

스마트폰뱅킹을 위한 공인인증서 복사 프로토콜의 취약점 분석

신 동 오*, 강 전 일*, 양 대 현*, 이 경 희°

On the Security of Public-Key-Certificate-Relay Protocol for Smart-Phone Banking Services

DongOh Shin*, Jeonil Kang*, DaeHun Nyang*, KyungHee Lee°

요 약

최근 대다수의 국내 은행들은 스마트폰뱅킹 서비스를 제공한다. 스마트폰뱅킹을 이용하기 위해서는 PC에 보관된 공인인증서와 개인키를 스마트폰으로 복사해야 한다. 하지만 사용자가 USB 케이블을 이용해 PC에서 스마트폰으로 직접 복사하는 방법은 편의성이 떨어지기 때문에 이를 해결하기 위해 다수의 보안 솔루션 기업은 중개서버를 두어 공인인증서를 복사해주는 방법을 택하고 있다. 이 논문에서는 공인인증서 복사 프로토콜의 보안성을 분석하고 이를 토대로 한 공격을 보인다. 우리는 네트워크 도청을 통해 사용자의 공인인증서와 개인키를 추출할 수 있었다. 또한, 공인인증서를 안전하게 복사하기 위한 방법을 모색한다.

Key Words : Public-key-certificate-relay, Financial security, Smart-phone banking, Network sniffing, Decompilation

ABSTRACT

Most of banks in Korea provide smartphone banking services. To use the banking service, public key certificates with private keys, which are stored in personal computers, should be installed in smartphones. Many banks provides intermediate servers that relay certificates to smartphones over the Internet, because the transferring certificates via USB cable is inconvenient. In this paper, we analyze the certificate transfer protocol between personal computer and smartphone, and consider a possible attack based on the results of the analysis. We were successfully able to extract a public key certificate and password-protected private key from encrypted data packets. In addition, we discuss several solutions to transfer public key certificates from personal computers to smartphones safely.

I. 서 론

2009년 6월 Apple사의 iPhone 3G가 국내의 전 파인증을 받고 KT를 통해 같은 해 11월에 출시된 이후 스마트폰의 보급이 확산된 이래 2012년 4월 말 기준 스마트폰 가입자 수는 2661만 명으로, 매

월 약 4%의 증가추세를 보이고 있다¹⁾. 스마트폰과 태블릿PC(이하 스마트폰)가 보급됨에 따라 2009년 말 하나은행과 기업은행에서 최초로 iOS용 스마트폰뱅킹을 서비스하기 시작했고, 이 후 금융권에서는 인터넷뱅킹에서 제공하던 금융서비스를 스마트폰에서도 제공하기위해 스마트폰 운영체제별 전용 어플

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2012-002146)

◆ 주저자 : 인하대학교 컴퓨터정보공학과 정보보호연구실, mannershin@isrl.kr, 정희원

° 교신저자 : 수원대학교 전기공학과, khlee@suwon.ac.kr, 정희원

* 인하대학교 컴퓨터정보공학과 정보보호연구실, dreamx@isrl.kr, nyang@inha.ac.kr

논문번호 : KICS2012-06-288, 접수일자 : 2012년 6월 26일, 최종논문접수일자 : 2012년 8월 14일

리케이션을 제작하여 배포하고 있다.

인터넷뱅킹이나 스마트폰뱅킹 사용자가 계좌 이체나 금융상품 가입 등의 서비스를 이용하기 위해서는 전자문서에 사용자의 개인키로 전자서명을 해야 그 효력이 발생한다. 공개키기반시스템에서 개인키로 만드는 전자서명생성정보는 공인인증기관으로부터 전자서명생성정보를 인증 받은 자에게 유일하게 속해야 하기 때문에, 이를 인터넷뱅킹과 스마트폰뱅킹에서 모두 이용하기 위해서는 공인인증서와 개인키를 복사해서 사용해야 한다²⁾. 일반사용자에게 있어서 공인인증서와 개인키는 흔히 접하는 파일형식도 아닐뿐더러, 일반적으로 접근하지 않는 폴더에 위치해있기 때문에 파일로 만들어진 자신의 공인인증서와 개인키를 식별하기 어렵다. 이를 스마트폰에 복사하는 일은 더욱 어렵게 느껴질 것이다. 금융회사에서는 이러한 사용자의 어려움과 불편을 해소시켜주기 위해 스마트폰뱅킹을 위한 공인인증서 복사 서비스를 제공해주고 있다.

공인인증서와 함께 복사되는 개인키는 전자상거래에서 계약서를 작성하거나 신원확인 등에 쓰이는 전자서명 생성에 이용되는 비밀이다. 이 비밀이 노출된다면 공격자는 소유자 대신 전자서명을 만들 수 있는 가능성이 존재하게 되며, 이는 곧 전자문서에 대한 인증기능과 부인방지가 그 효력을 상실하게 되는 문제를 발생시킨다. 따라서 공인인증서와 개인키는 외부에 노출되지 않는 안전한 방법으로

복사되어야 하지만, 실험결과 도청한 패킷으로부터 사용자의 공인인증서와 개인키를 추출할 수 있었다.

