IS-95A 인증 알고리듬의 개발†

*이국희, **이상곤° , ***정원영, ***김태근, *문상재
* 경북대학교 전자전기공학부
**동서대학교 전자기계공학부
*** 한국통신 무선연구소

Development of Authentication Algorithm for IS-95A Standard

*Kook-Heui Lee, **Sang-Gon Lee° , ***Won-Young Jeong,
***Tae-Guen Kim, *Sang-Jae Moon
* School of Electronic and Electrical Engr., Kyungpook National Univ.
** Division of Electronic and Mechanical Engr., Dongseo Univ.

요 약

이동통신에서의 사용자 인증 서비스는 통화도용 방지와 신뢰성 있는 과금을 위한 중요한 보호 서비스이다. 본 논문에서는 IS-95A 인증 시스템에 적용 가능한 안전한 인증 알고리듬과 인증 키 생성 알고리듬을 제안한다. 특히 인증 알고리듬을 활용하여 인증 키 생성알고리듬을 oracle 해쉬함수의 형태로 구현함으로써 인증 시스템의 높은 안전성과 간결성을 동시에 성취하였다. 그리고 통계적 분석 기법을 사용하여 개발된 알고리듬의 출력 특성을 분석한다.

I .서 론

이동통신서비스 이용자 수의 증가와 더불어 전파를 통신매체로 이용하는 이동통신의 특성 때문에 불법적인 사용이나 도청 또는 추적을 통한 불법적인 행위와 같은 각종 통신 범죄행위 등도 증가하고 있다. 특히 통화도용은 이동통신서비스 사업자에게는 요금징수와 관련한 피해를 주며, 가입자에게는 요금체계에 대한 불신감을 주어 이동통신 발달에 큰 장해요인이 된다. 이러한 통화도용을 방지하기 위하여 인증 서비스가 필요하다[1].

이동통신에서의[1] 인증 서비스란 기지국과 이동국간의 합법적인 통신을 위하여 서로 공유한 비밀 데이터가 일치하는지를 확인하는 것을 의미한다. IS-95A[2], GSM[3] 그리고 PACS[4] 등과 같은 이동통신 표준안은 기본적으로 인증 서비스를 제공하고 있다.

† 본 논문은 한국통신의 98년도 정보통신기초연구사업 결과의 일부분임.

본 논문에서는 국내 개인휴대통신의 표준안인 IS-95A의 인증 시스템에 적용 가능한 인증 알고리듬과 인증 키 생성 알고리듬을 제안한다. 인증 키 생성 알고리듬은 인증 알고리듬을 활용하여 oracle 해쉬함수[5, 6]의 형태로 구성함으로써 시스템의 높은 안전성과 간결성을 동시에 성취하였다. 그리고 통계적 해석 기법을 사용하여 개발된 알고리듬의 출력 특성을 분석하였다. 본 논문에서 제안한 알고리듬은 입력과 출력의 변화를 통하여 GSM과 PACS와 같은 다른 이동통신 표준안에도 적용될 수 있다.

Ⅱ. IS-95A 인증 알고리듬 개발

1. IS-95A 인증알고리듬

IS-95A, GSM 그리고 PACS는 모두 challenge-response 형의 인증 시스템을 채택하고 있다[7]. 즉, 기지국이 임의의 challenge 데이터를 주면 이동국은 자신의 비밀키로 인증 알고리듬을 수행하여 response, 즉, 인증서명 데이터, 를 전송한다. 그리고 기지국은 이동국과 동일한 방법으로 인증 서명 데이터를 생성하여 그 값이 일치하면 이동국을 합법적인 사용자로 간주한다.

IS-95A의 경우 challenge에 대한 response를 계산하는 알고리듬을 AUTH_SIGNATURE라 명한다. AUTH_SIGNATURE는 그림 1과 같이 152비트 데이터를 입력으로 하여 18비트의 response를 생성한다.

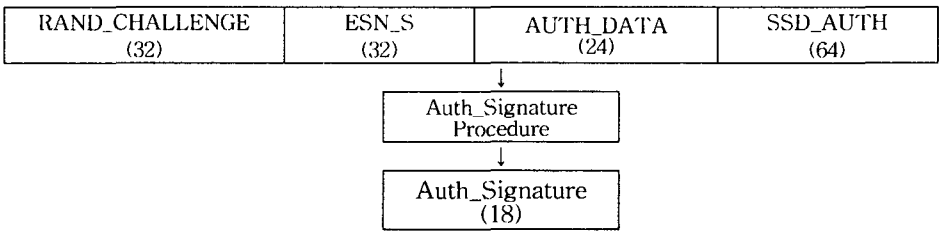


그림 1. AUTH_SIGNATURE의 계산.

