논문 19-44-07-19 The Journal of Korean Institute of Communications and Information Sciences '19-07 Vol.44 No.07
https://doi.org/10.7840/kics.2019.44.7.1373

형태보존암호 FE1 알고리즘 구현

이 창 현◆, 김 가 을*, 홍 석 우*, 이 선 영°

Implementation of the Format Preservation Encryption FE1 Algorithm

Changhyun Lee◆, Gaeul Kim*, Seokwoo Hong*, Sun-Young Lee°

요 약

최근 국내에서는 개인 정보 유출 사고 증가에 따른 피해의 규모가 커지고 있다. 이에 대한 방안으로 개인 정보 암호화에 대한 규제가 강화되어, 개인 정보의 암호화가 필수적으로 되었다. 개인 정보의 기밀성을 보존하기 위해 AES와 같은 암호 알고리즘이 사용되고 있지만 블록 암호의 특성상 암호문이 평문의 형태를 유지할 수 없어 데이터베이스를 효율적으로 사용할 수 없다는 문제점이 있다. 이를 해결하고자 평문의 형태와 암호문의 형태가 같음을 보장하는 형태보존암호가 제안되었다. 형태보존암호 중 FE1은 Unbalanced Feistel Network 구조를 기반으로 하는 암호 알고리즘으로 도메인의 크기가 큰 경우에도 비교적 큰 성능 저하 없이 사용 가능하며 다양한 길이의 평문을 패딩 없이 암호화할 수 있다. 그러나 기존에 구현된 FE1은 평문의 범위값이 고정되어 있기 때문에 해당 값만큼의 평문만 암호화되는 제한사항이 있어 프로그램에 응용할 수 없다. 따라서 본 논문에서는 FE1 알고리즘을 분석한 뒤 Java로 구현하고 제한사항을 개선하여 평문을 임의로 입력할 수 있도록 구현하였다.

Key Words : FPE, FE1, Private information Encryption, Database

ABSTRACT

Recently, the number of personal information leakage accidents have been occurred. To solve these problems, encryption of personal information have been strengthened, and personal information must be encrypted. Block cipher algorithms such as AES are used to preserve the confidentiality of private information, but because of the characteristics of the block cipher algorithm, there is a problem in that database can't be used efficiently, since the ciphertext can not maintain of plaintext format. To solve this problem, FPE that guarantees that format of plaintext and format of ciphertext are the same is proposed. Of the various FPE, FE1 FE1 is a cryptographic algorithm based on the Unbalanced Feistel Network structure. Even if the domain is large, it can be used without performance decline and various lengths of plaintext can be encrypted without padding. However, due to the fixed range value of a plain text, there is a restriction that only plaintext corresponding to the range value is encrypted, so it can not be applied to a program. In this paper, we analyzed the FE1 algorithm and implemented it in Java to solve the restrictions and to input plain texts without restrictions.

※ 본 연구는 한국연구재단(교육부)의 지원을 받아 수행된 기초연구사업(NRF-2018R1DIA1B07047656)이며, 순천향대학교 학술연구비 지원으로 수행하였음.

◆ First Author : Soonchunhyang University Department of Information Security Engineering, wnddylch3215@daum.net, 학생회원
° Corresponding Author : Soonchunhyang University Department of Information Security Engineering, sunlee@sch.ac.kr, 종신회원
* Soonchunhyang University Department of Information Security Engineering, kimatom23456@gmail.com; hsw0323@naver.com
논문번호 : 201902-440-D-RN, Received January 31, 2019; Revised April 16, 2019; Accepted April 19, 2019

Ⅰ. 서 론

최근 금융기관 및 대형 유통 업체 등에서는 개인 정보 유출 사고가 빈번히 발생하고 있으며 피해의 규모는 점점 늘어나고 있다. 이에 따라 개인 정보를 암호화하도록 규제가 강화되고 있고, 개인 정보의 기밀성을 보존하기 위해서 카드번호나 주민등록번호 등에 대한 암호화의 필요성이 높아졌다. 현재 개인 정보의 암호화를 위해 AES와 SEED를 비롯한 다양한 블록 암호 알고리즘들이 사용되고 있다. 그러나 이러한 블록 암호 알고리즘을 이용할 경우 최소 블록 단위를 처리하기 때문에 평문을 128비트의 비트열로 변경을 해주어야 한다. 또한 비트열 연산을 하기 때문에 출력값의 형태도 비트열로 출력되므로 평문과 같은 형태를 유지할 수 없다.