이 논문의 2장은 주요 시중은행의 스마트폰뱅킹 서비스 제공현황을 보여주고 각 은행에서 공인인증서 복사 서비스를 위해 사용하고 있는 보안프로그램에 대한 조사결과를 보여준다. 3장에서는 인증번호기반의 인증서복사 프로그램에 대한 보안성 분석을, 4장에서는 안전하지 않은 PRNG(Pseudo Random Number Generator)를 사용하는 인증서복사 프로그램에 대한 보안성을 분석한다. 5장은 분석한 내용을 토대로 한 공격과 그 결과를 보인다. 6장에서는 인증서 복사를 안전하게 하기 위한 방법을 모색한다. 7장은 이 논문의 결론을 담는다.

II. 주요 시중은행의 스마트폰뱅킹 서비스 제공 현황

표 1은 주요 시중은행 10개사에 대해 각 스마트폰 운영체제별 스마트폰뱅킹 서비스 지원여부와 공인인증서를 복사할 때 사용하는 인증번호의 길이, 공인인증서 복사 서비스를 제공하는 프로그램의 제조사를 보여준다. 2012년 6월 현재 주요 시중은행 모두 iOS와 안드로이드 운영체제용 스마트폰뱅킹 서비스를 시행중이며, 윈도우모바일의 경우는 산업은행을 제외한 나머지 은행에서 서비스를 시행하고 있다. 블랙베리나 바다 운영체제용 스마트폰뱅킹 서

표 1. 주요 시중은행의 스마트폰뱅킹 서비스 지원 여부
Table 1. Smart-phone banking systems of major commercial banks in Korea

Bank	Smart-phone operation systems					Confirmation of resident registration number	Length of authentication number		Program developers
	iOS	Android	Windows Mobile	Black berry	Bada				
Shinhan	×	×	×			×	16	Initech	
							8		
KEB	×	×	×	×	×	×	16	Initech	
							8	yessign	
Woori	×	×	×	×	×	×	8 12	Lumensoft	
							8	yessign	
Citybank	×	×	×	×		×	8	yessign	
Hana	×	×	×	×	×	×	8	BTWorks	
IBK	×	×	×	×	×	×	8	Lumensoft	
								yessign	
Kookmin	×	×	×				12	Korea Electronic Certification Authority	
KDB	×	×				×	16	Initech	
NH	×	×	×	×		×	16	BTWorks	
							8	yssign	
SC	×	×	×			×	16	Initech	

비스는 각각 6개 은행, 5개 은행에서 서비스 중으로, 다른 운영체제에 비해 다소 낮은 서비스 시행률을 보이고 있다. 5개의 보안회사가 주요 시중은행 10개사에 공인인증서 복사서비스를 제공하고 있으며, 모두 인증번호를 기반으로 공인인증서 복사 서비스를 제공하고 있다.

공인인증서를 복사하는 방법은 주민등록번호 입력을 요구하는 방법(이하 구버전)과 주민등록번호 없이 복사하는 방법(이하 신버전)으로 나뉜다. 구버전의 경우 우선 스마트폰뱅킹 프로그램에서 임의의 인증번호를 생성한다. 사용자는 전송할 공인인증서를 선택하고 암호를 입력함으로써 공인인증서의 소유자임을 우선 검증받는다. 이후 스마트폰뱅킹 프로그램에서 생성한 인증번호와 자신의 주민등록번호를 PC의 공인인증서복사 프로그램에 입력한다. 프로그램은 사용자가 공인인증서의 소유자임을 주민등록번호를 통해 확인한 뒤 사용자가 맞을 경우에만 공인인증서와 개인키를 암호화하여 중개서버로 전송한다. 스마트폰에서는 주민등록번호를 한 번 더 입력하여 사용자가 맞음을 거듭 확인한 후 공인인증서 가져오기를 통해 중개서버로부터 공인인증서와 개인키를 복호화한 뒤 스마트폰의 지정된 저장소에 저장한다. 보안프로그램에 따라 주민등록번호를 입력하는 절차의 순서가 조금씩 다르거나, PC나 스마트폰중 어느 한쪽에서만 확인하는 등의 차이만 있을 뿐이고 나머지 절차는 대개 같다. 신버전의 경우 주민등록번호를 이용한 소유자확인과정만 없을 뿐, 다른

과정은 구버전과 동일하게 동작한다.

Initech사의 경우 구버전에서는 16자리의 인증번호를, 신버전에서는 8자리의 인증번호를 사용하는 반면, Lumensoft사의 경우 구버전에서는 8자리의 인증번호를, 신버전에서는 12자리의 인증번호를 이용하고 있다. 두 프로그램 모두 신버전에서는 주민등록번호 입력란을 없앴다. 한국인터넷진흥원의 기술규격에 따르면 사용자 인증을 위해 개인키 암호화 비밀번호 확인 기능 등을 사용할 수 있다고 명시되어있다^[1]. 이것만으로도 소유자검증이 충분하기 때문에 신버전에서는 주민등록번호 입력란을 제외한 것으로 보인다. yesign사는 8자리, 한국전자인증사는 12자리, BTWorks사는 8자리와 16자리의 인증번호를 모두 이용한다. 프로그램 제조사, 버전마다 일관성이 없이 서로 다른 길이의 인증번호를 이용하는 것으로 보아 특별한 기준이 없는 것으로 보인다.

