논문 99-02-20

CDMA 디지털 이동통신 시스템의 인증 알고리즘 구현

Implementation of Authentication Algorithm for CDMA Digital Mobile Communication System

金 範 植*, 申 仁 澈*

(Bum-Sik Kim* and In-Chul Shin*)

요 약

이동통신에서의 사용자 인증 서비스는 통화도용 방지와 신뢰성 있는 과금을 위한 중요한 보호서비스이다. 최근 몇 년 동안 IS-41 북미 이동 전화 시스템에서 사용되어지는 몇몇 암호 시스템이 공격을 당하였다. 이러한 알고리즘은 ORYX, CMEA 그리고 CAVE 등이다. 이러한 알고리즘들을 대체하기 위한 작업이 이미 진행 중이다.

본 논문에서는 해쉬함수를 개발하고 이를 IS-95A 시스템의 인증 알고리즘에 적용하였다. 그리고 통계적 분석 기법을 사용하여 개발된 알고리즘의 출력 특성을 분석하였으며 C 프로그래밍 언어를 사용하여 알고리즘의 유효성을 시뮬레이션하였다.

Abstract

The user authentication service can be used to prevent telecommunications piracy and to demand reliable payment from subscriber. Over the last few years several of the cryptographic systems being used by the IS-41 North American Mobile telephones have been broken. These algorithms included ORYX, CMEA and CAVE.

The process of replacing these algorithms is already underway. In this paper we designed a hash function and applied it to the authentication algorithm of IS-95A authentication system. We also analyzed the randomness properties of designed algorithm using statistical analysis and simulated the validity of this algorithm using C programming language.

Keyword : Authentication, CDMA, Encryption, hash function, Mobile Communication,

I. 서 론

이동통신 기술의 발달로 인하여 이동통신서비스 가입자 수의 증가와 함께 전파를 통신매체로 이용하는 이동 통신의 특성으로 인한 불법적인 서비스 이용이나 도청 또는 추적을 통한 불법적인 행위와 같은 각종 통신 범죄행위 등도 증가하고 있다. 특히 통화

* 檀國大學校 電子・컴퓨터工學科
(Dept. of Electronics and Computer Eng., Dankook Univ.)

接受日: 1999年7月26日, 修正完了日: 1999年11月8日

도용은 이동통신서비스 사업자에게는 요금징수와 관련한 피해를 주며, 가입자에게는 요금체계에 대한 불신감을 주어 이동 통신 발달에 큰 장애요인이 되고 있다. 따라서 이러한 문제점을 방지하기 위한 보호 서비스가 필요한데 이를 신분인증 서비스라 한다[1].

이동 통신에서의 인증 서비스는 이동국과 기지국 간의 신분을 확인하기 위하여 서로 관련 정보를 교환하는 것으로 서로 공유한 비밀 데이터가 일치하는지를 확인하는 과정을 의미한다.

디지털 이동통신 시스템의 대표적인 표준안으로는 IS-95A[2], GSM(Global System for Mobile communication)[3] 그리고 PACS(PHP merged with WACS)[4]등이 있다. IS-95A는 미국을 중심으로 북미에서 서비스를 제공하고 있으며 1996년 서비스를 제공한 국내 CDMA (Code Division Multiple Access) 방식 디지털 셀룰러 이동통신의 표준안이다.

지난 몇 년 동안 IS-41 북미 이동전화 시스템에서 사용되어진 몇몇 암호 시스템들이 공격당하였는데 이는 사용자 데이터 보호용 스트림 암호 시스템인 ORYX[5]와 제어 채널 보호용 블록 암호 시스템인 CMEA[6], 그리고 사용자 인증을 위한 CAVE[7]알고리즘이다. 그리고 이러한 알고리즘들을 대치할 새로운 알고리즘의 개발 작업이 추진 중이다[8]. 따라서 새로운 인증 알고리즘의 개발이 시급하다 하겠다.

본 논문에서는 인증 알고리즘이 연산 처리 능력이 제한되어 있는 단말기에서 이상적인 시간 내에 처리되어져야 하기 때문에 알고리즘의 수행 속도가 문제가 됨으로 해쉬함수를 개발하여 IS-95A 인증 시스템에 적용함으로서 새로운 인증 알고리즘을 개발하였으며 통계적 해석 기법을 사용하여 개발된 알고리즘의 출력에 대한 통계 테스트를 수행하였다.

II. IS-95A에서의 인증방식

CDMA 이동 통신에서의 인증은 이동국의 정체성을 확인하기 위해 이루어지는 이동국과 기지국간의 일련의 정보 교환 과정이며 동일한 공유 비밀 데이터(SSD : Shared secrete Data)를 공유함으로서 인증이 이루어진다. 비밀 공유 데이터를 생성하기 위한 절차를 Auth_Signature Procedure라 하며 152비트 데이터를 입력으로 하여 18비트의 출력인 비밀 공유 데이터를 생성한다.

