情報保護學會論文誌
第13券 第6號, 2003. 12

# 자바 카드상에서의 3GPP, 3GPP2 인증 메커니즘 구현

조 승 환*†‡, 전 성 익*, 이 정 우*, 이 옥 연**, 한 진 희*, 이 세 광***

# Implementation of authentication mechanism for 3GPP, 3GPP2 on java card

Seung-hwan Jo*†‡, Sung-ik Jun*, Jeong-woo Lee*, Ok-yeon Yi**,
Jin-hee Han*, Se-kwang Lee***

요 약

현재 전세계적으로 이동통신 단말기의 발전은 빠른 속도 진행중이다. 기본적인 전화 통화이외에 다양한 멀티미디어 서비스 및 국제적 로밍이 가능한 3세대 이동통신으로 가고 있으며 이에 따라 더욱 생활 필수품이 될 것이다. 이처럼 많은 편리성을 제공해주는 이동통신 서비스에 있어 개인의 사생활 보호 및 안전한 데이터 전송은 필수적 요소이다. 3세대 이동통신 서비스를 제공하려는 비동기식, 동기식 진영 양쪽 모두 사용자의 인증 기능을 정의하고 있다. 이 인증기능은 자바카드 기반의 USIM, UIM application에 포함되어 제공될 것이다. 본 논문에서는 두방식의 인증 메커니즘과 키생성 메커니즘을 살펴보고 자바카드와 동일한 환경에서 구현한 내용과 테스트한 결과에 대하여 설명한다. 또 표준에서 불분명하게 정의되어 실제 Java card에서 구현시 발생할 수 있는 문제에 대하여 분석하였다.

ABSTRACT

The development of mobile phone is growing fast in the all over the world. Besides the basic voice communication, many multimedia services and global roaming service are capable in the 3rd generation mobile telecommunication. Because mobile phone has been the essential tool to communicate, the protection of privacy and the safe transmission are critical ones. In synchronous, asynchronous mode IMT2000 service, the mechanism of mutual authentication and generation of cipher key and integrity key are implemented in smart card chip called UIM, USIM. In this paper, we describe the authentication mechanism of 3GPP and 3GPP2 and its implementation results. Then, we specify a few problems which are not defined in standard.

**keyword :** *3GPP, 3GPP2, smart card, MILENAGE, authentication mechanism*

## Ⅰ. 소 개

국제적 로밍, 고속 데이터 전송, 멀티미디어 서비스 등으로 관심을 받아온 IMT2000에 보안 메커니즘은 2세대에서 지적되어온 문제점을 개선하여 사용자에게 더욱 안전한 기능 사용을 보장해준다. 이 보안 메커니즘에는 각 단간의 인증, 암호화, 무결성 확인 등을 포함하고 있다. 이중 2세대 보안 구조 중 특이한 것으로써 3세대에서도 채택이 되고 있는 것이 USIM(Universal Subscriber Identity Module), UIM(User Identity Module)이라 불리우는 스마트 카드의 사용이다. 이 USIM은 identification에 필요한 데이터나 인증

* 한국전자통신연구원({hongcha, sijun, jeow7, hanjh}@etri.re.kr)
** 국민대학교 자연과학대학 수학과(oyyi@kookmin.ac.kr)
*** 고려대학교 정보보호대학원(hongcha@cist.korea.ac.kr, gausslee@cist.korea.ac.kr)
† 주저자, ‡ 교신저자, 논문접수일 : 2003년 6월 19일, 심사완료일 : 2003년 10월 15일

과 관련된 데이터들, 데이터 생성구조를 포함하고 있다. 그리고 로밍 도중에 네트웍 망의 지원아래 사용자의 USIM에 저장된 민감한 데이터에 대한 노출 없이 보안이 이루어진다.[1] IMT2000 시스템에서의 인증에 관한 연구는 많이 이루어져 있어 여러 시뮬레이터를 통한 검증이 이루어져 있다. 하지만 USIM, UIM에 구현되어야 할 인증구조에 대한 연구와 실제 동일한 하드웨어상에서 구현한 예는 찾기 어렵다. 본 논문에서는 USIM, UIM이라 불리는 스마트 카드 내부에 구현되어야 할 인증 메커니즘에 대하여 설명하고, 실제 USIM, UIM과 동일한 하드웨어 환경에서 구현 테스트한 결과를 분석하였다. 또한 표준에 정확히 정의되어 있지 않아 구현상 문제를 일으킬 수 있는 점들을 지적하고 해결 방안에 대해서도 제안한다.

## II. 관계된 연구

### 2.1 자바 카드

자바 카드상에서 구현한 결과물들은 다음과 같은 여러 가지 잇점을 가진다.[2]

상호 동작성 - 자바카드 API로 개발한 애플릿은 자바 카드 기술을 적용한 스마트카드가 어느 회사가 만든 카드거나 동작한다.
안전성 - 자바 카드 기술은 자바 프로그래밍 언어의 안전성을 이어 받았고 공개적으로 검증 받은 기술이다.
멀티 애플리케이션 운영성 - 단일 스마트 카드상에 다양한 애플리케이션이 상호 안전하게 동작할 수 있게 한다.
현존하는 표준 지원성 - 자바 카드 API는 국제표준 ISO7816, EMV등을 지원한다.

