네트워크 보안을 위한 암호시스템 연구

서장원[0]
동서울대학 전산과
louis@duck.snut.ac.kr

A Study of Cipher System for Network Security

Jang-Won Suh[0]
Dept. of Computer Science, Dongseoul College

요 약

최근 인터넷의 폭발적인 증가로 외부 침입 문제를 해결하기 위한 네트워크 보안 대책의 중요성이 더욱 부각되고 있으며, 네트워크 서비스를 제공하는 자와 요구하는 자의 측면에서 보안 요구가 가장 하위의 IP에 요구되고 있다.

본 논문에서는 네트워크 상의 C/S 보안과 밀접하게 연계된 하위 레벨 API의 보안 요소 중 암호화를 이용하여 메시지의 보안을 설정하기 위한 체계를 단계별로 설정하고, 계산 복잡도와 신뢰성을 증가시킴으로써 이를 실제 네트워크에 활용하는 형태에 관해 연구하였다.

1. 서 론

최근 들어, 네트워크 기술이 비약적으로 발전하면서 온라인 상에서 다루어지는 정보와 자원의 양도 기하급수적으로 증가하고 있는 실정이다. 이러한 네트워크 기술의 발전은 많은 양의 정보나 자원을 공유하여 사용할 수 있다는 장점을 갖고 있지만 이에 따른 부작용도 동시에 유발하고 있다. 즉, 개인 정보의 유출이라든가 기관 또는 학내의 정보망에 대한 침입으로 인한 피해, 그리고 일상적으로 행해지는 네트워크 거래 등에 보안 문제의 필요성을 심각하게 고려하게 되었다.

따라서, 이러한 문제점들을 해결하기 위해서는 네트워크 상의 정보처리나 네트워크 거래에 대한 안전성을 확보할 수 있는 네트워크 보안 기술이 필수 선결요소라 할 수 있겠다.

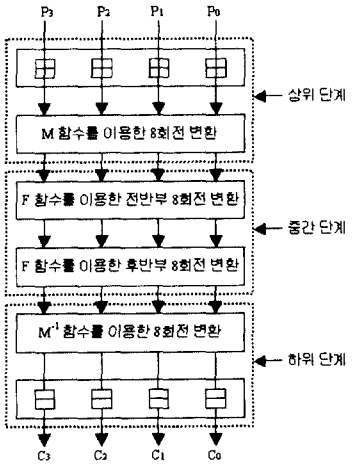
본 논문에서 제안한 새로운 암호시스템은 비선형 변환과 선형 변환의 적절한 조합에 의해 설계했으며, 암호시스템의 전체 구조는 SPN 구조를 적용하여 설계하였으며[1], 내부 함수 F에서는 데이타 블록의 좌/우 측면에 교대로 비선형 변환을 적용시키는 Type-3 Feistel 구조를 적용하여 설계하였다[2].

2. 암호시스템의 설계

2.1 전체구성

제안한 암호시스템은 128비트 평문을 4개의 32비트 블록 P_0, P_1, P_2, P_3로 분할하여 입력하고, 이를 토대로 키를 부가하여 입력 내용을 변환시키는 8라운드의 상위 단계와 실제적인 암호화를 수행하는 16라운드의 중간 단계, 그리고 상위 단계를 역으로 수행한 후 키를 감소시키는 8라운드 하위 단계를 거친 후에 4개의 32비트 블록 C_0, C_1, C_2, C_3를 산출한 후, 최종적으로 이것들을 조합하여 128비트 암호문을 생성하는 구조로 설계하였다. 암호시스템의 전체 구조는 <그림 1>과 같다.

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

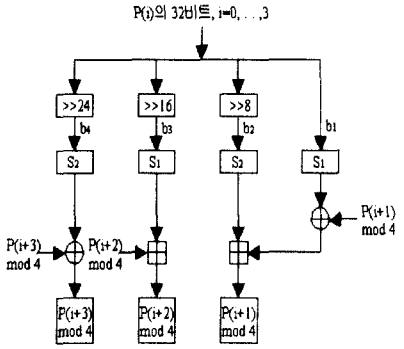


<그림 1> 암호시스템의 전체 구조

(1) 상위 단계

여기서는 4개의 각 평문 블록들에 키를 더한 후 8라운드의 Type-3 Feistel 연산을 수행한다. 각 라운드에서는 하나의 블록(초기블록)을 사용하여 다른 3개의 블록(목표블록)을 변경시킨다. 즉, <그림 2>와 같이 M 함수를 이용하여 초기 블록의 32비트를 우측으로 8비트씩 이동하면서 그 때마다

최하위 8비트를 2개의 S-Box S_1과 S_2에 대한 인덱스로 사용하여 이에 대응되는 S-Box의 내용들을 다른 3개의 목표 블록들과 더하거나 XOR 시킨다.