2.1 인증방식

IS-95A에서 제안된 인증절차의 종류는 다음과 같다[2].

- 등록 인증 절차
  이동국의 위치 등록을 위해 수행되는 절차
- 유일시도응답 절차
  상기과정 실패 시 수행됨
- 발호인증 절차
  이동국이 호를 시작하고자 할 때 수행되는 절차
- 착호인증 절차
  등록되어진 이동국이 기지국의 호출에 응하고자 할 때 수행되는 절차
- 기지국거부응답(SSD 생성 절차) 절차
  모든 인증과정이 실패하는 경우 수행되는 절차로서 SSD값을 갱신한다.

2.2 인증 관련 파라미터

① MIN (Mobile Identification Number)

34비트의 이동국의 전화번호에 해당되는 2진수값으로 10디지트의 전화번호로부터 유도되며 24비트의 MIN1과 10비트의 MIN2로 구성된다.

② ESN (Electronic Serial Number)

32비트의 2진수로서 이동국의 유일성을 확인하기 위한 값이다. 이는 제조업자에 의해서 지정되는 고유번호로서, 영구메모리 영역 내에 저장되기 때문에 변경이 불가능한 값이다. 이 값은 이동국과 지정된 인증국(HLR/AC : Home Location Register/Authentication Center)에서만 알 수 있다. 8비트의 MFR 코드, 6비트의 reserved 그리고 18비트의 serial number로 구성된다.

③ SSD (Shared Secrete Data)

이동국내에 저장된 128비트 값으로 기지국에서도 이용 가능한 정보이다. 이동국에서는 SSD를 반영구 메모리에 저장함으로서 접근이 불가능하다. 그러나 인증이 계속 실패되는 경우, 기지국은 이동국의 SSD를 수정할 수 있다. 인증용으로 이용되는 SSD_A와 CDMA 음성 정보 보호용으로 이용되는 SSD_B로 구성되며 각각은 64비트이다.

④ RAND

32비트의 난수 데이터인 이 값은 기지국에서 생성하여 이동국이 수신한다.

이외에도 이동국에서 유지되는 64진 계수기로서, 호출 채널 내의 parameter update order 메시지 수신시 갱신되는 COUNT, 이동국에 할당하는 64비트의 A Key와 HLR/AC에서 생성되며, SSD 갱신과정에서 사용되는 56비트의 난수 데이터인 RANDSSD 등이 있다 [2].

그림 1은 등록 인증 절차의 입력 파라미터와 출력 생성 절차를 나타내었다. 유일시도응답 절차, 발호인증 절차, 착호인증 절차, 기지국거부응답절차는 모두 동일한 알고리즘에 의해 비밀 공유 데이터를 생성하며 입력 파라미터만이 다르다. 그림 2는 모든 인증이 실패할 경우에 SSD값을 갱신하기 위한 절차이다. 본 논문에서는 그림 1의 Auth_Signature 절차와 SSD Generation 갱신 절차에 필요한 알고리즘을 개발하였다.

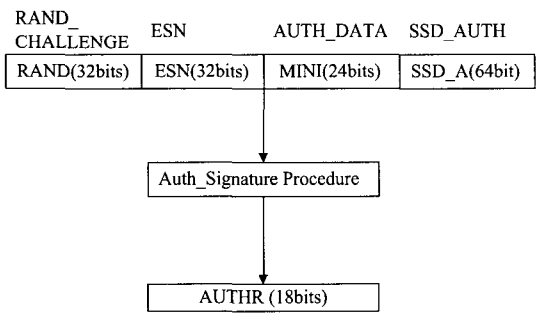

그림 1. AUTHR 계산과정 (등록절차)
Fig. 1. Computation of AUTHR (registration procedure).

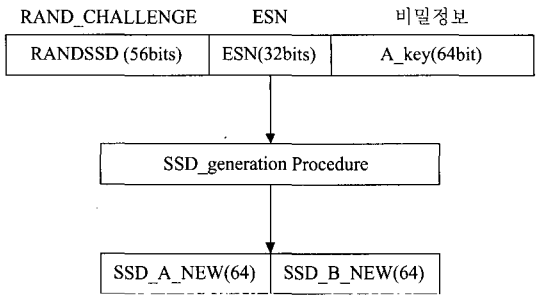

그림 2. 새로운 SSD 생성
Fig. 2. Computation of new SSD.