III. 12자리 인증번호를 사용하는 인증서복사 프로그램에 대한 보안성 분석

3.1. 네트워크 도청을 통한 공인인증서 복사 프로토콜의 분석

A사의 프로그램이 PC의 공인인증서를 스마트폰으로 복사하는 프로토콜은 그림 1과 같다. 먼저 스마트폰의 공인인증서 관리항목 중 ‘PC에서 공인인증서 복사하기’를 선택하면 ‘인증번호’라 불리는 12자리의 숫자를 생성하여 화면에 보여준다. 사용자는 PC에서 공인인증서 소유자 확인을 먼저 하고, 이 인증번호를 입력하여 공인인증서와 개인키를 중개서버로 전송한다. 스마트폰에서는 가져오기 버튼을 통해 중개서버로부터 공인인증서와 개인키를 받아온다.

그림 1에서는 사용자의 PC와 스마트폰 사이의 세션을 관리하는 부분을 찾아볼 수 없다. 대신 중개서버와 주고받는 네트워크상의 모든 패킷에서 40바이트의 문자열을 볼 수 있는데, 동일한 인증번호로부터 동일한 40바이트 문자열이 생성되는 것을 확인할 수 있었다. 이 논문에서는 PC와 스마트폰간의 세션역할을 대신하는 이 문자열을 인덱스라고 부른다.

서버로의 요청 패킷은 식별자, 인덱스, 데이터로 구성되어 있으며 서버의 응답 패킷은 식별자, 인덱스, 코드, 메시지(, 데이터)로 구성되어 있다. 식별자에 사용되는 ‘S’, ‘R’과 코드에 사용되는 ‘SC’의

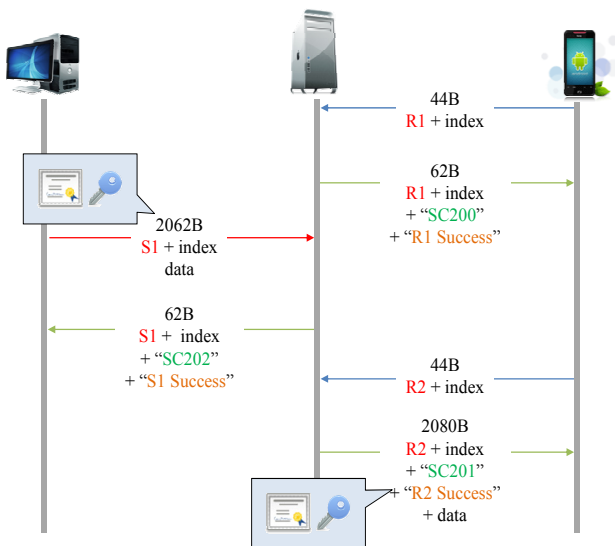


그림 1. 공인인증서 복사 프로토콜 (A社)
Fig. 1. A public-key-certificate-transport protocol (A-company)

경우 어떤 의미를 갖는 약어인지 자세히 알 수는 없으나, 현재의 전송 단계를 식별하거나 서버에서 보내는 데이터의 의미를 알려주는 것으로 보인다. 그림 1의 경우 올바른 인증번호를 입력했을 경우의 네트워크 패킷으로, 잘못된 인증번호를 입력했을 때 볼 수 있는 코드와 메시지는 각각 ‘PT117’, ‘No Exist Index’이다.

PC에서 인증번호를 입력한 직후 서버로 전송되는 패킷의 크기는 2062바이트이다. PC와 서버는 단 한 번의 패킷을 주고받기 때문에 공인인증서와 개인키는 이 패킷에 포함될 수밖에 없다. 공인인증서가 담긴 패킷에서 인덱스나 메시지 등을 제외하면 PC에서 중개서버로 보내는 데이터의 크기와 중개서버에서 스마트폰으로 보내는 데이터의 크기가 서로 같다.

공인인증서를 전송하는 채널은 암호화되어야 한다는 기술규격^[3]과 데이터의 크기가 서로 같다는 사실은 공인인증서와 개인키가 대칭키로 암호화되었다는 것을 말해준다. PC와 스마트폰은 사전에 키를 공유하지 않기 때문에 이 때 사용되는 대칭키는 12자리의 인증번호로부터 만들어졌음을 알 수 있다. 따라서 스마트폰이 임의로 생성한 인증번호는 곧 비밀번호로 사용된다. 중개서버는 인증번호를 알 수 없기 때문에 단지 전달자의 역할만을 수행한다.

3.2. 자바 디컴파일을 통한 안드로이드용 스마트폰뱅킹 어플리케이션 분석

3.1장의 분석을 통해 12자리 인증번호가 비밀로 이용됨을 알 수 있었지만, 구체적으로 대칭키를 만

드는 방법과 어떤 대칭키 암호화를 사용하는지는 알 수 없었다. 이를 확인하기 위해서 안드로이드용 스마트폰뱅킹 어플리케이션을 디컴파일하여 내부 구현을 살펴보았다.

분석을 위해 사용한 디컴파일 프로그램은 JD-GUI^[4]와 jad^[5]이다. 디컴파일 후 확인해 본 어플리케이션의 소스코드는 클래스명, 멤버 함수, 멤버 변수의 이름 모두가 식별하기 어려운 형태로 정의되어 있었다. 이는 어플리케이션 제작사가 디컴파일로부터 소스코드를 보호하기 위해 최적화와 난독화를 진행하였기 때문이다. 난독화된 소스코드는 프로그램의 분석을 어렵게 만들지만, 공격자에게 충분한 시간이 주어진다면 코드를 복원하는 것이 불가능한 것은 아니다.

