경량 스트림 암호 구현 적합성 검증 도구

강 주 성[†] · 신 현 구[††] · 이 옥 연[†††] · 홍 도 원[††††]

요 약

암호 알고리즘의 구현 적합성 평가는 제품에 사용될 알고리즘이 설계자의 의도에 맞게 정확히 구현되어 있는지를 평가하는 것이다. 대표적인 구현 적합성 평가 시스템으로는 미국 NIST 주관의 암호 모듈 적합성 검증 프로그램(CMVP)을 들 수 있다. CMVP는 미 연방표준 FIPS에 포함된 암호 모듈의 구현 적합성을 평가하는 것이며, FIPS 내에는 스트림 암호가 없는 관계로 CMVP 세부 항목에 스트림 암호에 대한 검증 도구는 포함되어 있지 않다. 본 논문에서는 CMVP에는 포함되어 있지 않아서 아직 구현 적합성 검증 기법이 알려지지 않고 있지만, 주로 무선 환경에서 표준으로 제정되어 널리 사용되고 있는 블루투스 표준 스트림 암호 E0와 제3세대 비동기식 이동통신 표준 스트림 암호 A5/3, WEP과 SSL/TLS 등에 사용되는 스트림 암호 RC4에 대한 구현 적합성 검증 방법을 제안하고, JAVA로 구현한 검증 도구를 보여준다.

키워드 : 암호 모듈 적합성 검증 프로그램, 경량 스트림 암호

Validation Testing Tool for Light-Weight Stream Ciphers

Ju-Sung Kang[†] · Hyun Koo Shin[††] · Okyeon Yi[†††] · Dowon Hong[††††]

ABSTRACT

Cryptographic algorithm testing is performed to ensure that a specific algorithm implementation is implemented correctly and functions correctly. CMVP(Cryptographic Module Validation Program) of NIST in US is the well-known testing system that validates cryptographic modules to Federal Information Processing Standards (FIPS). There is no FIPS-approved stream cipher, and CMVP doesn't involve its validation testing procedure. In this paper we provide validation systems for three currently used light-weight stream ciphers: Bluetooth encryption algorithm E0, 3GPP encryption algorithm A5/3, and RC4 used for WEP and SSL/TLS protocols. Moreover we describe our validation tools implemented by JAVA programing.

Key Words : CMVP(Cryptographic Module Validation Program), Light-weight Stream Cipher

1. 서 론

정보보호 제품에 사용되는 암호 모듈은 대칭키 및 비대칭키 암호 알고리즘, 해쉬 함수, 난수 발생기 등과 같은 여러 가지 세부 요소들로 이루어져 있다. 원하는 안전성 레벨(security level)과 상호 연동성을 보장하기 위해서 대부분의 암호 모듈은 표준화된 암호 알고리즘들을 사용한다. 이러한 암호 알고리즘들을 올바르게 구현하는 것은 이들의 구조적인 복잡성 때문에 그리 간단한 작업이 아니다. 그러므로 보안 관련 제품을 설계 또는 제조하는 업체는 항상 암호 알고리즘의 올바른 구현에 심혈을 기울여야 한다. 실제로 NIST (National Institute of Standards and Technology)에 의해서 조사된 바에 의하면 2002년 현재 미연방표준 FIPS(Federal Information Processing Standards)의 구현 적합성 테스트에 응모한 암호 모듈들 중 거의 절반 정도에서 결함이 발견되었다고 한다[1].

암호 모듈의 구현 적합성 검증 도구의 대표적인 예는 CMVP(Cryptographic Module Validation Program)[2]이다. CMVP는 1995년 미국 표준국인 NIST와 캐나다의 CSE (Communications Security Establishment)가 공동으로 개발한 것으로 FIPS 140을 따르는 암호 모듈의 구현 적합성 검증을 위한 프로그램이다. 1995년 7월에 시작된 CMVP는 1994년 NIST가 제정한 FIPS 140-1("Security Requirement for Cryptographic Modules")[3]과 2001년 개정된 FIPS 140-2[4], 그리고 암호 알고리즘 관련 FIPS 표준문서들을 근간으로 만들어졌다. CMVP에서 관장하는 암호 알고리즘으로는 대칭키 알고리즘, 비대칭키(공개키) 알고리즘, 해쉬 함수, 메시지 인증 알고리즘, 난수 생성 알고리즘 등이 있다. 특히, FIPS에서 표준으로 승인한 대칭키 알고리즘은 AES [5], DES[6], Triple-DES[6], Skipjack[7] 네 개 뿐이며, 이들

† 정 회 원 : 국민대학교 수학과 조교수
†† 정 회 원 : (주)엑서스 테크놀로지
††† 정 회 원 : 국민대학교 수학과 조교수
†††† 정 회 원 : 한국전자통신연구원 선임연구원
논문접수 : 2005년 2월 18일, 심사완료 : 2005년 4월 20일

은 모두 블록 암호(block cipher)이다. FIPS 표준으로 승인된 스트림 암호는 현재까지 없기 때문에 이에 대한 구현 적합성 검증 도구도 CMVP에는 포함되어 있지 않다.

블록 암호가 64-비트, 128-비트 등과 같이 큰 블록 단위로 동작하는 것과는 달리 스트림 암호는 일반적으로 한 비트 단위로 동작한다. 블록 암호와 스트림 암호는 알고리즘 내부의 핵심 구성 요소들도 상당한 차이를 보인다. 블록 암호는 S-box와 같이 비선형성과 복잡성(confusion)을 위한 논리와 자리바꿈(swapping) 또는 순열(permutation)과 같이 확산(diffusion) 효과를 부여하기 위한 논리들의 반복 조합으로 구성되는 것이 일반적이다. 스트림 암호는 일반적으로 LFSR(Linear Feedback Shift Register) 또는 FCSR(Feedback Carry Shift Register) 등을 이용하여 주기가 긴 수열을 생성한 후 여과 함수(feedforward function)를 작용시켜 최종 출력 수열을 생성한다. 이와 같이 블록 암호와 스트림 암호는 구조적인 차이를 가지고 있으므로 구현 적합성 검증 방법도 다르게 접근하여야 한다.

