

ISSN 1738-9593
제20권 1호

www.ksii.or.kr



2019 춘계학술 발표대회

논문집

4월 26일(금) ~ 27일(토)

동국대학교 서울캠퍼스 사회과학관



주최: 사단법인 한국인터넷정보학회

주관: 동국대학교 서울캠퍼스

후원: SK스토아(주), (주)카카오, (주)LGCNS, (주)아이티센

www.ksii.or.kr



KOREAN SOCIETY FOR INTERNET INFORMATION
사단법인 한국인터넷정보학회

B. 인터넷보안 (좌장 : 조규상 교수(동양대))

B-1 IoT 환경에 적합한 경량 블록암호 성능 분석	53
김의진, 곽진(아주대)	
B-2 의료정보접근을 위한 IoT 상황정보기반으로 한 자동인증시스템의 서비스 모델 구현	55
함규성, 주수중(원광대)	
B-3 ☆ ICS 사이버 공격 동향 분석	57
고영민, 김득훈, 곽진(아주대)	
B-4 Frida를 이용한 안드로이드 악성앱 분석기술 연구	59
이석훈, 류재철(충남대)	
B-5 ☆ 악성코드 작성자 그룹 식별을 위한 저자 클러스터링을 적용한 특징 추출	61
신건윤, 한명묵(가천대)	
B-6 최신 핀테크 서비스 현황 및 보안 동향 분석	63
고주성, 곽진(아주대)	
B-7 웹 브라우저 취약점 사례 분석 및 대응 방안	65
이민경, 최슬기, 곽진(아주대)	

C. 인터넷미디어 (좌장 : 이우찬 교수(인천대))

C-1 Conditional GAN을 이용한 단일 영상에서의 그림자 제거 방법	71
윤희진, 전준철(경기대)	
C-2 빅데이터 분석과 AHP를 이용한 악성 댓글 분류	73
김진우, 조혜인, 이봉규(연세대)	
C-3 U-Net을 이용한 간 분할 방법	75
김강직, 전준철(경기대)	
C-4 인터넷 먹방 프로그램의 긍정적 및 부정적 효과	77
지양슈에진, 황하성(동국대)	

B. 인터넷보안

IoT 환경에 적합한 경량 블록암호 성능 분석

김의진*, 콰진**

아주대학교 사이버보안학과

e-mail : *dmlwls0403@ajou.ac.kr, **security@ajou.ac.kr

A Performance Analysis of Lightweight Block Ciphers for IoT Environment

Eui-Jin Kim*, Jin Kwak**

Department of Cyber Security, Ajou University

요 약

IoT 기술이 발전함에 따라 IoT 기술과 다른 분야와의 융합이 이루어지고 있다. 대표적으로 IoT와 핀테크의 결합으로 인한 결제 및 금융 거래 서비스가 있으며, 이들은 모바일 및 웨어러블 기기를 이용한다. 하지만 이들 분야에서 이용되는 IoT 기기는 낮은 성능을 가지는 데 비해, 경량 환경을 위해 설계된 암호를 사용하지 않는 경우가 있다. 이는 핀테크 서비스에서 이용되는 기기 성능 저하의 원인이 되며, 기기 사용 환경 전체의 성능 저하로 이어질 수 있다. 따라서 본 논문에서는 국내의 경량 블록암호인 CLEFIA, PRESENT, LEA, HIGHT를 대상으로 CBC 운영모드를 이용한 암호 알고리즘별 특징, 암호·복호화 시간 분석을 통해 핀테크에서 개별 환경에 따라 경량 블록암호를 선택할 수 있도록 한다.

