초기인증에서 키 분배 및 복구를 지원하는 공개키 암호 인증시스템에 관한 연구

A Study on Public Key Cryptographic Authentication System Providing Key Distribution and Recovery in the Initial Authentication

신 광 철* 조 성 제**
Kwang-Cheul Shin Sung-Je Cho

요 약

본 논문에서는 인증서를 사용하는 모든 암호시스템 분야에 응용될 수 있는 인증 및 키 복원 프로토콜로 PKINIT를 응용한 안전한 초기인증 공개키 암호시스템 모델을 연구하였다. 멤버들에 대한 인증은 서버와 사용자간에 인증서를 기반으로 공개키 암호에 의한 초기인증과 세션 키 분배, 응용자원 서버들과의 비밀통신에서 키의 유실을 고려한 키 복구지원 프로토콜을 제안하였다.

Abstract

In this paper, we improved a cryptography system model based on the secure initial authentication public key with PKINIT of authentication and key recovery protocol. It is applied to all fields of cryptography system using certificate. This study presents two mechanisms to authenticate between member users. The first mechanism is initial authentication and distribution of session key by public key cryptography based on certificate between entity and server, and the second mechanism is a key recovery support protocol considering loss of session key in the secure communication between application servers.

☞ Keyword : PKI, Kerberos, Authentication, PKINIT, Key Recovery

1. 서 론

사용자 인증과 디지털서명은 사용자나 시스템이 정당한 사용자인지, 동일한지를 검증하는 것이며 이를 위해 인증기관이 서명하여 공개한 인증서를 사용하고 있다[1]. PKI(Public Key Infrastructure)는 사용자의 공개키에 대한 인증서를 발행하고 관리하는 암호학적 키와 인증서의 배달시스템으로 여러 응용분야에서 인증서의 사용을 용이하도록 하는 정책, 수단, 도구 등을 수립하고 제공하는 개체들의 네트워크이다[2].

* 정 회 원 : 성결대학교 e-비즈니스 IT 학부 교수(제 1저자)
skcskc12@sungkyul.edu
** 정 회 원 : 성결대학교 e-비즈니스 IT 학부 교수(공동저자)
chosj@sungkyul.edu
[2005/06/27 투고 - 2005/07/04 심사 - 2006/01/05 심사완료]

본 논문에서는 PKI구조에서 X.509 인증서로 운용되고 있는 암호시스템에 PKINIT(Public Key Cryptography for Initial Authentication[3])를 적용하여 초기인증과 안전한 키 분배, 키 복구를 지원하는 공개키 인증시스템 모델을 설계하였다.

각 개체는 PKI를 기반으로 생성한 공개정보를 이용하여 사용자 개체와 인증서버(NAS:New Authentication Server)간의 신뢰성을 보장하고 인증에 필요한 공개키(subjectPublicKey)정보를 기초로 하여 Diffie-Hellman(이후 D-H로 표기) 키 교환방식을 이용함으로써 키 복구를 제공하는 PKI 연동 인증시스템이다. 사용자의 공개키를 인증하고 인증된 개체에 대해 세션 키를 발행함으로써 무결성과 안전성, 신뢰성을 갖는 안전한 서비스를 제공한다.

이러한 서비스를 제공하는 단계를 암호모듈을

통해 개인키와 공개키를 생성하는 공개키 등록과 인증서발행 단계, 공개키 암호를 이용한 초기 인증 단계, 데이터 암호전송 단계, 키 복구 메커니즘을 이용한 인증 서비스 단계 등 4개 과정을 수행하도록 제안하였다. 본 고에서는 기존 시스템의 기술과 제한점을 알아보고 3장에서 제안방식을 설명한다. 4장에서 제안방식의 안전성과 효율성을 분석하고 5장에서 결론을 맺는다.

