情報保護學會論文誌
第13券 第1號, 2003. 2

SEED 구현 적합성 검증 시스템에 관한 연구

김 역*, 정 창 호*, 장 윤 석*, 이 상 진*, 이 성 재**

On the SEED Validation System

Yeog Kim*, Changho Jung*, Yunseok Jang*, Sangjin Lee*, Sungjae Lee**

요 약

본 논문에서는 정보보호 제품의 주요한 역할을 담당하는 암호 알고리즘의 구현 적합성에 대해 논의한다. 암호 알고리즘 구현 적합성 평가는 지정된 표준에 맞게 정확하게 구현했는가에 대한 평가이다. 따라서 해당 암호 알고리즘 기능별 또는 절차에 따라 알고리즘 평가가 수행된다. 본 논문에서 제시한 암호 알고리즘 평가 검증은 국내 표준인 SEED 알고리즘을 그 대상으로 하며, 알고리즘의 기능에 따라 평가를 수행한다. 제안한 검증 시스템은 SEED 알고리즘 구현물에 대해 충분히 테스트하기 위해 필요한 테스트 벡터를 생성하여 이용하는 테스트와 검증의 정확성을 높이기 위해 임의 데이터를 이용한 테스트를 제공한다. 제안한 검증 시스템은 SEED를 이용한 정보보호 제품에 모두 적용 가능하므로 각종 암호제품 평가 및 인증에 활용될 수 있다.

ABSTRACT

In this paper, we discuss a validation test for cryptographic algorithms. The cryptographic algorithms decide on the security and the confidence of a security system protecting sensitive information. So, the implementation of cryptographic algorithms is very critical of the system. The validation test specifies the procedures involved in validating implementations of the cryptographic standards and provides conformance testing for components or procedures of the algorithm. We propose a SEED Validation System(SVS) to verify that the implementation correctly performs the SEED algorithm. The SVS is composed of two types of validation tests, the Known Answer test and the Monte Carlo test. The System generates the testing data for the Known Answer tests and the random data for the Monte Carlo tests. This system can be used to validate and certify the cryptographic product.

Keyword : *암호알고리즘 평가, SEED, CMVPction*

Ⅰ. 서 론

초기의 정보보호 기술은 군용 및 외교통신 등 국가 기밀 분야에서 사용되었다. 따라서 암호 기술의 연구개발은 국가 주도로 이루어졌다. 정보보호 기술은 정보통신 기술 발달로 인하여 금융, 의료, 경제 등 사회 각 분야에서 급속히 확대되었고, 민간에서도 정보보호 시스템에 대한 수요가 폭증하게 되었으며, 이에 민간에서 개발한 정보보호 기술을 국가기관에서 사용하는 상황이 도래되고 있다. 따라서 신뢰성 있는 정보보호 기술 및 제품에 대한 요구가 높아지고 있고, 이러한 기술 및 제품을 평가하기 위해 보안 검증기술의 필요성이 증대하고 있다.

보안 기술은 안전하고 신뢰할 수 있는 통신 환경

* 고려대학교 정보보호기술연구센터(CIST)({yeog, zangho, axis}@cist.korea.ac.kr, sangjin@korea.ac.kr)
** 한국정보보호진흥원(sjlee@kisa.or.kr)

구축에 있어 반드시 필요한 기반 기술이다. 이러한 기술 하에 제작된 보안 시스템이 안전성과 신뢰성을 보장하지 못한다면, 해당 시스템 사용자나 기업체에 돌아오는 불이익은 온라인 특성으로 인해 오프라인에서 예상하기 힘들 정도로 빠른 시간 안에 막대한 영향을 미친다.

보안 시스템에 사용되는 기술은 여러 가지 암호 기술을 기반으로 제공되고 있고, 이러한 암호 기술은 복잡하고 다양한 기능을 포함하고 있다. 따라서 구현에서 미세한 취약점이 발생할 수 있고, 이러한 취약점은 전체 시스템에 치명적인 영향을 줄 수가 있다. 그러므로 암호 기술의 정확한 구현에 많은 주의가 요구된다.

본 논문에서는 이러한 암호 기술의 정확한 구현 검증의 필요성과 현재 진행되고 있는 CMVP(Cryptographic Module Validation Program) 소개를 통해 암호 알고리즘의 구현 무결성을 검증할 수 있는 평가 방법을 소개하며, 이에 기반하여 국내 표준 블록 암호인 SEED의 구현 검증을 위한 테스트 벡터를 제시하고자 한다.

II. 관련 연구

2.1 CMVP

CMVP는 1995년 NIST와 CSE에 의해 추진되었으며, 개발된 암호 모듈과 암호 시스템이 표준에 적합하고 안전하게 구현되었는가를 평가한다.[1] CMVP는 암호 모듈 평가를 위한 요구사항으로 FIPS 140-1[2]을 제안하였고, 이를 개선하여 FIPS 140-2[3]를 작성하였다. 또한 AES (Advanced Encryption Standard)[9], DES (Data Encryption Standard)[10], Triple-DES[10], Skipjack[11] 알고리즘에 대한 평가 시스템을 제시하고, NVLAP[1]를 통해 지정한 CMT Lab[1]에 의해 보안 시스템 검증을 수행하고 있다.

관련 제품을 제작한 업체들은 CMT Lab에 암호 모듈의 표준 적합성 테스트를 의뢰하고, CMVP에 의해 그 결과를 평가받음으로써 보안 레벨이 결정된다. 보안 레벨은 Level1, Level2, Level3, Level4로써 평가되며, Level1이 가장 낮은 단계이고 Level4가 가장 높은 단계이다. 각 단계별로 요구사항이 누적되어 처리된다.

현재 CMVP에서 제시한 표준 적합성 테스트에 의해 검증 받은 제품은 암호 모듈과 암호 알고리즘 평가로 다음과 같이 분류할 수 있으며, 인증된 제품 목록이 NIST 홈페이지에 게시되어 있다.

- FIPS 140-1
- FIPS 140-2
- AES Validation
- Triple-DES Validation
- DES Validation
- Skipjack Validation
- DSA Validation
- SHA-1 Validation
- MAC Validation
- FIPS 171 (ANSI X9.17 Key Management) Validation

2.2 MOVS와 AESAVS[9]

NIST에서 제시한 MOVS(Modes of Operation Validation System)[5]는 DES 알고리즘과 Skjpjack 알고리즘의 표준 적합성 평가 수행을 검증한다. DES 알고리즘인 경우 FIPS PUB 46-3[10]에 따라 구현되었는가를 Skipjack 알고리즘인 경우에는 FIPS 185[12]에서 참조한 Skipjack Algorithm[11]에 적합하게 구현되었는가를 확인하는 것이다. MOVS에서 제시하는 검증 방법은 Known Answer Tests와 Modes Tests로 나뉜다. 이때 검증받는 구현물을 IUT(Implementation Under Test)라 칭한다.

AESAVS(The Advanced Encryption Standard Algorithm Validation Suite)는 FIPS 197에 따라 AES를 구현했는가에 대해 평가한다. AESAVS에서 제시하는 검증 방법은 Known Answer Tests, Multi-block Message Test, Monte Carlo Test로 나뉜다.

2.2.1 Known Answer Tests

Known Answer Tests는 주어진 입력값에 대한 출력값과 예상 출력값에 대한 비교로 평가가 이루어진다. IUT는 주어진 입력값들을 이용해서 결과값을 산출한다. 따라서 IUT는 적합한 파라메터 입력이 가능해야 한다. 이러한 IUT는 산출된 결과값과 본래의 출력값을 비교하여 IUT의 구현 정확성을 검증한다. Known Answer Tests는 해당 테스트에 통과하기 위한 악의적인 목적으로 제작된 경우 정확한 구현 적합성 평가가 힘들다.

Known Answer Tests는 테스트하려는 알고리즘의 설계 특성에 따라 알고리즘을 이루는 구성요소별로 구현 적합성을 검증할 수 있다.

2.2.2 Modes Test와 Monte Carlo Test

Modes Test와 Monte Carlo Test는 의사 난수 데이

터를 사용하여 추출한 입력값들이 주어지고, 해당 입력에 의한 출력값과 예상 출력값에 대한 비교로 평가가 이루어진다. Modes Test와 Monte Carlo Test는 구현 과정에서 또는 구현물 운영에서 생길 수 있는 오류의 유무를 판단할 수 있다. 또한 Modes Test와 Monte Carlo Test를 통해 IUT가 Known Answer Tests만을 위해 설계된 것이 아님을 확인할 수 있다.

2.2.3 Multi-block Message Test

Multi-block Message Test는 여러 개의 메시지 블록에 대한 실행 가능성을 알아보기 위한 테스트이다. AESAVS에서 실시한 테스트로 테스트를 위해 여러 개의 블록으로 이루어진 평문 입력이 요구된다.

Ⅲ. SEED 구현물 검증을 위한 분석

SEED 알고리즘의 전체 구조는 Fesitel 구조로 이루어져 있다. 128 비트의 평문 블록 단위당 128 비트 키로부터 생성된 32비트의 라운드 키 16개를 입력받아 총 16라운드를 거쳐 128 비트 암호문 블록을 출력한다.

