보안 알고리즘 검증 도구 개발

안현수*, 황유동*, 김재성+, 박동규*

*순천향대학교, 정보기술공학부

+한국정보보호진흥원, 평가 1 팀

Implementation of Validation Program for Cryptographic Algorithms

HyunSu An*, YouDong Hwang*, JaeSung Kim+, DongGue Park*

*Dept. of Infomation and Technology Engineering, Soonchunhyang Univ

+Team 1 of IT Security Evaluation, Korea Information Security Agency

요 약

정보보호에 대한 사람들의 인식과 관심이 높아짐에 따라 그에 따른 다양한 정보보호제품들이 나오고 있다. 국내외의 정보보호제품들은 각 제품별로 그에 따른 여러 가지의 보안 알고리즘을 사용하여 구현되어 지고 있다. 하지만 그러한 제품들의 신뢰성 판단여부는 아직까지는 경험에 의존되어지고 있는 현실이다. 이에 본 논문에서는 각 보안 알고리즘의 객관적 검증 방법을 제안하고 검증 도구를 구현함으로써 이에 대한 대안을 제시하고자 한다.

I. 서론

최근 디지털 시대를 맞아 인터넷 인프라의 발달로 전자 상거래와 기업 업무의 전산화 등이 활발히 추진됨에 따라 정보보호의 중요성이 대두되고 있다. 이에 따라 많은 기업들이 자사의 정보를 보호하고 상거래의 신뢰도를 높이기 위해 정보보호 솔루션을 사용하고 있다. 국내 정보보호 제품의 수는 증가하고 국가적 차원에서 이를 평가하여 인증해 주고 있지만 다양한 암호 표준에 맞는 신뢰성 있는 효율적인 평가 방법의 부재로 인해 국내 정보보호산업은 다른 국가에 비해 경쟁력이 약화되어 온 것이 사실이다. 미국 등의 정보보호 선진국은 NIST나 NSA 같은 국가적 차원의 인증 및 검증 기관을 선정하여 자국의 제품에 대해 표준 평가와 인증을 해 줌으로서 시장 경쟁력을 확보해 주고 있다. 국내에서도 한국정보통신기술협회(TTA)나 KISA(Korea Information Security Agency)와 같은 기관에서 표준을 제정하고 암호 알고리즘의 인증을 해주고 있지만 급변하는 시장 환경에 알맞은 효율적인 검증방안이 시급한 실정이다. 다양한 정보보호 제품에 대한 객관적 신뢰성을 제공하고, 국산 정보보호제품에 대한 시장 경쟁력을 강화하기 위하여, 정보보호제품내의 보안 알고리즘 구현 무결성을 검증하는 것은 필수적이다. 따라서 다양한 국내외 공개 보안 알고리즘의 체계적인 분류와 구현 무결성 시험방법에 대한 검증방법 연구를 통하여 객관적인 보안 알고리즘의 평가방법을 제시하는 것이 필요하며, 보안 알고리즘의 구현 무결성 검증을 자동화시킨 시험도구의 개발은 평가인력의 효율적인 운용 및 평가기간 단축을 위하여 반드시 수행되어야 한다고 할 수 있다.

보안 알고리즘의 구현 무결성을 검증할 수 있는 평가방법은 미리 검증되어 있는 테스트 벡터를 대입하여 예상 결과 값이 도출되는지를 확인하여 보안 알고리즘의 구현 무결성을 검증할 수 있는 KAT(Known Answer Test)와 랜덤한 데이터를 사용하여 비정상적 동작 오류나 가변 데이터 사용에 대한 구현의 신뢰성을 측정할 수 있는 Modes Test(Monte Carlo Test)가 있다. KAT는 입력 데이터와 출력 데이터의 비교만으로도 검증을 할 수 있는 약식 테스트로서도 사용될 수 있으며, 빠른