2. 관련연구

2.1 Kerberos 메커니즘

초기에 Kerberos에서 사용하는 암호화키는 사용자의 평문입력에 의한 패스워드를 Kerberos 키로 사용하였으며 안전하게 분배된 공유의 비밀키로 상호인증을 실행하였다. 많은 서버 서비스를 단순히 패스워드를 이용한 키 방식만을 사용한다면 모든 서비스마다 동일한 키를 사용하는 결과를 초래한다. 또한 사용자의 서비스요구 시마다 빈번한 패스워드 입력과 평문전송을 한다는 것은 매우 부담스런 일이다. 이를 해결할 수 있는 방안으로 TGS(Ticket Granting Server)에서 발급하는 Ticket을 사용하는 것으로 최초 서비스에 접근하는 메시지만 패스워드를 사용하고 있다. 패스워드를 키로써 사용하는 것이 아니고 해쉬(hash)함수를 이용해서 산출된 값(digest)을 키로써 사용한다[4].

KDC(Key Distribution Center)에서는 사용자 패스워드의 해쉬값을 저장해 두고 있기 때문에 사용자의 사전인증 데이터가 일치한다면 정상적으로 사용자는 로그온 할 수 있다. 또 사용자의 패스워드가 확인되면 TGS에 접근하여 Ticket 발행을 요청할 수 있도록 KDC는 TGT(Ticket Granting Ticket)를 발행한다.

이 TGT를 다른 제 3자가 열어볼 수 없도록 KDC 자신의 long term key로 암호화하게 되며 사용자가 server의 TGS와 통신할 때 사용하게 될 세션 키를 발행하여 사용자의 비밀키(k_c)로 암호화하여 전송한다. Ticket은 도용을 대비한 timestamp와 lifetime 적용하고 변조방지를 위한 KDC와 TGS간의 공유키로 암호화하며 client의 식별을 위한 인증자(Authenticator)를 사용한다.

비밀키를 전송하여 프라이버시를 유지해 줄 수 있으며 서버는 client가 유용한 티켓을 가지고 접근했기 때문에 신뢰하게 되고 서비스를 제공할 수가 있다. Kerberos가 가지는 제한점은 다음과 같다. 첫째, Ticket 발행에 의해 사용자 인증은 실현하고 있으나 디지털서명기능은 제공하지 못하고 있다. 둘째, Kerberos 서버의 보안 문제로 상호 서버간 완전한 신뢰를 가정하고 있으며 패스워드에서 유도된 long term key를 client와 Kerberos가 비밀키로 공유하기 때문에 사전공격에 취약하다. 셋째, IETF CAT에서 PKINIT에 의한 공개키 사용의 대한 언급만 했을 뿐 구체적인 메커니즘이 제시되지 않고 있다[5-7]. 넷째, 각 유형별 서비스마다 각기 다른 세션 키를 사용하게 되는데 이 세션 키들의 유실에 대비한 키 복구지원기능이 없다.

2.2 Yaksha 메커니즘

Yaksha는 Kerberos 형태를 유지하면서 디지털 서명을 가능하게 하고 패스워드에 의한 사전공격의 문제점을 해결하기 위하여 Kerberos 시스템에 인증기관(CA)을 둔 공개키 기반으로의 응용이라고 볼 수 있다[8]. 암호 키는 개인키와 공개키를 사용하기 때문에 디지털 서명과 키의 공유, 암호화 통신을 제공한다. Kerberos 기반 시스템으로 공개키 암호를 사용함으로써 다음과 같은 Kerberos의 약점을 보완하는 프로토콜이다. 첫째, Kerberos 인증시스템은 대칭키 암호시스템이기 때문에 신뢰하는 중앙 Kerberos 서버가 Client에 대한 비밀키 보관으로 노출은 불리한 결과를 초래할 것이다. 둘째, 패스워드 추측에 의한 사전공격을 당하기 쉬울 것이다. 셋째, 서비스

에 대한 부인봉쇄 기능이 없어 디지털서명을 제공하지 못하고 있다.

