제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

USIM 에서의 3GPP 상호인증 프로토콜 구현

김건우, 이항록
한국전자통신연구원
e-mail : wootopian@etri.re.kr

Implementation of 3GPP AKA protocol in USIM

Keonwoo Kim, Hangrok Lee
ETRI

요 약

2 세대 GSM 통신에서는 SIM 을 이용하여 망이 사용자를 인식하는 단방향 인증 서비스만 제공한다. 하지만, 3 세대 비동기 방식의 IMT-2000 에서는 2 세대 인증 메커니즘의 취약성 보완 및 양방향 인증을 위해서 새로운 인증 프로토콜을 설계하였다. 이에 3GPP 는 인증 메커니즘의 핵심 알고리즘으로 Milenage 를 사용할 것을 권고하고 사업자는 이를 사용하여 인증 서비스를 제공할 것으로 예상된다. 본 논문은 JCOS 기반의 USIM 플랫폼에서 상호인증 기능을 수행하는 자바 애플릿 구현에 관한 것이다. 결과에 관해 에뮬레이터 환경에서 테스트하여 표준문서에서 제시하는 값과 동일함을 확인하였다.

1. 서론

3GPP 방식의 IMT-2000 시스템은 고속의 멀티미디어 서비스 제공 및 글로벌 로밍을 특징으로 하고 이러한 이동통신 환경의 변화는 정보보호에 대한 새로운 대책를 요구하고 있고 정보보호 기술도 새로운 패러다임 변화에 맞추어 발전해야 한다. 이에 3GPP(3rd Generation Partnership Project)는 GSM 방식보다 더 강력하고 안전한 정보보호 메커니즘을 개발하였는데 가입자와 네트워크간의 상호인증, 무선구간에서의 데이터 보호를 위해 암호화와 무결성 제공이 그것이다. 상호인증과 키 일치 메커니즘인 AKA(Authentication and Key Agreement) 를 위해서 3GPP 는 Milenage 를 사용할 것을 권고하고 있으며, 안전성에 관해서는 이미 여러 관점에서 검증이 되었다.

현재 국내의 W-CDMA 방식 IMT-2000 은 년내에 서비스 실시를 목표로 시스템을 구축하고 있다. 하지만, 지금껏 국내의 이동통신 시스템이 카드 형태의 가입자 인증 모듈을 사용하지 않았고 아직까지 3 세대용 USIM(User Subscribe Identity Module)이 국내에서 개발된 적이 없다. 그래서, 상호인증 및 암호화/무결성 키 생성 메커니즘을 실제 USIM 또는 USIM 과 유사한 에뮬레이터 환경에서 기능을 검증해서 실제 상용망에 적용시킬 필요가 있다.

본 논문의 2 장에서는 3GPP 인증 메커니즘, 특히, USIM 에서의 인증 기능에 관해서 간단히 설명하고, 3 장에서는 자바카드 기반 USIM 플랫폼에서 상호인증 및 키 생성 구현에 관해서 기술한다. 논문에서는 카드 플랫폼과 운영체제인 JCOS(Java Card Operating System)에 관해서는 구체적으로 언급하지 않고, 플랫폼 상위에서 어플리케이션 형태로 동작하는 상호인증 애플릿 설계와 구현에 관해서 주로 논한다. 마지막으로 4 장에서 결론을 내린다.

2. 3GPP 인증 메커니즘

3GPP 에서의 가입자 인증은 USIM 과 AuC 가 동일한 비밀키를 소유하고 있음을 증명함으로써 이루어지는데, VLR 은 서비스를 제공하기 전에 가입자를 인증하기 위해 HLR/AuC 에 의해서 생성된 AV(Authentication Vector)를 사용한다. MAC, RES 비교와 SQN 범위 검증등의 절차를 거쳐서 USIM 과 AuC 가 서로 상호인증이 성공했다고 판단하면 VLR 과 MS 는 보안모드에서 사용하는 CK(Ciphering Key)와 IK(Integrity Key)를 공유한다.

그림 1 은 3GPP 보안 구조를 보여주고 USIM 에서 AKA 를 수행하기 위해서 필요한 Milenage 에 사용되는 암호학적 함수를 분류하였다. 이들의 입출력으로 사용되는 값은 표 1 에 나타내었다.