본 논문에서는 미연방표준 FIPS에는 포함되어 있지 않지만 최근 무선 통신 환경에서 다양한 국제표준으로 널리 사용되고 있는 경량(light-weight) 스트림 암호에 대한 구현 적합성 검증 도구를 제안한다. 블루투스 표준 스트림 암호인 E_0[8], 제3세대 비동기식 이동통신 표준화 그룹인 3GPP 스트림 암호 A5/3[9], 그리고 SSL(Secure Socket Layer)[10] 프로토콜에 포함되어 있으며, 다양한 암호 제품에서 사실상의 표준으로 널리 사용되고 있는 RC4[11]에 대한 구현 적합성 검증 방법을 논한다. 또한, 이들에 대한 검증 프로그램을 JAVA로 구현한 결과를 제시한다.

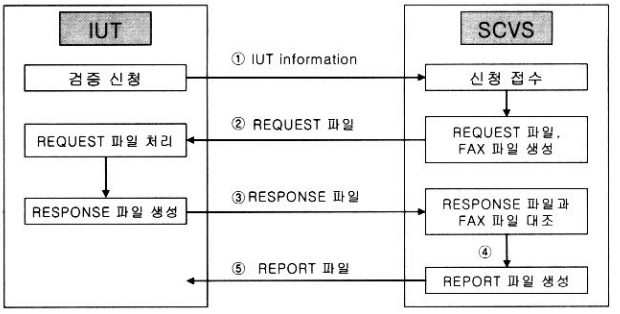
E_0, A5/3, RC4는 경량 스트림 암호의 효시가 되는 알고리즘들로 각기 서로 다른 구조적 특성을 지니고 있다. E_0는 전통적인 스트림 암호의 구조를 갖는 알고리즘으로 네 개의 LFSR과 덧셈 생성기(summation generator)로 이루어져 있다. A5/3는 블록 암호 KASUMI[12]를 핵심 함수로 사용하는 스트림 암호이며, RC4는 소프트웨어 구현에 적합한 스트림 암호로서 S-box에 랜덤성을 가하여 랜덤 순열을 생성하는 구조이다. 그러므로 이 세 가지 다른 형식의 스트림 암호에 대한 구현 적합성 검증 방법은 향후 다양한 형태의 스트림 암호에 대한 검증 기법 연구에도 원용될 것이란 점에서 그 의미가 크다고 하겠다.

2. 스트림 암호 구현 적합성 검증 방법

우리가 제안하는 스트림 암호 알고리즘 구현 적합성 검증 시스템을 편의상 SCVS(Stream Cipher Validation System)라 하고, 평가 대상이 되는 구현물을 CMVP에서와 같이 "IUT(Implementation Under Test)"라 부르기로 하자.

SCVS는 세 가지 스트림 암호(E_0, A5/3, RC4)를 (그림 1)에 나타난 절차에 따라 검증을 실시한다.

(그림 1)에 나타난 세부 절차는 다음과 같다.



(그림 1) SCVS의 검증 절차

① IUT는 구현물에 관한 정보를 SCVS에게 보냄으로써 검증을 신청한다.

② SCVS는 IUT의 해당 알고리즘에 관련된 REQUEST 파일과 FAX 파일을 생성한다. REQUEST 파일은 IUT에게 전송하고 FAX 파일은 보관한다.

③ IUT는 REQUEST 파일의 데이터를 이용하여 RESPONSE 파일을 생성하고 SCVS에게 RESPONSE 파일을 전송한다.

④ SCVS는 RESPONSE 파일과 FAX 파일을 대조함으로써 IUT에 대한 검증을 수행한다.

⑤ 평가의 결과는 "통과(PASS)" 또는 "실패(FAIL)"로 이루어지며 REPORT 파일에 기록된다.

검증 과정에서 사용되는 데이터는 다음과 같은 네 가지의 파일로 구성된다.

- REQUEST 파일 : SCVS가 IUT에게 보내는 암호 알고리즘 관련 테스트 벡터를 포함하는 파일
- FAX 파일 : SCVS가 보관하는 REQUEST 파일의 테스트 벡터에 대한 예상 결과값을 포함하는 파일
- RESPONSE 파일 : IUT가 SCVS로부터 받은 REQUEST 파일의 테스트 벡터를 이용한 결과값을 포함하는 파일
- REPORT 파일 : 검증에 관한 결과로써 "성공"과 "실패"를 표현하며, 결과가 "실패"인 경우에 IUT가 오류를 수정할 수 있도록 관련된 정보를 포함하는 파일

스트림 암호의 동작은 일반적으로 초기화 단계, 키스트림 생성 단계, 암호화 단계로 구분할 수 있다. 초기화 단계의 검증에서는 암호화키의 입력과정과 초기값의 입력과정에 대한 검증이 포함된다. 키스트림 생성 단계에서는 알고리즘 내부 함수들의 동작과정에 대한 검증을 수행한다. 암호화 단계는 동기식 스트림 암호의 경우 암호화 과정이 단순히 평문과 키스트림의 XOR연산으로 이루어지므로 정확히 구현된 것으로 간주하여 생략하고, 비동기식(Self-Synchronizing) 스트림 암호의 경우 암호화 과정에 대한 검증이 필요할 것으로 보인다. 본 논문의 SCVS에 포함된 스트림 암호는 모두 동기식 스트림 암호이므로 암호화 과정은 모두 정확히 구현된 것으로 간주한다.

구현 적합성 검증 방법은 크게 기지 출력 검증(Known Answer Test)과 몬테카를로 검증(Monte Carlo Test)으로 이루어진다. 기지 출력 검증은 주어진 테스트 벡터에 대한 출력값과 예상 출력값에 대한 비교로 평가가 이루어진다. IUT는 주어진 테스트 벡터를 이용해서 결과값을 산출한다. 테스트 벡터는 스트림 암호의 각 구성 요소에서 사용되는 연산이 표준에서 제시한 방법에 따라 정확히 구현되었는지를 확인할 수 있도록 설계된다. 테스트 벡터의 구성은 각 알고리즘을 구성하고 있는 함수들이 독립적으로 정확한 동작 수행 여부를 확인할 수 있도록 주어진다. 한편, 몬테 카를로 검증은 난수발생기를 이용하여 생성된 테스트 벡터에 대하여 출력값과 예상 출력값을 비교함으로써 평가가 이루어진다. 몬테 카를로 검증은 임의의 값을 사용함으로써 기지 출력 검증에서 발견하지 못한 오류를 찾아내기 위한 테스트이다.