2. 인증 알고리듬의 설계

IS-95A와 관련된 미국의 특허[8]에서는 인증 알고리듬과 인증 키 생성 알고리듬에 "Jumbel"이라는 해쉬함수를 사용하였으며, 다른 형태의 해쉬함수도 사용할 수 있다고 언급하였다. 해쉬함수는 부울함수를 사용하는 형태와 블록암호화 알고리듬을 사용하는 형태 그리고 모듈라 연산을 사용하는 형태로 구분된다[8]. 이동통신시스템에서는 고속처리가 가능한 Bool 함수를 사용한 해쉬함수를 사용하는 것이 유리하다[9]. 본 논문에서는 해쉬함수

를 사용하여 인증 알고리듬을 구현하였다.

부울함수를 사용한 해쉬함수로는 MD5[10], SHA-1[11], RIPEMD-160[12], HAVAL[13] 등이 있다. 본 논문에서는 이들 함수에 대한 공격[14, 15]과 반복연산의 구조를 분석하여 인증 알고리듬의 설계에 반영하였다. 그림 2는 AUTH_SIGNATURE의 구조이다. 각 라운드에서는 그림 3의 단계연산(step operation)을 8회 반복한다.

그림 3에서 a~h는 32비트 연쇄변수가 저장되는 레지스터이며, X>>>s 은 32비트 데이터 X를 s번 오른쪽으로 순환시키는 연산을 의미한다. M_j는 각 라운드의 j번째 단계연산에서 사용되는 32비트 입력 데이터이고 K_i는 각 라운드마다 한 개씩 있는 상수로서 2, 3, 4 라운드에서만 사용된다. 해쉬함수에서 사용되는 부울함수는 다음과 같은 안전성 기준[13, 16]을 만족하는 것을 선정하여 사용한다. 1, 2 라운드에는 7변수 부울함수를, 그리고 3, 4 라운드에는 5변수 부울함수를 사용한다.

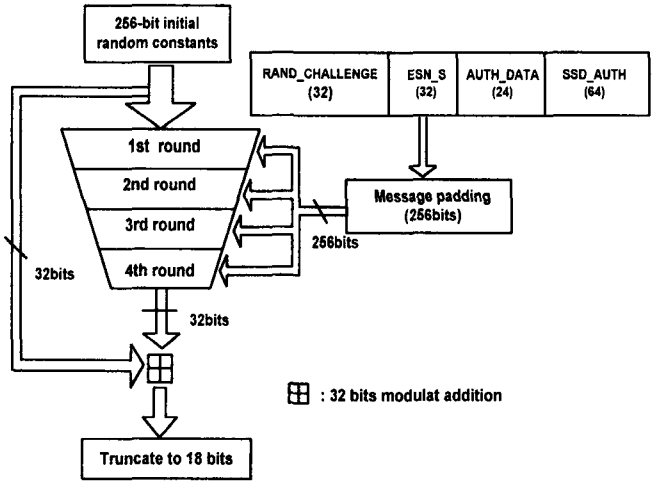
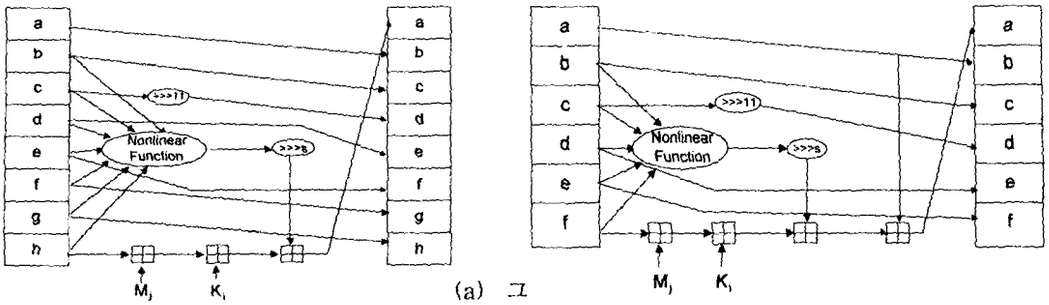


그림 2. AUTH_SIGNATURE 알고리듬의 구조.