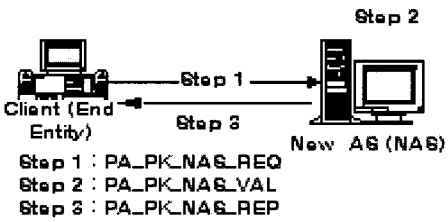
이와 같은 Kerberos의 단점을 보완하기 위해 client와 Yaksha Server(AS in Kerberos)간에 RSA 비밀키 d를 d_c와 d_{cy}로 나누어 공유한다. 이러한 방법으로 인해 Yaksha System은 키 위탁(Key escrow scheme)으로 사용될 수 있으며 공개키 암호를 뒷받침하기 위해 각각의 메시지는 확장이 가능하다.

3. 공개키 암호 인증시스템 설계

3.1 멤버 초기인증과 키 교환 알고리즘

본 절에서는 도메인 멤버에 대한 공개키 인증, 개인 키 소유에 대한 증명과 초기 세션 키 분배를 목적으로 한다. X.509 인증서로 운용되고 있는 모든 암호시스템에 적용시킬 수 있도록 공개 값을 이용하여 사용자 개체와 인증서버(NAS)간의 인증과 교환정보를 기초로 인증서버는 D-H 키 교환 방식을 이용하여 세션키를 생성, 분배하고 각 개체는 공개키 암호로 세션키 요청 메시지에 D-H 비밀 값과 자신의 공개키를 개인키로 서명하여 전송한다. 초기인증에서 안전한 키 교환 및 복구는 공개키 암호에 의한 인증과 NAS가 각 개체와 교환을 위해 생성하는 세션 키, 영역의 멤버들과 NAS간 공개 합의 하에 사용하는 D-H키 교환(세션 키 보호)에 의해 이루어진다. 그러나 서비스 유형이나 서비스 세션마다 상이한 세션키를 사용하기 때문에 키를 유실하는 상황이 발생하면 중요한 데이터에 대한 접근이 불가능하다는 위험이 있다.

이를 해결하기 위해서는 개체와 Application Server간 비밀통신을 보장하면서 키 분실과 같은 상황이 발생할 경우 세션 암호 키를 복구할 수 있도록 인증서버의 D-H 키 교환 정보를 활용한다. 공개키 생성과 인증과정을 통하여 도메인내의 각 개체들은 공개키를 등록하고 NAS로부터 인증과 함께 생성된 세션 키 개체들은 보유하게 된다. 과정은 공개키에 의한 인증서비스와 NAS가 세션 키와 D-H 세션 키를 생성하여 분배하며 공개키 초기인증(PA_PK_NAS_REQ)에서 영역에 등록하는 모든 개체가 생성하여 전송한 공개 값(subjectPublicValue)과 인증 결과로 entity의 서명정보(SigAuth-Pack)를 인증서 저장소에 저장한다.



〈그림 1〉 공개키 인증 및 키 분배

그림 1은 도메인내의 모든 entity들에 대한 공개키 초기인증 서비스를 위한 처리절차와 Entity로부터 인증요청, 공개키 인증(서명), 세션키 및 D-H 공유키 생성, 세션 키 분배 과정에 대한 키 교환 알고리즘을 도시하였다.

3.2 공개키 암호 인증 및 세션 키 분배

공개키 암호로 메시지에 암호데이터와 공개키 서명을 이용하여 사용자 인증을 함으로써 인증기관에 안전하게 접근할 수 있다. 절차는 entity가 서명된 공개키와 공개키 인증정보를 전송하여 NAS로부터의 인증과 함께 세션 키를 D-H의 공개 값이나 비밀 세션 키에 대해 선택적으로 요구한다. NAS는 수신한 메시지를 인증자의 공개 키 정보와 디렉터리 서버의 인증서 정보를 검증하고 D-H 공개정보와 세션 키를 응답한다.

