The Journal of Korean
Institute of CALS/EC
Vol. 6, No. 1, April 2001

전자상거래 보안을 위한 YK2 암호시스템의 구현

서장원*

An Implementation of YK2 Cipher System for Electronic Commerce Security

Jang-Won Suh

Abstract

EC(Electronic Commerce) which is done on the virtual space through Internet has strong point like independence from time and space. On the contrary, it also has weak point like security problem because anybody can access easily to the system due to open network attribute of Internet. Therefore, we need the solutions that protect the EC security problem for safe and useful EC activity. One of these solutions is the implementation of strong cipher system.

YK2(Young Ku Kang) cipher system proposed in this paper is good solution for the EC security and it overcome the limit of current block cipher system using 128 bits key length for input, output, encryption key and 32 rounds. Moreover, it is designed for the increase of time complexity by adapting more complex design for key scheduling algorithm regarded as one of important element effected to encryption.

Key Word : EC Security, Cipher system, YK2, Key scheduling algorithm

* 서울산업대학교 공동실험실습관

1. 서론

최근 들어, 통신과 컴퓨터 기술이 비약적으로 발전하면서 일상적으로 행하던 은행 업무, 쇼핑 등의 일부 거래가 컴퓨터 네트워크를 이용한 전자 공간에서 이루어지고 있다. 특히, 인터넷의 급속한 확산에 의하여 많은 수의 인터넷 쇼핑몰들이 개장되면서 전자상거래 역시 급속히 확산되고 있는 실정이다. 따라서, 전자상거래를 보다 활성화하기 위해서는 전자상거래상의 신뢰성을 확보할 수 있는 보안 기술이 필수 선결요소라 할 수 있겠다.

그러나, 전자상거래 상에서의 보안 기술 문제는 인터넷의 개방 지향적인 특성 때문에 누구든지 쉽게 접속이 가능하고 해킹에 의한 정보 접근이 용이하기 때문에 그리 쉬운 문제는 아니다. 그러므로, 정보의 암호화는 이러한 보안상의 문제점을 해결하기 위한 한 가지 대안으로 제시될 수 있다. 이를 바탕으로 본 논문에서는 효율적인 암호시스템인 YK2 (Young Ku Kang) 암호시스템을 제안하였다.

본 논문에서 제안한 YK2 암호시스템은 비선형 변환과 선형 변환의 적절한 조합에 의해 설계했으며, 암호시스템의 전체 구조는 SPN(Substitution Permutation Network) 구조를 적용하여 설계하였으며[5], 내부 함수 F에서는 데이타 블록의 좌/우 측면에 교대로 비선형 변환을 적용시키는 Type-3 Feistel 구조를 적용하여 설계하였다[4]. 이를 바탕으로 암호문의 생성 속도가 빠르게 진행되고, 외부 공격에도 대처할 수 있도록 설계하였다.

2. 전자상거래 보안 기술

2.1 전자상거래 상에서의 보안 문제

전자상거래라 함은 통합적으로 자동화된 정보체계 환경 하에서 거래 주체간의 모든 측면에 걸쳐 생산, 구매, 대금지불, 운반, 서비스 등의 제반 비즈니스 형태를 네트워크를 통해 전자적으로 행하는 것으로 정의할 수 있다[2]. 인터넷상에서 안전한 상거래를 보장하기 위해서는 거래 상대에 대한 인증과 거래 내용에 부정이 없다는 것을 증명하는 구조가 갖춰져 있어야 하며, 거래 내용이 제 3자에게 노출되지 않게 기밀성을 보장하는 암호화와 다른 사람에 의한 변조 또는 본인이 작성한 사실이나 내용을 부정할 수 없게 하는 전자서명 기능을 채택한 프로토콜 등이 필요하다[1].

최근 들어, 네트워크의 발달에 따른 개방형 통신망(open network)에 있어서 각 개인이나 기업 또는 정부의 각종 정보들이 인터넷을 통해 손쉽게 상대방에게 전달되고 있다. 바꾸어 말하면, 이러한 다수의 정보들을 제 3자가 인터넷을 통하여 손쉽게 접근 할 수 있다는 것을 의미한다. 더욱이 웹 기술의 발달에 따라 일반 사용자들도 이러한 정보들에 손쉽게 접근이 가능해짐으로써 인증이나 개인의 사생활 및 개인정보 등의 전자상거래 보안 문제는 더욱 더 중요한 과제로 대두되고 있다.

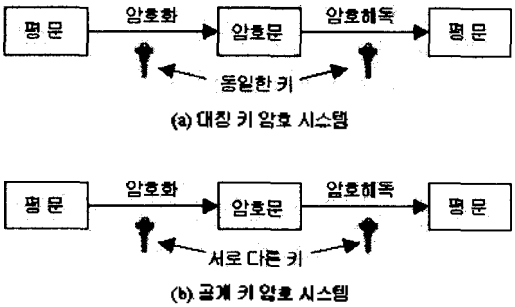
따라서, 전자상거래에서도 인터넷 보안 문제를 반드시 고려하여야 하며, 인터넷상의 정보보호 문제를 포함한 보안 문제를 해결하기 위한 제반 장치가 마련되지 않는다면 전자상

거래의 안전성과 신뢰성을 기대할 수가 없다. 이러한 정보보호 문제를 해결하기 위한 근본적인 방법 중의 하나가 네트워크 상에서 송/수신 메시지나 거래 내용에 관한 정보를 암/복호화하여 사용하는 암호 기술이다[7].

2.2 암호 기술

일반적으로 암호화를 기반으로 한 암호 기술은 대개 암호키의 운용 방식에 따라 대칭키 암호시스템과 공개키 암호시스템으로 크게 구별된다.

대칭키(대칭형) 암호시스템은 동일한 암/복호화 키를 사용하는 형태로써, 여기서 송/수신자는 동일한 비밀키를 공유해야 한다. 따라서, 정보 전송의 암호화나 사용자 및 메시지 인증은 통상 속도가 빠른 공개키 알고리즘 방식(DES, IDEA, RC5 등)을 이용한다.



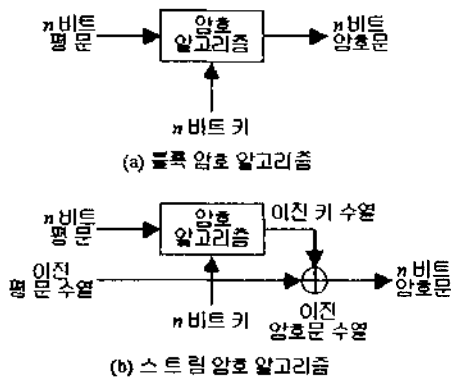
<그림 1> 대칭키 암호 시스템과 공개키 암호 시스템의 비교

이에 비해서 공개키(비대칭형) 암호시스템은 키 분배(또는 공유) 문제에 근거하여 공개키 분배 알고리즘과 공개키 관리 알고리즘으로 나누어질 수 있다. 이런 형태는 암/복호화 키가 서로 다르며, 어느 한 키를 공개한다 할지라도 대응되는 다른 키를 유도해 내는 것은 계산상 불가능하도록 설계되어진다[9]. 통상적으로 공개키 알고리즘 방식(RSA, DSA, Schnorr 등)은 이에 필요한 키 분배나 부인 방지를 위한 디지털 서명에 주로 이용된다.