(a) 그림 2의 1, 2 라운드를 위한 단계연산 블록선도.

(b) 그림 3의 3, 4 라운드를 위한 단계연산 블록선도.

그림 3. 그림 2의 단계연산 블록선도.

RIPEMD-160에서는 a 레지스터, 입력메시지, 상수값 그리고 부울함수 출력을 모듈라 연산한 결과와 c 레지스터 등 두 곳에 좌측 순환이동을 적용함으로써 differential attack[14]에 대비하였다. 그리고 SHA-1은 a와 b 레지스터 두 곳에 순환이동이 있으며, HAVAL은 부울함수의 출력과 h 레지스터에 순환이동을 적용함으로써 differential attack에 대비하였다. 위의 예들을 분석해 볼 때, 순환이동을 적용하는 레지스터의 위치는 적절히 선택하고 두 곳 정도에 순환이동을 적용함으로써 differential attack을 피할 수 있음을 알 수 있다. 본 알고리듬에서는 부울함수의 출력과 c 레지스터에 순환이동을 적용하였다.

RIPEMD-160과 MD5는 단계마다 다른 순환 이동량을 사용하며, HAVAL과 SHA-1은 동일한 순환 이동량을 사용하였다. RIPEMD-160에서는 순환 이동량을 5~15의 값으로 하였으며, MD5에서는 처리 데이터 단위(32비트)에 대하여 상대소수가 되도록 하였다. 이때 주의할 점은 각 순환이동 레지스터는 어떤 특정한 패턴을 갖지 않도록 하여야 한다는 것이다. 예를 들어, 전체 순환 이동량은 32의 배수가 되지 않아야 하며, 순환상수는 4의 배수가 되지 않도록 한다. 설계과정에서 해쉬함수 출력의 통계적 특성[17]을 분석해 본 결과 순환 이동량을 고정시키는 것보다는 라운드마다 그 값을 달리하는 것이 유리함을 알았다. 본 알고리듬에서는 부울함수의 출력과 c레지스터에 순환이동을 적용한다. 부울함수 출력의 경우 순환 이동량은 5, 6, 7, 10, 13, 14, 15를 매 라운드마다 적절히 배열하여 사용하며, c 레지스터의 경우는 11로 고정한다.

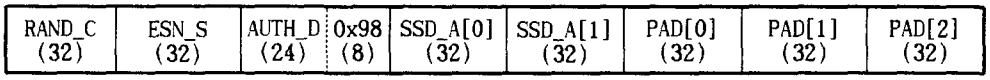
MD4와 RIPEMD에서는 각 단계연산에서 사용되는 메시지 입력 순서에 약점이 발견되어 공격을 당하였다[14, 15]. RIPEMD을 개선한 RIPEMD-160에서는 round 1-2에서 가까이 있었던 두 단어는 2-3라운드에서는 멀리 떨어져 있도록 하였다. 반대로 round 1-2에서 멀리 떨어져 있었던 두 단어는 2-3라운드에서는 가까이 있도록 하였다. 본 알고리듬에서도 이러한 면을 고려하여 아래 표 1과 같이 입력 메시지를 배열하였다. 256비트의 입력

데이터는 8개의 32비트 워드로 나뉘어져 각 라운드의 단계연산에 사용된다.

표 1. 입력 메시지의 처리순서

단계순서 / 라운드	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	3	6	0	4	1	7	2	5
3	1	5	6	2	0	3	7	4
4	6	4	2	1	3	5	0	7

개발한 인증 알고리듬은 256비트의 입력 데이터가 필요한 반면, IS-95A 인증 메카니즘에서는 152비트의 데이터가 입력되므로 104비트의 메워넣기(padding) 데이터를 만들어야한다. 그림 4는 152비트의 데이터를 사용하여 256비트의 입력 데이터를 만드는 방법을 도식화한 것이다. W(2)에서 AUTH_D는 24비트이므로 32비트로 만들기 위하여 입력 데이터의 길이 152(0X98)를 8비트로 표현하여 padding 데이터로 설정하였다.



PAD[0] = RAND ⊕ ESN_S ⊕ AUTH_D
PAD[1] = PAD[0] ⊕ SSD_A[0]
PAD[2] = PAD[1] ⊕ SSD_A[1]
w(2) = AUTH_D || 0x98, || : bit concatenation

그림 4. AUTH_SIGNATURE 알고리듬을 위한 데이터 메워넣기.

