論文2003-40SD-9-10

인증기능과 자기 키 생성기능을 가진 혼합형 암호시스템 설계에 관한 연구

(A Study of Hybrid Cryptosystem Design with the Authentication and Self-Key Generation)

李 善 根 *, 宋 濟 昊 **, 金 太 亨 ***, 金 煥 溶 *

(Seon Keun Lee, Je Ho Song, Tae Hyung Kim, and Hwan Yong Kim)

요 약

정보통신 및 네트워크의 급격한 발전으로 인하여 정보보호분야의 중요성은 매우 크다. 또한 사용자에 의한 서비스 수요가 증가하면서 개인정보보호에 대한 관심이 증가하였다. 이러한 이유로 대칭형 암호방식보다는 비대칭형 암호방식이 주류를 이루게 되었다. 그러나 비대칭형 암호방식은 대칭형 암호시스템에 비하여 처리속도가 낮아서 적용되는 분야가 한정된다. 그러므로 본 논문에서는 대칭형 암호시스템을 이용하여 비대칭형 암호시스템의 인증기능을 수행할 수 있도록 하는 암호시스템을 설계하였다. 제안된 암호시스템은 블록 암호방식과 스트림 암호방식을 혼용하여 사용하였다. 대칭형/비대칭형 암호시스템은 고정된 키에 의하여 암호/복호를 수행하지만 제안된 암호시스템은 평문의 정보에 의하여 키수열이 변화하도록 함으로써 비도 측면에서 매우 높은 안전성을 가지도록 하였다. 그러므로 비인가자가 키정보를 확보하여도 크랙하기가 매우 어려워진다. 그러므로 제안된 암호시스템은 대칭형 암호시스템의 처리속도와 비대칭 암호시스템의 인증기능을 가지고 있으므로 데이터 인증 또는 중요한 정보의 교환에 매우 유용하게 사용되리라 사료된다.

Abstract

The importance of protection for data and information is increasing by the rapid development of information communication and network. And the concern for protecting private information is also growing due to the increasing demand for lots of services by users. Asymmetric cryptosystem is the mainstream in encryption system rather than symmetric cryptosystem by above reasons. But asymmetric cryptosystem is restricted in applying to application fields by the reason it takes more times to process than symmetric cryptosystem. In this paper, encryption system which executes authentication works of asymmetric cryptosystem by means of symmetric cryptosystem. The proposed cryptosystem uses an algorithms that combines block cipherment with stream cipherment and has a high stability in aspect of secret rate by means of transition of key sequence according to the information of plaintext while symmetric/asymmetric cryptosystem conducts encipherment/ decipherment using a fixed key. Consequently, it is very difficult to crack although unauthenticator acquires the key information. So, the proposed encryption system which has a certification function of asymmetric cryptosystem and a processing time equivalent to symmetric cryptosystems will be highly useful to authorize data or exchange important information.

Keyword : authentication, hybrid cryptrosystem, cryptographic, design

* 正會員, 圓光大學校 電氣電子情報工學部
(Department of Electrical Electronic and Information Engineering Wonkwang University)
** 正會員, 國立益山大學 電氣科
(Department of Electrical Engineering Iksan National College)
*** 正會員, 國立益山大學 電子情報科
(Department of Electronic & Information Engineering Iksan National College)
接受日字:2002年4月23日, 수정완료일:2003年9月2日

Ⅰ. 서 론

급속한 인터넷 및 정보화 사회의 발전으로 인한 정보공유 및 교환은 현대문명에 커다란 영향을 주고 있다. 이러한 정보네트워크 발전과 더불어 단점도 존재한다. 이러한 단점은 네트워크 발전과 더불어 증대되고 있다.

이러한 단점들은 개인정보누출, 사생활 침해 그리고 금융업 등에 대한 정보개방으로 인한 물질적, 정신적 피해가 증대되고 있는 것이 가장 커다란 문제이다. 그러므로 이러한 단점으로부터 피해를 줄이고자 정보보안에 관한 분야는 네트워크 발달과 더불어 병행되어 발전되고 있다. 정보보호분야는 정보보호 알고리즘 개발, 정보보호시스템 개발등과 같이 전문성을 띄며 분야별로 발전되는 것이 현 추세이다. 그러므로 여러종류의 플랫폼에 대한 확장성을 고려하여 정보보호 시스템들은 여러 가지 모양으로 변화한다. <그림 1>과 같이 암호알고리즘은 대칭형(symmetric, secret) 암호방식과 비대칭형(asymmetric, public) 암호방식으로 분류된다.

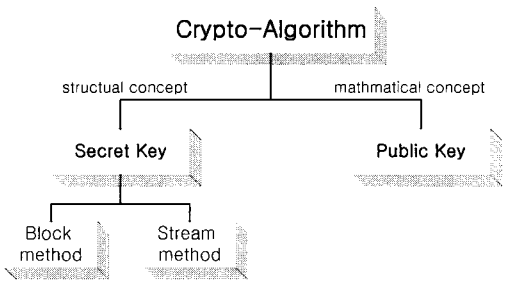


그림 1. 암호 알고리즘의 분류
Fig. 1. Distributary of cryptographic algorithms.

대칭형 방식은 구조적인 메카니즘을 통하여 암호화를 수행하며 비대칭형 방식은 수학적인 해를 얻기가 어려운 문제를 이용하여 암호화를 수행한다. 그러므로 대칭형 암호방식은 암호화 수행속도가 비대칭형 방식에 비하여 매우 빠른 장점을 가진다. 그러나 이러한 구조적 암호화 과정은 사용자의 수가 한정되어 있기 때문에 네트워크상의 정보보호를 위한 메카니즘에는 부적합하다는 것이 단점이다[1, 2].

대칭형 암호방식은 데이터 처리방식에 따라서 블록암호시스템(block cryptosystem), 스트림 암호시스템(stream cryptosystem)으로 분류된다. 블록 암호시스템은 암호화되는 데이터의 처리를 64 비트 또는 128 비트등과 같이 블록단위로 암호화를 수행하며 스트림 암호방식은 비트단위로 데이터를 처리하게 된다. 또한 블록암호 시스템은 Feistel 구조와 SPN(substitution and permutation network)구조를 사용한다.

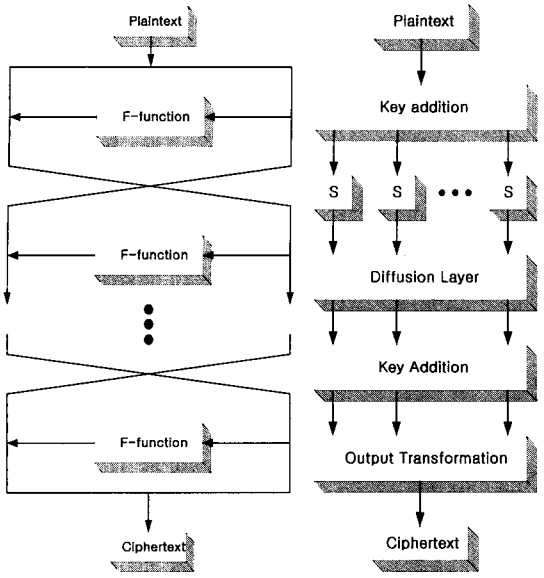
그러므로 본 연구에서는 가장 보편적인 암호화 방식인 블록암호 시스템인 DES를 중심으로 DES의 문제점으로 제시된 키의 길이에 대한 방어와 DC, LC로부터 보다 안전하며 네트워크통신에 적합하도록 하는 혼합형 암호화 알고리즘을 설계함으로써 정보통신 보안분야에 대한 해결책을 제시하고자 한다.