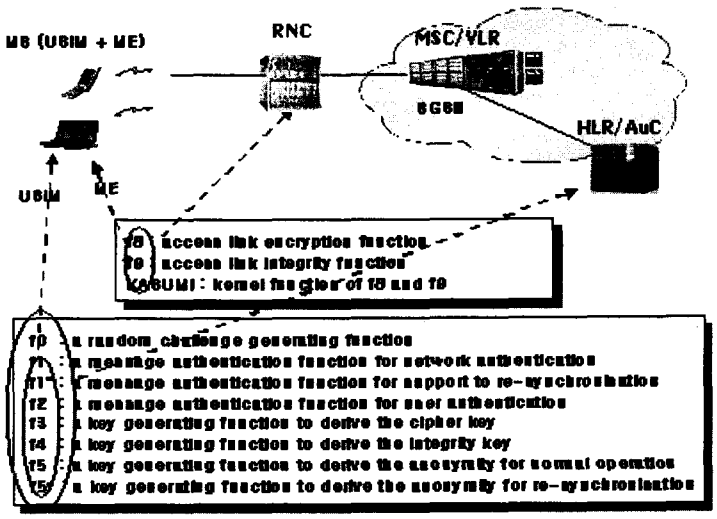

그림 1. 3GPP 보안구조와 USIM 에서 Milenage 에 사용되는 함수

표 1. f 함수의 입력과 출력

함수	입력	출력
f1	K, SQN, RAND, AMF	MAC(XMAC)
f1*	K, SQN, RAND, AMF	MAC*(XMAC*)
f2	K, RAND	RES(XRES)
f3	K, RAND	CK
f4	K, RAND	IK
f5	K, RAND	AK
f5*	K, RAND	AK*

Milenage 알고리즘은 핵심함수로 128 비트 블록 암호인 AES 를 사용하는데, 한번의 키 스케쥴과 7 번의 Rijndael 암호화가 반복과정으로 구성되어 있다.

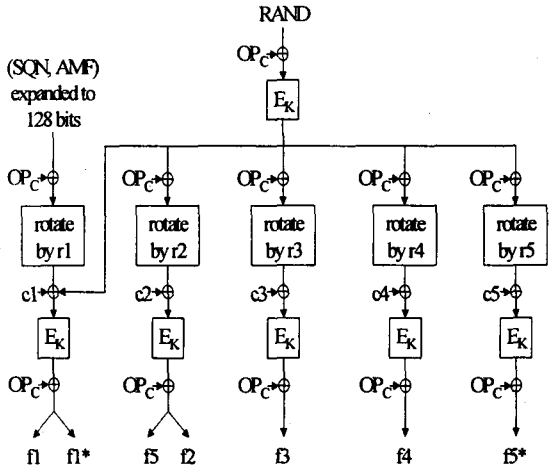

그림 2. f 함수

뿐만 아니라, USIM 은 GSM 과의 로밍을 위하여 f 함수의 결과를 conversion 함수인 c2, c3 를 이용하여 GSM 용 인증값(SRES)과 암호화 키(Kc)를 생성한다.

3. USIM 플랫폼에서 상호인증 애플릿 구현

상호인증 기능을 가진 자바카드 애플릿 구현을 위해서는 COS(Chip Operating System), JCVM(Java Card Virtual Machine), 그리고, JCAPI(Java Card Application Programming Interface)로 구성되는 USIM 카드 플랫폼을 필요로 한다. JCOS 는 USIM 카드를 위한 다양한 명령들을 분석하여 적절한 기능을 수행하는 카드 제어 및 운영 소프트웨어를 의미하고, JCPAI 는 실행 환경에서 자바 카드 애플릿 프로그램이 참조하는 패키지 형태의 클래스 파일을 의미한다.

망과 단말기에 삽입된 USIM 사이에서의 상호인증을 위해서는 실제 서비스되는 상용망을 이용하여야 하고 위에서 언급한 플랫폼 구조를 가진 USIM 카드에서 어플리케이션을 개발하는 것이 바람직하다. 하지만, 망 이용의 어려움과 카드제작의 반도체 공정상의 문제등 현실적인 어려움으로 JCOS 기반 자바카드 플랫폼을 구현한 에뮬레이터 보드로 대체하여 애플릿에 대한 테스트 스크립트를 작성하여 기능을 검증하였다. 에뮬레이터 등의 구체적인 플랫폼 환경은 본 논문에서는 언급하지 않는다.