### 2.2 USIM상에서의 3GPP 보안 메커니즘

#### 2.2.1 상호 인증

여기서 살펴보는 네트워과 사용자간의 상호인증은 USIM과 AuC(Authentication Center)간에 비밀키 K를 소유하고 있다는 정보를 보여주므로써 달성이 된다.

AuC는 사용자 비밀키 K와 SQN(Sequence Number), AMF(Authentication Management Field), RAND (Random Challenge)등을 활용하여 MAC값을 생성하고 이것과 함께 RES(Response)값을 생성한다. MAC값은 네트웍을 신뢰할수 있게하는 값이고 RES는 사용자를 신뢰 할수 있도록 하는 값이다.

네트웍인증은 USIM에서 이루어지고 사용자 인증은 USIM생성한 RES와 AuC가 생성하고 전송해준 XRES (Expected response)값을 VLR(Visitor Location Register)이 비교하므로써 이루어진다.

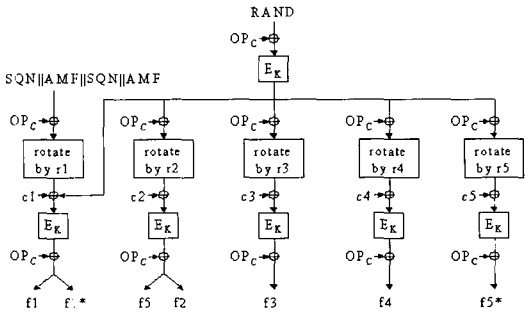
또한 USIM과 HE(Home Environment)는 $SQN_{MS}$와 $SQN_{HE}$를 각각 갱신 보관하여 네트웍인증을 지원한다. 여기서 $SQN_{MS}$는 USIM안의 SQN중 최고값이고 $SQN_{HE}$는 각 사용자마다 저장하는 가장 마지막으로 생성한 SQN이다.[3]

#### 2.2.2 MILENAGE

3GPP에서 사용하는 인증 구조는 MILENAGE라 불리우는 AES기반의 구조이다.

이 MILENAGE는 필수 사항이 아니다. 2세대 이동통신 GSM에서도 비동기식 진영은 인증 구조를 운영자에게 맡기는 정책을 취하였다. 하지만 이 때문에 잘못된 알고리즘 선택이 있어서 취약점이 발견되고서 다시 교체하는 일이 있었다. 그래서 3GPP에서는 부적절한 알고리즘의 선택을 피하기 위해 MILENAGE라는 검증된 알고리즘을 표준에 포함시켜서 필요하면 활용할 수 있게 하고 있다.

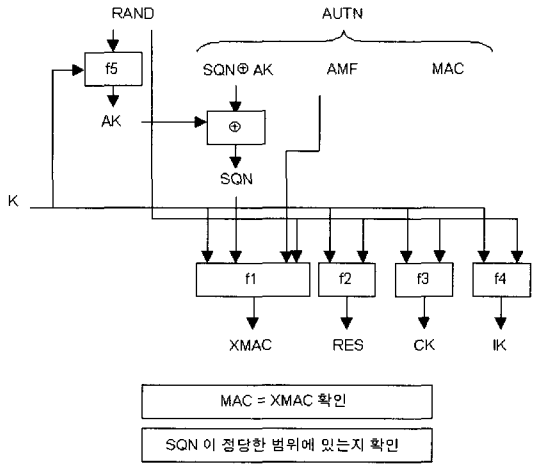
이 MILENAGE구조는 USIM과 AuC에서 사용이 되며 사용자인증, 네트웍 인증등의 기능을 수행한다.



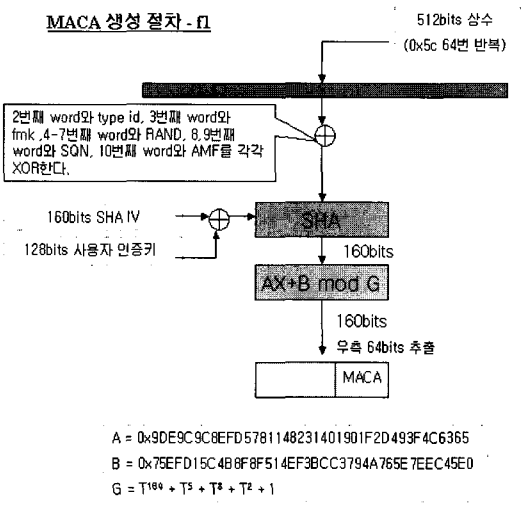

〔그림 1〕 MILENAGE

#### 2.2.3 검증

USIM은 MILENAGE를 써서 f1 XMAC(Expected Message Authentication Code), f1* MAC-S(Resynchronization Message Authentication Code), f2 RES, f3 CK(Cipher Key), f4 IK(Integrity Key), f5 AK(Anonymity Key), f5* AKS(Resynchonization Anonymity Key)를 각각 생성한다.




〔그림 2〕 사용자 인증과 SQN확인




〔그림 3〕 3GPP2의 f1 함수구조

USIM은 ME(Mobile Equipment)로부터 받은 RAND와 USIM과 AuC가 공유하고 있는 사용자 비밀키 K를 입력으로 f5함수를 동작시켜 AK를 얻어 낸다. 이 AK는 SQN를 암호화 하기 위하여 사용되는 키이다. SQN은 현재 3가지 운영 모드가 제안되고 있다. 이중 not time based SQN 운영에서만 AK를 사용하며 이로써 사용자의 익명성을 보장하여 준다.