이상에서 설명한 암호시스템을 그림으로 비교해 보면 다음의 <그림 1>과 같이 나타낼 수 있다.

2.3 대칭키 암호 시스템

대칭키 암호시스템은 메시지 처리 형식에 따라 <그림 2>와 같이 블록 암호 알고리즘과 스트림 암호 알고리즘으로 나누어진다.



<그림 2> 대칭키 암호시스템의 형태 비교

블록 암호 알고리즘은 <그림 2>의 (a)와 같이 고정된 크기의 입력 블록과 임의의 암호화에 사용된 키에 의해 고정된 크기의 출력 블록으로 변형되는 암호 알고리즘으로서,

출력 비트의 각 비트는 입력 블록과 암호화에 사용된 키의 모든 비트에 영향을 받아 결정된다[8].

블록 암호 알고리즘의 구조는 데이터 블록의 좌/우 측면에 비선형 변환을 적용시키는 Feistel 구조(Twofish, YK2 등)와 데이터 블록에 동시에 비선형 변환을 적용시키는 SPN 구조(Rijndael, Serpent 등)로 크게 분류되며, 최근에는 Feistel 구조를 변형시킨 Modified Feistel 구조(MARS, RC6 등)로도 설계되고 있는 추세이다.

3. YK2 암호시스템의 설계

3.1 YK2 암호시스템의 소개

본 논문에서 제안한 YK2 암호시스템은 대칭키 암호시스템에 바탕을 둔 128비트 블록 암호 알고리즘이다[11]. 또한, 암호키는 128비트의 블록 크기와 128~400비트 범위의 가변적인 키 크기를 가지며, 높은 보안성과 빠른 속도 구현의 융통성을 제공하기 위해 다양한 명령을 사용함으로써 보다 견고한 안정성을 제공한다.

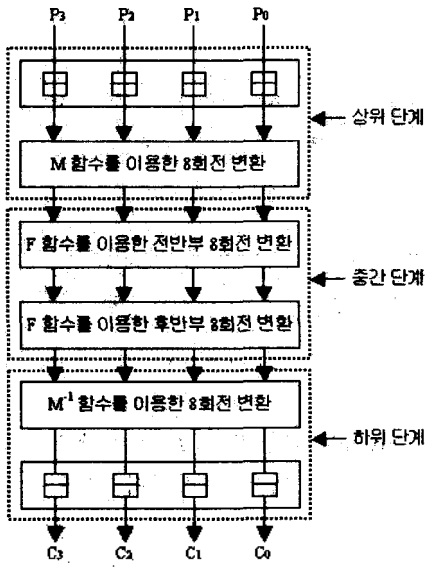
이러한 YK2 암호시스템의 전체 구조는 다음과 같은 특성들로 구성되어 있다.

- 성격 : 128 비트 블록 암호 알고리즘
- 전체 구조 : SPN
- 내부 구조 : Type-3 Feistel
- 입/출력키의 크기 : 128비트
- 암호키의 크기 : 128~400비트
- 키 확장 프로시져 : K[0…39]
- 회전 수 : 32회전
- 내부 함수 : M, F, M^{-1} 함수
- S-Box : 2개(S_1, S_2)

3.2 전체 구성

제안한 YK2 암호시스템은 128비트 평문을 4개의 32비트 블록 P_0, P_1, P_2, P_3로 분할하여 입력하고, 이를 토대로 키를 부가하여 입력 내용을 변환시키는 8회전의 상위 단계와 실제적인 암호화를 수행하는 16회전의 중간 단계, 그리고 상위 단계를 역으로 수행한 후 키를 감소시키는 8회전 하위 단계를 거친 후에 4개의 32비트 블록 C_0, C_1, C_2, C_3를 산출한 후, 최종적으로 이것들을 조합하여 128비트 암호문을 생성하는 구조로 설계되었다.

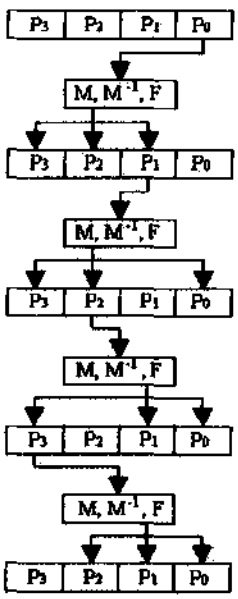
YK2 암호시스템의 전체 구조는 <그림 3>과 같다.



<그림 3> YK2 암호시스템의 전체 구조

<그림 3>에서, 상위 단계는 선택 평문 공격을 어렵게 하고 중간 단계에서 생성될 암호문에 대한 외부 공격을 어렵게 만들기 위한 과정으로서 M 함수를 이용한 8회전으로 수행된다. 중간 단계는 실제로 암호화가 이루어지는 핵심 과정으로서 암/복호화가 동일한 강도로 수행될 수 있도록 F 함수를 이용한 전반부 8회전과 후반부 8회전으로 구분하여 총 16회전으로 수행된다. 그리고, 하위 단계는 상위 단계에서 수행된 것을 역순으로 처리하는 과정으로서 M^{-1} 함수를 이용한 8회전으로 수행되는데, 이것은 계산 복잡도를 증가시키고 선택 암호문 공격에 안전하게 대처하기 위한 단계이다.

다음의 <그림 4>는 사용되는 각 함수의 수행 과정, 즉 Type-3 Feistel 연산을 도식화한 것이다. 여기서, M^{-1}는 M 함수를 역으로 수행하는 함수를 의미한다.



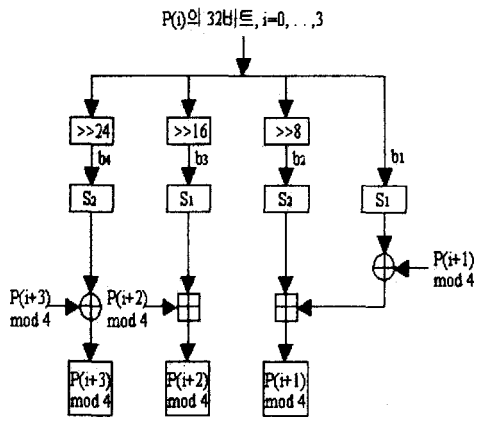
<그림 4> 각 함수의 변환 단계 구조

여기서, M^{-1}과 F^{-1} 함수는 각각 M과 F 함수를 역으로 수행하는 함수를 의미한다. 다시 말해서, 상위 단계에서 사용된 M 함수는 그것의 역인 M^{-1} 함수를 하위 단계에서 사용하고, F와 F^{-1} 함수는 중간 단계에서 사용된다.