대부분의 최신 프로세스들은 superscalar 구조로 설계되어 pipeline processing이 가능하므로 해쉬함수의 단계연산을 이러한 구조에 맞게 설계하면 소프트웨어 구현이 매우 효율적이다[18, 19]. 그림 3에서 현 단계연산에서 변경된 a 레지스터 값이 바로 다음 단계연산의 부울함수 입력으로 사용되지 않기 때문에 미리 계산해 둘 수 있으므로 pipeline processing이 가능하다.

3. 고 찰

인증 알고리듬의 출력이 가져야할 성질에는 랜덤성, SAC 그리고 입출력간의 비선형성이 있다[17]. SAC란 알고리듬의 입력중 한 비트가 변할 경우 출력의 각 비트가 변할 확률이 1/2이 됨을 의미하며, 입출력간의 비선형성이란 입력과 출력의 특정 비트들 간의 선형

관계가 성립하지 않음을 의미한다. 개발된 인증 알고리듬이 이러한 특성을 가지는지 검사하는 방법으로 통계적 분석방법을 사용한다. 출력의 랜덤성을 검사하기 위하여 equidistribution test, hamming weight 그리고 runs test를 사용한다. 입출력간의 비선형성 검사를 위하여 linear dependency test를 사용한다. equidistribution test을 통하여 출력 데이터 패턴이 균일한 분포를 가지는지 검사하고, hamming weight test를 통하여 출력의 각 비트에서 1이 발생할 확률이 1/2인지 검사한다. 그리고 runs test를 통하여 출력의 각 비트 위치에서 이전 비트에 대하여 변화가 발생할 확률이 1/2인지 검사한다.

통계적 분석(statistical analysis)이란 임의의 랜덤 프로세스로부터 구한 일련의 관측치(observations)를 가지고 있으며, 이러한 관측치가 특정한 성질을 가지는가 검사하는 것이다. 여기서 관측치가 가지기를 바라는 성질을 영가정(null hypothesis)이라 하며, 반대의 경우를 선택가정(alternate hypothesis)이라 한다. 통계적 분석을 위하여 관측치와 영가정간의 차이에 해당하는 통계치를 계산한다. 통계치의 계산방식은 다를 수 있으나 통계치가 작을수록 관측치가 영 가정에 가까워짐을 의미한다. 통계치의 확률분포가 P(x)라 주어질 때, P(x>a)를 임의의 통계치 a에 대한 유의수준(significance level)이라 부른다. 통계치로부터 계산된 유의 수준과 임계치를 비교하여 영 가정의 채택여부를 가린다. 일반적으로 사용되는 임계치는 0.05이다. 임계치가 0.05란 말은 영가정이 참이지만 거짓이라는 잘못된 결론을 내릴 확률이 0.05라는 의미가 된다. 그러므로 통계치가 임계치보다 클 경우에는 영가정을 채택한다. 본 논문에서는 chi squared goodness-of-fit test기법을[20] 사용하여 통계치의 계산과 영가정의 채택여부를 판별한다.

각각의 평가 항목에 대하여 동일한 실험을 다른 데이터를 사용하여 20번 반복하였다. Fisher-Pearson test를[21] 통하여 20개의 유의수준으로부터 하나의 유의수준을 얻고 이 값을 임계치인 0.05와 비교한다. 표 2는 각 항목의 통계적 분석 결과를 나타낸다. 표 2의 첫째 칸의 입력특성은 인증 알고리듬의 입력 데이터에서 1이 차지하는 비율이 20%, 50%, 80% 임을 의미한다. 결과를 분석해 볼 때 설계된 인증 알고리듬의 출력은 통계적으로 양호한 특성을 가짐을 알 수 있다. linear dependency 검사를 위한 행렬은 행 감소(row reduction)를 수행한 결과 full rank를 가지며, 이는 알고리듬의 입력과 출력간에 선형성이 존재하지 않음을 의미한다.