3. 스트림 암호 검증 시스템 설계

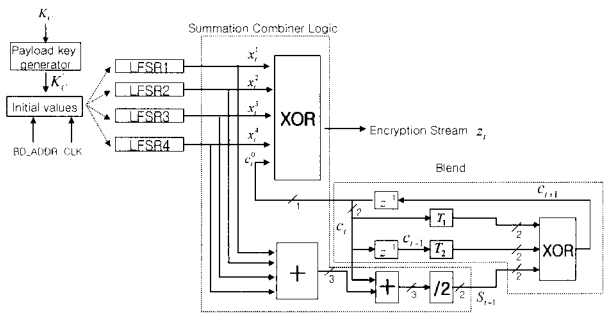
본 절에서는 SCVS에 포함된 세 가지 경량 스트림 암호인 E_0, A5/3, RC4에 대한 구현 적합성 검증 시스템을 설계한 세부 내용을 기술한다. 각 알고리즘에 대하여 내부 함수들에 대한 검증을 실시할 수 있는 테스트 벡터들을 찾아내는 것이 검증 시스템 설계의 핵심이라 할 수 있다.

3.1 E_0 알고리즘 구현 적합성 검증

E_0 알고리즘은 블루투스에서 각 개체들 간의 단대단(Point-to-Point) 암호화에 사용되는 알고리즘이다. 4개의 크기가 서로 다른 LFSR을 사용하는 Summation Generator 기반의 스트림 암호이며, 내부적으로는 2개의 2-비트 메모리가 사용된다. 현재 블루투스는 버전 1.2[8]까지 발표되어 있다.

E_0 알고리즘은 비밀키(K_C)를 이용하여 128비트 암호화키 K_C'을 생성하고, K_C'과 48비트 BD_ADDR, 26비트 CLK를 이용하여 4개의 LFSR과 두 개의 2비트 메모리 c_0, c_1을 초기화하는 과정을 거쳐 키스트림을 생성한다. (그림 2)는 E_0 알고리즘을 묘사한 것이다.

E_0에 대한 구현 적합성 검증은 알고리즘 각 부분의 동작에 초점을 둔 5가지 기지 출력 검증과 하나의 몬테카를로 검증으로 구성된다. 각 테스트는 출력된 첫 번째 128비트를 확인하는 것으로 적합성을 판단하며, 출력된 128비트는 가장 왼쪽 바이트를 최상위 바이트로 하고, 한 바이트의 가장 오른쪽 비트를 최상위 비트로 표현한다.



(그림 2) E_0 알고리즘 블록도

3.1.1 K_C' 생성 과정에 대한 기지 출력 검증

K_C'의 생성은 다음의 식으로 표현할 수 있다.

$$K_C' = g_2^{(L)}(x)(K_C(x) \bmod g_1^{(L)}(x))$$

블루투스는 최대 허용된 키 길이를 의미하는 L이라는 변수($1 \le L \le 16$)를 포함하고 있다. 이는 자신에게 허용된 키 길이를 바이트 단위로 표현한 것으로써, 각각의 블루투스 장치는 자신이 가진 L의 범위 내의 모든 경우에 대하여 정확한 구현을 수행할 수 있어야 한다. 테스트는 L의 크기에 따라 $g_1(x)$보다 작은 차수, 같은 차수 그리고 큰 차수로 구성된 임의의 벡터에 대하여 연산을 정확하게 수행하는지를 검증한다. $L=8$인 경우에 K_C에 대응하는 K_C'에 대한 테스트 벡터의 예를 살펴보면 다음과 같다.

- K_C : 770f05c3cb017aec
  → K_C' : 352711be1e7ac8f352e55fca120dccdc
- K_C : 18455bfa8f3145e6e
  → K_C' : 52737127c719f520e5f67fc41ae86901
- K_C : 26871ab6e88a7baec45079cf99c6a0bc
  → K_C' : 6728ca79005673fdd248dae5fc199a5f

3.1.2 키 입력 과정에 대한 기지 출력 검증

키 입력 과정은 총 202비트의 입력값이 초기화 과정에서 표준에서 설명한 위치에 정확히 설정되는지를 검증한다. 세 가지 입력 값의 모든 비트를 "0"으로 초기화 한 뒤, K_C'의 최상위 비트부터 "1"의 값을 지정하면서 BD_ADDR, CLK 순으로 수행한다. 테스트 횟수는 총 202회로 <표 1>과 같이 변화한다.

〈표 1〉 E_0 키 입력 과정 검증용 테스트 벡터

No.	K_C'	BD-ADDR	CLK	128-bit Output
1	0100000000000000 0000000000000000	000000000000	00000000	9adad6aa2231b6d2 1d3d3409305ad476
2	0200000000000000 0000000000000000	000000000000	00000000	19cc61e643e3974a 82cc8f5969279801
⋮	⋮	⋮	⋮	⋮
129	0000000000000000 0000000000000000	010000000000	00000000	b7bdde1d2477a921 be76e515ea57cc06
⋮	⋮	⋮	⋮	⋮
177	0000000000000000 0000000000000000	000000000000	01000000	3e91af0d3b8459ff ab9712608b9636dd
⋮	⋮	⋮	⋮	⋮
202	0000000000000000 0000000000000000	000000000000	00000002	0a8cbc1c67b3d10f 38d0be870492c63f

〈표 2〉 E_0 LFSR 동작 검증용 테스트 벡터

K_C'	BD-ADDR	CLK	128-bit Output
0000000000000000 0000000001010000	000700000000	00000103	0000000000000000 0000000000000000
0000000000000000 8000000000010000	000700000000	00000000	457f0256502b2008 1fad4f49bce759c5
0000000000000000 0080000000010000	000700000000	00000000	ac0cfacc5c7d2e8b dff45a3a4879a20b
0000000000000000 0000000000018000	000700000000	00000000	a2f2495745571421 438dd13083ee5b5e
0000000000000000 0000000000010080	000700000000	00000000	aa826e1201dc0319 50f550cfc8141b26