이제 각 단계별로 수행되는 과정을 살펴보도록 하자.

(1) 상위 단계

여기서는 4개의 각 평문 블록들에 키를 더한 후 8회전의 Type-3 Feistel 연산을 수행한다. 각 회전에서는 하나의 블록(초기 블록)을 사용하여 다른 3개의 블록(목표 블록)을 변경시킨다. 즉, <그림 5>와 같이 M 함수를 이용하여 초기 블록의 32비트를 우측으로 8비트씩 이동하면서 그 때마다 최하위 8비트를 2개의 S-Box(Substitution-Box) S_1과 S_2에 대한 인덱스로 사용하여 이에 대응되는 S-Box의 내용들을 다른 3개의 목표 블록들과 더하거나 XOR 시킨다.



<그림 5> M 함수의 구조

이를 좀 더 자세히 설명하면, 예를 들어 처음 초기 블록은 32비트의 P(0)이고 목표 블록 역시 각 32비트의 P(1), P(2), P(3) 등 3개 블록이라고 하자. 초기 블록 P(0)의 32비트를 b_1, b_2, b_3, b_4라 할 때(여기서 b_1은 최하위 8비트이고, b_4는 최상위 8비트이다), b_1과 b_3는 S-Box S_1의 인덱스로 사용하고 b_2와 b_4는 S-Box S_2의 인덱스로 사용하게 된다. 먼저, 첫 번째 목표 블록 P(1)에 $S_1[b_1]$을 XOR 하고 다시 그 목표 블록에 $S_2[b_2]$를 더한다. 또한, 두 번째 목표 블록 P(2)에는 $S_1[b_3]$를 더하고, 세 번째 목표 블록 P(3)에는 $S_2[b_4]$를 XOR 한다.

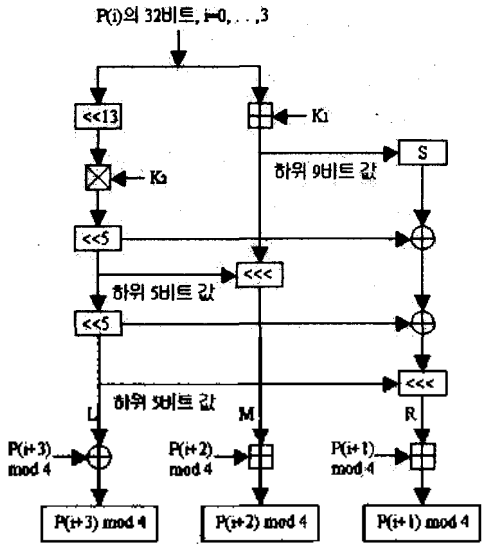
결국 이와 같은 연산을 수행하기 위해서는 <그림 5>에서 보는 바와 같이 초기 블록은 우측으로 24비트 회전이동 하게 된다.

그런 후에, 다음 회전을 위하여 4개의 평문 블록을 회전이동 시킨다. 즉, 현재의 첫 번째 목표 블록은 다음의 초기 블록이 되고 현재의 두 번째 목표 블록은 다음의 첫 번째 목표 블록, 현재의 세 번째 목표 블록은 다음의 두 번째 목표 블록, 그리고 현재의 초기 블록은 다음의 세 번째 목표 블록으로 각각 이동시킨다.

위와 같은 순서를 두 번 반복해서 8회전 수행한 후 산출된 결과를 중간 단계로 이동 시킨다.

(2) 중간 단계

YK2 암호시스템의 핵심 단계로서 16회전 Type-3 Feistel 연산을 수행한다. 각 회전에서는 F 함수를 이용하여 상위 단계에서 산출된 평문 블록을 암호문으로 변환시킨다. 여기서, F 함수는 하나의 초기 블록을 입력으로 하여 3개의 출력 블록을 반환하도록 하는 함수이며, 그 구조는 다음의 <그림 6>과 같다.



<그림 6> F 함수의 구조

이제 먼저 F 함수에 대하여 자세히 설명하기로 한다. 우선, 편의상 3개의 임시 변수 L, M, R을 설정한다. 초기 L 값은 초기 블록을 좌측으로 13자리 회전이동 시킨 비트 값으로, M 값은 초기 블록과 키(K_1)를 더한 값으로, R 값은 M의 하위 9비트를 S-Box 배열의 인덱스로 사용하여 이에 대응되는 S-Box의 비트 값으로 정한다.

그런 후에, 키(K_2)를 L과 곱하여 이를 5자리 좌측으로 회전이동 시킨다. 그리고, L을 R에 XOR 시키고, 또한 L의 하위 5비트 값을 회전이동 양으로 하여 M을 좌측으로 회전이동 시킨다. 다음으로 L을 5자리 좌측으로 더 회전이동 시키고 다시 R과 XOR 시킨다. 최종적으로 다시 L의 하위 5비트를 회전이동 양으로 하여 R을 좌측으로 회전이동 시킨다.

이러한 과정을 통하여 생성된 첫 번째 출력 블록을 L, 두 번째 출력 블록을 M, 그리고 세 번째 출력 블록을 R이라 한다.

중간 단계에서는 앞에서 언급했듯이 F 함수의 출력 블록과 목표 블록들의 연산을 두 번 반복하여 전반부 8회전과 후반부 8회전을 역순으로 적용한다. F 함수의 첫 번째 출력 블록 L과 세 번째 출력 블록을 XOR하고, 두 번째 출력 블록 M과 세 번째 출력 블록 R을 두 번째 목표 블록과 첫 번째 목표 블록에 각각 더한다. 또한, 후반부 8회전의 각 회전에서는 첫 번째 출력 블록 L과 첫 번째 목표 블록을 XOR하고, 두 번째 출력 블록과 세 번째 출력 블록을 두 번째 목표 블록과 세 번째 목표 블록에 각각 더한다.

추가로, 매 회전이 끝난 후에는 다음 회전을 위하여 초기 블록을 13자리 좌측으로 회전이동 시킨 후 세 번째 목표 블록으로 하고 첫 번째 목표 블록을 초기 블록으로, 두 번째 목표 블록을 첫 번째 목표 블록으로, 세 번째 목표 블록을 두 번째 목표 블록으로 하여 F 함수를 이용하여 16회전을 반복 수행한다.