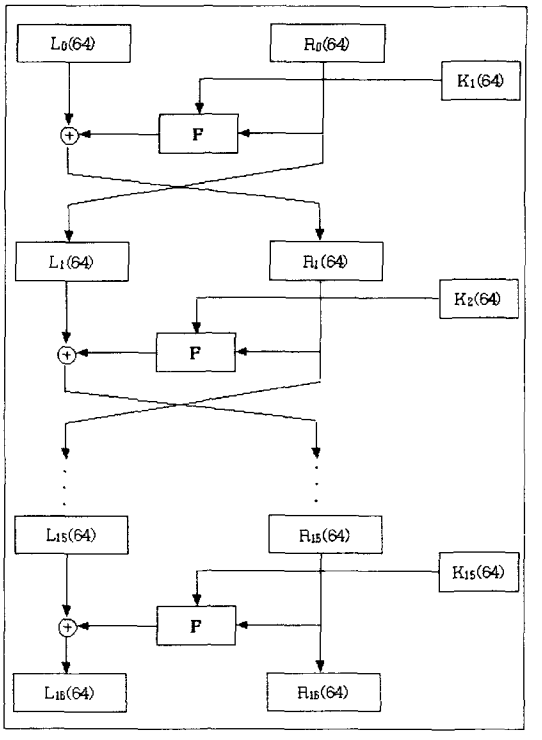
알고리즘 분석을 위해 먼저 알고리즘에서 사용되는 연산을 나열하면 다음과 같다.

- addition in modular 2^{32}
- subtraction in modular 2^{32}
- *bitwise exclusive OR*
- bitwise AND
- *left circular rotation by n bits*
- right circular rotation by n bits
- *concatenation*

SEED 알고리즘은 입력된 128비트 평문 블록을 64비트 두 개의 블록 L_{i-1}과 R_{i-1}로 나누어 시작한다. 이때 R_{i-1}는 키 생성 알고리즘에 의해 생성된 32비트의 라운드 키와 함께 F 함수의 입력으로 처리되며, F 함수의 출력과 L_{i-1} 블록의 값을 bitwise exclusive OR 연산을 수행하여 다음 라운드의 R_i 블록으로 처리된다. 이때 F 함수는 라운드키 생성과정을 통해 생성된 32비트의 라운드키를 입력으로 받아 처리한다.

S-box와 G 함수는 평문동작과정에서 사용되는 F 함수와 라운드키 생성과정에 포함된다.

SEED의 구현은 각 단계별로 표준에서 명시한 대로 연산을 수행하여 정확한 결과를 산출할 수 있어야 한다.



〔그림 1〕 SEED 전체 구조도

3.1 평문동작과정

평문동작과정은 64비트의 블록이 수행해야 하는 bitwise exclusive OR 연산과 F 함수, G 함수 수행에 해당된다. F 함수와 G 함수는 서로 독립적인 처리로 구별한다. 따라서 bitwise exclusive OR 연산의 정확한 수행이 평문동작과정에 해당된다.

3.2 F 함수 동작과정

F 함수 동작과정은 64비트의 평문 블록과 각 라운드에서 생성된 32비트의 부분키에 대한 일련의 연산을 수행한다. 이때 사용되는 연산은 bitwise exclusive OR 연산, addition in modular 2^{32} 그리고 G 함수이다. 따라서 F 함수 동작과정은 두 개의 연산을 정확하게 처리하는가에 해당된다.

3.3 G 함수 동작과정

G 함수 동작과정은 G 함수 입력 값 계산에 사용되는 연산과 G 함수 내에서 수행하는 연산 실행으로 나누어 볼 수 있다. 라운드키 생성과정에서 수행하는

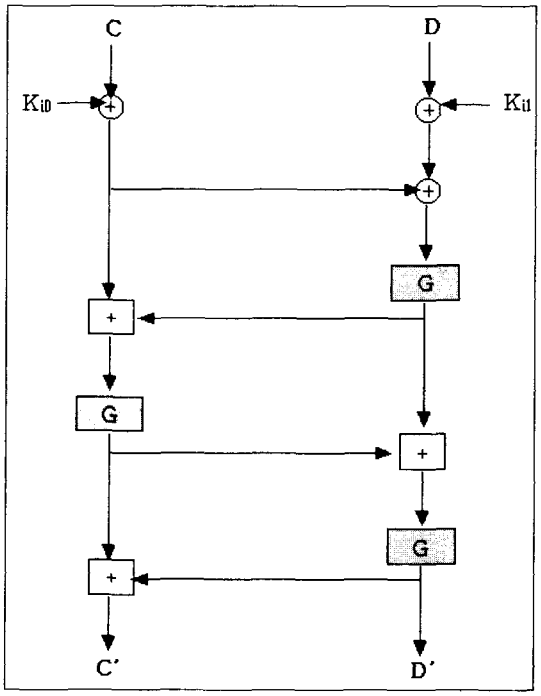
G 함수의 경우에는 입력 값 생성을 위해 주어진 상수 16개 KC_i[7]와 addition in modular 2^{32}, subtraction in modular 2^{32} 연산을 수행하며, G 함수 내에서 수행하는 연산은 S-box의 결과 값과 네 개의 상수 m0, m1, m2, m3와 bitwise AND 연산에 해당된다.

3.4 S-box 입출력과정

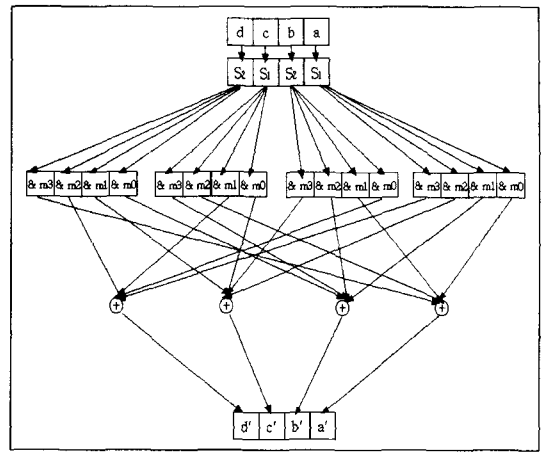
SEED는 두 개의 S-box를 갖는다. S-box 입출력과정은 각 S-box가 입력이 주어졌을 때 지정한 값을 출력한다.

3.5 라운드키 생성과정

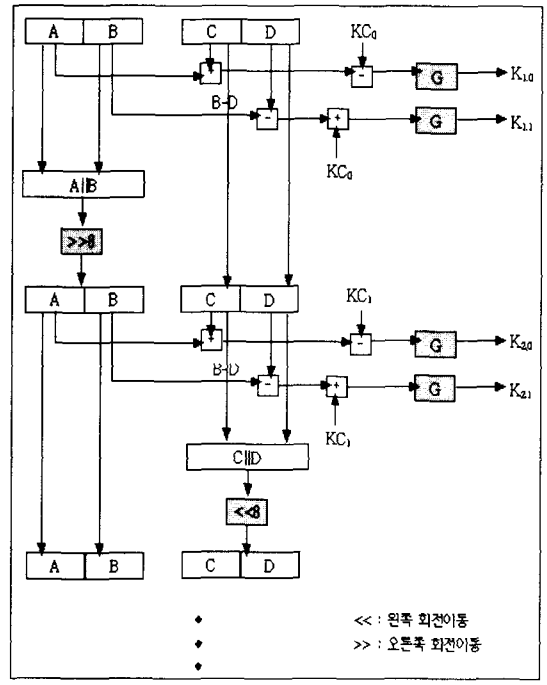
라운드키 생성과정은 각 라운드의 부분키를 생성한다. G 함수의 입력을 위해 addition in modular 2^{32}와 subtraction in modular 2^{32}를 수행하고 그 외에 left circular rotation by n bits, right circular rotation by n bits 그리고 concatenation 연산을 수행한다. G 함수 동작과정은 독립적으로 해당 처리에 대해 다룰 예정이므로 left circular rotation by n bits, right circular rotation by n bits 그리고 concatenation 에 대한 정확한 연산 처리가 라운드키 생성과정에 해당된다.



〔그림 2〕 F 함수 동작과정



〔그림 3〕 G 함수 동작과정



〔그림 4〕 라운드키 생성과정

Ⅳ. SEED 구현물 검증

SEED 구현물 검증은 표준에서 명시한 각 부분의 연산을 정확하게 구현했는가에 대한 적합성 평가로써 Known Answer Tests와 Modes Tests로 이루어진다.

Known Answer Tests는 SEED 알고리즘 구현 단계에 따라 연산 수행 여부를 정확히 판단할 수 있는 테스트 벡터를 이용하며, Modes Tests는 구현물의 비정상적인 동작 오류나 가변 데이터 사용에 대한 구현

의 신뢰성 테스트를 위해 수행한다. SEED Validation System을 이용한 테스트는 다음과 같이 이루어진다.

① 평문동작과정 테스트
② F 함수 동작과정 테스트
③ G 함수 동작과정 테스트
④ S-box 입출력과정 테스트
⑤ 라운드키 생성과정 테스트
⑥ Modes 테스트

각 테스트는 독립적으로 수행되며, 각 테스트에 명시된 테스트 벡터 값에 의해 연산의 정확한 수행 여부를 판단할 수 있다.

본 논문에서 제안하는 SEED 검증 시스템을 SVS (SEED Validation System)이라 칭하며, SVS에서 사용하는 비트열의 경우 가장 왼쪽 비트를 최하위 비트, 가장 오른쪽의 비트를 최상위 비트로 표현한다.

4.1 평문 동작과정 테스트

평문동작과정 테스트는 평문 128비트가 입력되어 64비트 블록으로 나뉘어져 서로 교차하면서 bitwise exclusive OR 연산을 정확하게 수행하는 가에 대해 각 비트별로 확인한다.

