인증서기반의 혼합방식 암호시스템 설계 및 구현* - CHyCK

홍 성민 °, 오 상엽, 조 기호, 이 경은, 천 명권, 이 영, 천 정희, 윤 현수
한국과학기술원 전산학과, 한국과학기술원 수학과

A Design and Implementation of Certificate-Based Hybrid Cryptosystem - CHyCK

Hong Seong-Min°, Oh Sang-Yeop, Cho Ki-Ho, Lee Kyung-Eun, Tcheun Myoung-Kwon, Lee Young, Tcheun Jeong-Hee, Yoon Hyun-Soo
Dept. of Computer Science, KAIST, Dept. of Mathematics, KAIST

요약

본 논문에서는 공개키방식알고리즘과 대칭키방식알고리즘을 혼합하여 사용하는 혼합방식 암호시스템(CHyCK:Certificate-based Hybrid Cryptosystem of KAIST) 의 구현에 대해 설명한다. CHyCK는 전송하고자 하는 메세지를 대칭키암호알고리즘을 이용해서 암호화하고 이 때에 사용되는 대칭키를 공개키방식으로 암호화하여 상대방에게 암호화된 메세지와 함께 보내게 된다. CHyCK는 공개키방식에서 사용되는 키쌍 중 공개키를 안전하게 가입자에게 분배해 주는 방법으로 인증서기반(certificate-based)방식을 채택하였다. 또한 이를 위해서는 인증서를 발급해 줄 인증기관(certifying authority)이 필요하게 된다. 그리고 시스템을 구성하는 기본 암호알고리즘들은 기존의 것들 중 적합한 것으로 선택했다. 마지막으로 본 논문에서는 시스템의 안전성을 키분배와 메세지 전송 측면에서 분석했다.

1 서론

1976년 Diffie와 Hellman[1]이 공개키(public-key)개념을 제안한 이래로, 공개키방식의 암호시스템에 대한 수많은 연구가 이루어져 왔고, 또한 그 결과로 많은 공개키방식의 암호알고리즘이 개발되었다[2]. 그러나 공개키방식이 기존의 대칭키(symmetric-key)방식에 비해 키관리 문제를 효과적으로 해결할 수 있게 해주고 전자서명을 가능하게 한다는 점은 사실이지만 키분배는 여전히 중요한 문제가 되고 있다.

공개키암호시스템은 공개키와 개인키(private-key)로 이루어진 키쌍(key-pair)을 가지고 있다. 개인키의 경우에는 가입자(subscriber) 자신만이 알고있는 키이므로 분배의 문제가 없지만 공개키의 경우에는 그렇지 않다. 공개키는 대칭키처럼 비밀성을 유지하며 전달되어야 할 필요는 없지만 가입자들 모두에게 변형되지 않고 전달되어야 하므로 가입자들은 각자 자신이 원하는 올바른 공개키인지를 확인할 수 있는 수단을 필요로 한다.

이러한 목적을 위해서 이루어진 많은 연구들 중에서 주목할 만한 것에는 식별자기반암호시스템(ID-based cryptosystem)과 인증서기반암호시스템(certificate-based cryptosystem)의 두 가지가 있다. 이 중 식별자기반암호시스템은 가입자의 식별자를 공개키로 사용하여 가입자들 모두가 별도의 통신을 통하지 않고도 공개키를 알 수 있도록 하자는 기본개념을 가진 암호시스템이다. 그러나, 이 방식에서는 신뢰성을 가지는 제3자가 가입자의 개인키와 공개키를 만들어 배포하게 되는데, 이로써 두 가지 문제가 생긴다. 하나는 가입자의 개인키가 한 기관(trusted third party)에 보관되어 있게 되므로 그 기관에 대한 신뢰성이 절대