III. 해쉬 함수

해쉬함수(hash function), 특히 암호학적 해쉬함수는 임의의 유한 길이 비트열을 고정된 길이의 비트열로 맵핑시키는 함수이다. 이 고정된 길이의 출력 값을 해쉬값(hash value), 메시지 다이제스트(message digest)라 한다. 해쉬함수 h와 입력 x가 주어졌을 때, $h(x)$를 계산하는 것이 용이해야 한다. 즉, 암호학적 일방향 해쉬함수는 다음 성질을 만족해야 한다[9][10].

- 원상 저항 (preimage resistance)

해쉬함수 h와 해쉬값 y가 주어졌을 때, $h(x)=y$을 만족하는 입력 x을 찾는 것이 계산상 실행 불가능하여야 한다.

- 2nd-원상 저항 (second preimage resistance)

해쉬함수 h와 입력 x에 대응하는 해쉬값 y가 주어졌을 때 $h(x)=h(x')$을 만족하는 입력 x'을 찾는 것이 계산상 실행 불가능해야 한다.

- 충돌 저항 (collision resistance)

해쉬함수 h가 주어졌을 때 $h(x)=h(x')$을 만족하는 입력 x와 x'을 찾는 것이 계산상 실행 불가능해야 한다.

거의 모든 해쉬함수의 처리 과정은 입력을 연속적인 고정된 블록들로 나누어 처리함으로서 임의 길이 입력을 해쉬하는 반복적인 처리과정이다. 먼저 입력 X는 블록 길이의 배수가 되도록 패딩(padding)되

고 t 개의 블록 X_1에서 X_t로 나누어진다. 해쉬함수 h는 다음과 같이 기술된다.

$$H_0 = IV$$

$$H_i = f(h_{i-1}, X_i), \quad 1 \le i \le t$$

$$h(X) = H_t$$

여기서 f는 압축함수 (compress function)이며, H_i는 단계 $i-1$과 단계 i 사이의 연쇄 변수 (chaining variable)이고, IV는 초기값 (initial value)이다.

압축 함수를 사용한 반복적인 해쉬함수의 블록도는 그림 3과 같다.

해쉬값의 계산은 연쇄 변수에 의존한다. 해쉬 계산을 시작할 때, 이 연쇄 변수는 알고리즘의 일부로 명시된 고정된 초기값을 가진다. 압축 함수는 해쉬 되어질 메시지 블록을 입력으로 받아 이 연쇄 변수의 값을 갱신한다. 이 과정이 모든 메시지 블록에 대해 순환적으로 반복되고, 연쇄 변수의 마지막 값이 그 메시지에 대한 해쉬값으로 출력된다.

해쉬함수는 내부 압축 함수로 어떤 구조를 사용하느냐에 따라 다음 3가지로 분류된다[9].

① 블록암호(block ciphers)에 기반한 해쉬함수
② 모듈러 연산(modular arithmetic)에 기반한 해쉬함수
③ 전용해쉬함수(dedicated hash functions)

전용해쉬함수는 빠른 처리 속도를 가지고 다른 시스템 서브 요소에 무관하도록 특별히 설계된 함수이다. 현재까지 제안된 전용해쉬함수는 대부분 1990년에 Rivest에 의해 설계된 MD4[11]에 기반한 구조를 가진다. 현재 널리 사용되는 MD계열 해쉬함수로는 MD5[12], SHA-1[13], RIPEMD-160[14], HAVAL[15]등이 있다. MD4는 Dobbertin[16]의 공격에 의해 완전히 해독되었고 MD5 역시 심각한 취약점이 발견되었다[17]. 현재까지 안전하다고 생각되는 함수로 SHA-1, RIPEMD-160, HAVAL 등이 있다.

특정 해쉬함수가 주어지면, 안전한 해쉬함수의 입증을 위해 해쉬함수를 공격하는 복잡도에 관한 하한을 증명할 수 있는 것이 바람직하지만 실제 그러한 방법은 거의 알려져 있지 않고 대부분의 경우에 적용 가능한 알려진 공격의 복잡도가 해쉬함수의 안전성으로 고려되어진다.

해쉬값이 균등한 확률 변수라고 가정하면, 다음은 잘 알려진 사실이다[9].

