Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 11 pp. 6815-6820, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.11.6815
ISSN 1975-4701 / eISSN 2288-4688

RFID system을 위한 AES 암호프로세서 설계에 관한 연구

강영진[1*]
[1]원광대학교 전자공학과

Study of the Cryptographic Processor Design appropriate for the RFID system

Young-Jin Kang[1*]
[1]**Dept. of Electronic Engineering, Wonkwang University**

요 약 유비쿼터스 활성화에 따른 RFID/USN 환경조성은 매우 빠르게 증가되고 있다. 그러나 RFID/USN 환경에 적합한 보안환경은 보안위협 증가속도에 따르지 못하는 것이 현 실정이다. 그러므로 본 논문은 RFID/USN에 적합한 MSNR을 제안하였다. 제안된 MSNR은 기존 AES에 비하여 1.3배의 처리율 증가를 보였으며, 전체적인 시스템 효율면에서 2배의 성능향상을 보임을 확인하였다. 그러므로 MSNR은 RFID/USN 등과 같은 환경적 자원제약 조건을 극복하기에 적합한 암호알고리즘으로 사료된다.

Abstract The creation of a RFID/USN environment has increased rapidly due to the activation of ubiquitous security environment suitable for RFID/USN environment but has failed to comply with the speed of security threats. Therefore, this thesis presents MSNR suitable for RFID/USN. The MSNR presented showed an increase in the processing rate of 1.3 times compared to the existing AES and showed 2 fold improvement in performance in terms of the overall system efficiency. Therefore, MSNR is considered to be a password algorithm suitable to overcome the conditions of environmental resource conditions, such as RFID/USN.

Key Words : MSNR, RFID, USN, Cryptographic, Processor

1. 서론

현재 RFID/USN의 사용범위가 증대됨에 따라 노드망에 대한 보안의 중요성이 더욱 강조되고 있다. 그러므로 다양한 보안기법들이 RFID/USN에 적용되고 있지만 다양한 공격에 대한 방어기법으로 아직은 부족한 상태이다[1]. 이에 본 논문은 RFID/USN에 사용되는 암호알고리즘의 성능향상을 위하여 2000년도에 표준화된 AES 알고리즘을 더욱 발전시켜 처리속도 증가 및 노드망 보안성을 보다 우수하게 유지하고자 하였다.

제안된 방식은 AES에 대하여 처리율 증가 및 망 관리 편리성을 위한 MSNR(Management of Speed and Network for Rijndael) 기법이며 AES 및 다른 대칭형 암호시스템들과 비교하여 성능분석을 수행하였다.

제안된 MSNR 암호알고리즘은 기존 대칭형 암호알고리즘에 비하여 처리속도가 빠르며 비트 및 바이트 연산을 혼용함과 동시에 한 주기를 N등분하는 PRN(Pseudo-Random Number)을 사용하여 망 관리가 우수하도록 하였다. 그러므로 MSNR은 복잡해져 가는 노드망을 가지는 RFID/USN 환경에 매우 필요한 암호시스템 중의 하나가 될 것으로 사료된다.

본 논문은 원광대학교 2013학년도 교내 연구과제로 수행되었음.
*Corresponding Author : Young-Jin Kang(Wonkwang Univ.)
Tel: +82-63-850-6743 email: yjkang@wonkwang.ac.kr
Received September 30, 2014 Revised November 5, 2014 Accepted November 6, 2014

2. N-등분 PRN을 포함하는 MSNR 암호알고리즘

RFID/USN에 보안을 강화하기 위하여 다양한 기법들이 제시되고 있으나 제한적인 환경이라는 특징 때문에 일반적인 암호알고리즘을 RFID/USN 환경에 적용하기에는 무리가 따른다.

그러므로 본 논문에서는 처리시간, 소비전력, 크기 등을 고려한 구현상의 문제 해결과 노드증가로 인한 보안 자원의 효율성 등을 위하여 MSNR 기법을 제안하였으며 이를 AES인 Rijndael 암호알고리즘에 적용하였다.

제안된 MSNR 기법은 기존 AES와 유사하게 기본연산자로 배타적 논리합을 사용하며 처리 수행단위는 바이트를 사용하였다. 바이트 연산은 처리속도를 매우 높게 수행할 수 있다는 장점과 더불어 역추적이 어렵기 때문에 비도 증가에도 매우 우수한 특성을 가진다.