*본 연구는 국민은행과 (주)한글과컴퓨터의 지원에 의해 수행중임

적인 것이어야 하고 또한 가입자의 입장에서는 자신의 개인키를 다른사람이 보관하고 있다는 것이 내키지 않을 수 있다. 다른하나는 신뢰성을 가지는 제3자가 각 가입자에게 개인키를 나누어 주는 방법에 대한 것인데, 가입자가 비밀성을 유지하며 자신의 개인키를 신뢰성을 가지는 제3자로부터 얻을 수 있는 안전한 통로(secure channel)가 필요하다는 점이다[3, 4, 5, 6, 7]. 다른하나인 인증서기반암호시스템은 믿을만한 제3자가 인증기관의 역할을 하여 각 가입자의 공개키에 서명하여 인증서를 발급하여 줌으로써 모든 가입자들이 다른가입자들의 공개키가 올바른 것인지 확인할 수 있도록 하는 방식으로[8, 9], 직관적으로 매우 자연스러운 개념이고 인증기관이 가져야 할 신뢰성이 그리 크지 않으므로 현실적으로 구현 가능하고 게다가 확장성이 좋으므로 규모가 큰 환경에도 적합하다. 따라서 CHyCK는 후자인 인증서기반방식을 채택하여 공개키분배문제를 해결하였고, 이를 위한 실제적인 방법으로 클라이언트/서버 모델을 사용하여 인증기관을 구현하였다.

또한 공개키암호알고리즘은 대칭키암호알고리즘에 비해 현저히 느리기때문에 공개키암호알고리즘만으로 시스템을 구현하게 되면 속도면에서 효율성이 크게 떨어지게 된다. CHyCK가 선택한 혼합방식(hybrid-method)은 실제로 전송하고자 하는 메세지를 속도가 빠른 대칭키암호알고리즘으로 암호화하고 이 때에 사용된 대칭키를 상대방의 공개키로 암호화하여 안전성을 위한 다른 정보들(예를 들면, 재사용방지코드나 메세지의 일방향해쉬값)과 함께 전송함으로써 공개키암호시스템과 대칭키암호시스템의 장점을 살려 서로의 단점을 보완할 수 있는 절충형 방식이다.

2 시스템 개요

CHyCK는 컴퓨터 네트워크상에서 통신정보를 보호하는 것을 주된 목표로 하는 암호시스템으로서(그림1(a)) 기본적인 주된 기능은 다음과 같다.

- 자료의 비밀성 유지(Secrecy)
- 자료 작성자의 정체 확인(Identification)
- 자료의 무결성 확인(Integrity)
- 자료 작성자의 부인 방지(Nonrepudiation)
- 자료의 적시성 확인(Timeliness)

또한 위와같은 기본적인 기능들을 제공하기 위해서 필요한 키관리(key management)면에서 개인키(private-key)관리와 인증기관의 공개키 관리, 그리고 공개키목록(public-key list) 관리의 3가지 기능을 제공한다.

2.1 용어정의

시스템의 구성에 대해 설명하기 전에 우선 앞으로 본 논문에서 CHyCK 시스템에 대해 기술하고 분석하는데 있어서 형식적인(formal) 접근을 위해 사용하게 될 용어들을 정의한다.

2.1.1 시스템 구성요소들의 약어

약어	설명
CA	인증기관(Certifying Authority)
GKL	인증기관이 가지고 있는 전체가입자의 공개키목록(Global Key List)
LKL	각 가입자마다 자신이 자주 사용하는 상대방의 공개키들을 저장하고 있는 공개키목록(Local Key List)
KA	클라이언트의 구성요소로 통신과 키관리를 담당하는 부분(Key Agent)
DPU	클라이언트의 구성요소로 데이타의 암/복호화 등의 처리를 하는 부분(Data Processing Unit)
API	클라이언트의 구성요소로 가입자에게 직접 서비스를 제공해 주는 부분 (Application Program Interface)
MM	컴퓨터 내의 주기억장소(Main Memory)
DK	키들을 저장하는 장소(DisK)