- n비트 해쉬함수 h에 대해, 2^n 연산으로 원상(preimage)와 2nd-원상(second preimage)를 발견하기 위한 추측 공격(guessing attack)을 기대할 수 있다.
- 메시지를 선택할 수 있는 공격자에 대해, 생일 공격(birthday attack)은 약 $2^{n/2}$ 연산으로 $hash(M) = hash(M')$인 충돌 메시지 쌍 M, M'을 발견할 수 있다.

n 비트 해쉬함수가 다음 두 성질을 만족한다면,

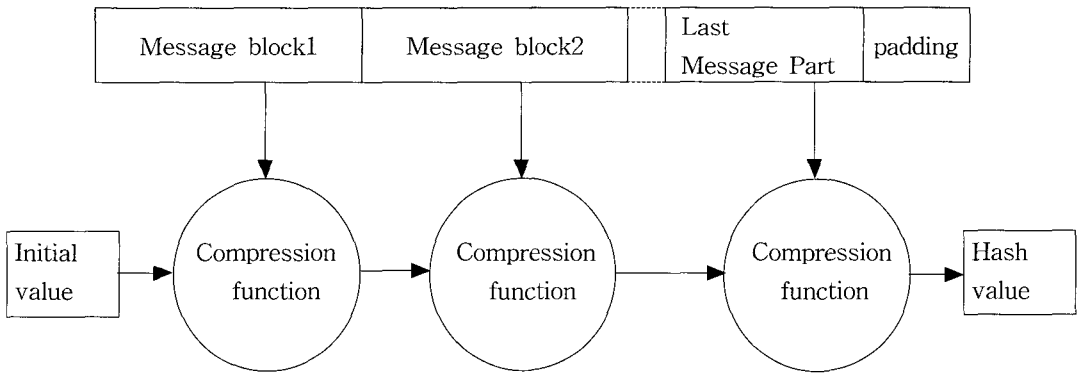

그림 3. 반복적인 해쉬함수의 일반적인 구조
Fig. 3. The use of a compression function in an iterative hash function.

이상적인 안전성을 가진다.

(1) 해쉬값이 주어지면, 원상(preimage)와 2nd-원상(second preimage)발견은 약 2^n 연산을 요구한다.

(2) 충돌발견은 약 $2^{n/2}$ 연산을 요구한다.

IV. 새로운 해쉬함수 알고리즘

본 논문에서 제안한 알고리즘은 32비트 프로세서에서 최적의 성능을 갖도록 설계되었다. 모든 연산은 32비트 단위로 이루어지며 역워드저장방식(little-endian storage scheme)/2의 보수 표현을 사용한다. 6개의 32비트 레지스터 a, b, c, d, e, f는 연쇄변수로서 최종 해쉬값을 갖는다. 이 레지스터들은 h_0 값으로 초기화되는데 그 값은 다음과 같다.

a = 0x01234567
b = 0x89ABCDEF
c = 0x56789ABC
d = 0xCBA98765
e = 0xFEDCBA98
f = 0x76543210

256 비트 메시지 블록은 32비트 단위로 나뉘어져 x0, x1, x2 ... x7 레지스터의 초기값으로 사용되어지고 이어지는 연산에 의해 h_i는 h_{i+1}로 갱신된다. 본 알고리즘은 3번의 패스로 이루어지며 그 사이에 키 스케쥴링이 이루어진다. 최종적으로 feedfowrd 단계에서는 레지스터 a, b, c, d, e, f의 현재값과 그들의 초기값에 대한 연산에 의해 최종 해쉬값 h_{i+1}을 만들어 낸다. 알고리즘을 설명하기 위한 식들은 C 프로그래밍 언어의 표현방식을 사용하였다.

4.1 알고리즘의 구조

```
save_abcdef
pass(a,b,c,d,e,f, 5)
key_schedule
pass(a,b,c,d,e,f, 7)
key_schedule
pass(a,b,c,d,e,f, 9)
feedforward
```

(1) save_abcdef는 feedforward 단계에서 사용할 초기 h_i 값을 저장한다.

```
aa = a;
bb = b;
cc = c;
dd = d;
ee = e;
ff = f;
```

(2) pass(a, b, c, d, e, f, mul)은 다음과 같이 구성된다.

```
round(a, b, c, d, e, f, x0, mul);
round(b, c, a, e, f, d, x1, mul);
round(c, a, b, f, d, e, x2, mul);
round(a, b, c, d, e, f, x3, mul);
round(b, c, a, e, f, d, x4, mul);
round(c, a, b, f, d, e, x5, mul);
round(a, b, c, d, e, f, x6, mul);
round(b, c, a, e, f, d, x7, mul);
```

round(a, b, c, d, e, f, mul)은 다음과 같다.

```
c ^= x;
a -= s1[c_0] ^ s2[c_1] ^ s3[c_2] ^ s4[c_3];
b += s1[c_3] ^ s2[c_2] ^ s3[c_1] ^ s4[c_0];
b *= mul;
f ^= x;
d -= s5[c_0] ^ s6[c_1] ^ s7[c_2] ^ s8[c_3];
e += s5[c_3] ^ s6[c_2] ^ s7[c_1] ^ s8[c_0];
e *= mul;
```

여기서 c_i는 c레지스터의 i 번째 ($0 \leq i \leq 3$) 바이트이며, s1, s2, s3, s4, s5, s6, s7, s8은 256개의 4바이트 값을 갖는 8개의 룩업(look-up) 테이블이다.