Ⅱ. 대칭형 암호알고리즘

블록 암호알고리즘의 대표적인 DES는 1977년 미 연방 표준으로 채택된 후 20년 이상 거의 세계표준으로 사용되어온 블록암호 알고리즘이다. 그러나 그동안 기술의 발달과 암호학적 분석기법의 발전으로 DES는 중요한 응용분야에서는 더 이상 사용될 수 없을 정도로 안전성을 상실하였다. DES를 대체할 차세대 블록암호를 위해 NIST는 AES(advanced encryption standard) 프로젝트를 진행시켜 Rijndael을 차기 블록암호 알고리즘으로 선정하였다.

기존의 블록암호는 대부분 64비트 블록크기와 56비트의 키를 지원하고 있으나 64비트 블록길이는 블록암호를 이용한 MAC(message authentication code)의 응용등에서 충분한 안전도를 제공하지 못하고 또한 64비트 정도의 키 길이로는 키 전수검색에 대한 충분한 저항성을 제공하지 못한다는 것이 일반화되어 있다. 그래서 AES의 기본 요구조건의 하나는 128비트 블록길이에 128, 192, 256비트 길이의 키를 지원하도록 하는 것이다.

<그림 2>에서와 같이 블록암호는 기본적으로 비선형 변환과 선형변환의 적절한 조합에 의해 설계되며 전체 구조는 크게 DES와 같이 데이터 블록의 좌/우반부에 교대로 비선형 변환을 적용시키는 Feistel 구조와 모든 데이터에 동시에 병렬로 비선형 변환을 적용시키는 SPN 구조로 분류된다. DES, Blowfish, CAST128, LOKI91, MISTY, RC5, CAST256, DFE, E2, MArs, RC6, Twofish등이 Feistel 구조 혹은 그 변형을 바탕으로 설계된 것들이고 SAFER, IDEA, Square, Crypto, Rijndael, SAFER+, Serpent 등이 SPN 구조를 바탕으



(a) Feistel Network (b) SP Network

그림 2. Feistel과 SPN 구조
Fig. 2. Structure of the Feistel and SPN.

로 설계된 알고리즘들이다[3].

블록암호에서 가장 중요한 비선형 변환에는 테이블을 이용하는 치환(substitution : S-box), 곱셈, 가변데이터에 의한 회전(data-dependent rotation) 및 이들과 서로 양립하지 않는 +/-/⊕들의 결합 등이 이용된다. 각 연산에 따라 안전도, 프로세서 또는 하드웨어에 따른 효율성에 있어서 장단점이 있으나 일반적으로 작은 단위의 S-box를 적절히 이용하는 것이 대부분의 플랫폼에서 보다 효율적이다. 블록암호가 다양한 통계적 분석 기법들에 대한 저항성을 갖기 위해서는 비선형 연산 못지 않게 비선형 연산결과를 가능한 많은 다른 데이터 비트들과 섞어주는 효율적인 선형변환을 사용하여야 한다. 실제적으로 가장 강력한 분석기법인 DC 및 LC 암호분석에서의 성공확률은 상당부분 얼마나 선형변환을 사용하느냐에 의존하기 때문이다. 가장 널리 사용되는 선형변환은 bit permutation, PHT(pseudo-hadamard transform), MDS matrix multiplication 등이다. 그러므로 이와 같은 이용 가능한 다양한 연산들을 적절히 결합하여 안전하면서도 효율성을 높일 수 있는 알고리즘을 설계하는 것이 암호 설계자의 목표이다. 블록암호에 대한 가장 단순하며 무식한 공격방법은 하나의 평문과 암호문쌍에 대해 모든 가능한 킷값으로 주어진 평문을 암호화하여 주어진 암호문이 나오는지를 검사하는 전수검사(exhaustive key search)이다. 이러한 전수검사방법은 기술의 발전에 따른 키 길이를 결정하는 기준이 된다. 한편 짧은 키 길이는 birthday paradox를 이용한 key collision attack에도 취약성을 보여준다. 이는 같은 평문을 서로 다른 키로 자주 암호화하는 경우에 적용되는 것으로 그 실현 가능성은 희박하나 키 길이에 따른 암호 알고리즘의 이론적인 강도를 주는 척도가 될 수 있다. 즉, 키 길이가 k비트일 때 같은 평문 P를 약 $2^{k/2}$개의 서로 다른 키 K_i로 암호화한 암호문 $C_i = E_{K_i}(P)$에 대해 (C_i, K_i)쌍을 유지한다면 임의의 키 K로 P를 암호화한 암호문 C가 추어졌을 때 birthday paradox에 의하면 이 C는 매우 높은 확률로 C_i중에 하나가 될 것이므로 C_i에 대응되는 K_i가 올바른 키 K가 될 것이다[4-6].

블록암호 알고리즘에 대한 암호분석 및 해석법은 평문과 암호문사이의 통계적 특성을 이용하여 키를 찾아내는 통계해석법(statistical cryptanalysis : SC)과 알고리즘의 대수적인 성질을 이용하는 대수해석법(algebric cryptanalysis : AC)으로 분류된다[8, 9].

스트림 암호 시스템은 일반적으로 두가지 형태로 분류된다. 첫째는 키 비트 스트림(key bit stream)이 평문(plaintext)과 독립적인 것이고 둘째는 키 비트 스트림이 평문 또는 암호문(ciphertext)에 대하여 종속적인 함수로 구성되어 있는 것이다. 최근까지는 스트림 암호 시스템은 군사 및 외교용으로 범용되어 사용되고 있었으나 정보통신망의 발달과 연계서비스(roaming service)로 인하여 상업용으로도 각광을 받으며 사용되고 있다. 선형궤환치환레지스터(linear feedback shift register : LFSR)에 의해 만들어지는 부호수열인 키 비트 스트림은 암호문을 만들기 위해 평문과 2-모듈러 합을 수행하게된다. 스트림 암호 시스템에서 LFSR의 역할은 암호 키를 생성하는 부호기(encoder)로 사용되기도 하고 LFSR의 입력이 평문인 경우 출력은 직접 암호문으로 구성해주는 기능도 수행한다. 일반적으로 LFSR은 쉽게 구현될 수 있고 또한 비교적 저렴한 구현비용으로 인하여 상용 스트림 암호 시스템에 널리 사용된다. LFSR에서 생성되는 키 스트림이 적어도 의사난수 계열(pseudo-random number sequence)이 되도록 하는 이유는 해커(hacker)가 스트림 암호 알고리즘에 대해 통계적 공격(statistical attack)을 어렵게 하기 위함이다. 이러한 스트림 암호방식은 비트단위(bit by

표 1. 블록암호시스템과 스트림 암호시스템 비교

Table 1. Comparison of the block and stream cipher system.

	블록암호시스템	스트림암호시스템
연산과정 (processing)	고정길이 데이터 블록 블록간 종속적 암호화	키 비트 수열 순차적 비트 2원 연산
종속성 (dependence)	종속관계를 갖는 평문 종속관계를 갖는 키 평문과 키의 복합함수	스트림 단위로 암호화 개별성을 유지
초기조건 (initial condition)	×	○
오류전파 (error propagation)	○	×
기능 (operation)	오류검출, 인증	단순 암호화

bit)로 암호화를 수행하며 기본연산은 배타적 논리합과 지연소자이다. 그러므로 LFSR은 의사난수와 동일하다. 의사난수(pseudo random number : PRN)는 비밀키 암호 시스템의 세션키(session key) 또는 초기값(initialize vector : IV) 생성, 비대칭형 암호나 디지털 서명의 공개키/비밀키 및 시스템 변수의 생성, 보안 통신프로토콜에서 사용되는 Random Challenge 등 각종 암호 시스템 전반에 걸쳐 사용되고 있다. 대부분의 암호시스템에서 사용되는 의사난수 생성방법은 초기입력으로 사용되는 seed 값을 비밀키 암호알고리즘이나 일방향 해쉬 알고리즘(one-way hash algorithm)에 적용하여 생성하고 있다. 비밀키 암호 알고리즘이나 일방향 해쉬 알고리즘이 랜덤한 값을 생성한다고 가정할 때, 의사난수에 대한 안전성은 입력으로 사용되는 seed 값에 대한 엔트로피(entropy)와 seed 및 state 정보가 비인가자로부터 안전한 보관 및 유지에 달려 있다. 즉 LFSR의 초기값 및 키의 안전한 관리가 필요하다. 또한 스트림 암호시스템은 데이터 처리방식이 DES와 같은 블록암호시스템과 비교하여 볼 때 비트 단위로 데이터를 처리하므로 전송도중 발생되는 암호문의 오류는 복원된 평문과 비교하여 볼 때 해당비트만 오류가 검출된다. 즉 오류의 확산이 없다는 것이다[7, 10, 11].

