를 적용하기에는 어렵다. 또한, 카메라를 통해 얻는 이미지, 영상 데이터는 크기가 크기 때문에 이를 화이트박스 암호화하기 위해 크기가 큰 문제를 해결해야 한다.

따라서 본 연구에서는 화이트박스 암호가 룩업 테이블 크기 기준으로 암호화를 처리하는 특성을 활용하여 짧은 길이의 평문을 모아서 한 번에 처리하는 방안과 데이터의 유형을 고려한 데이터 압축 기법을 적용한 화이트박스 암호화 방안을 제안한다. 짧은 길이의 평문을 모아서 한 번에 처리하는 방안은 Chow와 XiaoLai 방식의 테이블 크기가 각각 720KB(KiloBytes), 18,000KB라고 가정했을 때, 제안 방식의 메모리 사용량이 Chow 방식에서 평균 약 29.9%, XiaoLai 방식에서 평균 약 1.24% 감소하는 결과를 얻을 수 있었다. 또한, 제안 방식의 시간 지연도는 15Mbps(Mega bit per second) 이상의 Traffic Load Rate일 때, Chow와 XiaoLai 방식에서 각각 평균적으로 약 3.36%, 약 2.6% 감소했다. 데이터 처리 효율성을 향상시키기 위해 허프만 코딩(Huffman coding)과 주성분 분석(Principal Component Analysis, PCA)을 이용한 압축 기법을 도입한 방안은 이미지 데이터의 경우 주성분 분석을 이용한 압축 기법이 시간 지연도와 압축률 측면에서 효과적임을 확인하였다. 또한, 여기에 짧은 길이의 평문을 모아서 한 번에 처리하는 방안을 함께 적용하였을 때, 주성분 분석은 원본 데이터와 허프만 코딩 대비 8개의 이미지 데이터를 더 처리할 수 있음을 실험 결과로서 얻을 수 있었다.

목 차

표 차 례

그 림 차 례

Ⅰ. 서 론

화이트박스(Whitebox) 암호는 암호 키를 공격자로부터 안전하게 보호하기 위해 암호 알고리즘인 룩업 테이블 안에 숨기는 암호화 방식이다. 하드웨어 기반의 암호화 방식은 한 번 만들어진 후 수정이 어렵지만, 소프트웨어 기반의 화이트박스 암호화 방식은 룩업 테이블의 크기가 매우 크기 때문에 공격자는 암호 키를 유추하기 어렵고, 새로 발견된 취약점의 패치를 비교적 쉽고 빠르게 반영할 수 있다. 그러나 소프트웨어 기반의 암호화 방식은 공격자가 공격 대상에 대해서 알 수 있는 정보의 양과 질에 따라 블랙박스(Blackbox), 그레이박스(Greybox), 화이트박스 공격을 받을 수 있다. 블랙박스 공격은 암호 모듈의 입출력값 정보만을 볼 수 있다고 가정한 공격이며, 그레이박스 공격은 블랙박스 공격자가 알 수 있는 정보 외에도 패턴, 전자기파 등의 부채널 정보도 알 수 있어 부채널 공격이라고도 불린다[1]. 마지막으로 화이트박스 공격은 메모리에 접근해서 수정까지 가능할 정도로 공격 대상에 대해 거의 모든 정보를 알 수 있는 공격이고 이러한 공격에도 안전할 수 있도록 만든 암호가 화이트박스 암호이다. 즉, 화이트박스 암호는 공격자에 의해 메모리 접근 및 수정이 이루어져도 암호 키를 보호할 수 있다.

화이트박스 암호는 2002년에 S.Chow가 처음 제안하였으며, DES(Data Encryption Standard) 방식과 AES(Advanced Encryption Standard) 방식을 제안하였다[2]. 그리고 그 이후 XiaoLai가 제안한 화이트박스 암호는 난독화가 AddRoundKey와 SubBytes 연산이 결합된 룩업 테이블인 T-box 2개 이상에서 작동하여 1개의 T-box에서 난독화가 이루어지는 Chow 방식의 화이트박스 암호 대비 Billet 공격으로부터 안전하도록 보안성 측면에서

개선했다[3]. 그러나, 난독화 과정이 복잡해진 만큼 암·복호화 속도와 사용되는 메모리 양이 증가하였다. 그 밖의 화이트박스 암호로는 S.Lee가 제안한 방식이 있다. S.Lee의 방식은 차분 계산 분석(Differential Computation Analysis, DCA) 및 차분 전력 분석(Differential Power Analysis, DPA) 공격으로부터 보호하기 위해 화이트박스 암호를 마스킹하는 것을 제안하고 공격으로부터 안전함을 증명하였다[4].

화이트박스 암호에 관한 연구가 진행되면서 쉽게 암호 키를 얻을 수 없다는 장점을 이용해 다양한 분야에 적용하려는 움직임이 나타나고 있다. 적용 연구로서는 드론 내부에 저장된 데이터가 악의적인 의도를 가진 사용자에게 노출되지 않아야 하므로, 키를 드론 내부에 저장하고 복호화 모듈을 드론에 구현하지 않는 연구[5]와 암호화폐 거래소의 개인 키를 관리하고[6] 디지털 콘텐츠를 불법적으로 사용하는 것을 막기 위해 암호화하여 정상 사용자만 이용할 수 있도록 하는 기술인 DRM(Digital Rights Management)[7] 등이 있다.

이외에도 인공지능, 빅데이터 등의 기술이 발전하면서 TV, 냉장고, 스피커 등 다양한 사물 간에 통신이 가능한 사물인터넷(IoT, Internet of Things) 기기가 증가하여 일상생활과 밀접한 부분에 사물인터넷 기기들이 활용되고 있다. 사물인터넷 기기의 보안 취약점을 이용한 공격 피해 규모가 커지면서, 사물인터넷 기기에도 화이트박스 암호를 적용하기 위한 보안 기술에 관한 연구도 진행되고 있다. 그러나, 사물인터넷 기기의 메모리와 배터리 용량이 제한적이기 때문에 상용 암호 대비 룩업 테이블의 크기가 크고 암·복호화 속도가 느린 화이트박스 암호를 적용하기에는 어렵다. 또한, 이미지나 영상 데이터는 데이터 자체의 크기가 크기 때문에 화이트박스 암호와 함께 사용할 경우 데이터를 처리하는 속도가 낮아지고 사용되는 메모리 양이 매우 클 수 있다.

따라서 본 연구에서는 화이트박스 AES 암호인 Chow와 XiaoLai 방식의 암호를 평문 데이터의 크기별로 키생성하고 암·복호화 하는 속도와 메모리 사용량을 비교하고, 화이트박스 암호를 사물인터넷 기기, 임베디드 장치 등 경량 장치에 적용할 수 있는 방안을 제안한다. 본 연구에서는 화이트박스 암호 방식이 룩업 테이블 크기 기준으로 암호화를 처리하기 때문에 처리 효율 향상을 위해 짧은 길이의 평문을 모아서 한 번에 처리하는 방안을 적용하였다. 또한, 데이터 유형에 따른 다른 압축 기법을 적용하고, 적용한 결과를 비교하여 가장 적합한 데이터 압축 방식을 확인한다. 그리고 압축을 통해 데이터를 처리할 때 효율성이 얼마나 개선되는지도 확인한다.

본 연구에서는 다음과 같이 세 가지 주요 기여점이 있다.

1) 화이트박스 암호가 룩업 테이블 크기 기준으로 암호화를 처리하는 특성을 활용하여 짧은 길이의 평문을 모아서 한 번에 처리하는 방안을 제안하였다.

2) 평문 데이터 응집(aggregation) 기법을 적용하는 제안 방식이 기존 대비 15Mbs 이상의 속도에서부터 효율적임을 실험을 통해 증명하였다.

3) 데이터의 특성을 고려한 적응적인(adaptive) 압축 선택 기법을 제안하여 효율적인 화이트박스 암호화를 수행하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 화이트박스 암호와 암호를 사물인터넷 등에 적용한 관련 연구를 비교 분석한 후 압축 기법에 대해서 설명한다. Ⅲ장에서는 화이트박스 암호의 룩업 테이블 사이즈를 고려한 암호화 기법을 제안한 후 Chow 방식의 화이트박스 암호와 XiaoLai 방식의 화이트박스 암호를 비교한 뒤, 제안 방식과 종래 화이트박스 방식을 비교한다. Ⅳ장에서는 데이터 처리 효율성 향상을 위해 데이터의 유형에 따른 압축 기법을 활용하여 화이트박스 암호의 문제점을 개선한다. 그리고 Ⅲ장

과 Ⅳ장의 실험 결과를 바탕으로 Ⅴ장에서 결론을 내리며 향후 연구 방향으로 마무리 짓는다.

Ⅱ. 관련 연구

화이트박스 암호는 리버스 엔지니어링 분석을 시도하는 공격자가 중간 연산 값이나 키를 유추하지 못하는 안전한 암호 기술로 주목받고 있다[8]. 또한, 화이트박스 암호는 물리적인 하드웨어 기기에 종속되지 않으므로 암호 알고리즘과 암호 키를 유연하게 적용할 수 있고, 하드웨어로 구현된 암호 칩과 비교할 때, 암호 알고리즘의 오류나 취약점 보완이 가능하다는 장점이 있다[1].

1. 화이트박스 암호

S.Chow는 표준 블록 암호 DES와 AES를 기반으로 암호화 연산을 할 때 테이블을 참조하여 수행하는 화이트박스 DES와 화이트박스 AES 개념을 처음으로 제안하였다[2]. 현재 대부분의 화이트박스 암호는 S.Chow 외 3인이 제안한 CEJO(S.Chow, P.Eisen, H.Johnson & Paul C. Van Oorschot) framework를 따르고 있다.

Fig. 1은 기존 AES-128과 화이트박스 AES 연산 순서를 나타낸 것으로 [1]의 그림을 재구성하였다. 화이트박스 AES의 암호 동작은 기존 AES 메커니즘과 동일하지만, 연산 순서에는 약간의 차이가 있다. AES-128 암호는 SubBytes, ShiftRows, MixColumns, AddRoundKey 순서의 9번 반복으로 암호화가 진행되며, 마지막 라운드는 SubBytes, ShiftRows, AddRoundKey(K_{10}) 순서로 실행한다. 여기에서 K_i는 i번째 라운드에서 사용되는 암호 키를 의미한다. 기존 AES 메커니즘과 달리, Chow 방식 화이

트박스 AES의 1~9라운드는 초기 AddRoundKey가 1번째 라운드에서 수행되면서 라운드의 마지막에 수행되는 AddRoundKey가 다음 라운드에서 연산 되기 때문에 ShfitRows, ShiftRows가 적용된 AddRoundKey, SubBytes, MixColumns 순서의 반복으로 이루어진다. 그리고 마지막 10번째 라운드는 ShiftRows, ShiftRows가 적용된 AddRoundKey(K_9), SubBytes, AddRoundKey(K_{10})로 연산이 수행된다[8]. 이때, 하나의 큰 룩업 테이블을 작은 룩업 테이블로 여러 개 만들기 위해 화이트박스 AES는 라운드 연산을 MixColumns로 마무리한다. ShiftRows는 AddRoundKey나 SubBytes와 순서를 변경해도 연산 결과는 동일해서 구현의 편의상 각 라운드의 맨 앞부분에서 계산하고 있다[8].