3.2.1. 시스템 계수

: 개체

A_I : 알고리즘 식별자

$Authenticator_c$: entity(client)의 인증자

Authpack : 재전송 방지 값을 포함한 NAS 및 entity의 공개정보

$Dk_{\#,\#}$: #개체와 # 개체간의 D-H 키 교환에 의해 생성된 공유키

$ID_{\#}$: #의 식별자

$k_{\#,\#}$: 해당 #개체와 #개체간의 세션 키

PE[] : 공개키 암호알고리즘(RSA)으로 생성된 암호문

$pk_{\_\#}$: # 해당개체의 공개키 (pk_c : 개체 c의 공개키)

SE[] : 대칭키 암호알고리즘(DES)으로 생성된 암호문

Sig[] : 공개키 암호알고리즘(RSA)으로 생성된 서명문

$sk_{\_\#}$: # 해당개체의 개인키 (sk_c : 개체 c의 개인키)

$xi_{\_\#}$: # 개체의 비밀 값

$yi_{\_\#}$: # 개체의 공개 값

3.2.2. Entity Request(STEP 1 : PA_PK_ NAS_REQ)

공개키 인증 및 세션 키 요청에서 EE(End Entity)의 공개키와 암호알고리즘을 서명하고 공개키 인증정보를 NAS의 공개키를 이용하여 전송한다. 초기인증은 EE의 공개키를 인증하고 NAS로부터 세션 키를 교환하는 절차이다. EE는 그림 1의 Step 1과 같이 NAS의 공개키를 이용하여 PA_PK_NAS_REQ 메시지를 전송한다.

Entity ----> NAS : PE_{pk_nas}[Sig_{sk_c}[SigAuth-pack], Authpack, TrustedCertifiers]

- SigAuth-Pack : [IDc, xi_c, A_I] /* entity의 비밀정보와 알고리즘 식별자로 공개정보 생성과 정확성을 인증하는 요소 */
- Authpack : [pkAuthenticator, entityPublic-Value] /* 공개키 인증자(pkAuthenticator)와 Diffie-Hellman 암호를 사용하기 위한 파라미터 값(entityPublicValue) */
  - pkAuthenticator : [realm, cusec, ctime, nonce, pachecksum] /* 인증자의 공개키 정보로써 cusec(entity time), ctime(server time), nonce(재전송 방지 위한 난수), pachecksum(sha1 or rsa-md5) */
  - entityPublicValue : [algorithm, subjectPublicKey] algorithm : [algorithm, parameters] / * algorithm(OID)과 parameters(prime(p), base(g), length) */subjectPublicKey /* 소수 p의 원시근 g에 대한 public exponent 산출 $yi\_c = (g^{xi\_c}) \bmod p$ */
- TrustedCertifier : [caname, issuerandserial] /* NAS가 보증하고 신뢰하는 인증 데이터로 caname(X.509에 의해 정의된 X.500 full name)과 issuerandserial(entity를 신뢰할 수 있는 이미 부여된 NAS 발행의 인증서 일련번호 */

3.2.3. NAS Validation(STEP 2 : PA_PK_ NAS_VAL)

인증 메시지(PA_PK_NAS_REQ)를 수신한 NAS는 사용자의 인증정보를 검증하기 위해 개인키의 생성여부, 공개키 인증자 정보, 자신이 발급한 인증서 일련번호를 검증서버(Validation Server)를 통해 그림 1의 Step 2를 통하여 확인한다.

검증내용은 CA의 인증서 서명여부, SigAuth-Pack 검증실패, 인증서 유효기간, 인증서의 취소여부, 인증서의 entity이름, entity의 서명검증, NAS와 entity의 time 들을 말한다. entity의 subjectPublicKey로 D-H 세션 키 $Dk_{c,nas} = (g^{xi_c})^{xi_nas} \bmod p$와 세션 키($k_{c,nas}$)를 생성하고 NAS의 세션키 저장소에 사용횟수, 유효기간, 재사용 방지를 위한 생성시간을 보관한다.

3.2.4 NAS Response(STEP 3 : PA_PK_ NAS_REP)

NAS는 entity에서 제공한 공개키 인증자의 검증결과로 자신과 entity가 공유할 세션 키와 D-H 정보를 생성하여 entity의 공개키로 (그림 1)의 Step 3과 같이 응답한다.