(3) key_schedule은

```
x0 -= x7 ^ 0xA5A5A5A5;
x1 ^= x0;
x2 += x1;
x3 -= x2 ^ ((~x1) << 7);
x4 ^= x3;
x5 += x4;
x6 -= x5 ^ ((~x4) >> 23);
x7 ^= x6
x0 += x7;
x1 -= x0 ^ ((~x7) << 7);
x2 ^= x1;
x3 += x2;
x4 -= x3 ^ ((~x2) >> 23);
x5 ^= x4;
x6 += x5;
x7 ^= x6 ^ 0x01234567;
```

다음과 같다. 여기서 >>, <<은 산술 쉬프트가 아닌 좌, 우 논리적 쉬프트 연산자이다.

(4) feedforward 은

```
a ^= aa;
b -= bb;
c += cc;
d ^= dd;
e -= ee;
f += ff;
```

이다. 여기서 a, b, c, d, e, f레지스터는 192비트의 중간 해쉬 값인 h_i+1이며 알고리즘의 종료 후 최종 해쉬값이 된다.

4.2 S-box의 설계

본 알고리즘에서 설계된 S-box는 제안된 해쉬 알고리즘을 사용하여 생성한다. S-box 테이블을 구성하는 값들이 랜덤성을 띠도록 구성함으로서 압축함수의 입력인 비밀키 값을 모른다면 동일한 S-box 구성할 수 없도록 하였다. 이는 S-box를 구성하여 만들어진 암호 시스템의 가장 강력한 공격법으로 알려진 차분 공격법(DC:Diffrential Cryptanalysis)에 대응할 수 있을 것이다. S-box 테이블은 256개의 32비트 값을 갖는 8개의 테이블로 구성되며 구성방법은 다음과 같다.

(1) 테이블 초기화

테이블의 초기화는 8개의 테이블에 대해 동일하며 바이트 단위로 이루어진다.

```
for(i=0; i<2048; i++)
  for(col=0; col<4; col++)
    S_box[i][col] = i&255;
```

(2) 제안한 해쉬 알고리즘의 수행에 의한 S_box의 구성값들의 치환

알고리즘 수행 후의 해쉬값들을 바이트 단위로 사용하여 S_box 내의 값들이 치환되어질 테이블 내의 위치로 사용한다. 테이블을 구성하는 값들은 바이트 단위로 치환되어지며 해쉬값을 모를 경우 테이블 내의 치환되어질 위치를 모르므로 테이블을 구성할 수 없다. 치환을 통해 구성된 테이블 값들은 서로 동일한 값을 갖지 않도록 구성하였으며 또한 8개의 테이블 각각은 서로 다르게 구성된다.

4.3 제안한 해쉬함수의 인증 알고리즘에의 적용

인증 알고리즘에 적용하기 위해 제안된 해쉬함수는 256비트의 입력 데이터를 사용 하지만 IS-95A에서의 인증은 152비트의 입력을 사용함으로 104비트의 비트 채워넣기(padding)가 필요하다. 따라서 152비트의 입력을 다음과 같이 변형하여 나머지 104비트를 만들

어 낸다. Pad_bit[0], Pad_bit[1], Pad_bit[2]는 각각 32비트이고 Pad_bit는 알고리즘의 버전을 나타내며 크기는 8비트이다.

```
Pad_bit[0] = RAND_CHALLENGE ^ ESN;
Pad_bit[1] = ESN ^ SSD_AUTH;
Pad_bit[2] = SSD_A ^ SSD_B;
Pad_bit = 0x5A
```

해쉬함수의 출력은 192 비트이지만 인증 알고리즘의 출력은 18비트여야 함으로 a, b, c, d, e, f 레지스터의 상위 3비트를 선택하여 최종 출력인 18비트의 AUTHR값을 만들어 낸다. 또한 SSD_generation procedure에 의해 생성되는 새로운 SSD값은 128비트의 출력을 가짐으로 a, b, c, d, e, f 레지스터 중 4개의 a, b, e, f 값이 새로운 SSD값이 된다.