2.1.2 여러 암호알고리즘들과 사용되는 키들

기호	설명
M	가입자가 전송하고자 하는 메세지
ID_A	가입자 A의 식별자
MK	가입자나 인증기관이 개인키를 저장할 때에 사용하는 대칭키(Master Key)
PP	패스문구를 나타내는 기호(Pass-Phrase)
TK	대칭키암호화알고리즘에 사용되는 임시키(Temporary Key)
PK_A	공개키알고리즘에 사용되는 가입자 A의 공개키(Public-Key)
SK_A	공개키알고리즘에 사용되는 가입자 A의 개인키(Private-Key)
SE_{TK}	임시대칭키 TK를 사용한 대칭키암호화알고리즘(Symmetric Encryption)
SD_{TK}	임시대칭키 TK를 사용한 대칭키복호화알고리즘(Symmetric Decryption)
AE_{PK}	공개키 PK를 사용한 공개키암호화알고리즘(Asymmetric Encryption)
h	일방향해쉬함수(one-way Hash function)
lh	일방향해쉬값중 상위 64비트를 추출하는 함수(Left one-way Hash value)
rh	일방향해쉬값중 하위 64비트를 추출하는 함수(Right one-way Hash value)
rdc	재사용감지코드(Replay Detection Code)
TS	재사용감지코드로 사용되는 시각기록(Time Stamp)
$Cert_A$	가입자 A의 공개키에 대한 인증서(Certificate)

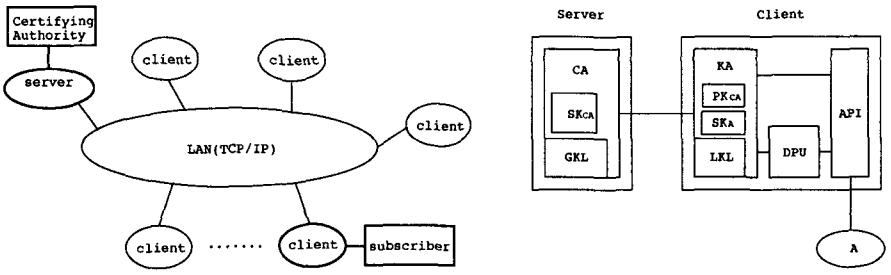
2.1.3 시스템 동작 시나리오를 설명하기 위한 기호들

기호	의미
$=?$	좌변과 우변이 동일한지를 조사하는 검증(verify)을 의미한다.
$\rightarrow$	좌면에서 우변으로 메세지가 전송됨을 의미한다.
$\rightsquigarrow$	데이타의 저장 및 복원을 나타낸다.
$\uparrow$	데이타의 생성을 의미하는데, 암호화된 정보의 복원도 포함한다.
$:$	좌변은 동작의 주체나 메세지 전송의 방향을, 우변은 동작 혹은 전송메세지의 구조를 나타낸다.
$\|\|$	좌변과 우변의 문자열붙임(string concatenation)을 나타낸다.

2.2 공개키 분배를 위한 시스템 구성

가입자들의 공개키가 변형되지 않고 정확히 분배되도록 하기 위해 CHyCK는 클라이언트/서버 모델을 이용한다(그림1(b)). 클라이언트는 각 가입자가 사용하는 시스템모듈이고 서버는 인증기관의 역할을 하는 시스템모듈이다. 클라이언트와 서버 각각의 역할과 기능을 살펴보면, 먼저 서버는 인증기관의 역할을 하므로 다음과 같은 기능이 필요하다.

- 가입자의 공개키에 대한 인증서 발급
- 전체 가입자의 공개키 목록 관리
- 공개키인증서의 안전한 분배



(a) (b)

그림 1: (a) 시스템 구성 (b) 클라이언트/서버 모델

클라이언트가 가입자에게 필요한 서비스를 제공하기 위해서는 다음과 같은 기능을 필요로 한다.

- 인증기관과의 통신
- 공개키목록과 개인키 관리
- 안전한 통신을 위한 데이타 처리
- 가입자가 서비스를 받을 수 있도록 해주는 인터페이스

이러한 기능들을 수행하기 위한 클라이언트의 구조를 살펴보면 KA, DPU, 그리고 API의 세부분으로 이루어져 있다. DPU는 데이타의 암호화와 복호화, 서명의 생성과 검증, 메세지에 대한 일방향해쉬값의 계산, 그리고 압축의 기능을 가진다. 압축기능이 필요한 이유는 전송할 메세지를 암호화하기 전에 미리 압축을 함으로써 전송할 데이타의 크기를 줄일 수 있고 또한 비도를 증가시키는 효과도 얻을 수 있기때문이다[10]. KA는 가입자의 개인키와 공개키 그리고 부분공개키목록을 관리하는 역할과 인증기관 또는 다른가입자와의 통신을 담당한다. 마지막으로 API는 가입자가 클라이언트와 접촉하게 되는 모듈로서 가입자에게 보이는 부분은 바로 이 부분뿐이다. API는 인증기관에의 등록/갱신/삭제, 전송메세지의 암호화, 수신메세지의 복호화, 서명생성, 그리고 서명검증의 7가지 기능을 가입자에게 제공한다.