(3) 하위 단계

이 단계에서는 초기 블록이 상이한 순서로 처리되는 것을 제외하고는 상위 단계에서의 과정을 역순으로 처리한다. 즉, M^{-1} 함수를 이용하여 상위 단계의 출력 블록을 하위 단계의 입력 블록으로 처리하는 것을 의미한다. 하위 단계에서도 상위 단계와 동일하게 각 회전에서는 하나의 초기 블록을 이용하여 다른 3개의 목표 블록들을 변경시킨다. 다시 말해서, 초기 블록의 32비트를 b_1, b_2, b_3, b_4라 할 때 b_1과 b_3는 S-Box S_2의 인덱스로 사용하고 b_2와 b_4는 S-Box S_1의 인덱스로 사용하여, 첫 번째 목표 블록에서 $S_2[b_1]$을 XOR 시키고, 두 번째 목표 블록에서 $S_1[b_4]$를 빼고, 세 번째 목표 블록에서 $S_2[b_3]$를 뺀 후 여기에 $S_1[b_2]$를 XOR 한다. 결과적으로 초기 블록은 좌측으로 24 비트 회전이동 시킨 결과가 된다.

그런 후에, 다음 회전을 위하여 4개의 평문 블록을 회전 이동시킨다. 즉, 현재의 첫 번째 목표 블록은 다음의 초기 블록이 되고 현재의 두 번째 목표 블록은 다음의 첫 번째 목표 블록, 현재의 세 번째 목표 블록은 다음의 두 번째 목표 블록, 그리고 현재의 초기 블록은 다음의 세 번째 목표 블록으로 각각 이동시킨다.

위와 같은 순서를 반복적으로 8회전 수행한 후 그 결과를 최종적으로 암호문으로 산출하게 된다.

3.3 키 스케줄링 알고리즘

키 스케줄링 알고리즘은 실제 암호화를 수행하는 중간 단계인 F 함수 내부에서 사용되는 K_1, K_2의 키 확장 프로시져로써, 임의로 구성된 각 32비트의 n개($4 \le n \le 14$) 키를 40개의 키로 확장하는 프로시져를 의미한다. 그런데, 키 확장 프로시져는 F 함수 내의 곱셈에서 사용되는 키들이 다음과 같은 특성들을 갖도록 보장하는 키 스케줄링 알고리즘이 있어야 한다.

- 곱셈에 사용된 키에서 2개의 최하위 비트는 1 이다.
- 어떤 키도 10개 비트가 연속적으로 0 이거나 혹은 1을 포함해서는 안 된다.

이러한 특성을 유지하기 위한 절차는 다음과 같은 단계를 통해 구성될 수 있다.

1. 먼저 15개의 임시 테이블 T[]에 초기 키들을 n 만큼 복사하고 나머지는 0으로 한다.

$$T[0\cdots n-1] = k[0\cdots n-1],$$
$$T[n] = n,$$
$$T[n+1\cdots 14] = 0.$$

2. 그런 다음, 아래와 같은 과정을 4번 반복하여 매 반복 때마다 다음 10개의 확장키를 계산한다.
   - 배열 T[]에 다음과 같은 선형 공식을 적용하여 변환시킨다.

$$T[i] = ((T[i-7 \bmod 15] \oplus T[i-2 \bmod 15]) <<< 3) \oplus (4i+j)$$

   여기서 i는 0…14이며, j는 반복횟수로 j=0일 때 1회전, j=1일 때 2회전임을 의미한다.

   - 4회전의 Type-1 Feistel 연산을 적용하여 배열 T[]를 혼합시킨다.

$$T[i] = (T[i]+S[T[i-1] \bmod 15\text{의 하위 9비트}]) <<< 9$$

   - 그런 다음, T[]에서 10개를 취하여 확장키 배열 K[]에 재배치시킨다.

$$K[10j+i] = T[4i \bmod 15]$$

3. 최종적으로, 암호를 위한 곱셈(K_2 키)에 사용되는 16개를 K[5], K[7],…, K[35]로 선정하여 이것들이 앞에서 언급한 두 가지 특성을 만족하도록 아래와 같이 조정한다.
   - j=K[i]∧3으로 하여 K[i]의 최하위 2비트를 j 값으로 하고 ω=K[i]∨3으로 하면, ω의 최하위 2비트는 1이 된다.
   - 10개 또는 그 이상의 연속적인 0 이나 1을 가지고 있는 ω에 대한 메스크 M을 만든다. 즉, M은 ω_i이 10개 이상 이 연속적인 0이나 1로 되어 있다면 M_i=1로 하고, 0 이나 1이 연속적으로 10개 이상이 아닐 경우에는 M_i=0으로 한다. 그리고, ω에서 0 이나 1의 양쪽 마지막에 대응되는 M에서의 1을 0으로 하고, M의 최하위 2비트와 최상위 비트를 0으로 한다. 즉, i<2, i=31 이 거나 또는 ω의 i번째 비트가 (i+1)번째 비트나 (i-1)번째 비트와 다르면 M 의 i번째 비트를 0으로 한다.
     예를 들어, $\omega=0^3 1^{13} 0^{12} 1011$ 이라면, 처음에 M은 $M=0^3 1^{25} 0^4$로 된다. 그런 후에, 4, 15, 16, 28 자리의 1을 다시 0으로 하면 M은 $M=0^4 1^{11} 001^{10} 0^5$이 된다.
   - 다음으로 ω를 수정하기 위해 32비트 테이블 B를 이용하는데 이때, B 테이블의 요소들은 7개의 연속적인 0 이나 1을 포함하지 않도록 선택한다. 특히, B[]={a4a8d57b, 5b5d193b, c8a8309b, 73f9a978}을 사용하는데, 이것들은 S- Box 내 S[265]~S[268]의 값들이다.
     이러한 요소들을 선택한 이유는 이 요 소들에서는 2번씩 나타나는 유형이 단 지 14개의 8비트 유형만 있고, 또한 어떤 유형도 2번 이상은 나타나지 않기 때문이다.

- 최종적으로, p를 매스크 M과 AND 연산을 시킨 후 XOR하여 그 결과를 K[i]로 한다. 결국 M의 최하위 2비트 가 0이므로 K[i]의 최하위 2비트는 1 이 될 것이고 배열 B[]의 값들 때문에 K[i]는 10개의 연속적인 0 이나 1을 갖지 않는다는 것이 보장된다.

이런 키 확장 프로시져는 K[5], K[7],…, K[35]가 위에서 언급한 여러 가지 특성을 갖고 있다는 것을 보장할 뿐만 아니라 무작위 특성도 갖게 된다.

이와 같은 키 스케쥴링 알고리즘은 다음과 같다.