또한 기존 Feistel과 SPN 구조를 혼용하기 때문에 암/복호화의 동시 수행이 가능하여 Rijndael[2,3] 또는 Serpent[4] 암호알고리즘에서 발생되는 효율 저하가 발생하지 않는다.

이러한 특징은 AES에 대한 충분한 전제조건을 만족함과 동시에 AES 다음 버전에 대한 내용을 제시할 수 있다.

이러한 특징으로 인하여 MSNR 기법이 가미된 암호알고리즘은 실시간 처리 및 비도 그리고 구현상의 문제점, 노드증가로 인한 관리문제 등을 해결할 수 있다.

2.1 MSNR 암호알고리즘

MSNR 암호알고리즘에 사용되는 입/출력/키 블록 크기는 128 비트이며 평문, 암호문 그리고 키의 크기도 1:1:1이 된다.

MSNR 암호알고리즘은 AES 암호알고리즘과 마찬가지로 다음과 같은 네 가지 기능블록을 포함하며 각 단계를 거치는 동안 바이트 단위의 변환으로 구성된 라운드를 이용하여 암/복호화를 수행한다.

ⅰ) Inv/SubByte : 조건 상태(CS : Condition State) 기능을 가진 S-box를 이용하여 바이트 치환 수행 기능

ⅱ) Inv/ShiftDiagonal : CS에 대한 행 방향 이동기능

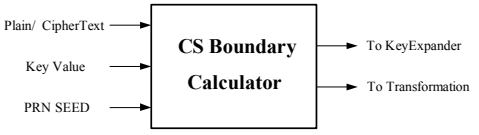
ⅲ) Inv/MixColumn : CS의 각 열에 해당하는 바이트들의 혼합기능

ⅳ) AddRoundKey : CS와 라운드 키에 대한 1:1 덧셈 기능

평문/암호문 128 비트의 입력은 CS 블록으로 초기 저장된다. CS는 MSNR 암호알고리즘을 형성하기 위한 비선형 특성을 가진 상태를 의미한다. CS는 식 (1) 및 Fig. 1과 같이 외부에서 주어지는 파라미터를 가지고 현재상태를 결정하며 결정된 현재상태는 불확실한 미래상태를 형성하는 기준값이 된다.

$$CS_{next} \leftarrow CS_{present}(prn_{seed} \mod 8) \tag{1}$$

CS는 입력으로 사용되는 여러 가지 파라미터를 이용하여 키 값을 생성하는데 필요한 자료를 만드는 동시에 암/복호화하는 과정 중에 치환변환을 제어하는 기능을 수행한다.



[Fig. 1] Characteristic of a condition state

식 (1)에서 mod 8은 입력 데이터들에 대한 모듈러 연산을 의미하는 것으로서 입력값들에 대한 포맷을 바이트로 변환하기 위한 과정이다. 또한 PRN[5] SEED는 식 (2)와 같이 입력 데이터와 키 값을 이용하여 생성한다.

$$SEED = INPUT \bigoplus KEY_{7,15,\cdots,111,119,127} \tag{2}$$

식 (2)에 의하여 생성된 SEED는 식 (3)과 같은 PRN을 통하여 2 바이트의 출력값을 산출하게 된다.

$$PRN = PRN_{odd} \| \ PRN_{even} \tag{3}$$

$$PRN_{odd} = x^{16} + x^{13} + x^{12} + x^{11} + x^{7} + x^{6} + x^{5} + x^{4} + 1$$

$$PRN_{even} = x^{16} + x^{14} + x^{10} + x^{9} + x^{8} + x^{6} + 1$$

산출된 2 바이트 중 odd 정보는 MSNR의 내부 바이트 치환 제어에 사용되어지며, even 정보는 키 확장 제어에 사용된다.

