CMVP 테스트를 적용한 SEED 암호 알고리즘 모듈 구현

박성근*, 정성민*, 서창호**, 김일준***, 신승중*, 김석우*

*한세대학교 IT학부, ** 공주대학교 응용수학과, *** 국가보안기술연구소

e-mail: parkavatar@hotmail.com

The Implementation of SEED Cipher Algorithm Test Module Applied CMVP Test

Park Seong Gun*, Jeong Seong Min*, Seo Chang Ho**, Kim Il Jun, Shin Seung Jung*, Kim Seok Woo*

* Dep. of Information & Communication Hansei Univ.

** Dep. of Applied Mathematics Kongju Univ.

*** National Security Research Institute

요 약

정보보호 평가는 크게 시스템 평가인 CC(Common Criteria)평가와 암호모듈 평가인 CMVP(Cryptographic Module Validation Program)평가로 나눌 수 있다. 본 논문은 국내 표준 암호 알고리즘 SEED를 북미의 CMVP의 3가지 블록 알고리즘 시험방법인 KAT(Known Answer Test), MCT(Monte Calro Test), MMT(Multi-block Message Test)를 JAVA환경에 적용하여 시범 구현하였다. 테스트 방법으로 CMVP의 MOVS, TMOVS, AESAVS를 선정하여 FIPS 표준을 적용하였다. 구현 환경으로는 JCE기반의 Cryptix를 채택하여 CMVP의 블록 암호 알고리즘 테스트 시스템 중 일부를 구현하였다.

1. 서론

정보보호에 대한 인식이 확산되면서 정보보호 제품 수요가 증가되고 있으며, 이에 따른 국내 CC평가 인증제도가 2002년 하반기부터 시행되고 있다. 반면에, 상용 암호 서비스에 대한 평가제도는 한국 정보보호진흥원의 K1~7E 평가에서 시작되어 최종적으로 암호알고리즘에 대한 평가를 병행하고 있다. 본 논문은 북미에서 현재 활발히 진행 중인 CMVP[8]내의 암호 알고리즘 시험 기술을 국내에 보다 효율적으로 통합하기 위한 작업의 일부로써, 국내 표준 암호 알고리즘인 SEED를 대상으로 시험 모듈을 Java 환경에서 시범 구현하였다. 본 논문에서는 CMVP의 평가 방식중의 하나인 MOVS와 AESAVS를 분석하고[5], 국내 표준 알고리즘 SEED를 CMVP의 평가 방식으로 알고리즘을 Java 환경하의 KAT, MCT Test를 구현하였다.

- 본 논문은 2002년도 정보 통신부 대학 기초 사업 (2002-018-3)지원에 의해 연구 되었음 -

본 논문은 제2장에서 CMVP의 블록 암호 알고리즘 평가 방식인 KAT, MCT, MMT에 대해 기술하고, 제3장에서는 국내 표준 암호 알고리즘인 SEED를 Java 기반의 CMVP 평가 방식을 적용 테스트한 기술을 설명하였고, 마지막으로 결론 및 향후 연구방향에 대해 기술하였다.

2. 블록 암호 알고리즘 구현 적합성 검증 방식

미국 NIST에서는 블록암호 알고리즘에 대한 다양한 검증방식을 개발해서 자국 내에서 개발한 정보 보호 제품을 평가하는데 있어 NIST에서 표준화한 암호 알고리즘 검증 방식으로 알호 알고리즘의 구현 적합성을 평가 하고 있다.

AES 알고리즘을 검증하기 위한 AESAVS[3], DES와

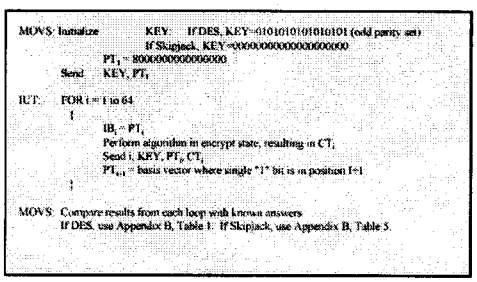
SkipJack 알고리즘을 검증하기 위한 NIST SP 800-17 MOVS[2] , 3DES 암호 알고리즘을 검증하기 위한 SP 800-20 TMOVS가 있다.

이러한 검증 방식의 기반이 되는 테스팅 방법인 KAT, MCT , MMT 에 대해 모드별 테스팅 방법을 기술하고, 실제로 자바 암호 라이브러리를 기반으로 국내 표준 블록 암호 알고리즘인 SEED를 KAT · MCT 모드별 검증방식으로 테스팅 한 것을 보여주고자 한다.

2.1 KAT(Known Answer Test)

KAT는 크게 3가지 타입으로 구분할수 있다. 키의 값은 일정하게 두고 평문을 일정한 값으로 변화 시키면서 테스팅 하는 VP(Variable Plaintext) KAT, 평문 값은 일정하게 두고 키값을 일정한 값으로 변화 시키면서 테스팅하는 VK(Variable Key) KAT, 그리고 마지막으로 해당 알고리즘의 요소별(AES의 GFSbox • KeySbox , DES[1]의IP · IP^{-1}등) 테스팅으로 구분한다.