위의 과정을 통해 얻은 SQN 그리고 K, AMF와 RAND를 입력으로 USIM의 f1함수는 XMAC을 생성한다.

이 자신이 계산한 XMAC과 AUTN에 포함되어 온 MAC을 비교하여 일치하면 사용자는 네트웍을 인증할 수 있다.

또 MAC값 확인 뒤에 SQN과 $SQN_{ms}$ 를 비교한다. ME로부터 받아온 SQN 과 비교하는 것은 SQN의 freshness를 보장하되 빈번한 authentication failure를 막기 위함이다.

MAC 과 SQN의 verification이 끝나면 f2의 결과로 얻은 사용자 인증을 위한 값 RES, f3의 결과를 얻은 암호화 키 CK, f4의 결과로 얻은 무결성 키 IK를 생성한뒤 ME에 보내준다.[3]

### 2.3 UIM상에서의 3GPP2 보안 메커니즘

3GPP2에서도 인증 및 키일치를 위한 함수 구조가 필수항목으로 정해져 있지는 않다. 3GPP와 마찬가지로 UIM(3GPP에서의 USIM)과 AuC에서만 구현되면 되기 때문에 동일한 크기의 입력과 출력 그리고 암호학적 안전성이 보장이 되는 함수들이면 어떠한 것도 사용될 수 있다.[4]

3GPP2 표준문서에는 SHA-1 기반에 AX+B mod G라는 다항식 연산을 덧붙여서 만든 f1 부터 f5까지의 함수들을 제안하고 있다.

F함수들을 살펴보면 처음 0x5C가 64번 반복된 512bits 상수와 RAND, type id, family key, SQN, AMF를 정해진 위치에 XOR한 뒤 SHA1의 입력 메시지로 받고 SHA1의 IV와 사용자 인증 키를 left-most로 XOR하고 해쉬한다. 해쉬한 결과값은 160bits로 나오고 이 결과를 AX+B mod G의 연산 중 X값으로 넣어서 고정 A,B,G에 대한 다항식 연산을 취하고 결과를 구한다.[4]

각 F함수들은 다른 type id를 사용하고 RAND혹은 Counter를 사용하는 등 약간씩 차이점이 있다.

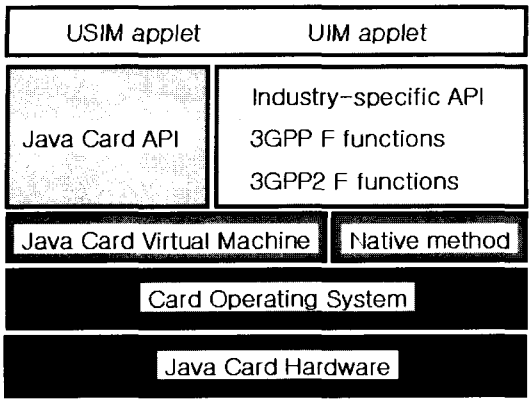
아래의 [그림 3]은 방금 설명한 3GPP2의 f함수 중 MACA값을 생성하는 f1함수에 대한 구조이다.

## Ⅲ. 구 현

본 구현물은 자바카드와 동일한 환경을 제공해주는 JCOP IDE와 실제 자바 카드와 동일한 하드웨어 환경을 가진 보드 에뮬레이터에서 동시에 구현되었다.

이 보드 에뮬레이터는 Java Card 2.1.1 API를 가지고 있으며 ARM7TDMI 32-bit microprocessor를 가지고 있다.

또 본 구현에서 사용되는 AES, SHA-1, AX+B mod G모듈을 native method로써 Card Operating System에




〔그림 4〕 자바카드 계층도

가지고 있다.

[그림 4]에서 보여주는 것은 자바카드의 계층도로써 본 구현물이 위치하는 곳을 나타내준다.

3GPP와 3GPP2의 f함수들과 그것들을 구성하는 암호함수들은 Industry-specific API 부분에 위치하고 이것들을 이용하여 USIM과 UIM애플릿이 동작하게 된다.

## 3.1 USIM 애플리케이션 구현

3GPP의 인증 메커니즘인 MILENAGE는 AES에 기반을 두고 있다. 각각의 f함수들은 2번씩 AES를 거쳐야 얻어진다. 이외에는 OPc와 XOR이나 rotation정도 이기 때문에 AES를 제외하고서는 무시할 정도의 수행시간을 가진다. 따라서 MILENAGE의 수행속도는 AES에 큰 영향을 받는다고 할 수 있다. 이러한 이유로 AES를 패키지안에서 구현하고 JCOP에서 테스트한후 에뮬레이터 보드로 올리는 프로그램은 COS상의 native함수를 호출하여 속도를 향상시켰다.

아직까지는 8bit CPU스마트카드가 보편적이고 개발툴도 8bit(byte) 혹은 16bit(short)를 기준으로 만들어져있기 때문에 애플릿 소스는 8bit 즉 byte단위로 프로그래밍하였고 Java Card API를 사용하는 경우 short 입력을 받는 부분만 short로 구현하였다.

애플리케이션은 애플릿파일과 USIM애플리케이션용 API파일로 이루어져있다.