3 키 관리

3.1 공개키 생성 및 분배

공개키방식에서 사용되는 각 가입자의 키쌍(key pair)의 생성은 클라이언트에서 이루어진다. 공개키방식으로 RSA를 사용하므로, 먼저 큰 소수 p와 q를 생성하고 나서 $(p-1)(q-1)$과 서로 소인 e를 찾아낸다. 그런 다음 $e \times d = 1 \ mod \ (p-1)(q-1)$인 d와 $n(=pq)$을 찾는다. 그러면 (e,n)은 공개키가 되고 d는 개인키가 된다[8]. 생성된 키쌍 중 공개키에 대해서 클라이언트는 인증기관과의 통신을 통해 공개키인증서를 받게 되고, 인증기관은 이를 인증기관 자신의 전체공개키목록(Global Key List)에 보관하게 되는데(그림2(a)), 이러한 공개키인증서의 형태는 다음과 같다.

$$Cert_A = AE_{SK_{CA}}(ID_A||PK_A||TS)$$

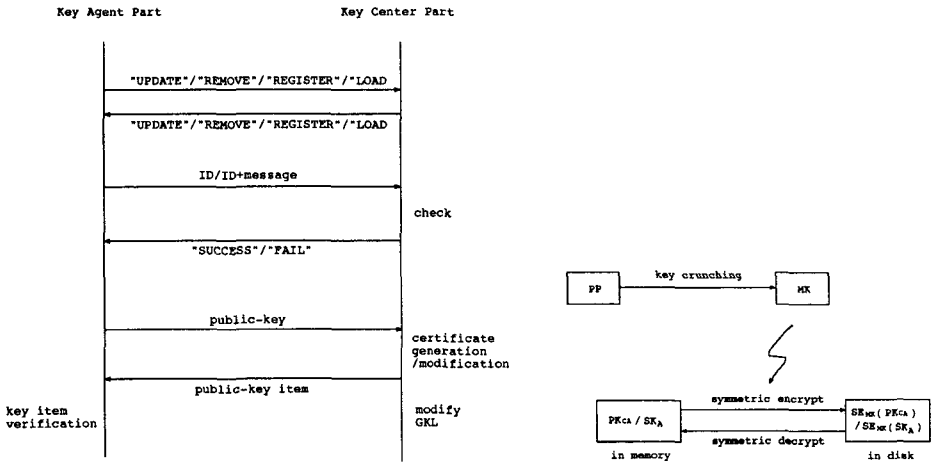


그림 2: (a) 공개키인증서 생성 및 분배를 위한 프로토콜 (b) 개인키 저장 및 사용

인증기관이 공개키인증서를 생성한 후에 이것을 클라이언트에게 보내줄 때 공개키덩어리(public-key item)의 형태로 보내주게 되는데, 이는 가입자가 공개키인증서를 확인할 수 있도록 하기 위한 것으로 다음과 같은 항목들을 가지고 있다.

- 가입자의 식별자(ID_A)
- 가입자의 공개키(PK_A)
- 가입자의 공개키인증서($Cert_A$)
- 가입자의 공개키인증서가 만들어진 시각(TS)

이와같은 공개키덩어리를 받은 가입자는 올바르게 작성된 인증서인지를 확인하기 위해서 검증작업을 한다.

$$AE_{PK_{CA}}(Cert_A) \ =?\ ID_A||PK_A||TS$$

가입자가 다른가입자에게 메세지를 보내기 위해서 상대방의 공개키를 알고자 할 때에도 그림2(a) 의 프로토콜을 이용해서 인증기관으로부터 원하는 상대방의 공개키인증서를 얻는다.

3.2 키 저장방식

공개키알고리즘에 사용되는 키쌍 중에서 개인키는 공격자의 시스템 침입시 비밀성을 유지하기 위해서 저장할 때에 마스터키(master key)를 이용해서 대칭키암호알고리즘으로 암호화시켜 저장하고, 사용할 때에는 복호화시켜서 사용하도록 구현하였다(그림2(b)).