KAT검증 방식은 블록 암호 알고리즘의 4가지 운영 모드(ECB, CBC, CFB, OFB)에 따라 암 · 복호화 테스팅을 수행한다.



(그림 2) ECB 모드 상에서 VP KAT

위의 그림은 DES 암호 알고리즘의 검증방식인 MOVS에서 ECB 모드상에서 VP KAT 테스팅 알고리즘을 간략하게 의사 코드로 표현한 그림이다. 초기값으로 DES의 키값은 010101…, 즉 8의 배수번째 비트만 1의 값으로 초기화 시켰다는 것을 알수 있다. 이 키값을 토대로 평문을 일정하게 변화시켜 라운드당 출력되는 암호문인 CT를 검증방식에 준한 테이블값과 비교해서 해당 암호 알고리즘의 구현 적합성을 평가한다. 위 테스팅 방식에서 DES는 64개의 암호문 CT를 생성한다.

키의 길이 N비트에 비례해 암호문 CT는 N개 생성된다는 것을 알수 있다.

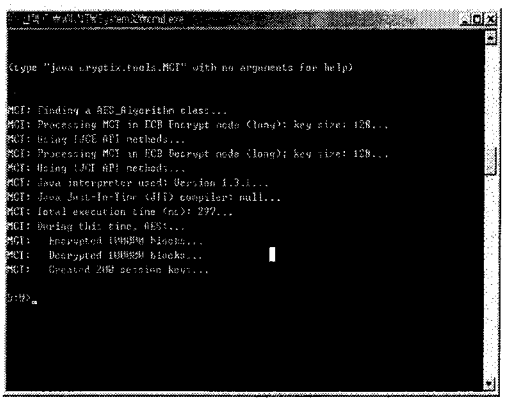
2.2 MCT(Monte Carlo Test)

MCT는 KAT처럼 정형화된 키값이나 평문을 통한 테스팅이 아닌 키값, 평문, 초기 벡터값을 임의의 값으로 한다는데 차이가 있다. 또한 KAT는 테스팅 과정중에 암복호화가 수십내지 수백번정도 일어나지만, MCT는 라운드당 수천에서 수만번의 암복호화 과정이 일어나므로 최종적으로는 수십만내지 수백만번의 암복호화 과정을 통해서 테스팅을 수행한다는데 있다.

```
Key[0] = Key
PT[0] = PT
For i = 0 to 99
  Output Key[i]
  Output PT[0]
For j = 0 to 999
  CT[j] = AES(Key[i], PT[j])
  PT[j+1] = CT[j]
  Output CT[j]
If ( keylen = 128 )
  Key[i+1] = Key[i] xor CT[j]
If ( keylen = 192 )
  Key[i+1] = Key[i] xor (last 64-bits of
  CT[j-1] || CT[j])
If ( keylen = 256 )
  Key[i+1] = Key[i] xor (CT[j-1] || CT[j])
PT[0] = CT[j]
```

(그림 2) ECB 모드 상에서 AES MCT 테스팅

위의 그림 2는 AES 암호 알고리즘 검증방식인 AESAVS에서 ECB모드 Encryption MCT 테스팅을 수행하는 알고리즘을 나타낸 그림이다. 여기서 Key[0], PT[0]는 초기 값으로서 임의의 값이 입력되고 초기 평문인 PT[0]가 암호화되서 CT가 생성된다. 이 CT[0]가 다음에 평문값인 PT[1]으로 입력으로 들어가고 그에 해당하는 암호화된값인 CT[1]이 생성된다. 이 과정이 CT[999]가 될 때까지 반복된다. 최종적으로 CT[999]가 해당 라운드의 암호문으로 생성되고 CT[999]가 다음 라운드의 평문으로 입력이 된다는 것을 알수 있다.



(그림 4) 자바 암호 API기반 AES MCT Test

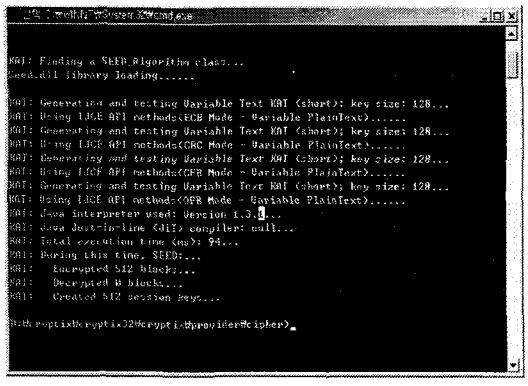
그림 3은 실제 공개용 자바 암호 API를 기반으로 기존 API를 개선해 AES MCT를 실제로 ECB모드상으로 테스팅한 예를 보여주고 있다.
위의 그림에서처럼 AES MCT는 수행도중 암복호화가 각각 10만번씩일어났다는 것을 알수 있다.