먼저 애플릿 파일의 역할은 ME로부터 받은 APDU로부터 INS를 추출하여 해당하는 명령을 수행하고 인증 과정을 처리하기 위하여 MAC, RES, CK, IK, AK등 데이터를 임시 저장할 필드부와 install, process, MAC verification, SQN verification 등의 메소드부를 포함하고 있다.

API파일은 f함수들을 포함하는 milenageFs.java가 구현되어 있고 에뮬레이터로 테스트할 때는 속도를 향상시키기 위해 AES를 COS상에서 구현된 것을 호출하여 사용하였다.

### 3.1.1 변환 메소드

3GPP에서 GSM으로의 인자 변환 메소드를 포함하였다. c2메소드의 경우 3G의 RES를 2G의 SRES로 변경시키는 메소드인데 RES길이는 3G에서 32~128bit로 operator에게 남겨놓은 사항이다. 여기서는 3G의 f2의 RES로 64bit를 사용하였고 SRES로 변경하기 위해 절반을 접어 사용한다.

USIM이 RAND만을 전송받았을 경우 c2, c3를 구한후 결과값 SRES와 $K_C$를 한 데이터 안에 넣어 ME에 보내준다.[3]

### 3.1.2 MAC 확인

ME로부터 USIM으로 오는 APDU의 data부는 아래와 같은 구조를 가지고 전달이 된다.

이 data중에서 RAND, AUTN을 가지고 MILENAGE f1함수를 통해 XMAC값을 구해주고 XMAC==MAC일

〔표 1〕 인증 명령 파라미터

| byte(s) | 설명 | 길이 |
|---|---|---|
| 1 | RAND길이 | 1 |
| 2~17 | RAND | 16 |
| 18 | AUTN길이* | 1 |
| 19~34 | AUTN* | 16 |
| * 3G security context시에 존재 | | |

〔표 2〕 인증 응답 파라미터

| byte(s) | 설 명 | 길이 |
|---|---|---|
| 1 | 3G 인증성공시 tag "DB" | 1 |
| 2 | RES길이 | 1 |
| 3~10 | RES | 8 |
| 11 | CK길이 | 1 |
| 12~27 | CK | 16 |
| 28 | IK길이 | 1 |
| 29~44 | IK | 16 |
| 45 | $K_C$길이* | 1 |
| 52 | $K_C$* | 8 |
| *USIM이 conversion함수로 $K_C$를 계산할수 있을때 | | |

경우 USIM은 RES, CK, IK 그리고 $K_C$를 ME에게 전송하여 준다.[5]

구현시에는 두 가지 방법으로 구현하였는데 MAC 확인이 끝난 후 RES를 전송하고 CK, IK, $K_C$를 개별적으로 ME가 USIM으로부터 가져올수 있게 메소드를 구현하였고 또 표준에 맞추어 RES, CK, IK 그리고 $K_C$를 한꺼번에 전송하도록 구현하였다. 이는 ME에게 한번에 보낼수 있는 데이터양이 한정되어 있을 경우를 위해 별도의 INS를 통해 RES이외의 값, CK, IK, $K_C$를 가져오게 한 것이다.

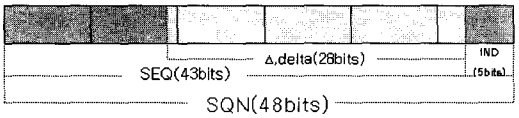
[5]에서 정의한 response안의 data에 들어갈 형식은 [표 2]와 같다.

### 3.1.3 SQN 확인

3GPP에서는 SQN의 재사용을 막기 위해 USIM에서는 이전에 받았던 SQN을 IND에 따라 저장시킨다. 이때 저장시키는 공간 크기에 대해서는 정해진 것은 없으나 32개를 예로 들고 있다.[3] 따라서 이번 구현에서도 32개의 byte array를 애플릿에 저장시키도록 하였다. 그리고 검증에서 필요한 f함수들은 milenageFs 객체를 생성하여 구현하였다.

SQN은 43bit의 SEQ와 5bits의 IND로 나누어져있고 SEQ중 일부 비트를 선택해 SQN range check을 위한 delta값으로 사용하였다. 본 구현에서는 표준에서 SEQ, IND 그리고 delta에 대해 [그림 5] 같이 권고 혹은 예시로 든 길이를 사용하였다.

SQN 확인과정시 delta이상의 bit에서 차이가 존재할시에 재동기과정으로 넘어가게 된다. 재동기시에는 아래와 같은 포멧으로 ME에게 데이터를 보내주면 된다. AUTS에는 AUTN과 달리 AMF가 포함되지 않는다. 재동기시에는 AMF를 모두 0으로 셋팅한 상태로 f1*함수를 동작시켜주고 따라서 의미 없는 AMF




〔그림 5〕 SQN구조

〔표 3〕 AUTS 동기화 응답 파라미터

| byte(s) | 설명 | 길이 |
|---|---|---|
| 1 | 동기화실패 tag "DC" | 1 |
| 2 | AUTS길이 | 1 |
| 16 | AUTS | 14 |

은 전송하지 않는다.[3]

재동기시에 ME로 전송하는 응답데이터는 [표 3]과 같은 형식이다.[5]

### 3.1.4 기타

USIM에서 저장하고 관리해야할 다른 요소로는 IMSI (International Mobile Subscriber Identity), TMSI(Temporary Mobile Subscriber Identity), Threshold, START 등이 있다. 이 IMSI, TMSI, Threshold, START등은 EF(Elementary File)에 저장되어 read, update등의 메소드를 통해 관리된다.