비트별로 연산을 정확하게 수행했는가를 확인하기 위하여 키 값과 평문을 모두 0으로 초기화한 후 테스트를 실행하는데 이때 평문은 오른쪽 최상위 비트부터 1의 값을 차례로 지정하면서 테스트한다.

테스트 벡터 값 지정에서 최상위 비트부터 값을 지정한 이유는 SEED 알고리즘의 경우 가장 오른쪽 비트를 최상위 비트로 사용했기 때문이다. 이와 반대로 AES 알고리즘은 가장 오른쪽 비트의 값을 최하위 비트로 사용하였다.

테스트 횟수는 128번으로 테스트 동안 평문 128비트의 각 비트별로 값을 확인할 수 있다. 테스트 횟수에 따라 평문은 다음과 같이 변화한다.

```
1회  : 00000000000000000000000000000001(16)
2회  : 00000000000000000000000000000003(16)
3회  : 00000000000000000000000000000007(16)
4회  : 0000000000000000000000000000000f(16)
5회  : 0000000000000000000000000000001f(16)
.....................................
127회: 7fffffffffffffffffffffffffffffff(16)
128회: ffffffffffffffffffffffffffffffff(16)
```

〔표 1〕 평문동작과정 테스트

```
SVS: 초기화
  KEY =00000000000000000000000000000000(16)
  PT1 =00000000000000000000000000000000(16)
  KEY, PT1를 IUT에 전송한다.

IUT:
  FOR i=1 to 128
  {
    PT1의 128-i번째 비트에 1의 값을 지정한다.
    PT1와 KEY를 입력하여 암호화 알고리즘을
          수행, 암호문 CT1를 산출한다.
    i, KEY, PT1, CT1를 SVS에 전송한다.
  }

SVS: 반복을 통해 얻은 결과 값과 예상 기대값을 비교
하여 검증의 성공/실패 여부를 판단한다.
```

이를 알고리즘으로 나타내면 [표 1]과 같다.

각 횟수별 입력과 출력 테스트 벡터는 부록의 "(1) 평문 동작과정 테스트 벡터"를 참조한다.

4.2 F 함수 동작과정 테스트

F 함수 동작과정 테스트는 F 함수의 구현 적합성을 테스트한다. F 함수는 G 함수와 bitwise exclusive OR 연산과 addition in modular 2^{32}를 수행한다. G 함수는 독립적으로 처리하므로 F 함수 처리 테스트에서 제외되고 bitwise exclusive OR 연산 테스트도 평문입력 처리 테스트에서 해당 연산에 대해 확인 가능하므로 제외한다. 따라서 F 함수에서 확인해야 할 사항은 addition in modular 2^{32}연산이다. 설명을 위해 각 비트열 블록을 다음과 같이 칭한다.

- 평문1 - F 함수에 입력된 64비트의 평문 블록에서 0번째부터 31번째의 비트열
- 평문2 - F 함수에 입력된 64비트의 평문 블록에서 32번째부터 63번째의 비트열
- 키1 - F 함수에 사용되는 라운드 키에서 $k_{i,0}$의 키 값
- 키2 - F 함수에 사용되는 라운드 키에서 $k_{i,1}$의 키 값
- 결과1 - 평문1과 키1의 bitwise exclusive OR 연산의 결과 값
- 결과2 - 평문2와 키2의 bitwise exclusive OR 연산의 결과 값

F 함수에서 addition in modular 2^{32} 연산은 총 3번 수행된다. 이때 해당 연산은 같은 방법으로 구현된

것으로 간주한다. 따라서 테스트는 첫 번째 연산을 대상으로 테스트 벡터를 생성한다. 첫 번째 연산은 결과1과 결과2의 bitwise exclusive OR 연산의 결과가 G 함수를 수행한 결과 값과 결과1을 입력으로 수행한다. addition in modular 2^{32} 연산의 구현 정확성 테스트는 올림수에 대한 처리와 각 입력값의 정확한 연산을 수행하는가에 대한 확인으로 이루어진다. 따라서 addition in modular 2^{32} 연산이 수행되기 바로 전의 두 개의 32비트 블록 값이 최하위 비트에서 오른쪽으로 각 비트에 1의 값을 채우면서 각 테스트에서 32비트 블록의 값이 다음과 같이 변하도록 한다.

1회 : $80000000_{(16)}$
2회 : $c0000000_{(16)}$
3회 : $e0000000_{(16)}$
4회 : $f0000000_{(16)}$
5회 : $f8000000_{(16)}$
.....
31회 : $fffffffe_{(16)}$
32회 : $ffffffff_{(16)}$

F 함수 입력에서 위와 같은 값이 되기 위한 임의의 입력 테스트 벡터 값을 산출하였다. 관련 데이터는 부록 "(2) F 함수 동작과정 테스트 벡터"에서 제시한 값을 참조하길 바란다.

4.3 G 함수 동작과정 테스트

G 함수 동작과정 테스트는 G 함수의 구현 적합성을 테스트한다. G 함수는 G 함수의 입력 값 생성에 사용되는 연산과 G 함수 내에서 처리한 각 비트 연산으로 나눌 수 있다. G 함수 내에서 처리하는 연산은 S-box, bitwise AND, bitwise exclusive OR 연산이다. 이때 bitwise AND 연산은 S-box 입출력과정 테스트에서 다루었고, bitwise exclusive OR 연산은 평문 동작과정 테스트에서 다루었다. S-box는 독립적으로 테스트하기 때문에 제외한다. 따라서 G 함수 동작과정 테스트에서는 32비트의 비트열에 대해 G 함수가 정확하게 수행되는가 만을 확인하도록 한다. SEED 알고리즘에서 G 함수는 F 함수에서 세 번 호출되고, 키 생성에서는 두 번 호출된다. 이때 G 함수는 같은 방식으로 구현되었다는 것을 가정한다.

테스트 대상은 라운드 키 생성과정에서 사용되는 첫 번째 라운드의 첫 번째 G 함수를 대상으로 하며, 이때 G 함수에 입력되는 32비트의 입력값을 다음과 같이 조절하려 한다.

1회 : $00000001_{(16)}$
2회 : $00000002_{(16)}$
3회 : $00000004_{(16)}$
4회 : $00000008_{(16)}$
5회 : $00000010_{(16)}$
.....
31회 : $40000000_{(16)}$
32회 : $80000000_{(16)}$

이를 위해 입력된 128비트 키를 32비트의 네 개의 블록에서 해당 G 함수 입력값 산출에 사용되는 첫 번째와 세 번째 블록 그리고 라운드 키 생성과정에 사용되는 16개의 상수 중에 첫 번째 KC_0의 값을 이용해서 키 값을 생성한다. 1회 테스트에서 $00000001_{(16)}$값은 첫 번째 블록과 세 번째 블록 합에서 상수 KC_0의 값을 뺀 결과이다. 따라서 세 번째 블록의 값을 모두 0으로 하고, 첫 번째 블록의 값을 앞의 값이 나오도록 계산하기 위해 임의의 상수값이 필요하다. 상수 Const의 값을 16진수로 다음과 같이 초기화한다.

$$Const=9e3779b900000000000000000000000_{(16)}$$

위의 상수 Const를 이용하여 특정의 값을 더하면 의도된 첫 번째 블록이 산출된다. 사용하는 특정의 값들은 다음과 같다.

1회 : 00000001000000000000000000000000(16)
2회 : 00000002000000000000000000000000(16)
3회 : 00000004000000000000000000000000(16)
4회 : 00000008000000000000000000000000(16)
5회 : 00000010000000000000000000000000(16)
.....
31회 : 40000000000000000000000000000000(16)
32회 : 80000000000000000000000000000000(16)

G 함수 동작과정 테스트를 통해 G 함수의 수행과 subtraction in modular 2^{32} 연산의 구현 정확성을 확인할 수 있다.

각 횟수별 입력과 출력 테스트 벡터는 부록 "(3) G 함수 동작과정 테스트 벡터"를 참조한다.

이를 알고리즘으로 나타내면 다음과 같다.

〔표 2〕 G 함수 처리 테스트

SVS: 초기화
Const =9e3779b9000000000000000000000000$_{(16)}$
$TEMP_1$=00000001000000000000000000000000$_{(16)}$
PT =00000000000000000000000000000000$_{(16)}$
Const, TEMP1, PT를 IUT에 전송한다.

IUT:
FOR i=1 to 32
{
KEY_i=$TEMP_i$+Const
KEY_i와 PT를 입력하여 알고리즘을 수행,
암호문를 CT_i를 산출한다.
i, KEY_i, PT, CT_i를 SVS에 전송한다.
$TEMP_{i+1}$=$TEMP_i$<<1;
}

SVS: 반복을 통해 얻은 결과 값과 예상 기대값을 비교하여 검증의 성공/실패 여부를 판단한다.

4.4 S-box 입출력과정 테스트

S-box 입출력과정 테스트는 S-box를 이루는 각 값들이 입력에 의해 정확하게 계산되어지는가를 테스트한다. 따라서 랜덤한 입력값에 의해 S-box의 모든 값들을 확인할 수 있게 테스트되어야 한다. 두 개의 S-box를 S-box1, S-box2로 표기한다. 이때 다음 사항을 가정한다.

- 각 라운드에 사용되는 S-box1과 S-box2는 같은 방법으로 구현되었다고 간주한다.
- 입력과 키로부터 결정되는 각 라운드의 S-box1과 S-box2의 입력은 랜덤하게 분포하는 것으로 가정한다.