이 때에 사용되는 마스터키는 가입자가 알고 있어야 하는 패스문구(pass phrase)를 이용해서 생성하는데, 이러한 마스터키의 생성을 키크런칭(key crunching)이라 한다. 키크런칭

은 일방향해쉬함수인 MD5로 계산한 패스문구의 일방향해쉬값을 상위 64비트와 하위 64비트로 나눈 후, 이 두 값을 XOR함으로써 이루어진다.

$$MK = lh(PP) \oplus rh(PP)$$

$$MM \rightsquigarrow DK : SE_{MK}(SK_A)$$

$$DK \rightsquigarrow MM : SD_{MK}(SE_{MK}(SK_A))$$

또한 가입자가 자신의 공개키를 인증기관에 등록할 때 인증기관으로부터 전송받게 되는 인증기관의 공개키는 시스템공격자에 의한 변형방지를 위해서 가입자 자신의 개인키와 같은 방법으로 저장하여 사용한다.

공개키목록은 전체공개키목록(GKL)과 부분공개키목록(LKL)으로 나뉘는데 전체공개키목록은 가입자가 원할 때마다 즉시 공개키인증서를 전달해 줄 수 있도록 하기위해서 인증기관이 관리하는 공개키목록이며, 부분공개키목록은 각 가입자마다 자신이 자주 통신을 하는 상대방 가입자와 공개키를 가입자 자신이 가지고 관리함으로써 시스템의 효율성을 높일 수 있도록 해주는 공개키목록이다.

공개키목록(전체공개키목록과 부분공개키목록 모두)은 인증기관이 생성한 공개키인증서를 포함하는 공개키인증서덩어리 형태 그대로 저장한다. 그리고 사용할 때마다 저장해 두었던 공개키가 공격을 받아 변형되었는지 아닌지를 검증하는 과정을 거쳐 사용한다. 검증하는 방법은 처음에 인증기관으로부터 전송받을 때 검증하는 방법과 동일하다.

4 메세지의 전달 방법

본 절에서는 앞의 2절에서 밝힌 CHyCK가 제공하는 기본적인 5가지 기능들을 지원하기 위해서 취해지는 메세지 전달방법에 대해 기술한다.

4.1 메세지의 구조

한 가입자로부터 다른가입자에게로 전송될 메세지는 암호화속도의 이유로 대칭키암호알고리즘을 사용해서 임시대칭키로 암호화 된다. 중요한 문제는 이 때 사용된 임시대칭키를 어떠한 방법으로 상대방에게까지 안전하게 전달할 것인가이며, 이를 위한 메세지의 구조에 대해 살펴 보겠다.

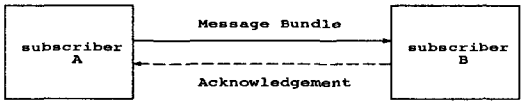


그림 3: 안전한 메세지 전달

메세지의 전달을 위해서 다음의 다섯 개의 모듈[11]이 사용된다.

- $SE_{TK}(M)$
- $AE_{PK_B}(TK)$
- $h(M)$

- $AE_{SK_A}(h(M))$
- rdc

이와같은 구성모듈들을 가지고 메세지 전달에 있어서의 다섯가지 기본조건인 자료의 비밀성, 무결성, 적시성, 작성자의 정체확인, 부인방지, 그리고 선택조건인 전달증명을 만족시킬 수 있도록 메세지를 구성하였다.

$$\text{message bundle} = ID_A||SE_{TK_A}(M)||AE_{PK_B}(TK_A)||AE_{SK_A}(ID_A||h(M)||\text{rdc})$$

이는 그림3에서 가입자A가 가입자B에게 보내는 메세지덩어리(message bundle)의 구조가 되며, 이에 대한 안전성 분석은 6장에서 하도록 하겠다. 전달증명을 제외한 5가지 기능만을 만족시키기 위해서는 상호교환(interactive communication)이 필요치 않지만 전달증명 기능까지 만족시키기 위해서는 알림신호(acknowledge signal)가 필요하므로 부가적 기능으로 알림신호의 필요유무를 지정할 수 있도록 했다.