NAS ----> Entity : dhSignedData, encKeypack

- dhSignedData : [subjectPublicKey, nonce, dhKeyExpiration] /* D-H 암호방식을 사용할 경우 NAS에서 인증한 D-H 파라미터 제공 */
  -subjectPublicKey /* yi_nas = g^{xi_nas} mod p */
  -nonce /* 재전송에 의한 공격이 아님을 확인하는 난수 */
  -dhKeyExpiration /* D-H 키의 유효시간 */
- encKeyPack : PE_{pk_c}[nonce, $SE_{Dkc,nas}$[$k_{c,nas}$, Expiration, Trans-number] /* entity의 공개키를 이용하여 세션 키를 암호화하며 dhSignedData를 다시 세션 키로 암호화한 enveloped data */
  -Expiration /* 세션 암호 키의 유효시간 */
  -Trans-number /* NAS의 정책에 따라 세션 암호 키를 사용할 수 있는 처리횟수 지정 */

NAS는 세션 암호 키(kc,nas)와 재전송 방지 응답용 값(nonce), 세션 암호 키의 만료시간을 나타내는 Expiration과 키의 사용횟수로 NAS의 정책에 따라 결정하는 Transaction number를 세션 키로 암호화하고 entity의 공개키로 암호화한 encKeyPack을 전송함으로써 entity의 인증과 함께 암호 세션 키를 보유하게 된다.

그림 1의 [step 3]을 수신한 EE는 encKeyPack을 복호하기 위하여 자신의 개인키를 사용하며 세션키를 알아내기 전에 D-H 키를 생성하여야 한다. D-H 키는 NAS의 subjectPublicKey를 이용한다.

이 세션 키는 만료시간이 경과되면 새로운 세션 키를 NAS로부터 발급 받기 때문에 주기적으로 갱신이 이루어진다. 이와 같이 사용자에 대한 공개키 인증과 D-H 키 생성, 세션 키 분배 후 각 개체는 entityPublicValue에 의해 키를 생성할 수 있는 정보를 보유[표 1]하게 된다.

모든 멤버들과 공개키 암호에 의한 초기인증이 완료되면 NAS는 세션키 저장소에 생성한 D-H 공유키와 D-H Value 등을 인증결과로 저장하며 저장된 D-H value는 사용자가 응용서버와 통신할 때 세션 키를 생성하는 인자이며 NAS가 사용자의 키 분실 시 복구해 주는 역할을 한다. 인증서 저장소는 검증서버에 의해 수집된 경로와 인증서, 개체의 식별자 등을 저장한다.

3.3 응용개체 간 키생성 및 분배

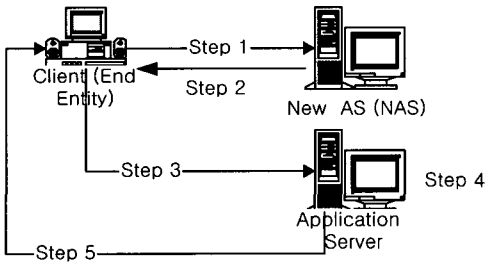
Entity가 Application Server와의 통신을 하기 위해서는 NAS의 승인을 받아야 한다. 이 때 NAS는 Entity와 Application Server와 안전한 통신을 위한 세션 키 생성인자(A, B)를 초기인

〈표 1〉 개체들이 보유한 키 생성정보

개체	DH 공개정보 (yi_#)	다른 개체 보유정보					
		개체	비밀정보(xi_#)	공개정보(yi_#)	prime	base	생성
NAS	g^{xi_nas} mod p	entity	PE_{pk_nas}[xi_c]	g^{xi_c} mod p	p	g	.
		entity1	PE_{pk_nas}[xi_c_1]	g^{xi_c1} mod p	p	g	.
		:	:	:	:	:	:
entity	g^{xi_c} mod p	NAS	.	g^{xi_nas} mod p	p	.	.
Server	g^{xi_s} mod p	NAS	.	g^{xi_nas} mod p	p	g	g^r

중단계에서 각 entity로부터 전송받은 entity-PublicKey에서 생성하여 전송(그림 2의 Step 2)한다.