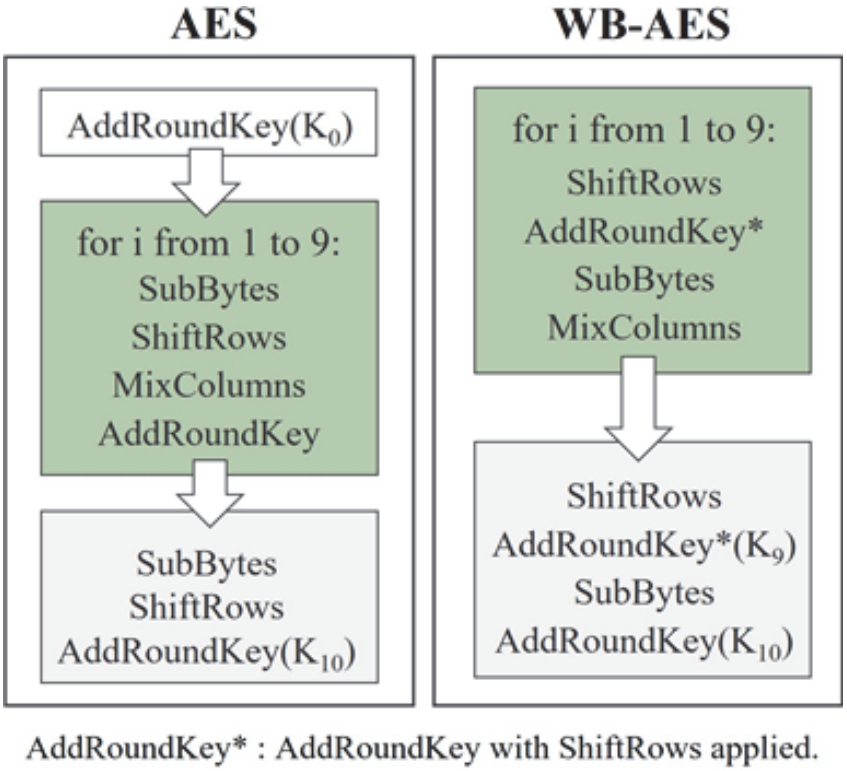

FIGURE 1. Operation of AES and WB-AES[1]

또한, 화이트박스 AES는 암호 알고리즘을 작은 룩업 테이블로 만들고 키를 숨기는 과정에서 중간값이 노출되지 않도록 인·디코딩 과정을 수행한다는 점이 AES와 차이가 있다. Fig. 2는 화이트박스 AES의 기본 원리를 나타낸 것으로, [8]의 그림을 재구성하였다. 별도의 테이블에서 계산되는 i개의 암호화 동작을 X_i라고 할 때, 암호화 동작 사이에 내부 인코딩(M_i)과 내부 디코딩(M_i^{-1}) 과정이 추가로 수행된다. 인코딩과 디코딩 과정이 상쇄되면서 기존의 암호화 동작인 X_i와 같아진다. 그러나, 안전성 문제로 외부 인코딩 G와 외부 디코딩 F^{-1}이 수행되기 때문에 기존 AES와의 암·복호화 결과와는 다른 결과를 나타낸다[8]. 외부 인코딩과 외부 디코딩 과정을 추가하고, 룩업 테이블을 여러 개로 나누어서 각각의 입력값과 출력값에 내부 인·디코딩을 적용함으로써 키 정보 등과 같은 민감한 정보가 연산 과정에 직접적으로 드러나지 않게 된다. 따라서 공격자는 룩업 테이블의 중간 연산 값을 예측하기 어렵다[9]. 하지만 종래 AES와 비교했을 때 화이트박스 AES는 큰 룩업 테이블 공간이 추가로 필요하다는 문제가 있다.

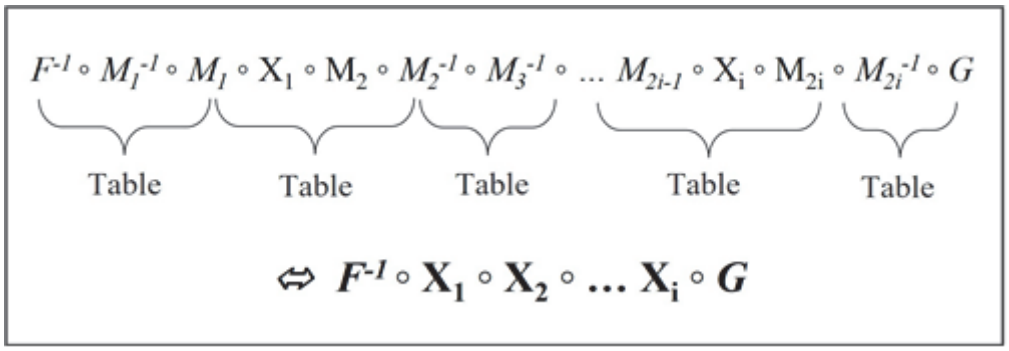

FIGURE 2. Basic Principles of WB-AES[8]

XiaoLai 방식[3]은 Chow 방식의 화이트박스 AES의 안정성을 개선하기 위한 암호 알고리즘 연구이다. Chow 방식의 화이트박스 암호는

ShiftRows 연산이 암호에 무관하기 때문에 MixColumns 연산이 난독화되지 않는 문제가 발생한다. XiaoLai 방식의 암호는 연산 과정을 수정하여 암호화 강도를 높이고자 하였다. 이에 따라 XiaoLai 방식의 룩업 테이블 크기가 커졌으며, Chow 방식의 룩업 테이블 크기는 770,048B(Bytes), XiaoLai 방식의 룩업 테이블은 20,994,048B로 Chow 방식 대비 XiaoLai 방식이 대략 27배 정도의 룩업 테이블 크기를 가지게 된다[3]. 보안 관점에서 XiaoLai 방식은 Chow 방식보다 안전하지만, 속도와 메모리 측면에서 더 느리고, 더 큰 메모리 용량이 필요하다.

따라서 종래 화이트박스 암호는 룩업 테이블 크기로 인해 암·복호화 연산 속도가 느리며, 큰 메모리 용량을 요구하기 때문에 모든 응용 분야에서 기존 암호 기술을 대체할 수 있는 기술은 아니다. 특히, 저전력, 경량, 저지연 장치를 지원하는 네트워크 애플리케이션의 성능 요구사항을 달성하기 어렵다. 최근에는 사물인터넷 환경에서 화이트박스 암호를 적용하기 위한 연구가 진행되고 있다. 종래 연구[10]는 암호화 과정이 사물인터넷 디바이스에서 이루어지고, 복호화 과정은 암호문을 별도의 컴퓨터 및 서버에 전송하여 사물인터넷 디바이스가 아닌 곳에서 진행되기 때문에 암호화 과정에만 화이트박스 암호를 적용하고 복호화 과정은 화이트박스 암호를 적용하는 대신 블랙박스 시나리오를 적용하는 방식을 제안하였다. 이는 기존 방식 대비 암·복호화 속도를 20% 이상 개선했고 사물인터넷 환경에 적용하는 실험까지 진행하였다. 그러나, 이 연구에서 제안한 방식은 화이트박스 암호를 개선한 것이 아니라, 복호화를 담당하는 곳을 블랙박스 시나리오로 변경한 것이다. 다른 종래 연구[11]는 사물인터넷 보안 체계를 개발할 때 블랙박스 기반 암호화 방식이 아닌 화이트박스 모델 도입 필요성을 강조하였다. 다양한 위협에 노출된 사물인터넷 환경의 보안성을 강화하기 위해서 화이트박스 암호에 적합한 블록 운용 방식을 실험하여, CBC(Cipher Block

Chaining) 모드가 화이트박스 암호 알고리즘에 적합함을 증명하였다. 그러나 해당 알고리즘의 암·복호화 속도와 크기를 고려하지 않아서 사물인터넷 환경에서 활용하기 어렵다는 문제가 있다. 경량 화이트박스 암호 기법을 제안한 연구[12]는 룩업 테이블을 포함한 정적 암호화 데이터 크기를 줄임으로써 리소스가 제한된 임베디드 장치에 적용하고자 했다. 또한, 적은 비용으로 효율적인 키 업데이트를 지원하여 실제 임베디드 장치에 적용 가능하다는 것을 입증했다. 경량 블록 암호의 화이트박스 구현 가능성을 확인한 연구[13]는 종래 경량 블록 암호인 KLEIN, LBlock(Lightweight Block cipher), PRESENT를 화이트박스 암호화하여 기존의 AES 기반 화이트박스 암호와 비교 분석했다. 블록 및 룩업 테이블의 크기를 줄임으로써 암호화 처리 속도를 향상시켰으나, 보안 강도에 대한 비교 분석은 없어서 종래 화이트박스 방식보다 암호의 보안 강도가 저하될 수 있다.

2. 데이터 압축

데이터는 압축을 통해 크기를 줄일 수 있다. 데이터를 압축하는 기법은 손실 여부에 따라 분류할 수 있다. 손실 여부에 따른 압축 기법에는 손실 압축(Lossy Compression)과 무손실 압축(Lossless Compression)이 있다. 손실 압축은 압축 및 압축 해제를 거친 데이터가 원본 데이터와 매우 유사하지만 데이터의 일부가 손실되는 기법이다[14]. 무손실 압축은 정보 손실이 일어나지 않는 기법이다. 두 압축 기법은 압축해야 하는 데이터의 유형에 따라 하나가 활용된다. 일반적으로 손실 압축은 이미지, 오디오 및 비디오 파일에서 사용되며, 무손실 압축은 텍스트, 프로그램, 이미지, 오디오 파일에서 사용된다. 특히 이미지, 비디오 데이터는 픽셀을 이용해 화질을 선명하게 표현할 수 있는데, 화질이 높을수록 픽셀의 수가 많고 잘게 쪼개어져 이들이 반복된 형태로 구성된다. 이를 활용하여 반복된 형태를 줄이는 방식으로 이미지, 비디오 데이터는 압축할 수 있고, 픽셀의 수가 줄어들더라도 일정 기준치까지는 원본 데이터와 유사한 데이터로 인식할 수 있다. 반면, 텍스트 데이터는 일부만 누락 및 변경되어도 다른 데이터로 인식되기 때문에 무손실 압축 방식이 활용된다. 손실 압축은 상대적으로 무손실 압축에 비해 압축률이 높고 손실율도 높다[15]. 손실 압축은 변환 코딩, 이산 웨이블릿 변환(Discrete Wavelet Transform, DWT), 이산 코사인 변환(Discrete Cosine Transform, DCT), 프랙탈 압축(Fractal compression), RSSMS(Rectangle Segmentation and Sparse Matrix Storage)의 방식이 있으며, 무손실 압축에는 LZW(Lempel - Ziv - Welch), 허프만 인코딩(Huffman Coding), 산술 인코딩(Arithmetic Coding), Shannon Fano 코딩 방식이 있다[16][17].

TABLE I

Lossy Compression and Lossless Compression

	손실 압축	무손실 압축
기법	변환 코딩, DWT, DCT, 프랙탈 압축, RSSMS	LZW, 허프만 인코딩, 산술 인코딩, Shannon Fano 코딩
데이터 형식	이미지, 오디오 및 비디오	텍스트, 프로그램, 이미지, 오디오
압축률	높음	낮음
손실율	높음	낮음

종래 연구에서는 이미지 데이터를 전처리한 후, 허프만 코딩과 LZW를 사용하여 압축률과 품질을 고려한 압축 방안을 제안하였다[18]. 그러나, 사물인터넷 환경 같이 제한된 환경에서는 압축으로 인해 소요되는 시간 지연도, 메모리 사용량은 고려해야 하지만, 본 연구에서는 자원이 제한된 환경을 고려하지 못하였다. 또한 주성분 분석(Principal Component Analysis, PCA)과 허프만 코딩을 함께 사용하는 방안을 제안한 연구도 있었다[19]. 주성분 분석으로 인해 발생한 코딩 중복을 제거하기 위해 허프만 코딩을 사용하였으며, 제안한 방식은 실험을 통해 JPEG2000보다 성능을 개선시켰음을 확인하였다. 해당 연구도 이전과 동일하게 데이터를 압축하면서 소모된 시간 지연도, 메모리 사용량은 고려하지 않았다.

Ⅲ. 룩업 테이블 사이즈를 고려한 암호화 기법

1. 제안 아이디어

최근 사물인터넷 환경에서 데이터를 전송하거나 암·복호화할 때 효율을 높이기 위해 평문 데이터를 단편화(Fragmentation)하거나 응집화(Aggregation)하는 방안에 관한 연구도 진행되고 있다. 단편화는 작은 단위의 데이터로 분할하는 것으로, 단편화된 일부가 외부에 노출되어도 전체 데이터에 위협이 되지 않는다. 응집화는 여러 데이터를 하나의 큰 데이터로 합치는 과정으로, 불필요한 오버헤드를 줄여서 지연시간을 개선할 수 있다. 종래 연구[20]는 Additively Homomorphic Encryption and Fragmentation scheme(AHEF)를 제안하여 무인 무선 센서 네트워크(Unattended Wireless Sensor Networks, UWSN)에서의 스토리지 오버헤드와 데이터 저장 및 전송 비용을 크게 낮추었다. 그러나 제안한 방식과 기존 방식을 메모리 측면에서 비교하지 않았다는 한계점이 있다.

본 연구에서는 기존 방식의 Chow와 XiaoLai 방식의 화이트박스 암호의 평문 크기에 따른 시간 지연도, 메모리 사용량을 측정하는 실험을 통해 시간 지연도와 메모리 사용량을 분석한 후, 평문을 각각의 암호에 맞는 룩업 테이블 크기만큼 응집하여 키 생성, 암·복호화 하는 방식을 제안한다.

Fig. 3은 제안하는 방식의 동작 원리를 보여준다. 화이트박스 암호의 입력(Input) 값으로 평문(Plain Text)를 넣고, 평문의 길이가 룩업 테이블 크기를 넘지 않을 경우, 버퍼에 임시로 저장한다. 이때, 평문의 길이가 룩업 테이블 크기와 같다면 버퍼에 저장하는 과정은 생략한다. 룩업 테이블의 크기와 같거나 커질 때까지 평문을 버퍼에 저장하며, 버퍼에 저장된 평문

의 길이가 룩업 테이블의 크기와 같아지면 응집하여 화이트박스 암호의 입력 값으로 넣어 암호화를 진행한다. 룩업 테이블의 크기보다 합산된 평문의 크기가 커지면 마지막으로 들어온 평문에서 룩업 테이블 크기를 넘는 부분만 잘라서 평문을 버퍼에 남겨둔 후 다음 평문이 들어올 때까지 기다린다. 출력(Output)은 암호문(Encrypted Text)으로 암호문의 길이는 룩업 테이블의 크기보다 크거나 같을 수 있다.