상호인증과 글로벌 로밍 기능을 가지는 USIM 애플릿은 3GPP TS 33.102, 3GPP TS 35.205, 3GPP TS 35.206, 3GPP TS 35.207, 3GPP TS 35.208, 3GPP TS 35.909, 3GPP TS 31.102, ETSI TS 102.221, ISO 7816-4 표준문서를 참고하여 기능을 설계하고 구현하였다.

상호인증이 USIM 에서 제대로 수행되기 위해서는

USIM management 절차, 보안관련 절차, 가입관련 절차, USAT 관련 절차가 모두 인증 애플릿과 연계가 되어야 한다. 우리는 profile exchange, 사용되는 보안 파라미터 요구, 사용자 신원 요구, RRC 연결설정 및 해제 등의 보안관련 절차를 USIM 인증 애플릿에 구현하였다. 한편, USIM 은 EF_{DIR} 에 의해서 USIM 인증 애플릿이 선택되어야 하고, SELECT command 를 사용한 파일접근(read/update)이 가능해야 한다. 또한, SW 를 통한 카드상태 체크와 USIM 초기화를 고려해야한다. 초기화 과정에서 단말기는 emergency call code(reading/updating EF_{ECC})와 Language indication(reading/updating EF_{LI})를 요구하고, user verification 절차를 수행해서 검증이 실패하면 USIM 초기화를 중지한다. 이어서 administrative information(reading EF_{AD}), USIM service table(reading EF_{UST}), Enabled Service table(reading EF_{EST})를 요구한다. 이러한 절차가 모두 수행되어 USIM 초기화가 성공적으로 완료되고 나서야 단말기는 3G 세션을 준비하고 특정 STATUS command 를 USIM 에게 보냄으로서 이것을 통보한다.

인증과 로밍 구현에 직접적으로 필요한 EF 파일은 EF_{IMSI}, EF_{Keys}, EF_{PSKeys}, EF_{LOCI}, EF_{PSLOCI}, $EF_{START-HFN}$, $EF_{Threshold}$, EF_{Kc}, EF_{KcGPRS} 이고, 인증 애플릿은 파일시스템 API 인터페이스와 연동이 되어야 한다. 파일구조에 따라 Binary 를 사용할 것인지 Record 를 사용할 것인지가 달라지지만, 상호인증 애플릿에서는 transparent EF 만 사용하면 되기 때문에 READ RECORD, UPDATE RECORD 를 구현할 필요는 없다. 한편, 표준문서에서는 가입자의 영구키인 K 의 저장형태에 관해서는 언급하지 않는데 실제 카드에 구현하기 위해서는 파일로 저장할 것인지 애플릿에 변수로 저장할 것인지를 고려해야 하고, 우리는 이 값의 중요성을 반영하여 access condition 을 적용한 EF 로 저장하는 방법을 채택하였다. 이것은 인증 애플릿외의 다른 애플릿에서 K 에 접근하는 것을 막기 위해 AID 확인과정을 추가하고, 파일의 중요성을 고려하여 PIN 확인 과정을 거치는 것이다.

위의 조건을 만족하는 USIM management 환경이 구현된 상태에서 상호인증 애플릿은 인증에 필요한 파일을 read/update 할 수 있어야 한다. 그리고, AUTHENTICATE command 에 응답하기 위해 3G security context 와 GSM security context 둘다 구현되어야 한다. READ & UPDATE Binary command, AUTHENTICATE command 와 이에 대한 response 는 ETSI TS 102.221 와 ISO 7816-4 의 Command & response APDU 구조 및 코딩규칙을 따르고, 인증/로밍 구현에 필요한 CLS, INS, P1, P2 코딩값, 그리고 SW response 도 표준값을 그대로 준수하였다.

