Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.4, Aug. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.4.753

개인정보암호화에 효율적인 새로운 형태보존암호화 알고리즘

송 경 환,[1†] 강 형 철,[1] 성 재 철[2‡]
[1]고려대학교, [2]서울시립대학교

An Efficient New Format-Preserving Encryption Algorithm to encrypt the Personal Information

Kyung-hwan Song,[1†] Hyung-chul Kang,[1] Jae-chul Sung[2‡]
[1]Korea University, [2]University of Seoul

요 약

최근 금융기관 및 대형 유통업체 등에서 대량의 개인정보유출사고가 연이어 발생하고, 그 피해는 날로 늘어나는 추세에 있다. 이에 따라 개인식별정보를 암호화하도록 강제하는 등 규제가 강화되고 있다. 이러한 개인정보를 암호화하는데 있어서 효율적인 기술이 형태보존암호화이다. 일반적인 암호화방식은 입력 데이터 길이보다 출력 데이터 길이가 확장되며 형태가 변경된다. 형태보존암호화는 입력 데이터의 길이와 형태를 보존해주기 때문에 데이터베이스 및 응용프로그램 수정을 최소화하는 효율적인 방식이다. 본 논문에서는 블록암호 운영모드를 이용한 개인정보 암호화에 효율적인 형태보존암호화방식을 새롭게 제안한다.

ABSTRACT

Recently financial institutions and large retailers have a large amount of personal information leakage accident occurred one after another, and the damage is a trend of increasing day by day. Regulation such as enforcing the encryption of the personal identification information are strengthened. Efficient technology to encrypt personal information is Format-preserving encryption. Typical encryption expand output data length than input data length and change a format. Format Preserving Encryption is an efficient method to minimize database and application modification, because it makes preserve length and format of input data. In this paper, to encrypt personal information efficiently, we propose newly Format Preserving Encryption using Block cipher mode of operation.

Keywords: Format-Preserving Encryption; Modes of Operation; Block cipher; Data format; Data Length

I. 서 론

국내외를 불문하고 대량의 개인정보유출사고가 연이어 발생하고 있고, 이에 따라 개인정보보호법 개정과 같은 강화된 규제와 개인정보보호 문제에 대한 소비자의 인식변화는 개인식별정보를 암호화하고 데이터 손실의 영향을 최소화하려는 많은 기업들에게 동기를 부여했다. 하지만 개인식별정보 암호화 적용의 장벽 중 하나는 암호화 된 정보를 수용하기 위해서 데이터베이스 및 응용프로그램을 수정하는 데 소요되는 비용이다. 이러한 비용은 두 가지 변경과 관련되어 있다. 첫째, 주민등록번호와 같은 개인식별정보는 종종

접수일(2014년 6월 17일), 수정일(2014년 8월 5일), 게재확정일(2014년 8월 5일)
† 주저자, km8944@korea.ac.kr
‡ 교신저자, jcsung@uos.ac.kr(Corresponding author)

데이터베이스의 키 또는 인덱스로 사용되고 있으며, 데이터를 암호화하여 이러한 필드를 랜덤화하는 것은 중요한 스키마 변경이 필요할 수 있다. 둘째, 암호화된 데이터는 길이와 형태가 변형될 수 있다[1].

이러한 요구사항에 효율적으로 적용하기 위해 연구된 암호알고리즘이 형태보존암호화(FPE: Format-Preserving Encryption)이다. 형태보존암호화는 신용카드번호나 개인식별번호와 같은 구조화 된 데이터를 암호화하는 새로운 접근방식이다. 이러한 방식은 평문을 원문과 동일한 길이와 형식을 가지는 암호문으로 암호화하는 데 사용된다. 문자 데이터를 위한 형태보존암호화는 데이터베이스 암호화 및 네트워크 전송에 중요한 데이터 보호와 같은 많은 응용프로그램에 적용할 수 있다. 그것은 확장된 데이터 길이와 형태 변이가 발생하는 전통적인 블록암호에 비해 여러 가지 면에서 장점이 있다. 예를 들어 16자리 신용카드번호가 암호화 된다면 AES에 의해 생성된 암호문은 이진 데이터의 블록이지만 형태보존암호화에 의한 암호문은 또 다른 16자리 숫자일 것이다[2].

본 논문에서 제안하는 효율적인 새로운 형태보존암호화는 기존 암호알고리즘보다 실제 환경에 효율적으로 적용할 수 있도록 보다 나은 성능과 안전성을 제공하기 위해 전처리(Pre-computation)를 할 수 있는 카운터모드 또는 출력피드백모드를 사용하였다.

논문의 2장에서는 블록암호 운영모드, 3장에서는 기 제안된 형태보존암호화기법에 대해 살펴보고 4장에서는 다양한 개인정보암호화에 적용 가능한 효율적인 새로운 형태보존암호화를 제안한다. 또한 5장에서는 제안 알고리즘이 기존 방식과 비교하여 얼마나 효율성을 가지는지 설명하고, 6장에서는 본 논문에 대한 결론을 맺는다.

II. 블록암호 운영모드

암호학에서 운영모드는 기밀성이나 신뢰성 등의 정보서비스를 제공하기 위하여 블록암호를 사용하는 알고리즘이다[3]. DES나 AES와 같은 블록암호는 한 블록의 평문을 암호화하기 위하여 설계되었다. DES의 경우는 64비트, AES의 경우는 128비트의 평문을 암호화하기 때문에 아스키코드를 사용하는 경우 각각 여덟 문자와 열여섯 문자만을 처리할 수 있다. 하지만 실제로 사용되는 평문은 다양한 크기를 가지며 보통 64비트나 128비트보다는 훨씬 큰 데이터이다. 이러한 경우를 처리하기 위해서 필요한 것이 블록암호 운영모드이다. 대표적인 운영모드 다섯 가지는 전자코드북(Electronic CodeBook, ECB), 암호블록체인(Cipher Block Chaining, CBC), 암호피드백(Cipher FeedBack, CFB), 출력피드백(Output FeedBack, OFB), 카운터(Counter, CTR)모드이다. 이 중 효율성 측면에서 출력피드백 및 카운터 모드는 평문 및 암호문이 준비되지 않아도 초기 벡터와 카운터를 이용해 암·복호화에 사용할 키스트림을 미리 준비할 수 있어 전처리가 가능하므로 효율적인 방식이다. 또한 카운터모드는 여러 평문 또는 암호화의 암·복호화를 동시에 수행할 수 있는 병렬처리가 가능하다.