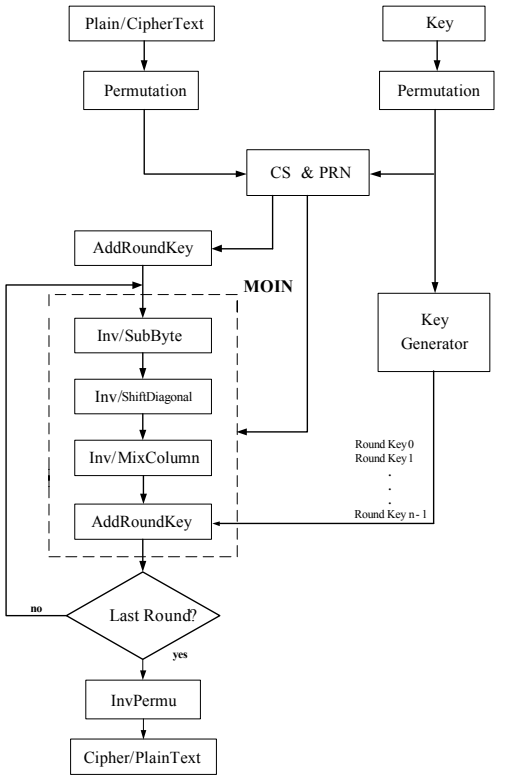
odd와 even 정보는 예측 불가 함수를 생성하는 기능을 가짐으로서 내부 치환정보가 외부로 유출될 가능성이 적으며, 입력 데이터와 키 정보만을 가지고 생성된 것이

기 때문에 별도의 프로세싱이 필요 없다.

식 (3)은 이동통신망에서 사용되는 PRN과 동일한 방정식으로서 2바이트를 하나의 방정식으로 간주하여 출력값을 산출하게 된다.

CS는 초기 라운드 키와 배타적 논리합을 수행한 후 n번의 라운드를 수행하게 된다. 모든 라운드가 실행되면 암/복호화를 마치게 된다. 이러한 MSNR 암호알고리즘에 대한 전체적인 흐름은 Fig. 2와 같다. MSNR 암호알고리즘의 가장 큰 특징은 Fig. 2에서 보는바와 같이 암/복호화가 동시에 수행된다는 점이다. 제어신호에 의하여 암/복호 모드가 결정되며 처리되어지는 연산은 순/역으로 동작한다.

각 라운드마다 Inv/SubByte, Inv/ShiftDiagonal, Inv/MixColumn, Inv/AddRoundKey에 대한 데이터 값들은 CS & PRN에 의하여 별개로 동작하게 된다. 그러므로 라운드 수에 따라서 비도가 결정된다.



[Fig. 2] The algorithm of the MSNR

이러한 네 가지 변환을 MOIN(Minimum Operation for Inverse/Non-inverse)이라고 정의하면 식 (4)와 같다.

기존 AES들은 데이터와 키의 길이를 128, 192, 256 비트들로 가변시키며, 변화하는 길이에 따라 최적화된 라운드 수를 결정하게 된다. 그러므로 기존 AES인 경우 데이터의 블록길이에 따라 라운드 수가 결정된다. 이러한 결과로 인하여, 데이터 심볼 크기를 파악하게 될 경우, 라운드 수를 파악할 수 있으며 라운드 수와 키 및 데이터와의 DC 및 LC에 의하여 크래킹이 가능해진다. 그러나 MSNR인 경우 고정된 블록 및 키 크기를 가지고 있어도 내부적으로 라운드 수에 따라 데이터 내용이 변화되므로 라운드 수를 데이터 심볼 크기만을 가지고 파악할 수 없다는 장점이 있다.

$$
\begin{aligned}
&MOIN_{n-\partial} \\
&\quad \leqq Inv/_{Byte(odd)_n} + Inv/ShiftDiagonal(odd)_n + \\
&\qquad Inv/MixColumn(odd)_n + AddRoundKey(odd)_n \\
&MOIN_{(n-1)-\partial} \\
&\quad \leqq Inv/_{Byte(even)_{n-1}} + Inv/ShiftDiagonal(even)_{n-1} + \\
&\qquad Inv/MixColumn(even)_{n-1} + AddRoundKey(even)_{n-1}
\end{aligned}
\tag{4}
$$

2.1.1 Inv/SubByte 변환

Inv/SubByte 변환은 바이트 단위로 구성된 S-box를 이용하여 CS에 의한 외부상태값들을 각각의 레지스터에 저장하고, 저장된 값들은 각각 독립적으로 존재하는 바이트들을 비선형적으로 변형하여 비선형 변형된 바이트 집합을 생성하게 된다. Inv/SubByte 변환에 사용되는 S-box는 역변환이 가능하며 유한체 GF(2^8)에서 곱에 대한 역이 존재하며 식 (5)와 같이 정의되는 affine 변환을 GF(2^8)에 적용할 수 있는 비선형 변환이 가능한 함수들의 집합이다.