그림 2는 복원된 소스코드의 분석을 통해 PRNG(Pseudo Random Number Generator)로부터 인덱스가 만들어지기까지의 모든 과정을 추약해서 보여준다. 스마트폰의 어플리케이션에서는 인증번호를 랜덤하게 생성하기 위해서 PRNG를 이용해 임의의 숫자를 생성한 뒤 이를 SHA1의 입력으로 이용한다. SHA1의 출력결과 20바이트중 처음 12바이트를 취한 뒤 각 바이트를 ‘0’부터 ‘9’까지의 문자로 변환한다. 이는 화면에 보이는 12자리의 인증번호로 사용된다.

이 인증번호는 다시 SHA1의 인자로 이용되고, 같은 작업을 1023회 반복한다. 1023회 째 SHA1 연산의 출력결과 20바이트중 앞의 16바이트는 SEED의 비밀키로, 뒤의 16바이트는 SEED의

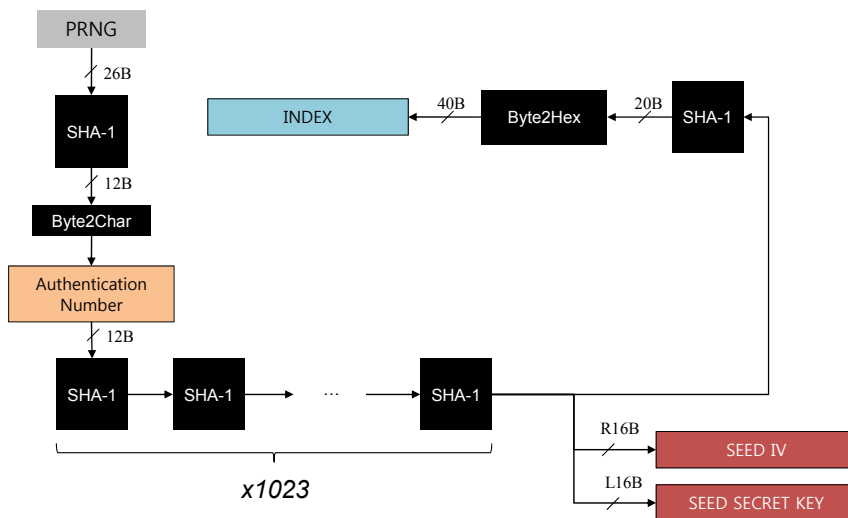


그림 2. PRNG로부터 인증번호, SEED 비밀키와 IV, 인덱스가 만들어지는 과정 (A社)
 Fig 2. The process of generating authentication number, the secret key and IV for SEED, and the index from PRNG (A-company)

IV(Initial Vector)로 사용되며 공인인증서와 개인 키를 SEED-CBC모드로 암호화한다. 이 20바이트는 다시 한 번 SHA1의 인자로 이용되며, 마지막 출력결과 20바이트는 4비트씩 잘라 16진수 문자로 변환하여 길이가 40바이트인 인덱스로 만든다.

이로써 3.1절에서 분석한 내용들의 구체적인 구현사항을 확인할 수 있었다. 12자리의 인증번호는 공인인증서와 개인키를 암호화하는데 사용되는 비밀이고, 그 보안강도는 40비트이다.

3.3. 공격 알고리즘의 구성

현재 주요 시중은행 10개사에서 사용하고 있는 스마트폰 공인인증서 복사 프로토콜은 인증번호를 기반으로 이루어지고 있으며, 짧게는 8자리, 길게는 16자리의 인증번호를 사용한다. 인증번호의 크기는 8자리의 경우 27비트, 12자리는 40비트, 16자리는 52비트로, 그 키의 크기가 최근에 요구되는 비밀키의 크기에 비해서 상대적으로 작기 때문에 공격자는 무작위 공격을 시도해볼 수 있다.

그림 3은 패킷으로부터 공인인증서와 개인키를

추출하는 알고리즘이다. 이 알고리즘은 PC에서 중개서버로 공인인증서를 전달하는 과정 또는 중개서버에서 스마트폰으로 전달하는 과정 중 어느 한 곳에서 도청한 패킷을 입력으로 받는다.

먼저 패킷으로부터 암호문과 인덱스를 각각 얻는다. 무작위 입력 값으로 만들어낸 인덱스와 패킷에서 얻은 인덱스가 일치한다면 비밀을 찾은 것이다. 공격자는 무작위로 생성한 인증번호에 대해 1023회의 SHA1연산을 수행하고 출력 값 160비트 중 왼쪽 128비트를 *seed_key*로, 오른쪽 128비트를 *seed_iv*로 저장한다. SHA1의 출력 값은 한번 더 SHA1의 입력으로 이용된다. 마지막 SHA1의 출력 값은 16진수 문자열로 변환하여 패킷에서 얻은 인덱스와 일치하는지 비교한다.