2.1 출력피드백(OFB)모드

출력피드백모드(Fig. 1, 2)는 블록 암호를 동기식 스트림 암호로 변환한다. 배타적논리합(XOR)명령의 대칭성 때문에 암호화와 복호화방식은 완전히 동일하다.

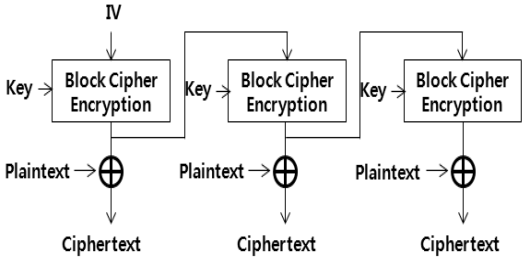

Fig. 1. OFB mode encryption

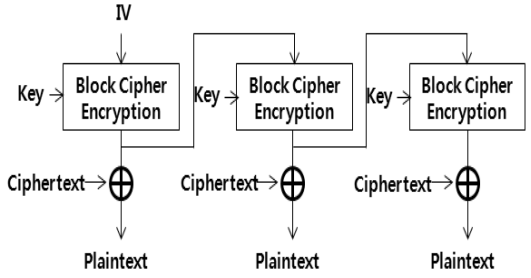

Fig. 2. OFB mode decryption

출력피드백모드의 경우 전처리는 가능하나, 연산이 순차적으로 이루어져야 하므로 병렬처리는 불가능하다.

2.2 카운터(CTR) 모드

카운터모드(Fig. 3, 4)는 블록암호를 스트림암호처럼 사용할 수 있게 한다. 카운터는 각 블록마다 현재 블록이 몇 번째 값인지를 나타내며, Nonce와 결합하여 블록암호의 입력으로 사용한다. 그렇게 각 블록암호에서 연속적인 난수를 얻은 후 암·복호화하려는 문자열과 XOR한다. 카운터모드는 각 블록의 암호화 및 복호화가 이전블록에 의존하지 않으므로 병렬적으로 동작하는 것이 가능하다. 또는, 암호화된 문자열에서 원하는 부분만 복호화하는 것도 가능하다.

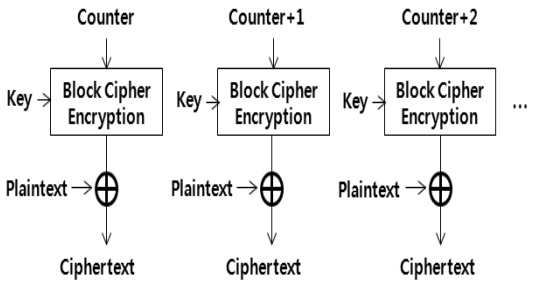

Fig. 3. CTR mode encryption

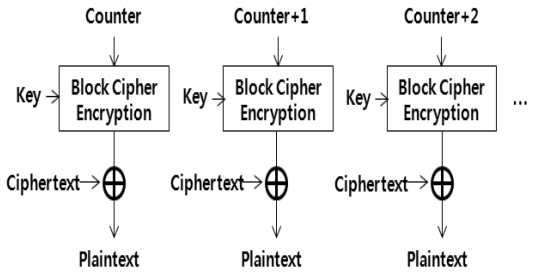

Fig. 4. CTR mode decryption

III. 기존 형태보존암호화 알고리즘

형태보존암호화는 신용카드번호나 사회보장번호와 같은 구조화된 데이터를 암호화하는 새로운 접근법으로서 과거에는 불가능했거나 어려웠던 현재 응용프로그램 프레임웍 상에서 데이터레벨의 암호화를 가능하게 해준다,

이러한 형태보존암호화는 1981년에 처음으로 미국국립표준기술연구소의 전신인 미국표준규격국이 발간한 미국연방정보처리표준74의 부록에서 임의의 알파벳을 임의의 문자열로 암호화하는 접근법으로 표현되었다[4]. 이후 1997년 Brightwell과 Smith이 형태보존암호화 문제를 명확하고 일반화한 논문을 소개하였다[5]. 그들은 데이터형태보존암호화라고 불렀다. Black과 Rogaway가 2002년에 개최된 암호학 커뮤니티에서 이에 대한 정의와 그 해결책을 제공하였다[6]. 그리고 Spies는 현재 사용되는 형태보존암호화라는 용어를 최초로 2003에 사용하였다[1].

미국국립표준기술연구소(NIST)에 기제안된 형태보존암호로는 페이스텔(Feistel)구조를 기반으로 하는 FFX모드와 BPS가 있으며, 블록암호 운영모드를 사용하는 VFPE, CSPEM, FCEM이 있다. 본 장에서는 위의 다섯 가지 형태보존암호화 알고리즘에 대해서 소개한다.

3.1 FFX암호알고리즘

FFX암호알고리즘은 페이스텔(Feistel-based)암호구조를 이용하고, 그 내부함수로 블록암호 AES기반 CBC-MAC[8], CMAC[9], HMAC[10]을 사용한다[7]. 입력값으로는 평문/암호문(알파벳, 숫자, 비트열 등), 비밀키 K, tweak T을 사용한다. 암호화 과정은 아래의 Fig. 5.와 같이 두 가지 방법을 제공한다.