[알고리즘 1] 키 스케쥴링 알고리즘

```
/* B[]의 초기화 */
B[] = {a4a8d57b, 5b5d193b, c8a8309b, 73f9a978}

/* 임의로 설정된 키로 T[] 초기화 */
T[0…n-1]=k[0…n-1], T[n]=n, T[n+1…14]=0

/* 4 회전 반복, 매번 반복마다 K[]의 10개키 계산 */
for j=0 to 3 do
   for i=0 to 14 do
     T[i]=((T[i-7 mod 15]⊕T[i-2 mod 15]) << 3)⊕
          (4i+j)
   repeat 4 times
     for i=0 to 14 do
     T[i]=(T[i]+S[low 9 bits of T[i-1 mod 15]])<<9
   end repeat
   for i=0 to 9 do
      K[10j+i]=T[4i mod 15]
   end for

/* 곱하기 사용 키 변환 */
for i=5,7,…35 do
   j=K[i]의 최하위 2비트∧3
   ω=K[i]의 최하위(2비트∨3) 비트
/* ω의 비트 매스크 키 생성 */
   r=K[i-1]의 하위 5비트 값
   p=B[j] << r
   K[i]=ω⊕(p∧M)
end for
```

3.4 S-Box 설계

S-Box는 대개의 블록 암호 알고리즘에서 사용되는 비선형 대입 연산을 수행하는 비선형 함수로서, Lucifer 알고리즘에서 최초로 사용됐으며[4], 그 후 대개의 암호 알고리즘에서 널리 사용되고 있다. 이런 S-Box는 암호화에 민감한 영향을 미치므로 결국 S-Box의 구성을 어떻게 하느냐에 따라 견고한 암호시스템을 구축할 수 있다.

암호화에 있어서 S-Box의 특성은 암호문 생성에 매우 중요한 요소라 할 수 있는데, 본 논문에서 설계한 S-Box S_1, S_2는 의사난수(pseudo-random) 형태로 생성되어 만들어졌다. S-Box의 요소들은 i=0…102, j=0…4, $S[5i+j]=SHA\text{-}1(5i|c_1|c_2|c_3)_j$로 하여 생성하였다. 여기서, $SHA\text{-}1(\cdot)_j$는 SHA-1의 출력에서 j번째 비트이고 i는 32비트 정수를 나타내며, c_1, c_2, c_3는 고정 상수 값이다[9].

YK2 암호시스템에서 S-Box는 다음과 같은 특성들을 만족하게끔 설계하였다.

- S-Box는 32비트가 모두 0 이거나 또는 모두 1인 것은 포함하지 않는다.
- S-Box를 S_1과 S_2로 나눌 때, S_1과 S_2 에서 각각 두 개의 요소들은 32비트 중 적어도 3개가 달라야 한다.
- S-Box는 $S[i]=S[j]$, $S[i]=\neg S[j]$ 또는 $S[i]=-S[j]$가 되는 두 가지의 요소 S[i],

S[j]를 포함하지 않는다. 이 때, i와 j는 $i \neq j$ 이다.

- S에서 두 개 원소를 XOR하면 $\binom{512}{2}$개, 뺄셈을 하면 $2\times\binom{512}{2}$개의 다른 요소가 된다.
- S의 모든 두 요소는 적어도 4비트가 다르다.

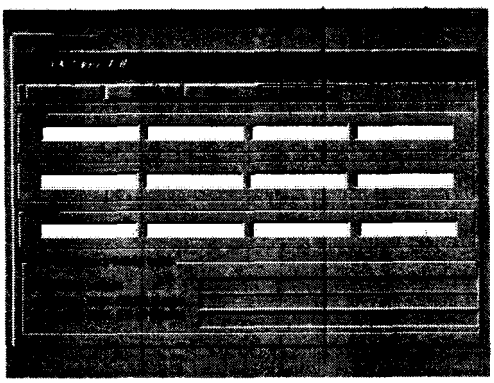
위와 같은 모든 특성들을 만족하게끔 설계된 S-Box S_1 테이블과 S_2 테이블의 내용들은 부록의 <표 2>와 <표 3>에서 나타내었다.

4. YK2 암호시스템의 구현

4.1 구현 환경

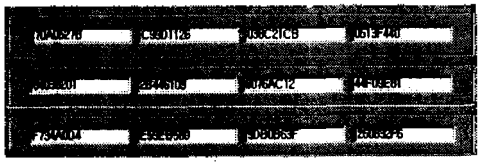
YK2 암호시스템의 구현에 사용된 시스템 환경과 구동시 초기화면의 구성은 다음과 같다.

- 시스템 : Pentium II 400MHz
- 시스템 소프트웨어 : Windows 98
- 컴파일러 : MS Visual C++ 6.0
- 메모리 : 128MB

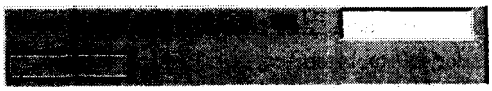


4.2 암/복호화 과정

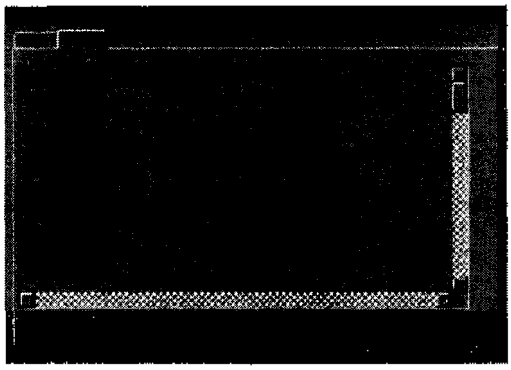
YK2 암호시스템에서의 암호화 과정은 평문과 암호문, 그리고 키의 크기가 각각 128비트이다. 이것은 임의의 128비트 평문을 입력한 후 "RUN" 탭을 클릭하여 암호화에 사용된 키와 암호문을 산출하는 구조로 설계되었다.



또한, 수행되는 회전 수를 조정할 수 있게 설계함으로서 프로그램에 의해 산출된 128비트의 암호문을 해독하는 과정에 있어서 전수조사 공격(exhaustive attack) 이나 차분 공격(differential attack) 등으로 암호해독을 수행할 때, 수작업으로 3회전 정도는 확인해 볼 수 있으므로, 화면 구성에서 "Round 회수를 입력하세요(1-32)"에 관한 메시지박스에서 회전 수를 입력하여 위의 공격 방법에 의한 산출된 결과와 비교해 볼 수 있다.

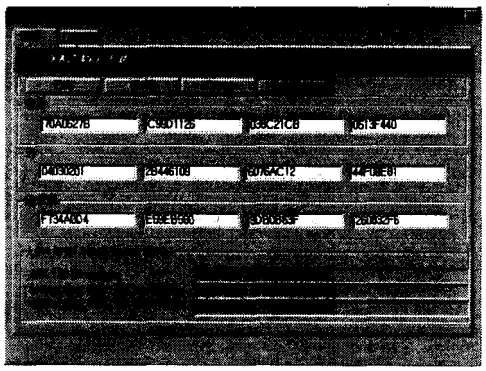


내부적으로 키는 키 스케줄링 알고리즘에 의해 키를 40개(K[0]~K[39])로 확장하여 암호문을 생성하는데 이용하였다. 이와 같은 내용을 단계별로 확인하기 위해서는 메인 화면에서, "디버깅" 탭을 이용한다.