3.1.3 LFSR 동작 과정에 대한 기지 출력 검증

총 5가지 테스트 벡터로 구성되며, 첫 번째 테스트 벡터는 각각의 LFSR 내부가 모두 채워지는 시점에서 정확한 동작을 수행하는지 테스트한다. 정확한 동작을 수행하였다면 출력 값은 모두 "0"인 수열을 생성하도록 테스트 벡터를 구성한다. 나머지 4가지 테스트 벡터는 각 LFSR의 동작을 검증한다. 각 LFSR이 정확히 작동하는 지에 대한 테스트로써, 테스트되는 하나의 LFSR을 제외한 나머지 LFSR은 모두 "0"으로 설정되도록 하고, 테스트되는 LFSR은 키 입력 과정이 끝나는 시점, 즉 마지막 입력값이 "1"이 되고 나머지는 모두 "0"이 되도록 테스트 벡터를 설정한다. 이때, Summation Combiner Logic의 입력값은 기껏해야 하나의 비트만 1이 되므로 LFSR의 출력값이 전체의 출력값이 되는 결과를 갖는다. 구체적인 테스트 벡터는 <표 2>에 나타나 있다.

3.1.4 Blend 함수 동작 과정 기지 출력 검증

Blend 함수는 두 개의 T-table과 두 개의 2비트 메모리 c_{t-1}, c_t로 구성된다. 테스트벡터는 2가지로 구분되며, 하나는 두 개의 T-table의 특징을 이용한 것이다. 즉, c_{t-1}과 c_t가 모두 "00"으로 초기화 되는 시점에서 LFSR은 모두 "0"으로 설정된 것과 같은 역할을 하고 c_{t-1} 과 c_t를 "00"과 "11"이 되도록 설정하게 하는 테스트 벡터를 선택한다. 이 경우에 c_{t-1}과 c_t는 "11", "10", "01", "00"이 반복되는 특성을 갖고 c_t의 하위 비트가 출력 비트가 되므로 "1"과 "0"이 반복되는 수열이 생성되게 된다. 두 번째 테스트 벡터는 2비트 메모리 c_{t-1}, c_t의 입출력과 c_{t+1}을 계산하는 과정을 테스트한다. 세 개의 2비트 변수 c_{t-1}, c_t, c_{t+1}에서 발생할 수 있는 모든 경우의 수는 2^6가지이지만, 실질적으로 16가지 경우는 발생하지 않는다. 그러므로 테스트는 나머지 48가지 경우 모두가 발생하는 벡터를 선택하여 테스트한다. 구체적인 테스트 벡터의 예는 다음과 같다.

- K_C' : 00004005000500000100000011040000
- BD_ADDR : 85d702020002
- CLK : 00108400
- 출력값 : 6969694ed73c26645f6a410a2f656108

3.1.5 Summation Combiner Logic 과정에 대한 기지 출력 검증

Summation Logic 과정에서 정수의 덧셈연산과 나눗셈 연산이 정확하게 수행하는가를 검증한다. 테스트 벡터는 Summation Logic의 입력 비트인 4개의 LFSR 출력 비트와 c_t가 이루는 2^6가지 경우가 모두 발생하는 벡터를 선택한다. 구체적인 테스트 벡터의 예는 다음과 같다.

- K_C' : 000000e20048c9b8a4ed760759050100
- BD_ADDR : 0123456789ab
- CLK : 00000000
- 출력값 : 6edcfbc9127e4bf3dd95419e869803aa

3.1.6 몬테카를로 검증

SCVS는 IUT가 임의로 각각 100개씩의 K_C', BD_ADDR, CLK를 IUT에게 제공한다. IUT는 제공된 하나의 벡터에 대하여 202비트 키스트림을 생성하여 새로운 K_C', BD_ADDR, CLK를 생성하는 과정을 1,000번 반복한다. 몬테카를로 검증을 위한 테스트 벡터의 예는 <표 3>에 나타나 있다.

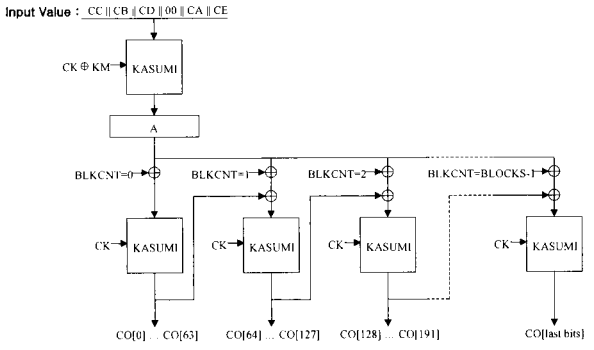
〈표 3〉 E_0 몬테카를로 검증용 테스트 벡터의 예

No.	K_C'	BD-ADDR	CLK	Answer
1	d203bb4f40443c5e b62c73233fa01bb0	4ec95b8d0699	4a125703	ef0273165fa2af1b6383519d2f 48afe0a190521eae8ff6556101
⋮	⋮	⋮	⋮	⋮
100	e46b026fa9c3f0cd 548073d14e03fc09	5f9120e4ac45	8bb80c02	c7255e34fbcacb9ef350a922fe db9837412325f7c3785532ac01

3.2 A5/3 알고리즘 구현 적합성 검증

3GPP TS55.216[9]에서는 3개의 암호 알고리즘 (GSM A5/3, ECSD A5/3, GPRS GEA3)을 제안하고 있다. 이 알고리즘은 모두 블록 암호 KASUMI[12]를 이용한 변형된 OFB 모드 형태의 핵심함수 KGCORE를 키스트림 생성기로써 사용하는 스트림 암호이다. A5/3의 적합성 검증을 엄밀히 수행하기 위해서는 블록 암호 KASUMI에 대한 검증이 전제되어야 한다. 블록 암호의 구현 적합성 검증 방법은 NIST에서 AES, DES, Skipjack 등에 대해서 실시한 방법과 국내 표준 블록 암호 SEED에 대한 검증 방법[13]이 알려져 있다. KASUMI에 대한 구현 적합성 검증도 유사한 방법으로 가능할 것이므로 여기에서는 KASUMI에 대한 세부적인 검증은 생략하기로 한다. 알고리즘의 핵심 함수인 KGCORE의 블록도는 (그림 3)과 같다.