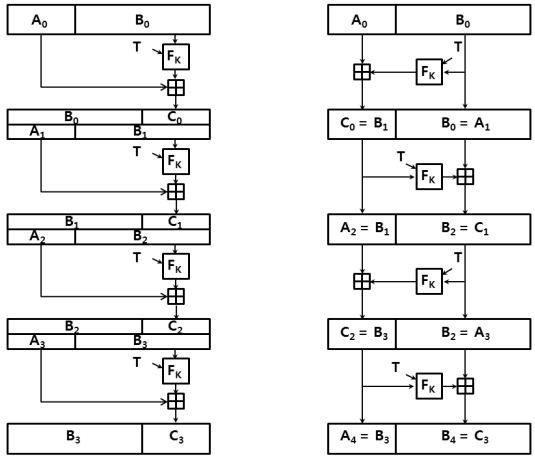

Fig. 5. Illustration of FFX mode encryption when method=1(left) and method=2(right). The first four rounds are shown

먼저 입력된 평문을 임의의 길이로 나눈다. 이때 나누는 방법은 사용자가 결정하면 된다. 즉, 반으로 나

누어도 되고(예. $|A_0|=|B_0|$), 불균등하게 나누어도 된다(예. $|A_0|=2|B_0|$). 방식 1의 경우, 둘로 나눈 평문 중 한 부분을 라운드함수(F_K)로 암호화한 후, 나온 결과값을 나머지 평문의 부분과 모듈러 덧셈한다. 그리고 나온 결과값을 라운드 함수의 입력값과 합쳐서 평문을 나눈 방법과 동일하게 다시 나누어 다음 라운드를 진행한다. 방식 2의 경우, 둘로 나눈 평문을 한 블록씩 라운드 함수로 암호화 한다. 두 방식 모두 한 라운드 함수(F_K)는 모듈러 덧셈이 될 블록의 길이에 맞춰서 결과값을 출력한다.

정해진 암호화 라운드 수는 없지만 저자들은 평문의 길이에 따라 최소한의 필요한 라운드는 제안하였다. 모든 라운드가 끝나면 결과값을 암호문으로 출력한다.

복호화 과정은 위와 동일하나 모듈러 덧셈 대신 모듈러 뺄셈을 사용한다. 더 자세한 내용은 [7]을 참조하라.

3.2 BPS모드

BPS암호알고리즘[11]은 단순하고 효율적이다. 내부함수 초기벡터값이 0인 암호블록체인(CBC)모드와 매우 유사하며, tweak 입력값에 카운터를 결합하였다. 16비트 카운터는 32비트 tweak T_L와 T_R의 왼쪽과 오른쪽 16비트씩 XOR된다. 그러므로 FFX방식이 128비트 이하 이진문자를 입력값으로 하는 반면에 BPS방식은 형태보존암호화 응용프로그램에서 충분히 사용할 수 있는 최대 2^{16}블록까지 입력값으로 사용할 수 있다.

아래의 Fig. 6.은 블록함수를 통해 세자리 십진수 평문을 암호화하는 과정을 보여준다.

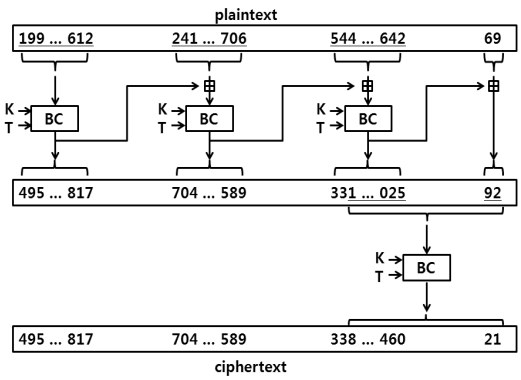

Fig. 6. Example of operating mode encryption process with 10 digits and length 3.

BPS는 페이스텔 기반의 알고리즘으로 내부블록함수로는 TDEA[12], AES[13], 또는 SHA-2[14]를 사용하며, 페이스텔 라운드 수는 8이 권장된다.

3.3 비자형태보존암호화방식(VFPE)

비자형태보존암호화알고리즘(VFPE,VISA Format Preserving Encryption)은 VISA에서 개발한 모듈러-2 덧셈 대신 모듈러-n으로 일반화된 카운터 모드라고 생각하면 쉽다. 비자형태보존암호화는 b비트 크기의 블록, 키값 k 그리고 T1,T2,..의 카운터 블록의 순서를 가진 AES 또는 TDEA블록암호를 이용한 카운터모드를 사용한다[15].

카운터모드로 길이 L의 평문P를 암호화하는 것은 많은 출력 블록들을 생성하는 것이 필요하다. 즉, $L=pk-r$이고, $\frac{L}{k}\le p\le\frac{L}{k}+1$ 과 $0\le r\le k$를 만족하는 유일한 정수 p,r 그리고 출력 블록 $G_1,G_2,...,G_p$을 생성한다. 그리고 각 n자리 평문$P[i]$는 $C[i]$로 암호화하기 위해 출력블록들의 연결 $G_1\parallel G_2\parallel ...\parallel G_p$과 모듈러-n으로 더해진다.

$$Enc: C[i]=(P[i]+(G_1\parallel G_2\parallel ...\parallel G_p)[i])\bmod n$$
$$Dec: P[i]=(C[i]-(G_1\parallel G_2\parallel ...\parallel G_p)[i])\bmod n$$

VFPE는 위와 같이 카운터모드를 이용하며, 입출력값 집합은 숫자 0 ~9 총 10개, 숫자+알파벳 대소문자+특수문자 아스키코드 32~126 총 95개를 대상으로 한다.