먼저, S-box 테스트를 위해 S-box를 이루는 각각의 값들을 모두 확인하려면 적어도 몇 개의 테스트 벡터가 필요한가에 대해 계산하자. 이를 위하여 "1부터 256까지의 모든 수가 적어도 한번 나오게 하려면 몇 번의 입력 시도가 필요한가"에 대한 값을 고려해야 한다. 이 입력시도를 n이라 가정하고, n의 기대값을 구한다. n의 기댓값은 1부터 256까지 수 가운데에서 n개를 입력했을 때 모든 값이 한번 이상 입력될 확률을 이용한다. 따라서 1부터 256까지의 수 가운데 n번의 입력 시도에 의해 모든 값이 적어도 한번 이상 나타날 확률 계산을 위해 먼저 {1,2,..... n}에서 {1,2,....r}으로의 함수 중 전사 함수의 개수 K(n,r)를 구하면 다음과 같다.

$$K(n,r) = r^n - {}_rC_{r-1}(r-1)^n + {}_rC_{r-2}(r-2)^n - {}_rC_{r-3}(r-3)^n + \ldots + (-1)^{r-1}C_1$$

이를 이용하여 1부터 r(r=256)까지의 수 가운데 n개를 뽑아서 나열할 때, 모든 값이 적어도 한번 이상 나타날 확률 $P_1(n,r)$은 다음과 같다.

$$P_1(n,r) = \frac{K(n,r)}{r^n}$$

그렇다면 n번째에 성공할 확률은 n-1번째까지는 성공하지 못하고 n번째에 비로소 모든 값이 한번 이상 나타날 확률인 $P(n,r)$은 다음과 같다.

$$P(n,r) = P_1(n,r) - P_1(n-1,r)$$

이때 확률 값이 1에 근사토록 계산한 결과 $\sum_{n=256}^{4000} P(n,256) = 0.999959$의 값을 얻었다. 따라서 4,000까지의 기대값을 구한 결과는 다음과 같다.

$$E = \sum_{n=256}^{\infty} n \cdot P(n,256) = \sum_{n=256}^{\infty} n\left(\frac{K(n,256)}{256^n} - \frac{K(n-1,256)}{256^{n-1}}\right) = 1567.66$$

S-box는 SEED 알고리즘 수행에서 16라운드를 거치고, 독립적으로 S-box1이 두 번 호출되고, S-box2가 두 번 호출된다. 또한 키 생성 알고리즘 중에 이러한 처리가 독립적으로 두 번 호출되므로 하나의 입력에 대하여 모든 값을 주기 위해서는 약 24 (≈1567.66/(16×2×2))개의 랜덤한 입력, 키 쌍이 필요함을 예상할 수 있다.

다음은 24개의 랜덤한 키 값을 이용해서 테스트한 결과 S-box1과 S-box2의 입력 값 회수를 16진수로 나타난 결과로써 1부터 256까지의 각 입력 횟수를 순서대로 적은 것이다. 각 값은 10개씩 하이픈(-)기호로 구분했고, 마지막에 6개의 값에 대한 결과 값을 표기했다. 이를 통해 S-box의 모든 값이 주어진 24개의 랜덤한 키 값으로 한번도 입력되지 않은 값이 발생하지 않은 것을 알 수 있다.

-S-box1
5636b94297-7265984246-9379965746-9665839926-8785637367-5b45844735-83515a7955-275c567826-aa55754286-478454a584-56c558a843-4644869793-5552127463-8442649456-83a6475585-c953479284-7434a57667-9876775467-5653774347-432673765a-6685887389-44759594b5-35749a55ba-7569849739-5a86465855-997238

-S-box2
392657755b-2564434877-5634928472-3936a5a563-d574874574-8694337779-416b452852-5354a2b6a7-9573684688-896885ba37-2436773342-b548994697-3676596b39-8637446478-7415725886-442849a775-55574998b5-a546424767-2547562b4a-56c4877654-a576559469-45395357e8-7887966435-8533c66584-6444654669-b77773

위의 실험을 통해 각 벡터 수와 성공율을 확인한 결과는 다음과 같다.

벡터수	1	2	3	4	5	6	7	8	9
성공율(%)	22.9	40.6	52.7	63.5	72.9	78.9	82.6	86.3	89.8

벡터수	10	11	12	13	14	15	16	17	18
성공율(%)	92.2	93.9	94.9	96.5	97.1	97.7	98.2	98.6	98.8

벡터수	19	20	21	22	23	24
성공율(%)	99.0	99.2	99.4	99.6	99.8	100

S-box 테스트 벡터의 이론적인 최소 개수는 8이다. 현재 실험을 통해 14개의 테스트 벡터만으로도 S-box가 검증될 수 있음을 확인하였다. 다음은 16진수로 표기한 테스트 벡터 값들이다. 이때 평문은 0으로 한다.

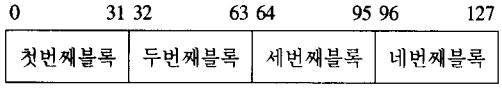
1회 - 키 : 23301f932fff1964f42c6a013f41b989
암호문 : 8c5bdf5472ab60799aca32cdc56252dd
2회 - 키 : af6bc0c75d54791d6ee658ccfa23eb54
암호문 : e26583c00fbd27fdd0c3259789dfd274
3회 - 키 : 4d0e430752e95ee0be257d5b73fc3920
암호문 : 8321a26bd9a743772f1761de3e000924
4회 - 키 : ce2fe16c8b4e1d9791321c854dfc3004
암호문 : 9672ca119de663763636980608f600af
5회 - 키 : 585d2b7d16a87ee1a7048483450a30ab
암호문 : 33b3ae3aeea0e738b38538a0b31a1ee3
6회 - 키 : 6bee8547f80899d91481bc23f6102e48
암호문 : 67270e4eb690f8a08834a00ca3d34f88
7회 - 키 : 0cc98b094e9861799cb51c2f55c361c0
암호문 : a21989bb55a1910a07f8149c9423ca6e
8회 - 키 : aed002b21b39f0739b4d08b3c1e0abae
암호문 : f6e970b50c5c9e9f5bf8a58ffc3409b8
9회 - 키 : 5839720717656eecc0b5ad3c3dd4a216
암호문 : 38ddca8c597ea35a18272088674a45af
10회 - 키 : 60e4b88b4e1bcdb6d8928db45a9ee7b9
암호문 : 79d197fe4e8011e2454e17cf27d2ed02
11회 - 키 : 19352f75009bdc549ddc9a7b7d4c0abb
암호문 : 5acfcd01fa9505945c8923a1db05bb28
12회 - 키 : 43fae274fa0ed9c0c155b9daa649b193
암호문 : 688db44d3b1b8bf880695f4d16db51b1
13회 - 키 : 2b775639c1155238413ce697b092e022
암호문 : 5ab59c42f8e38226e7b9c4d4be4e1052
14회 - 키 : 71c2ca7b39f6d01ea68522430edcf070
암호문 : 16d73275bd29681d7485eacf1899e8eb

S-box의 테스트 벡터는 부록 "(4) S-box 입출력처리 테스트 벡터"를 참조한다.

4.5 라운드키 생성과정 테스트

라운드키 생성과정 테스트는 라운드키 생성과정에서 수행되는 연산이 정확한가에 대해 확인한다. 라운드키 생성과정에서는 G 함수를 제외한 경우 right circular rotation by n bits와 left circular rotation by n bits 연산, 그리고 concatenation 연산이 반복적으로 수행된다.

검증을 위해 평문은 0으로 지정하고, 128비트의 키에 일련의 변화를 부여한다. 이때 128비트의 키는 32비트의 네 개의 블록으로 다음과 같이 구분한다.



첫 번째 블록과 두 번째 블록은 right circular rotation by n bits과 concatenation 연산 테스트에 사용되며, 세 번째 블록과 네 번째 블록은 left circular rotation by n bits과 concatenation 연산 테스트에 사용된다.

이 때 각 연산은 나누어서 테스트한다. 테스트를 위해 키 값은 모두 0으로 초기화하여 right circular rotation by n bits과 concatenation 연산 테스트가 32번 수행되고, 이어서 32번의 left circular rotation by n bits과 concatenation 연산 테스트가 수행된다. 테스트 각 횟수마다 해당 키 블록의 값을 다음과 같이 재설정한다.

· right circular rotation by n bits와 concatenation의 경우

첫 번째 블록은 concatenation 연산을 테스트하고, 두 번째 블록은 right circular rotation by n bits 연산 실행을 테스트한다. 이를 위하여 첫 번째 블록과 두 번째 블록의 최상위 비트를 1로 지정하고, 테스트 횟수마다 왼쪽 방향으로 해당 비트의 값을 1로 지정한다. 왜냐하면 로테이트되는 값이 첫 번째 블록과 두 번째 블록 순서로 결합되어 이루어지기 때문이다. 따라서 가장 오른쪽에 위치하게 될 두 번째 블록의 오른쪽 비트에 항상 1의 값을 지정하고, 결합 후 로테이트로 인해 첫 번째 블록의 오른쪽 비트의 값이 두 번째 블록으로 정확하게 이동되었는가를 확인할 수 있다. 테스트 횟수에 따라 키 값은 다음과 같이 변화한다.