평가시간과 정확성을 가지고 있다. 반면 MCT는 정밀 검사의 일종으로 KAT보다 많은 시간을 요하며, 프로그램의 강인성(Robustness)과 구현상의 오류들도 찾을 수 있는 좀더 광범위한 테스트로서 테스트 결과에 신뢰를 높여주는 역할을 한다. MCT는 알고리즘의 구현상의 오류로 발생할 수 있는 취약키(Weak key)의 존재 여부와 간접적이지만 에러 처리 능력(Error Handling)을 평가하기도 한다.[15][16][17]

본 논문에서는 국내외 표준 알고리즘 가운데 SEED, IDEA의 검증 템플리트 DB를 구축한 뒤, 각 알고리즘에 맞는 평가방법을 연구하여 암호 알고리즘의 무결성 시험도구를 구현한다.

II. 본론

1. 검증 방법론

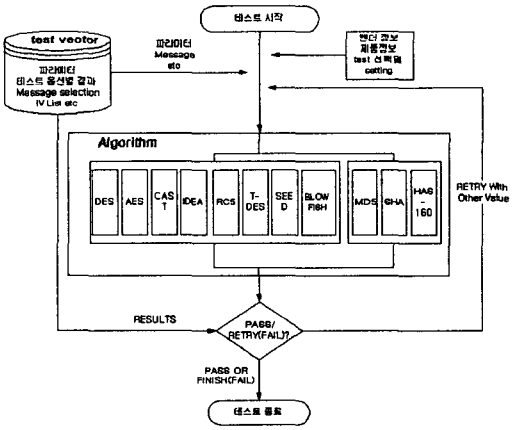

그림 1 KAT(Known Answer Test) 의 구조

KAT(Known Answer Test)는 테스트 하고자 하는 암호 모듈과 동일한 알고리즘을 사용한 신뢰할 수 있는 모듈을 이용하여 입력 데이터와 출력 데이터를 생성하여 테스트 하고자 하는 모듈에 입력, 출력 데이터를 대입하여 테스트하는 모듈에 의한 결과데이터와 비교함으로서 테스트한 암호 모듈의 구현이 알고리즘에 충실하게 구현되었는지의 여부를 측정하는 방법이다. 즉 미리 검증되어 있는 테스트 벡터를 대입하여 예상 결과가 도출되는 지를 확인하여 알고리즘의 무결성을 검증한다.

그림 1은 KAT의 동작에 대한 대략적인 구조를 보여준다. 평가자는 의뢰자(Vendor)로부터 평가할 제품에 대한 제품 정보와 테스트 옵션 에 대한 설정을 제공받아 평가를 시작한다. 평가가 시작되기 이전에 의뢰자는 평가받을 대상 모듈이 표준형식 규격(standard format)에 맞는지를 우선 검사한다.

테스트 데이터베이스에서 제공받은 키값과 IV(Initialize Vector), PLAIN TEXT, 등을 테스트하고자 하는 알고리즘 모듈에 입력하여 나온 결과 데이터를 데이터베이스의 결과 데이터와 비교한다. 테스트의 결과는 PASS와 FAIL로서 표시되며 몇몇 알고리즘은 특성상 다른 검증벡터(Test Vector)를 사용하여 재평가 될 수 있다.

KAT 내에서는 사전에 규격이 정해진 프로토콜을 통해 프로그램이 구동되며, 암호화와 복호화하는 과정을 모두 테스트함으로서 암호 알고리즘이 단 방향성을 띠는 것을 지양하고 실제 구동에 있어서 정확하게 작동하는지에 대하여 초점을 맞추어 테스트를 진행하게 된다. KAT를 수행하기 위해서는 각 알고리즘마다 필요한 파라미터의 정의와 입,출력 파일 형식(I/O File Format)이 정의되어야 하며, 알고리즘별로 제공된 테스트 벡터의 형식도 다양하므로 이를 통합하여 표준 형식을 설정하는 것도 중요하다.