3.4 문자집합보존암호화방식(CSPEM)

문자집합보존암호화방식(CSPEM,Character Set Preserving Encryption Mode)은 암호피드백(CFB)모드를 이용하고, 그 내부함수로 블록암호 AES, TDEA 등을 사용한다. 이 제안은 대칭키 블록암호 알고리즘에 대해 허용된 미국연방정보처리표준 74-8의 일반화이다[16]. Fig. 7, 8는 암·복호화 과정을 보여준다.

여기에서 LSB는 Least Significant Bit, MSB는 Most Signicant Bit를 의미한다.

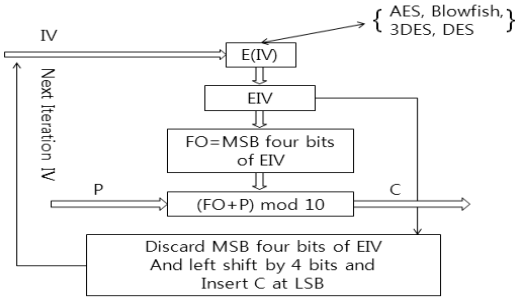

Fig. 7. Encryption Process of CSPEM

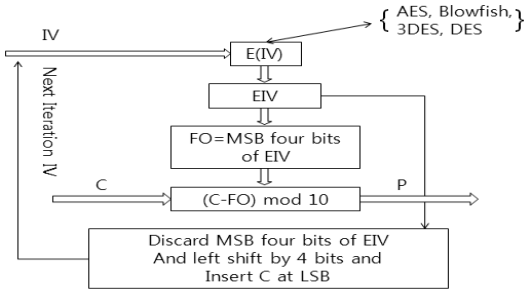

Fig. 8. Decryption Process of CSPEM

3.5 형태제어암호화방식(FCEM)

형태제어암호화방식(FCE, Format-Controlling Encryption)은[17] 데이터형태보존암호화(DTP, Datatype-Preserving Encryption)의 확장개념이다. Table 1는 FCEM에서의 데이터형태보존암호화 단계를 보여준다. 또한, 역순으로 복호화된다.

Table 1. The steps for Datatype-Preserving Encryption

order	step
1	Select algorithm and key
2	Select input and output alphabets
3	Put index values of characters to be encrypted into a work buffer
4	Add position-sensitive offsets according to a data-dependent scheme
5	Shuffle the work buffer according to a data-dependent scheme
6	Ripple data by modularly adding pairwise from left to right then from right to left
7	Select initial value for internal cipher
8	For each index value, encrypt internal buffer, set ciphertext using a modular addition over index value encrypted output, and update internal buffer using plaintext feedback
9	Output character representation of the encrypted index values

IV. 효율적인 새로운 형태보존암호화 알고리즘

4.1 알고리즘 개요

본 논문에서 제안하는 효율적인 형태보존암호화(EFPE, Efficient Format-Preserving Encryption)는 세 가지 입력 형태를 제공하며 블록단위로 나누어 암호화한다. 내부함수로는 블록암호 AES-128를 사용하며, 응용프로그램에서 효율적인 처리를 위해 전처리와 병렬처리가 가능한 카운터모드와 전처리가 가능한 출력피드백모드도 사용한다.

형태보존암호화를 수행하기 위해서 먼저, 암호화하고자 하는 데이터에 맞는 입·출력값의 집합에 대한 정의가 필요하다. 예를 들어 주민등록의 경우 숫자(0~9) 총 10개의 원소를 갖는 집합이 된다. 각 집합이 정의되면 크게 초기 카운터 또는 초기벡터를 내부함수를 통해 암호화시켜 나온 출력 값 중 유효값을 찾는다(Effective Value(EV) search). 그리고 해당 집합에 일대일 변환(set conversion)을 시킨 값과 모듈러 덧셈 또는 뺄셈한다. 이렇게 나온 결과값을 다시 집합에 일대일 변환을 시킨 후, 최종적으로 암호문 또는 평문을 출력한다.

암호화 가능한 입출력값의 집합은 아래와 같다.

- 숫자(0~9): 총 10개의 원소
- 아스키코드 32("space") ~ 126("~")
  : 총 95개의 원소
- 영어 대·소문자(A~Z, a~z), 숫자(0~9)
  : 총 62개의 원소

각 카운터모드와 출력피드백모드로 암호화하는 순서도는 아래 Fig. 9, 10과 같다. 또한 평문과 암호문의 위치 및 모듈러 덧셈을 모듈러 뺄셈으로 바꾸면 동일한 순서를 통해 복호화하는 과정도 설명할 수 있다.

즉, 정의된 입출력값 집합에 맞게 초기 카운터 또는 초기벡터를 내부함수를 통해 암호화시킨 후 유효값 검색과정, 일대일변환을 통한 입출력값 집합변환과정을 거쳐 암복호화를 수행한다.

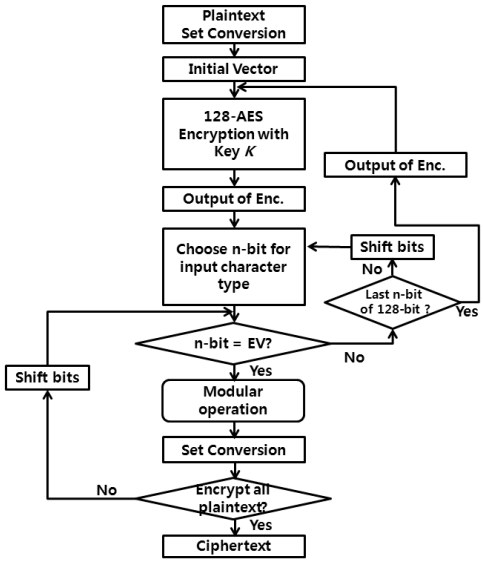

Fig. 9. Flowchart of EFPE Encryption process when operation mode=OFB.