인증 애플릿은 모든 자바카드 애플릿에 대한 base class 가 되는 javacard.freamwork.Applet 을 extend 해서 구현하는데 먼저 생성자를 호출하는 install( ) 메소드에서 애플릿 인스턴스를 생성한 다음 JCRE 에 등록한다. 애플릿의 실질적인 기능은 process(APDU apdu)에서 정의되는데, 구현하고자 하는 인증 및 로밍 기능을 위해서는 processReadBinary(apdu), processUpdateBinary(apdu), processAuthenticate(apdu) 가 필요하다. processReadBinary(apdu)와 processUpdateBinary(apdu)에서는 AKA 와 직접적인 관계는 없더라도 프로토콜이 시작되기 전이나 완료된 후의 파일을 단말이 읽고 갱신하기 위해서 필요하고, processAuthenticate(apdu)에서 실제 인증 command APDU 를 받아서 Milenage 알고리즘을 이용해서 인증벡터를 생성하고 이를 비교한 다음 response 를 단말에 보낸다. 이때 USIM 에서 벡터가 생성되는 원리는 그림 3 과 같다.

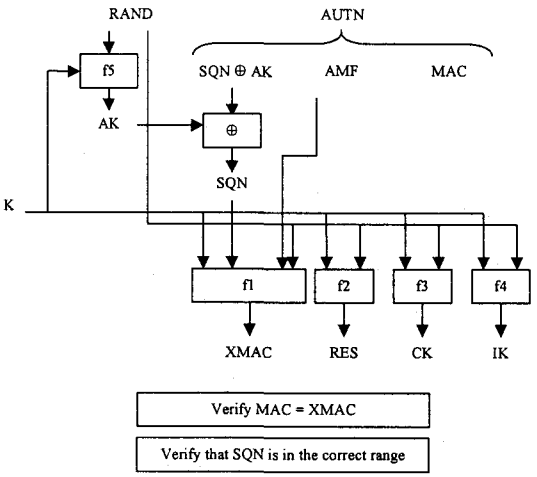

그림 3. USIM 에서 인증 기능의 수행

3G security context 인 경우 USIM 은 00 ‖ 88 ‖ 00 ‖ 81 ‖ 22 ‖ data 로 구성된 APDU 를 수신하는데 데이터는 다음과 같다.

- Length(RAND) : 1 바이트('10')
- Rand : 2nd~17th
- Length(AUTN) : Rand 다음의 한 바이트 ('10')
- SQN : 19th 부터 6 바이트;
- AMF : 25th 부터 2 바이트;
- MAC : 27th 부터 8 바이트

그러면 USIM 은 Milenage 알고리즘을 수행해서 SQN, X-MAC, RES, CK, IK 를 얻는다. X-MAC 과 MAC 을 비교하여 값이 일치하고, SQN 이 SQN_{MS} 와 비교하여 유효한 값이면 SQN_{MS} 를 SQN 값으로 갱신하고 data ‖ '90 00'을 전송한다. 이때 단말로 보내지는 데이터는 다음과 같다.

- 'DB' : 인증 성공하였음을 의미
- length(RES) : '08'
- RES : 8 바이트
- length(CK) : '10'
- CK : 16 바이트
- length(IK) :'10'
- IK : 16 바이트
- length(Kc) : '08'
- Kc : 8 바이트

한편, MAC 검증은 성공하더라도, SQN 유효성 검증

이 실패하는 경우는 재동기화 알고리즘 수행하여 AUTS 를 계산해서 단말에 Data || '90 00' 을 전송한다. 이때의 데이터는 다음과 같다.

- 'DC' : 동기화 실패를 의미
- Length(AUTS) : '0E'
- AUTS = Conc(SQN_{MS}) || MACS
- Conc(SQNMS) = SQN_{MS} xor $f5^*_K$(RAND)
- MACS = $f1^*_K$(SQN_{MS} || RAND || AMF*)

여기서, 사업자가 재동기에 사용되는 AMF* = '0000' 라고 정의하면 AMF* 를 보내지 않아도 된다.

만일 MAC 검증이 실패하는 경우는 SW1 || SW2 = '9862' 를 단말기로 보낸다.