데이터베이스에서 암호화를 사용하는 경우 가능한 데이터베이스의 구조를 변경하지 않으면서 암호화 기능을 활용할 수 있어야 하며, 데이터 처리와 저장소의 부하를 고려해야 한다. 그러나 기존 블록 암호 알고리즘들을 사용할 경우 입력 데이터 길이보다 출력 데이터 길이가 늘어나며 형태가 변경되고, 데이터베이스의 정의된 필드 및 자료형과 일치하지 않아 스키마 변경이 요구된다. 이러한 문제들을 해결하기 위해 평문의 형태와 암호문의 형태가 같음을 보장하는 암호화 기술인 형태보존암호(Format Preserving Encryption)가 주목받고 있다[1,2].

형태보존암호는 다양한 알고리즘이 있으며 그 중 FE1 알고리즘은 Unbalanced Feistel Network 구조를 기반으로 만들어졌다. 기존에 구현된 FE1 알고리즘은 사용자가 코드상에서 평문의 값을 직접 설정해 주어야 한다. 평문의 범위를 계산하는데 사용하는 모듈러 값 또한 설정해 주어야 하기 때문에 설정된 모듈러 값의 크기만큼 평문만 암호화가 적용이 되며 크기가 맞지 않으면 형태가 보존되지 않는 제한사항이 있다. 따라서 본 논문에서는 사용자가 입력한 개인 정보의 기밀성을 보존하기 위해 형태보존암호 알고리즘 중 FE1 알고리즘을 분석한 뒤 제한사항을 개선하여 Java로 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 형태보존암호의 전반적인 알고리즘과 형태보존암호 알고리즘 중 Unbalanced Feistel Network 구조를 살펴보고 3장에서는 FE1의 알고리즘과 라운드 함수 구조를 살펴본다. 4장에서 Java로 FE1 알고리즘을 구현하는 과정과 구현 결과를 설명한 뒤 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1 형태보존암호(Format Preserving Encryption)

형태보존암호는 Fig. 1과 같이 블록 암호에 기반을 두어 특정한 데이터 형식의 평문을 같은 형태로 암호화하는 암호 알고리즘이다. 형태보존암호는 다른 암호들과 달리 트윅(Tweak)을 사용한다. 트윅은 특정 형태의 데이터와 함께 형태보존암호의 입력으로 사용되는 부가정보이며 비밀로 보호되어야 하는 요소는 아니다. 또한 트윅이 변경될 때마다 암호문이 달라지므로, 암호문으로부터 평문을 유추할 수 있는 문제점을 방지할 수 있다. Spies가 도메인 크기별, 알고리즘 복잡도 별로 제안한 3가지 방법들이 여러 FPE 알고리즘의 기반이 되었다. Table 1은 3가지 방법들을 정리한 것이다[3].

Prefix Cipher는 알고리즘이 간단하고 도메인 크기가 작은 경우 사용에 용이하다. 그러나 도메인의 크기가 커지면 매핑 테이블의 크기 역시 커져 저장 및 사용이 어려워진다. Cycle Walking Cipher는 Prefix Cipher와 달리 전체 매핑 테이블을 유지할 필요는 없지만, 알고리즘이 언제 종료될지는 예측하지 못한다. 또한 원본 형태의 허용 범위가 작을수록 지나치게 많은 반복으로 실용성이 떨어지고 허용 범위값의 범위가 블록 암호화 알고리즘의 고정된 블록 크기에 영향을 받는다. 그에 반해 Generalized Feistel Cipher는 Prefix Cipher, Cycle Walking Cipher와는 달리 도메인 크기가 큰 경우에도 비교적 큰 성능 저하 없이 사용 가능하여 이를 기반으로 다양한 알고리즘이 제안되었다. 본 논문에서 사용한 FE1 알고리즘은 3가지 구조 중 각 라운드의 블록 암호 알고리즘의 출력이 보조키가 되는 Feistel Network를 변형한 Unbalanced Feistel Network 구조를 사용하였다[4].