인덱스가 일치하는 경우 저장한 SEED의 비밀키와 IV로 암호문을 평문으로 복호화하고, 이 평문으로부터 공인인증서파일과 개인키 파일을 각각 추출할 수 있다. 인덱스가 일치하지 않는다면 무작위 입력 값을 바꾸고 다시 공격을 시도한다. 지금까지의 역사적 사실¹⁶⁾과 컴퓨터의 발전속도로 미루어 보건

Algorithm : GetPublicCertAndPrivateKey

```

input: a byte-stream of packet
Result: 'SignCert.der' and 'SignPri.key' files are created
1  begin
2      ciphertext ← GetCiphertext(packet)
3      index ← GetIndex(packet)
4      auth_nums ← '000000000000' // a 12-place authentication number
5      seed_key[0..15] ← {0}
6      seed_iv[0..15] ← {0}
7      repeat
8          index_t ← SHA1(auth_nums)
9          for i=0 to 1022
10             | index_t ← SHA1(index_t)
11         end
12         seed_key[0..15] ← index_t[0..15] // left most 16 bytes = the secret key of SEED
13         seed_iv[0..15] ← index_t[4..19] // right most 16 bytes = IV of SEED
14         index_t ← SHA1(index_t)
15         auth_nums ← auth_nums + 1
16     until index = index_t
17     plaintext = DecryptSEED(ciphertext, seed_key, seed_iv)
18     cert = GetPublicCert(plaintext)
19     private_key = GetPrivateKey(plaintext)
20     SaveToFile(cert, 'SignCert.der')
21     SaveToFile(private_key, 'SignPri.key')
22 end
    
```

그림 3. 알고리즘 GetPublicCertAndPrivateKey (A社)
 Fig. 3. Algorithm GetPublicCertAndPrivateKey (A-company)

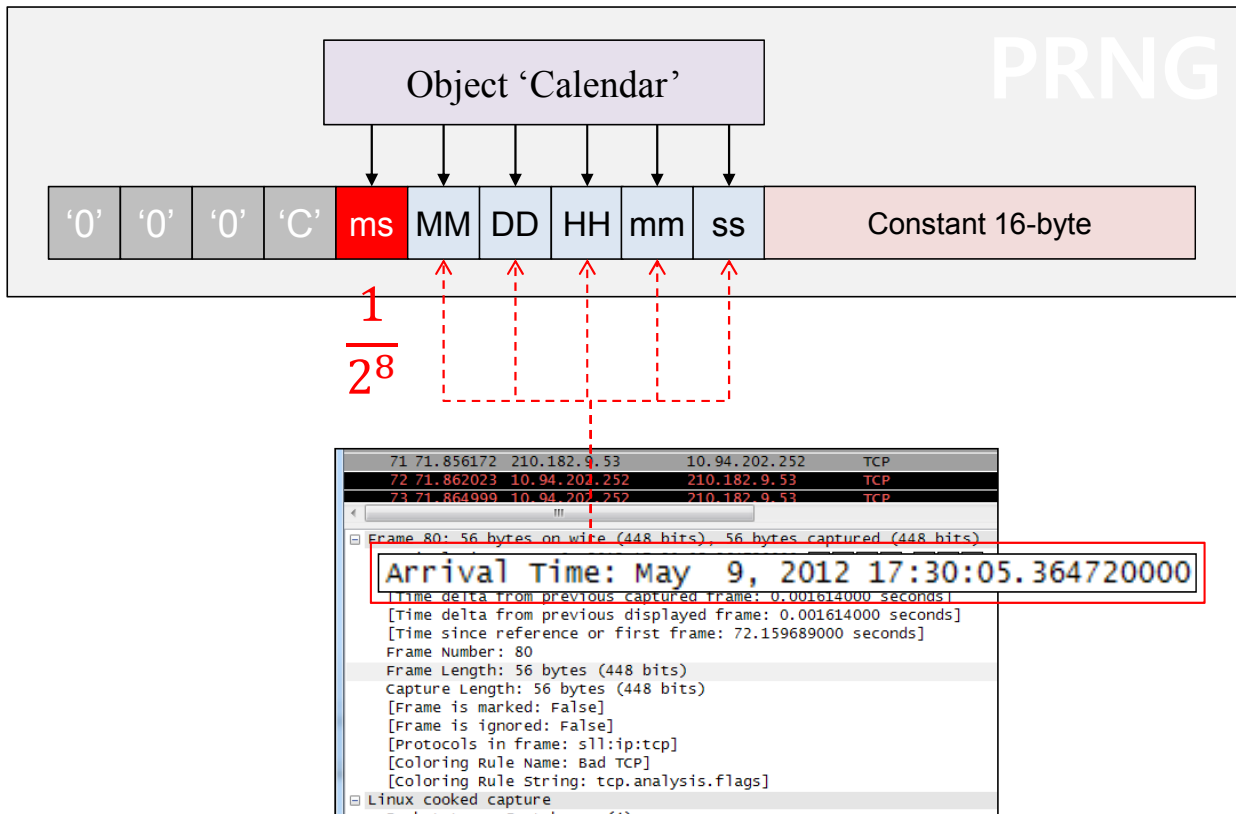


그림 4. PRNG 구성 (A社)
 Fig. 4. Construction of PRNG (A-company)

데 40비트 정도의 비밀은 현재 대다수의 컴퓨팅 환경에서 안전하다고 보기 힘들다.

공인인증서 복사과정에서 도청한 패킷을 그림 3의 알고리즘의 입력으로 이용하면 공인인증서 파일 SignCert.der와 개인키 파일 SignPri.key를 만들 수 있다. SignPri.key파일은 PKCS#8^[7]의 저장포맷에 따라 ASN.1 형식으로 저장되어 있고 PKCS#5^[8]에서 정의하고 있는 PBKDF1(Password Based Encryption Scheme)을 이용하여 공인인증서 패스워드로 암호화되어있기 때문에 공격자는 오프라인 사전 탐색 공격 등을 통하여 개인키를 얻어낼 수 있을 것이다.