각 파일들은 파일 번호에 의해 접근이 가능하고 MS의 로밍시나 암호화 사용량에 따라 변경될 필요가 있고 이러한 명령들을 ME로부터 USIM이 받아 처리하여야한다.

OPc나 SQN의 경우 저장 EF가 정의 되어있지 않으므로 별도의 array에 저장하는 방식으로 구현되었다. OPc의 경우 초기에 저장하고 USIM안에서만 활용하기 때문에 다른 값들과 달리 getOPc()와 같은 ME가 USIM안에 저장된 OPc를 가져오는 메소드는 구현하지 않았다.

## 3.2 UIM 애플리케이션 구현

3GPP2의 인증 메커니즘을 구현하기 위해 두 부분으로 기능을 구분하였다.

첫 번째는 애플릿으로 구현이 된 부분으로 ME로부터 받은 RAND, AUTN을 가지고 f함수들을 실행시켜 MAC값 확인, SQN 확인 하는 부분이고 두번째는 클래스구현이 된 부분으로 f함수들에서 사용되는 SHA-1 , AX+B mod G 그리고 이 두개를 기본으로 운영되는 f함수들이 구현되어 있다. 3GPP와 달리 스마트카드와 ME간의 command, response형식에 대해 구체적 정의가 없기 때문에 가장 간단한 형태로 구현하였다. 또한 SQN의 운영이나 구조에 대한 정의 역시 이루어져있지 않기 때문에 SQN은 0부터 1씩 증가하는 형태(3GPP에서의 not time based 방식)로 가정하고 AK를 사용하여 XOR된 상태에서 UIM으로 전달되는 것을 가정하였다. 또 SQN은 array로 관리되지 않고 SQN확인이 성공된 후 무조건 SQN을 갱신하는 형태로 구현되었다.

### 3.2.1 SHA-1

3GPP2의 SHA-1은 일반적인 SHA-1과 비교하여

다르다. 즉 사용자 인증키(128bits)를 Initial Vector (160bits)와 Leftmost로 XOR하는 HMAC형태이다. 일반적인 SHA-1의 경우 고정 Initial Vector값을 사용하기 때문에 메시지 입력만을 받지만 여기서 사용되는 SHA-1에는 메시지와 사용자 인증키를 입력받아야 한다.

일반적인 SHA-1의 경우 임의의 길이의 메시지에 대한 해쉬값을 계산하여주지만 3GPP2에서 사용되는 SHA-1은 512bits의 입력만을 받아 해쉬값을 계산한다. 이때 512bits의 입력뒤에 padding을 하지 않고 해쉬값을 계산한다.

### 3.2.2 다항식 연산

여기서 사용된 다항식 연산은 덧셈과 곱셈 그리고 modular연산이다. 덧셈의 경우 byte단위의 XOR로 구현이 된다.

곱셈의 경우 여러 가지 방법으로 구현 가능하다. 일반적인 shift-and-add field multiplication 이외에 right-to-left comb method 와 left-to-right comb method 그리고 Left-to-right comb method with windows of width =4 등이 있다.[6] 이중 right-to-left comb method와 left-to-right comb method가 shift-and-add 방식보다 빠르다. 그리고 Left-to-right comb method with windows of width=4가 넷중에 제일 빠르다.[6]

하지만 LR comb method with windows of width = 4는 20bytes array 곱하기 20bytes array 시에 $2^4 \times 20$ bytes의 저장 공간이 필요하고 사전계산이 필요하다. 따라서 본 구현에서는 적은 메모리 공간을 사용하는 프로그램이고 multiplication이 각 f함수마다 단 1번 이루어지기 때문에 속도상의 월등함에도 택하지 않았다.

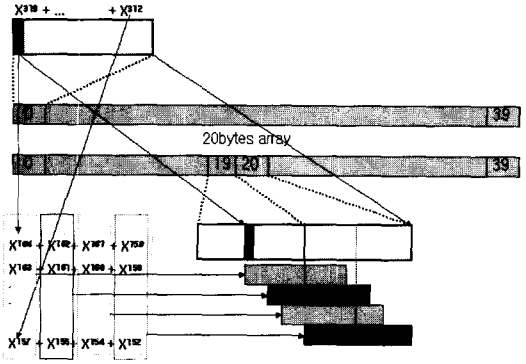
따라서 별도의 저장 공간을 필요로 하지 않는 세 가지 알고리즘으로 shift&add와 LR comb method 그리고 RL comb method가 있고 앞에서 언급한 것처럼

```
Input: Binary polynomials a(x) and b(x) of degree at most m-1
Ouput: c(x)=a(x) * b(x)
1.C←0
2.For k from 31 downto 0 do
  2.1 For j from 0 to t-1 do
    If the kth bit of A[j] is 1 then add B to c[j]
  2.2 If k≠0 then C←C * x
3. Return(C).
```

〔그림 6〕 Left-to-Right comb method

〔표 4〕 알고리즘별 소요시간 비교(10만번 수행,msec)

| 시행횟수 | 1회 | 2회 | 3회 | 4회 | 5회 |
|---|---|---|---|---|---|
| shift&add | 12,687 | 12,797 | 12,656 | 12,688 | 12,687 |
| RL comb | 6,750 | 6,766 | 6,765 | 6,718 | 6,796 |
| LR comb | 6,578 | 6,687 | 6,750 | 6,657 | 6,734 |




〔그림 7〕 modular reduction

[6]에서의 실험 결과는 RL comb method의 수행속도가 가장 빠른 것으로 나와 있었다. 이에 8bit구현과 자바 구현시 다른 결과가 있을 수 있어 세가지 알고리즘에 대하여 [표 4]와 같이 각각 10만번씩 수행을 시키고 이런 시행을 5번을 거친 결과 LR comb method가 빠른 결과를 보였고 이 방법으로 구현하였다.