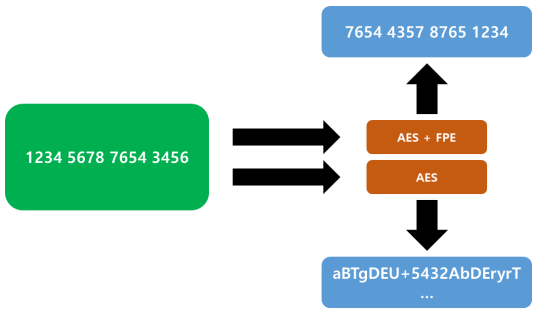


그림 1. 형태보존암호
Fig. 1. Format Preserving Encryption

표 1. FPE 알고리즘 기반 메소드
Table 1. Method For FPE

Method	Description
Prefix Cipher	When a domain to which encryption is applied is given as an integer, a pseudo-random weight is calculated and sorted by using a block encryption algorithm such as AES or DES for each integer belonging to the domain.
Cycle Walking Cipher	Using the block encryption algorithm to continue the algorithm until the same type of result as the original
Generalized Feistel Cipher	The method using Feistel network and using the output of the block encryption algorithm as the auxiliary key of each round

2.2 Unbalanced Feistel Network 구조

Unbalanced Feistel Network 구조는 Feistel Network 구조와 달리 두 개로 나누어진 문자열의 길이가 서로 다르게 입력된다. 라운드가 반복되는 과정에서 사용되는 함수 F는 라운드 함수(round function)이라 한다. 전체 알고리즘 과정은 다음과 같다. 먼저 입력값을 분할한 상태에서 왼쪽 값을 L이라 하고 오른쪽 값을 R이라 가정했을 때 L을 라운드 함수에 넣고 계산한다. 계산이 완료된 값을 R과 XOR 연산하여 출력값의 L로 설정하고 입력값의 L은 출력값의 R로 설정한다. 이후에는 이 과정을 반복하게 된다. Fig. 2는 Unbalanced Feistel Network 구조의 과정을 그림으로 보여준다.

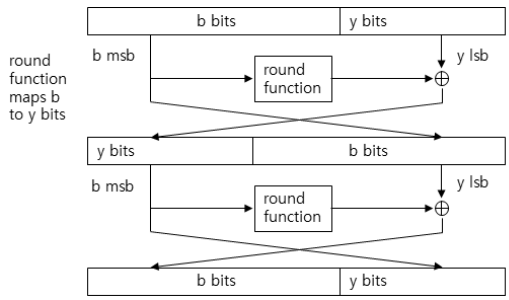


그림 2. Unbalanced Feistel Network 구조
Fig. 2. Unbalanced Feistel Network Diagram

Ⅲ. FE1 알고리즘

FE1 알고리즘은 Unbalanced Feistel Network 구조를 이용하여 다양한 길이의 평문을 패딩 없이 암호화할 수 있으며 어떤 라운드 함수를 사용해도 암호화와 복호화를 완전히 동일한 구조로 실현할 수 있다. FE1 알고리즘의 입력값으로는 알파벳, 숫자, 비트열 등의 형태이며 비밀키 K, 트윅(Tweak) T, 문자열의 길이 N을 사용한다. Table 2는 FE1 알고리즘에 사용되는 비밀키, 트윅(Tweak), 라운드 함수를 정의한 것이다.

FE1 알고리즘의 전체 과정은 Fig. 3과 같다[4].

FE1 알고리즘은 AES와 같은 블록 암호 또는 SHA256 같은 해시 함수를 라운드 함수에 적용한다. 본 논문에서는 HMAC-SHA256을 라운드 함수에 적용했다. HMAC-SHA256은 시간 효율성 측면에서 AES보다 약 67% 더 좋은 성능을 보여주었으며 키의 크기에 따른 성능의 차이가 없음이 알려져 있다[5].