블록 암호와 스트림 암호의 차이점은 <표 1>과 같다.

Ⅲ. 제안된 혼합형 암호알골리즘

비밀키 암호시스템은 크게 데이터 암호부분과 키 스케쥴부분으로 분류된다. 데이터 암호부분에서 사용되는 기법은 Feistel 구조 또는 SPN 구조를 사용한다. 또한 데이터의 처리를 어떠한 포맷으로 하느냐에 따라서 스트림 암호방식과 블록 암호방식으로 분류된다. 블록 암호방식과 스트림 암호방식의 가장 큰 차이점은 반복성에 있다. 블록 암호방식은 한 라운드에 대하여 기본 안전성을 획득하기까지 일정 횟수의 iteration을 수행하게 된다. 스트림 암호방식은 기존 안전성을 획득하기 위하여 무한수열에 가까운 LFSR의 주기특성을 가지도록 한다. 그러나 iteration의 증가 또는 LFSR 주기길이의 증가는 비밀키 암호시스템의 효율을 저하시키는 요인이 된다. 그러므로 본 논문에서 개발한 혼합형 암호시스템은 iteration을 사용하지 않고 단일 라운드를 사용하며 LFSR의 stage 길이를 일정한 크기로 설정하며 설정된 LFSR을 다수개 사용하여 실제적으로는 매우 긴 주기수열을 생성할 수 있도록 하였다.

제안된 혼합형 암호시스템은 데이터 재배열(permutation), 치환(substitution), 데이터 암호블록, 키 스케쥴(key schedule)로 구성되어 있다. 데이터 암호화 과정은 128 비트 평문블록을 64 비트씩 2개의 블록으로 분할하고 확장재배열(expansion)을 거친 후 80 비트의 크기를 가지는 혼합형 키(hybrid key : HK)를 사용하여 암호화한다. 기존 블록 암호시스템인 경우 내부적으로 16라운드(16-round)의 암호화 과정을 거치고, 복호시에도 암호화에 사용된 동일한 키를 역순으로 사용하여 16라운드의 복호화 과정을 수행한다. 그러나 제안

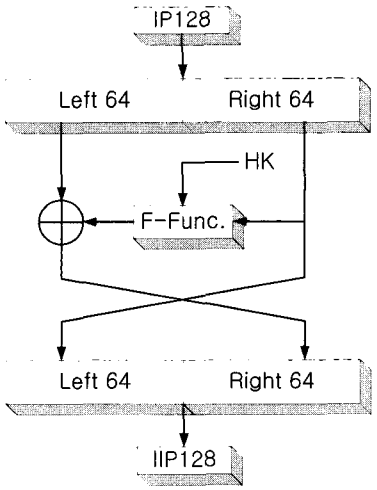


그림 3. 제안된 Feistel 구조

Fig. 3. Proposed Feistel structure.

된 혼합형 암호화 시스템은 단일 라운드만을 사용하여 기존의 16 라운드에 해당하는 비도(security level)를 얻기 위하여 혼합형 키와 블록 암호시스템의 비선형 부분인 F 암호함수부분을 보다 더 비선형화 시켰다.

제안된 혼합형 암호시스템은 <그림 3>과 같이 기존 Feistel 구조와 거의 동일한 형태를 취한다. 식 (1)과 같이 128 비트에 대하여 초기치환(initial permutation : IP)을 수행하고 IP 수행결과를 F 암호함수와 혼합 키(HK)를 이용하여 내부연산을 수행한 후 교번하여 출력을 내보낸다.

$$\begin{aligned} L_1 &= R_0 \\ R_1 &= L_0 \oplus f(R_0, HK) \end{aligned} \tag{1}$$

여기에서 L_1은 다음 stage의 왼쪽 데이터블록이며 R_0는 첫 stage의 오른쪽 데이터블록이다.

식 (1)은 기존 Feistel 구조와 동일하지만 iteration에 관한 i 파라미터가 없고, 대신 이전과 이후에 관한 방정식만 존재한다. 여기에서 입력 128 비트는 IP를 거친 후 L/R로 좌우 64비트씩 분리된다.

각각 L/R의 좌우로 분리된 64 비트는 <그림 3>과 같이 오른쪽 64 비트는 왼쪽으로 이동하며 왼쪽 64 비트는 혼합형 키와 F 암호함수의 연산과정 후 XOR 연산을 통하여 오른쪽으로 이동하게된다. 이러한 연산을 수행한 후 마지막으로 역 초기치환(inverse initial permutation : IIP)을 수행한 후 암호화된 데이터를 출력하게된다.

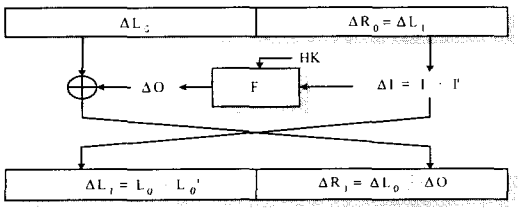


그림 4. 단일 라운드에 대한 Feistel 구조 특성
Fig. 4. Characteristic of Feistel structure with the single round.

<그림 4>는 <그림 3>에 대하여 단일 라운드 사용을 위한 Feistel 구조를 보여준다. 또한 식 (1)은 단일 라운드에 대한 수식적인 표현이다. 일반 블록 암호시스템인 경우 iteration을 최소 16회 정도 반복 수행하지만 본 논문에서 사용된 Feistel 구조는 단지 1회의 라운드에 대해서만 수행하도록 한다. 이러한 이유는 F 암호함수에 있다. 즉 F 암호함수는 일반적으로 사용되는 키 스케쥴에 의해 발생된 키를 사용하는 것이 아니고 단순 키 기능과 인증 및 비대칭형 암호 개념을 가진 킷값을 사용하여 연산을 수행함으로써 비도를 보다 더 높일 수 있도록 하였기 때문이다.

암호함수 F의 비도 결정은 확장재배열과 재배열, 치환에 결정적으로 영향을 받으며 이러한 확장, 재배열, 치환은 미리 설정된 표에 의하여 결정된다.

혼합형 암호방식에서 다양한 통계적 분석기법들에 대한 저항성을 갖기 위하여 비선형 연산 못지 않게 비선형 연산결과를 가능한 많은 다른 데이터 비트들과 섞어주는 효율적인 선형변환을 사용하였다. 이와 같이 선형과 비선형 연산을 섞어 사용하는 이유는 암호 해석기법인 DC 및 LC 암호분석에서의 성공확률은 상당부분 얼마나 선형변환을 사용하느냐에 의존하기 때문이다. 그러므로 제안된 혼합형 암호시스템에서는 가장 널리 사용되는 선형변환들 중 bit permutation개념과 본 논문에서 제안한 matrix 개념을 도입하였다.