$$
\begin{aligned}
[ab]_i = [ab]_i \bigoplus [ab]_{(i+4)\bmod 8} \bigoplus [ab]_{(i+5)\bmod 8} \bigoplus \\
[ab]_{(i+6)\bmod 8} \bigoplus [ab]_{(i+7)\bmod 8} \bigoplus [ac]_i
\end{aligned}
\tag{5}
$$

여기에서 ab는 CS & PRN의 출력 정보 중 내부 정보를 변환시켜 주는 부분인 a 부분의 비트 블록을 의미하며 $0 \leq i \leq 7$일 때 $[ab]_i$는 각각 독립적으로 분리되어 동작하는 바이트들의 i 번째 해당 비트이고 $[ac]_i$는 특정 ac 바이트 블록의 i 번째 비트를 의미한다. $\bmod 8$은 내부 연산시 바이트 단위로 수행되지만 실제적인 연산은 비트 단위이다. 그러므로 비트와 바이트에 대한 분리 표

현을 위하여 모듈러 연산을 유한체 GF(2^8)상에 나타냄으로서 MSNR 암호알고리즘은 1 바이트 연산이 기준이 된다는 것을 나타낸다. 이 변환을 행렬 형태로 표현하면 식 (6)과 같다. 여기에서 ab'는 변환된 새로운 상태배열을 의미하며 $s_{00} \sim s_{77}$는 S-box에 대한 값이며 PN_0는 랜덤수를, α는 PRN에 대한 SEED값을 의미한다.

2.1.2 Inv/ShiftDiagonal 변환

Inv/ShiftDiagonal 변환은 데이터 상태배열의 행 및 열 단위를 기준으로 대각선 방향으로 동시에 이루어진다. 이러한 변환은 식 (7)과 같은 방정식으로 표현할 수 있다.

$$\begin{bmatrix} ab'_0 \\ ab'_1 \\ ab'_2 \\ ab'_3 \\ ab'_4 \\ ab'_5 \\ ab'_6 \\ ab'_7 \end{bmatrix} = \begin{bmatrix} s_{00} & s_{01} & \cdots & s_{07} \\ s_{10} & s_{01} & \cdots & s_{17} \\ & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdots \\ \cdot & & s_{60} & s_{61} \\ \cdots & s_{67} & s_{70} & s_{61} \\ \cdots & s_{77} & & \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \times \begin{bmatrix} PN_0 \\ PN_1 \\ PN_2 \\ PN_3 \\ PN_4 \\ PN_5 \\ PN_6 \\ PN_7 \end{bmatrix} + \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{bmatrix} \tag{6}$$

$$\begin{aligned} S_a'(03,12,21,30) &= S_a(00,11,22,33) \\ S_b'(13,22,31,00) &= S_b(01,12,23,30) \\ S_c'(23,32,01,10) &= S_c(02,13,20,31) \\ S_d'(33,02,11,20) &= S_d(03,10,21,32) \end{aligned} \tag{7}$$

식 (7)에서 각 상태배열에 대하여 행과 열이 대각을 중심으로 변환됨을 알 수 있다. 이러한 대각변환은 역 치환 및 계산 수행이 용이할 뿐만 아니라 랜덤성을 보장하게 된다. 그러므로 Inv/ShiftDiagonal 변환은 Rijndael 또는 Serpent 암호알고리즘과 같은 계산속도의 1/2배의 특성을 가지며 비도는 2배 증가를 가지게 된다. 식 (7)과 같이 대각방향으로 이동되는 Inv/ShiftDiagonal 변환은 행번호가 0인 첫행 첫열은 3번 0번 shift 되며, 마지막 행 및 열은 0번 3번 이동하게 된다. 이와 같은 대각 변환은 같은 행에 있는 바이트들이 행의 번호가 낮은 위치로 이동하는 결과를 가져오며 행 번호가 낮은 위치의 바이트는 상위행의 위치로 이동하게 된다.