표 2. FE1 알고리즘 파라미터
Table 2. FE1 Algorithm Parameter

Parameter	Description
Keys	The key space, a finite none empty set of binary strings.
Tweaks	The tweak space, a none empty set of strings. Variable length of tweak allows data to be used within a more flexible range.
F	The round function, a function that takes in a key K ∈ Keys, a permitted length n ∈ Lengths, a tweak T ∈ Tweaks.

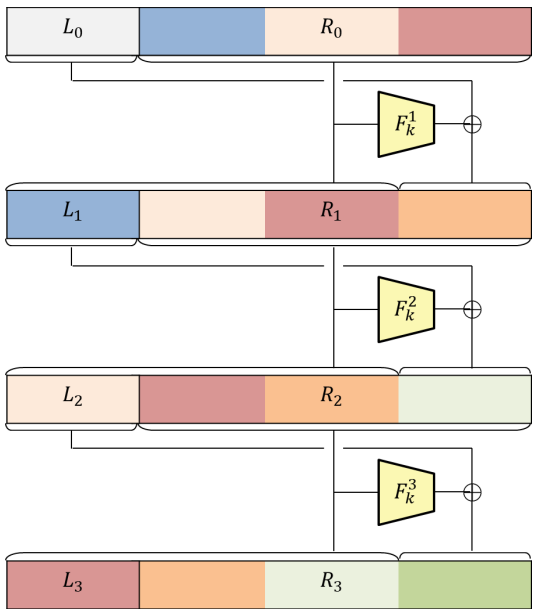


그림 3. FE1 알고리즘 구조
Fig. 3. FE1 Algorithm Diagram

3.1 HMAC

HMAC-SHA256은 HMAC(Hash-based Message Authentication Code, 해시 기반 메시지 인증 코드)와 SHA(Secure Hash Algorithm)-256 약자의 조합으로, 인증 코드를 해시와 키를 이용해 만드는 방법을 말한다[6].

Fig. 4는 HMAC을 이용하여 송·수신자 간의 메시지 무결성을 확인하는 과정을 나타낸다. 공유키와 사용자의 아이디를 HMAC Algorithm을 통해 메시지 인증 코드를 생성한다. 생성된 메시지 인증 코드와 사용자의 아이디 값을 함께 수신 측에 전송한다. 이후에 수신자는 송신자의 아이디와 공유키를 HMAC 알고리즘을 이용해 메시지 인증 코드를 생성하고 전송받은 메시지 인증 코드가 자신의 것과 일치하는지 비교한다. 수신자가 생성한 메시지 인증 코드와 전송받은 메시지 인증 코드가 일치한다면, 수신자는 송신자가 전송한 메시지의 무결성을 검증하게 된다.

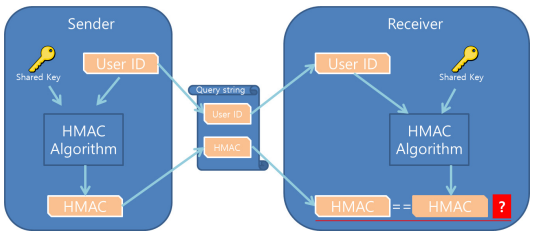


그림 4. HMAC 알고리즘
Fig. 4. HMAC Algorithm

3.2 SHA256

SHA256은 해시 함수로서, 메시지 인증을 제공하는 HMAC의 기본 구성 요소로 사용되며 데이터 무결성 검증에 주로 사용된다.

SHA256의 데이터 블록의 크기는 512비트 단위로 해시 연산에 적용되며, 해시 연산의 최소 단위는 32비트 워드가 된다. 총 64라운드의 해시 연산을 수행하며 연산 결과로 256비트의 고정된 길이의 값을 출력한다[7,8].

Ⅳ. FE1 알고리즘 구현

기존의 공개된 FE1 알고리즘은 고정된 평문의 범위값을 설정하기 때문에 해당 길이만큼의 평문만 암호화가 되는 문제점이 있다. 본 논문에서는 상기의 문제점을 해결한 FE1 알고리즘을 Java로 구현하였다.