Modular reduction의 경우 one word at a time 방식[6]을 변형한 one byte at a time방식으로 구현하였다.

3GPP2에서 f함수 계산 시 AX+B mod G 과정이 있고 여기서 modular연산이 필요하다. G는 $T^{160}+T^5+T^3+T^2+1$로 정의 되어 있다.

위의 [그림 7]에서 20bytes의 byte array 두 개중 첫 번째 것이 reduction해야할 입력 byte array이고 두 번째 byte array가 reduction된 후의 출력 byte array 이다. 입력 byte array의 0번째 byte는 다항식으로 따지면 $X^{319}+X^{318}+X^{317}+X^{316}+X^{315}+X^{314}+X^{313}+X^{312}$ 이다.

$G=T^{160}+T^5+T^3+T^2+1$ 이므로 $X^{319}$의 경우 $X^{164}+X^{162}+X^{161}+X^{159}$ 로 reduction된다. 따라서 $X^{319}+X^{318}+X^{317}+X^{316}+X^{315}+X^{314}+X^{313}+X^{312}$의 경우, 즉 입력의 0번째 byte를 위의 그림처럼 reduction시킨 후 나열하면 8행 4열이 된다. 이때 열로 선택하면 0번째 byte와 같고 이것은 두 번째 byte array에 19번째와 20번째 byte에 걸쳐 XOR된다.

즉 다음 [그림 8]과 같은 구현이 된다.

```
byte[] reducedC=new byte[20];
byte Tm;
byte[] mod(byte[] C){
  for(byte i=0;i< 20;i++){
    Tm=C[i];
    C[i+19]=(byte)(C[i+19] ^ ((Tm>>>3)&0x1f)
     ^ ((Tm>>>5)&0x07) ^ ((Tm>>>6)&0x03));
    C[i+20]=(byte)(C[i+20] ^ ((Tm<<5)
    &0xe0) ^ ((Tm<<3) &0xf8)
     ^ ((Tm<<2) &0xfc) ^ Tm);
     }
     for(byte i=0;i<20;i++){
        reducedC[i]=C[20+i];
     }
     return reducedC;
  }
```

〔그림 8〕 modular reduction in 3GPP2

## Ⅳ. 표준에 정의된 불분명한 구조에 대한 제안

위 앞에서 USIM인증 설명시 SQN관리를 위해서는 SQN array 32개를 저장하여야 한다고 했다. 이때 SQN은 지속적으로 USIM에 저장이 되어 AuC가 생성한 인증벡터로부터 추출한 SQN과 비교되고 갱신된다. USIM은 이러한 지속적인 저장이 필요한 데이터들을 EF파일들로 정의하여 저장토록하고 있다. 하지만 EF 파일들을 정의한 [5]를 보면 SQN을 저장하기 위한 EF가 정의되어 있지 않다. 따라서 SQN은 임시 array로 저장이 되고 이것은 USIM의 전원이 끊겨 reset과정을 거치면 초기값으로 바뀌거나 데이터가 빈 상태가 된다. 이로 인해 빈번한 재동기과정이 발생하게 된다. 그러므로 SQN의 저장을 위한 EF가 정의되어야하고 이 $EF_{SQN}$에 대해 access condition을 부여한 형태를 [그림 9]와 같이 나타낼 수 있다. 표현 방법은 [5]의 방법을 준용하였으나 identifier나 SFI에 대해서는 기존의 EF와 중복되지 않게 정의 되어야 할 것이다. 또한 USIM애플리케이션에 의해 SQN이 지속적으로 읽기와 갱신이 되어야하므로 READ와 UPDATE는 PIN에 의해 모두 가능하여야 할 것이다.

Identifier: 정해야함 Structure: transparent Mandatory
SFI: 정해야함
File size: 6n+1(n≥1) bytes Update activity: low

Access Conditions

| | |
|---|---|
| READ | PIN |
| UPDATE | PIN |
| DEACTIVATE | ADM |
| ACTIVATE | ADM |

| Bytes | Description | M/O | Length |
|---|---|---|---|
| 1 | Size of SQN array | M | 1 byte |
| 2 to 7 | SQN 1 | M | 6 bytes |
| : | | | |
| 6n-4 to 6n+1 | SQN n | O | 6 bytes |

〔그림 9〕 $EF_{SQN}$에 대한 정의

SQN이외에도 K나 OPc의 경우도 [5]에 정의되어 있지 않다. 이 두 가지 값들은 갱신하지 않을 경우 고정값으로 구현하면 EF를 필요로 하지 않으나 갱신하고자 할 때는 역시 EF가 필요하다. 이 두가지 EF는 SQN과 달리 UPDATE를 사용자에 의해서가 아닌 외부 administrator를 통해서 변경하여야 하기 때문에 ADM으로 access condition을 정의해야한다.