V. 알고리즘의 통계적 특성 및 결과

5.1 빈도 테스트 (Frequency test)

키 스트림을 테스트하는 방법은, 먼저 키 스트림으로부터 충분한 량의 이진 수열 (S_t)를 얻어 0의 개수를 n_0, 1의 개수를 n_1이라 한다. 이상적인 랜덤 수열이라면 어떤 한 비트가 0이 될 확률은 1/2이므로 키 스트림 내에서의 0의 기대치는 $\frac{n}{2}$이고 1의 기대치도 $\frac{n}{2}$이다. 그러므로 0의 측정치 n_0와 1의 측정치 n_1으로부터 다음과 같은 검정 통계량을 갖는 x^2 (chi-square)분포를 생각할 수 있다.

$$x^2 = \frac{(n_0 - n_1)^2}{n} \tag{1}$$

x^2(chi-square) 분포로부터 유의수준 5%의 값은 3.84이므로 통계량 x^2의 값이 3.84보다 큰 키 스트림은 빈도 테스트에 대해 랜덤성이 없다고 판단되어 기각한다.

5.2 시리얼 테스트 (Serial Test)

이진수열 (S_t)에서 한 비트가 다음 비트로 가는 천이 확률을 테스트하는 것으로서 한 개의 비트 즉, '0' 비트나 '1'비트가 주어졌을 때 그 다음 비트가 '0'인 경우와 '1'인 경우를 테스트한다. 테스트하는 방법은 우선 이진 수열 (S_t)에서 테스트할 데이터를 얻은 다음, 이 수열을 연속된 2비트로 분리하여, '00'의 개수를 n_{00}, '01'의 개수를 n_{01}, '10'의 개수를 n_{10}, '11'의 개수를 n_{11}이라 하고 전체 N 비트 중 '0'의 개수를 n_0, '1'의 개수를 n_1이라 하자. 그러면 다음과 같은 식을 얻는다.

$$\begin{aligned} &n_{00} + n_{01} = n_0 \text{ 혹은 } n_0 - 1 \\ &n_{10} + n_{11} = n_1 \text{ 혹은 } n_1 - 1 \\ &n_{00} + n_{01} + n_{10} + n_{11} = N - 1 \\ &n_0 + n_1 = N \end{aligned} \tag{2}$$

n_{ij}의 기대치는 $(N-1)/4$ 이고 다음의 통계량은 근사적으로 자유도가 2인 x^2 분포를 따른다.

$$T = \sum_{i,j=0}^{1} \frac{n_{ij} - (N-1)/4}{(N-1)/4} - \sum_{i=0}^{1} \frac{(n_i - N/2)^2}{N/2} \tag{3}$$

이것을 다른 식으로 표현하면 다음과 같다.

$$\begin{aligned} T = &\frac{4}{N-1}(n^2_{00} + n^2_{01} + n^2_{10} + n^2_{11}) \\ &- \frac{2}{N}(n^2_0 + n^2_1) + 1 \end{aligned} \tag{4}$$

위의 T는 자유도 2인 x^2의 분포를 뜻하며, x^2 분포에서의 유의수준 5%, 자유도 2의 값은 5.99이므

로 검정 통계량의 값이 5.99보다 큰 키 스트림은 시리얼 테스트에 대해 랜덤성을 갖지 못하므로 기각한다.

5.3 포카 테스트 (Porker Test)

8비트 ASCII 코드의 종류는 2^8가지인데, 길이 n인 키 스트림을 취하면, $F=n/8$ 개의 8비트 ASCII 문자가 생긴다. F개의 8비트 이진코드 중에서 00000000의 개수를 f_0, 00000001의 개수를 f_1, ... , 11111111의 개수를 f_{255}라 하자. 만일 키 스트림이 균일하게 분포되어 있다면, 256가지의 8비트 ASCII 코드 문자는 각각 동일한 1/256이 되어 그에 따른 기대치는 $F/256$이 된다. 이러한 f_i에 대해 다음의 검정 통계량을 갖는 x^2을 생각할 수 있다.

$$T=\frac{256}{F}\sum_{i=0}^{255}(f_i-F/256)^2 \tag{5}$$

위의 T는 자유도 255인 x^2 분포를 하며, x^2 분포에서 유의수준 5%, 자유도 255의 값은 292.84이므로 이 값보다 큰 키 스트림은 포커 테스트에 대해 랜덤성을 갖지 못하므로 기각한다.

5.4 런 테스트 (Run Test)

이진 수열 (S_t)내에 0이나 1이 연속해서 나오는 것을 말하며, 알고리즘 자체의 어떤 취약성에 의해 키 스트림이 이러한 특성을 갖게 되는지를 테스트하는데 필요하다. 이것을 테스트하는 방법은 다음과 같다.