그리고, 제안한 YK2 암호시스템에서 키 셋팅과 암/복호화 처리 속도를 산출하기 위해 "Low level block timing tests" 항목을 설계하였다.

최종적으로, 본 논문에서 제안한 "YK2 Block Cipher Algorithm Testing" 프로그램의 과정을 거쳐 생성된 최종화면은 다음과 같다.



본 논문에서는 복호화 과정에 대해서는 생략하는데, 복호화 과정 역시 암호화 과정에서 산출된 암호문과 동일한 키를 사용하여 "RUN" 탭을 클릭하면 평문을 산출할 수 있다.

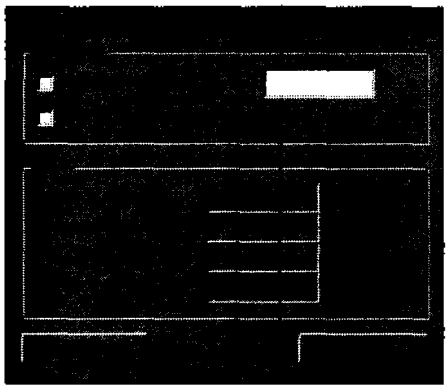
5. YK2 암호시스템의 비교 분석

본 논문에서는 제안한 YK2 암호시스템과 AES(Advanced Encryption Standard) 계열의 블록 암호 알고리즘들[13]에서 사용된 키 스케줄링 알고리즘에 근거한 키 처리 속도와 암/복호화 처리 속도를 각각 측정하여 비교 분석하였다.

테스팅을 위한 실험 환경은 YK2 암호시스템 구현에 사용된 환경과 동일하며, 실험 범위는 다음과 같이 설정하였다.

- 키 처리 속도는 임의의 키와 고정키를 사용하는 암호 알고리즘을 구별하여 동일한 방식으로 처리하였으며, 키에 대 한 키 스케줄링 알고리즘을 반복적으로 수행하여 이를 평균으로 산출한다.
- 암/복호화 처리 속도는 임의의 단위 블록을 반복적으로 암호화하는 과정과 이것을 반복적으로 복호화하는 과정을 수행하여 이를 평균으로 산출한다.
- 각 알고리즘에 대한 키 처리 속도와 암/복호화 처리 속도를 측정하기 위해 동일하게 10초간 수행하여 그 속도를 측정한다.

실험은 다음과 같이 개발한 알고리즘 속도 측정 툴에 의해 수행하였다.



위의 툴을 이용하여 다음과 같은 내용들을 실험하였다.

- 지정회수 사용 : 실험할 회수 지정
- 평문과 암호키를 각 회수마다 난수로 입력 : 평문과 암호키 값을 난수와 고 정 값으로 구분하여 선택
- 측정 회수 : 실제 실험에 측정된 회수 출력
- 서브키 계산 속도 : 키 생성 알고리즘 에 의한 서브키 생성 속도 출력
- 알고리즘 계산 속도 : 서브키를 제외한 알고리즘 자체 속도 출력
- 전체 계산 속도 : 전체 암/복호화 수행 속도 출력

다음의 <표 1>은 위의 실험 환경과 실험 범위를 바탕으로 AES 블록 암호 알고리즘과 YK2 암호시스템의 성능을 비교 분석한 것이다.

<표 1> 블록 암호 알고리즘의 성능 비교

알고리즘명	키 처리 속도	암·복호화 처리 속도
MARS	9.099 μsec	7.443 Mbps
RC6	5.058 μsec	11.304 Mbps
Rijndael	7.173 μsec	9.702 Mbps
Serpent	7.227 μsec	7.389 Mbps
Twofish	4.194 μsec	10.638 Mbps
YK2	3.872 μsec	6.867 Mbps

위의 실험 결과에서, YK2 암호시스템의 암/복호화 처리 속도가 6.867 Mbps로 다른 AES 암호 알고리즘 보다 빠르게 처리되며, 키 처리 속도가 3.872 μsec로 산출되므로 MAC(Message Authentication Code)이나 해쉬 함수의 블록 구축시 상당히 효율적으로 응용될 수 있다.

참고로 MARS와 Twofish 같은 경우는 미국 IBM에서 상당히 많은 연구를 진행하고 있으며, RC6는 RSA 부설 연구소에서 관련 연구를 수행하고 있다. 또한, Rijndael과 Serpent는 SPN 구조를 선호하는 유럽 국가들에서 표준 제정을 위해 초창기부터 연구가 활발히 진행되어 왔다.

6. 결론

인터넷의 급속한 확산에 따라 인터넷을 상거래에 이용하는 전자상거래가 활발히 이루어지고 있다. 또한, 인터넷의 영향력이 날로 증대되고 인터넷 사용자 수가 점점 많아지면서 전자상거래는 향후 일반적인 거래 형태로 정착될 전망이다. 따라서, 이러한 전자

상거래를 안전하게 실현하기 위해서는 암호 기술이 필수이며, 보안의 중요성이 점점 크게 부각되고 있다.

본 논문에서 제안한 YK2 암호시스템은 이러한 암호 기술에 적합하도록 안전성과 효율성을 고려하여 설계하였으며, 또한 일반적으로 암호화에 있어서 영향을 미치는 S-Box를 생성하기 위해 특성에 맞게 설계된 두 개의 S-Box S_1과 S_2를 사용하였고, 외부 공격에 의해 암호화에 사용된 키가 쉽게 발견되지 않도록 키 스케줄링 알고리즘을 설계하였다.

제안한 YK2 암호시스템은 민간분야의 전자상거래 상에서의 안전성과 신뢰성을 보장하기 위한 목적으로 개발했으며, 개인 및 기업의 컴퓨터 내 중요 정보보호나 전자우편 시스템에서의 메시지 암호화, 그리고 인터넷을 이용한 전자상거래, 특히 전자화폐나 전자지불 시스템에 적합하게 사용될 수 있다. 또한, 금융 네트워크에서의 정보보호나 유료 수업 내용의 보호가 필요한 가상교육 시스템, 위성방송사업과 연계되어 유료 방송 내용에 관한 암호화를 필요로 하는 CAS(Conditional Access System) 등에 유용하게 사용될 수 있을 것이라 예측된다.

향후, YK2 암호시스템의 보안 수준을 좀 더 향상시키기 위한 방법의 하나로 암호화에 영향을 미치는 키 스케줄링 알고리즘을 견고하게 생성하기 위해 키 확장 프로시져를 좀 더 보완하고, 동일한 구조로 설계된 다른 형태의 S-Box를 적용시켜 봄으로서 그것을 분석하고, 내부 함수 F 내부에 또 다른 함수를 하나 더 추가하여 계산 복잡도를 증가시켜 업그레이드된 버전의 더욱 견고한 암호 알고리즘을 구축할 계획이다.