1. 서 론

모바일 기기를 통한 결제 및 금융 거래 서비스는 실생활에서 많이 이용되는 서비스 중 하나이다. 점차 IoT 기기에 모바일 결제 서비스 기능을 탑재한 스마트 워치와 같은 웨어러블 기기를 이용한 결제 서비스가 이용되고 있다[1]. 이러한 모바일 및 IoT 기기는 네트워크상에서 데이터를 전송하기 때문에 정보의 유출과 변조에 쉽게 노출될 수 있으며, 이를 방지하기 위해서는 데이터 암호화가 필요하다. 모바일 및 IoT 기기는 하드웨어 및 소프트웨어의 자원이 제한적이기 때문에 저전력 및 저성능에 적합한 경량암호를 사용해야 한다[2]. 대표적인 국내 경량 블록암호로는 LEA와 HIGHT, 국외 경량 블록암호로는 PRESENT와 CLEFIA 등이 있다. 하지만 실생활에 이용되는 IoT 기기의 경우 낮은 성능을 가지는 데 비해 블록암호를 사용하지 않는 경우가 있다.

따라서 본 논문에서는 2장에서 국내·외 경량 블록암호인 LEA, HIGHT, PRESENT, CLEFIA를 설명하고, 3장에서 경량 블록암호별 CBC 운영모드에서의 암호·복호화를 수행한 후 암호별 속도를 비교한 뒤, 4장으로 결론을 맺는다.

2. 관련 연구

2.1 국내 암호

□ LEA

LEA는 2013년 국내에서 개발된 경량 블록암호이며 국가보안기술연구소(NSR)에서 제안되었다. LEA는 ARX(Addition, Rotation, XOR)연산을 기본으로 하는 GFN(Generalized Feistel Network)구조를 갖는다. 평문 및 암호문은 128비트, 비밀키는 128/192/256비트 크기와 각 24/28/32라운드를 갖는 대칭키 방식의 블록암호이다. LEA는 TTAKKO-12.0223으로 국내 표준이 되었으며, 국제 표준인 ISO/IEC 29192-2로 반영을 추진하고 있다[3].

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT연구센터 지원사업의 연구결과로 수행되었음 (HITP-2019-2016-0-00304)

□ HIGHT

HIGHT는 2005년 한국인터넷진흥원, 국가보안기술연구소, 고려대가 공동으로 개발한 블록암호이다. ARX연산을 기본으로 한 변형 Feistel 구조이며, 초기변환과 최종변환을 제외한 32라운드를 갖는다. 평문 및 암호문은 64비트, 비밀키는 128비트 크기를 갖는 대칭키 방식의 블록암호이다. HIGHT는 2006년 한국정보통신기술협회(TTA) 표준으로, 2010년 ISO/IEC 18033-3으로 국제 블록암호 표준으로 제정되었다[4].

2.2 국외 암호

□ PRESENT

PRESENT는 Orange Lab(프랑스), Ruhr University Bochum(독일), Technical University of Denmark(덴마크)에 의해 개발되었다[5]. AES기반의 SPN(Substitution Permutation Network)구조의 경량 블록암호이며, 31라운드를 갖는다. 평문 및 암호문은 64비트, 비밀키는 80/128비트 크기를 갖는다[6]. PRESENT는 국제 표준화 기구에 ISO/IEC-29192로 등록되어 있다[5].

□ CLEFIA

CLEFIA는 SONY사에 의해 개발되었다. Feistel구조를 기반으로, SPN형태를 가지며, 평문 및 암호문은 128비트, 비밀키는 128/192/256비트 크기를 갖고, 각각 18/22/26라운드를 갖는다. CLEFIA는 ISO/IEC-29192와 IETF(Internet Engineering Task Force)에 RFC 6114로 등록되어 있다[7].

3. 국내·외 경량 블록암호 암호·복호화 시간 측정

3.1 측정 필요성

기존에 사용되는 IoT 기기 중에는 경량 환경을 위해 설계된 암호를 사용하지 않는 경우가 있다[8]. 따라서 사용자가 자신의 환경에 맞게 경량 블록암호를 적용하여 IoT 기기들 간의 통신의 효율성을 높일 필요가 있다. 이를 위해 국내·외의 경량 블록암호의 암호·복호화 시간 측정 결과를 제시한다.

3.2 측정 환경과 암호별 결과 분석

따라서 본 절에서는 경량 블록암호 적용 결과를 측정·비교한다. 표 1은 측정을 진행한 PC 환경이며, 표 2는 측정을 진행한 PRESENT[9], HIGHT[10], LEA[10], CLEFIA[11] 암호의 블록, 비밀키 크기 및 측정에 사용된 초기 벡터, 평문, 비밀키를 나타낸다.