이러한 방식은 화이트박스 암호가 테이블 단위로 연산을 하기 때문에 메모리 사용량 측면에서 효율적이며, 시간 지연도는 룩업 테이블 크기가 될 때까지 다음 평문을 기다려야 하므로 일정 수준 이상의 속도로 평문이 입력될 경우에 효율적이다.

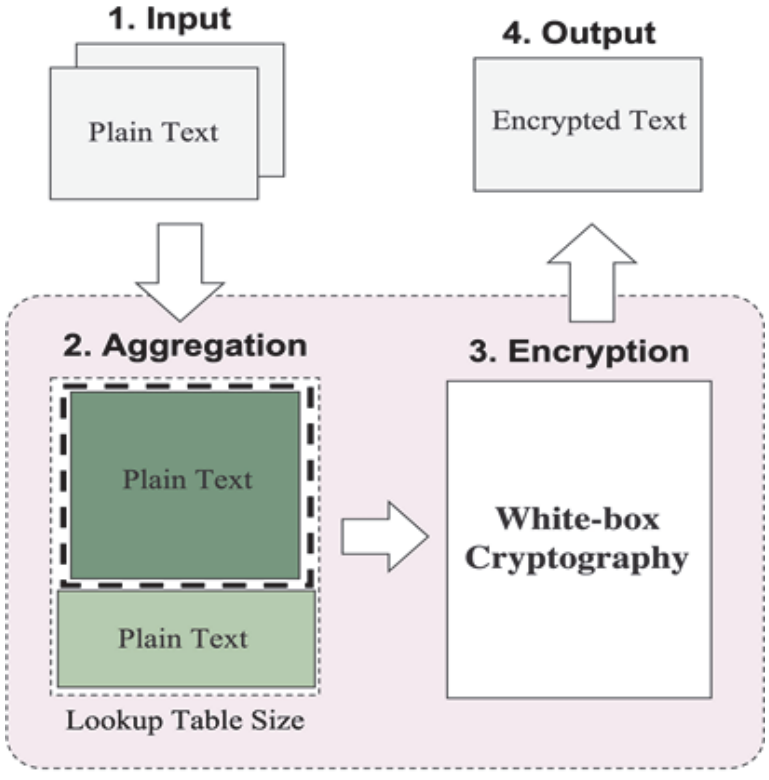

FIGURE 3. Proposed Scheme

2. 실험 조건 및 환경

본 연구는 데스크톱 Windows 10 pro 운영체제에서 VMware workstation pro에 Ubuntu 20.04 LTS(Long Term Support)를 설치한 실험 환경에서 수행되었으며, 자세한 실험 환경 및 버전은 Table I과 같다. Github에 공개된 Chow 방식의 Open Whitebox AES 암호와 XiaoLai 방식의 Open Whitebox AES 암호를 CBC 모드로 Go 언어, 파이썬, OpenSSL(Open Secure Socket Layer)를 사용하여 구현하였다[21]. 또한, 키를 생성하여 데이터를 암호화하고 복호화하기까지 걸리는 시간 지연도와 소모되는 메모리 사용량을 측정 및 비교하는 실험을 진행하였다.

시간 지연도는 파이썬의 time 모듈을 사용하였고 코드 실행 후 시간 지연도에서 코드 실행 전 시간 지연도를 빼서 최종 시간 지연도를 측정하였다. 메모리 사용량은 RAM 사용량을 기준으로 측정하였으며, 본 연구에서는 파이썬의 psutil(python system and process utilities) 모듈을 사용하여 메모리를 구하였다. 메모리 사용량은 코드를 실행하기 전 메모리 사용량을 측정한 후 코드를 실행한 직후 다시 메모리 사용량을 측정하였다. 이 사용량에서 코드를 실행하기 전의 메모리 측정량을 빼 코드 실행으로 인한 메모리 사용량을 계산하였다.

실험에 사용한 데이터는 데이터의 크기가 중요하므로, 무작위의 문자 및 숫자들을 배열하여 텍스트 데이터를 원하는 일정한 길이로 만들어 실험에 사용하였다.

TABLE Ⅱ

Experimental Environment and Version

구성요소	실험 환경 및 버전
OS	Windows 10 pro
CPU	Intel(R) Core(TM) i9-10850K CPU @ 3.60GHz
VMware	VMware workstation 15
VMware OS	Ubuntu 20.04 LTS
RAM	4GB(GigaBytes)
Go	v 1.13.8
Python	v 3.8
Openssl	v 3.0.0-alpha12

3. 실험 결과 및 분석

Fig. 4와 Fig. 5는 Chow와 XiaoLai 방식 화이트박스 암호의 하나의 룩업 테이블에 포함되는 평문 크기가 입력값일 때, 키 생성, 암호화, 복호화에 측정된 시간 지연도와 메모리 사용량을 표현한 그림으로 Fig. 4는 시간 지연도를 나타냈고 Fig. 5는 메모리 사용량을 나타냈다. Chow 방식의 룩업 테이블 크기는 770,048B이므로 이를 임의로 삼등분하여 240KB(Kilo Bytes), 480KB, 720KB의 평문 크기로 설정하였다. XiaoLai 방식의 경우는 룩업 테이블 크기가 20,994,048B이므로 평문 크기를 6,000KB, 12,000KB, 18,000KB로 만들었고 1,000번 실험을 반복하여 얻은 값을 평균 내었다.

Fig. 6과 Fig. 7은 Chow와 XiaoLai 방식 화이트박스 암호에 룩업 테이블의 1배, 2배, 3배, 4배에 해당하는 크기의 평문이 입력값일 때, 키 생성, 암호화, 복호화에 측정되는 시간 지연도와 메모리 사용량을 표현한 그림이며, Fig. 6은 시간 지연도를 나타냈고 Fig. 7은 메모리 사용량을 나타냈다. Chow 방식의 한 테이블 크기를 770KB, XiaoLai 방식의 경우는 테이블 하나의 크기를 20,000KB로 설정하였고, 1,000번을 반복한 결과의 평균값을 표현했다.

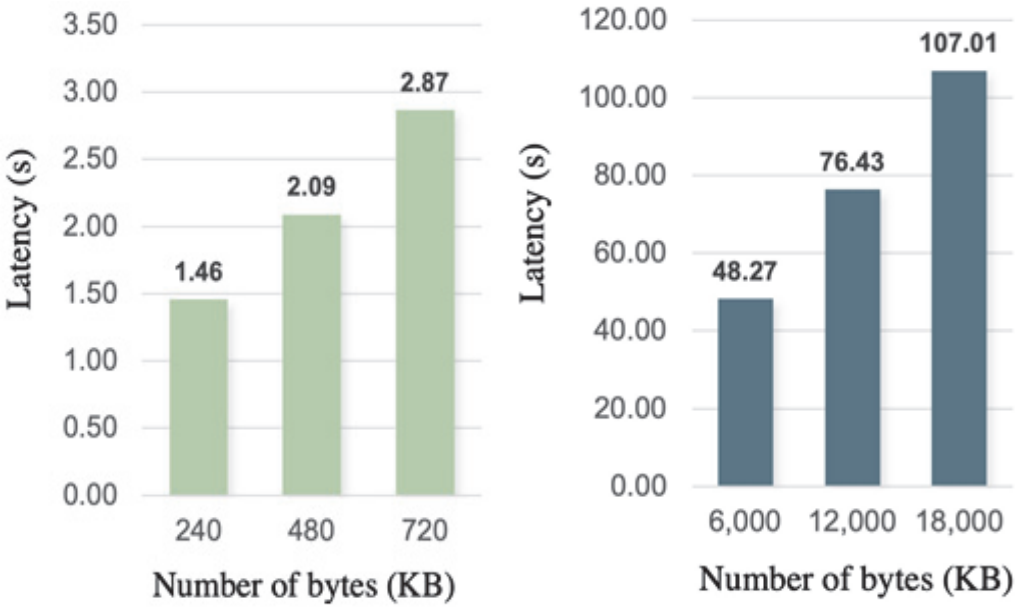

FIGURE 4. Latency Comparison of Chow and XiaoLai Schemes with Increasing Kilobytes: (a) Chow Scheme; (b) XiaoLai Scheme

Fig. 4는 평문의 바이트 증가에 따른 Chow와 XiaoLai 방식의 시간 지연도 측정 결과이다. 시간 지연도는 키를 생성하여 데이터를 암호화하고 복호화하는 총 시간을 측정한 것이다. 실험을 진행한 결과, Fig. 4 (a)에 해당하는 Chow 방식과 Fig. 4 (b)인 XiaoLai 방식은 평문의 바이트 수가 증가할수록 시간 지연도도 증가하였다. Chow 방식의 결과를 기준으로 살펴보았을 때, 240KB, 480KB, 720KB의 평문은 240KB 평문의 1배, 2배, 3배 크기에 해당하지만, 시간 지연도는 평문 크기가 증가한 배수만큼 증가하지는 않았다. 즉, 240KB의 평문은 720KB 평문에 비해 3배 작지만, 720KB의 시간 지연도인 2.87s는 240KB의 시간 지연도인 1.46s 대비 약 96.6%밖에 증가하지 않았다. 또한, XiaoLai 방식에서는 6,000KB의 시간 지연도인 48.27s 대비 18,000KB의 시간 지연도가 107.01s로 약 121.7%의 증가율을 보였다.

즉, XiaoLai 방식이 Chow 방식보다 테이블 내 시간 지연도 증가량이 크다는 것을 확인할 수 있었다.

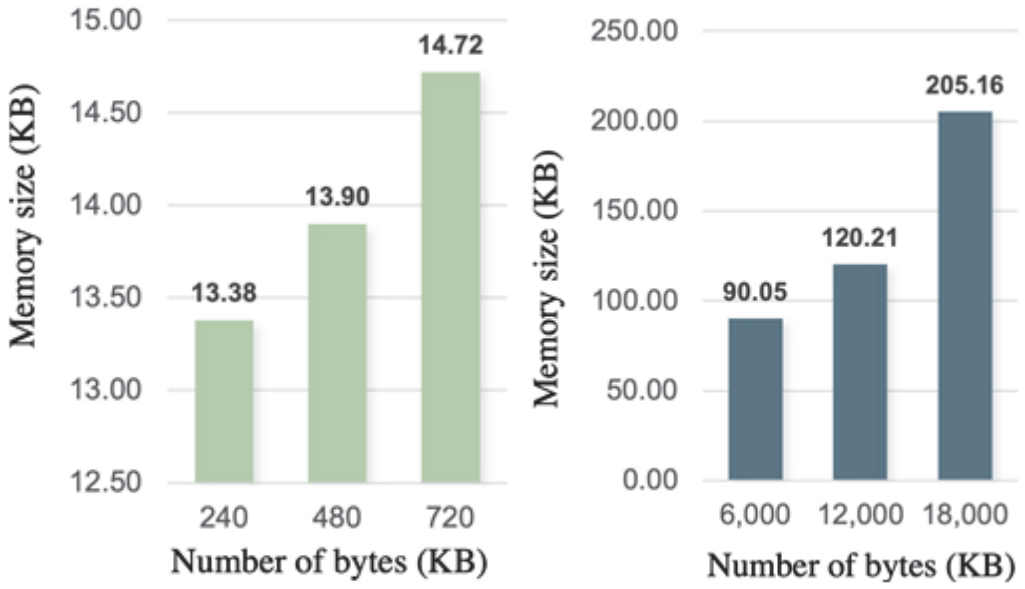

FIGURE 5. Memory size Comparison of Chow and XiaLai Schemes with Increasing Kilobytes: (a) Chow Scheme; (b) XiaoLai Scheme

Fig. 5는 평문의 크기에 따른 Chow와 XiaoLai 방식의 메모리 사용량을 표현한 그림이다. Fig. 5 (a)에 해당하는 Chow 방식과 Fig. 5 (b)인 XiaoLai 방식은 평문의 바이트 수가 증가할수록 메모리 사용량도 증가하였다. 그리고 평문의 크기가 증가한 배수만큼 메모리 사용량이 증가하지는 않았으며, 오히려 시간 지연도보다 더 작은 증가세를 보였다. Chow 방식의 경우, 240KB의 평문은 13.38KB의 메모리를 사용하였지만 720KB 크기의 평문은 14.72KB의 메모리를 사용했다. 이 실험 결과에 따르면, 240KB의 평문을 3번 처리할 때는 40.14KB의 메모리를 사용했지만, 720KB의 평

문이 14.72KB의 메모리를 사용하여 약 64% 메모리 공간이 절약되었다. XiaoLai 방식의 경우, 6,000KB의 평문은 90.05KB의 메모리를 사용하였지만 18,000KB의 평문은 205.16KB로 6,000KB의 메모리 사용량에서 대략 127.8%가 증가하였다. 이를 통해 18,000KB의 평문에서 6,000KB의 평문을 3배 한 270.15KB보다 적은 메모리를 사용하는 것을 확인할 수 있었다. 이러한 결과는 Chow 방식 대비 XiaoLai 방식이 테이블 내 메모리 사용량의 증가율이 더 크다는 것을 보여준다.