4.2 시스템 동작 시나리오

CHyCK에서 가입자 A가 가입자 B에게 메세지를 전달하는 과정은 다음과 같이 세 단계로 구성된다.

1. 가입자 A는 인증기관에 자신의 공개키를 등록한다.
   (a) A : $\uparrow (PK_A, SK_A)$
   (b) A $\rightarrow$ CA : $ID_A||PK_A$
   (c) CA $\rightarrow$ A : $AE_{SK_{CA}}(ID_A||PK_A||TS)||ID_A||PK_A||TS$
   (d) CA : $AE_{SK_{CA}}(ID_A||PK_A||TS)||ID_A||PK_A||TS \rightsquigarrow GKL$
   (e) A : $AE_{PK_{CA}}(AE_{SK_{CA}}(ID_A||PK_A||TS)) =? ID_A||PK_A||TS$
   (f) A : $AE_{SK_{CA}}(ID_A||PK_A||TS)||ID_A||PK_A||TS \rightsquigarrow LKL$
   (g) A : $\uparrow MK_A \ (= lh(PP_A) \oplus rh(PP_A))$
   (h) A : $SE_{MK_A}(SK_A) \rightsquigarrow DK$
   (i) A : $SE_{MK_A}(PK_{CA}) \rightsquigarrow DK$
2. 가입자 A는 메세지를 암호화하여 가입자 B에게 전송한다.
   (a) A : $\uparrow TK_A$
   (b) A : $SE_{TK_A}(M)$
   (c) A $\rightarrow$ CA : ID_B
   (d) CA $\rightarrow$ A : $AE_{SK_{CA}}(ID_B||PK_B||TS)||ID_B||PK_B||TS$
   (e) A : $\uparrow MK_A \ (= lh(PP_A) \oplus rh(PP_A))$
   (f) A : $DK \rightsquigarrow PK_{CA} \ (= SD_{MK_A}(SE_{MK_A}(PK_{CA}))$
   (g) A : $DK \rightsquigarrow SK_A \ (= SD_{MK_A}(SE_{MK_A}(SK_A))$
   (h) A : $AE_{PK_{CA}}(AE_{SK_{CA}}(ID_B||PK_B||TS)) =? ID_B||PK_B||TS$
   (i) A : $AE_{PK_B}(TK_A)$
   (j) A : $AE_{SK_A}(ID_A||h(M)||TS)$
   (k) A $\rightarrow$ B : $ID_A||SE_{TK_A}(M)||AE_{PK_B}(TK_A)||AE_{SK_A}(ID_A||h(M)||TS)$
3. 가입자 B는 전송받은 메세지를 해독하고, 메세지의 전송자가 가입자 A임을 인증한다.

(a) B : $\uparrow MK_B$ (= $lh(PP_B) \oplus rh(PP_B)$)
(b) B : $DK \rightsquigarrow SK_B$ (= $SD_{MK_B}(SE_{MK_B}(SK_B))$)
(c) B : $\uparrow TK_A$ (= $AE_{SK_B}(AE_{PK_B}(TK_A))$)
(d) B : $\uparrow M$ (= $SD_{TK_A}(SE_{TK_A}(M))$)
(e) B $\rightarrow$ CA : ID_A
(f) CA $\rightarrow$ B : $AE_{SK_{CA}}(ID_A||PK_A||TS)||ID_A||PK_A||TS$
(g) B : $DK \rightsquigarrow PK_{CA}$ (= $SD_{MK_B}(SE_{MK_B}(PK_{CA}))$)
(h) B : $AE_{PK_{CA}}(AE_{SK_{CA}}(ID_A||PK_A||TS))$ =? $ID_A||PK_A||TS$
(i) B : $AE_{PK_A}(AE_{SK_A}(ID_A||h(M)||TS)$ =? $ID_A||h(M)||TS$

5 시스템 구현

CHyCK는 UNIX를 운영체제로 하는 컴퓨터들로 이루어진 LAN상에서 TCP/IP 프로토콜로 구현되었고 프로그래밍에 사용된 언어는 C언어이다. 서버와 클라이언트 사이의 통신에는 버클리소켓(Berkeley Socket)을 이용하고, 연결지향프로토콜(connection-oriented protocol)로 구현하였다[12].