출력 수열을 '0'의 나열인 갭(gab)과 '1'의 나열인 블럭(block)으로 나눈다.. r_{0i}는 길이 i인 갭의 개수이고 r_{1i}는 길이 i인 블럭의 개수라면, 갭의 전체의 개수와 블럭의 전체의 개수는 다음과 같이 주어진다.

$$r_0=\sum_{i=1}^{17} r_{0i} \ (\ r_0 : \text{전체의 gab 개수}) \tag{6}$$

$$r_1=\sum_{i=1}^{17} r_{1i} \ (\ r_1 : \text{전체의 block 개수}) \tag{7}$$

이 때, 키 스트림의 수열이 충분히 크고 빈도 테스트를 통과하였다면, '0'의 런 수 와 '1'의 런 수는 거의 같으므로 두 개의 런을 구별할 필요는 없다.

$n_i=n_{0i}+n_{1i}$, $n=r_0+r_1$이라 하면, 어떤 런이 길이 i가 될 이상적인 확률은 $1/2^i$ 이므로 기대치는 $n/2^i$가 된다. 그러므로 실제 측정치인 n_i와 추정치인 $n/2^i$ 사이의 x^2 테스트를 생각할 수 있다.

$$x^2=1/n\sum_{i=1}^{17}2^i(n_i-n/2^i)^2 \tag{8}$$

x^2 테스트로부터 자유도 16인 x^2 분포의 5% 유의 수준에 의한 값은 26.300이므로 x^2의 값이 이것보다 크면 런 테스트에 대하여 랜덤성을 갖지 못하므로 기각한다.

5.5 자기상관 테스트 (Autocorrelation test)

이진 수열 (S_t)와 이로부터 양의 정수 n만큼 천이시킨 수열 (S_{t+n})와의 상관관계를 조사하는 것이다. 임의의 n개의 비트를 $a_1a_2a_3\ \ldots\ a_n$이라 하면, 각 자기상관 값은 다음과 같이 주어진다.

$$A(d)=\sum_{i=1}^{n-d}a_ia_{i+d} \quad (0\le d< n-1) \tag{9}$$

여기에서

$$A(0)=\sum_{i=1}^{n-d}(a_i)^2=n_1 \tag{10}$$

즉, 1의 개수가 됨을 알 수 있다.

만일 키 스트림으로부터 n개의 수열이 n_0개의

'0'을 갖고 1개의 '1'을 가지고 있으며 균일하게 분포되어 있다면, 수열의 자기상관성의 이상적인 값은 다음과 같다.

$$u(d)=\frac{n^2{}_i(n-d)}{n^2} \quad (d=0) \tag{11}$$

이때, 측정치 $A(d)$와 기대치 $u(d)$ 사이의 차이를 $x(d)=A(d)-u(d), \quad (1\le d< n-1)$라 하고, $x(1),x(2), \ldots ,x(n-1)$은 정규분포를 이룬다고 하며, 전체 키 스트림의 $x(1),x(2), \ldots ,x(n-1)$의 모평균 u는 0이라고 하자.

각 $x(d)$ 값들의 표준 평균을 X, 표본 분산을 S라 하면, t 검정에서의 자유도 $n-2$인 검정 통계량은 다음과 같다.

$$t=\frac{X-u}{S/\sqrt{(n-1)}} \tag{12}$$

그러나 자유도가 큰 t 분포 (>> 30)는 표준정규분포를 이루게 된다. 이를 이용하면 유의 수준 5%로 하는 한계치는 1.96 이므로 이 값보다 크면 자기상관성이 랜덤성을 가지지 못하는 것으로 판단하여 기각한다.

아래 표 1에는 테스트별 임계값을 나타내었으며 표 2에는 제안된 인증 알고리즘의 테스트 결과이다. 결과에서 알 수 있듯이 제안된 알고리즘의 출력에 대한 통계적 특성이 모두 만족됨을 알 수 있다.

VI. 결 론

본 논문에서는 IS-95A 인증 시스템의 인증 및 암호화를 위한 세션키 생성 알고리즘을 개발하였다. 이 알고리즘은 연산 처리 능력이 제한되어 있는 단말기에서 이상적인 시간 내에 처리되어져야 하기 때문에 알고리즘의 수행 속도가 문제가 된다. 따라서 해쉬함수를 개발하고 이를 이용하여 인증 및 암호화를 위한 세션키 생성 알고리즘을 개발하였으며 알고리즘은 펜티엄II 333Mhz에서 C 언어로 구현하였다. 그리고 통계적 분석 방법을 사용하여 개발된 알고리즘의 출력 특성을 분석하였다. 출력 특성 분석에 사용된 검사로는 빈도 테스트(Frequency test), 시리얼 테스트(Serial test), 포커 테스트(Porker test), 런 테스트(Run test) 그리고 자기상관 테스트(autocorrelation test)등이다. 통계적 분석을 통하여 제안된 알고리즘의 출력 특성이 양호함을 알 수 있었다.