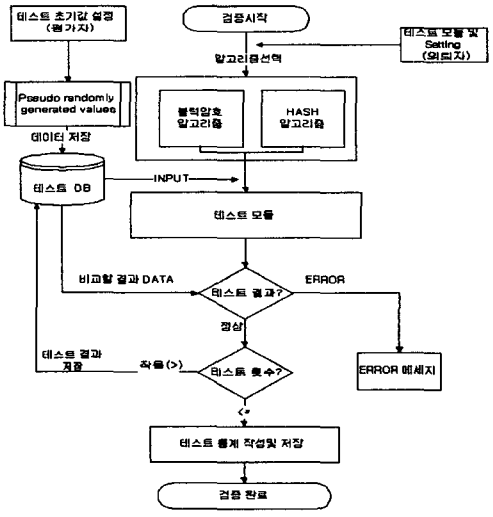

그림 2 MCT 테스트의 구성도

무결성 테스트의 신뢰도를 입증하기 위해 KAT의 프로세스(Process)가 공개되어 있는 상황에서 테스트를 받으려는 의뢰인은 KAT를 통과하기 위해 공개되어 있는 테스트 벡터와 프로세스를 역이용 할 우려가 있다. 이를 방지하기 위해 의사 자

유적으로(pseudo random) 생성된 키 값을 사용하고 테스트 횟수를 증가시켜 다양한 키 값을 적용해 봄으로서 알고리즘상의 Pointer error, 특정한 값에서의 에러 발생 여부를 측정하기 위해 MCT (Monte Carlo Test) 테스트가 추가되었다. MCT는 테스트 하고자 하는 암호 모듈과 동일한 알고리즘을 사용한 신뢰할 수 있는 모듈을 이용하여 무수히 많은 테스트 데이터를 생성한 다음 이 데이터를 사용하여 비정상적 동작 오류나 가변 데이터 사용에 대한 구현의 신뢰성을 측정하고 예측하지 못한 에러의 처리(Error Handling)에 대해서도 테스트한다.

그림 2는 MCT의 전체 구성도 이다. 테스트 과정에서는 평가자가 테스트를 실행하기 위해 랜덤한 값을 생성시키기 위한 초기 값과 테스트에서의 반복횟수 등 초기 설정을 정한다. 의뢰자는 테스트 모듈과 테스트할 알고리즘과 구현 모듈이 가진 특성에 맞는 설정값(I/O 형식과 알고리즘 분류상의 구분 등)을 평가자에게 제공한다. 테스트를 진행하면 평가자가 주어진 초기 값에 맞춰 테스트 벡터를 자동으로 생성해 테스트 데이터베이스에 저장하고 테스트가 시작된다. 테스트는 반복적이고 통계적으로 시행되며, 테스트 도중 에러가 발생하면 테스트는 자동으로 종료된다. 평가자가 지정한 테스트 횟수에 도달하게 되면 저장된 테스트 결과를 가지고 통계 및 분석결과를 파일로 저장한다.

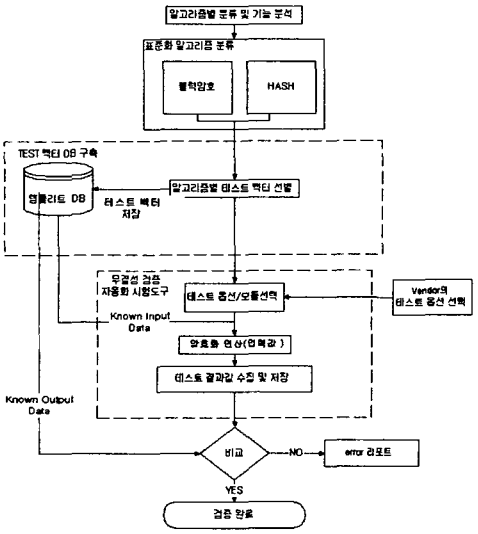

그림 3 무결성 자동화 도구의 개발 구성도

MCT는 특히 테스트 신뢰성의 이유로 평가 절차가 공개되어야 하는 상황에서 테스트 과정과 테스트 벡터가 부정한 방법으로 역이용되는 것을 방지하는 차원에서 더욱 중요한 테스트 과정이라고 할 수 있다.