한편, GSM security context 인 경우, USIM 은 00 || 88 || 00 || 80 || 11 || data 로 구성된 APDU 를 수신하는데 데이터는 RAND 길이를 나타내는 1 바이트와 16 바이트의 RAND 이다. USIM 은 GSM SIM 의 A3, A8 알고리즘을 포함하지 않기 때문에 f 함수와 conversion 함수를 이용하여 GSM security context 의 response 를 구한다. 이때는 Milenage 알고리즘의 f2, f3, f4 만 이용해서 XRES, CK, IK 를 구하고 Data | | '90 00' 을 전송한다. 이때의 데이터는 length(SRES)='04' || SRES || length(Kc)='08' || Kc 이다. SRES 는 c2, Kc 는 c3 함수를 이용하여 구한다.

- SRES = XRES*1 xor XRES*2 xor XRES*3 xor XRES*4
- Kc = CK1 xor CK2 xor IK1 xor IK2
- XRES* = XRES || 00,,00
- XRES* = XRES*1 xor XRES*2 xor XRES*3 xor XRES*4
- CK = CK1 || CK2
- IK = IK1 || IK2

한편 잘못된 CLA, INS, P1, P2 가 사용되면 USIM 은 각각의 에러 Exception 을 의미하는 Status Word 를 보낸다.

이렇게 구현된 상호인증 애플릿을 가진 USIM 기능을 망에서 검증하는 것은 너무 어려운 문제라서 앞에서 언급한 에뮬레이터 보드에 넣어서 시험해보았다. 망을 통해 값을 내려받을수 없으므로 표준문서에 제시된 테스트 벡터를 이용하여 스크립트를 작성해 기능을 검증하였다. 우선 인증이 성공하는 경우에 대해 RAND || AUTN 과 개인키 K, 그리고, OPc 등으로 스트립트로 구성하여 테스트 한 결과 표준문서에서 제시된 f1, f2, f3, f4, f5 와 동일함을 확인할 수 있었다. 그리고, 테스트 환경에서 MAC 실패를 시험하기 위해서는 인증이 성공하는 경우에서 사용되는 데이터나 개인키의 일부 값을 임의로 변경하여 확인하는 수 밖에 없다. 그래서, K 와 MAC 값을 바꾸어 각각 테스트 한 결과 MAC 실패로 인한 인증 실패 에러를 확인 할수 있었다. 마지막으로 Sequence number 동기 실패로 인해 재동기화 과정이 제대로 수행되는지 또 그 과정에서 망으로 전송되는 데이터가 정확한 값을 출력하는지 시험하였다. 테스트 환경에서는 망으로부터 받는 SQN 보다 큰 값을 USIM 에 미리 저장하여 재동기화 과정이 수행되도록 하여 결과가 표준문서에 제시된 f1*, f5* 값과 동일함을 확인하였다. 여기서는 사업자가 사용하는 32 개의 SQN 에 대해서 발생할 수 있는 모든 경우를 대소비교 하였다.

4. 결 론

상호인증 어플리케이션이 구현된 USIM 은 곧 서비스가 예상되는 비동기방식 IMT-2000 에서 사용자와 망과의 상호인증을 위해서 반드시 필요하다. 지금까지 우리나라에서 사용하는 CDMA 방식 이동통신 시스템과는 달리 사용자는 자신만의 키를 저장한 USIM 모듈만 가지고 어떠한 단말기도 사용할 수 있다. 이러한 환경의 변화에 따라 표준규격을 준수하는 USIM 용 카드 플랫폼을 개발해야 하고, 또한 USIM 의 기본기능인 인증 및 키 분배 메커니즘이 플랫폼과 연동하여 구현되어야 한다. 본 논문에서는 실제로 자바카드 플랫폼에서 상호인증 기능을 구현하여 이를 테스트 한 것에 관하여 설명하였다. 추후 이러한 기능을 Gemplus 3G explore 와 같은 외국의 상용 제품과 비교할 필요가 있을 것이다.

참고문헌

[1] 3GPP TS 33.102
[2] 3GPP TS 35.205
[3] 3GPP TS 35.206
[4] 3GPP TS 35.207
[5] 3GPP TS 35.208
[6] 3GPP TS 35.909
[7] 3GPP TS 31.102
[9] 3GPP TS 31.900
[9] ETSI TS 102.221
[10] ISO 7816-4
[11] Java Card 2.2 Application Programming Interface
[12] Java Card 2.2 Development Kit User's Guide