## Ⅴ. 실험과 결과

실험은 [7]에서 제시된 test vector값을 사용하여 이루어졌다. 실험환경은 JCOP IDE version 0.98에서 확인한뒤 실제 카드와 동일한 환경을 제공하는 애뮬레이터 보드에서 검사하였다.

### 5.1 3GPP의 실험결과

다음 [그림 10]은 IBM java card development tool인 JCOP의 shell상에서 ME에서 USIM으로 command를 보내고 USIM에서 ME로 response를 보낸 결과이다.

```
cm> /send 00880081221023553cbe9837a89d218ae64dae47bf351055f328b43577b9b94a9ffac354dfafb335
 -> 00 88 00 81 22 10 23 55 3C BE 98 37 A8 9D 21 8A    ....".#U<..7..!
    E6 4D AE 47 BF 35 10 55 F3 28 B4 35 77 B9 B9 4A    M.G.5.U.(.5w..J
    9F FA C3 54 DF AF B3 35                            ...T...5
 (31 msec)
 <- DB 08 A5 42 11 D5 E3 BA 50 BF 10 B4 0B A9 A3 C5    ...B....P.......
    8B 2A 05 BB F0 D9 87 B2 1B F8 CB 10 F7 69 BC D7    .*...........i..
    51 04 46 04 12 76 72 71 1C 6D 34 41 08 EA E4 BE    Q.F..vrq.m4A....
    82 3A F9 A0 8B 90 00                               .:.....
Status: No Error
```

〔그림 10〕 올바른 MAC에 대한 command/response

다음 [그림 11]은 위의 결과를 설명한 것으로 APDU command, response를 분석한 것이다.

```
Command
CLA: 00 INS: 88 P1:00 P2:81 LC:22
DATA:
10 (Length of RAND) 23 55 3C BE 98 37 A8 9D 21 8A E6 4D AE 47 BF 35 (RAND)
10 (Length of AUTN) 55 F3 28 B4 35 77 B9 B9 4A 9F FA C3 54 DF AF B3 (AUTN)
LE:35
Response
DATA DB (successful 3G authentication tag) 08 (Length of RES) A5 42 11 D5 E3 BA 50 BF (RES)
10 (Length of CK) B4 0B A9 A3 C5 8B 2A 05 BB F0 D9 87 B2 1B F8 CB (CK)
10 (Length of IK) F7 69 BC D7 51 04 46 04 12 76 72 71 1C 6D 34 41 (IK)
08 (Length of Kc) EA E4 BE 82 3A F9 A0 8B (Kc)
90 00 (status ok)
```

〔그림 11〕 올바른 MAC에 대한 command/response에 대한 설명

그 이외에 MAC을 고의로 틀린 값으로 하여 command를 전송하였을 때 Authentication failure respones, RAND만 받았을 경우 SRES와 Kc response, SQN가 특정범위 밖에 있을 때 재동기를 위해 보내는 AUTS response, 그리고 SQN이 허용범위 내에 속해 있을 때 정상 인증등을 확인 할 수 있었다.

### 5.2 3GPP2의 실험결과

앞에서 설명한 것처럼 3GPP2에서는 UIM과 ME간의 command, response형태에 대한 언급이 없기 때문에 가장 간단한 형태로 주고 받도록 구현하였다.

다음 [그림 12]는 JCOP의 shell상에서 ME에서 UIM으로 command를 보내고 UIM에서 ME로 response를 보낸 결과이다. 올바른 MAC이 포함된 AUTN이 UIM으로 전달되었을 때 응답이다.

```
cm> /send 00020000214b052b20e2a06c8ff700da512b4e111e594cc7c17c0700016abdc4da73c61b8d10
 => 00 02 00 00 21 4B 05 2B 20 E2 A0 6C 8F F7 00 DA    ....!K. + ..l....
    51 2B 4E 11 1E 59 4C C7 C1 7C 07 00 01 8A BD C4    Q+N..YL..|....j..
    DA 73 C6 1B 8D 10                                  .s....
(282 msec)
 <= D8 2E 28 2A DC 13 C0 F1 68 65 66 33 9B F2 7E B6    ..(*....hef3..~.
    90 00                                              ..
Status: No Error
```

〔그림 12〕 올바른 MAC에 대한 command/response

아래 [그림 13]은 위의 결과를 설명한 것으로 APDU command, response를 분석한 것이다.

3GPP2에서는 RES만을 response로 보내도록 구현한 점이 차이이다.

```
Command
CLA: 00 INS: 02 P1: 00 P2: 00 LC: 21
DATA:
4B 05 2B 20 E2 A0 6C 8F F7 00 DA 51 2B 4E 11 1E (RAND)
59 4C C7 C1 7C 07 00 01 8A BD C4 DA 73 C6 1B 8D (AUTN)
LE: 10
Response
DATA: D8 2E 28 2A DC 13 C0 F1 68 65 66 33 9B F2 7E B6 (RES)
90 00 (status ok)
```

〔그림 13〕 올바른 MAC에 대한 command/response에 대한 설명

마찬가지로 3GPP2에 대해서도 MAC을 틀린 값으로 설정하여 command를 전송하였을 때 UIM의 Authentication failure response확인, SQN이 특정범위 밖에 있을 때 재동기를 위해 보내는 AUTS 확인, 차이는 있으나 허용범위 내에 있을 때의 정상 인증과정을 살펴 볼수 있었다.