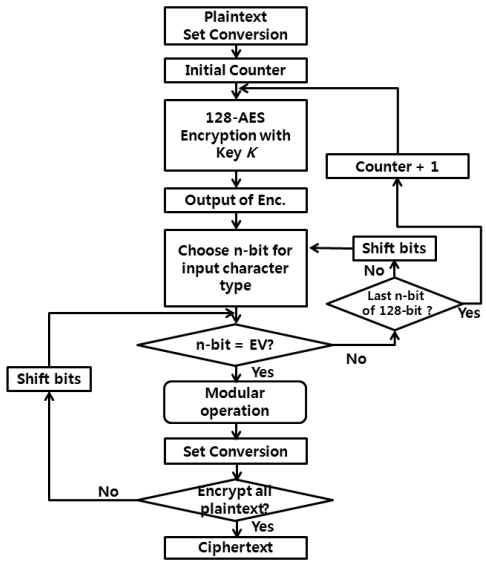

Fig. 10. Flowchart of EFPE Encryption process when operation mode=CTR.

4.2 입출력값 집합 변환과정

아스키코드, 영어 대소문자 및 숫자의 경우 입력값 집합을 모듈러연산을 수행하면 출력값 집합과 달라지게 된다. 따라서 암복호화를 위해서 입력값 집합과 출력값 집합을 같게 해주는 특별한 변환과정이 필요하다.

4.2.1 아스키코드 집합 일대일 변환과정

아스키코드 집합 $X=\{32,33,34,\cdots,126\}$에 대해 어떤 값을 더한 후 모듈러 연산(mod 127)을 수행하면 출력값 집합$X'=\{0,1,\cdots,126\}$ 으로 두 집합이 서로 달라진다. 입·출력값 집합을 동일하게 해주기 위해서는 다음과 같은 일대일 변환 과정이 필요하다.

$$X' = X-32 \tag{1}$$

집합 변환 후 암복호화를 수행하고, 수행한 결과값 Y에 다시 집합변환과정을 거친다.

$$Y' = Y+32 \tag{2}$$

4.2.2 영어 대소문자, 숫자 집합 일대일 변환과정

영어 대소문자와 숫자를 암·복호화하기 위해서는 입력값 집합을 특정한 집합으로 변환해야 한다. 영어 대소문자와 숫자의 집합은 다음과 같이 집합 $X=\{48,49,\cdots,57,65,66,\cdots,90,97,98,\cdots 122\}$이 며 사이에 비어 있는 값(58 ~ 64, 91 ~ 96)이 있다. 그렇기 때문에 아스키코드 집합 변환과정과 같이 동일한 방법으로 수행하면 안되며, 영어 대문자, 영어 소문자, 숫자에 서로 다른 연산을 수행해야 한다.

$$\begin{aligned}&\text{for}\ \forall x\in X, \forall y\in X',\\ &\text{숫자} \quad\quad : y=x-48,\ 48\le x\le 57,\\ &\text{영어 대문자} : y=x-55, 65\le x\le 90,\\ &\text{영어 소문자} : y=x-61, 97\le x\le 122.\end{aligned} \tag{3}$$

집합 변환 후 암복호화를 수행하고, 수행한 결과값을 다시 집합 변환과정을 거친다.

$$\begin{aligned}&\text{for}\ \forall x\in Y', \forall y\in Y,\\ &\text{숫자} \quad\quad : x=y+48,\ 0\le y\le 9,\\ &\text{영어 대문자} : x=y+55, 10\le y\le 35,\\ &\text{영어 소문자} : x=y+61, 36\le y\le 61.\end{aligned} \tag{4}$$

4.3 유효값 검색(EV Search)과정

유효값 검색 과정은 *암호문 균등성(uniform)을 보장하기 위한 것이다. 이 과정은 정의된 입·출력값에

* 정의된 입·출력값에 따라 초기 카운터 또는 초기벡터가 암호화되어 나온 값에서 모듈러 연산을 할 경우 암호문이 균등하지 않다. 예를 들어, 숫자의 경우 00012(1)과 10112(11)의 경우 mod10의 결과가 동일한 값으로 된다. 그러므로 출력값의 경우 0,1,2,3,4,5와 6,7,8,9는 균등하지 않다. 그러므로 암호문 균등성을 보장하기 위해 유효값 검색과정이 필요하다.

따라 초기 카운터 또는 초기벡터가 암호화되어 나온 값에서 모듈러 덧셈 또는 뺄셈에 사용할 유효값을 찾는 것이다. 숫자의 경우, 4-비트단위로 유효값 EV_4을 검색한다. 4-비트는 00002(0) ~ 11112(15)의 범위를 가지며, 모듈러연산(mod 10)에 사용할 유효값은 00002(0) ~ 10012(9)의 값이다. 만약 유효값인 경우, 해당 4-비트를 이용하여 암·복호화를 수행하고 다음 4-비트를 검색한다. 만약 유효값이 아닌 경우, 우측으로 4-비트 이동하여 다음 4-비트에 대해 유효값 검색과정을 수행한다. 출력값 128-비트까지 검색과정이 완료되면 카운터모드의 경우, 카운터를 하나 증가시켜서 다시 암호화하여 유효값 검색과정을 반복한다. 출력피드백모드의 경우에는 전 블록의 출력값을 암호화하여 유효값 검색과정을 반복한다(Fig. 11).

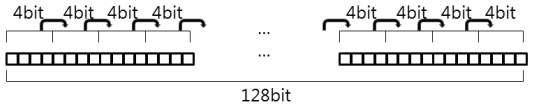

Fig. 11. EV_4 search process

아스키코드의 경우, 7-비트 단위로 유효값 EV_7을 검색한다. 7-비트는 00000002(0) ~ 11111112(127)의 범위를 가지며, 모듈러 연산(mod 95)에 사용할 유효값은 00000002(0) ~ 10111102(94)의 값이다. 아스키코드 암·복호화에 대한 유효값 검색과정은 숫자의 경우와 조금 다르다. 먼저, 출력값 128-비트를 8-비트씩 나누고, 각 8-비트에 대해서 상위 한 비트를 버리고 나머지 7-비트에 대해서 유효값 검색 과정을 수행한다(Fig. 12).