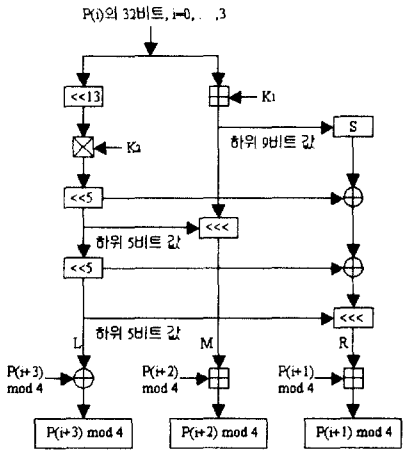


<그림 2> M 함수의 구조

그런 후에, 다음 회전을 위하여 4개의 평문 블록을 회전 이동 시킨다. 위와 같은 순서를 두 번 반복해서 8라운드 수행한 후 산출된 결과를 중간 단계로 이동시킨다.

(2) 중간 단계

암호시스템의 핵심 단계로서 16라운드 Type-3 Feistel 연산을 수행한다. 각 라운드에서는 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 3>과 같다.



<그림 3> 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 8라운드와 후반부 8라운드를 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 8라운드의 각 라운드에서는 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 더해준다.

추가로, 매 라운드가 끝난 후에는 다음 라운드를 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 16라운드를 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 이동 시킨다. 위와 같은 순서를 반복적으로 8라운드 수행한 후 그 결과를 최종적으로 암호문으로 산출하게 된다.

2.2 키 스케쥴링

키 스케쥴링은 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.

[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
```

2.3 S-Box

S-Box는 대개의 암호시스템에서 사용되는 비선형 대입 연산을 수행하는 비선형 함수로서, 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 S-Box S_1, S_2는 의사난수 형태로 생성되어 만들어졌다. S-Box 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_3는 고정 상수 값이다[3].

새로운 암호시스템에서 S-Box는 다음과 같은 특성들을 만족하게끔 설계하였다[4].

- S-Box는 32비트가 모두 0이거나 또는 모두 1인 것은 포함하지 않는다.
- S-Box를 S_1과 S_2로 나눌 때, S_1과 S_2에서 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.
- S의 모든 두 요소는 적어도 4비트가 다르다.

3. 암호시스템의 비교 분석

본 논문에서는 제안한 암호시스템과 AES 계열의 블록 암호 알고리즘들[5]에서 사용된 키 스케쥴링 알고리즘에 근거한 키 처리 속도와 암/복호화 처리 속도를 각각 측정하여 비교 분석하였다. 테스팅을 위한 실험 환경은 다음과 같다.

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

실험 범위는 다음과 같이 설정하였다.

- 키 처리 속도는 임의의 키와 고정 키를 사용하는 암호 알고리즘을 구별하여 동일한 방식으로 처리하였으며, 키에 대한 키 스케쥴링 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 속도 측정을 위해 동일하게 10초간 수행하여 그 속도를 측정한다.

다음의 <표 1>은 위의 실험환경과 범위를 바탕으로 각각의 AES 블록 암호 알고리즘과 제안한 암호시스템의 성능을 비교 분석한 것이다.

<표 1> 암호시스템의 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
New	3.872 μsec	6.867 Mbps

위의 실험 결과에서, New 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.

4. 결 론

네트워크의 확산에 따라 정보 공유가 폭넓게 이루어지고 있다. 또한, 네트워크 상의 사용자 수가 많아지고 처리되어지는 데이터의 양이 증가할수록 안전한 데이터 처리와 보호를 위한 제반 장치가 필요한데, 이를 위해서는 보안을 강화시킬 수 있는 견고한 암호 알고리즘의 설계가 필수적이다.

본 논문에서 제안한 암호시스템은 이러한 암호 체계에 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 스케쥴링 알고리즘을 설계하였다.

향후, 제안한 암호시스템의 보안 수준을 좀 더 향상시키기 위한 방법의 하나로 암호화에 영향을 미치는 키 스케쥴링 알고리즘을 견고하게 생성하기 위해 키 확장 프로시져를 좀 더 보완하고, 동일한 구조로 설계된 다른 형태의 S-Box를 적용시켜 봄으로서 그것을 분석하고 계산 복잡도를 증가시켜 업그레이드된 버전의 더욱 견고한 암호 알고리즘을 구축할 계획이다.

참고문헌

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

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

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

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

[5] NIST, AES Conference: AES Round2 Information, 1999.