4.1 FE1 알고리즘 구현 환경

FE1 알고리즘 구현은 다음 Table 3과 같은 PC 환경에서 구현하였다.

표 3. FE1 알고리즘 구현 환경
Table 3. FE1 Algorithm Implementation Environment

Classification	Specifications
OS	Window 10 64-bit
CPU	Intel(R) Quad Core i7 @ 2.9GHz
RAM	16GB
Develop Tool	Eclipse
Develop Language	Java

4.2 FE1 클래스 구성

본 논문에서 구현한 FE1은 FE1 알고리즘을 구현한 FE1 클래스, 예외처리를 설정한 FPE Exception 클래스, 소인수분해를 구현한 NumberTheory 클래스, 바이트 계산, 엔디안 등을 설정하는 Utility 클래스 총 4가지 클래스로 구성하였다.

4.3 FE1 알고리즘 구현

본 논문에서는 FPE 암호화 과정을 통해 유동적인 평문의 길이에 맞추어 암호화를 할 수 있도록 구현하였다. 또한 라운드 함수 메소드를 구현하고 이를 바탕으로 전체 FE1 알고리즘을 구현하였다.

4.3.1 메소드

(1) FPE 암호화기(FPE Encryptor)

FPE Encryptor는 형태보존암호 알고리즘에서 필요한 정보들을 HMAC을 사용하여 암호화하는 메소드이다. Fig. 5는 FPE 암호화기를 도식화하여 나타내었다. FPE Encryptor 파라미터는 키, modulus, 트윅이다. 비밀키인 key, 평문과 암호문의 범위인 modulus, 트윅인 tweak을 입력받는다. 평문의 범위인 modulus

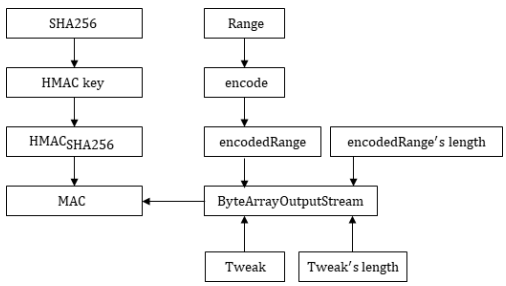


그림 5. FPE Encryptor 구조
Fig. 5. FPE Encryptor Diagram

표 4. FPE Encryptor 파라미터
Table 4. FPE Encryptor parameters

Parameter	Description
byte[] key	Secret key for HMAC_ALGORITEM
BigInteger modulus	Range of PlainText and Encrypt Text
byte[] tweak	tweak

를 측정하고 length함수를 이용하여 트윅의 길이를 구한 뒤 앞서 입력받은 값들과 함께 바이트 배열 스트림에 삽입한다. modulus를 측정하는 과정은 이후의 FPE 암호화 과정에서 진행한다. 입력받은 비밀키는 SHA-256을 적용한 후에 사용되며 HMAC-SHA256을 통해 MAC을 생성한다. FPE Encryptor의 파라미터는 Table 4와 같다.

(2) 라운드 함수(Round Function)

라운드 함수 메소드는 Fig. 3에서의 F_K를 구현한 것이다. 먼저 Table 5와 같이 총 라운드 수와 Unbalanced Feistel Network 구조에서의 우측 이진 문자열 R을 파라미터로 갖는다. 총 라운드 수와 우측 이진 문자열 R을 입력받으면 Fig. 6에서의 Round Function과 같이 FPE Encryptor의 결과값인 MAC과

표 5. 라운드 함수 파라미터
Table 5. Round Function Parameters

Parameter	Description
int roundNo	Total rounds
BigInteger r	Right binary string

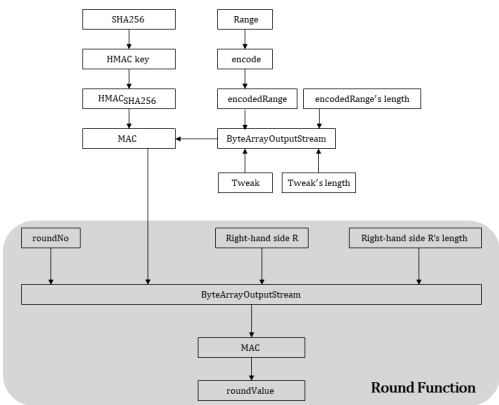


그림 6. 라운드 함수 구조
Fig. 6. Round Function Diagram

함께 MAC에 입력함으로써 암호화된 라운드 함수 결과 값을 도출한다[5].