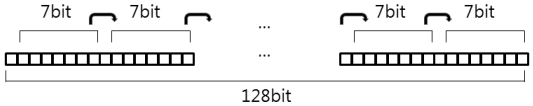

Fig. 12. EV_7 search process

영어 대소문자와 숫자의 경우, 6-비트 단위로 유효값 EV_6을 검색한다. 6-비트는 0000002(0) ~ 1111112(63)의 범위를 가지며, 모듈러 연산(mod 62)에 사용할 유효값은 0000002(0) ~ 1111012(61)의 값이다.

영어 대소문자와 숫자 암·복호화에 대한 유효값 검색과정은 아스키코드 경우와 유사하게 8-비트 중 상위 두 비트를 버리고 수행한다(Fig. 13).

Fig. 13. EV_6 search process

4.4 암·복호화과정

4.4.1 숫자 암·복호화

유효한 값을 찾았을 경우, 1개의 문자(P_i 또는 C_i)와 모듈러 연산(mod 10)하여 암·복호화를 한다.

$$\begin{aligned} &\text{암호화}: C_i = (P_i + EV_4) \bmod 10 \\ &\text{복호화}: P_i = (C_i - EV_4) \bmod 10 \end{aligned} \tag{5}$$

4.4.2 아스키코드 암·복호화

유효한 값을 찾았을 경우, 1개의 문자(P_i 또는 C_i)와 모듈러 연산(mod 95)하여 암·복호화를 한다.

$$\begin{aligned} &\text{암호화}: C_i = (P_i - 32 + EV_7) \bmod 95 + 32 \\ &\text{복호화}: P_i = (C_i - 32 - EV_7) \bmod 95 + 32 \end{aligned} \tag{6}$$

4.4.3 영어 대소문자, 숫자 암·복호화

유효한 값을 찾았을 경우, 1개의 문자(P_i 또는 C_i)와 모듈러 연산(mod 62)하여 암·복호화를 한다.

암호화 과정은 다음과 같다(수식 (7), (8)).

$$\begin{aligned} &\text{숫자}: && Out_i = (P_i - 48 + EV_6) \bmod 62 \\ &\text{영어 대문자}: && Out_i = (P_i - 55 + EV_6) \bmod 62 \\ &\text{영어 소문자}: && Out_i = (P_i - 61 + EV_6) \bmod 62 \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{for } Out_i, \\ &0 \sim 9 \text{ 값일 경우}: C_i = Out_i + 48 \\ &10 \sim 35 \text{ 값일 경우}: C_i = Out_i + 55 \\ &36 \sim 61 \text{ 값일 경우}: C_i = Out_i + 61 \end{aligned} \tag{8}$$

복호화 과정은 다음과 같다(수식 (9), (10)).

$$\begin{aligned} &\text{숫자}: && Out_i = (C_i - 48 - EV_6) \bmod 62 \\ &\text{영어 대문자}: && Out_i = (C_i - 55 - EV_6) \bmod 62 \\ &\text{영어 소문자}: && Out_i = (C_i - 61 - EV_6) \bmod 62 \end{aligned} \tag{9}$$

$$\begin{aligned} &\text{for } Out_i, \\ &0 \sim 9 \text{ 값일 경우}: P_i = Out_i + 48 \\ &10 \sim 35 \text{ 값일 경우}: P_i = Out_i + 55 \\ &36 \sim 61 \text{ 값일 경우}: P_i = Out_i + 61 \end{aligned} \tag{10}$$

4.5 운영모드

4.5.1 카운터모드

카운터(CTR, counter)는 총 128-비트로 상위 n-비트는 데이터베이스의 경우, 평문의 식별번호(ID)로 사용하며, 하위 (128-n)-비트는 내부함수가 암호화를 수행할 때마다 1씩 늘어난다(Fig. 14). 이때, ID Number는 상황에 따라 다른 값을 사용해도 무관하다. 또한, 암호화할 데이터의 길이가 길면 (128-n)-비트를 충분히 크게 사용하여야 안전하다.

ID	Phase 1	Phase 2	...	Phase 2^{64} -1
1	ID CTR 000...001 000...001	ID CTR 000...001 000...010	⋯	ID CTR 000...001 111...111
2	ID CTR 000...010 000...001	ID CTR 000...010 000...001	⋯	ID CTR 000...010 111...111
	⋮	⋮		⋮
2^{64} -1	ID CTR 111...111 000...001	ID CTR 111...111 000...010	⋯	ID CTR 111...111 111...111

Fig. 14. Example of using counter(n=64)

내부함수를 통해 암호화된 값은 유효값 검색과정을 거쳐, 유효값에 대해 모듈러 연산을 수행한다. 최종 128-비트까지 사용되면, 카운터를 증가시켜 내부함수 암호화 과정을 반복 수행한다.

카운터모드를 이용한 EFPE의 암복호화 과정은 Fig.15,16과 같다.