IV. 안전하지 않은 PRNG를 사용하는 인증서복사 프로그램에 대한 보안성분석

4.1. A사의 PRNG구성

그림 4는 그림 2에서 사용된 PRNG의 구성 모습을 보여준다. A사의 PRNG는 앞쪽의 상수 4바이트, 시간정보를 이용한 6바이트, 뒤쪽의 상수 16바이트로 구성되어있으며, 이는 인증번호를 만드는 SHA1

의 입력으로 이용된다. 공격자는 도청한 패킷으로부터 시간 정보를 초 단위까지 알 수 있고, 디컴파일이나 리버스 엔지니어링을 통해 나머지 상수 20바이트를 알 수 있기 때문에 A사의 보안 프로그램에서 사용하고 있는 PRNG의 보안성은 밀리 초에 의존하게 된다. 여기에서 밀리 초 데이터는 바이트 크기의 메모리에 담기게 되므로 A사의 PRNG는 사실상 1바이트의 보안성을 갖는다. 따라서 인증번호는 사실상 40비트가 아닌 8비트의 보안성을 갖는다. 어플리케이션을 마켓에 등록하기 위해서는 어플리케이션 패키지 파일에 전자서명을 해야 하기 때문에 사용자별로 서로 다른 20바이트의 상수 값을 할당하는 것은 현실적으로 어렵다.

만약 공격자가 은행 등의 내부관계자일 때 이 PRNG는 더욱 큰 문제를 야기한다. 스마트폰으로 공인인증서를 복사하는 과정에서 사용자의 공인인증서와 개인키는 중개서버에서 임시로 보관하는데, PRNG를 재현할 수 있는 공격자는 이 서비스를 이용하는 모든 사용자의 공인인증서와 개인키를 복원할 수 있기 때문이다. 공격자는 은행 내부의 데이터 베이스에서 얻을 수 있는 다른 비밀까지 더 하여

```

0000 52 32 081c 34 61 62 31 37 33 66 34 61 34 36 38 R2...4ab173f4a468 // 081c = 2076
0010 63 33 62 37 32 66 38 32 39 65 32 38 32 61 37 39 c3b72f829e282a79
0020 37 34 39 33 38 32 33 30 66 39 65 38 05 53 43 32 74938230f9e8.SC2
0030 30 31 000a 52 32 20 53 75 63 63 65 73 73 07e0 01...R2 Success... // 07e0 = 2016
0040 ac 77 11 a7 a7 39 65 fe 34 a7 b4 3a 54 length .w...9e.4...:T.bC
    
```

그림 5. 공인인증서와 개인키가 포함된 패킷 (A社)
 Fig. 5. A packet containing a public key certificate and a private key (A-company)

계좌이체 등의 거래를 만들고 서명을 할 수 있을 것이다.

4.2. 올바른 PRNG의 사용을 위한 운영체제별 권고사항

암호학적으로 안전한 의사난수를 생성하기 위해서는 알고리즘과 시드 값이 모두 안전해야한다⁹⁾. 안드로이드 운영체제에서는 안전한 시드 값을 사용하는 SecureRandom() 함수를 통해 의사난수를 생성할 수 있다¹⁰⁾. 이 함수는 내부에서 /dev/urandom 와 같은 엔트로피가 높은 자원을 시드 값으로 이용하여 의사 난수를 생성한다. 시드 값을 인자값으로 전달할 수는 있지만 이는 보안성을 떨어뜨릴 수 있으므로 사용을 권고하지는 않는다. iOS에서는 SecRandomCopyBytes() 함수를 제공하며, 안드로이드 OS와 유사하게 /dev/random파일을 시드 값으로 이용한다¹¹⁾.

V. 실제 공격 시도 결과

우리는 PC의 공인인증서를 스마트폰으로 복사해 주는 과정에서 스마트폰의 무선 통신을 도청하여 패킷을 훔쳤다. 공인인증서가 포함된 패킷은 그림 5에서 볼 수 있듯이 데이터의 길이를 의미하는 바이트(볼드, 이탤릭체) 뒤에 실제 데이터가 기술되는 방식으로 구성되어있다. 패킷의 전체 길이는 2080 바이트이고 5~44 바이트는 인덱스를 의미한다. 공인인증서와 개인키를 합친 길이는 2016바이트이지만 그 내용은 암호화되어 있어서 패킷만으로는 그 내용을 알 수 없다.

우리는 인증번호가 생성된 후 서버로 보내는 패킷으로부터 인증번호가 만들어진 시간을 확인하였고, 이후 4.1절에서 설명한 것과 같이 8비트의 키 공간에 대해 그림 3의 알고리즘을 적용하여 암호화된 패킷으로부터 공인인증서와 개인키를 추출하는데

성공하였다.