참고문헌

[1] N.Heintze, D.Tygar, "Model Checking Electronic Commerce Protocol", ARPA contract F33615-93-1-1330, Nov. 1995.

[2] A.Froomkin, "The Essential Role of Trusted Third Parties in Electronic Commerce", Ver. 1.02 Oct. 1996.

[3] M.Green, "Role of Certificate Authrity in Internet Commerce", 1997.

[4] H.Feistel, "Cryptography and Computer Privacy", Scientific American, V. 228, N. 5, May 1973.

[5] H.Heys, S.Tavares, "Substitution-permutation networks resistant to differential and linear cryptanalysis", J. Cryptology, 9(1), 1996.

[6] M.Naor, O.Reingold, "On the construction of pseudo-random permutations: Luby-Rackoff Revisited", Proceeding of the 29th ACM Symposium on Theory of Computing, 1997.

[7] H.Sun, "Computer and Network Security", Lecture by Rivest of MIT. http://theory.lcs.mit.edu/~rosario/6.915/lecture6.

[8] J.Daemen, L.Knudsen, V.Rijmen, "The Block Cipher Square", Fast Software Encrytion , 4th International Workshop Proceedings, Springer-Verlag, 1997.

[9] W.Madryga, "A High Performance Encryption Algorithm", Computer Security: A Global Challenge, Elsevier Science Publishers, 1984.

[10] K.Aoki, K.Ohta, "Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability", IEICE Transactions Fundamentals of Elections, Communications and Computer Sciences, Vol. E80-A, No. 1, pp. 2-8, 1997.

[11] NIST, "Announcing Development of a Federal Information Standard for Advanced Encryption Standard", Federal Register, Vol. 62, No.1, Jan. 1997.

[12] E.Biham, A.Shamir, "Differential cryptanalysis of the data encryption standard", Springer-Verlag, 1993.

[13] NIST, 2nd AES Conference: AES Round2 Information, 1999. http://csrc.nist.gov/encryption/aes.

[14] J. Daeman, "Cipher and Hash Function Design", Ph.D. thesis, Katholieke Universiteit Leuven, Mar. 1995.

부록

<표 2> S_1 S-Box 테이블

0	09d0c479	1	28c8ffe0	2	84aa6c39	3	9dad7287	4	7dff9be3	5	d426836f	6	c96da1d4	7	7974cc93
8	85d0582e	9	2a4b5705	10	1ca16a62	11	c3bd279d	12	0f1f25e5	13	5160372f	14	c695c1fb	15	4d7ff1e4
16	ae5f6bf4	17	0d72ee46	18	ff23de8a	19	b1cf8e83	20	f14902e2	21	3e981e42	22	8bf53cb6	23	7f4bf8ac
24	83631f83	25	25970205	26	76afe784	27	3a7931d4	28	4f846450	29	5c64c3f6	30	210a5f18	31	c6986a26
32	28f4e826	33	3a60a81c	34	d340a664	35	7ea820c4	36	526687c5	37	7eddd12b	38	32a11d1d	39	9c9ef086
40	80f6e831	41	ab6f04ad	42	56fb9b53	43	8b2e095c	44	b68556ac	45	d2250b0d	46	294a7721	47	e21fb253
48	ae136749	49	e82aae86	50	93365104	51	99404a66	52	78a784dc	53	b69ba84b	54	04046793	55	23db5c1e
56	46cac1d6	57	2f2e8134	58	5a223942	59	1863cd5b	60	c190c6c3	61	07dfb846	62	6eb88816	63	2d0dcc4a
64	a4ccae59	65	3798670d	66	cbfa9493	67	4f481d45	68	eafc8ca8	69	db1129d6	70	b0449e20	71	0f5407fb
72	6167d9a8	73	d1f45763	74	4daa96c3	75	3bec5958	76	ababa014	77	b6ccd201	78	38d6279f	79	02682215
80	8f376cd5	81	092c237e	82	bfc56593	83	32889d2c	84	854b3e95	85	05bb9b43	86	7dcd5dcd	87	a02e926c
88	fae527e5	89	36a1c330	90	3412e1ae	91	f257f462	92	3c4f1d71	93	30a2e809	94	68e5f551	95	9c61ba44
96	5ded0ab8	97	75ce09c8	98	9654f93e	99	698c0cca	100	243cb3e4	101	2b062b97	102	0f3b8d9e	103	00e050df
104	fc5d6166	105	e35f9288	106	c079550d	107	0591aee8	108	8e531c74	109	75fc3578	110	2f6d829a	111	f60b21ac
112	95e8eb8d	113	6699486d	114	901d7d9b	115	fd6d6e31	116	1090acef	117	e0670dd8	118	dab2c692	119	cd6d4365
120	e5393514	121	3af345f0	122	6241fc4d	123	460da3a3	124	7bcf3729	125	8bf1d1e0	126	14aac070	127	1587ed55
128	3afd7d3e	129	d2f29e01	130	29a9d1f6	131	efb10c53	132	cf3b870f	133	b414935c	134	664465ed	135	024acac7
136	59a744c1	137	1d2936a7	138	dc580aa6	139	cf574ca8	140	040a7a10	141	6cd81807	142	8a98be4c	143	accea063
144	c33e92b5	145	d1e0e03d	146	b322517e	147	2092bd13	148	386b2c4a	149	52c8dd58	150	58656dfb	151	50820371
152	41811896	153	e337ef7e	154	d39fb119	155	c97f0df6	156	68fca01b	157	a150a6e5	158	55258962	159	eb6ff41b
160	d7c9cd7a	161	a619cd9e	162	bcf09576	163	2672c073	164	f003fb3c	165	4ab7a50b	166	1484126a	167	487ba9b1
168	a64fc9c6	169	f6957d49	170	38b06a75	171	dd805fcd	172	63d094cf	173	f51c999e	174	1aa4d343	175	b8495294
176	ce9f8e99	177	bffcdf770	178	c7c275cc	179	378453a7	180	7b21be33	181	397f41bd	182	4e94d131	183	92cc1f98
184	5915ea51	185	99f861b7	186	c9980a88	187	1d74fb5f	188	b0a495f8	189	614deed0	190	b5778eea	191	5941792d
192	fa90c1f8	193	33f824b4	194	c4965372	195	3ff6d550	196	4ca5fec0	197	8630e964	198	5b3fbbd6	199	7da26a48
200	b203231a	201	04297514	202	2d639306	203	2eb13149	204	16a45272	205	532459a0	206	8c5f4872	207	f966c7d9
208	07128dc0	209	0d44db62	210	afc8d52d	211	06316131	212	d838c7ce	213	1bc41d00	214	3a2e8c0f	215	ea83837e
216	b9847 37d	217	13ba4891	218	c4f8b949	219	a6d6acb3	220	a215cdcc	221	8359838b	222	6bd1aa31	223	f579dd52
224	21b93f93	225	f5176781	226	187dfdde	227	e94acb76	228	2b38fd54	229	431de1da	230	ab394825	231	9ad3048f
232	dfea32aa	233	659473e3	234	623f7863	235	f3346c59	236	ab3ab685	237	3346a90b	238	6b56443e	239	c6de01f8
240	8d421fc0	241	9b0ed10c	242	88f1a1e9	243	54c1f029	244	7dcad57b	245	8d7ba426	246	4cf5178a	247	551a7cca
248	1a9a5f08	249	fcd651b9	250	25605182	251	e11fc6c3	252	b6fd9676	253	337b3027	254	b7c8eb14	255	9c5fd030