(표 1) 테스트 환경

운영체제	Windows 10, 64-bit
프로세서	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
RAM	8.00GB

(표 2) 각 암호별 측정 조건

	PRESENT	HIGHT	LEA	CLEFIA
블록크기 (bits)	64	64	128	128
비밀키크기 (bits)	128	128	128	128
초기 벡터	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10			
평 문	0x41u, 0x50u, 0x61u, 0x63u, 0x64u, 0x65u, 0x66u, 0x67u, 0x68u, 0x69u, 0x6Au, 0x6Bu, 0x6Cu, 0x6Du, 0x6Eu, 0x6Fu, 0x6Gu, 0x6Hu, 0x6Iu, 0x6Ju, 0x6Ku, 0x6Lu, 0x6Mu, 0x6Nu, 0x6Ou, 0x6Pu, 0x6Qu, 0x6Ru, 0x6Su, 0x6Tu, 0x6Uu, 0x6Vu, 0x6Wu, 0x6Xu, 0x6Yu, 0x6Zu, 0x6au, 0x6bu, 0x6cu, 0x6du, 0x6eu, 0x6fu, 0x6gu, 0x6hu, 0x6iu, 0x6ju, 0x6ku, 0x6lu, 0x6mu, 0x6nu, 0x6ou, 0x6pu, 0x6qu, 0x6ru, 0x6su, 0x6tu, 0x6uu, 0x6vu, 0x6wu, 0x6xu, 0x6yu, 0x6zu			
비밀키	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10			
운영 모드	CBC			

그림 1~4는 측정에 사용된 코드의 일부이며, 그림 1은 PRESENT-CBC, 그림 2는 HIGHT-CBC, 그림 3은 CLEFIA-CBC, 그림 4는 LEA-CBC를 나타낸다.

```

1 for (int i = 1; i < text_len / BLOCK_SIZE; i++) {
2     memcpy(temp, cipher + (i * BLOCK_SIZE), BLOCK_SIZE);
3
4     present_decrypt(temp, key_1);
5
6     temp[0] ^= cipher[(i - 1) * BLOCK_SIZE];
7     temp[1] ^= cipher[(i - 1) * BLOCK_SIZE];
8     temp[2] ^= cipher[(i - 1) * BLOCK_SIZE];
9     temp[3] ^= cipher[(i - 1) * BLOCK_SIZE];
10    temp[4] ^= cipher[(i - 1) * BLOCK_SIZE];
11    temp[5] ^= cipher[(i - 1) * BLOCK_SIZE];
12    temp[6] ^= cipher[(i - 1) * BLOCK_SIZE];
13    temp[7] ^= cipher[(i - 1) * BLOCK_SIZE];
14
15    memcpy(decrypt_text + (i * BLOCK_SIZE), temp, BLOCK_SIZE);
16 }

```

(그림 1) PRESENT-CBC의 복호화 소스코드 일부

```

// Encryption Algorithm //
for (int i = 0; i < 100000; i++) {
    encStart = clock();
    nCipherTextLen = HIGHT_ENC_Encrypt(pbszUserKey, pbszIV, plaintext, 0, nPlainTextLen, pbszCipherText);
    encEnd = clock();
    encTime += (double)(encEnd - encStart) / CLOCKS_PER_SEC;
}

```

(그림 2) HIGHT-CBC의 암호화 소스코드 일부

```

for (int i = 0; i < 100000; i++) {
    encStart = clock();
    lea_set_key(&lea_key, mk, key_len);
    lea_cbc_enc(src2, src1, TEST_BLOCK_SIZE, iv, &lea_key);
    encEnd = clock();
    encTime1 += ((double)(encEnd - encStart)) / CLOCKS_PER_SEC;
}