한 테이블 크기 내의 평문에 대한 시간 지연도와 메모리 사용량을 비교한 결과를 종합해 보면, 시간 지연도와 메모리 사용량 모두 한 테이블 내에 해당하는 평문의 크기가 증가할수록 함께 증가하였지만, 평문의 크기에 정비례하게 증가하지 않았다. 즉, 한 테이블 안에 해당하는 평문을 화이트박스 암호화할 때에는 짧은 길이의 평문을 여러 개 암호화하는 것보다는 긴 길이의 평문을 암호화하는 것이 효율적이다.

룩업 테이블 크기에 해당하는 만큼 평문 길이를 증가시키면서 실험한 Fig. 6에 따르면 Chow와 XiaoLai 방식의 시간 지연도는 증가하는 경향을 보이지만 테이블 수와 비례하게 증가하지는 않는다. Chow 방식의 경우, 평문이 룩업 테이블 1개에서 2개에 해당하는 크기로 증가할 때, 시간 지연도는 2.96s(Second)에서 5.18s로, 2.22s 증가하여 대략 75% 증가하였으나, 평문 크기가 룩업 테이블 2개에서 3개 크기만큼 증가할 때, 시간 지연도는 1.14s만큼 증가하여 대략 27.3% 증가하였다. XiaoLai 방식의 경우, 평문이 룩업 테이블 1개에서 2개에 해당하는 크기만큼 증가할 때, 시간 지연도는 134.19s에서 212.12s로, 77.92s 정도 증가하여 대략 58.07% 증가하였다. 또한, 평문이 룩업 테이블 2개에서 3개 크기만큼 증가할 때에는 시간 지연도가 98.11s 정도 증가하여 대략 46% 증가한 것으로 보인다. 결론적으로 룩업 테이블 크기만큼 평문 크기가 증가할 때, Chow 방식은 최소 약 27.3%

에서 최대 약 75%의 증가율을 보였으며, XiaoLai 방식은 최소 약 30.03%에서 최대 약 58.07%의 증가율을 보였다.

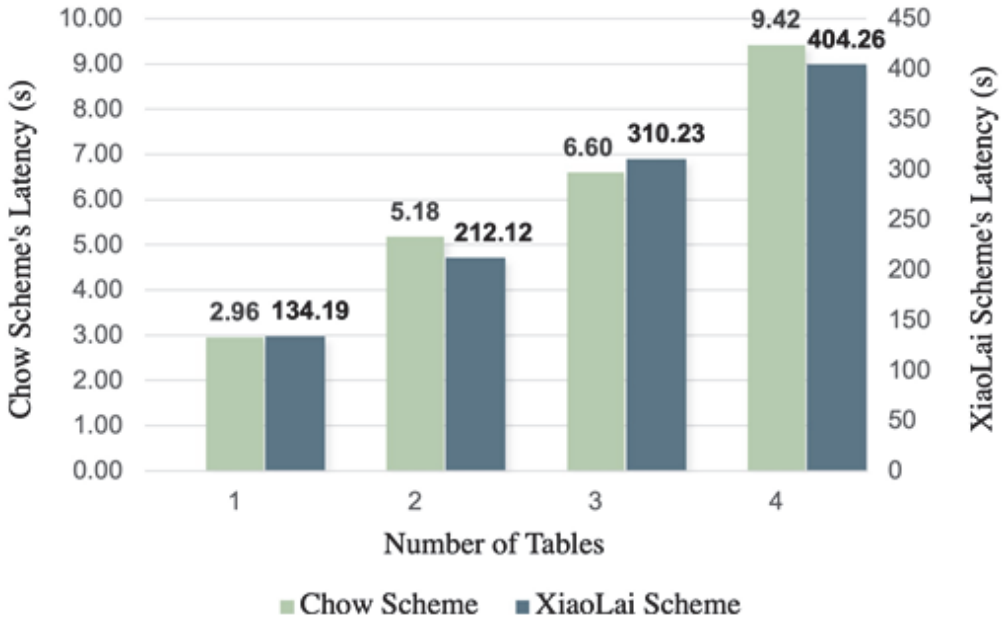

FIGURE 6. Latency Comparison of Chow and XiaoLai Schemes with Increasing Numer of Tables

평문을 룩업 테이블 크기에 해당하는 만큼 증가시켰을 때, Fig. 7에서는 Chow와 XIaoLai 방식 모두 메모리 사용량이 증가하는 경향을 보였다. Chow 방식의 경우, 평문이 룩업 테이블의 2개 크기만큼 증가하면, 메모리 사용량은 14.72KB에서 16.57KB로 1.85KB 증가하여, 대략 12.5%의 증가율을 보였다. XiaoLai 방식의 경우, 평문이 1개의 테이블에서 2개의 테이블 크기만큼 증가하면, 메모리 사용량은 136.92KB 증가하여, 대략 64.2%의 증가율을 보였다. 따라서 Chow 방식은 최소 약 12.5%에서 최대 약 14.1%의 증가율을 보였으며, XiaoLai 방식은 최소 약 25.7%에서 최대 약 75.7%의 증가율을 보였다.

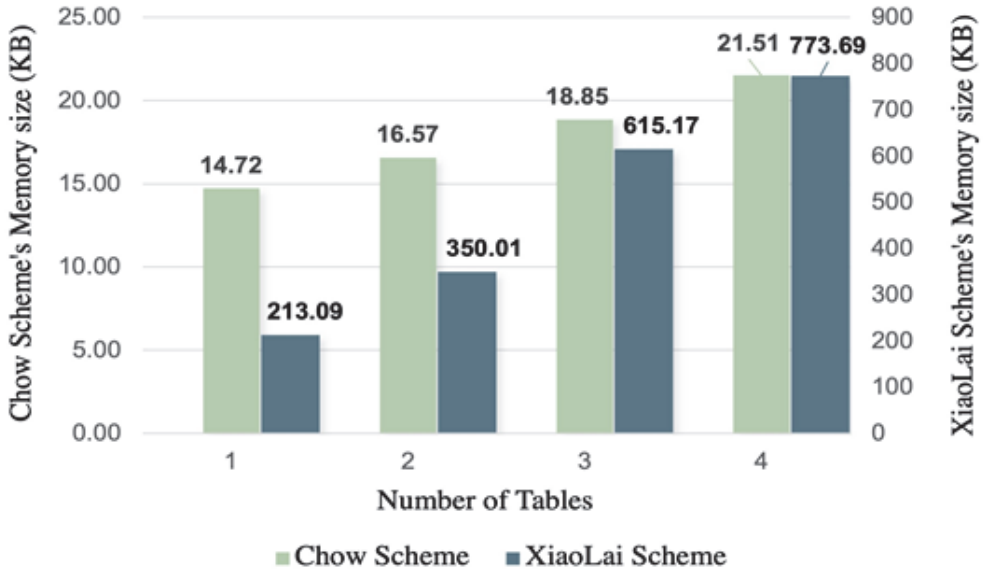

FIGURE 7. Memory Size Comparison of Chow and XiaoLai Schemes with Increasing Number of Tables

결론적으로 룩업 테이블 크기만큼 평문 크기가 증가할 때, Chow와 XiaoLai 방식은 시간 지연도와 메모리 사용량이 증가하는 경향을 보이며, 비례하게 증가하지는 않는다. 이는 테이블 내에서의 평문 크기에 따른 시간 지연도 및 메모리 사용량 비교와 동일하게 룩업 테이블의 개수에 비례한 평문에서도 짧은 길이의 평문을 여러 번 암호화하는 것보다는 길이가 긴 평문을 암호화하는 것이 효율적임을 보여준다. 또한, XiaoLai 방식이 메모리 사용량 측면에서 Chow 방식 대비 큰 폭으로 증가하는 경향을 보인다.

Fig. 8과 Fig. 9는 네트워크 성능에 따라 제안하는 방식과 종래 화이트박스 AES 암호 방식을 시간 지연도와 메모리 사용량 측면에서 비교하기 위해 Traffic Load Rate를 X축으로 설정한 그림으로, 10,000,000번을 반복하여 평균을 낸 실험 결과이다. 또한, 두 그림은 Chow 방식과 XiaoLai 방식의 한 테이블 크기가 각각 720KB, 18,000KB인 상황에서 룩업 테이블 크기

만큼 데이터를 모아서 처리하는 제안 방식과 데이터가 발생하는 즉시 처리하는 종래의 방식을 비교하였다. Fig. 8은 Chow 방식의 화이트박스 AES 실험 결과이고, Fig. 9는 XiaoLai 방식의 화이트박스 AES 실험 결과이다. Fig. 8과 Fig. 9 모두 X축은 Traffic Load Rate로, 5Mbps(Mega bit per second)에서 25Mbps까지 5Mbps 간격으로 설정하였다. Y축은 메모리 사용량으로 KB 단위이며, 보조 축은 시간 지연도로 초(second) 단위이다.

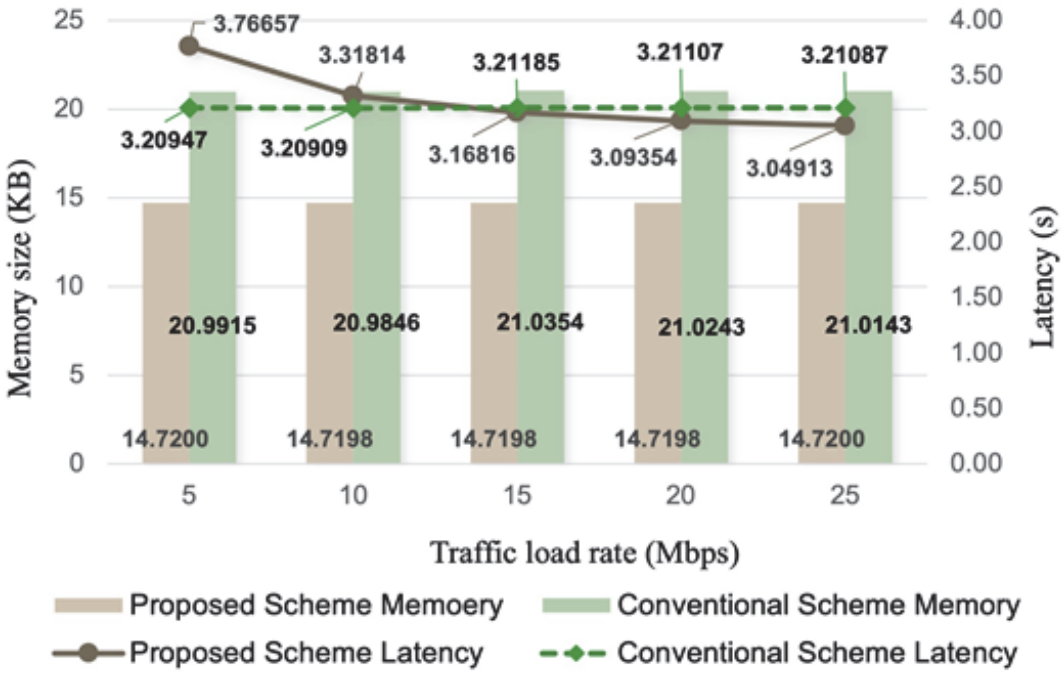

FIGURE 8. Comparison of Proposed and Conventional Chow Scheme with Traffic Load Rate

Chow 방식의 화이트박스 암호에서 종래의 방식과 제안하는 방식을 비교한 Fig. 8을 살펴보면, 메모리 사용량 측면에서는 Traffic Load Rate에 상관없이 모든 속도에서 제안 방식의 메모리 사용량 약 14.7KB로 종래 방식

의 메모리 사용량 약 21KB 대비 대략 29.9% 감소하였다. 또한, 시간 지연도 측면에서는 네트워크 성능이 10Mbps 이하인 경우 5Mbps, 10Mbps에서 제안 방식의 시간 지연도가 종래 방식 대비 각각 약 17.4%, 약 3.4% 증가했지만, 10Mbps 초과부터는 제안 방식의 시간 지연도가 종래 방식 대비 최소 약 1.36%에서 최대 5%까지 감소하였다. 이는 초기 5Mbps 대비 Traffic Load Rate가 5Mbps 단위로 증가할수록 제안 방식의 시간 지연도는 최소 약 11.9%에서 최대 약 19%까지 감소했기 때문이다. 이러한 결과가 나온 이유는 Traffic Load Rate가 낮아짐에 따라 테이블 단위로 키 생성 및 암·복호화를 진행하기 위해 다음에 들어올 평문을 기다리는 시간이 증가하기 때문이다. 즉, 네트워크 환경이 15Mbps 이상일 경우에는 제안 방식이 나으며, 이러한 속도를 내기 어려운 상태에서 시간 지연도가 중요한 환경에서는 기존의 방식을 사용하는 것이 효율적이다.