KGCORE 함수는 128비트 암호화키 CK와 64비트의 입력값(IV)을 이용하여 64의 배수가 되는 비트 스트림을 생성하며, 피드백 데이터는 고정된 64비트 레지스터 A와 증가하는 64비트 블록 카운터 BLKCNT가 된다. KGCORE 함수는 키의 길이가 128비트로 고정되어있으므로, 64비트 키를 사용하는 세 가지 알고리즘은 키를 반복해서 연접함으로써 128비트 키를 생성하여 사용한다. 각 알고리즘의 검증은 64비트



(그림 3) KGCORE 함수

레지스터 A를 초기화하는 과정, IV를 입력하는 과정, 키스트림을 생성하는 과정으로 나누어 검증을 수행한다. 검증에 사용되는 테스트 벡터는 가장 왼쪽 바이트의 가장 왼쪽 비트를 최상위 비트로 표현한다. 여기에서는 GSM A5/3에 대한 내용만 기술하기로 한다.

GSM A5/3 알고리즘은 KGCORE 함수의 입력 IV 중에서 CC에 해당하는 22비트 COUNT를 제외한 나머지 42비트를 고정하여 사용하며 64비트 키를 사용한다.

3.2.1 GSM A5/3 초기화 과정 기지 출력 검증

초기화 과정 검증은 64비트 비밀키 K_C가 128비트로 확장되어 고정된 128비트 KM과의 XOR 연산이 정확하게 수행되어 64비트 레지스터 A를 생성하는 과정을 검증한다. 입력되는 IV는 모두 "0"으로 설정하고 64비트 K_C의 최상위 비트부터 "1"의 값을 지정하면서 수행한다. 총 64회의 테스트가 수행되며 테스트 벡터는 <표 4>와 같이 구성된다.

〈표 4〉 A5/3 초기화 과정 검증용 테스트 벡터

No.	K_C	Output block for GSM	
		Block1	Block2
1	8000000000000000	495020bc0c3411a c7272ab71e3e780	7fad42cff407fb7 26a0038cdefde40
2	4000000000000000	5c8e36f0ec588e6 81e7ec29e2ec9c0	5898529cde09c17 6e59546e3b02ac0
⋮	⋮	⋮	⋮
64	0000000000000001	b0e7bad55a46336 4c46a6351f78580	755f13ad823f103 e129cc209de4dc0

3.2.2 GSM A5/3 IV 입력 과정 기지 출력 검증

KGCORE 함수로의 입력값 64비트 중에 CC에 해당하는 22비트 COUNT 값만이 변수로 작용한다. 그러므로 고정된 42비트에 대하여 COUNT가 정확히 입력되는가를 확인한다. 테스트 벡터는 CK를 모두 "0"으로 고정하고, COUNT에 해당하는 값의 최상위 비트부터 "1"을 지정하면서 수행한다. 구체적인 테스트 벡터의 예는 다음과 같다.

- COUNT : 800000
  - BLOCK1 : 60cf2db1d414fd626d3bcf38775140
  - BLOCK2 : c78615373244ab6e9798c2e8f114c0
- COUNT : 400000
  - BLOCK1 : 010e87d07974c305c0d5594bf28500
  - BLOCK2 : 165814401812f949e82ec6f56d9440
- COUNT : 200000
  - BLOCK1 : a3050455e48db7b6bd29fa01be5c40
  - BLOCK2 : a10ae3a9d18205d70f958b8ac1a580

3.2.3 GSM A5/3 키스트림 생성 과정에 대한 기지 출력 검증

키스트림 생성 과정 검증은 레지스터 A와 BLKCNT 그리고 피드백 벡터의 XOR 연산의 정확성을 검증한다. 입력 벡터와 키를 모두 "0"으로 고정하고 생성되는 키스트림 블록을 확인함으로써 검증을 수행한다. 구체적인 테스트 벡터의 예는 다음과 같다.

- COUNT : 000000
  - BLOCK1 : af67271520b09c0fff195efa49f1c0
  - BLOCK2 : 92468cb3d57b3252ab21db3e19d080

3.2.4 GSM A5/3 몬테 카를로 검증

SCVS는 임의로 각각 100개씩의 K_C와 COUNT를 생성하여 IUT에게 제공한다. IUT는 제공된 하나의 벡터에 대하여 2개의 키스트림 블록을 생성하여 첫 번째 블록은 새로운 K_C로 두 번째 블록의 상위 22비트는 새로운 COUNT로 생성하는 과정을 1,000번 반복한다.

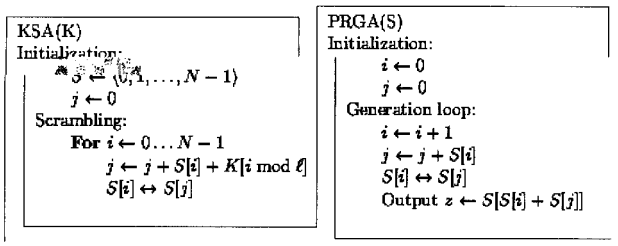
3.3 RC4 알고리즘 구현 적합성 검증

RC4는 바이트 단위의 암복호화를 위해서 RSA Data Security사의 Rivest에 의해서 1987년도에 개발된 스트림 암호이다[11]. 1994년도에 인터넷을 통해 알고리즘이 공개되었으며, 기존의 하드웨어 기반 스트림 암호와 달리 고속의 소프트웨어 구현을 목적으로 개발되었다. RC4는 현재 SSL, MS Window, WEP, Oracle Secure SQL 등에 사용되면서 가장 널리 사용되는 스트림 암호가 되었다.