```

(그림 3) LEA-CBC 암호화 소스코드 일부

```

1 for (int i = 1; i < pt_len / BLOCK_SIZE; i++) {
2     ClefiaDecrypt(temp, cipher + (i * BLOCK_SIZE), rk, r);
3
4     temp[0] ^= cipher[(i * BLOCK_SIZE)];
5     temp[1] ^= cipher[(i * BLOCK_SIZE)];
6     temp[2] ^= cipher[(i * BLOCK_SIZE)];
7     temp[3] ^= cipher[(i * BLOCK_SIZE)];
8     temp[4] ^= cipher[(i * BLOCK_SIZE)];
9     temp[5] ^= cipher[(i * BLOCK_SIZE)];
10    temp[6] ^= cipher[(i * BLOCK_SIZE)];
11    temp[7] ^= cipher[(i * BLOCK_SIZE)];
12    temp[8] ^= cipher[(i * BLOCK_SIZE)];
13    temp[9] ^= cipher[(i * BLOCK_SIZE)];
14    temp[10] ^= cipher[(i * BLOCK_SIZE)];
15    temp[11] ^= cipher[(i * BLOCK_SIZE)];
16    temp[12] ^= cipher[(i * BLOCK_SIZE)];
17    temp[13] ^= cipher[(i * BLOCK_SIZE)];
18    temp[14] ^= cipher[(i * BLOCK_SIZE)];
19    temp[15] ^= cipher[(i * BLOCK_SIZE)];
20
21    memcpy(decrypt_text + (i * BLOCK_SIZE), temp, BLOCK_SIZE);
22 }

```

(그림 4) CLEFIA-CBC 복호화 소스코드 일부

표 3은 실험 환경에서 4가지 경량 블록암호의 암호·복호화 속도를 측정결과를 정리한 것으로, LEA, HIGHT, CLEFIA, PRESENT의 순서로 시간이 적게 소요되었음을 알 수 있다. 경량 블록암호의 가시적인 지표를 위해 100,000회 반복하여 진행했다.

(표 3) 각 암호별 암호·복호화 시간 (100,000회 반복)

	PRESENT	HIGHT	LEA	CLEFIA
Encryption Time(sec)	11.625	0.517	0.237	9.553
Decryption Time(sec)	13.867	0.434	0.152	9.607

4. 결 론

본 논문에서는 국·내외의 경량 블록암호의 성능을 측정하였다. CBC 운영모드에서의 경량 블록암호의 암호·복호화 시간을 측정하였고, 측정값을 통해 핀테크 모바일 및 IoT 환경에서 기기의 성능을 고려한 적절한 경량 블록암호를 선택할 수 있게 한다.

참 고 문 헌

- [01] 정해식, “핀테크 시장 최근 동향과 시사점”, 정보통신기술진흥센터 주간기술동향 1841호, pp. 16-25, Apr 2018
- [02] 박해원, 신정욱, “64비트 블록암호 알고리즘 HIGHT의 효율적인 하드웨어 구현”, 한국정보통신학회논문지 제 15권 9호, pp. 1993-1999 Sep 2011
- [03] 한국정보통신기술협회, “128비트 경량 블록암호 LEA”, 한국정보통신기술협회, Dec 2013
- [04] 한국인터넷진흥원, “HIGHT 블록암호 알고리즘 사양 및 세부 명세서”, 한국인터넷진흥원, Jul 2009
- [05] Wikipedia, “PRESENT”, <https://en.wikipedia.org/wiki/PRESENT>
- [06] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, “PRESENT: An Ultra - Lightweight Block Cipher”, CHES07, pp. 450-466, 2007
- [07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata, “The 128-bit Blockcipher CLEFIA”, FSE’07, LNCS 3329, pp. 1-15, 2007
- [08] 한국인터넷진흥원, 미래창조과학부, “사물인터넷(IoT)환경에서의 암호·인증기술 이용 안내서”, KISA-GD-2016-0001, Mar 2017
- [09] kurtfu, “PRESENT block cipher”, <https://github.com/kurtfu/present>
- [10] 한국인터넷진흥원, “암호알고리즘 소스코드”, <https://seed.kisa.or.kr/kisa/reference/EgovSource.do>
- [11] Sony, “CLEFIA The 128-bit Blockcipher”, <https://www.sony.net/Products/cryptography/clefia/download/index.html>