<그림 5>는 혼합형 키를 사용하여 비선형 함수 및 선형함수를 생성하는 블록으로써 기능블록은 다음과 같다. E 함수는 확장재배열기능으로써 키와 데이터간의 비트수를 맞추어주며 비도를 증가시키기 위하여 사용되는 선형 암호블록이다. 혼합형 키인 hybrid key는 키 스케쥴러에 의하여 80 비트의 크기를 가진다. 또한 Data_in은 입력 정보 128 비트의 양분한 크기이므로 64 비트의 크기를 가진 데이터들이다. 혼합형 키와 데이터에 대한 모듈러 합을 수행하기 위하여 먼저 입력데이터 64 비트를 80 비트로 확장할 필요성이 있다. 이와 같이 64 비트를 80 비트의 데이터로 변환시켜주는 부분이 80 E 함수이다.

80 E 함수에서는 16 비트의 데이터가 2번씩 반복되어 연산에 참여한다. 그러므로 64 비트들에 대해서 16 비트가 첨가되므로 전체 80 비트의 데이터값들이 산출되게된다. 80 C 함수는 압축(compression)을 수행하는 기능블록으로써 <그림 5>에서와 같이 혼합형 키 80 비트와 80 비트로 확장된 입력데이터들끼리 bit by bit XOR 연산을 수행한 후 유입되어 S 함수로 유출시켜주는 기능을 수행한다. S 함수는 6 비트씩 모두 12개의 하부 S 함수로 구성되어 있기 때문에 6×12=72 비트의 값으로 재조정을 수행해야 한다.

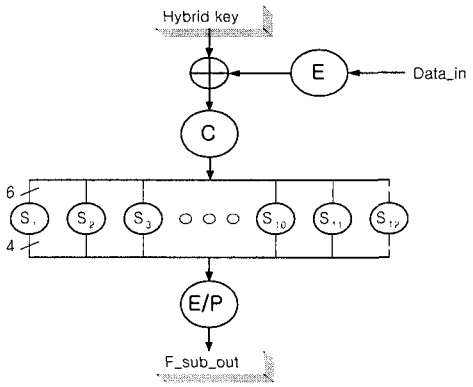


그림 5. F 암호함수
Fig. 5. F crypto-function.

80 비트 입력을 받아서 72 비트 크기의 값을 산출하기 위하여 미리 설정된 표에 의하여 압축과정을 수행하게 된다. 9, 19, 29, 39, 49, 59, 69, 79번째에 해당하는 데이터들은 소거시켜 72 비트의 데이터값을 얻는다. 비선형 S 함수는 <그림 5>에서와 같이 암호함수 F 내부에서 비선형 함수를 발생시키는 기능을 수행하게 된다. S 함수의 입력은 72 비트이고 출력은 48 비트의 크기를 가진다. S 함수 내부에는 입력 6 비트, 출력 4비트를 산출하는 하부 S 함수가 12개 존재하며 12개 하부 S 함수의 조합이 전체 S 함수를 구성하게된다.

$$S = \sum_{i=1}^{12} S_i \tag{2}$$

식 (2)는 비선형 S 함수를 표현한 것으로서 하부 S 함수가 12개로 구성되어 있다. 본 논문에서 사용된 S 함수는 기존의 S 함수와 기존 S 함수의 짝수에 해당하는 S 함수를 사용하게 되는데 S 함수의 내용을 변화시키지 않은 것은 기존 S 함수가 가지고 있는 비선형 특성을 유지하면서 더욱 비도를 증가시키며 S 함수에 대한 자원의 재사용을 위함이다. S 함수를 결정하기 위해서는 S 함수의 입출력이 특별한 관계를 가지도록 구성해야 한다. 블록 암호시스템의 가장 핵심은 S 함수에 있기 때문에 S 함수의 비선형성을 매우 중요하게 생각해야할 파라미터이다.

본 논문에서 제안한 혼합형 암호화 시스템은 *iteration* 이 없고 단지 1회의 라운드만을 수행하기 때문에 기존의 블록 암호시스템 S 함수의 설계방법과는 약간 다른점이 있다. 즉 S 함수에 대한 재배열관계를 무시하여도 S 함수에 대한 비선형특성이 변화되지 않는다는 것이다. 또한 기존 S 함수는 라운드에 대한 iteration을 수행할수록 비선형성이 증가하도록 되어있지만 혼합형 암호시스템에서는 iteration 이 없이 단지 1회 연산만을 이용하여 암호화를 수행하기 때문에 하부 S 함수의 개수를 4개 더 증가시켜 설계하였다. E/P 함수는 확장과 치환(expansion and permutation) 기능을 수행하는 블록이다. S 함수의 출력 48 비트에 대하여 F 함수 64 비트로 변환하기 위하여 확장표를 사용한다.

데이터 암호블록은 블록 암호시스템으로 구성되어 있으며 키 스케줄러는 스트림 암호시스템으로 구성되어 있다.

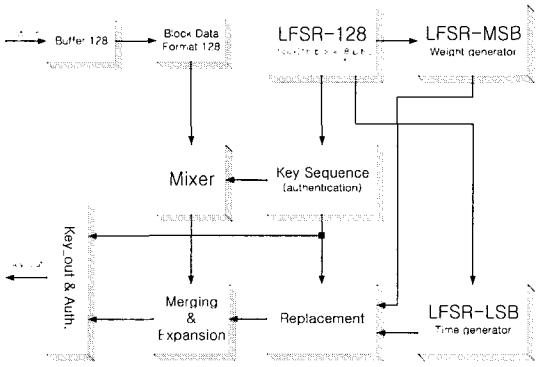


그림 6. 스트림 암호시스템의 키 스케줄러
Fig. 6. Key schedule of the stream cryptosystem.

<그림 6>은 스트림 암호방식을 적용한 키 스케줄러 전체블록으로써 Buffer 128, Block data format 128, Mixer, LFSR-128, Weight generator, Time generator, Key sequence, Replacement, Merging & Expansion로 구성되어 있다.

키 스케줄러의 입력으로는 키 데이터가 별도로 존재하는 것이 아니고 암호화하고자 하는 데이터가 키 스케줄러의 입력이 된다. 그러므로 <그림 7>에서와 같이 Mixer 입력은 128 비트의 입력데이터와 LFSR로부터 생성된 키 수열이 된다. 이 두 가지의 데이터는 16 비트씩 8개의 블록으로 분리되며 분리된 16 비트 8 블록은 각각 XOR 연산을 수행하게된다. 동시에 16 비트의 데이터 8 블록들이 XOR 연산을 수행한 후 Matrix permutation을 수행한 후 Merging & Expansion 블록

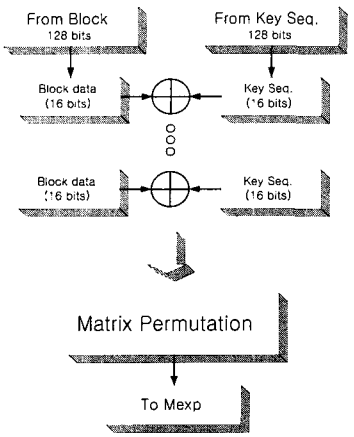


그림 7. Mixer 블록
Fig. 7. Mixer block.

의 입력으로 사용된다.