2.3 MMT(Multi-block Message Test)

MMT는 블록 암호 알고리즘이 여러개의 복수 메시지들을 처리하는 수행 능력을 평가하는데 있다. 테스트 되는 모드는 4가지 운영 모드에서 수행이 되고, 4가지 모드에서 블록의 길이 만큼 테스트를 수행하며, 블록의 길이가 1≤blocklength≤10이면 모드에 따라 10개의 메시지를 지원한다.

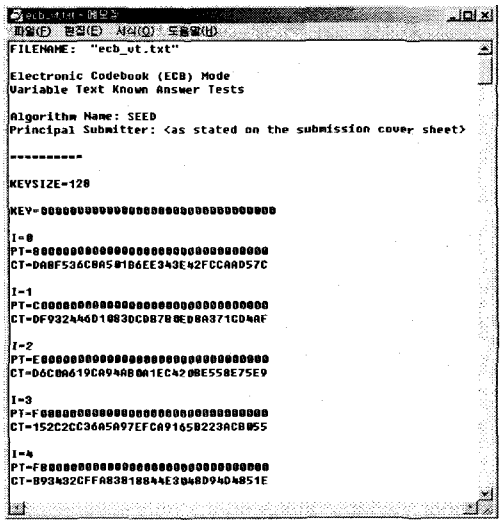
3. 자바 암호 API 기반으로한 SEED 암호 알고리즘 구현 적합성 테스팅

이 장에서는 실제로 이러한 검증방식을 국내 표준 블록 암호 알고리즘인 SEED에 적용해서 KAT, MCT를 수행한다.
여기서 구현한 테스팅 모듈은 공개용 자바 API인 Cryptix[6]를 기반으로 했고, 이 자바 API[7]는 국내 표준 암호 알고리즘인 Seed가 첨가 되어 있지 않은 관계로 Seed 알고리즘 스펙[4]을 분석해서 자바 기반 Seed 모듈을 API에 첨가해서 테스팅을 진행했다.
SEED.java 구현에 있어서 자바의 기본적인 암호API 메소드와 실질적인 암복호화를 수행하는 라이브러리인 SEED.DLL을 자바의 JNI(Java Native Interface)를 통해 수행이 되도록 개발했다.



(그림 5) SEED 모드별 VP KAT 수행

위의 그림4는 실제 SEED를 자바 기반 API로 4가지 운영 모드별로 VP KAT 암호화 테스팅을 수행한 예이다.



(그림 6) SEED ECB모드 VT KAT 테스트 벡터값

위의 그림 5는 AESAVS 기반의 검증방식으로 SEED 암호 알고리즘을 ECB모드로 암호화 테스팅을 수행한 결과 생성된 테스트 벡터 값들이다.
PT는 AESAVS처럼 라운드가 N이면 왼쪽 비트순으로 N+1번째 비트까지 전부 1비트가 되도록 일정한 패턴으로 암호화 테스팅을 진행했다.

4. 결론 및 향후 연구 방향

북미 CMVP 내에서 표준 암호알고리즘 DES, AES, DSA, SHA-1등 알고리즘 구현 정확성 및 모듈 적합성 시험을 수행하고 있다. 국내의 경우에도 MOVS등과 같은 암호 알고리즘 및 모듈 적합성 시험 환경이 활발히 구축 중에 있으며, 현재 국내 표준화 알고리즘인 SEED, HAS-160, KCDSA 및 향후 개발될 기타 알고리즘들에 대한 검증 툴이 절실히 필요하게 되리라 예측된다.[4] 본 논문에서는 국내의 암호알고리즘 시험 환경 활성화의 일환으로써, CMVP 시험 및 표준화 연구, 자바 기반 시험환경 하에서 국내 표준 암호알고리즘인 SEED를 정확성 시험 모듈의 일부를 적용·구현하였다. 향후, 국내 표준화 암호알고리즘 시험에 관한 선행 연구 결과들을 이식·포함하고, CMVP의 시험방법 및 절차를 국내 환경으로 보완·확장하여 추가의 테스트 베드 구축을 구현코저 한다. 본 연구 결과가 관련 연구 분야 및 향후 연구 목표에 자그마한 초석이 될 수 있기를 기대한다.

참고문헌

[1] NIST, *FIPS 46-3 Data Encryption Standard*, NIST, Oct. 1999

[2] NIST, *NIST SP 800-17 : MOVS*, NIST, 1998
[3] NIST, *The Advanced Encryption Standard Algorithm Validation Suit(AESAVS)*, NIST, July 2002
[4] TTA, "128-bit Symmetric Block Cipher(SEED)", 한국 정보 통신 기술 협회, 1999
[5] "블록암호알고리즘 정확성 테스트 모듈 구현" 정보보호학회, 2002
[6] http://www.cryptix.org
[7] http://java.sun.com/security
[8] http://csrc.nist.gov/cryptval