1회 : $00000001000000010000000000000000_{(16)}$
2회 : $00000003000000030000000000000000_{(16)}$
3회 : $00000007000000070000000000000000_{(16)}$
4회 : $0000000f0000000f0000000000000000_{(16)}$
5회 : $0000001f0000001f0000000000000000_{(16)}$
.....
31회 : $7fffffff7fffffff0000000000000000_{(16)}$
32회 : $ffffffffffffffff0000000000000000_{(16)}$

· left circular rotation by n bits의 경우

세 번째 블록은 left circular rotation by n bits 연산의 실행을 테스트하고, 네 번째 블록은 concatenation 연산을 테스트한다. 세 번째 블록과 네 번째 블록의 최하위 비트를 1로 지정하고, 테스트 횟수마다 오른쪽 방향으로 해당 비트의 값을 1로 지정하면서 테스트한다. 테스트 횟수에 따라 키 값은 다음과 같이 변화한다.

1회 : $ffffffffffffffff8000000080000000_{(16)}$
2회 : $ffffffffffffffffc0000000c0000000_{(16)}$
3회 : $ffffffffffffffffe0000000e0000000_{(16)}$
4회 : $fffffffffffffffff0000000f0000000_{(16)}$
5회 : $fffffffffffffffff8000000f1000000_{(16)}$
.....
31회 : $ffffffffffffffffffffffffeffffffe_{(16)}$
32회 : $ffffffffffffffffffffffffffffffff_{(16)}$

이를 알고리즘으로 나타내면 [표 2]와 같다.

각 횟수별 입력과 출력 테스트 벡터는 부록 "(5) 라운드키 생성과정 테스트 벡터"를 참조한다.

〔표 2〕 키 생성처리 테스트

SVS: 초기화
  KEY_1 =$00000000000000000000000000000000_{(16)}$
  PT =$00000000000000000000000000000000_{(16)}$
  KEY_1, PT를 IUT에 전송한다.

IUT:
 FOR i=1 to 32
 {
  KEY_{i+1}의 첫 번째 블록에서 32-i번째 비트와
    두 번째 블록에서 32-i번째 비트에
    1을 지정하여 $KEY1_i$에 저장한다.
  PT와 KEY_i를 입력하여 암호화 알고리즘을
    수행, 암호문 CT_i를 산출한다.
  i, KEY_i, PT_i, CT_i를 SVS에 전송한다.
 }

 FOR i=1 to 32
 {
  KEY_{i+1}의 세 번째 블록에서 i-1번째 비트와
    네 번째 블록에서 i-1번째 비트에
    1을 지정하여 KEY_i에 저장한다.
  PT와 KEY_i를 입력하여 암호화 알고리즘을 수행,
    암호문 CT_i를 산출한다.
  i, KEY_i, PT_i, CT_i를 SVS에 전송한다.
 }

SVS: 반복을 통해 얻은 결과 값과 예상 기대값을 비교하여 검증의 성공/실패 여부를 판단한다.

4.6 Modes 테스트

Modes 테스트를 통해 Known Answer Tests에서 발견할 수 없었던 구현상의 오류나 운영상의 오류를 발견할 수 있다. Modes 테스트는 랜덤한 값으로 평문과 키 값을 초기화한 후 400번 반복한다. 이때 암호문은 평문을 랜덤하게 작성하기 위해 10,000번 반복하여 생성한 암호문을 평문으로 재입력해서 사용한다. 이때 400번과 10,000번의 반복 횟수는 MOVS[5]와 AESAVS[6]의 모드 테스트의 횟수를 참조하였다.

이를 알고리즘으로 나타내면 [표 3]과 같다.

Ⅵ. 결 론

본 논문은 SEED 구현물의 표준 적합성을 판단하기 위한 평가 시스템 설계를 제안하였다. SEED 평가 시스템은 SEED 알고리즘 기능에 따라 검증을 수행하며, 주어진 테스트 벡터를 사용하는 Known Answer 테스트와 임의의 데이터를 이용한 Monte Carlo 테스트가 제공된다.

(표 3) Modes 테스트

SVS: PT_0 ,KEY_0를 초기화한다.
PT_0 ,KEY_0를 IUT에 전송한다.

IUT:
FOR i=0 to 399
{
i, KEY_i ,PT_i를 저장한다.
FOR j=0 to 9,999
{
IB_j=PT_j
KEY_j 와 PT_j를 입력하여 알고리즘을 수행, 암호문를 CT_j를 산출한다.
PT_{j+1}=CT_j
}
CT_{j-1}를 저장한다.
i, KEY_i, PT_0, CT_{j-1}을 전송한다.
KEY_{i+1}= KEY_i XOR CT_{j-1}
PT_0=CT_{9999}
}

SVS: 반복을 통해 얻은 결과 값과 예상 기대값을 비교하여 검증의 성공/실패 여부를 판단한다.

SEED 구현물 검증을 위한 Known Answer 테스트에서는 구현 적합성을 평가할 수 있는 정확한 테스트 벡터가 필요하다. 이에 SVS에서 사용하는 테스트 벡터 생성 알고리즘을 소개하였고, 또한 S-box 테스트를 위해 S-box 입력이 모두 가능하도록 필요한 테스트 벡터 개수와 해당 테스트 벡터에 대해 실험을 통해 산출하였고, Monte Carlo 테스트를 통해 시스템의 구현 및 운영상의 오류를 검출토록 하였다. 이를 통해 SEED 구현물에 대한 정확한 평가가 이루어짐을 확인할 수 있다.

본 논문에서 제시한 설계대로 SVS를 구현하고 테스트 해보았다. 테스트 방법은 50명에게 SEED 알고리즘을 설명한 후 각자 독립적으로 구현하여 제출케 하고, 제출한 구현물을 테스트 해 본 결과 Known Answer Test를 통해 잘못 구현된 부분을 찾아낼 수 있었다.

따라서 본 논문에서 제시한 SEED 알고리즘 검증 시스템의 경우 SEED 구현물의 기능에 따라 각 구성요소에 정확한 평가가 가능함을 알 수 있다.

참 고 문 헌

[1] Cryptographic Standards and Validati- on Programs at NIST, http://csrc.nist.gov/cryptval/

[2] Security Requirements for Cryptographic Modules, FIPS Publication 140-1, 1994.

[3] Security Requirements for Cryptographic Modules, FIPS Publication 140-2, 2001

[4] DES Modes of Operation specifies several modes of operation for the DES and Skipjack algorithms, FIPS Publication 81, National Institute of Standards and Technology, 1980.

[5] Sharon Keller and Miles Smid, NIST Special Publication 800-17, Modes of Operation Validation System(MOVS) : Requirements and Procedures, 1998.

[6] Lawrence E. Bassham III, The Advanced Encryption Standard Algorithm Validation Suite(AESAVS), National Institute of Standards and Technology Information Technology Laboratory Computer Security Division, 2002.

[7] 128비트 블록암호알고리즘 표준, 박성준, 이인수, 박형선, 김병천, 김승주, 한국정보통신기술협회, 1999.

[8] 박성모, 염용진, 박일환, 김대호, MOVS 소개 및 테스트 벡터 분석, 국가보안기술연구소, 한국정보보호학회 충청지부, 2001.

[9] Advanced Encryption Standard(AES), FIPS Publication 197, National Institute of Standards and Technology, 2001.

[10] Data Encryption Standard(DES), FIPS Publication 46-3, 1999.

[11] Skipjack and KEA Algorithm Specifications, Version 2.0, 1998.

[12] Escrowed Encryption Standard, FIPS Publication 185, 1994.

부 록

테스트 벡터 값은 모두 16진수로 표기했다.

(1) 평문 동작처리 테스트 벡터 (키=00000000000000000000000000000000)