Matrix permutation은 16 비트씩 8 블록들이 독립적으로 XOR 연산을 수행하게 되며 수행된 결과도 역시 16 비트 8 블록으로 구성된다. 이때 행 데이터들에 대한 연산결과는 열 데이터의 형태로 출력되어진다. 입력 데이터의 집합을 I, 키 수열의 집합을 K라고 하였을 경우 I, K에 대한 표현은 식 (3)과 같다.

$$\begin{aligned} I &= \{i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7\} \\ K &= \{k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7\} \end{aligned} \tag{3}$$

식 (3)에서 i 와 k는 각각 16 비트씩으로 구성된 스트림 데이터들이다. XOR 연산을 수행한 결과를 M이라 하였을 경우 XOR 연산을 수행한 입력 데이터들은 식 (4)와 같이 표현된다.

$$M = I \oplus K \tag{4}$$

식 (4)에서 M 역시 16 비트 8 블록이므로 식 (5)와 같은 수열을 가진다.

$$M = \{m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7\} \tag{5}$$

그러므로 식 (4)에 식 (3)을 대입하고 식 (4)의 연산 수행을 행렬로 표현하면

$$M = \begin{bmatrix} ik_{00} & ik_{01} & ik_{02} & ik_{03} & ik_{04} & ik_{05} & ik_{06} & ik_{07} \\ ik_{10} & ik_{11} & ik_{12} & ik_{13} & ik_{14} & ik_{15} & ik_{16} & ik_{17} \\ ik_{20} & ik_{21} & ik_{22} & ik_{23} & ik_{24} & ik_{25} & ik_{26} & ik_{27} \\ ik_{30} & ik_{31} & ik_{32} & ik_{33} & ik_{34} & ik_{35} & ik_{36} & ik_{37} \\ ik_{40} & ik_{41} & ik_{42} & ik_{43} & ik_{44} & ik_{45} & ik_{46} & ik_{47} \\ ik_{50} & ik_{51} & ik_{52} & ik_{53} & ik_{54} & ik_{55} & ik_{56} & ik_{57} \\ ik_{60} & ik_{61} & ik_{62} & ik_{63} & ik_{64} & ik_{65} & ik_{66} & ik_{67} \\ ik_{70} & ik_{71} & ik_{72} & ik_{73} & ik_{74} & ik_{75} & ik_{76} & ik_{77} \end{bmatrix} \tag{6}$$

여기에서 ik_{32}는 $i_{32} \oplus k_{32}$의 연산결과임을 나타낸다. 식 (6)은 Mixer블록의 matrix permutation의 입력으로 사용된다. 그러므로 식 (6)에 대한 매트릭스 치환은 식 (7)과 같은 결과를 가진다.

$$M^T = MP = \begin{bmatrix} ik_{00} & ik_{10} & ik_{20} & ik_{30} & ik_{40} & ik_{50} & ik_{60} & ik_{70} \\ ik_{01} & ik_{11} & ik_{21} & ik_{31} & ik_{41} & ik_{51} & ik_{61} & ik_{71} \\ ik_{02} & ik_{12} & ik_{22} & ik_{32} & ik_{42} & ik_{52} & ik_{62} & ik_{72} \\ ik_{03} & ik_{13} & ik_{23} & ik_{33} & ik_{43} & ik_{53} & ik_{63} & ik_{73} \\ ik_{04} & ik_{14} & ik_{24} & ik_{34} & ik_{44} & ik_{54} & ik_{64} & ik_{74} \\ ik_{05} & ik_{15} & ik_{25} & ik_{35} & ik_{45} & ik_{55} & ik_{65} & ik_{75} \\ ik_{06} & ik_{16} & ik_{26} & ik_{36} & ik_{46} & ik_{56} & ik_{66} & ik_{76} \\ ik_{07} & ik_{17} & ik_{27} & ik_{37} & ik_{47} & ik_{57} & ik_{67} & ik_{77} \end{bmatrix} \tag{7}$$

이상과 같은 연산을 통하여 생성된 64 비트는 Merging & Expansion 모듈의 입력으로 사용된다.

8단 LFSR의 초기상태가 모두 '0'일 경우 LFSR 탭범위는 $1 \le i \le 8$이므로 지연소자의 초기값 s_i는 모두 영의 값을 가지게 된다. 그러므로 LFSR 키 스트림 생성수열은 식 (8)과 같다.

$$\begin{bmatrix} z_8 + k_8 \\ z_9 + k_9 \\ z_{10} + k_{10} \\ z_{11} + k_{11} \end{bmatrix} = \begin{bmatrix} k_7 & k_6 & k_5 & k_4 & k_3 & k_2 & k_1 & k_0 \\ k_8 & k_7 & k_6 & k_5 & k_4 & k_3 & k_2 & k_1 \\ k_9 & k_8 & k_7 & k_6 & k_5 & k_4 & k_3 & k_2 \\ k_{10} & k_9 & k_8 & k_7 & k_6 & k_5 & k_4 & k_3 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} z_{12} + k_{12} \\ z_{13} + k_{13} \\ z_{14} + k_{14} \\ z_{15} + k_{15} \end{bmatrix} = \begin{bmatrix} k_{11} & k_{10} & k_9 & k_8 & k_7 & k_6 & k_5 & k_4 \\ k_{12} & k_{11} & k_{10} & k_9 & k_8 & k_7 & k_6 & k_5 \\ k_{13} & k_{12} & k_{11} & k_{10} & k_9 & k_8 & k_7 & k_6 \\ k_{14} & k_{13} & k_{12} & k_{11} & k_{10} & k_9 & k_8 & k_7 \end{bmatrix} \begin{bmatrix} g_5 \\ g_6 \\ g_7 \\ g_8 \end{bmatrix}$$

$$G(x) = x^7 + x^5 + 1 \tag{9}$$

제안된 키 스케줄러의 생성다항식은 식 (9)와 같으며 식 (9)의 생성다항식에서 탭 계수 $g_1 = g_6 = g_8 = 1$을 만족하며 초기값은 모두 0을 제외한 값을 사용하였다. 식 (8)과 같은 키 생성수열을 식 (9)의 생성다항식에 이용하면 키 출력수열 Z는 식 (10)을 얻는다.

$$\begin{aligned} z_0 + k_0 &= g_1 s_1 + g_6 s_6 + g_8 s_8 \\ z_1 + k_1 &= g_1 k_0 + g_6 s_5 + g_8 s_7 \\ z_2 + k_2 &= g_1 k_1 + g_6 s_4 + g_8 s_6 \\ z_3 + k_3 &= g_1 k_2 + g_6 s_3 + g_8 s_5 \\ z_4 + z_4 &= g_1 k_3 + g_6 s_2 + g_8 s_4 \\ z_5 + k_5 &= g_1 k_4 + g_6 s_1 + g_8 s_3 \\ z_6 + k_6 &= g_1 k_5 + g_6 k_0 + g_8 s_2 \\ z_7 + k_7 &= g_1 k_6 + g_6 k_1 + g_8 s_1 \end{aligned} \tag{10}$$

$$\begin{aligned} z_8 + z_8 &= g_1 k_7 + g_6 k_2 + g_8 k_0 \\ z_9 + k_9 &= g_1 k_8 + g_6 k_3 + g_8 k_1 \\ z_{10} + k_{10} &= g_1 k_9 + g_6 k_4 + g_8 k_2 \\ z_{11} + k_{11} &= g_1 k_{10} + g_6 k_5 + g_8 k_3 \\ z_{12} + z_{12} &= g_1 k_{11} + g_6 k_6 + g_8 k_4 \\ z_{13} + k_{13} &= g_1 k_{12} + g_6 k_7 + g_8 k_5 \\ z_{14} + k_{14} &= g_1 k_{13} + g_6 k_8 + g_8 k_6 \\ z_{15} + k_{15} &= g_1 k_{14} + g_6 k_9 + g_8 k_7 \end{aligned}$$

식 (10)은 식 (9)의 생성다항식에 의한 키 수열을 나타내는 것으로써 본 논문에서는 식 (9)의 MSB와 LSB만을 이용하도록 하였다. 일반적으로 의사난수발생기의 주기는 2^m-1로써 표현되어지며 이때 m값이 클수록, 주기가 길수록 선형특성에 안전하다. 그러므로 m값이 긴 의사난수발생기를 설계하게되는데 이는 안전성에는 좋은 특성을 가지지만 주기에 비례하여 처리시간이 길어지며 특히 동기시스템인 경우 오류발생시마다 동기획득에 걸리는 시간이 길어지는 단점을 가진다. 그러므로 제안된 스트림 암호방식의 특징 중 난수발생은 한 주기안에 절대적으로 존재한다는 것을 이용한다. 즉 한 주기안에 존재하는 모든 의사난수를 이용하는 것이 아니고 초기 난수를 암호화에 이용하여 주기 시스템에 대한 주기시간의 단축을 꾀한다. 즉 LFSR의 초기값에 대하여 m만큼의 시간이 지날 때 출력되는 의사난수출력 Z_m을 LFSR의 출력으로 사용한다.