CHyCK에서 사용한 모듈별 알고리즘을 보면, 공개키암호알고리즘으로는 RSA를 구현하였고, 대칭키암호알고리즘으로는 DES를 구현하여 사용했다. 전자서명알고리즘으로는 공개키암호알고리즘으로 사용되는 RSA를 이용했으며, 전자서명에 필요한 일방향해쉬함수로는 MD5를 사용했다[8, 13, 14]. 그리고 압축프로그램으로는 gzip을 사용하였다.

6 안전성 분석

CHyCK를 공격하는 데에는 시스템의 안전성 자체를 위협하는 공격과 전송중인 메세지에 가하는 공격이 있을 수 있다.

6.1 시스템 공격

암호시스템 자체를 위협하는 공격방법에는 다음과 같은 세 가지가 있을 수 있다.

- 시스템을 구성하고 있는 모듈들을 공격하는 방법
- 인증기관 또는 가입자의 개인키를 알아내는 방법
- 가입자의 공개키를 거짓으로 퍼뜨리는 방법

첫번째 방법의 공격이 성공하게 되면 시스템 전체의 안전성이 위험하게 된다. 예를 들어 RSA가 깨어지게 되거나, MD5에서 같은 해쉬값을 가지는 서로다른 메세지를 찾는 것이 계산상 가능하게 되는 등 시스템을 구성하고 있는 모듈들의 안전성을 믿을 수 없게 되는 경우에 CHyCK는 더이상 사용할 수 없게 된다. 이러한 공격에 대해서는 시스템구성모듈을 선택할 때에 신중하게 고려해서 선택하는 방법이 최선이다.

두번째 방법의 공격이 성공하게 되어 공격자가 다른 가입자 A의 개인키를 알게 되면 가입자 A에게로 가는 모든 비밀 메세지를 해독할 수 있고, 또한 다른가입자들에게 가입자 A인 것처럼 가장할 수 있게 된다. 물론 인증기관의 개인키가 탄로나게 되었을 경우는 시스템 전체가 더이상 사용할 수 없게 된다. 이러한 공격을 피하기 위해 CHyCK에서 사용하는 방법은 패스문구로부터 만들어낸 마스터키를 이용해서 암호화된 상태로 저장하는 방법이다. 패스문구는 외우기 쉽고 적당히 긴 문장이면 된다.

세번째 방법의 공격은 처음 두 가지 방법에 비해 상대적으로 덜 치명적이나 시스템에 혼란을 줄 수 있으므로 마찬가지로 중요하다. 이러한 공격에 대해서 CHyCK에서는 인증서기반(certificate-based)방식을 채택하여 인증기관이 인증서를 만들어 분배하는 역할을 하도록 시스템을 구성하였다. 인증기관의 인증서는 가입자의 공개키를 인증기관 자신의 개인키로 서명한 것이므로 인증기관의 공개키가 바르게 가입자에게 전달 및 보관되고 인증기관의 개인키가 안전하게 보관되면, 안전하게 공개키를 분배할 수 있다.

지금까지 설명한 대로 일반적으로 있을 수 있는 공격방법에 대해 시스템의 안전성을 유지하기 위한 장치들을 마련하였다.

6.2 전송 메세지 공격

다음으로 전송중인 메세지덩어리에 가하는 공격을 생각해 볼 수 있다. 4.1장에서 보인 메세지덩어리의 구조를 Fumy와 Munzert가 제안한 방법[11]을 이용해서 만들어나감으로써 안전성을 분석해 보았다. 먼저 기본적으로 생각해 볼 수 있는 메세지덩어리의 구조는 다음과 같다.

$$SE_{TK}(M)||AE_{PK_B}(TK)$$

보내고자 하는 메세지를 임시키를 사용해 대칭키방식으로 암호화하고, 이 때 사용된 임시키를 보내고자 하는 상대방 가입자(B)의 공개키로 암호화하여 보내게 되면 자료의 비밀성이 만족된다. 여기에 보내고자 하는 메세지 M의 해쉬값을 추가시키면 자료의 무결성 기능까지 할 수 있게 되고, 재사용감지코드를 추가시켜 적시성을 조사할 수 있도록 했다. 또한 자료작성자의 정체확인과 부인방지 기능을 위해서 해쉬값을 전송자 자신의 개인키를 사용해서 암호화하도록 했다. 마지막으로 전달증명을 위해서는 알림신호를 필요로 하게 되므로 정리하면 다음과 같다.