MCT는 Modes Test와 Monte Carlo Test로 나누어진다. Modes test는 DES 알고리즘을 검증하기 위해 만들어진 검증방법이며 T-DES와 AES의 테스트를 위한 운용 MODE가 추가되면서 MCT로 개념을 확장하고 명칭을 변경하였다. Modes Test는 Monte Carlo Test에 기반 하여 DES, Skip Jack 알고리즘을 위한 테스트이며 알고리즘 구현상의 에러를 판단하며, 외형적인 동작에러를 테스트한다.

본 논문에서는 기본적으로 신뢰할 수 있는 테스트 벡터를 이용하여 테스트 하고자하는 암호 모듈의 무결성을 검증할 수 있는 KAT를 이용하여 암호 모듈의 응답성을 테스트하고, 테스트하는 암호 모듈의 비정상적 동작 오류나 가변 데이터 사용에 대한 구현의 신뢰성과 측정하기 위하여 신뢰할 수 있는 암호 모듈을 이용하여 무수히 많은 테스트 데이터를 생성하여 데이터 베이스로 구축한 다음 MCT를 통하여 암호 모듈의 신뢰성을 측정한다. 자동화 된 검증 도구를 구현하기 위해서는 시험 도구 내에 신뢰할 수 있는 암호 알고리즘이 내포되어야 한다. 그림 3은 자동화 도구의 구성도이다. 알고리즘 표준과 특성별로 분류한 암호 군(群)에서 알고리즘별 테스트 벡터를 선별하여 지정된 포맷에 맞도록 변환하여 테스트 데이터베이스를 구축한다. 저장된 테스트 벡터는 테스트가 진행되면 입력 데이터와 출력 데이터를 비교하는데 쓰인다. 의뢰자가 제공한 정보에 따라 테스트 모듈과 환경설정이 끝나면 테스트를 진행하며 OUTPUT이 입력 데이터에서 계산된 결과 데이터와 다를 경우 에러 메시지를 표시하고 일치할 경우 검증결과를 저장하고 종료하게 된다.

2.블록 암호 알고리즘 검증

1) SEED 알고리즘

SEED 알고리즘의 테스트 방법 및 절차는 그림 4와 같다. 테스트가 시작되면 의뢰자가 선택한 설정(Setting)에 의해 SEED 알고리듬의 I/O 포맷에 맞춰 테스트가 진행된다.[10] 연산의 결과로 128bit의 암호문(Cipher Text)을 얻으면 테스트 DB에서 제공한 결과 데이터와 비교하여 무결성 여부를 상태 창에 표시한 뒤 테스트를 종료하게 된다.SEED 알고리즘의 테스트에 사용되는 데이터베이스 테이블은 표 1과 같다.

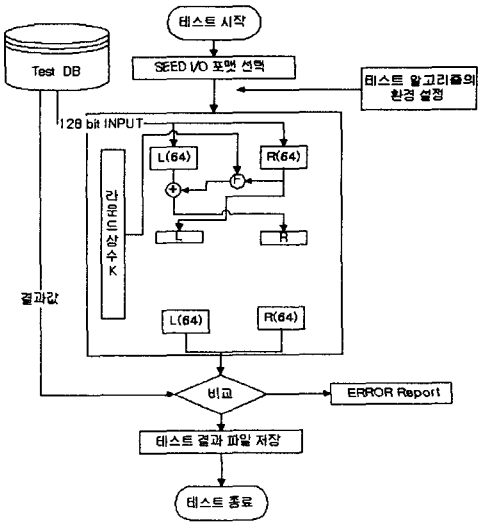

그림 4 SEED의 KAT 구성도

표 1 SEED 알고리즘의 테스트에 사용되는 데이터베이스 테이블

Seed_Tbl		
FIELD_ NAME	DATA_ TYPE	DESCRIPTION
MODE	VARCHAR(5)	동작모드
KEY	CHAR(16)	입력키
PLAIN_ TEXT	VARCHAR(16)	원문
CIPHER_ TEXT	VARCHAR(16)	암호문
STATE	CHAR(7)	테스트 벡터의 용도 (NULL 허용)
ROUND	NUMBER	만일 라운드별 테스트가 필요할 경우