EE는 Application Server와의 세션 키를 생성하기 위해서 상대방 생성인자 (B)를 보관하고 자신의 생성인자 (A) 값을 전송(그림 2의 Step 3)한다.



Step 1 : Request
Step 2 : A , B
Step 3 : A
Step 4 : $(A)^{xi\_s} \mod p$
Step 5 : g^r

〈그림 2〉 키 생성 및 분배
〈Fig. 2〉 Session Key Generation and Distribution

3.3.1 초기설정

NAS는 전체 공개 값으로 p와 g를 중앙 디렉토리의 세션 키 저장소에 저장하고 있다. 공개키 초기인증과정(PA_PK_NAS_REQ)에서 영역에 등록하는 모든 개체가 생성하여 전송한 subject-PublicValue 정보를 활용한다. 각 개체들은 키 복구를 위한 D-H 공개키(값)을 (표 1)과 같이 보유하고 있으며 NAS는 모든 개체의 D-H 공개값을 등록 받아 보유하고 있다.

3.3.2 키 생성 및 분배

- 각 개체는 비밀정보를 이용하여 D-H 공개정보 값을 계산한다.
- p는 1, 2, ..., p-1 범위의 Galois field 상에서 정의된 소수이다.
- g는 차수가 g인 곱셈군의 생성원을 나타낸다.

그림 2의 Step순 절차는 다음과 같다.

Step 1 : End Entity는 응용서버와의 통신을 위해 키생성 정보를 NAS에 요청한다.

Step 2 : NAS는 공개키 초기인증과정에서 entity와 server로부터 전송받은 subject-PublicKey에 자신의 비밀정보(xi_nas)를 지수승하여 $A=(g^{xi_c \cdot xi_nas} \mod p)$와 $B=(g^{xi_s \cdot xi_nas} \mod p)$를 생성하고 결과를 entity에 전송한다.

Step 3 : entity는 B에 자신의 비밀키를 지수승하여 구한 B'를 보관하고 값 A를 server에 전송한다.

Step 4 : Server는 A에 자신의 비밀키를 지수승하여 A'를 구하고 랜덤수 r를 구하여 생성원 g에 지수승한다.

- random number r 생성
- $g^r \mod p \quad (r \in_R Z)$

값 A에 g^r을 곱하여 세션키($k_{c,s}$)를 생성한다.

- $k_{c,s} = g^{xi\_s * xi\_nas * xi\_c} * (g^r \mod p) \mod p$
  $= g^{xi\_s * xi\_nas * xi\_c + r} \mod p$

Step 5 : Server는 생성원 g^r을 entity로 전송한다.

Entity는 보관중인 B값에 생성원 g^r을 사용하여 응용서버와 동일한 세션키를 생성한다.

- $k_{c,s} = (g^{xi\_s})^{xi\_nas})^{xi\_c} * g^r \mod p$
  $= g^{xi\_s * xi\_nas * xi\_c + r} \mod p$

4. 메커니즘 분석

4.1 보안에 대한 공격

일반적으로 정보의 흐름은 파일이나 주기억장

치의 한 부분과 같은 정보 출처로부터 정보 목적지로 이어지게 된다. 이러한 정보의 정상적인 흐름의 보안에 대한 위협은 메시지의 내용공개나 트래픽분석과 같은 데이터의 손실이 없는 소극적인 공격과 방해, 가로채기, 재전송, 불법수정과 같은 적극적인 공격으로 구분한다. Interception은 인증되지 않은 Diffie-Hellman 키 동의 상에서 이루어지는 공격이다.