순번	평문	암호문
1	00000000000000000000000000000001	a809864319cbbe1af28fdfb20b22c6d6
2	00000000000000000000000000000003	91cb9c42bc26d979d637fe87178f6ff6
3	00000000000000000000000000000007	b9d0c835424ef313c99ae7295eb053b0
4	0000000000000000000000000000000f	b3cc36e0afe4fc8b4fa2cce7a4cb0058
5	0000000000000000000000000000001f	e7e1b6cbcfb99543d7d792c9c6164d5c
6	0000000000000000000000000000003f	e2ca217104a6cc02db606ba11924ec1b
7	0000000000000000000000000000007f	37ada3f8d41198f6bb34463cd29056d4
8	000000000000000000000000000000ff	588114a92a9f612c2f406b4dcebd6c7b
9	000000000000000000000000000001ff	48ac8cd56f001b37ed2430069d35fa16
10	000000000000000000000000000003ff	919b95217480f2f2d2e36043975727f0
11	000000000000000000000000000007ff	5bee38f3eee7488426d66f4fae8e772c
12	00000000000000000000000000000fff	6246dda2dbdf30aca0559e575e8fcfe1
13	00000000000000000000000000001fff	963a9e403a50e6b22df020a8bc64e8fa
14	00000000000000000000000000003fff	cca19f4a2ed86f38e51f76365c0c175a
15	00000000000000000000000000007fff	b5ecd88bde5841497a18a92e7ac650b6
16	0000000000000000000000000000ffff	3223ef4aeca6d5df1f4cc0c871ec469c
17	0000000000000000000000000001ffff	f65499dbe71487b32eece9dc2111c0d9
18	0000000000000000000000000003ffff	cdafc5750f8354af9cffb699fdb18488
19	0000000000000000000000000007ffff	b10ac6de2238656e657a9faca6efefa4
20	000000000000000000000000000fffff	ded6a7a73eeec24d6475189a99ebe161
21	000000000000000000000000001fffff	43c289063a1e5c8d84865f6ff756551a
22	000000000000000000000000003fffff	e005eff0215d3d150f24d6a42a6c7a27
23	000000000000000000000000007fffff	3ada185a97414be354835a694a5993be
24	00000000000000000000000000ffffff	3f170289429460642f6f857c8e8821d0
25	00000000000000000000000001ffffff	cd88471e833db6ba7900aa64e72cbd6a
26	00000000000000000000000003ffffff	b0a78d345a62bcd48a3e225b55cb404b
27	00000000000000000000000007ffffff	1b7295721405bf5f984c2f8fe01e43d4
28	0000000000000000000000000fffffff	1ec6902b25217cb0c886babd858fae39
29	0000000000000000000000001fffffff	54480198c3a9a5f2eee6dc7880366bc0
30	0000000000000000000000003fffffff	7ef52f5be48a03913a93bfa402298793
31	0000000000000000000000007fffffff	95f88f7845fc6d9988b01fb9c6aef708
32	000000000000000000000000ffffffff	3e9c74f77aefb4bd258554e36bc5c498
33	000000000000000000000001ffffffff	183f4a21eb88bb248626c56a93e7917a
34	000000000000000000000003ffffffff	30b361338d4030a2cdb6fb47c6116a27
35	000000000000000000000007ffffffff	b85a8a31f4a92eb3dd64156647f6a885
36	00000000000000000000000fffffffff	180da8c15b7aef727ab2dfb22d961508
37	00000000000000000000001fffffffff	6ec665b67f1d9c685861dc8bf412dbce
38	00000000000000000000003fffffffff	f2e0db6f8aee54f7b2a8f00672261854
39	00000000000000000000007fffffffff	f02906ff88e1aaa6212e53e3f2f3f75a
40	0000000000000000000000ffffffffff	2b2aef2e2921c19c12e42f3e2e8fd964
41	0000000000000000000001ffffffffff	b27edc0c8a1ba8325bc53bf81f61f6e0
42	0000000000000000000003ffffffffff	cb1d7d89537fe211c6a9321254858cbd
43	0000000000000000000007ffffffffff	14b4e9bdf19d244ba459c4b9253968c0
44	000000000000000000000fffffffffff	4891b68ba901d4ca2445a6aa7de8aa59
45	000000000000000000001fffffffffff	52d857c9cb95e18f851f11565c958062
46	000000000000000000003fffffffffff	2e108e7ab44e635e11f20ef02e2d8f79
47	000000000000000000007fffffffffff	24bf7bed4549c53870ec7652dec9ca7e

순번	평문	암호문
48	00000000000000000000ffffffffffff	51081a62b1d9b924a7119b8bac4f5071
49	00000000000000000001ffffffffffff	5ee506a44eaa2ce468a0e6decdcdb466
50	00000000000000000003ffffffffffff	bcf55e86707ed659167113fbbb5ed2a2
51	00000000000000000007ffffffffffff	72b67f99da327328bfd6cbd008b2e826
52	00000000000000000000ffffffffffff	76c80cb00408625b24a051a475159d12
53	00000000000000000001ffffffffffff	1093a85433d8a4bc7eaaa08b00ad28c4
54	00000000000000000003ffffffffffff	43124b9bdab9406ce416b2acffb1c295
55	00000000000000000007ffffffffffff	8732751384172c4446a159bb13951d50
56	00000000000000000000ffffffffffff	a492f676ef725bd9e65fd9ea6edd245e
57	00000000000000000001ffffffffffff	30e5d8aef9821365ea3a621ffeea4a77
58	00000000000000000003ffffffffffff	07c5bf26b984a1720e009facfde932ff
59	00000000000000000007ffffffffffff	31f9d60576bbd2a7235005e8472d5d2f
60	00000000000000000000ffffffffffff	1e7bb72941c6bf5ee8aa83178cf29d73
61	00000000000000000001ffffffffffff	7b7f647e81c851e8a1bf5cc1adccc5ab
62	00000000000000000003ffffffffffff	6879d144f37d6c0853c7137513d5d4f9
63	00000000000000000007ffffffffffff	ec566c142dffc66cb53beefe27b6c4db
64	00000000000000000000ffffffffffff	9cb5914212c2ca7834e0ffd319726907
65	00000000000000000001ffffffffffff	07bb805169b718982e18ca8e83dc5464
66	00000000000000000003ffffffffffff	55f6f5fe4b88416c0073f11d33c8f04f
67	00000000000000000007ffffffffffff	7a1a4ab083a2906808e9c758b816e62d
68	00000000000000000000ffffffffffff	7c7a630e08d924fa45f476780b49724b
69	00000000000000000001ffffffffffff	1ac46126540d689f6d0c3fc012a479ca
70	00000000000000000003ffffffffffff	f3ecbf775f210ea0bc8d43ac8db1ad1a
71	00000000000000000007ffffffffffff	87741b219400770f5c443c5760bc94ff
72	00000000000000000000ffffffffffff	e33ab532bc8adcd244652052f54d2ec9
73	00000000000000000001ffffffffffff	5d5c4eac123d5a198d0e7ca4fdad838c
74	00000000000000000003ffffffffffff	97a1e5d2ab8fc28ec542ecf9fdd16aa5
75	00000000000000000007ffffffffffff	d7f60c2a64420c01d937791a6db7b642
76	00000000000000000000ffffffffffff	a8a85b08f089f6c2092d12f73e69c9a7
77	00000000000000000001ffffffffffff	22cae48ebcf0e3cea663649f7f9918c7
78	00000000000000000003ffffffffffff	d9adf33c8c024facccbd9a22b03e1746
79	00000000000000000007ffffffffffff	ab96be2a017fd9e6469d6b1a13736f45
80	00000000000000000000ffffffffffff	6918f6d79103d568ba4f00775a9a7296
81	00000000000000000001ffffffffffff	6a71eb6d08feb4e5a14dbcf2eda1ddbe
82	00000000000000000003ffffffffffff	1961a7ca2f867b61e24a6c7b2e76dc88
83	00000000000000000007ffffffffffff	cc1abecaada13b11017045f6a35fcbd5
84	00000000000000000000ffffffffffff	03f78eabd54d81febc563dd3d988ca43
85	00000000000000000001ffffffffffff	68940dc91bd6c994f4123d303a16f92d
86	00000000000000000003ffffffffffff	10ced1bdc102d3b068e63043f43bf540
87	00000000000000000007ffffffffffff	4d946e4ec2e7590eadcb9c7ed28bba35
88	00000000000000000000ffffffffffff	c4cbd39cf3938f4ea6270555d268d007
89	00000000000000000001ffffffffffff	bc0925bd1af5dd2d6e4bf49a96ae454e
90	00000000000000000003ffffffffffff	14265fde1475a2f50056f39079410875
91	00000000000000000007ffffffffffff	8cae1bddc2c1c22201ecd7b09cf9fa51
92	00000000000000000000ffffffffffff	b37f569ad3548093d01bc9124a1c01e9
93	00000000000000000001ffffffffffff	734e7857ed187f6c6586b263a81969f0
94	00000000000000000003ffffffffffff	c68ed44d4835f5ec475e073997c4868e
95	00000000000000000007ffffffffffff	3c3df8e67b88a071a1e0d0dfb5b649a1
96	00000000000000000000ffffffffffff	6886a2a255bac982f6a404cd822c30ab
97	00000000000000000001ffffffffffff	c16e0d9af3368bc23fab6abe4d55642a
98	00000000000000000003ffffffffffff	4a002ac6f904cbc7b5cebdf7f1c29964
99	00000000000000000007ffffffffffff	f1f253fa255a7b5858da42d84993db12
100	00000000000000000000ffffffffffff	6464743decd5794d474bf16e5bcd697e
101	00000000000000000001ffffffffffff	12ff0df560c6674e8b5fcd54c0cbf4de

순번	평문	암호문
102	0000003fffffffffffffffffffffffff	08360f0c931907b40c4db429f9c2983d
103	0000007fffffffffffffffffffffffff	e31c5a915d71ef26f0151d25d77e68d3
104	000000ffffffffffffffffffffffffff	1f01f4c070ec47683080494f41adb2f0
105	000001ffffffffffffffffffffffffff	d85ea5473b1c185be46cf89c6b45b212
106	000003ffffffffffffffffffffffffff	e4ddfa531a52e25ae7628ddf6dd9c595
107	000007ffffffffffffffffffffffffff	fb9203b168ed42adf3ef821667d7ab39
108	00000fffffffffffffffffffffffffff	c4c9d398a5d1b6e46bea2e8823c5c79a
109	00001fffffffffffffffffffffffffff	4eb3768b5af7e2ffc830081ea89d271e
110	00003fffffffffffffffffffffffffff	265e1c93521e21e3c2b30399c807ab6d
111	00007fffffffffffffffffffffffffff	b09263bb6afc7f0917e668d25b49b610
112	0000ffffffffffffffffffffffffffff	8665b8f03c1eb73d1ec3568c28734416
113	0001ffffffffffffffffffffffffffff	5d6137cfbb1c18ee13f7cd16bf695811
114	0003ffffffffffffffffffffffffffff	05599246e5ee9adf1768b62d875b1e58
115	0007ffffffffffffffffffffffffffff	c223d6228c1420eb136004bcaeb3aa01
116	000fffffffffffffffffffffffffffff	a2b70c3fa22310e571b42bbeb8a65d19
117	001fffffffffffffffffffffffffffff	1989f31d645f50cca2488d0045d5b1f4
118	003fffffffffffffffffffffffffffff	a99b07bb96b191ddfb1330a3fba8ada4
119	007fffffffffffffffffffffffffffff	df3c2cc4042b858f574d157f28e28bde
120	00ffffffffffffffffffffffffffffff	f73da52fc7d16cc854992e61ff0f1935
121	01ffffffffffffffffffffffffffffff	a8b79d01c00fa50e84b3f08cca3da7bd
122	03ffffffffffffffffffffffffffffff	819825dade0149e736fe6e8f0e976e3c
123	07ffffffffffffffffffffffffffffff	66bb57e0ac2a7adb3a515037c8d883e6
124	0fffffffffffffffffffffffffffffff	f26f0f03936b9823807557b0d25aca2b
125	1fffffffffffffffffffffffffffffff	e924d5ba3488549c8b899df4c875472a
126	3fffffffffffffffffffffffffffffff	6d3f517ad1c3ce3d8dbc2d52c791a98c
127	7fffffffffffffffffffffffffffffff	ae376db7c01d439b3ee9b1eccda5254e
128	ffffffffffffffffffffffffffffffff	845f78166a0be9c9450c09c36f8a38bb