표 2. 인증 알고리듬에 대한 유의수준

분석항목 / 입력특성	equidistribution	Hamming weight	runs	SAC
20%	0.0714665	0.0662347	0.839401	0.670356
50%	0.143139	0.413732	0.156449	0.199088
80%	0.522117	0.546972	0.175676	0.34747

본 알고리듬을 C언어로 구현하여 Pentium-100MHz CPU의 IBM-PC 호환 컴퓨터에서 속도를 측정해 본 결과 120000회 반복 수행에 0.87초가 걸렸다. 이 정도의 속도면 실제 시스템에 구현 가능한 것으로 여겨진다.

Ⅳ.인증키 생성 알고리듬 개발

1. 인증키 생성 메카니즘

인증 알고리듬의 입력 데이터로 사용되는 SSD는 SSD_Generation 알고리듬을 사용하여 갱신된다. SSD_Generation 알고리듬은 이동국 특정 정보, 랜덤 데이터, 이동국 A-key를 입력으로 가진다. A-key는 64비트이며 이동국의 영구 보안 및 식별 메모리에 저장된다. A-key는 단지 이동국과 그에 관련된 HLR/인증 센터만 알 수 있다. 이동국이 SSD 갱신 메시지를 수신하면, 그림 5와 같이 SSD_Genaration 절차의 입력 파라메타를 설정하여 SSD_Generation 절차를 수행한다. 이동국은 SSD_A_NEW와 SSD_B_NEW 값을 SSD_Generation 절차의 출력 값으로 설정한다.

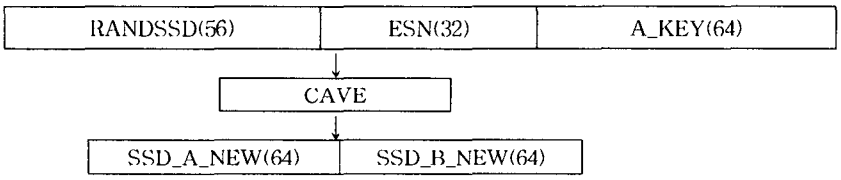


그림 5. 공유 비밀 데이터(SSD)의 계산.

2. Oracle 해쉬함수

인증 키 생성 알고리듬의 간결성을 도모하기 위하여 인증 알고리듬과 동일한 루틴을 사용하도록 한다. 본 절에서는 인증 키 생성 알고리듬의 근본적 구조를 제공하는 oracle 해쉬함수의 특성[5, 6]에 간략히 관하여 서술한다.

Oracle 해쉬함수는 다음과 같은 성질을 갖는다.

성질 1) 확률적이다

동일한 입력에 대하여 해쉬함수를 수행할 때마다 서로 다른 해쉬값을 낸다. 기존의 해쉬함수는 결정적 구조이다. 왜냐하면 동일한 입력에 대하여 해쉬함수를 두 번 구동할 경우 같은 동일한 결과를 내기 때문이다. 그러나 확률적 성질을 갖는 해쉬함수 H는 동일한 입력에 대하여 구동할 때마다 서로 다른 출력을 낸다. 즉 H(x)는 H의 랜덤 선택에 의존하는 새로운 랜덤변수이다. 이러한 랜덤화는 H(x)가 x에 관한 모든 부분 정보(partial information)를 숨기는 요구조건을 제공한다.

성질 2) 검증능력을 갖는다.

위의 확률적 성질을 만족함에도 불구하고 주어진 해쉬값에 대하여 어떤 입력으로부

터 생성되었는지 검증할 수 있는 능력을 갖는다. 주어진 입력 x와 해쉬값 c에 대하여 c가 x의 해쉬인지 아닌지를 올바로 판단하는 검증 알고리듬(verification algorithm) V가 있어야 한다.

Oracle hashing의 정의는 아래 3개의 요구조건으로 구성된다.

조건 1) 완전성(Completeness)

c가 x에 H를 적용하여 생성되었다고 할 때, 검증 알고리듬(verification algorithm) V는 x와 c의 쌍을 받아들일 것이다.

조건 2) 정확성(Correctness)

c가 x에 대하여 해쉬함수 H를 적용하여 생성되지 않았다고 할 때, x와 c 쌍을 받아들이도록 V를 속일 수 없다. 다시 말하면, c를 x와 y의 합법적 해쉬값으로 받아들일 수 있는 두 개의 다른 입력 x , y 와 c를 찾는 것이 불가능하다.

조건 3) 비밀성(oracle security)

c=H(x)인 c를 가지더라도 x에 관한 아무런 정보를 얻을 수 없다. 게다가 c=H(x)되는 x를 찾기 위해 전 도메인을 완전탐색 할 수도 없다.