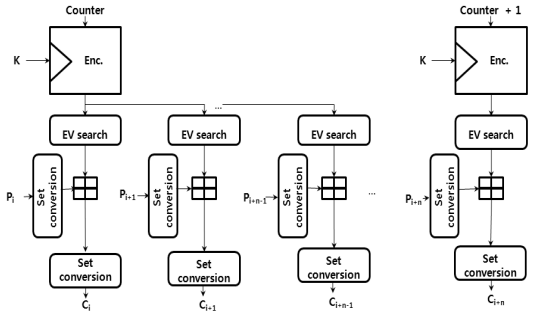

Fig. 15. Encryption Process using CTR mode

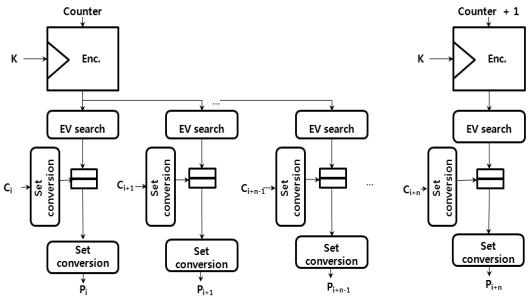

Fig. 16. Decryption Process using CTR mode

4.5.2 출력피드백모드

출력피드백 모드는 초기벡터(Initial Vector)를 내부함수를 이용하여 암호화 한다. 내부함수를 통해 암호화된 값은 유효값 검색 과정을 거쳐 유효값만에 대해 모듈러 연산을 수행한다. 최종 128-비트까지 사용되면, 이전 블록의 출력값을 다음 초기 벡터로 이용하여 내부함수 암호화 과정을 반복 수행한다.

출력피드백 모드를 이용한 EFPE의 암복호화 과정은 Fig.17, 18과 같다.

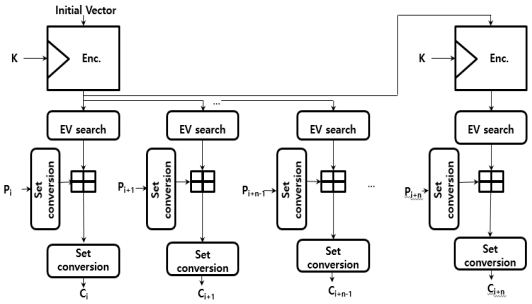

Fig. 17. Encryption Process using OFB mode

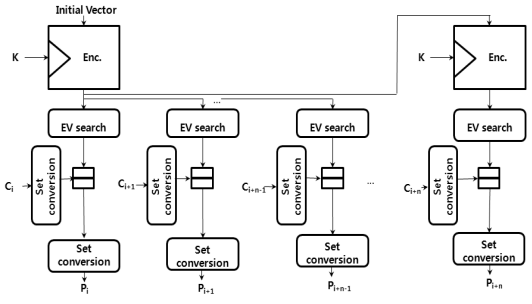

Fig. 18. Decryption Process using OFB mode

V. 제안 형태보존암호화의 효율성 및 안전성 분석

5.1 효율성 분석

암호알고리즘의 효율성은 대부분 내부함수인 AES 암호화 횟수에 비례하며, 이것은 암호화하려는 평문의 자릿수에 비례한다[18].

대표 개인식별번호인 주민등록번호의 경우 4비트 13자리로 구성되어 있다. 따라서 128블록 암호를 이용할 경우 카운터모드나 출력피드백모드의 출력에서 4비트짜리 32개의 출력에서 4비트 짜리 유효한 값-숫자의 경우 4비트 값이 0~9사이인 값을 의미-을 13개를 찾을 수 있으면 1번의 암호화 연산과, 유효값 검사, 13번의 모듈러 덧셈으로 주민번호를 암호화 할 수 있다. 즉 1번의 암호화 연산으로 주민번호를 암호화 할 확률은 약 99.64%이며(수식(11)) 2번 이내의 암호화 연산으로 주민번호를 암호화 할 확률은 약 99.99%(=99.9999999999047%,수식(12))로 충분하다. 그럼에도 불구하고 암호화하지 못했을 경우, 카운터를 증가시키거나 출력피드백모드를 한번 더 연산하는 과정을 반복할 수 있다. 1번의 암호화 연산으로 주민번호를 암호화 할 확률은 다음과 같다.

$$1-\left\{{}_{32}C_0(\frac{10}{16})^0(\frac{6}{16})^{32}+\cdots+{}_{32}C_{12}(\frac{10}{16})^{12}(\frac{6}{16})^{20}\right\} \quad (11)$$

또한, 2번의 암호화 연산으로 암호화 할 확률은 다음과 같다.

$$1-\left\{{}_{64}C_0(\frac{10}{16})^0(\frac{6}{16})^{64}+\cdots+{}_{64}C_{12}(\frac{10}{16})^{12}(\frac{6}{16})^{52}\right\} \quad (12)$$

Table 2는 기 제안된 다섯 가지 형태보존암호화와 EFPE를 블록암호 AES암호화 횟수를 이용하여 효율성을 비교한 것이다.

Table 2. Comparison with number of block cipher AES encryption

algorithm	number of encrytion
FFX [4]	24 AES enc. (= feistel 12rnds × AES CBC-MAC 2block)
BPS [8]	8 AES enc. (= feistel 8rnds)
VFPE [12]	13 AES enc. (by 1-digit enc.) or 7 AES enc. (by 2-digit enc.)
CSPEM [13]	13 AES enc.
FCEM [14]	16 AES enc. + *α(by 1-digit enc.) or 10 AES enc. + *α(by 2-digit enc.)
EFPE [this paper]	**1 AES enc. + EV_4 (99.64%)** **2 AES enc. + EV_4 (99.99%)**

* α: operations in addition to the AES encryption

Table 3은 주민등록번호 13자리를 두 번에 암호화 했을 때(99.99%)를 기준으로 기 제안된 다섯 가지 형태보존암호화의 효율성을 비교한 것이다.