XiaoLai 방식의 화이트박스 암호에서 기존의 방식과 제안 방식을 비교한 Fig. 9를 살펴보면, 메모리 사용량 측면에서는 Traffic Load Rate에 상관없이 모든 속도에서 기존 방식의 메모리 사용량 약 207.7KB 대비 제안 방식의 메모리 사용량이 약 205.1KB로 약 1.24% 감소하였다. 또한, 시간 지연도 측면에서는 네트워크 성능이 10Mbps 이하인 경우 제안 방식의 시간 지연도가 기존의 방식 대비 5Mbps, 10Mbps에서 각각 약 11.8%, 약 1.98% 증가했지만, 10Mbps 초과부터는 제안 방식의 시간 지연도가 종래 방식 대비 최소 약 1.24%에서 최대 약 3.8%까지 감소하였다. 이는 초기 5Mbps 대비 Traffic Load Rate가 5Mbps 단위로 증가할수록 제안 방식의 시간 지연도는 최소 약 8.7%에서 최대 약 13.9%까지 감소했기 때문이다. 이 결과는 Traffic Load Rate가 낮아짐에 따라 테이블 단위로 키 생성 및 암·복호화를 진행하기 위해 다음에 들어올 평문을 기다리는 시간이 증가하기 때문이다.

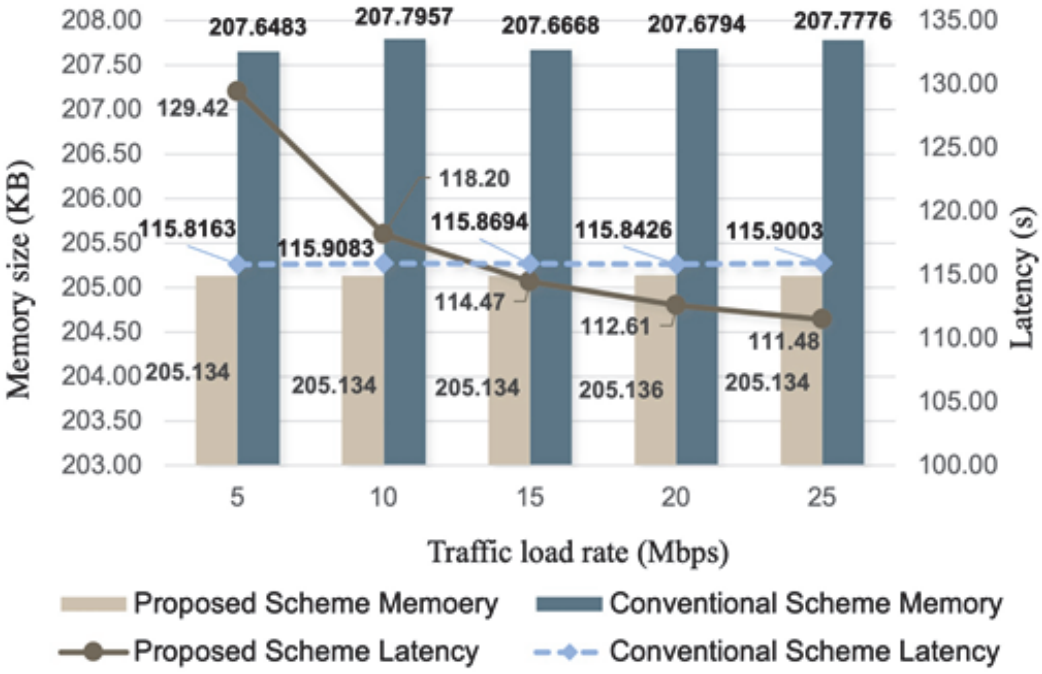

FIGURE 9. Comparison of Proposed and Conventional XiaoLai Scheme with Traffic Load Rate

또한, Chow 방식과 비교해 볼 때, XiaoLai 방식은 룩업 테이블 크기가 Chow 방식 대비 약 27배 크기 때문에 테이블 크기를 채울 수 있는 평문이 생기기 전까지의 시간이 상대적으로 더 길다. 이로 인해 네트워크 상황이 좋지 않을수록 다음 평문을 기다리는 시간에 의해 시간 지연도 증가량이 커진다. 즉, XiaoLai 방식은 룩업 테이블 크기가 크기 때문에 보안성이 중요하면서도 네트워크 환경이 15Mbps 이상인 환경에서는 제안한 방식을 사용하는 것이 좋고, 10Mbps 이하의 네트워크 환경이면서 시간 지연도가 중요한 경우에는 기존의 방식을 사용하는 것이 효율적이다.

Ⅳ. 데이터 처리 효율성 향상을 위한 압축 기법

1. 제안 아이디어

최근 인공지능 기술과 사물인터넷 기술이 발달하면서 다양하고 규모가 큰 데이터들이 끊임없이 생성되고 있다. 이러한 데이터들을 화이트박스 암호에 적용시키려면 제한적인 기기 환경과 느린 암호화 처리 속도의 문제를 해결해야 한다. 이는 원본 데이터가 아닌 압축된 데이터를 활용함으로써 문제를 해결할 수 있다. 데이터 압축 방식은 데이터의 형태에 따라 나뉘며, 이미지 및 영상 데이터는 손실 압축이 사용되고 텍스트같이 손실되면 안 되는 데이터는 무손실 압축을 사용하여 압축된다.

본 연구에서는 압축을 하지 않은 원본 데이터를 화이트박스 암호화 및 복호화는 경우와 이미지 데이터에 손실 압축 방식을 적용하고 텍스트 데이터에는 무손실 압축 방식을 적용한 경우, 그리고 이미지와 텍스트 데이터 모두 무손실 압축 방식을 적용한 경우를 실험한다. 이 실험에서는 시간 지연도, 메모리 사용량, 압축률을 측정하여 압축 방식의 효율성을 입증한다. 또한, 압축된 데이터를 화이트박스 암호의 룩업 테이블 크기만큼 응집하고 분산시키는 아이디어의 적용 가능성을 설명한다.

Fig. 10은 제안하는 방식의 동작 원리를 보여준다. 화이트박스 암호화를 하기 전, 데이터의 유형은 텍스트 데이터(Text Data)와 이미지 데이터(Image Data)로 분류되고 분류된 데이터들은 허프만 코딩(Huffman Coding)과 주성분 분석(Principal Component Analysis, PCA)를 통해 압축(Compress)된다. 그리고 압축된 데이터는 화이트박스 암호를 사용한 암호화(Encrypt), 복호화(Decrypt)를 거친 후 마지막으로 압축 해제

(Decompress) 단계에 도달한다.

이러한 방식은 화이트박스 암호의 시간 지연도와 메모리 문제를 해결할 수 있을 것이고 테이블 단위로 연산하는 특성에도 적용 가능하다. 압축 방식을 적용할 때에도 데이터를 테이블 단위로 쪼개거나 응집시키는 방식을 함께 사용하면, 메모리 사용량 측면에서 더욱 효율적이며 시간 지연도는 룩업 테이블 크기가 될 때까지 다음 평문을 기다려야 하므로 일정 수준 이상의 속도로 평문이 입력될 경우에 효율적이다.

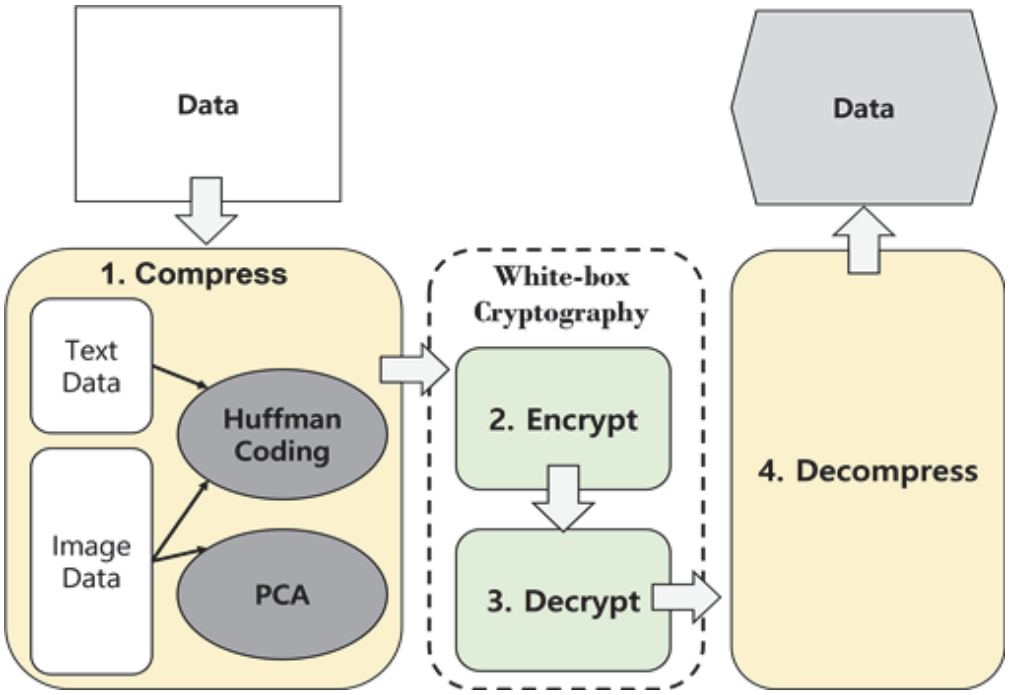

FIGURE 10. Proposed Scheme

2. 실험 조건 및 환경

본 연구는 데스크톱 Windows 10 pro 운영체제에서 VMware workstation pro에 Ubuntu 20.04 LTS를 설치한 실험 환경에서 수행되었으며, 자세한 실험 환경 및 버전은 Table Ⅲ과 같다. Github에 공개된 Chow 방식의 Open Whitebox AES 암호를 CBC 모드로 Go 언어, 파이썬을 사용하여 구현하였다[21]. 실험에 사용한 데이터는 두 종류의 이미지 데이터이다. 하나는 이미지넷 미니 데이터셋(Imagenet mini dataset)이며, 나머지는 UHDSR(Ultra High Definition Super Resolution)4K와 UHDSR8K 데이터셋을 사용하였다. 이미지넷 미니 데이터셋의 훈련 데이터(train data) 중에 RGB(Red, Green, Blue)를 가진 컬러 이미지 5000개를 선별하였고 텍스트 데이터는 영문자, 숫자, 기호를 랜덤으로 사용하여 선별된 이미지 데이터와 동일한 크기로 만들었다[22]. UHD4K, UHD8K 데이터셋은 각각 3, 840 × 2, 160 해상도, 7, 680 × 4, 320 해상도의 고화질(High Resolution, HR) 이미지로 구성되어 있다[23][24]. 동일한 이미지의 저화질(Low Resolution, LR)도 함께 구성되어 있었지만, 본 연구에서는 고화질 원본 이미지만 사용하였다. UHD4K는 5,999개의 훈련 데이터셋과 2,100개의 테스트 데이터셋, UHD8K는 2,029개의 훈련 데이터셋과 937개의 테스트 데이터셋으로 구성되어 있으며, 연구에서는 4가지 데이터셋에서 무작위하게 500개의 이미지를 선정하여 실험을 진행하였다. 또한, 해당 데이터셋에서도 이미지넷 데이터셋과 동일한 방식으로 텍스트 데이터를 생성하였다.

무손실 압축 기법으로는 허프만 코딩(Huffman Coding), 손실 기법으로는 PIL 라이브러리를 활용한 주성분 분석(PCA)을 사용하여 실험을 진행하였다. 주성분 분석은 고차원의 데이터에서 특성을 추출하여 원본 데이터의 특성은 갖지만 낮은 차원의 데이터로 변경하여 압축할 수 있는 방식이다.

본 연구에서는 추출한 이미지 데이터에서 최대로 할 수 있는 주성분 개수(n_components)가 38개였으며, 이를 기준으로 주성분 분석을 이용한 이미지 압축을 진행하였다.

데이터 특성에 따라 압축하고 그 데이터를 암호화하고 복호화하기까지 걸리는 시간 지연도와 소모되는 메모리 사용량, 압축률을 측정 및 비교하는 실험을 진행하였다.

TABLE Ⅲ. Experimental Environment and Version

구성요소	실험 환경 및 버전
OS	Windows 10 pro
CPU	Intel(R) Core(TM) i9-10850K CPU @ 3.60GHz
VMware	VMware workstation 15
VMware OS	Ubuntu 20.04 LTS
RAM	4GB
Go	v 1.13.8
Python	v 3.8

3. 실험 결과 및 분석

첫 번째 실험은 이미지넷 미니 데이터셋을 사용하였으며, 실험의 비교 대상은 압축을 하지 않은 원본 텍스트 데이터와 이미지 데이터 집합(Original, 원본 데이터), 허프만 코딩을 적용한 텍스트 데이터와 이미지 데이터 집합(Huffman, 허프만), 허프만 코딩을 적용한 텍스트 데이터와 주성분 분석을 이용한 압축 기법을 적용한 이미지 데이터 집합(PCA, 주성분 분석)로 분류하였다. 이후부터는 본 연구에서 비교 대상에 대해 상술하는 용어는 위와 같이 통일시켰다.