Inv/ShiftDiagonal 변환은 대각 변환을 이용하여 변환되기 때문에 역변환도 동일한 과정으로 변환하게 된다.

2.1.3 Inv/MixColumn 변환

Inv/MixColumn 변환은 고정된 다항식인 $a(x)$를 곱하여 새로운 변환 배열을 생성한다. 이때 전체 변환식은 곱셈연산이 기본이 된다. 즉 변환된 함수 $s'(x)$는 변환 전 함수인 $s(x)$에 대하여 $a(x)$를 곱한 형태를 가지게 된다. 이러한 $s'(x) = a(x) \otimes s(x)$ 곱셈형태는 식 (8)과 같이 표현된다.

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 02 & 01 & 03 & 03 \\ 01 & 03 & 03 & 02 \\ 03 & 03 & 02 & 01 \\ 03 & 02 & 01 & 03 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{8}$$

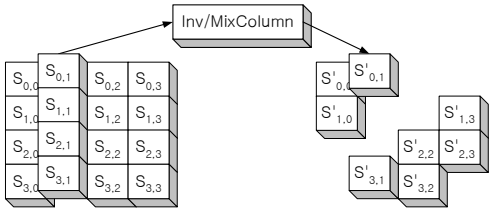
식 (8)의 결과값은 상태 배열에서 네 개의 바이트 열들로 식 (9)와 같이 변환된다.

식 (9)에서 동일한 계수값에 대하여 연속적인 계산을 요구하는 항이 존재하게 된다. 이러한 동일 연산은 배타적 논리합을 이용하면 소거되므로 식 (9)를 다시 정리하면 식 (10)과 같이 변형될 수 있다.

$$\begin{aligned} s'_0 &= (\{02\} \cdot s_0) \oplus (\{01\} \cdot s_1) \oplus (\{03\} \cdot s_2) \oplus (\{03\} \cdot s_3) \\ s'_1 &= (\{01\} \cdot s_0) \oplus (\{03\} \cdot s_1) \oplus (\{03\} \cdot s_2) \oplus (\{02\} \cdot s_3) \\ s'_2 &= (\{03\} \cdot s_0) \oplus (\{03\} \cdot s_1) \oplus (\{02\} \cdot s_2) \oplus (\{01\} \cdot s_3) \\ s'_3 &= (\{03\} \cdot s_0) \oplus (\{02\} \cdot s_1) \oplus (\{01\} \cdot s_2) \oplus (\{03\} \cdot s_3) \end{aligned} \tag{9}$$

$$\begin{aligned} s'_0 &= (\{02\} \cdot s_0) \bigoplus (\{01\} \cdot s_1) \\ s'_1 &= (\{01\} \cdot s_0) \bigoplus (\{02\} \cdot s_3) \\ s'_2 &= (\{02\} \cdot s_2) \bigoplus (\{01\} \cdot s_3) \\ s'_3 &= (\{02\} \cdot s_1) \bigoplus (\{01\} \cdot s_2) \end{aligned} \tag{10}$$

식 (10)을 이용하여 Inv/MixColumn 변환을 수행하면 Fig. 3과 같이 상태배열의 일부만이 변환된다는 것을 알 수 있다. Inv/ShiftDiagonal 변환과 비슷하게 대각변환을 수행하고 있지만, Inv/MixColumn 변환은 $y = x$에 해당하는 대각변환만을 수행하게 된다. 즉, MSNR 암호알고리즘의 대각변환을 수행함에 있어 식 (11)과 같이 대각변환이 구별되어진다.



[Fig. 3] Inv/MixColumn transformation

$$y = -x; \quad Inv/ShiftDiagonal \tag{11}$$

$$y = x; \quad Inv/MixColumn$$

식 (11)과 같이 하나의 상태배열에 대하여 대각 변환을 x방향과 $-x$방향으로 수행함으로서 전체 대각을 변환시킬 수 있다. 이러한 대각변환은 기존 암호알고리즘과 같이 곱셈연산을 수행해야하는 번거로움을 줄일 수 있다. 제안된 MSNR 암호알고리즘은 식 (10)과 같이 다른 값들에 대해서만 곱셈계산을 수행함으로서 수행속도의 단축 및 자리올림 현상이 발생하지 않으며 동시에 반대방향으로 대각변환을 수행함으로서 비도를 향상시킬 수 있다.