4.3.2 FPE 암호화 (FPE Encryption)

FPE Encryptor와 라운드 함수를 이용하여 Fig. 6을 바탕으로 Java Math 클래스를 사용하여 구현하였다. Fig. 7은 FPE Encryption의 전체 과정이다.

FE1 알고리즘 메소드의 파라미터는 Table 6과 같다. 평문을 입력받고 평문의 길이는 length 함수를 통해서 구한다. 기존의 것과는 달리 평문과 암호문의 범위를 구하기 위해서 Java API 중 Math 클래스의 제곱함수(pow)를 사용한다. 밑은 10으로 지정하고 지수를 평문의 길이로 설정하여 나온 제곱 결과값에 1을 뺀 값을 평문과 암호문의 범위인 modulus로 설정한다. 예를 들어 평문이 '1234'이면 평문의 길이는 4가 된다. 따라서 10^4-1을 계산하면 평문과 암호문의 범위가 '9999'가 된다. 이 과정을 통해 평문의 길이에 제한 없이 형태가 보존된 암호화가 진행된다.

설정한 평문과 암호문의 범위인 modulus, 비밀키

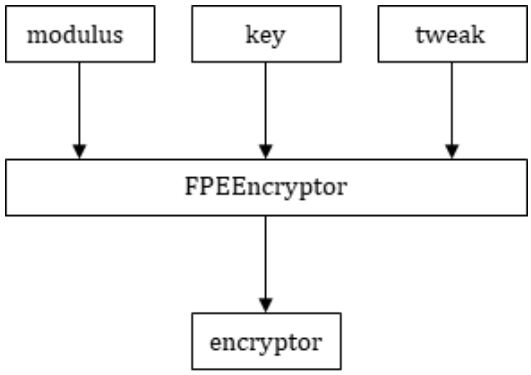


그림 7. FPE Encryption 과정
Fig. 7. FPE Encryption process

표 6. Encryptor 파라미터
Table 6. Encryptor Parameters

Parameter	Description
String input	plaintext
long cnt	plaintext's length
BigInteger Modulus	Range of PlainText and EncryptText
BigInteger plaintextValue	plaintext converted to BigInteger
byte[] key	Secret key for HMAC_ALGORITEM
byte[] tweak	tweak

인 key, 트윅인 tweak을 Fig. 7과 같이 FPE Encryptor에 입력하여 결과값 encryptor를 갖는다.

Fig. 8은 평문과 암호문의 범위인 modulus를 firstFactor와 secondFactor로 소인수분해한 뒤 이를 바탕으로 라운드 함수의 총 횟수를 결정하는 과정을 나타낸다.

Fig. 9는 Fig. 3을 바탕으로 라운드 함수를 수식화 한 것이다. Fig. 9를 토대로 Java API 중 Math 클래스를 사용하여 구현하였다. Table 7은 Fig. 9에서 사용한 용어들을 정의하였다.

Fig. 9에서 보이는 알고리즘은 라운드 수만큼 반복된다. 좌측 이진 문자열은 입력받은 평문을 secondFactor로 나눈 값이며 우측 이진 문자열은 입력받은 평문을 secondFactor로 모듈러 연산을 한 값이다. 좌측 이진 문자열과 우측 이진 문자열을 연접한 값이 W가 되며 이를 우측 이진 문자열과 firstFactor를 곱한 값에 더하면 한 라운드가 끝나게 된다.