Fig. 11은 원본 데이터, 허프만, 주성분 분석의 시간 지연도에 관한 그래프로, x축은 텍스트 데이터와 이미지 데이터의 개수의 합인 5000개를 기준으로 x값이 증가할수록 이미지 데이터의 비율을 증가시키고 텍스트 데이터의 비율을 감소시켰다. 이미지 데이터 비율이 증가함에 따라 주성분 분석을 제외한 원본 데이터, 허프만은 시간 지연도가 증가하였다. 반면, 주성분 분석은 이미지 데이터가 증가함에 따라 서서히 시간 지연도가 감소하였다. 또한, 허프만은 원본 데이터에 비해 시간 지연도가 약 2배로 측정되었다. 이미지 데이터의 비율이 0일 때부터 허프만과 주성분 분석의 시간 지연도가 원본 데이터보다 2배 큰 것은 압축으로 인한 것으로 보인다. 이는 사용한 데이터의 크기가 작기 때문에 압축을 통한 감소 효과보다는 압축 과정의 연산으로 인해 증가한 것으로 보인다. 반면, 이미지 데이터의 비율이 증가함에 따라 원본 데이터는 시간 지연도가 약 700s 증가하였지만, 주성분 분석과 허프만은 상대적으로 시간 지연도가 적게 증가하거나 오히려 감소하는 것을 확인할 수 있었다. 크기가 큰 데이터를 사용하여 다시 실험한다면, 이미지 데이터 증가에 따른 시간 지연도 변화가 압축했을 때(주성분 분석, 허프만) 더 효과있다는 것을 보여줄 것이다.

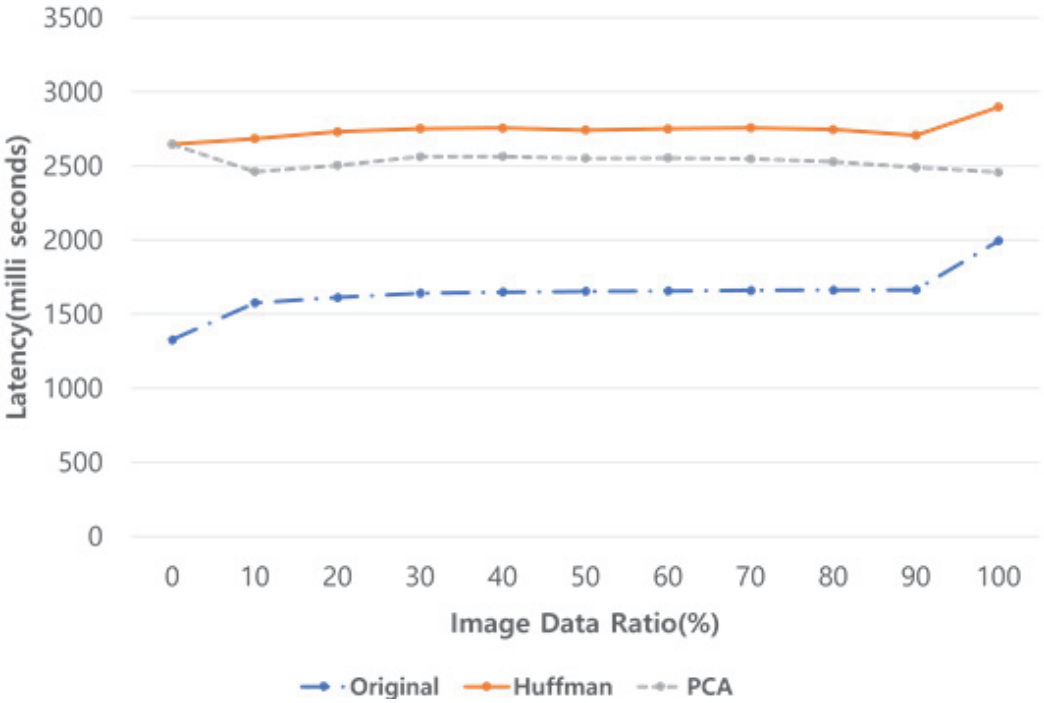

FIGURE 11. Comparison of Latency of Original and Huffman and PCA with Image Data Ratio

Fig. 12는 원본 데이터, 허프만, 주성분 분석의 메모리 사용량에 관한 그래프로, x축은 텍스트 데이터와 이미지 데이터 개수의 합인 5000개를 기준으로 이미지 데이터의 비율이 증가하는 것을 나타낸다. 여기에서의 메모리 사용량은 실험을 통해 사용된 RAM의 크기를 의미하며, RAM의 전체 크기인 4GB 중 얼마를 사용하였는지를 퍼센트로 표현하였다. 허프만과 주성분 분석의 메모리 사용량은 이미지 데이터의 비율이 0%일 때부터 원본 데이터의 메모리 사용량보다 작았으며, 이미지 데이터 비율이 증가함에 따라 사용량 차이가 비례하게 변하지는 않았다. 허프만과 주성분 분석은 시작할 때에는 메모리가 동일하였지만 이미지 데이터 비율이 증가함에 따라 주성분 분석의 메모리가 더 크게 증가하다가 70%부터 허프만의 메모리 사용량이 주성분 분석보다 더 커졌다. 허프만과 주성분 분석 둘 다 이미지

비율이 50%일 때, 메모리 사용량이 가장 적었다. 그리고 50% 이상부터는 메모리 사용량이 다시 증가하는 것을 확인할 수 있었다.

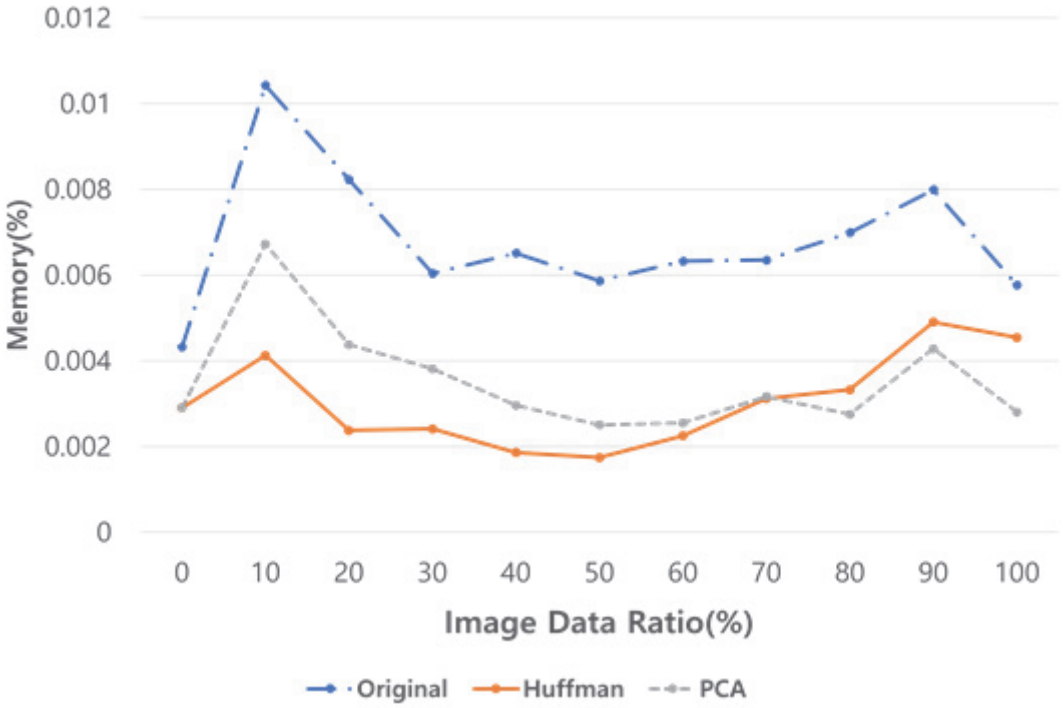

FIGURE 12. Comparison of Memory of Original and Huffman and PCA with Image Data Ratio

Fig. 13은 허프만, 주성분 분석의 압축률(Compression Rate)에 관해 비교한 그래프이다. 여기에서의 압축률은 Eq. (1)로 정의된다.

$$Compression\,Rate = \frac{Original\,Data\,Size - Compressed\,Data\,Size}{Original\,Data\,Size} \times 100\% \qquad (1)$$

압축률은 원본 데이터 크기(Original Data Size)에서 압축된 데이터 크기(Compressed Data Size)를 빼고, 이를 다시 원본 데이터 크기로 나눈 후 100을 곱하여 구할 수 있다.

원본 데이터의 경우, 압축이 적용되지 않았기 때문에 압축률을 나타내는 Fig. 13에는 포함되지 않았다. x축은 텍스트 데이터와 이미지 데이터 개수의 총합을 5000개로 할 때, x값이 커질수록 이미지 데이터의 비율이 증가하는 것을 나타냈다. 이미지 데이터 비율이 증가함에 따라 이미지 데이터 압축에 적합한 주성분 분석이 허프만에 비해 전반적으로 압축률이 9~22% 더 높았다. 이미지 데이터의 비율이 0일 때에는 허프만과 주성분 분석의 압축률이 동일하다가 비율이 증가한 이후에는 압축률이 최대 22% 차이가 났다. 이미지 비율이 100일 때에는 허프만의 압축률이 0에 가까워지는 것으로 볼 때, 이미지 압축은 허프만보다 주성분 분석이 효과적임을 알 수 있다.

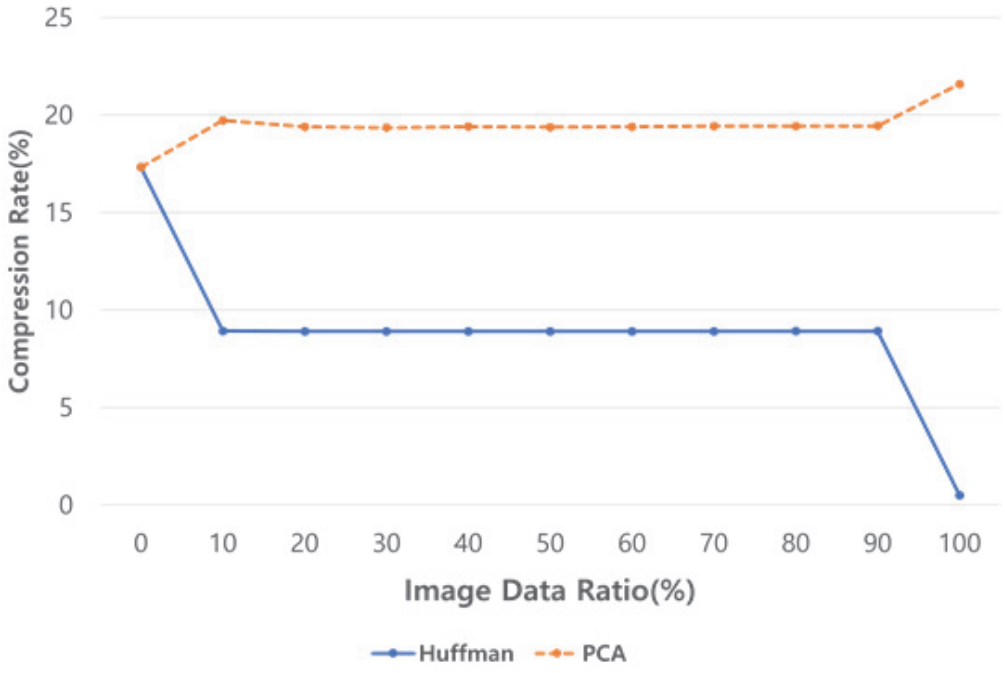

FIGURE 13. Comparison of Compression Rate of Huffman and PCA with Image Data Ratio

Fig. 14는 원본 데이터, 허프만, 주성분 분석의 압축률을 기반으로 화이트박스 AES Chow Scheme의 룩업 테이블 크기인 약 770,000B에 맞게 데이터를 응집시켜서 처리하면 몇 개의 데이터를 처리할 수 있는가에 대해 그래프로 표현하였다.

$$\begin{aligned} &The\ Number\ of\ Original\ Data \\ &= Whitebox\ Look up table\ Size(770000) \div Original\ Data\ Size(24062) \end{aligned} \tag{2}$$

원본 데이터는 압축률이 없으므로 데이터 원본을 사용하게 되며, 원본 데이터 수(The Number of Original Data)는 원본 데이터 크기에서 최대로 응집하여 처리 가능한 데이터 수를 의미한다. 원본 데이터 수는 Eq .(2)로 정의되며, 원본 데이터 크기는 실험에서 사용한 데이터의 평균을 구하여 사용하였다. 즉, Chow의 룩업 테이블 크기인 약 770,000B에서 이미지 데이터들의 평균 크기인 약 24,062B를 나누어 원본 데이터 수를 구하였다.

$$\begin{aligned} &The\ Number\ of\ Huffman\ Data \\ &= Whitebox\ Lookuptable\ Size \div (Original\ Data\ Size \times Huffman\ Compression\ Rate) \end{aligned} \tag{3}$$

허프만 코딩을 거친 데이터는 Fig. 13에서 구한 압축률을 사용하였으며, 허프만 데이터 수(The Number of Huffman Data)는 허프만 코딩을 적용한 데이터 크기에서 최대로 응집하여 처리 가능한 데이터 수를 의미한다. 허프만의 데이터 수는 Eq .(3)과 같이 정의된다. 허프만 코딩을 거친 이미지 데이터들의 평균 크기는 허프만 코딩의 압축률에 이미지 데이터들의 평균 크기인 약 24,062B를 곱하여 구하였다. 그리고 데이터 수는 Chow의 룩업 테이블 크기인 약 770,000B를 허프만 코딩을 거친 이미지 데이터들의 평균 크기로 나누어 구하였다.

$$\begin{aligned} &The\ Number of PCA\ Data \\ &= Whitebox\ Lookuptable\ Size \div (Original\ Data\ Size \times PCA\ Compression\ Rate) \end{aligned} \tag{4}$$

주성분 분석의 압축 기법을 거친 데이터도 Fig. 13에서 구한 압축률을 사용하였으며, (The Number of Huffman Data)는 주성분 분석의 압축 기법을 적용한 데이터 크기에서 최대로 응집하여 처리 가능한 데이터 수를 의미한다. 주성분 분석의 데이터 수는 Eq .(4)로 정의되며, 주성분 분석 압축을 거친 이미지 데이터들의 평균 크기는 주성분 분석의 압축률에 이미지 데이터들의 평균 크기인 약 24,062B를 곱하여 구하였다. 그리고 주성분 분석 데이터 수는 Chow의 룩업 테이블 크기인 약 770,000B를 주성분 분석 압축을 거친 이미지 데이터들의 평균 크기로 나누어 구하였다.