## Ⅵ. 결 론

3세대 이동통신 서비스에 있어서 3GPP의 경우 2세대에서 문제점으로 지적되었던 것들이 개선되었고 3GPP2역시 3GPP와 비슷한 수준의 보안 메커니즘을 제공하기 위해 표준을 만들고 있다.

이에 본 논문에서는 서비스에 앞서 ME에 장착되는 스마트 카드와 동일한 환경에서 인증 메커니즘을 구현하고 표준과 비교하였다. 3GPP의 경우 사업자 결정 사항으로 남겨놓은 부분들이 많이 있고 3GPP2의 경우 3GPP에 비해 표준진행이 느려 미결사항이 있지만 미리 그 부분들을 설정하고 확인함으로써 확실한 검증결과를 도출할 수 있었다. 기존의 검증들이 시뮬레이션 프로그램 개발을 통한 검증인데 반해 이번 구현은 실제 스마트 카드의 하드웨어 조건에서 구현한 실제 USIM, UIM 애플리케이션이라는데 구현결과의 의미가 있다. 또한 표준에 명확히 정의되어 있지 않아 정상적인 SQN 관리가 불가능함을 지적하였고, 이에 따라 규정이 부족한 부분에 대한 정의를 하였다.

앞으로의 후속 연구로서 3GPP2의 인증 메커니즘 표준이 완성되면 UIM부터 AuC까지의 과정을 연결하여 구현해보고 안전한 키 분배와 관리 및 상호인증과정에 대한 연구와 3GPP와 3GPP2의 인증 및 키 일치 구조의 상호연계에 대한 연구가 필요할 것이다.

## 참 고 문 헌

[1] K. Boman, G. Horn, P. Howard, and V. Niemi, "UMTS Security", *issue of IEE Electronics & Communication Engineering Journal*, October 2002.

[2] http://java.sun.com/products/javacard/index.html #about.

[3] 3GPP TS 33.102, Security Architecture.

[4] 3GPP2 S.S0055, "Enhanced Cryptographic Algorithms".

[5] 3GPP TS 31.102, Characteristic of USIM.

[6] Darrel Hankerson, Julio Lopez Hernandez, and Alfred Menezes, "Software Implementation of Elliptic Curve Cryptography over binary fields".

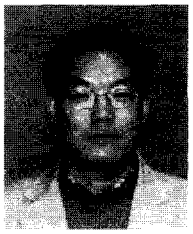
[7] 3GPP TS 35.207, An example algorithm set for the 3GPP authentication and key generation function f1,f1*,f2,f3,f4,f5 and f5*; Document 3:Implementors' Test Data.

## 〈著者紹介〉

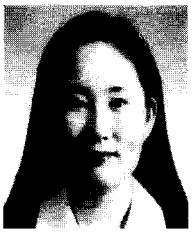
**조 승 환 (Seung-hwan Jo) 정회원**
1997년 2월 : 고려대학교 수학교육과 이수
2001년 8월 : 고려대학교 컴퓨터학과 졸업
2003년 8월 : 고려대학교 정보보호대학원 정보보호학과 졸업(공학석사)
2003년 9월~현재 : 한국전자통신연구원 컴퓨터S/W연구소 S/W개발도구연구팀 연구원
<관심분야> 이동통신보안, 무선랜 보안, 센서네트워크

**전 성 익 (Sung-ik Jun) 정회원**
1985년 2월 : 중앙대학교 전산학과 졸업 (공학사)
1987년 2월 : 중앙대학교 전산학과 졸업 (공학석사)
1987년~현재 : 한국전자통신연구원 정보보호연구본부 책임연구원
2003년~현재 : 한국전자통신연구원 정보보호연구본부 IC카드연구팀 팀장
<관심분야> IC Card, 이동통신 보안기술

**이 정 우 (Jung-woo Lee) 정회원**
1999년 2월 : 성균관대학교 정보공학과 졸업 (공학사)
2001년 2월 : 성균관대학교 전기전자컴퓨터공학과 졸업 (공학석사)
2001년 1월~현재 : 한국전자통신연구원 정보보호연구본부 IC카드연구팀 연구원
<관심분야> IC Card, Network Security, DRM



**이 옥 연 (Ok-yeon Yi) 정회원**
1988년 2월 : 고려대학교 수학과 (이학사)
1990년 2월 : 고려대학교 수학과 (이학석사)
1996년 8월 : 미국 University of Kentucky 이학박사
2001년 9월~현재 : 국민대학교 수학과 교수
<관심분야> 이동통신 보안기술, 무선랜 보안, IC Card보안



**한 진 희 (Jin-hui Han) 정회원**
1997년 2월 : 숭실대학교 정보통신공학과 졸업 (공학사)
1999년 2월 : 광주과학기술원 정보통신공학과 졸업 (공학석사)
1999년 6월~현재 : 한국전자통신연구원 정보보호연구본부 IC카드연구팀 연구원
<관심분야> IC Card, Internet Security, Biometry

**이 세 광 (Se-kwang Lee) 학생회원**
2001년 2월 : 고려대학교 수학과 졸업 (이학사)
2002년 3월~현재 : 고려대학교 정보보호대학원 석사과정
<관심분야> 이동통신보안, 무선랜보안