RC4는 40-비트부터 256-비트까지의 가변 길이 키를 초기 치환으로 변환시키는 KSA(key-scheduling algorithm) 부분과 초기 치환을 이용해서 출력 수열을 발생시키는 PRGA(pseudo-random generation) 부분으로 구성되어 있다. KSA 부분과 PRGA 부분은 (그림 4)에 나타나 있다.

3.3.1 키 입력 과정 기지 출력 검증

KSA에서 128비트 키는 8비트 워드의 단위로 나누어져 덧셈 연산에 사용된다. 키 입력 과정 테스트는 8비트 워드의 가능한 경우의 수인 $2^8 = 256$가지 경우의 입력이 가능하도록 테스트 벡터를 선택하여 각각의 입력에 대한 2^8을 법으로 하는 덧셈 연산의 과정을 테스트한다. 테스트 벡터는 각 바이트별로 0~255 사이의 값이 입력되도록 <표 5>와 같이 구성된다.



(그림 4) RC4의 구성

〈표 5〉 RC4 키 입력 과정 검증용 테스트 벡터

No.	Key	128-bit Output
1	000102030405060708090a0b0c0d0e0f	e99c40f947e219cc06db97c60edd2a4f
2	101112131415161718191a1b1c1d1e1f	4bb000ccfd001bf237bee5d78c3bfca2
⋮	⋮	⋮
16	f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff	9917f066cc905aeaafe5e8fdea6a7983

3.3.2 Swap 함수 기지 출력 검증

Swap 함수는 단순히 치환 S의 두 값을 교환하는 함수이지만 RC4에서 가장 중요한 역할을 하는 함수이다. Swap 함수 동작 과정의 테스트 벡터는 총 8개로 구성된다. 테스트 벡터는 KSA 과정에서 8비트 크기로 사용되는 K[$i \bmod l$]가 혼합 과정에서 하나의 값으로 고정되도록 구성된다. 즉, 키를 일정한 값으로 유지하여 Swap 함수의 동작을 검증한다. <표 6>은 Swap 함수 동작 과정의 입력 테스트 벡터이다.

〈표 6〉 RC4 Swap 함수 검증용 테스트 벡터

No.	Key	128-bit Output
1	01010101010101010101010101010101	06080e0e18202929393495766768783
2	03030303030303030303030303030303	71a211d6404bea71cd1be2ad397f9f66
⋮	⋮	⋮
8	ffffffffffffffffffffffffffffffff	6d252f2470531bb0394b93b4c46fdd9c

3.3.3 몬테 카를로 검증

SCVS는 임의의 난수 값으로 100개의 128비트 키 K를 생성하여 IUT에게 제공한다. IUT는 주어진 K를 이용하여 128비트 키스트림을 생성하고, 생성된 키스트림을 새로운 K로 설정하는 과정을 1,000번 반복한다. 1,000번의 반복 횟수는 AESAVS의 몬테 카를로 검증 횟수를 참고한다.

4. JAVA 시뮬레이션 구현

3절에서 기술한 내용을 바탕으로 스트림 암호 알고리즘 구현 적합성 검증 시스템의 검증 과정을 JAVA 프로그램을 이용하여 구현하였다. 검증 프로그램은 SCVS 프로그램과 IUT 프로그램으로 구성된다. SCVS 프로그램은 검증을 주관하는 프로그램으로써 IUT 프로그램을 통해 전송된 데이터를 평가하여 결과를 제공하는 프로그램이다. IUT 프로그램은 검증 받고자 하는 알고리즘의 평가 데이터를 SCVS에게 전송하는 프로그램이다.

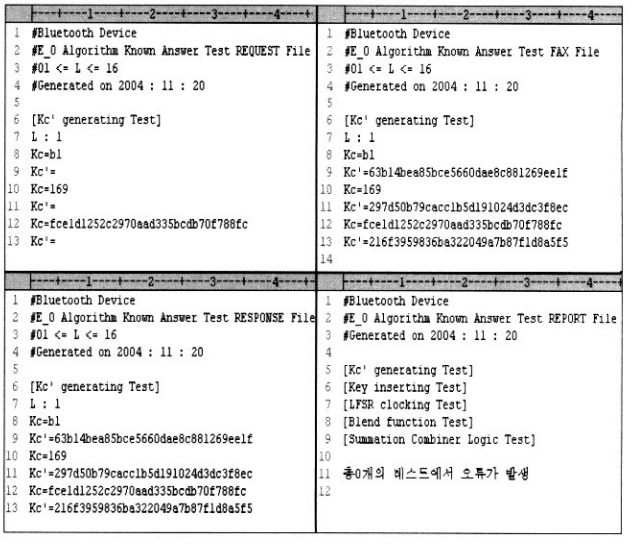
SCVS 프로그램은 검증 통과 목록을 보여주며 내부적으로는 서버로써의 기능을 수행한다. 검증 통과 목록을 보여주는 부분은 각 알고리즘에 해당하는 이미 검증이 완료된 IUT에 대한 목록을 볼 수 있다. 서버의 기능은 프로그램과 함께 시작된다.

IUT 프로그램은 세 가지 부분으로 이루어져 있다. 첫 번째 부분은 스트림 암호 알고리즘을 선택하고 구현적합성 검증을 요청하기 위한 IUT의 정보를 입력하여 SCVS에게 기지 출력 검증(KAT)과 몬테 카를로 검증(MCT)을 요청하는 역할을 한다. 두 번째 부분은 SCVS로부터 받은 REQUEST 파일에 따라 생성된 RESPONSE 파일을 SCVS에게 보내기 위해 파일을 선택하여 전송하는 역할을 한다. 세 번째 부분은 검증이 진행되는 과정을 보여준다.