Fig. 14의 x축도 이전과 동일하게 텍스트 데이터와 이미지 데이터 개수의 총합을 5000개로 두고 이미지 데이터의 비율을 증가시킬 경우를 나타낸 것이다. 허프만과 주성분 분석은 텍스트 데이터도 압축되기 때문에 이미지 데이터의 비율이 0일 때부터 원본 데이터보다 처리 가능한 데이터 수가 많았다. 또한, 원본 데이터는 압축이 전혀 이루어지지 않았으므로 이미지 데이터셋의 비율에 따른 데이터 처리 개수도 변화가 없었다. 허프만과 주성분 분석은 시작할 때에는 처리 가능한 데이터 수가 동일하였지만, 이미지 데이터 비율이 증가함에 따라 주성분 분석의 데이터 수가 크게 증가하여 비율이 100일 때 약 8개의 데이터가 더 처리되는 것을 확인할 수 있었다. 이를 통해 주성분 분석이 허프만보다 이미지 압축에 효과적임을 알 수 있다.

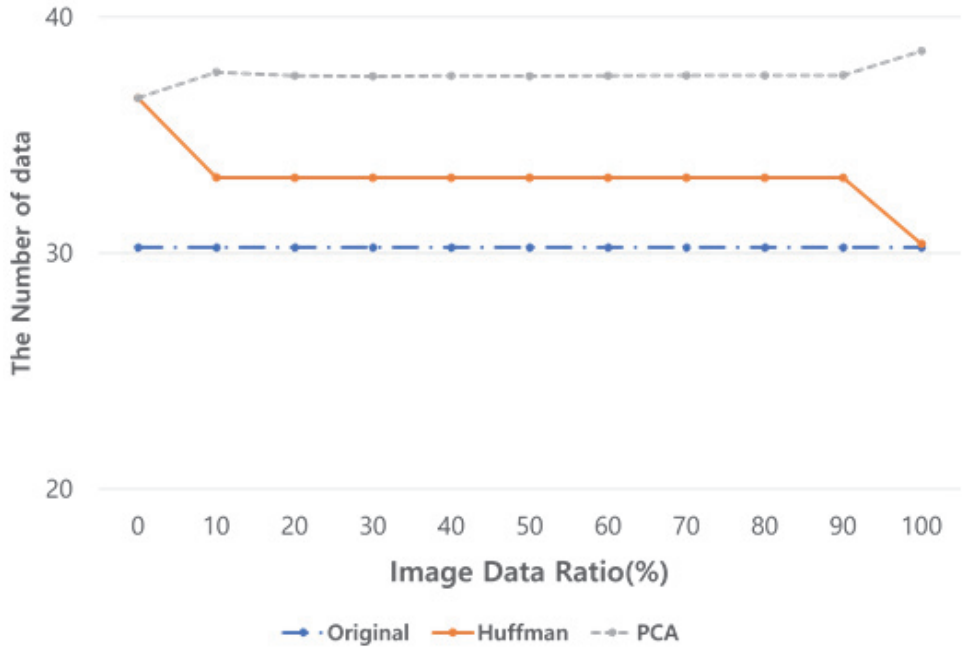

FIGURE 14. Comparison of The number of data of Original and Huffman and PCA with Image Data Ratio

두 번째 실험은 화이트박스 암호 Chow의 룩업테이블 크기가 770,000B이므로, 이에 상응하는 크기의 데이터인 UHDSR4K, UHDSR8K의 고화질 이미지를 실험 데이터로서 사용하였다. 실험의 비교 대상은 원본 텍스트 데이터(Original-Text)와 원본 이미지 데이터(Original-Image), 허프만 코딩을 적용한 텍스트 데이터(Huffman-Text),와 허프만 코딩을 적용한 이미지 데이터(Huffman-Image), 주성분 분석을 이용한 압축 기법을 적용한 이미지 데이터 집합(PCA)로 분류하였다. 이후부터는 본 연구에서 비교 대상에 대해 상술하는 용어는 전의 용어와 동일하게 사용하였다. 두 번째 실험에서 데이터 크기별 압축 효과를 확인하기 위해 약 500B, 1000B, 1500B, 2,000B, 2,500B, 3,000B 3,500B, 4000B 크기의 이미지들을 선별하여 총 1000번씩 시간 지연도, 메모리, 압축률을 구한 후 이 값들의 평균값을 비교

하였다.

Fig. 15는 데이터 사이즈가 증가함에 따라 각 데이터 유형의 시간 지연도 변화를 나타낸 그림이다. 텍스트 데이터는 이미지 데이터와 동일한 크기로 만들기 때문에 데이터 유형이 다르더라도 원본 데이터와 허프만에서의 텍스트와 이미지 데이터의 시간 지연도는 유사한 비율로 증가하였다.허프만은 압축 과정으로 인해 시간 지연도가 증가하여 원본 데이터보다 약 2.1배~2.4배 증가한 결과를 확인할 수 있었다. 반면 주성분 분석은 압축을 함으로써 원본 데이터에 비해 시간 지연도를 약 67.4%~86.4% 감소시킨 결과를 확인할 수 있었다. 종합해 보았을 때, 이미지 데이터 사이즈가 증가할수록 주성분 분석을 사용하여 압축하는 것이 시간 지연도 측면에서 효율적이며, 텍스트 데이터의 경우, 원본 데이터를 사용하는 것이 시간 지연도 측면에서 효율적임을 확인할 수 있었다.

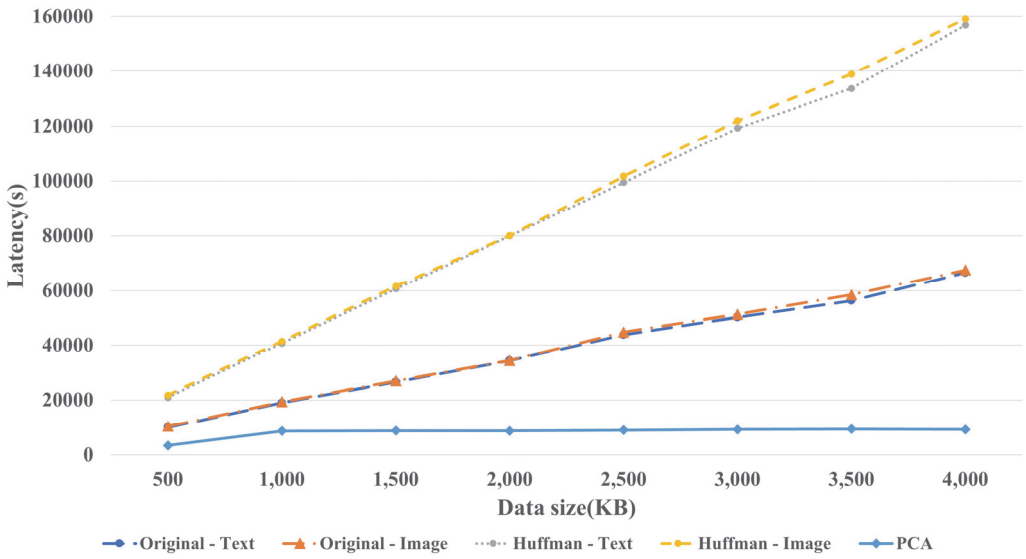

FIGURE 15. Comparison of latency variations between Original, Huffman, and PCA according to data size

Fig. 16은 데이터 사이즈가 증가함에 따라 각 데이터 유형의 메모리 사용량 변화를 나타낸 그림이다. 텍스트 데이터와 이미지 데이터는 동일한 크기를 가졌지만, 데이터를 압축하는 과정이 다르기 때문에 시간 지연도 측정 때와는 달리 두 유형의 데이터에서 메모리 사용량이 크게 차이가 났다. 또한, 압축 과정으로 인해 원본 데이터에 비해 허프만과 주성분 분석에서의 메모리 사용량이 배로 증가한 모습을 확인할 수 있었고, 특히 주성분 분석은 원본 데이터에 비해 최대 약 32.3배의 메모리를 사용하는 것을 확인할 수 있었다. 즉, 메모리 사용량 측면에서는 압축 과정으로 인해 메모리 사용량이 크게 증가하기 때문에 원본 데이터를 사용하는 것이 효율적임을 확인할 수 있었다.

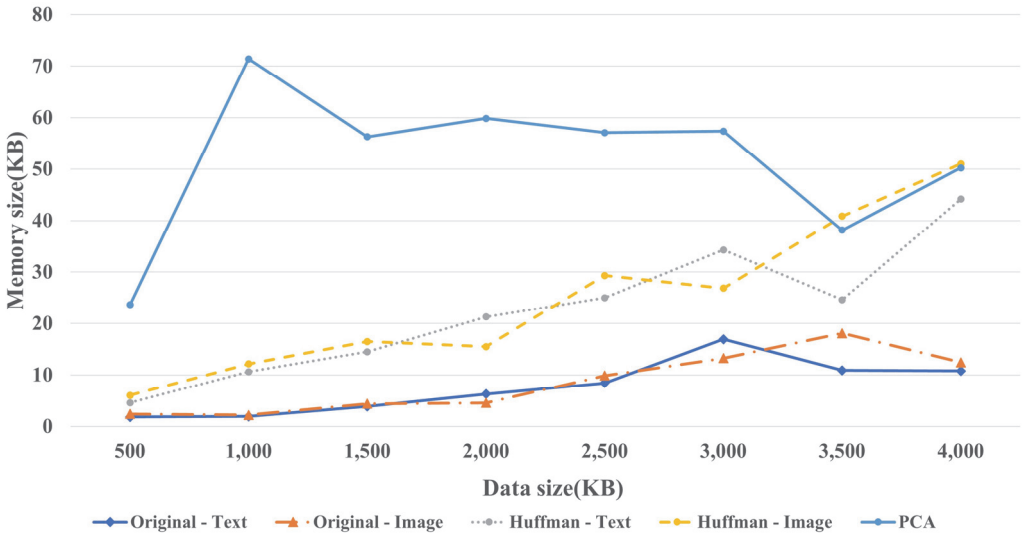

FIGURE 16. Comparison of Memory Size Changes in Original, Huffman, and PCA according to Data Size

Fig. 17은 데이터 사이즈가 증가함에 따라 각 데이터 유형의 압축률의 변화를 나타낸 그림이다. 원본 데이터의 경우, 압축 과정을 거치지 않았기 때문에 텍스트 데이터를 다룬 허프만, 이미지 데이터를 다룬 허프만, 주성

분 분석을 비교 대상으로 설정하였다. 허프만은 데이터 유형에 상관없이 압축하는 방법이 동일하기 때문에 허프만 이미지 압축률과 허프만 텍스트 압축률도 동일하였다. 주성분 분석은 허프만에 비해 압축률이 약 5.5~5.8배 큰 것을 확인할 수 있었다. 이미지 데이터를 압축하더라도 압축을 크게 해야 한다면 주성분 분석을 하는 것이 효율적임을 확인할 수 있었다.