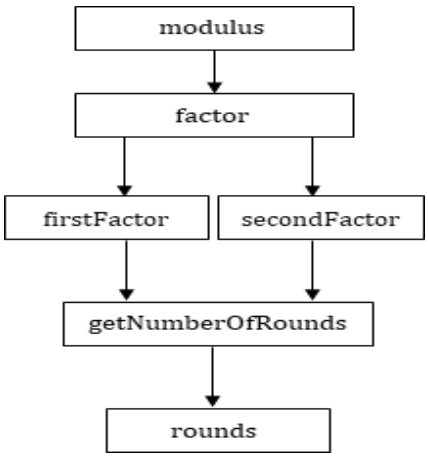


그림 8. 소인수 분해 과정
Fig. 8. Factorization in prime factors process

algorithm $FE1_K^{N,T}(X)$
$(a, b) \leftarrow N;\ X_0 \leftarrow X$
for i = 1,...,r(N) do
  $L_{i-1} \leftarrow X_{i-1}$ div b
  $R_{i-1} \leftarrow X_{i-1}$ mod b
  $W_i \leftarrow (L_{i-1} + F_K(N, T, i, R_{i-1}))$ mod a
  $X_i \leftarrow aR_{i-1} + W_i$
ret $X_{r(N)}$

그림 9. FE1 알고리즘[4]
Fig. 9. FE1 Algorithm[4]

표 7. FE1 알고리즘 선언
Table 7. FE1 Algorithm Declaration

Declaration	Explanation
N	Plaintext length
T	Tweak
K	Key
X	Plaintext
L	Left binary string
R	Right binary string
a	firstFactor
b	secondFactor
W	Left binary string and right string linked values
F	Round function

4.3.3 구현결과

Fig. 10을 통해 기존에 구현된 FE1 알고리즘에서는 설정된 modulus 값보다 입력값이 작게 입력될 경우 형태보존이 되지 않은 암호문이 생성되는 것을 알 수 있다.

상기의 문제점을 수정하여 Fig. 11과 같이 입력값의 길이가 유동적이어도 형태가 보존된 암호문이 생성된다.

```
PlaintText: 1234 EncryptText: 57042812
```

그림 10. 기존에 구현된 FE1 알고리즘의 구현결과
Fig. 10. Result of implementation of original FE1 algorithm

```
1234
PlainText: 1234 EncryptText: 5522
```

그림 11. 수정한 FE1 알고리즘의 구현결과
Fig. 11. Result of implementation of modified FE1 algorithm

표 8. 입력값의 길이에 따른 소요시간 측정
Table 8. Measurement of the time required by length of the input value

Input value	Input Length	Time(s)
12345678912 (Phone number)	11	0.00026
1234567891234 (Resident registration number)	13	0.00027
1234567891234567 (Card number)	16	0.00031

Table 8은 다양한 입력값의 길이로부터 암호화가 적용되는 시간을 측정한 것이다.

테스트 환경은 Table 3을 토대로 진행하였다. Table 8을 통해 암호화하는 평균 시간은 0.00028초가 걸리는 것을 알 수 있다. 또한 평문을 주민등록번호 형태 혹은 카드 번호 형태로 입력하면 형태가 보존되며 암호문이 생성되는 것을 Fig. 12에서 확인할 수 있다.

기존의 공개된 FE1 알고리즘은 평문의 범위값이 고정되어 있기 때문에 해당 길이만큼의 평문만 암호화가 되는 문제점이 있어 모듈러 값을 유동적으로 계산하도록 수정하였다. 이와 같은 알고리즘 수정이 안전성에 미치는 영향에 대한 연구를 향후 진행할 계획이다.