SEED 알고리즘이 테스트를 위한 프로그램을 실행하게 되면 우선 그림 5와 같은 화면이 뜨게 된다. 여기에서 테스트를 위한 동작모드 및 입력 데이터의 크기 와 출력 데이터 즉, 암호화된 데이터를 어떤 형식으로 취할 것인지를 결정하게 된다. 모두 결정하였으면 '검사파일작성' 이라는 버튼을 눌러서 다음 동작을 취한다

사용자가 설정한 값에 의해 키 파일, 원문, 암호문을 생성하게 된다. 파일이 생성되면 프로그램은 사용자로 하여금 테스트 받을 프로그램의 실행 파일의 위치와 파일명을 묻게 된다. 그림 6은 사용자가 다이얼로그를 통해 실행파일을 선택하는 모습이다. 사용자가 실행 파일을 선택하면 프로그램은 이전에 생성된 키파일과 암호문을 실행파일의 인자로 전달해서 실행하게 된다.

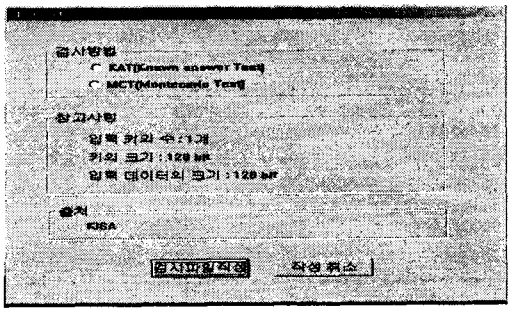

그림 5 SEED 동작 프로그램의 설정모습

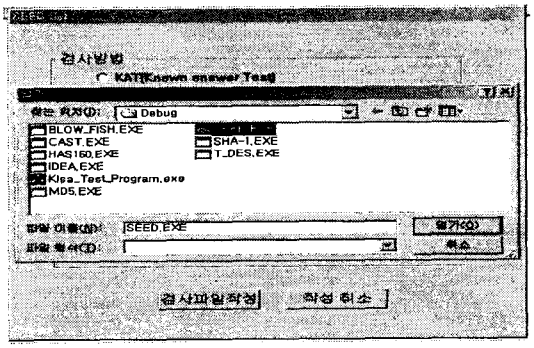

그림 6 SEED 검사 모듈 프로그램의 선택 화면

2) IDEA 알고리즘

그림 7은 IDEA의 테스트 구성도 이다. IDEA는 DES와 같은 4가지 운영모드(ECB, CBC, CFB ,OFB)를 사용하여 테스트 할 수 있다. 테스트가 시작되면 IDEA 모듈과 사용자 설정을 반영하여 운영모드를 결정한다. 데이터베이스는 64bit의 입력 값을 모듈에 공급하고 IDEA의 알고리즘을 수행한뒤 64bit 암호문을 만들어 출력한다. 이 암호문을 테스트 데이터베이스에 있는 결과 데이터와 비교한 뒤 무결성 여부를 결정하여 상태 창에 표시하고 테스트 결과를 파일로 저장하여 테스트를 종료한다.[8]

IDEA 알고리즘의 테스트에 사용되는 데이터베이스 테이블은 표2와 같다.