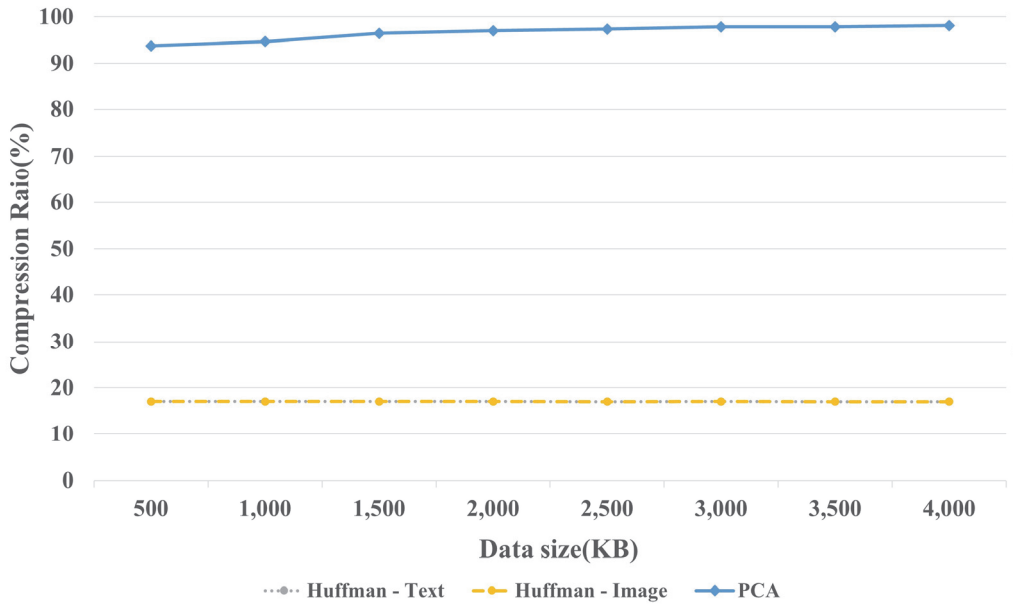

FIGURE 17. Comparison of Compression Ratio Changes between Original, Huffman, and PCA according to Data Size

V. 결론 및 향후 연구

화이트박스 암호는 소프트웨어 기반으로 구현되어 암호 연산의 중간값이나 키를 유추하기 어렵게 만들어서 보안성이 높은 암호 기술로 주목받고 있다. 하지만, 룩업 테이블 크기로 인해 연산 속도가 느리고 큰 메모리 용량이 필요하기 때문에 저전력, 저지연 장치를 지원하는 네트워크 애플리케이션의 암호 기술을 대체하기에는 한계가 있다.

따라서 본 연구에서는 대표적인 화이트박스 암호인 Chow와 XiaoLai 방식을 평문 데이터 길이에 따라 시간 지연도와 메모리 사용량을 비교하고, 종래 화이트박스 암호 방식의 메모리를 최적화하는 방안을 제안하였다. 실험을 통해 하나의 룩업 테이블 안에 해당하는 평문을 화이트박스 암호화할 때, 짧은 길이의 평문을 반복해서 암호화하는 방식보다 짧은 길이의 평문을 모아서 한 번에 암호화하는 것이 효율적임을 확인했다. 제안하는 방식은 종래 Chow 방식의 화이트박스 암호 대비 메모리 사용량을 평균적으로 약 29.9% 감소시켰으며, XiaoLai 방식의 화이트박스 암호 대비 평균 약 1.24% 감소시켰다. 또한, 제안하는 방식은 네트워크 성능이 Chow 방식의 화이트박스 암호와 XiaoLai 방식의 화이트박스 암호 모두 15Mbps 이상일 경우 기존 방식 대비 시간 지연도가 각각 평균 약 3.36%, 약 2.6% 개선되는 결과를 보였다. 두 번째 연구에서는 화이트박스 암호화를 진행할 때 데이터 처리 효율성을 향상시키기 위해 데이터 유형에 따른 압축 기법을 적용하는 방안을 제안하였다. 실험을 통해 이미지 데이터는 시간 지연도와 압축률 측면에서 주성분 분석을 활용한 압축을 하는 것이 효율적임을 확인하였다. 또한, 이미지 데이터를 압축할 경우, 화이트박스 암호화된 24,062B 크기의 데이터를 원본 데이터와 허프만 코딩 대비 최대 8개 더 처리할 수

있음을 확인하였다. 즉, 주성분 분석이 이미지 데이터 압축에 효과적이었다. 그러나, 압축 해제 상태의 데이터와 원본 데이터 간의 정확도를 비교하지 못하여 향후 연구에서는 머신러닝에 원본 데이터를 학습시킨 후 정확도, 정밀도 등을 측정 및 비교하는 실험을 진행할 예정이다. 또한, 본 연구는 두 가지 실험을 시뮬레이션으로 진행했기 때문에 하드웨어적인 제약이 있는 임베디드 시스템의 조건을 전부 반영하지는 못했다. 따라서 향후 연구에서는 임베디드 시스템 환경의 Real test-bed에서 제안하는 방식과 종래의 화이트박스 암호 기법을 비교·분석할 계획이다.

ACKNOWLEDGMENTS

본 논문의 Ⅲ. 룩업 테이블 사이즈를 고려한 암호화 기법은 2022년 9월 한국정보통신학회 논문지에 게재된 내용을 기반으로 작성되었습니다[25].

본 논문들을 지도해주신 이일구 교수님과 공저자로서 함께 연구에 참여한 김소연 학생에게 감사드립니다.

참 고 문 헌

[1] S. K. Lee, Y. S. Kang, “Principles and Statistical Analysis of White Box Ciphers,” ETRI, Daejeon, IITP Weekly ICT Trends-1977, 2020.

[2] Chow, S., Eisen, P., Johnson, H. and Oorschot, P.C.V., "White-box cryptography and an AES implementation," in International Workshop on Selected Areas in Cryptography, Berlin, Heidelberg. pp. 250-270, 2003.

[3] Y. Xiao and X. Lai, "A Secure Implementation of White-Box AES," in 2009 2nd International Conference on Computer Science and its Applications, Jeju. pp. 1-6, 2009. DOI: 10.1109/CSA.2009.5404239.

[4] Lee, S., Kim, T., & Kang, Y., “A masked white-box cryptographic implementation for protecting against differential computation analysis,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 10, pp. 2602-2615, Oct. 2018. DOI: 10.1109/TIFS.2018.2825939.

[5] S. M. Cho, and S. H. Seo, "Current status of cryptographic technology applied to drone security," Review of Korea Institute of Information Security and Cryptology, vol.30, no. 2, pp.11-19. Apr. 2020.

[6] D. H. Choi, C. K. Hong, "A Study on Key Protection Method based on WhiteBox Cipher in Block Chain Environment," Journal of Convergence for Information Technology, vol. 9, no. 10, pp. 9-15. 2019. DOI : 10.22156/CS4SMB.2019.9.10.009.

[7] Ghiţă, Stefan-Vladimir, Victor-Valeriu Patriciu, and Ion Bica, "A new DRM architecture based on mobilel code and white-box encryption," in 2012 9th International Conference on Communications, Bucharest, Romania. pp. 303-306, 2012. DOI: 10.1109/ICComm.2012.6262567.

[8] S.H. Kim, Y.K Lee, and B.H. Chung, "Analysis on Trends for White-Box Cryptography and Its Application Technology," Electronics and Telecommunications Trends, vol. 25, no. 5, pp. 137-146, Oct. 2010. DOI:10.22648/ETRI.2010.J.250512.

[9] Y. C. Lee, S. H. Jin, H. V. Kim, H. S. Kim and S. H. Hong, "New Higher-Order Differential Computation Analysis on Masked White-Box AES," Journal of The Korea Institute of Information Security and Cryptology, vol. 30, no. 1, pp. 1-15, 2020. DOI: 10.13089/JKIISC.2020.30.1.1.

[10] Albricci, Daniele Giacomo Vittorio, Michela Ceria, Federico Cioschi, Nicolò Fornari, Arvin Shakiba, and Andrea Visconti, "Measuring performances of a white-box approach in the IOT context," Symmetry, vol. 11, no. 8, pp. 1-19, Aug. 2019. DOI:10.3390/sym11081000.

[11] A. Saha and C. Srinivasan, "White-Box cryptography based data encryption-decryption scheme for IoT environment," in 2019 5th International Conference on Advanced Computing &Communication Systems, Coimbatore, India, pp. 637-641, 2019. DOI: 10.1109/ICACCS.2019.8728331.

[12] Y. Shi, W. Wei, H. Fan, M. H. Au and X. Luo, "A Light-Weight White-Box Encryption Scheme for Securing Distributed Embedded

Devices" IEEE Transactions on Computers, vol. 68, no. 10, pp. 1411-1427, Oct. 2011. DOI: 10.1109/TC.2019.2907847.

[13] Zhou, L., Su, C., Wen, Y., Li, W. and Gong, Z., "Towards practical white-box lightweight block cipher implementations for IoTs," Future Generation Computer Systems, vol. 86, no. 507-514, Sep. 2018. DOI: 10.1016/j.future.2018.04.011.

[14] Goyal, V. K., Fletcher, A. K., & Rangan, S. (2008). Compressive sampling and lossy compression. IEEE Signal Processing Magazine, 25(2), 48-56.

[15] Theis, Lucas, et al. "Lossy image compression with compressive autoencoders." arXiv preprint arXiv:1703.00395 (2017).

[16] Jasmi, R. Praisline, B. Perumal, and M. Pallikonda Rajasekaran. "Comparison of image compression techniques using huffman coding, DWT and fractal algorithm." 2015 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2015.

[17] Telagarapu, Prabhakar, et al. "Image compression using DCT and wavelet transformations." International Journal of Signal Processing, Image Processing and Pattern Recognition 4.3 (2011): 61-74.

[18] Djusdek, Djuned Fernando, Hudan Studiawan, and Tohari Ahmad. "Adaptive image compression using adaptive Huffman and LZW." 2016 International Conference on Information & Communication Technology and Systems (ICTS). IEEE, 2016.

[19] Vaish, Ankita, and Manoj Kumar. "A new Image compression technique using principal component analysis and Huffman coding." 2014 International Conference on Parallel, Distributed and Grid Computing. IEEE, 2014.

[20] Kapusta, Katarzyna, Gerard Memmi, and Hassan Noura, “Additively homomorphic encryption and fragmentation scheme for data aggregation inside unattended wireless sensor networks,” Annals of Telecommunications, vol. 74, no. 3-4, pp. 157-165, Apr. 2019. DOI: 10.1007/s12243-018-0684-x.

[21] doegox. Deadpool [Internet]. Available: https://github.com/SideChannelMarvels/Deadpool.

[22] Imagenetmini-1000, ILYA FIGOTIN [Internet]. Available: https://www.kaggle.com/datasets/ifigotin/imagenetmini-1000.

[23] Zhang, Kaihao, et al. "Benchmarking ultra-high-definition image super-resolution." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[24] HDCV Lab. Benchmarking Ultra-High-Definition Image Super-resolution (ICCV2021). [Internet]. Available: https://github.com/HDCVLab/UHD4K_UHD8K.

[25] 이진민, 김소연, and 이일구. "안전한 사물인터넷을 위한 AES 기반 경량 화이트박스 암호 기법." 한국정보통신학회논문지 26.9 (2022): 1382-1391.

ABSTRACT

Whitebox Encryption Optimization Techniques for the Internet of Things

Jin-Min Lee
Department of Future Convergence Technology Engineering
Graduate School of Sungshin University

IoT(Internet of Things) technology is being used in areas close to life, but security considerations are insufficient. In addition, IoT devices operate in a low-memory, low-capacity, ultra-power saving mode, making it difficult to apply complex security mechanisms. IoT devices continue to accumulate data, so if exposed to security vulnerabilities, the damage will increase. Even if encryption is applied to the device to enhance security, information stored on the device can be exposed to malicious users if the device is stolen and the encryption key is identified. Typically, drones and closed cameras use the device's camera to photograph streets and people, and problems have been raised that if such devices are taken away, confidential information and personal face data may be exposed. To solve this problem, research has begun on whitebox ciphers that hide keys so that encrypted data cannot be

decrypted even if a device is hijacked by an attacker at any time. Whitebox encryption is an encryption method that hides the encryption key in the lookup table, a cryptographic algorithm, to protect it from attackers. Hardware-based encryption is difficult to modify once created. Still, software-based white box encryption makes it difficult for attackers to infer encryption keys because of the large lookup table, and can reflect patches of newly discovered vulnerabilities relatively easily and quickly. However, IoT devices and drones have limited memory, capacity, and batteries, making it challenging to apply white box ciphers with a large lookup table size and slow encryption and decoding speed compared to the currently applied encryption method. In addition, since the image and image data obtained through the camera are extensive, it is necessary to solve the problem of significant size to white box encryption. Therefore, this study proposes a method of collecting and processing short-length plaintexts at once by utilizing the characteristics that white-box encryption processes encryption based on lookup table size and a method of white-box encryption using data compression techniques considering data types. First, it is a method of collecting short lengths of plain text and processing them at once. Assuming that the table sizes of the Chow and XiaoLai methods are 720KB(KiloBytes) and 18,000KB, respectively, the memory usage of the proposed method was reduced by about 29.9% on average in the Chow method and about 1.24% in the XiaoLai method. In addition, the time delay of the proposed method decreased by about 3.36% and about 2.6% on average in the Chow and XiaoLai methods, respectively, at Traffic Load Rate above

15Mbps(Mega bit per second). The method of introducing compression techniques using Huffman coding and Principle Component Analysis(PCA) to improve data processing efficiency confirmed that compression techniques using principal component analysis are effective in terms of time delay and compression rate for image data. In addition, when applying the method of collecting short-length plaintexts and processing them at once, it was obtained from the experimental results that principal component analysis can process eight more image data than the original data and Huffman coding.