훔친 공인인증서로 로그인하기 위해서는 파일의 위치를 지정된 장소에 두어야 한다. 공인인증서는 보통 ‘저장매체:\NPKI\user\주체정보폴더\’ 아래에 위치하는데, 주체정보는 공인인증서의 X.509 정보로부터 알아낼 수 있다. 그림 6에서는 X.509 정보 중 ‘Subject’에 주체정보가 기록되어있다는 것을 볼 수 있다. 우리는 추출한 공인인증서와 개인키를 이용하여 은행사이트에 정상적으로 로그인을 하였다. 악의적인 공격자는 훔친 패킷으로부터 공인인증서와 개인키를 추출하고 오프라인 사전 탐색 공격을 통해 공인인증서의 개인키를 알아낼 수 있을 것이다. 국내에는 은행을 포함하여 공인인증서만으로도 로그인할 수 있는 사이트가 여럿 있으며, 그 중 특히 주민등록등본 및 기타 민원서류 온라인 발급이 공인인증서 로그인만으로도 가능하기 때문에 이러한 취약점은 추가적인 피해로 이어질 수 있다.

VI. 안전한 인증서 복사를 위한 방법 모색

앞서 실험을 통해 현재 사용 중인 공인인증서 복사 프로토콜에 다양한 문제가 복합적으로 존재하고 있음을 보였다. 이 장에서는 각 문제점의 원인과 해결 방안에 대해 모색한다.

6.1. 긴 인증번호의 사용

한국인터넷진흥원은 공인인증서 및 개인키의 전송을 위해 RSA 1024비트 이상의 안전성을 만족하는 암호화기법의 이용을 권고하고 있으나 현재 사용되고 있는 서비스들은 이를 만족하지 못하고 있다³⁾. RSA 1024비트의 키 길이는 대칭키 80비트의 키 길이와 유사한 안전성을 가지며¹²⁾, 이 때 가능한 키공간의 크기는 약 1×10^{24} 이다. 현재 공인인증서 및 개인키의 복사 서비스에서 사용되고 있는 인증번호의 길이는 10진수 8자리, 12자리, 16자리이므로



그림 6. 암호화된 패키지로부터 공인인증서와 개인키를 꺼낸 모습 (A社)
 Fig. 6. Extracting a public key and a private key from the encrypted packet (A-company)

한국인터넷진흥원의 권고사항을 만족하지 않고 있다. 지금처럼 사용자 편의성을 위해 10진수만을 인증번호로 이용한다면 인증번호의 길이는 최소한 24 자리 이상이 되어야 한다. 이는 사용자의 편의성을 크게 떨어뜨리므로 현실적으로 사용하기 어렵다.

6.2. 공개키 시스템 사용

PC와 스마트폰 사이의 대칭키를 이용한 암호화와 복호화는 공격자에게 많은 힌트를 제공할 수 있다는 단점이 있다. 이를 대체할 수 있는 것으로 공개키시스템을 사용하는 방법이 있다. 우선 스마트폰에서 임시로 사용할 공개키와 개인키 쌍을 생성한 뒤 공개키만을 안전한 채널을 통해 중개서버로 보내고, 중개서버는 마찬가지로 안전한 채널을 통해 이 공개키를 PC로 전달한다. PC에서는 스마트폰으로부터 받은 공개키로 공인인증서와 개인키를 암호화해서 중개서버로 보낸다. 스마트폰에서는 중개서버로부터 암호화된 데이터를 받고, 프로토콜의 시작 시점에 생성한 개인키로 데이터를 복호화함으로써 공인인증서와 개인키를 얻을 수 있다. 이 방법은 대칭키방법에 비해서 더 많은 연산이 요구되지만 중간에 공격자가 개입할 수 있는 여지를 남기지 않아 안전하다고 할 수 있다.

6.3. 인증서의 직접 복사

네트워크를 통하지 않고 공인인증서 및 개인키를 복사하는 방법도 고려해볼 수 있다. 양대현은 QR코드를 이용해 휴대기기에 공인인증서를 전송하는 시스템 및 방법을 고안하였다¹³⁾. 사용자PC에서 공인인증서 및 개인키의 정보를 담아 QR코드로 인코딩하면 스마트폰에서 이 QR코드를 디코딩한다. 이 방식으로 공인인증서와 개인키는 네트워크를 통하지 않고 스마트폰으로 복사될 수 있다.

마지막으로 USB케이블을 이용해 공인인증서를 복사하는 방법이 있다. 안드로이드 운영체제의 경우 스마트폰을 PC에 연결해 USB 저장소를 적재한다. 공인인증서의 전송서비스 제공 소프트웨어는 전송하고자 하는 사용자가 공인인증서의 소유자임을 확인한 뒤, 적재된 USB 저장소에 공인인증서와 개인키를 복사할 수 있다. iOS의 경우 어플리케이션간에 파일 공유가 Document폴더만 가능하므로, 소유자가 확인이 되었다면 금융서비스를 제공하는 어플리케이션의 Document 폴더에 임시로 공인인증서와 개인키를 복사한다. 이 후 어플리케이션에서 내부 폴더로 다시 한 번 복사과정을 거치고 Document 폴더에 있는 공인인증서와 개인키를 삭제한다. 이렇게 직접 케이블을 이용해 복사하는 방법은 사용자 편

의성 면에서 좋지 않으므로, 공인인증서 복사를 위한 다양한 프로토콜이 개발되어야 한다.