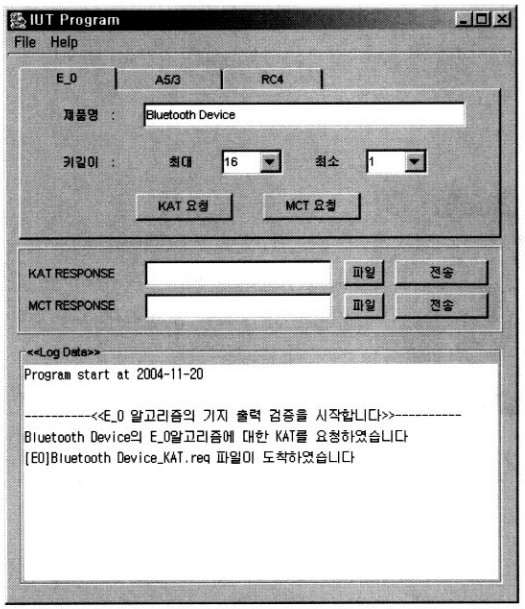
E_0에 대한 검증을 예로 하여 프로그램이 동작하는 과정을 설명해 보자. 먼저 (그림 5)는 E_0 알고리즘의 기지 출력 검증에 사용되는 네가지 파일(REQUEST 파일, FAX 파일, RESPONSE 파일, REPORT 파일)의 실제 구성을 보여준다.

E_0 알고리즘을 이용한 제품명은 "Bluetooth Device"라고 하고, 키길이는 최대 "16", 최소 "1"로 지정한다. 먼저 알고리즘 선택 탭에서 "E_0"선택하고 제품명에는 "Bluetooth Device"를 입력하며, 키길이는 최대 "16", 최소 "1"을 선택한다. "KAT 요청"버튼을 클릭하면 SCVS에게 정보를 전송하고 SCVS는 REQUEST 파일을 생성하여 IUT에게 전송한다. 전송된 REQUEST 파일의 파일명은 "[E0]Bluetooth Device_KAT.req"가 된다. (그림 6)은 기지 출력 검증 요청 과정을 보여준다.

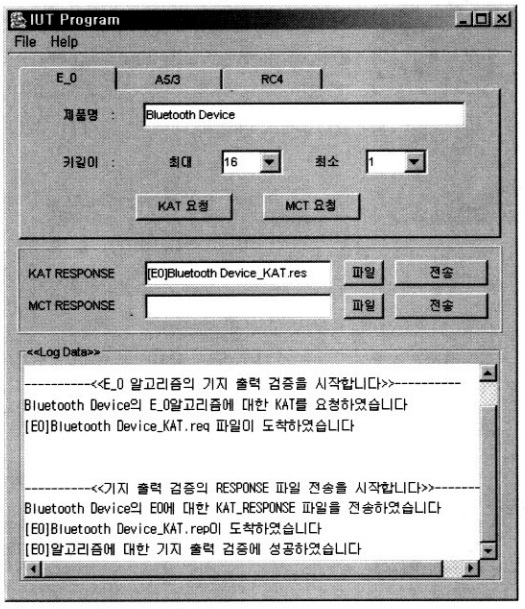
SCVS로부터 REQUEST 파일이 도착하면 RESPONSE 파일을 생성하여 파일명을 "[E0]Bluetooth Device_KAT.res"로 저장한다. 저장된 파일은 "KAT RESPONSE"의 "파일" 버튼 또는 파일의 경로를 입력하는 방식으로 선택하여 "전송" 버튼을 클릭하면 SCVS에게 전송된다. SCVS는 IUT로부터 받은 RESPONSE 파일과 FAX 파일을 비교하여 결과를 "[E0]Bluetooth_Device_KAT.rep"파일을 생성하여 IUT에



(그림 5) E_0 기지 출력 검증용 데이터 파일



(그림 6) 기지 출력 검증 요청

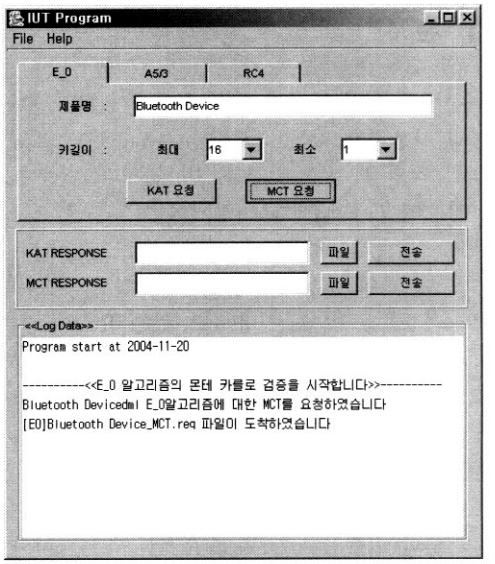


(그림 7) 기지 출력 검증 종료

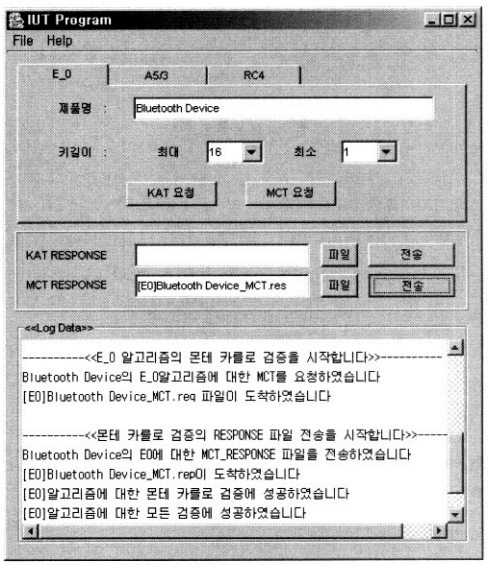
게 전송하고 Bluetooth Device에 대한 기지 출력 검증을 종료한다. (그림 7)은 기지 출력 검증에 대한 결과를 전송하는 과정을 보여준다.

몬테카를로 검증 역시 기지 출력 검증과 동일한 과정으로 진행된다. (그림 8)은 몬테카를로 검증 요청 과정을 보여주며, (그림 9)는 몬테카를로 검증 종료 과정을 보여준다.