해쉬함수의 입력에 랜덤성을 부과하여 아래와 같이 3개의 간단한 oracle hashing을 구현할 수 있다[5].

형태 1) H(x, r) = r, h(r, x)

h는 해쉬함수이고 r은 길이가 β인 랜덤열이다. MD5의 경우는 β=128, SHA의 경우에는 β=160이 적절하다. 검증과 완전성은 간단하다. 즉 주어진 r에 대하여 x의 해쉬값을 계산하여 이전에 계산된 것과의 일치여부만 확인하면 되는 것이다. 정확성은 해쉬함수 h의 충돌 저항성을 따른다. 비밀성의 요구조건은 h에 관한 아래의 정의를 따른다.

정의 (τ, δ)-비밀성[5, 22]

만약 시간 τ 내에 수행되는 어떤 adversary A와 distinguishier D에 대하여 아래의 조건이 성립하면, 해쉬함수 h는 H(x, r) = r, h(r, x)와 $\{0,1\}^*$ 상의 어떤 Δ분포에 대하여 (τ, δ)-비밀성이다.

$$| \mathrm{Prob}(\mathrm{D}(x, A(r, h(r,x))) = 1) - \mathrm{Prob}(\mathrm{D}(x, A(r, h(r,y))) = 1) | \leq \delta$$

여기서 $\{0,1\}^*$는 길이가 유한한 이진열을 나타낸다. 그리고 x와 y는 각각 Δ와 $r \in_r \{0,1\}^\beta$로부터 독립적으로 추출한 것이다.

형태 2) H(x, r) = r, h(r, h(x))

완전성과 정확성은 형태 1)의 구조와 같다. 안전성도 유사하게 정의할 수 있다. 형태 2)

가 형태 1) 보다 안전하다. 왜냐하면 형태 2)가 공격당하면 형태 1)도 공격당하나 역은 성립하지 않기 때문이다.

형태 3) H(x, r) = r, h(r_1, h(r_2, x))

이 구조는 HMAC (Hash based MAC)[23]에 기초를 하여 설계하였다. $r_1 = r \oplus opad$,

$r_2 = r \oplus ipad$ 그리고 ipad 와 opad는 고정된 상수 값이다. 완전성과 정확성은 위의 경우와 같다. 형태 3)은 훨씬 더 안전하다. 왜냐하면 형태 3)이 공격당하면 형태 1)과 형태 2)는 공격당하나 역은 성립되지 않기 때문이다.

3. Oracle 해쉬함수를 이용한 인증 키 생성 알고리듬

개발된 인증 키 생성 알고리듬은 2절에서 제시한 구조 중에서 h(r, h(x)) 형의 oracle hash 함수 구조를 취한다. 인증 키 생성 메카니즘에서는 oracle hash 알고리듬에서 H가 선택하는 랜덤변수 r 값에 대응되는 값은 없다. 그러나 152비트의 입력 중 인증 키 갱신 과정에서 인증 키 생성 함수가 수행될 때마다 바뀌는 RANDSSD(56비트)가 있으며, 이를 랜덤변수 r 값 대용으로 사용한다.

개발된 인증 키 생성 알고리듬의 전체적인 구조는 그림 6에 나타나 있으며 자세한 과정은 다음과 같다. 256비트 데이터가 해쉬함수에 입력되어 4라운드의 단계연산을 거쳐 192비트의 데이터가 출력된다. 이 해쉬함수를 수식으로 표현하여 h(w)라 하자. 여기서 w는 256비트의 입력 데이터이다.

과정 1) 그림 8과 같이 152비트 입력 데이타를 256비트의 1차 해쉬함수의 입력, W_1으로 만든다.

과정 2) 과정 1)의 W_1을 입력으로 한 1차 해쉬함수의 출력 h_out은 다음과 같다.

h_out = h(W_1)

= {h_out[0], h_out[1], h_out[2], h_out[3], h_out[4], h_out[5]}

과정 3) 1차 해쉬값으로 나온 192비트와 randssd 56비트를 이용하여 그림 8과 같이 2차 해쉬함수의 입력 데이터 W_2를 구성한다.

과정 4) 2차 해쉬함수의 출력은 다음과 같이 계산된다.

h_out = h(W_2)

= {h_out[0], h_out[1], h_out[2], h_out[3], h_out[4], h_out[5]}

과정 5) h_out의 192비트로부터 인증키 SSD_A와 SSD_B를 다음과 같이 취한다.