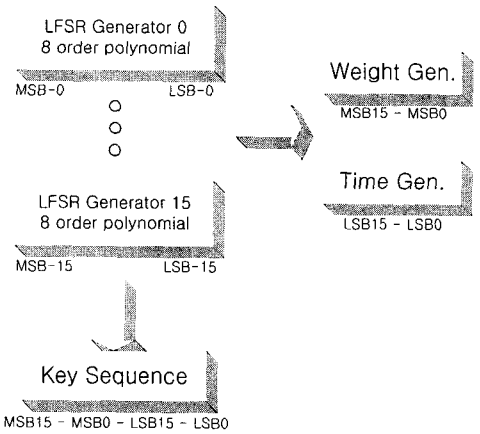


그림 8. LFSR 128 블록
Fig. 8. LFSR 128 block.

<그림 8>은 8단 LFSR을 16개 동일하게 사용하여 LFSR 128 블록을 구성한 것이다.

LFSR Generator 0에서부터 LFSR Generator 15까지는 동일한 생성다항식을 사용하는 LFSR로써 모두 16개의 LFSR로써 구성되어 있다. 초기입력이 인가된 후 8T(period)까지 출력되는 출력수열은 모두 128 비트의 크기를 가진다. 이때 각각의 8 비트들에 대하여 MSB와 LSB로 분류하여 Weight generator, Time generator의 입력으로 사용한다. 또한 MSB 그룹(group)과 LSB 그룹을 모두 합한 128 비트는 Key sequence로 입력된다.

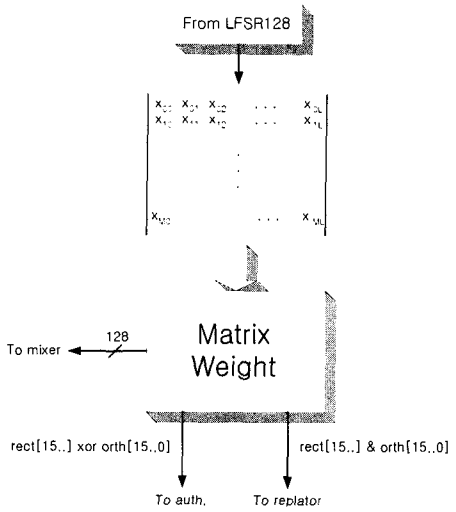


그림 9. 키 수열 블록
Fig. 9. Key sequence block.

<그림 9>는 매트릭스 연산을 수행하는 블록으로써 유입되는 128 비트의 데이터들에 대하여 행렬로써 표현하면 식 (11)과 같이 표현된다.

$$\begin{vmatrix} x_{00} & x_{01} & x_{02} & \cdots & x_{07} \\ x_{10} & x_{11} & x_{12} & \cdots & x_{17} \\ & & \vdots & & \\ x_{150} & x_{151} & x_{152} & \cdots & x_{157} \end{vmatrix} \tag{11}$$

식 (11)은 LFSR 128에 대한 데이터들에 대한 행렬 표현식이다. 식 (11)의 행렬에서 대각행렬에 대한 데이터들을 별도의 데이터로 취급하기 위하여 별도의 데이터 포맷팅을 수행한다. 이때 생성되는 별도의 데이터는 rect와 orth이다. 각각의 데이터들은 16 비트의 크기를 가지고 있으며 rect와 orth에 대한 데이터값들의 복원 및 생성을 용이하도록 하기 위하여 128 비트의 유입데이터들에 대하여 정방행렬의 대각만을 데이터로 추출

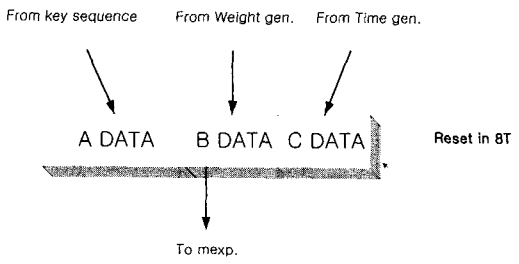


그림 10. Replator 블록
Fig. 10. Replator block.

하여 사용하였다.

정방행렬의 대각에 대한 값을 추출하기 위하여 식 (11)에서 64 비트씩 두 블록으로 분리하여 rect와 orth 값을 추출하였다. 식 (12)는 분리된 두 블록을 표시한다.

$$\begin{vmatrix} x_0 & x_1 & x_2 & \cdots & x_7 \\ x_8 & x_9 & x_{10} & \cdots & x_{15} \\ & & \vdots & & \\ & & \vdots & & \\ x_{120} & x_{121} & x_{122} & \cdots & x_{127} \end{vmatrix} = \begin{vmatrix} x_0 & x_1 & x_2 & \cdots & x_7 \\ x_8 & x_9 & x_{10} & \cdots & x_{15} \\ & & \vdots & & \\ x_{56} & x_{57} & x_{58} & \cdots & x_{63} \\ x_{64} & x_{65} & x_{66} & \cdots & x_{71} \\ x_{72} & x_{73} & x_{74} & \cdots & x_{79} \\ & & \vdots & & \\ x_{120} & x_{121} & x_{122} & \cdots & x_{127} \end{vmatrix} \quad (12)$$

여기에서 두 블록의 범위는 식 (13)과 같다.

$$\begin{aligned} ax &= (x_0, \cdots, x_{63}) \\ bx &= (x_{64}, \cdots, x_{127}) \end{aligned} \quad (13)$$

$ax+bx$는 mixer의 입력으로 사용되며 rect와 orth는 인증용, 즉 공개키로 사용하기 위하여 XOR 연산을 수행하게된다. 또한 비밀키로 사용하기 위하여 rect와 orth는 단순합인 &을 수행하여 replacement기능을 수행하는 replator의 입력으로 유입된다.

<그림 10>의 Replator는 LFSR로부터 생성된 의사난수에 대하여 8T시간동안에 새롭게 데이터들에 대한 재배치를 위한 포맷팅 기능을 수행한다. LFSR의 취약성을 보완하는 기능을 수행하기 위하여 Key sequence와 Weight generator 그리고 Time generator의 출력을 받아 64 비트의 새로운 키 수열을 생성한다. Replator는 <그림 10>과 같이 Key sequence와 Weight generator 그리고 Time generator의 출력을 받아서 데이터를 재포맷(re-formating)하는 블록으로써 Key sequence에서 출력되는 key sequence 32 비트와 Time generator 16 비트 그리고 Weight generator 16 비트의 출력을 Time generator의 출력인 time stamp를 이용하여 64 비트 크기를 가지는 암호용 키 데이터를 생성시키기 위한 16 by 8 weight and time sequence를 생성한다.