표 1. 각 Test 임계값
Table 1. The Critical value of each tests.

Test	Freq. Test	Serial Test	Poker Test	Autocorr. Test	Run Test
임계값	3.480	5.990	292.840	1.960	26.30

표 2. 통계 테스트 결과
Talbe 2. The results of each tests.

Test	Freq. Test	Serial Test	Poker Test	Autocorr. Test	Run Test
임계값	0.217	3.297	233.514	1.020	22.03

본 논문에서 제안한 해쉬함수는 입력데이터의 형태와 출력 데이터의 길이를 변화시킴으로서 디지털 서명, 디지털 워터마킹, GSM 인증 시스템 등에 적용되어질 수 있을 것이다.

참 고 문 헌

[1] ISO/IEC 7498-2, *Information Processing-OSI Basic Reference Model - Part2 : Security Architecture*, 1989

[2] TIA/EIA IS-95A, *Mobile-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System*, July 1993

[3] ETSI, *European Digital Cellular Telecommunication System(phase 2)-Security Related Network Functions*, July 1993

[4] JTC, *Personal Communications Services PACS Air*

Interface Specification, Jan. 1995

[5] D. Wagner, L. Simpson, E. Dawson, J. Kelsey, W. Millan, and B. Shneier, "Cryptanalysis of ORYX", In workshop on Selected Areas in Cryptology 1998, *Proceedings, To appear in Springer-Verlag Lecture Notes in Computer Science*, pp. 403-417, 1998

[6] D. Wagner, B. Shneier, and J. Kelsey, "Cryptanalysis of the Cellular Message Encryption Algorithm", *In Advances in Cryptology - Crypto'97 Proceedings*, Lecture Notes in Computer Science 1294, pp. 526-537, Springer-Verlag, 1997

[7] W. Millan, "Cryptanalysis of the Alleged Cave Algorithm", *in the 1st International Conference on Information Security and Cryptology, Proceedings*, pp. 107-119, 1998

[8] Telecommunications Industry Association, New security algorithms : Call for development, 1998, can be found at http://scitech.ctia.org/Security/index.html

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997

[10] B. Preneel, "Analysis and design of cryptographic hash functions", *Doctoral Dissertation*, Katholieke Universiteit Leuven, 1993

[11] R. L. Rivest, "The MD4 message-digest algorithm", *Advances in Cryptology-Crypto'90*, Lecture Notes in Computer Science, vol. 537, pp303-311, 1991

[12] R. L. Rivest, "The MD5 message-digest algorithm," Request For Comment(RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April 1992

[13] FIPS 180-1, "Secure hash standard", *Federal Information Processing Standards Publication 180-1*, U.S. Department of Commerce / Nist, 1995

[14] H. Dobbertin, A. Bosselaers and B. Prennel, "RIPEMD-160 : A strengthened version of RIPEMD", *Fast Software Encryption-Cambridge Workshop*, Lecture Notes in Computer Science, vol. 1039, Springer-Verlag, pp. 71-82, 1996

[15] Y. Zheng, J. Pieprzyk, and J. Sebery, "Haval - a one-way hashing algorithm with variable length of output", *Advances in Cryptology-AUSCRYPT'92*, LNCS, vol. 718, pp. 83-104, 1993

[16] H. Dobbertin, "Cryptanalysis of MD4", *Fast Software Encryption-Cambridge Workshop*, Lecture Notes in Computer Science, vol. 1039, Springer-Verlag, pp.53-69, 1996

[17] H. Dobbertin, *The status of MD5 after recent attack*, CryptoBytes, 2(2), pp.1-6, Sep. 1996

저 자 소 개

金 範 植 (會員申請中)

1993년 단국대학교 전자공학과 졸업 (공학사), 1995년 단국대학교 대학원 전자공학과 졸업(공학석사), 1996년～1999년 2월 단국대학교 대학원 전자공학과 박사과정 수료.

주관심 분야 : 정보보호, 이동통신.

申 仁 澈 (正會員)

본 학회 논문지 제 2권, 제 2호, Dec., 1998 참조.

1973년 고려대학교 전자공학과 졸업(공학사).

1978년 고려대학교 대학원 전자공학과 졸업(공학석사).

1986년 고려대학교 대학원 전자공학과 졸업(공학박사).

1984년～1985년 미국 미시간 주립대학교 교환 교수.

1990년 9월～1991년 8 단국대학교 공과대학 부학장.

1979년～현재 단국대학교 전자·컴퓨터공학과 교수.

주관심 분야 : 병렬처리, 퍼지 제어, 암호이론.