2.1.4 AddRoundKey 변환

AddRoundKey 변환은 라운드 키와 상태 배열(CS)을 더하는 연산을 수행한다. 각 라운드 키는 키 스케쥴로부터 천이상태가 발생될 때마다 별개의 독립된 값을 산출하며, 산출된 값들은 식 (12)와 같은 연산을 수행한다.

$$[s'_0, s'_1, s'_2, s'_3] = [s_0, s_1, s_2, s_3] \bigoplus [w_{round} \cdot PN_{round}] \tag{12}$$

여기에서 w_{round}는 라운드 수행에 대한 키 스케쥴 워드이며, PN_{round}는 CS & PRN에 대한 라운드 수행 범위를 의미한다. AddRoundKey 변환은 라운드 범위에 따라서 계산되는 횟수는 다르지만 계산되어지는 양은 동일하게 단순 더하기 기능만을 수행하게 된다.

2.2 MSNR 키 스케쥴링

일반적인 암호알고리즘은 키 생성 알고리즘 및 키 스케줄링 작업을 수행한다. 이러한 키 스케줄링 작업은 보다 복잡한 키를 생성하기 위한 수단으로서 알고리즘의 안전도에 절대적인 역할을 수행한다.

MSNR 암호알고리즘의 경우, 키 스케줄링을 라운드에 따라 변화하는 방법을 사용하지 않고, 단지 CS & PRN 방법을 사용하여 생성한다. 암호문을 생성하고자 하는 평문의 일부를 이용하여 PRN의 SEED로 사용하고 PRN은 각 event가 발생할 때마다 라운드과정으로 인식하여 암호화에 필요한 키를 생성하게 된다. 이러한 키 생성은 각 라운드마다 필요한 별개의 키를 효율적으로 생성할 수 있으며 생성된 키를 이용하여 암호화를 수행할 경우 라운드를 구별하는 기준점을 별도로 설정할 필요성이 없어지게 된다.

키 생성은 PRN의 event 발생 때마다 변화하는 값을 이용하며 내부의 암호문은 MOIN 블록, 즉 네 단계의 변환을 수행할 때마다 변화하는 동시에 PRN의 값이 변환을 조절하게 된다. 이러한 이유로 다른 암호알고리즘에서는 키와 암호문과의 크기 조절을 위하여 키 길이의 확장 및 축소과정을 거치게 되지만, MSNR 암호알고리즘은 이러한 키 길이의 조작을 별도로 수행할 필요성이 없다.

3. MSNR 암호시스템 설계 및 모의실험

MSNR 암호알고리즘의 구현은 VHDL을 이용하여 Top-down 방식으로 진행하였으며 회로합성은 Synopsys Design Analyser Ver. 1999.10, QUARTUS 7.0을 사용하였고, 모의실험에 사용된 툴은 Synopsys VHDL Debegger, ModelSim 5.8C를 사용하였다. 구현을 위한 테스트베드는 ALTERA Cyclone EP1C6Q240C8N 디바이스를 사용하였다.

입력 데이터 128 비트에 대하여 키 값 역시 128 비트이며 키 값 128 비트 중 일부 데이터는 입력데이터와 2진 곱 연산을 수행한 후 PRNG SEED 값으로 사용된다. 암호/복호모드에 따라서 PRNG의 even, odd가 결정되며 결정된 PRNG는 키 정보를 SEED값으로 받아 랜덤한 키 정보를 출력한다. MSNR 암호시스템은 입력 128 비트에 대하여 출력 128 비트를 산출하며 암호문 또는 평문에 대하여 평문 또는 암호문을 출력하게 된다. 암/복호화를 수행하는데 있어 필요한 정보인 키 정보는 실제적으로 암/복호화를 수행할 때는 필요 없으며, 단지 새로운 키 정보를 생성하기 위한 기본 자료로 활용될 뿐이다.

Table 1은 기존 암호시스템과 제안된 MSNR 암호시스템을 상호 비교 분석한 표이다[6-9].