Key sequence는 <그림 11>과 같이 interleaving된 mixer 블록의 출력과 replator에서 생성된 weight-time stamp를 이용하여 64 비트의 크기를 가지는 새로운 블록 데이터를 생성한다. mixer(M)의 MSB와 replator(R)의 MSB를 XOR 시키고 mixer의 LSB와 replator의

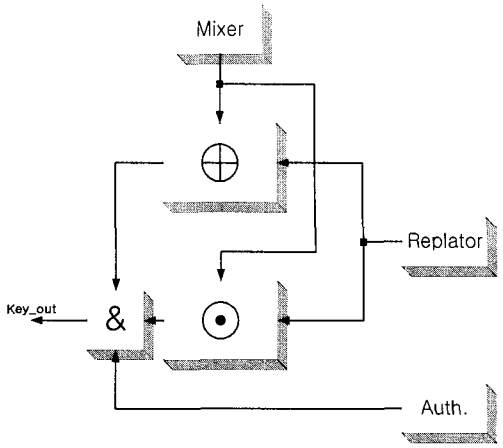


그림 11. Merging & Expansion 블록
Fig. 11. Merging & Expansion block.

LSB를 XNOR 시킨후 MSB와 LSB를 합하여 출력한다. 이때 출력되는 데이터는 Auth. 데이터를 포함하여 80 비트의 크기를 가지게 된다. Auth. 데이터는 key sequence 블록에세 생성된 16 비트의 데이터로써 송신자의 인증용으로 사용되게 된다. <그림 11>에서 &는 단순 데이터 통합을 의미한다.

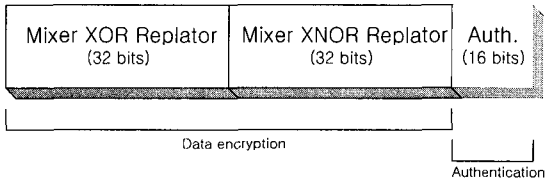


그림 12. Merging & Expansion 프레임 포맷
Fig. 12. Merging & Expansion frame format.

출력되는 데이터의 프레임 구조는 <그림 12>와 같다. <그림 12>에서와 같이 키 스케쥴러에서 출력되는 최종 출력값은 80 비트의 크기를 가지며 이중 64 비트는 데이터 암호화에 사용되며 나머지 16 비트는 송신자에 대한 인증용으로 사용된다.

mixer의 데이터 $M = \{m_0, m_1, \cdots, m_{62}, m_{63}\}$, replator의 데이터 $R = \{r_0, r_1, \cdots, r_{62}, r_{63}\}$ 에 대하여 정리하면 식 (14)와 같다.

$$key-out = M(MSB) \oplus R(MSB) \text{ and } M(LSB) \odot R(LSB) \quad (14)$$

<그림 11>의 전체 출력 Key_out은 식 (14)의 결과식과 Auth. 데이터를 합한 것으로써 다음 식 (15)와 같다.

$$key-out \text{ and } Auth. = M(MSB) \oplus R(MSB) \text{ and } M(LSB) \odot R(LSB) \text{ and } Auth. \quad (15)$$

여기에서 and는 &와 같이 단순합을 의미하며 ⊕는 비트들끼리의 XOR, ⊙는 비트들끼리의 XNOR 연산을 의미한다. 식 (15)를 행렬식으로 표현하면 다음 식 (16)과 같다.

$$MSB = \begin{bmatrix} m_{32} & m_{33} & \cdots & m_{39} \\ m_{40} & m_{41} & \cdots & m_{47} \\ & & \vdots & \\ m_{56} & m_{57} & \cdots & m_{63} \end{bmatrix} \oplus \begin{bmatrix} r_{32} & r_{33} & \cdots & r_{39} \\ r_{40} & r_{41} & \cdots & r_{47} \\ & & \vdots & \\ r_{56} & r_{57} & \cdots & r_{63} \end{bmatrix} \quad (16)$$

$$LSB = \begin{bmatrix} m_{0} & m_{1} & \cdots & m_{7} \\ m_{8} & m_{9} & \cdots & m_{15} \\ & & \vdots & \\ m_{24} & m_{25} & \cdots & m_{31} \end{bmatrix} \odot \begin{bmatrix} r_{0} & r_{1} & \cdots & r_{7} \\ r_{8} & r_{9} & \cdots & r_{15} \\ & & \vdots & \\ r_{24} & r_{25} & \cdots & r_{31} \end{bmatrix}$$

식 (16)에서와 같이 XOR, XNOR의 연산이 bit by bit 연산을 기본으로 수행하기 때문에 각 스트림들에 대한 연산은 가중치(weight)가 없는 단순 2진 데이터 연산이다. 이러한 단순연산의 결과 산출되어지는 데이터 스트림은 <그림 12>와 같이 사용목적에 따라 별도의 포맷을 수행하게된다. 즉 데이터 암호용으로는 XOR, XNOR연산의 결과값을 사용하며 인증용으로는 Auth.의 값을 사용하게 된다.

일반적으로 혼합형 암호시스템은 블록 암호시스템과 스트림 암호시스템의 결합을 의미하며 LFSR의 출력수열을 블록 암호시스템인 DES의 키 스케쥴로써 사용한다. 그러므로 DES의 데이터 블록의 크기가 m인 경우 LFSR의 출력수열의 크기는 m을 가진다. 즉 블록 암호시스템의 블록 크기가 스트림 암호시스템의 키 수열의 크기와 같아야 한다. 그러므로 안전성을 위하여 블록크기를 매우 크게 해야 한다는 결론을 가진다. LFSR에 의한 의사난수발생은 LFSR의 출력수열에 대한 주기가 길수록 안전성을 가지게 된다. 그러나 LFSR의 출력수열의 안전성을 위하여 주기를 매우 크게 설정할 경우 블록 암호시스템의 블록 데이터의 크기가 따라서 증가해야한다는 결론이다. 결국에는 너무 많은 데이터의 암호화를 위하여 너무 많은 잉여분(redundancy)을 산출하게 된다. 그러므로 암호화에 대한 효율성이 떨어지게 된다.

대칭형 암호시스템의 특징중의 하나는 비밀키를 가지고 있다는 것이다. 그러므로 사용자의 수가 증가하거나 암호화하여 전송해야할 데이터의 블록크기가 매우 길어진다면 대칭형 암호시스템의 장점인 처리속도 효율이 크게 저하될 것이다. 또한 암호화된 데이터와 키 데이터량이 방대하여 암호화 자체 성능도 저하된다.

그러므로 본 논문에서는 기존의 혼합형 암호시스템과 같이 데이터와 별개의 키 데이터를 가지고 암호화를 수행하는 것이 아니고 <그림 13>과 같이 암호화하고자하는 데이터를 키 데이터로 사용하며 키 수열 생성은 LFSR과 키 재포맷팅을 이용하여 블록 데이터의 크기에 상관없이 키 수열을 생성할 수 있도록 하였다.

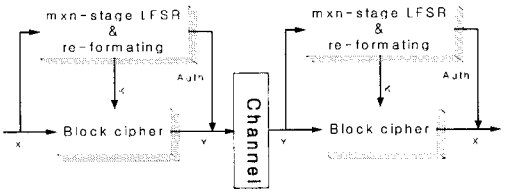


그림 13. 제안된 혼합형 암호시스템
Fig. 13. Proposed hybrid cryptosystem.

<그림 13>에서와 같이 입력 데이터 X에 대하여 키 스케쥴러에 의하여 X에 의한 키 값 K를 생성하고 생성된 킷값 K와 입력데이터 X를 이용하여 암호화된 출력값 Y를 생성한다.