Entity와 NAS은 각각 비밀값 xi_c, xi_nas을 보유하고 있다. 이 때 제3자는 xi_c'와 xi_nas'을 생성하여 g^{xi_c}, g^{xi_nas}을 $g^{xi\_c'}$, $g^{xi_nas'}$로 교체한다. 결국 entity가 보유하는 세션 키는 $Dk_{c,nas} = g^{xi_c \ + \ xi_nas'}$이 되고 NAS가 보유하는 키는 $Dk_{c,nas} = g^{xi_c' \ + \ xi_nas'}$이 되어 제3자는 이 키를 모두 보유하게 된다.

entity와 NAS는 비밀정보가 노출이 되지 않았기 때문에 안전하게 교환되었다고 믿게 된다. 초기인증요구의 PA_PK_NAS_REQ 단계에서 entity의 비밀정보 xi_c는 SigAuth-Pack에 포함되어 2중으로 암호화되어 있고 entityPublicValue 또한 AuthPack에 포함되어 암호로 보장된다. 응답의 subjectPublicValue는 노출되어 전송되지만 제3자가 entity의 비밀정보를 모르면서 D-H의 공유키인 $Dk_{c,nas}$과 $k_{c,s}$을 생성해 낼 수는 없다.

4.2 Forward Secrecy에 대한 안전성

Forward secrecy는 통신 대상의 개체 모두의 비밀키가 노출되더라도 공격자는 두 사용자 사이에 설정된 이전의 세션 키를 계산할 수 없는 경우로써 노출되는 사용자의 키에 따라 두 통신 대상자 중 한 사용자의 비밀키가 노출된 경우에만 세션 키가 안전한 경우인 half forward secrecy와 두 사용자의 비밀키가 모두 노출된 경우에도 세션 키가 안전한 경우인 full forward secrecy로 구분한다.

제안된 스킴에서 사용되는 세션 키의 안전성은 다음과 같다. 세션 키는 초기인증에서 NAS가 entity에 대한 인증이 이루어 진 후 D-H 키로 암호화되어 전송한다. 이 키는 다음 세션에서 사용할 Ticket의 성격을 가지며 padata를 암호화하기 위한 목적으로 분배된다. NAS가 세션 키 생성과정에서 사용자의 패스워드에서 추출한 long-term 비밀정보와 관련이 없는 임의의 세션 키로 쌍방간 비밀정보(xi_nas, xi_c)를 사용하여 D-H 공유키를 생성한다. Client와 응용 간에 실제 데이터를 암호화하기 위해 설정된 세션 키는 D-H키로 보호되며 client의 비밀 값(xi_c)과 응용서버의 비밀 값(xi_s)이 모두 노출이 되더라도 NAS의 비밀 값(xi_nas)을 알아야 한다. 따라서 두 사용자의 비밀정보가 모두 노출되더라도 공격자는 현재의 세션 키를 구하는 것은 불가능하므로 perfect forward secrecy하다.

4.3 사전공격

사전 공격은 비밀키 암호 알고리즘의 키를 client가 원하는 값으로 생성할 경우에 적용 가능한 공격 방법으로 사용자와 관련된 값을 차례대로 대입하여 키를 찾을 수 있는 확률이 매우 높다.

기존의 Kerberos system의 경우를 살펴보면 사용자들의 패스워드는 암호화되어 passwd 파일에 저장된다. 이 경우 사전 공격을 할 경우, 상당수의 사용자의 패스워드를 알아낼 수 있다. kdc.conf 파일에서 정의한 디렉터리 중 패스워드 DB로부터 사용자의 one-way hash가 유출된다면 brute force attack이나 사전 cracking공격을 당하게 된다.

제안된 시스템에서는 client에서 SigAuth-Pack의 서명과 NAS의 공개키로 암호화 하여 전송하는 PA_PK_NAS_REQ의 메시지는 padata로 client의 인증과 키 생성정보를 포함하고 있으며 NAS에 의해 생성한 세션 키를 D-H키와 client의 공개키로 암호화하여 전송함으로써 사전 공격의 대상이 되질 않는다.

4.4 비교분석

표 2는 인증서기반의 초기인증과 세션 키의 분배, 키 복원 프로토콜을 제안한 메커니즘의 결과를 기존의 인증시스템과 비교분석하였다.