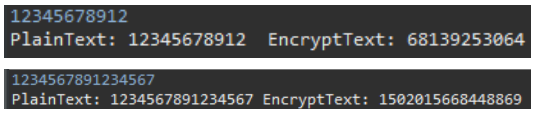


그림 12. 구현 결과
Fig. 12. Implementation Result

V. 결 론

개인 정보의 기밀성을 보존하기 위해 AES나 SEED와 같은 기존의 블록 암호를 사용하는 경우 비트 스트링을 다른 비트 스트링으로 변환하는 과정을 거쳐야 하기 때문에 더 많은 시간을 필요로 한다. 또한 암호문의 길이가 평문의 길이보다 늘어나고 형태가 변경되어 데이터베이스에 정의된 필드 및 자료형과 일치하지 않게 된다. 그에 반해 형태보존암호는 데이터베이스의 구조를 변경하지 않으면서 암호화 기능을 활용할 수 있다. 본 논문에서 사용한 FE1 알고리즘은 Unbalanced Feistel Network 구조를 기반으로 만들어진 형태보존암호 알고리즘이다. 다양한 길이의 메시지를 패딩 없이 암호화할 수 있으며 도메인의 길이에 상관없이 암호화가 가능하다. 그러나 기존에 구현된 FE1 알고리즘은 평문의 범위가 고정값으로 사용되어서 입력한 평문이 전체적으로 암호화가 되지 않는 한계점이 있었다. 이를 해결하고자 평문의 범위를 연산하는 기능을 추가함으로써 평문에 따른 암호문을 계산할 수 있도록 변경하고 평문을 유동적으로 입력할 수 있도록 FE1 알고리즘을 Java로 구현하였다. 향후에 알고리즘 수정에 따른 안전성을 평가할 필요가 있다.

References

[1] Z. Liu, C. Jia, J. Li, and X. Cheng, “Format-preserving encryption for DateTime,” *ICIS*, vol. 2, pp. 201-205, Dec. 2010.

[2] P. Chandrashekar, S. Dara, and V. N. Muralidhara, *IEEE, Efficient format preserving encrypted databases*, pp. 1-4, Jul. 2015.

[3] T. Spies, *Format preserving encryption*(2008), Retrieved Nov. 01, 2018, from www.voltage.com.

[4] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers, “Format-preserving encryption,” *Lecture Notes in Computer Science.*, pp. 295-312, San Diego, USA, Aug. 2009.

[5] L. Ertaul, J. N. Shah, and S. Ammar, “A comparison of HMAC-based and AES-based FFX mode of operation for format-preserving encryption,” *Int. Conf. Secur. and Management*, p. 171, Las Vegas, USA, Jan. 2015.

[6] S. Kelly and S. Frankel, *Using hmac-sha-256, hmac-sha-384, and hmac-sha-512 with ipsec*(2007), Retrieved Nov. 01, 2018, from https://www.rfc-editor.org/rfc/rfc4868.txt.

[7] A. L. Selvakumar and C. S. Ganadhas, *The evaluation report of SHA-256 crypt analysis hash function*, IEEE, pp. 588-592, 2009.

[8] S.-H. Lee and K.-W. Shin, “An area-efficient design of SHA-256 hash processor for IoT security,” *J. KIICE,* vol. 22, no. 1, pp. 109-116, Seoul, Korea, Jan. 2018.

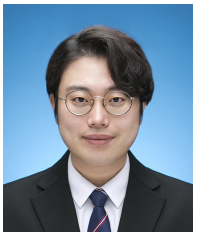
이 창 현 (Changhyun Lee)

2014년 3월~현재 : 순천향대학교 정보보호학과 학사과정

<관심분야> 암호학, 정보보호학

[ORCID:0000-0002-6635-3349]

홍 석 우 (Seokwoo Hong)



2014년 3월~현재 : 순천향대학교 정보보호학과 학사과정

<관심분야> 안드로이드 개발, 데이터베이스

[ORCID:0000-0003-0139-6775]

김 가 을 (Gaeul Kim)

2015년 3월~현재 : 순천향대학교 정보보호학과 학사과정

<관심분야> 암호학, 정보보호학

[ORCID:0000-0003-1847-4549]

이 선 영 (Sun-Young Lee)

1993년 2월 : 부경대학교 전자계산학과(이학사)

1995년 2월 : 부경대학교 전자계산학과(이학석사)

2001년 3월 : 일본동경대학 전자정보공학(공학박사)

2004년 3월~현재 : 순천향대학교 정보보호학과 교수

<관심분야> 콘텐츠 보안, 암호이론, 정보이론, 정보보안

[ORCID:0000-0002-4686-9436]