$$A \rightarrow B : SE_{TK_A}(M)||AE_{PK_B}(TK_A)||h(M)||rdc||AE_{SK_A}(h(M))$$

$$B \rightarrow A : SE_{TK_A}(h(M))$$

그런데 가입자 A에서 가입자 B로 가는 메세지덩어리의 구조가 중복되는 항목들이 있으므로 이를 잘 조합해 보면 4.1장의 메세지덩어리 구조가 된다.

7 결론

지금까지 혼합방식 클라이언트/서버 모델 암호시스템인 CHyCK의 설계와 구현에 대해 기술하였다. CHyCK의 각 모듈들을 클라이언트와 서버로 나누어 각각의 기능과 구성에 대해 살펴보았고, 공개키의 분배문제와 개인키의 저장문제에 대한 CHyCK의 해결책을 설명하였다. 그리고 본 논문의 2절 앞부분에서 나열하였던 CHyCK의 기본적인 기능의 수행원리를 구조적인 측면에서, 그리고 실제적인 동작시나리오를 통해서 살펴보았다. 또한 구체적인 시스템의 구현환경에 대해 간략하게 언급하고, CHyCK의 안전성을 키의 분배 및 관리의 측면과 메세지 전송 측면에서 분석해 보았다. 본 논문에서 구현에 대해 논한 CHyCK의 특징을 간단히 정리하면 다음과 같다.

- 공개키방식과 대칭키방식의 혼합방식
- 인증서기반 방식으로 공개키 분배
- 컴퓨터 네트워크 상에서 클라이언트/서버 모델을 이용
- 암호알고리즘으로 RSA와 DES사용

- 일방향해쉬함수로 MD5 사용
- 전자서명 방식으로 RSA 사용

현재 CHyCK의 튜닝(tunning)과 성능분석중이며, 향후 CHyCK의 실시간수행을 위해 수행 속도향상에 주력할 예정이다.

참고 문헌

[1] W.Diffie and M.E.Hellman, "New directions in cryptography," *IEEE Trans. Computers*, vol. IT-22, pp. 644–654, June 1976.

[2] W. Diffie, "The first ten years of public-key cryptography," in *PROCEEDING OF THE IEEE*, vol. 76,NO.5, pp. 560–576, May 1988.

[3] S.Tsujii, K.Kurosawa, and T.Itoh, "New noninteractive identity-based key distribution system," *Electronic Letters*, vol. 24, pp. 1357–1358, Oct. 1988.

[4] C. G.Gunther, "An identity-based key-exchange protocol," in *Eurocrypt89*, pp. 29–37, 1989.

[5] E.Okamoto and K.Tanaka, "Key distribution system based on identification information," *IEEE Journal on Selected Areas in Communication*, vol. 7, pp. 481–485, May 1989.

[6] S.Tsujii and T.Itoh, "An id-based cryptosystem based on the discrete logarithm problem," *IEEE Journal on Selected Areas in Communication*, vol. 7, pp. 467–473, May 1989.

[7] U.M.Maurer and Y.Yacobi, "Non-interactive public-key cryptogaphy," in *Eurocrypt91*, pp. 498–507, 1991.

[8] R.L.Rivest, A.Shamir, and L.Adleman, "A method for obtaining digital signatures and public key crytosystems," *CACM*, vol. 21, pp. 120–126, 1978.

[9] T.ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, July 1985.

[10] C.E.Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. XXVII, pp. 379–423, July 1948.

[11] W.Fumy and M.Munzert, "A modular approach to key distribution," in *Crypto90*, pp. 274–283, 1990.

[12] W. Stevens, *Unix Network Programming.* Prentice-Hall,Inc., 1991.

[13] H. C. van Tilborg, *An Introduction to Cryptology.* Kluwer Academic Publishers, 1988.

[14] B. Schneier, *Applied Cryptography.* John Wiley & Sons, Inc., 1994.