<표 3> S_2 S-Box 테이블

0	6b57e354	1	ad913cf7	2	7e16688d	3	58872a69	4	2c2fc7df	5	e389ccc6	6	30738df1	7	0824a734
8	e1797a8b	9	a4a8d57b	10	5b5d193b	11	c8a8309b	12	73f9a978	13	73398d32	14	0f59573e	15	e9df2b03
16	e8a5b6c8	17	848d0704	18	98df93c2	19	720a1dc3	20	684f259a	21	943ba848	22	a6370152	23	863b5ea3
24	d17b978b	25	6d9b58ef	26	0a700dd4	27	a73d36bf	28	8e6a0829	29	8695bc14	30	c35b3447	31	933ac568
32	8894b022	33	2f511c27	34	ddfbcc3c	35	006662b6	36	117c83fe	37	4e12b414	38	c2bca766	39	3a2fec10
40	f4562420	41	55792e2a	42	46f5d857	43	ceda25ce	44	c3601d3b	45	6c00ab46	46	efac9c28	47	b3c35047
48	611dfee3	49	257c3207	50	fdd58482	51	3b14d84f	52	23becb64	53	a075f3a3	54	088f8ead	55	07adf158
56	7796943c	57	facabf3d	58	c09730cd	59	f7679969	60	da44e9ed	61	2c854c12	62	35935fa3	63	2f057d9f
64	690624f8	65	1cb0bafd	66	7b0dbdc6	67	810f23bb	68	fa929a1a	69	6d969a17	70	6742979b	71	74ac7d05
72	010e65c4	73	86a3d963	74	f907b5a0	75	d0042bd3	76	158d7d03	77	287a8255	78	bba8366f	79	096edc33
80	21916a7b	81	77b56b86	82	951622f9	83	a6c5e650	84	8cea17d1	85	cd8c62bc	86	a3d63433	87	358a68fd
88	0f9b8d3c	89	d6aa295b	90	fe33384a	91	c000738e	92	cd67eb2f	93	e2eb6dc2	94	97338b02	95	06c9f246
96	419cf1ad	97	2b83c045	98	3723f18a	99	cb5b3089	100	160bead7	101	5d494656	102	35f8a74b	103	1e4e6c9e
104	000399bd	105	67466880	106	b4174831	107	acf423b2	108	ca815ab3	109	5a6395e7	110	302a67c5	111	8bdb446b
112	108f8fa4	113	10223eda	114	92b8b48b	115	7f38d0ee	116	ab2701d4	117	0262d415	118	af224a30	119	b3d88aba
120	f8b2c3af	121	daf7cf70	122	cc97d3b7	123	e9614b6c	124	2baebff4	125	70f687cf	126	386c9156	127	ce092ee5
128	01e87da6	129	6ce91e6a	130	bb7bcc84	131	c7922c20	132	9d3b71fd	133	060e41c6	134	d7590f15	135	4e03bb47
136	183c198e	137	63eeb240	138	2ddbf49a	139	6d5cba54	140	923750af	141	f9e14236	142	7838162b	143	59726c72
144	81b66760	145	bb2926c1	146	48a0ce0d	147	a6c0496d	148	ad43507b	149	718d496a	150	9df057af	151	44b1bde6
152	054356dc	153	de7ced35	154	d51a138b	155	62088cc9	156	35830311	157	c96efca2	158	686f86ec	159	8e77cb68
160	63e1d6b8	161	c80f9778	162	79c491fd	163	1b4c67f2	164	72698d7d	165	5e368c31	166	f7d95e2e	167	a1d3493f
168	dcd9433e	169	896f1552	170	4bc4ca7a	171	a6d1baf4	172	a5a96dcc	173	0bef8b46	174	a169fda7	175	74df40b7
176	4e208804	177	9a756607	178	038e87c8	179	20211e44	180	8b7ad4bf	181	c6403f35	182	1848e36d	183	80bdb038
184	1e62891c	185	643d2107	186	bf04d6f8	187	21092c8c	188	f644f389	189	0778404e	190	7b78adb8	191	a2c52d53
192	42157abe	193	a2253e2e	194	7bf3f4ae	195	80f594f9	196	953194e7	197	77eb92ed	198	b3816930	199	da8d9336
200	bf447469	201	f26d9483	202	ee6faed5	203	71371235	204	de425f73	205	b4e59f43	206	7dbe2d4e	207	2d37b185
208	49dc9a63	209	98c39d98	210	1301c9a2	211	389b1bbf	212	0c18588d	213	a421c1ba	214	7aa3865c	215	71e08558
216	3c5cfcaa	217	7d239ca4	218	0297d9dd	219	d7dc2830	220	4b37802b	221	7428ab54	222	acce0347	223	4b3fbb85
224	692f2f08	225	134e578e	226	36d9e0bf	227	ae8b5fcf	228	edb93ecf	229	2b27248e	230	170eb1ef	231	7dc57fd6
232	1e760f16	233	b1136601	234	864e1b9b	235	d7ea7319	236	3ab871bd	237	cfa4d76f	238	e31ba782	239	0dbcb469
240	abb96061	241	5370f85d	242	ffb07e37	243	da30d0fb	244	ebc977b6	245	0b98b40f	246	3a4d0fe6	247	df4fc26b
248	159cf22a	249	c298d6e2	250	2b78ef6a	251	61a94ac0	252	ab561187	253	14eea0f0	254	df0d4164	255	19af70ee

저자소개

서장원 (e-mail : jwsuh@duck.snut.ac.kr)

1992년 서울산업대학교 전자계산학과를 졸업하고, 1992년부터 1998년까지 서울산업대학교 전자계산소에서 근무하였다. 1996년 숭실대학교 정보과학대학원 전산공학과를 졸업하고, 2000년 숭실대학교에서 공학박사 학위를 취득하였다. 1997년부터 1998년까지 숭실대학교 학부 강사를 역임하였고, 1995년부터 2000년까지 육군사관학교 전산학과 컴퓨터 교관과 신흥대학 컴퓨터정보계열 강사를 역임하였다. 1998년부터 현재까지 서울산업대학교 공동실험실습센터 시스템지원실에서 근무하고 있다. 연구 관심분야로는 암호시스템, 정보보호 이론, 네트워크 보안, 전자상거래 보안 솔루션 등이다.