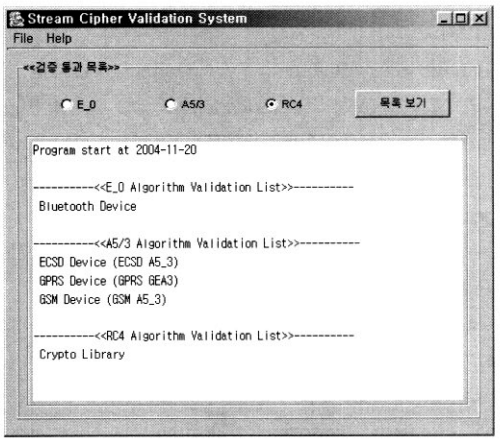
기지 출력 검증과 몬테 카를로 검증에 대한 모든 테스트를 통과한 알고리즘은 SCVS에 제품명이 기록된다. 검증 통과 목록에서 해당 알고리즘을 선택하고 "목록 보기" 버튼을 클릭하면 검증에 통과한 제품에 관한 정보를 볼 수 있다. (그림 10)은 검증을 통과한 제품의 목록을 보여준다.



(그림 8) 몬테카를로 검증 요청



(그림 9) 몬테카를로 검증 종료



(그림 10) 검증 목록 보기

5. 결 론

본 논문에서는 최근 무선 통신 환경에서 다양한 국제표준으로 널리 사용되고 있는 경량(light-weight) 스트림 암호에 대한 구현 적합성 검증 도구를 제안하였다. 블루투스 표준 스트림 암호인 E_0, 제3세대 비동기식 이동통신 표준화 그룹인 3GPP 스트림 암호 A5/3, 그리고 SSL(Secure Socket Layer) 프로토콜에 포함되어 있으며, 다양한 암호 제품에서 사실상의 표준으로 널리 사용되고 있는 RC4에 대한 구현 적합성 검증 방법을 논하였다. E_0, A5/3, RC4는 경량 스트림 암호의 효시가 되는 알고리즘들로 각기 서로 다른 구조적 특성을 지니고 있다. E_0 알고리즘은 전형적인 하드웨어 기반이라 할 수 있는 LFSR 기반의 스트림 암호이고, A5/3는 키스트림 생성기로써 블록 암호 KASUMI의 운영모드를 이용한 스트림 암호이다. 그리고 RC4는 소프트웨어 구현에 적합하도록 설계된 스트림 암호이다. 그러므로 이 세 가지 다른 형식의 스트림 암호에 대한 구현 적합성 검증 방법은 향후 다양한 형태의 스트림 암호에 대한 검증 기법 연구에도 원용될 것이란 점에서 그 의미를 둘 수 있겠다.

참 고 문 헌

[1] Government Computer News, "FIPS testing finds lots of mistakes in crypto IT", 10/29/2002, http://www.gcn.com

[2] NIST, "Cryptographic Standards and Validation Programs", http://csrc.nist.gov/cryptval

[3] FIPS 140-1, "Security Requirements for Cryptogra-phic Modules", January 4, 1994.

[4] FIPS 140-2, "Security Requirements for Cryptogra-phic Modules", May 25, 2001. Change Notices 2, 3, 4: 12/03/2002.

[5] FIPS 197, "Advanced Encryption Standard (AES)", November 26, 2001.

[6] FIPS 46-3, "Data Encryption Standard (DES)", October 25, 1999.

[7] FIPS 185. "Escrowed Encryption Standard (EES)", February 9, 1994.

[8] Bluetooth SIG, "Bluetooth Specification", Version 1.2, 2003.

[9] 3GPP TS 55.216, "Specification of the A5/3 Encryption Algorithms for GSM and ECSD and the GEA3 Encryption Algorithm for GPRS; Document 1: A5/3 and GEA3 Specifications", Version 6.2.0, 2003.

[10] Netscape Communications Corporation, "The SSL Protocol", Version 3.0, Internet Draft, March, 1996.

[11] RC4 page, http://www.wisdom.weizmann.ac.il/ ~itsik/RC4/rc4.html

[12] 3GPP TS 35.202, "Specification of the 3GPP Confidentiality and Integrity Algorithms; Document: A5/3 and GEA3 Specifications", Version 5.0.0, 2002.

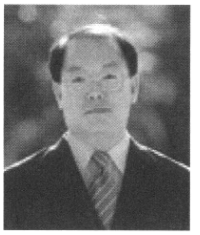
[13] 김역, 정창호, 장윤석, 이상진, 이성재, "SEED 구현 적합성 검증 시스템에 관한 연구", 정보보호학회논문지, 제13권, 제1호, pp.69-85, 2003.

강 주 성

e-mail : jskang@kookmin.ac.kr
1989년 고려대학교 수학과(학사)
1991년 고려대학교 일반대학원 수학과 (이학석사)
1996년 고려대학교 일반대학원 수학과 (이학박사)
1996년~1997년 과학재단 박사후연구원
1997년~2004년 한국전자통신연구원 선임연구원, 팀장
2001년~2002년 벨기에 루벤대학 COSIC 방문연구원
2004년~현재 국민대학교 수학과 조교수
관심분야 : 암호 알고리즘, 정보보호 프로토콜, 응용확률론 등

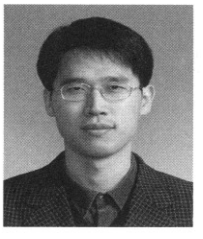
신 현 구

e-mail : kmmeyak@hanmail.net
2003년 국민대학교 수학교육과(학사)
2005년 국민대학교 일반대학원 수학과 (이학석사)
2005년~현재 (주)엑서스 테크놀로지 근무
관심분야 : 암호 이론, 정보보호 이론, 이동통신 정보보호 등



이 옥 연

e-mail : oyyi@kookmin.ac.kr
1988년 고려대학교 수학과(학사)
1990년 고려대학교 일반대학원 수학과 (이학석사)
1996년 University of Kentucky 수학과 (이학박사)
1999년~2001년 한국전자통신연구원 선임연구원, 팀장
2001년~현재 국민대학교 수학과 조교수
관심분야 : 정보보호, 이동통신, 암호론



홍 도 원

e-mail : dwhong@etri.re.kr
1994년 고려대학교 수학과(학사)
1996년 고려대학교 일반대학원 수학과 (이학석사)
2000년 고려대학교 일반대학원 수학과 (이학박사)
2000년~현재 한국전자통신연구원 선임연구원, 팀장
관심분야 : 암호 이론, 정보보호 이론, 이동통신 정보보호 등