Table 3. Comparison with performance of previous Format Preserving Encryption

algorithm	efficiency	reference
FFX	12	[4]
BPS	4	[8]
VFPE	6.5 or 3.5	[12]
CSPEM	6.5	[13]
FCEM	8 or 5	[14]
EFPE	**1**	**[this paper]**

5.1. 안전성 분석

본 논문에서 제시한 형태보존암호화알고리즘의 안전성은 사용되는 암호알고리즘의 안전성과 카운터모드 혹은 출력피드백모드에 기반하고 있다. 그 이유는 암호화할 때 다른 초기벡터나 카운터가 사용되어 암호화에 사용되는 키수열은 난수적 성질을 가지고 있다. 그리고 입력형태에 따라 키수열을 선택하므로 - 예를 들어 숫자의 경우 4비트 출력값이 0부터 9사이인 값만 선택- 암호문도 균등성과 난수성을 지니고 있다. 또한 본 논문에서 제시된 알고리즘을 공격하는 공격자가 있다면 카운터모드나 출력피드백모드를 공격하는 공격자를 쉽게 구성할 수 있다. 따라서 이와 같은 특성으로 본 논문에서 제안된 새로운 알고리즘의 안전성은 사용되는 블록암호와 카운터모드 혹은 출력피드백모드에 기반 한다고 할 수 있다.

VI. 결 론

본 논문은 개인식별정보를 효율적으로 암호화할 수 있는 효율적인 새로운 형태보존암호화 알고리즘을 제안한다. 제안하는 알고리즘은 내부함수로 블록암호 AES를 사용하며 운용모드로 카운터 모드 및 출력피드백 모드를 사용한다. 그리고 유효값 검색 과정으로 인해 적은 연산량으로도 안전하게 사용할 수 있다.

또한, 제안하는 알고리즘은 13자리 주민등록번호를 두 번에 암호화했을 때(99.99%)를 기준으로 기 제안된 형태보존암호화 알고리즘 중 가장 효율이 좋은 VFPE의 블록암호 라운드 횟수 7회보다 2회로 횟수를 줄여 약 71.4%의 적은 횟수를 요구한다.

따라서 새롭게 제안한 효율적인 형태보존암호화 알고리즘은 데이터베이스 및 응용프로그램의 수정이 제한된 환경에서 개인식별정보 암호화에 효율적이고 안전하게 사용할 수 있다.

References

[1] Terence Spies, "Format Preserving Encryption," Unpublished white paper, www.voltage.com Database and Network Journal, Dec. 2008.

[2] Min Li, Zheli Lie, Jingwei Li, and Chunfu Jia, "Format Preserving Encryption for Character Data," Journal of Networks, vol. 7, no. 8, pp. 1239-1244, Aug. 2012.

[3] NIST Computer Security Division's (CSD) Security Technology Group (STG), "Block cipher modes," Cryptographic Toolkit. NIST, Apr. 2013.

[4] National Bureau of Standards[USA]. FIPS PUB 74, "Guidelines for implementing and using the NBS Data Encryption Standard," 1981.

[5] M. Brightwell and H. Smith, "Using data-type-preserving encryption to enhance data warehouse security," 20th National Information Systems Security Conference Proceedings(NISSC), pp. 141-149, 1997.

[6] J. Black and P. Rogaway, "Cipher with arbitrary finite domains," RSA Data Security Conference, Cryptographer's Track(RSA CT '02). LNCS vol. 2271, pp.114-130, Springer, 2002.

[7] Mihir Bellare, Phillip Rogaway, and Terence Spies, "The FFX Mode of Operation for Format-Preserving Encryption," Unpublished NIST proposal, Feb. 2010.

[8] ISO/IEC9797-1:1999, "Information technology - Security technologies - Messages Authentication Codes(MACs) - Part 1:Mechanisms using a block ciphers," International Organization for Standardization/ International Electrotechnical Commission, 1999.

[9] M. Dworkin, "Recommendation for cipher block modes of operation: the CMAC mode for authentication," National Institute of Standards and Technology. SP800-38B, 2005.

[10] Turner and James M, "The keyed-hash message authentication code(HMAC)," National Institute of Standards and Technology. FIPS 198, 2002.

[11] E. Brier, T. Peyrin, and J. Stern, "BPS: a format-preserving encryption proposal," Ingenico, France, 2010.

[12] National Institute of Standards and Technology. SP800-67: "Recommendation for the Triple Data Encryption Algorithm(TDEA) Block Cipher," May 2004.

[13] National Institute of Standards and Technology. FIPS 197: "Advanced Encryption Standard," Nov. 2001.

[14] National Institute of Standards and Technology. FIPS 180-2: "Secure Hash Standard," Aug. 2002.

[15] John Sheets, Kim R, and Wagner, "VISA Format Preserving Encryption," VISA USA Inc, Oct. 2011.

[16] Gary S. Sarasin, "Character Set Preserving Encryption Mode(CSPEM)," National Institute of Standards and Technology, Nov. 2011.

〔17〕 Ulf T. Mattsson, "Format-Controlling Encryption using Datatype-Preserving Encryption," IACR Cryptology ePrint Archive, 2009.
〔18〕 Michael Scott,"A Note on the Implementation of Format Preserving Encryption Modes," Certivox Labs, http://www.certivox.com/certivox-research/

〈저자소개〉

송 경 환 (Kyung-hwan Song) 정회원
1997년 2월: 고려대학교 수학과 학사졸업
1997년 1월~현재: 국민은행 재직 중
2013년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 정보보호, 금융보안, 시스템 및 네트워크 보안

강 형 철 (Hyung-chul Kang) 학생회원
2010년 2월: 고려대학교 산업시스템정보공학과 학사 졸업
2010년 3월~현재: 고려대학교 정보보호대학원 석박사통합과정
〈관심분야〉 블록 암호화 해쉬 함수 설계 및 분석, 인증 암호화 모드 설계

성 재 철 (Jae-chul Sung) 종신회원
1997년 8월: 고려대학교 수학과 학사 졸업
1999년 8월: 고려대학교 수학과 석사 졸업
2002년 8월: 고려대학교 수학과 박사 졸업
2002년 8월~2004년 1월: 한국정보보호진흥원 선임연구원
2004년 2월~현재: 서울시립대학교 수학과 전임강사, 조교수, 부교수
〈관심분야〉 암호 알고리즘 설계 및 분석