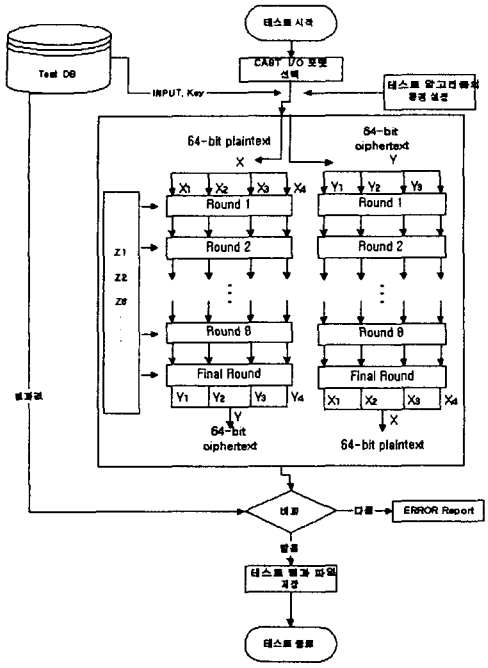

그림 7 IDEA의 테스트 구성도

표 2 알고리즘의 테스트에 사용되는 데이터베이스 테이블

IDEA_Tbl		
FIELD_NAME	DATA_TYPE	DESCRIPTION
MODE	VARCHAR(5)	동작모드
KEY	CHAR(16)	입력 키
CV	CHAR(16)	초기화 벡터
PLAIN_TEXT	VARCHAR(16)	원문
CIPHER_TEXT	VARCHAR(16)	암호문
STATE	CHAR(7)	테스트 벡터의 용도(NULL 허용)
ROUND	NUMBER	만일 라운드별 테스트가 필요할 경우

IDEA 알고리즘도 SEED 마찬가지 형태로 테스트가 진행된다. 그림 8과 같은 설정화면에서 테스트 방식과 모드등을 설정하게 되면 이에 따른 파일들이 생성되고, 그 후에 그림 9에서 보여주는 것 같이 검증받을 프로그램을 선택하여 테스트 하게 된다.

3. 알고리즘별 평가결과 비교

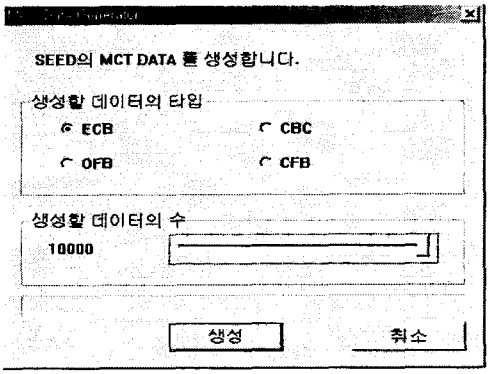

그림 8 MCT용 데이터를 생성하기 위한 설정모습

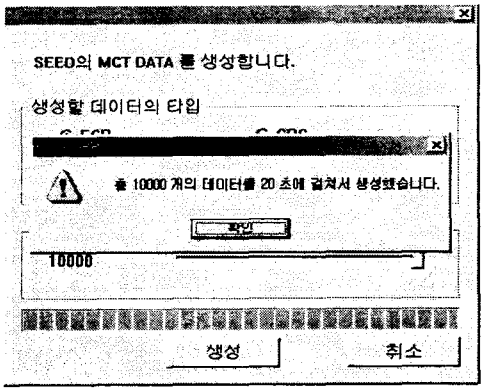

그림 9 MCT 용 데이터 생성모습

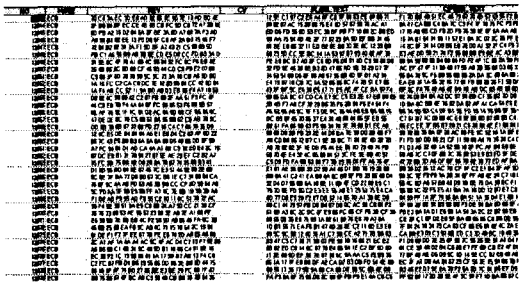

그림 10 생성된 MCT용 데이터

위와 같은 방법의 테스트를 위해서는 각 알고리즘별로 신뢰성 있는 데이터의 구축이 필요하다. 우리는 신뢰성 있는 SEED, IDEA의 알고리즘을 토대로 MCT 용 테스트 벡터를 생성하고 이를 가

지고 실험하였다.