(2) F 함수 동작과정 테스트 벡터 (키=00000000000000000000000000000000)

순번	평문	암호문
1	00000000000000000fc8f8c7e0dbae876	634cc605fc6e425557b589649fc1091e
2	00000000000000000bc8f8c7e4483e176	28426c175e00d8604564815feb748ab2
3	00000000000000009c8f8c7e6423173e	9f255d70cab91042c76cdc41ab95169d
4	00000000000000008c8f8c7e74da18bd	49d6344b6d18cc9e03a5736b216eda66
5	00000000000000000848f8c7eafdaf5bb	1028ccd777123573234aaf9ab60bd37c
6	00000000000000000808f8c7ea4da6b8f	d39473e4c2671d13146963a3f2e297ac
7	00000000000000000828f8c7ec5016be5	9550bf4caaea2869b29579d0ec60230e
8	00000000000000000838f8c7e3c456b0e	5ab1b9debe01ec01287f25c25920e98b
9	00000000000000000830f8c7eed856b5c	fd3925a9a95235de8ac73a61f3fae86c
10	00000000000000000834f8c7eb9006b9c	d341ad509b8831b75fe27ce3e31765c5
11	00000000000000000836f8c7e05cad99c	02cd1c1f39835d219666db53b1a3f6bf
12	00000000000000000837f8c7e14dd3d9c	ed75c3ab320f789c3860957beb7c445a
13	00000000000000000083778c7e14e58772	51eafa72df015182f71a7d2e0a11ac86
14	00000000000000000083738c7e1473e15e	5b1b4830dcc6aac1bd8254a648e80d1a
15	00000000000000000083718c7e15718aed	d6a9abc1e8098365d2a15a8c8753315d
16	00000000000000000083708c7edb70b2c1	d2e9fc4a25744b487963eba4b2c89d36
17	00000000000000000083700c7e25709d0c	5115226d3e5650d0a47fd00e4b3fcf73
18	00000000000000000083704c7ef170c5bd	9d9ccd7ec0bcb936e79e2f89e0247603
19	00000000000000000083706c7e43aae518	9792eea9bd81f48af38302527d1db9e8
20	00000000000000000083707c7ea7eaf576	aed9b6eb7d7e645d675171f628ec08f5
21	00000000000000000083707 47e1d5a0b76	9d46f11f0cba0b33bb89edfc3eb882a3
22	00000000000000000083707 07e7b3fdf76	947feec35a19ffbee0db8e3c0202a98a
23	00000000000000000083707 27e7b206aac	8c786a68096d75b60d4d7f9166d459e0

순번	평문	암호문
24	00000000000000008370737e7b59c936	9dec21f2aa39c3ba0d480d1088552a52
25	0000000000000000837073fe7bac9097	d82b14e0e6348e6e98fb212064224e6b
26	0000000000000000837073be7b25c2bd	8e5c7b4e3151c355e401c0a23d3ca3ec
27	00000000000000008370739e8d25d82c	2629ab241385223418dd428b2a360278
28	00000000000000008370738e822594ae	3ee964d50ad68548989a901a76f9cb19
29	000000000000000083707386f63947d	69122a438eb1afb80643eebb9e6bd6a8
30	00000000000000008370738 2f1c59460	c875586330e39161bbe48fc97f4a86dc
31	0000000000000000837073808abeef62	c88ba4592d17aa4ebda95c6b669e174d
32	00000000000000008370738158783d63	d4d5af6ed569f0fd002578c21c04d130

(3) G 함수 동작과정 테스트 벡터 (평문=00000000000000000000000000000000)

순번	키	암호문
1	9e3779ba00000000000000000000000	52fba30f30a2b657d29fd763e89839cb
2	9e3779bb00000000000000000000000	7c0fafe44a7b836a727b095afdfe14af
3	9e3779bd00000000000000000000000	cdfa76c3dff6e0db7f807e5c63822182
4	9e3779c100000000000000000000000	29fe29ffb78204bb93611b9b01dbb244
5	9e3779c900000000000000000000000	d9c3a6f708ed963227659f5e541b681c
6	9e3779d900000000000000000000000	09054c3c4be35a2445a43fca7e8ab8c7
7	9e3779f900000000000000000000000	de076e06f9806917559b335489c428f1
8	9e377a3900000000000000000000000	fd1811d00a598a87d30831b27a5e4b13
9	9e377ab900000000000000000000000	3ec4d916c5289de49d782caf9e67d688
10	9e377bb900000000000000000000000	84f21df288afd8a07637f83582d488f7
11	9e377db900000000000000000000000	733952e114b1f806bc0377aedeb07234
12	9e3781b900000000000000000000000	5437a36016f6d0b656ccd02a164484a3
13	9e3789b900000000000000000000000	a95e2ff8cb52be57dbfd4d557dc88668
14	9e3799b900000000000000000000000	e920e56fc89eb3b3f8f78a7dbb225d7c
15	9e37b9b900000000000000000000000	5286ceb001d06707a8aaf4620c1ee430
16	9e37f9b900000000000000000000000	4f69a8016f92918a497373512e88bb9b
17	9e3879b900000000000000000000000	02584088a80de2098a11a0a989ce30e2
18	9e3979b900000000000000000000000	83156859a317be8df8daac9e38161120
19	9e3b79b900000000000000000000000	d8d12c2f8cc5cb958970d03b01502874
20	9e3f79b900000000000000000000000	e1dd31df42bf749ecbedeb5310457c03
21	9e4779b900000000000000000000000	f29e33994965251a9bf305a76a33a4b7
22	9e5779b900000000000000000000000	69706603171ca66b2049094612fb883a
23	9e7779b900000000000000000000000	026ccd8cf72ecf969d22cfcaf0b7126d
24	9eb779b900000000000000000000000	d4a88c1fb2acaf2e2692edc504d61b51
25	9f3779b900000000000000000000000	ce7bcaa2d10122ab51ed7bb0f03b144c
26	a03779b900000000000000000000000	7de7cedb56ddae311a1c64dfc84335a0
27	a23779b900000000000000000000000	5eb4e71d1d9941c775dddf11e90d7d06
28	a63779b900000000000000000000000	4f840c4014e65ef86fc814aac2cdf425
29	ae3779b900000000000000000000000	2294cef061028672aefdba619333d4a8
30	be3779b900000000000000000000000	4b8d0d98c86af33ed5b85b24d72f4524
31	de3779b900000000000000000000000	e05e695f1a347e65310f8638a5c6973e
32	1e3779b900000000000000000000000	d34ed0130d5657aed92385a42e80c9c6

(4) S-box 입출력과정 테스트 벡터 (평문=00000000000000000000000000000000)

순번	키	암호문
1	fc389a719a41d5efebffb8bd203b8412	ca4f0a8fc64337d9dd671b501920d646
2	367790fe5c02e2363698a3ed391b5254	91b00d0bc5456a00f3a3f5d689d732c0
3	a7c79df599478836c53b563bb0cc6094	cceef90ba02820dd13e7cd6b0ef0e4a5
4	6b2964fe396fa8ebd6dc9b50be3c8431	292d5753fa361807ab871995af09f406
5	420433adc359b0ec7d5b82c511357037	56031fdc8a0af22002829cbd17b44228

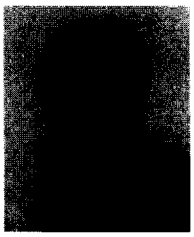
순번	키	암호문
6	14072c71495342cc7fd91e780681321f	9516d0c71bc2fdfc139ab802de30f550
7	8111fcb65291bf306c43a67ad8b1c74f	e6855eb8c0cfa8dce9343709241fa66a
8	6794a20e925e17ec85779173fcae6125	bfb2bfc788831e3ce18ae8d911cca314
9	d8261dc91add09d064fd79aaed62c231	ad4668be1a44739817510b7969e4605f
10	7560757700997a4873ac72d38486a26e	690fb02c6fbd2dcae639f07ae19bf8ff
11	1c6fc55b6f245782959582a9651d5a91	343cd0d40941eaa311206ed225bf5455
12	2853158f6665bd2184b5ec7b40a20ec4	dd81772fc5c7f6c879aa9350a28f7871
13	f5d262a0a3a24be9fd1f7f2be22d76b5	3ef591c66f83b84108b1114d8ba46542
14	cb27f366cc21f3a8f5ae6e66698913f7	d163d9426be92268f62f872bc8c5a7c1
15	1a442a24a7c8d1c00381e94da14cb436	07bca1df154d95cc78c6e9a22b9bf2a7
16	1df88bfbb285440c7b4700ef8ad74691	23917fd89bb78d276c410e5d0bf996c6
17	3e69f4232932c92b17060eb2812ed057	f9bc723dfa81ea292e16a7de3faad0cc
18	c7d5861ed3b323023910a96cbe84009b	3f2071532c6172386431470e1aea5ccc
19	f8f5f74dffd2513a5d21ee62a46e5c57	ba2031a30a0632c05e7d021963f99fb1
20	46c42c9a64cc9b36f2d46ae62041df54	f6eca2ad5d16a971e23dacef0ee26e08
21	b0288e3739da324710e9c6092ea3b15c	22724d0f9f5a5a3b5a4641eabbef6c10
22	925ac338a680687c4aaf87e3954cdd4c	7bfb9ed81afb4b4eaba91207f12c7fcc
23	e9a15de0bc7b2332e10695e46460a280	e446fed088e2c20e7154ca146744506a
24	8b1882c9128aaa24e109c8e081dd27b3	1422edadd10de690a95ba6612cd6237c