Table 1에서 기존 대칭형 블록 암호시스템에 비하여 MSNR 암호시스템이 처리율면에서 130% 증가됨을 확인하였다. 또한 라운드 횟수와 비도에 의한 암호 효율 측면에서 MSNR 암호시스템은 암호화에 사용되어지는 키 정보가 내부 CS와 PRN에 의하여 생성되며 암/복호화가 동일한 시스템에서 동시에 실행 가능함으로서 기존 블록 암호시스템에 비하여 암/복호측면에서 2배의 효율을 가짐을 알 수 있다. 그러므로 전체적인 시스템 효율은 기존 블록 암호시스템에 비하여 MSNR 암호시스템이 2배의

성능을 가짐을 알 수 있다. 이와 같이 MSNR 암호알고리즘은 기존 블록 암호알고리즘에 비하여 RFID/USN과 같은 자원제약조건을 가진 환경에서 효율적임을 확인할 수 있었다.

[Table 1] Analysis Table for performan of MSNR

@50MHz	structure	No. of Rounds	Length of data (bits)	Length of key (bits)	Ratio (Mbps)
DES	Feistel	16	64	56	31.6
3DES	Feistel	48	64	112/168	15.6
SEED	Feistel	16	128	128	313.7
Serpent	SPN	32	128/192/256	128	197.3
AES	SPN	10	128/192/256	128	387.9
MSNR	Feistel & SPN	10	128	128	532.0

4. 결론

현대사회는 RFID/USN 등과 같은 노드망에 의한 다양한 응용분야의 확산이 매우 빠르다. 그러나 RFID/USN 등은 환경적 자원제약이라는 단점으로 인하여 보안을 유지하기 위한 암호알고리즘을 선택하기 매우 힘들다. 이에 본 논문에서는 이러한 RFID/USN 자원제약에 적합한 암호알고리즘인 MSNR 블록 암호알고리즘을 제안하였다.

제안된 MSNR 암호시스템은 암호화를 수행하는 자원으로서 자체 정보만을 가진다. 또한 처리시간 및 비도는 동시다발적인 연산으로 인하여 기존 대칭형 암호시스템에 비하여 처리율면에서 130% 증가를 가져왔다. 또한 암/복호화를 하나의 시스템으로 처리 가능하므로 전체적인 시스템 효율면에서 2배의 성능을 가짐을 확인하였다. 본 논문에서 제안한 MSNR 암호알고리즘은 기존 암호알고리즘에 비하여 높은 전송률 및 시스템 효율을 가지며, 특정 길이의 키 결정을 수행할 필요가 없는 구조적 기반 알고리즘이기 때문에 시스템 복잡도가 매우 낮고 처리시간이 빠르다. 그러므로 제안된 새로운 MSNR 암호알고리즘은 RFID/USN 등과 같은 환경적 자원제약 조건을 극복하기에 적합한 암호알고리즘으로 사료된다.

References

[1] "information protection in ubiquitous environments", http://www.eic.re.kr, 2008. 11.

[2] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved Cryptanalysis of Rijndael", Seventh Fast Software Encryption Workshop, Springer-Verlag, 2000.

[3] NIST, "AES Algorithm (Rijndael) Information", http://csrc.nist.gov/archive/aes/rijndael

[4] I. Damaj, M. Itani, H. Diab, "Serpent Cryptography on Static and Dynamic Reconfigurable Hardware," aiccsa, pp.680-684, IEEE International Conference on Computer Systems and Applications, 2006.
DOI: http://dx.doi.org/10.1109/AICCSA.2006.205163

[5] NIST, "pseudo-random number generator", http://www.itl.nist.gov/div897/sqg/dads/HTML/pseudorandomNumberGen.html

[6] Microelectronic Systems Laboratory, "Implementation of DES Algorithm Using FPGA Technology", http://www.alagger.com/des-vhdl/report.pdf, 2002.

[7] "DES and 3DES cores", http://www.heliontech.com/des.htm

[8] "SEED block cryptographic algorithm", http://service2.nis.go.kr/pw_certified/seed.jsp

[9] "A candidate block cipher for the advanced encryption standard", http://www.cl.cam.ac.uk/~rja14/serpent.html

강 영 진(Yung-Jin Kang) **[정회원]**

- 1989년 2월 : 건국대학교 대학원 전자공학과 (공학박사)
- 1982년 3월 ~ 현재 : 원광대학교 전자공학과 교수

<관심분야>
초고주파 및 광통신