VII. 결 론

이 논문에서는 스마트폰에서 인터넷뱅킹을 이용하기 위해 공인인증서와 개인키를 복사하는 프로토콜에 대해 분석하고 취약점과 대응방안에 대해 논하였다. 스마트폰의 보급률이 높아짐에 따라 PC에서 이용하던 전자금융서비스를 스마트폰, 태블릿에서 이용하는 비율도 점차 늘어나는 추세이기 때문에 앞으로도 공격자는 스마트폰에서 제공되는 서비스에 관심을 보이며 취약점을 찾아내려 노력할 것이다. 특히 바이트 코드를 가상기계(Virtual Machine)에서 실행하는 안드로이드와 윈도우모바일의 경우 디컴파일을 통해 그 구현내역을 파악하는 것이 다른 운영체제보다 용이하다. 다양한 스마트폰 운영체제에서 동일한 서비스를 제공하기 위해 개발되는 프로그램은 그 설계가 동일하거나 비슷하게 될 것이기 때문에 공격자는 그 중에서 자신이 분석하기 쉬운 운영체제의 응용프로그램을 분석하고 공격할 것이다. 결국, 공격자는 시스템을 전부 파악할 것이라는 가정 하에 시스템을 설계해야 한다. Kerckhoffs가 그의 논문에서 이야기한 ‘군사용 암호화 기법의 여섯 가지 디자인 원칙,^[14]에 따라 비밀키 이외에는 전부 공개해도 안전할 정도의 시스템을 설계하는 것이 바람직한 모습이라 할 것이다.

References

[1] Korea Commun. Commission, “Statistics of subscribers to telecommunications services,” Retrieved from <http://www.itstat.go.kr/board/boardDetailView.htm?identifier=02-008-120529-000003>, Apr. 2012.

[2] Law no. 10008, “Electronic signatures act,” Feb. 2010.

[3] Korea Internet & Security Agency, “Certificate transmission between PC to mobile device,” Mar. 2010.

[4] Java Decompiler-Graphic User Interface, Retrieved from <http://java.decompiler.free.fr/>,

June 2012.

[5] JAVa Decompiler, Retrieved from [http://en.wikipedia.org/wiki/JAD_\(JAVa_Decompiler\)](http://en.wikipedia.org/wiki/JAD_(JAVa_Decompiler)), June 2012.

[6] B. Schneier, *Applied cryptography*, 2nd Ed., John Wiley & Sons. ISBN 0-471-11709-9, pp. 153-154, 1996.

[7] PKCS#8 v1.2, “Private key information syntax standard,” May 2008.

[8] PKCS#5 v2.0, “Password-based cryptography standard,” Mar. 1999.

[9] C. Ellison, “Cryptographic random numbers,” *IEEE P1363 Appendix E*, Draft v1.0, Retrieved from <http://world.std.com/~cme/P1363/ranno.html>, Nov. 1995.

[10] SecureRandom, Retrieved from <http://developer.android.com/reference/java/security/SecureRandom.html>, June 2012.

[11] SecRandomCopyBytes, Retrieved from <http://developer.apple.com/library/ios/#DOCUMENTATION/Security/Reference/RandomizationReference/Reference/reference.html>, June 2012.

[12] E. Barker, “Recommendation for key management - Part 1: General rev.3,” NIST Special Pub. 800-57, May 2011.

[13] DaeHun Nyang, “System and method for transmitting certificate to mobile apparatus and system and method for transmitting and certifying data using multi-dimensional code,” Pat. no. KR-10-1113446, Dec. 2010.

[14] A. Kerckhoffs, “La cryptographie militaire,” *Jour. des sciences militaires*, IX, pp. 5-38, Jan. 1883.

신 동 오 (DongOh Shin)



2010년 2월 인하대학교 컴퓨터
정보공학과 공학사
2012년 2월 인하대학교 컴퓨터
정보공학과 석사
2012년 9월~현재 인하대학교
컴퓨터 정보공학과 박사과정
<관심분야> 인터넷 보안, 네트

워크 보안, 금융 보안

강 전 일 (Jeonil Kang)



2003년 2월 인하대학교 컴퓨터
공학과 학사
2006년 2월 인하대학교 정보통
신공학과 석사
2006년 3월~현재 인하대학교
정보통신공학과 박사과정
<관심분야> RFID 보안, 생체

인식 보안, WSN 보안, 무선 인터넷 보안, 웹 인
증 보안

양 대 현 (DaeHun Nyang)



1994년 2월 한국과학기술원 과
학기술 대학 전기 및 전자
공학과 졸업
1996년 2월 연세대학교 컴퓨터
과학과 석사
2000년 8월 연세대학교 컴퓨터
과학과 박사

2000년 9월~2003년 2월 한국전자통신연구원 정보
보호연구본부 선임연구원

2003년 2월~현재 인하대학교 컴퓨터정보공학과 부
교수

<관심분야> 암호이론, 암호프로토콜, 인증프로토콜,
무선 인터넷 보안

이 경 희 (KyungHee Lee)



1993년 2월 연세대학교 컴퓨
터과학과 학사
1998년 8월 연세대학교 컴퓨
터 과학과 석사
2004년 2월 연세대학교 컴퓨
터 과학과 박사
1993년 1월~1996년 5월 LG

소프트(주) 연구원

2000년 12월~2005년 2월 한국전자통신연구원 선
임연구원

2005년 3월~현재 수원대학교 전기공학과 조교수
<관심분야> 바이오인식, 정보보호, 컴퓨터비전, 인
공지능, 패턴인식