그림 10은 프로그램에서 SEED알고리즘의 MCT를 하기 위해서 테스트 벡터를 자동 생성하도록 하는 모습이다. 사용자는 원하는 데이터의 수를 정하고 프로그램은 설정한 수만큼 테스트 벡터를 생성하여 데이터 베이스에 저장한다.

그림 11은 알고리즘 검증 및 MCT 데이터 생성들의 작업들에 대한 기록을 확인한 예이다.

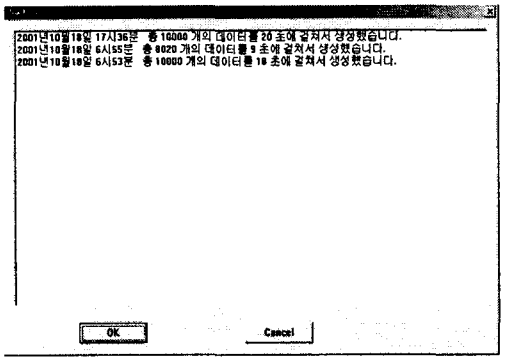

그림 11 프로그램에서 남긴 logo List

구현 및 평가에 사용된 시스템 환경은 다음 표 3과 같다

표 3 구현 시스템 환경

	시스템 사양
CPU	P3-600×2
RAM	512 Mb
OS	Windows2000
개발툴	MS Visual Studio 6

각 알고리즘별 MCT용 데이터를 생성하는데 소요되는 시간은 다음 표 4와 같다.

표 4 MCT용 데이터를 생성하는데 소요된 시간

알고리즘 / 생성수	SEED	IDEA
100	1s 미만	1s 미만
1000	2s	1s
10000	19s	11s

III. 결론

보안 알고리즘의 구현 무결성을 검증할 수 있는 평가방법은 미리 검증되어 있는 테스트 벡터를 대입하여 예상 결과 값이 도출되는지를 확인하여 보안 알고리즘의 구현 무결성을 검증할 수 있는 KAT(Known Answer Test)와 랜덤한 데이터를 사용하여 비정상적 동작 오류나 가변 데이터 사용에 대한 구현의 신뢰성을 측정할 수 있는 Modes Test(Monte Carlo Test)가 있다.

본 논문에서는 표준화된 보안 알고리즘에 대한 종류별 검증 템플리트 데이터베이스를 기반으로 평가자가 IUTs(Implementations Under Test)를 효율적으로 검증할 수 있도록 평가자 중심적인 인터페이스를 구현하여 자동화된 무결성 시험도구를 개발하였다.

향후 좀 더 다양한 알고리즘과 다양한 플랫폼에서 동작하는 프로그램을 개발하고자 한다.

참고문헌

[1] http://www.counterpane.com/blowfish.html (blowfish)
[2] http://www.esat.kuleuven.ac.be/~rijmen/rijndael/
[3] http://www.counterpane.com/bfdobsoyl.html
[4] http://www.rfc-editor.org/rfc/rfc2144.txt
[5] http://www.rfc-editor.org/rfc/rfc2612.txt
[6] http://www.rfc-editor.org/rfc/rfc2612.txt
[7] http://csrc.nist.gov/cryptval/des/tripledes-vectors.zip
[8] http://www.media-crypt.com/pages/fidea.html
[9] http://www.cacr.math.uwaterloo.ca/hac/about/chap7.pdf
[10] http://www.kisa.or.kr/technology/sub1/128-seed.pdf
[11] http://www.kisa.or.kr/evaluation/
[12] http://www.tta.or.kr/index.html
[13] http://www.tta.or.kr/TtaContent?StnNum=TTAS.KO-12.0011
[14] http://www.rfc-editor.org/rfc/rfc1321.txt
[15] http://csrc.nist.gov/cryptval/shs.html
[16] http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.pdf