(5) 라운드 키 생성과정 테스트 벡터 (평문=00000000000000000000000000000000)

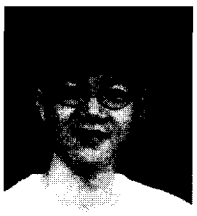
순번	키	암호문
1	00000001000000010000000000000000	d5470211a6d802385da725601a3873d7
2	00000003000000030000000000000000	d1814fb991f2f08ce1ace0fa2ed0a26f
3	00000007000000070000000000000000	9920cb744711a0f18a8c81ecd3d94fb0
4	0000000f0000000f0000000000000000	038ec0957b582b36621663362d8270c5
5	0000001f0000001f0000000000000000	7c7bc8ad70128cb6fa00a1e6c03ab42d
6	0000003f0000003f0000000000000000	7fc04e1a38d441dbf7a53f59ceb06a0e
7	0000007f0000007f0000000000000000	ee09b99dba72ed87f7d1670f8c82ca79
8	000000ff000000ff0000000000000000	7ae7afdc95b0cf46883f367c8ad3651d
9	000001ff000001ff0000000000000000	fd2d9a972ec7734239cc9139eda57dcf
10	000003ff000003ff0000000000000000	deb646cef8a0cb91b991c7581754b200
11	000007ff000007ff0000000000000000	04c76c2d75b875ec42e4c1c2bba9c103
12	00000fff00000fff0000000000000000	6dec0be131f0d3b298a5c51745bbf0b9
13	00001fff00001fff0000000000000000	412dc1045814b9130000c87a5689ecce
14	00003fff00003fff0000000000000000	0e2d270b19c375e72f8e906fb1c84c26
15	00007fff00007fff0000000000000000	9d84d863c7ab6e3d609ad78297a62ef4
16	0000ffff0000ffff0000000000000000	da53a8e39710e6fde5dfb8a24530ca20
17	0001ffff0001ffff0000000000000000	abff658c38fcd14745e422b5f1ab6d7c
18	0003ffff0003ffff0000000000000000	06aa4d08e465daf82d067d15579dc291
19	0007ffff0007ffff0000000000000000	2deaf4f127b1cedf303c629ca1174f73
20	000fffff000fffff0000000000000000	9126d06cbe519c123d7dcf26735c1677
21	001fffff001fffff0000000000000000	545e2ce30650a9099530df8e2a0097b3
22	003fffff003fffff0000000000000000	0bd0b881f69ed915be6cc9d5caa69e2e
23	007fffff007fffff0000000000000000	a1687e36d21da73f47371500c629c52b
24	00ffffff00ffffff0000000000000000	0706d11a8a40b2d5832839b9fd61d80f
25	01ffffff01ffffff0000000000000000	57975cc598895253bb4a504789dfc465
26	03ffffff03ffffff0000000000000000	0780f8c4602db4b421a88c43f2e66a3e
27	07ffffff07ffffff0000000000000000	2b787daa248a64b8f013334aca1ea0a0
28	0fffffff0fffffff0000000000000000	c356e461e60a9e6a9459ebcb349b603c
29	1fffffff1fffffff0000000000000000	397b2034556ac68582e20085add4db84
30	3fffffff3fffffff0000000000000000	d4829b3a74d82b8f911898550efa2e9a
31	7fffffff7fffffff0000000000000000	e0514bbf8349375761ec73331fbb342e

순번	키	암호문
32	ffffffffffffffff0000000000000000	4c1837562c096d722e5727a59a47b7a5
33	ffffffffffffffff8000000080000000	2416cf4e6b771b31d456465620623f9a
34	ffffffffffffffffc0000000c0000000	f1eda9878aa4ddd31ad1cde7e3308400
35	ffffffffffffffffe0000000e0000000	03d81a60947d0a6c3da3ded20dbcb3ad
36	fffffffffffffffff0000000f0000000	3b0e133e7fe7c4cd129e398856b55d02
37	fffffffffffffffff8000000f8000000	a456178d6f3a2c32364b71e6c38c4e40
38	ffffffffffffffffc000000fc000000	84f81bf3c3b0617ab43a3515698eb55f
39	ffffffffffffffffe000000fe000000	ba5dbbac5b5ec9712bfc84d8161766c7
40	fffffffffffffffff000000ff000000	1ff107e61242f6c8d8b2ed9444c60f57
41	fffffffffffffffff800000ff800000	486c6082e1fa826eb9148a4dc83aa260
42	ffffffffffffffffc00000ffc00000	200acc1e458a6e38cbd8eff7455b7d6b
43	ffffffffffffffffe00000ffe00000	926dfc88ff03b3d9b543733b414484cb
44	fffffffffffffffff00000fff00000	c252d807e4cbb46294fee622e17255a4
45	fffffffffffffffff80000fff80000	429bb93907eb52816f4fd7a8eeab93bf
46	ffffffffffffffffc0000fffc0000	603f8398ffbb51e0575217ff2f07152b
47	ffffffffffffffffe0000fffe0000	7789c1837f110c0031bb8f28fd288cbc
48	fffffffffffffffff0000ffff0000	fce6136113ffed37156f5ce952708400
49	fffffffffffffffff8000ffff8000	12837d1488aeac28c102f0f5cd401e86
50	ffffffffffffffffc000ffffc000	4248de60f137e5e06ffab6c1404bf7b3
51	ffffffffffffffffe000ffffe000	00e1df430caaa2fff1b4930bf0c10a0b
52	fffffffffffffffff000fffff000	919380e8493498c1fbddad6ac2217d85
53	fffffffffffffffff800fffff800	f72b85ff836af1b7d03f8cf742b635e3
54	ffffffffffffffffc00ffffc00	9968ff2ea7c3b33741e9cf438975b738
55	ffffffffffffffffe00ffffe00	a6248e94b3cc443e43856227f8ae899e
56	fffffffffffffffff00fffff00	04182e05b65e3f4d115b773ba51ee796
57	fffffffffffffffff80fffff80	64bc00a95e91edf36b61d9f0a9994d84
58	ffffffffffffffffc0ffffc0	bbec2cf2f4a72e1873af1894cbfa7e19
59	ffffffffffffffffe0ffffe0	4bcb6f57f110ad5af6098925dcabd679
60	fffffffffffffffff0fffff0	9cc7c4d66e61bf44ac0a3039bd7bceea
61	fffffffffffffffff8fffff8	daa716a19b5d70d1e641d61b829b4c39
62	ffffffffffffffffcffffffc	75854b3933ae645d44889475d424d2f9
63	ffffffffffffffffeffffffe	e60845ee8e2284c55292cd1a371c8dcc
64	ffffffffffffffffffffffffffffffff	5a54cf44c50290c4cbf71b1083fd12e1

〈著者紹介〉



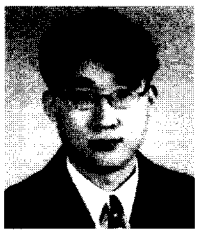
김 역 (Yeog Kim)
1992년 2월 : 성신여자대학교 전산학과 학사
2003년 2월 : 고려대학교 정보보호대학원 석사
2003년 3월 : 고려대학교 정보보호대학원 박사과정
<관심분야> 암호 모듈 평가, 평가 모델 설계



정 창 호 (Chang-Ho Jung)
2002년 2월 : 고려대학교 응용생명환경화학과 학사
2002년 9월~현재 : 고려대학교 정보보호대학원 석사과정
<관심분야> 블록 암호, CMVP, 정보은닉

장 윤 석 (Yun-Seok Jang)
2002년 2월 : 연세대학교 경영정보학과 학사
2002년 3월~현재 : 고려대학교 정보보호대학원 석사과정
<관심분야> 블록 암호, CMVP

이 상 진 (Samg-jin Lee) 정회원
1987년 2월 : 고려대학교 수학과 학사
1989년 2월 : 고려대학교 수학과 석사
1994년 2월 : 고려대학교 수학과 박사
1989년 2월~1999년 2월 : 한국전자통신연구원 선임 연구원,
1999년 3월~2001년 8월 : 고려대학교 자연과학대학 조교수
2001년 9월~현재 : 고려대학교 정보보호대학원 부교수
<관심분야> 관용 암호의 분석 및 설계, 정보은닉



이 성 재 (Sung-jae Lee)
1997년 8월 : 고려대학교 수학과 학사
1999년 8월 : 고려대학교 수학과 석사
1999년 9월~현재 : 한국정보보호진흥원 연구원
<관심분야> 블록 암호 및 스트림 암호 분석 및 설계