또한 키 스케쥴러에서는 암호화용 킷값과 인증용 수열을 동시에 생성하며 이 두값은 전송로(channel)를 통하여 수신자에게 전송된다. <그림 13>의 제안된 혼합형 암호시스템은 기존 혼합형 암호시스템과는 확연한 차이를 가진다. 즉 기존 혼합형 암호시스템은 비도를 증가시키기 위해서는 LFSR의 주기를 길게해야 하며 이로 인한 블록 데이터의 값도 따라서 증가되는 단점을 가지고 있다. 또한 데이터와 킷값을 별도로 사용함으로써 암호화에 필요한 자원관리를 항상 염두에 두고 있어야 한다. 특히 킷값은 고정된 값으로 존재하게 되므로 비인가자의 킷값 획득은 암호화의 필요성을 무력화시키게 되는 중요한 요소를 가지게된다. 그러나 본 논문에서 제안한 <그림 13>과 같은 혼합형 암호시스템은 유입되는 정보데이터를 기반으로 암호화에 사용되는 킷값을 생성하므로 데이터에 따라서 암호화 패턴이

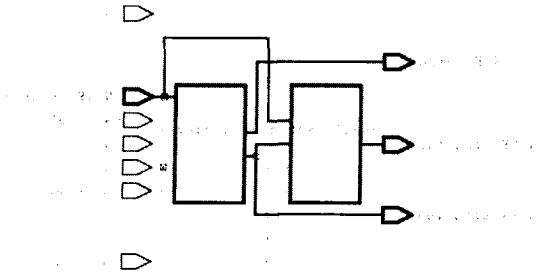


그림 14. 제안된 혼합형 암호시스템의 전체 블록도
Fig. 14. Proposed hybrid cryptosystem.

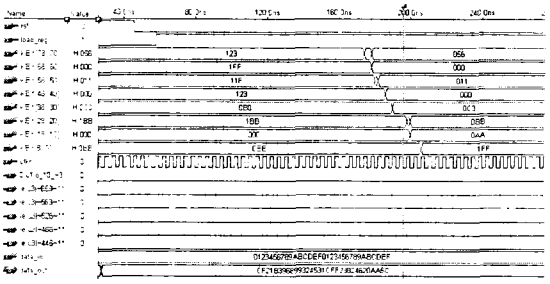


그림 15. 제안된 혼합형 암호시스템의 모의실험 결과
Fig. 15. Proposed hybrid cryptosystem simulation.

표 2. 기존 암호알고리즘과 새로운 혼합형 암호알고리즘의 비교
Table 2. Comparison of general & new hybrid cryptoalgorithm.

	DES	3DES	SEED	AES	hybrid
구 조	Feistel	Feistel	Feistel	SPN	Feistel &SPN
라운드수	16	48	16	10	1
데이터 길이 (bit)	64	64	128	128/192/256	128
키 길이 (bit)	56	112/168	128	128	80
시스템 속도 (MHz)	80	80	40	33	40
처리율 (Mbps)	50	25	251	256	640

변화된다.

<그림 14>와 같이 본 논문에서 제안된 혼합형 암호시스템의 전체 블록도는 Synopsys ver 1999.10으로 설계하였고 128비트 데이터 길이, 80비트 키 길이, 40MHz의 시스템 속도를 Altera MAX+PlusII툴로 모의실험한 결과 데이터 처리율이 640Mbps로 <그림 15>에서 확인하였다. 입력 데이터는 "0123456789ABCDEF 0123456789AB CDEF"이며 출력 데이터는 "0F21B396 899324531CFF238 24620AA5C"이다.

<표 2>는 제안된 알고리즘과 기존의 알고리즘을 6개의 파라미터(parameter)로 비교해 보면 다음과 같이 성능이 향상됨을 확인하였다.

VI. 결 론

비대칭형 암호 알고리즘의 비도는 수학적 난이도의 해법에 의존한다. 그러므로 비도를 증가시키기 위해서는 더욱 복잡한 수학적인 배경을 요구하게된다. 이러한 수학적 난이도에 대한 알고리즘 구현은 실시간처리의 어려움을 가지므로 시스템 효율면에서 비경제적이다. 그러므로 본 논문에서는 대칭형 암호알고리즘 중 블록암호방식과 스트림 암호방식을 적용한 혼합형 암호 알고리즘을 변형하여 비대칭형 암호시스템 수준의 서비스를 수행할 수 있도록 하였다.

기존 혼합형 암호알고리즘은 평문과 암호용 키가 독립적이었으나 제안된 혼합형 암호 알고리즘은 평문을 바탕으로 키 스케쥴러에 의하여 종속변수로 변환되고 이를 이용하여 암호문을 생성하는 메카니즘으로 구성되어 있다. 그러므로 기존 혼합형 암호알고리즘의 단점이었던 비도 증가와 의사난수발생기의 주기와의 비례관계를 제안된 알고리즘에서는 적용 받지 않는다. 또한 기존 대칭형 암호시스템에서 사용되는 iteration을 제안된 암호시스템에서는 사용하지 않으므로 처리속도가 매우 증가된다. 그러므로 제안된 혼합형 암호시스템은 대용량 평문에 대한 암호화와 인증이 필요한 네트워크에서의 실시간 암호화가 가능하리라 사료된다.

참 고 문 헌

[1] E. Biham, "On the Applicability of Differential Cryptanalysis to Hash Functions", Lecture at EIES Workshop on Cryptographic Hash Functions, Mar. 1992.

[2] E. Biham, "On Matsui's Linear Cryptanalysis", Advances in Cryptology-EURO-CRYPT'94 Proceedings, Springer-Verlag, pp. 398-412, 1995.

[3] H. Feistel, "Step Code Ciphering System", U.S. Patent #3,798,360, 19 Mar. 1974.

[4] E. F. Brickell, J. H. Moore, and M. R. Purtill,

"Structure in the S-Boxes of the DES", Advances in Cryptology-CRYPTO'86 Proceeding, Springer-verlag, pp. 3-8, 1987.

[5] A. G. Broscius and J. M. Smith, "Exploiting Parallelism in Hardware Implementation of the DES", Advances in Cryptology-CRYPTO'91 Proceeding, Springer-verlag, pp. 367-376, 1992.

[6] L. Brown, J. Pieprzyk, and J. Seberry, "Key Scheduling in DES Type Cryptosystems", Advances in Cryptology-CRYPTO'90 Proceeding, Springer-verlag, pp. 221-228, 1990.

[7] R. Durstenfeld, "Algorithm 235: Random Permutation", Communications of the ACM, Vol. 7, No. 7, pp. 420, Jul. 1964.

[8] E. Biham, A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard", Springer-verlag, 1993.

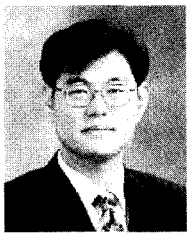
[9] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", Advances in Cryptology-CRYPTO'90 Proceeding, Springer-verlag, pp. 2-21, 1991.

[10] M. Gude, "Concept for a High Performance Random Number Generator Based on Physical Random Phenomena", Frequenz, Vol. 39, pp. 187-190, 1985.

[11] R. Gottfert, H. Neiderreiter, "On the Linear Complexity of Products of Shift Register Sequences", Advances in Cryptology-CRYPTO'93 Proceeding, Springer-verlag, pp. 151-158, 1994.

저 자 소 개

李 善 根(正會員) 第38卷 SD編 第12號 參照



宋 濟 昊(正會員)
현재 : 국립 익산대학 전기과 교수. 원광대학교 전자공학과 공학박사. <주관심분야 : 스마트 카드, 이동통신, ASIC>

金 太 亨(正會員)
현재 : 국립 익산대학 전자통신과 교수. 원광대학교 전자공학과 공학박사. <주관심분야 : 이동통신, ASIC, 음향시스템, 그래픽 이퀄라이저>

金 煥 溶(正會員) 第38卷 SD編 第12號 參照