SSD_A = {h_out[0], h_out[1] } ,

SSD_B = { h_out[2], h_out[3]}

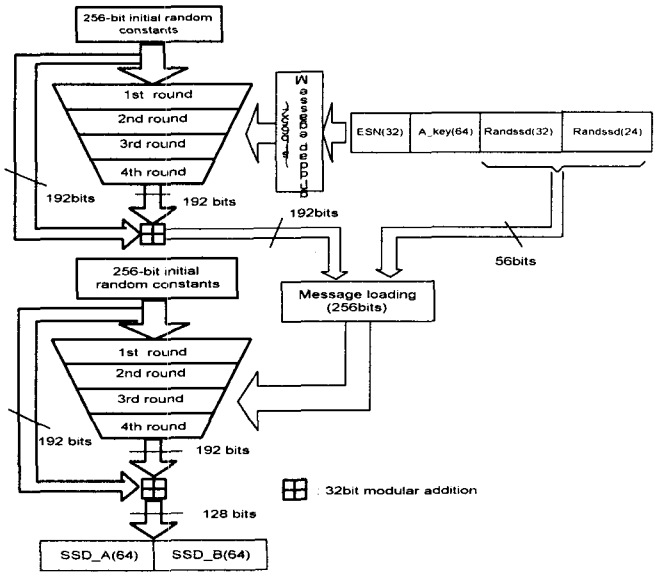


그림 6. 인증 키 생성 알고리듬의 블럭선도.

ESN (32)	A_key[0] (32)	A_key[1] (32)	Randssd[0] (32)	Randssd[1] (24)	0x98 (8)	PAD[0] (32)	PAD[1] (32)	PAD[2] (32)
	$W_1(0)$	$W_1(1)$	$W_1(2)$	$W_1(3)$	$W_1(4)$		$W_1(5)$	$W_1(6)$ $W_1(7)$

PAD[0] = ESN ⊕ A_key[0] ⊕ Randssd[0]
PAD[1] = PAD[0] ⊕ A_key[1]
PAD[2] = PAD[1] ⊕ (Randssd[0] || 0x98), || : bit concatenation

그림 7. 1차 해쉬를 위한 데이터 메워넣기.

h_Out[0] (32)	h_out[1] (32)	h_out[2] (32)	h_out[3] (32)	h_out[4] (32)	Randssd[0] (32)	Randssd[1] (24)	0x98 (8)	PAD[0]
	$W_2(0)$	$W_2(1)$	$W_2(2)$	$W_2(3)$	$W_2(4)$	$W_2(5)$	$W_2(6)$	$W_2(7)$

PAD[0] = h_out[5] ⊕ (Randssd[1] || 0x98), || : bit concatenation

그림 8. 2차 해쉬를 위한 입력 데이터 배치.

4. 인증 키 생성 알고리듬의 안전 특성 검토

인증 알고리듬에서의 분석방법과 동일하게 equidistribution test, hamming weight test, runs test를 수행하였으며, 그 결과를 표 3에 나타내었다. 모든 test들의 유의수준이

임계치보다 크므로 인증 키 생성 알고리듬의 출력특성도 양호함을 알 수 있다.

표 3. 인증 키 생성 알고리듬에 대한 유의수준

분석항목 / 입력특성	equidistribution	Hamming weight	runs
20%	0.286993	0.994108	0.928421
50%	0.358905	0.799294	0.989545
80%	0.587574	0.246302	0.913677

V. 결 론

본 논문에서는 부울함수를 사용한 해쉬함수를 이용하여 IS-95A 인증 시스템에 적용 가능한 인증 알고리듬을 개발하였다. 개발된 알고리듬은 4라운드의 해쉬함수이며 pipeline processing이 가능한 구조를 가진다. 1라운드와 2라운드에는 7변수 부울함수를 사용하였으며, 3라운드와 4라운드에는 5변수 부울함수를 사용하였다. 개발된 인증 알고리듬은 C언어로 구현되었으며, Pentium-100MHz에서 속도를 측정한 결과 120,000번 수행에 0.87초가 소요되었다. 또한 개발된 인증 알고리듬을 oracle 해쉬함수의 형태로 사용하여 인증 키 생성 알고리듬을 구현하였다. 그리고 통계적 분석 방법을 사용하여 개발된 알고리듬의 출력 특성을 분석하였다. 출력 특성 분석에 사용된 검사로는 equidistribution test, hamming weight test, runs test, SAC test 그리고 linear dependency test가 있다. 통계적 분석을 적용한 결과 본 논문에서 제안한 인증 알고리듬과 인증 키 생성 알고리듬은 통계적으로 양호한 특성을 나타내었다.

본 논문에서 제안한 알고리듬은 입력 데이터의 형태와 출력 데이터의 길이를 변경하면 GSM과 PACS와 같은 다른 이동통신 인증 시스템에도 동일하게 적용될 수 있다.

[참고문헌]

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

[2] TIA/EIA IS-95A, *Mobile-Base Station Compatibility Standard for Dual-Mode Wideband Spread spectrum Cellular System, Interim Standard 95,* 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] Ran Canetti, "Toward realizing Random Oracles : Hash Functions that Hides All Partial Information", *Crypto'97, Spring-Verlag ,LNCS 1294,* pp.455-469, 1997.

[6] M. Bellare, J. Kilian and P. Rogaway. "Random oracles are practical : a paradigm for designing efficient protocols," *1st ACM Conference on Computer and Communications Security,* pp.62-73, 1993.

[7] 이국희, 류정수, 하재철, 문상재, "디지털 이동통신 인증 서비스의 기술동향," *Telecommunication Review,* vol.6, no.2, 1996

[8] C.J. Mitchell, K. Piper and P. Will, *Contemporary Cryptology : The Science of Information ntegrity,* G.J.Simmons, editor, IEEE Press, pp.325-378, 1991.

[9] J. A. Reeds Ⅲ, P. A. Treventi, Service Provision Authentication Protocol, *U.S.A Patent No. 5153919,* Oct. 6, 1992.

[10] R. Rivest, "The MD5 message digest algorithm," *Requests for Comments(RFC) 1321,* 1992

[11] U.S. Department of Commerce(NIST), Secure Hash Standard, *FIPS PUB 180-1,* 1995, April 17.

[12] H. Dobbertin, A. Bosselaers and B. Preneel, RIPEMD-160 : A strengthened version of RIPEMD, *Fast Software Encryption,* LNCS1039, Springer-Verlg, 1996.

[13] Y. Zhang, J. Pieprzyk, and J. Seberry, "HAVAL - A One-way Hashing Algorithm with Variable Length of Output," *Auscrypt'92 Abstract,* 1992.

[14] B. den Boea and A. Bosselaers, "Collision for the compression function of MD5," *Advances in Cryptology-Eurocrypt '93,* LNCS 773, Spring-Verlag, pp.293-304, 1994.

[15] H. Dobbertin, "RIPEMD with two-round compression function is not collision free," *Journal of Cryptology.*

[16] S. Bakhtiary, R. Safavi-Naini, J. Pieprzky, "Keyed Hash Function," *Cryptography : Policy and Algorithm,* Springer-Verlag, July 1995.

[17] 이국희, 정명준, 이상곤, 문상재, 정원영, 김태근," 이동통신을 위한 인증 알고리듬 개발," *WCT '97 워크샵 자료집,* pp.171-175, 1997. 12. 16.

[18] A. Bosselaers, R. Govaerts, and J. Vandewalls, " Fast hashing on the Pentium," *Advances in Cryptology-Crypt '96, LNCS 1109,* Spring-Verlag, pp.298-312., 996.

[19] A. Bosselaers, R. Govaerts, and J. Vandewalls, " SHA : a design for parallel architecture," *proceeding of Eurocrypt '97, LNCS 1233,* Spring-Verlag, pp.348-362, 1997.

[20] Paul G. Hoel, *Introduction to Mathematical Statics*, 4th ed., John Wiley & Sons, 1971.

[21] H. M. Gustafson, E. P. Dawson, and J. Dj. Golić, "Randomness Measures Related to Subset Occurrence," *Cryptography: Policy and Algorithms*, Brisbane, Queensland, Australia, pp. 132-143, July 1995.

[22] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Cryptographic Hash Functions: A Survey," *Tech. rep. 95-09, Department of Computer Science, University of Wollongong,* July 1995.

[23] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Practical and Secure Message Authentication," in *Series of Annual Workshop on Selected Areas in Cryptography(SAC '95),* (Ottawa, Canada), pp. 55-68, May 1995.