〈표 2〉 제안시스템 비교
〈Table 2〉 Comparison of a Proposal System

구분	Kerberos	Yaksha	제안시스템
안전성	사전공격에 약함	사전공격 견딤	사전공격강함
서명	미 제공	제공	제공
인증서버 의존도	모든 비밀키 보관	비밀키 분리보관	비밀키 분리보관
기본방식	Needham-Schroeder Denning-Sacco	Kerberos와 공개키	PKINIT와 PKI
초기인증	패스워드	공개키사전등록	인증서
키 복구	기능없음	가능	가능

Yaksha는 Kerberos의 제한점을 보완한 새로운 공개키 개념의 인증시스템이다. 제안 논문에서는 공개키 시스템과 디렉터리서버, 검증서버를 운용함으로써 인증서 경로구축과 검증을 통해 다른 도메인에 대한 영역인증이 가능하고 새로운 멤버 가입에 따른 확장성이 용이하게 되었다.

4.5 키 복구 단계

사용자는 데이터를 암호화한 키가 분실되었을 때 복구필드와 대응되는 인증정보와 함께 NAS에 전송한다. entity의 키 복구는 NAS의 비밀키가 있어야 해결할 수 있다. 키를 복구해야 하는 상황이 발생할 경우 entity의 요청에 의해 다음과 같이 수행된다.

- entity는 server로부터 전송받은 g^{xi_s} 값에 자신의 비밀키를 지수승하여 C를 NAS로 전송한다.
- NAS는 전송된 값 C에 자신의 비밀키로 지수승한 D를 entity로 전송한다.
- entity는 D에 server로부터 전송된 g^r을 곱하여 $k_{c,s}$를 생성한다.

- $k_{c,s} = (((g^{xi\_s})^{xi\_c} \mod p)^{xi\_nas} \mod p$
  $= g^{xi\_s * xi\_nas * xi\_c + r} \mod p$

5. 결 론

본 논문에서는 공개키 기반구조의 인증서를 사용하는 암호시스템에 적용할 수 있도록 키 복구를 지원하는 PKI 연동 새로운 인증시스템을 제안하였다.

암호알고리즘으로 관용키 암호방식의 DES와 공개키 암호방식 및 디지털 서명 알고리즘으로 RSA을 적용하여 설계되었으며 PKINIT에서 초기 개체들은 인증과정에서 D-H 공개정보를 제공함으로써 세션 키 생성과 키 복구 메커니즘을 설계하는데 활용된다. 기존의 인증방식과 비교하여 사전공격에 강하고 암호 키에 대한 정책을 설정하여 키의 비도를 높였다. 사용자의 인증정보는 사용자의 공개키를 서명한 메시지와 인증서를 비교함으로써 제3자는 정당한 사용자로 사칭할 수 없다. 데이터 암호화 세션 키로 Diffie-Hellman 키 교환 방식을 이용하여 세션 키를 제공함으로써 키를 유실하는 상황이 발생했을 때 인증기관(NAS)에 의해 복구가 가능하고 안전한 서비스를 지원하는 동일 도메인 인증과 외부 도메인 간 효율적인 인증모델을 제시하였다.

본 제안 논문은 공개키 암호의 대중성과 패스워드 추측에 의한 사전공격에 대한 보호, 분산 네트워크 환경으로의 적용, 키 관리의 단순화, 좀더 단순하게 상호영역의 인증, 디지털 서명 등을 충족시킨다. 향후 연구과제로 서로 다른 다중영역(Multi-Domain)간의 효율적인 인증메커니즘을 필요로 한다.

참 고 문 헌

[1] Kwangcheul Shin, Jinwook Jung, Ilyong Chung, "An Efficient Kerberos Authentica-

tion Mechanism Associated with Directory System", proceeding of the international conference on security and management, SAM02, p369-375. June, 2002

[2] IETF Draft, "Internet X.509 Public Key Infrastructure Certificate and CRL profile", 1998

[3] RFC 1510, Public Key Cryptography for Initial Authentication in Kerberos, draft-ietf-cat-kerberos-pk-init-26.txt, IETF, 2005

[4] J. G. Steiner, B. C. Neuman, and J. I. Schiller, "Kerberos: An Authentication Service for Open Network System," pp. 191-202 in Usenix Conference Proceedings, Dallas, texas (Feb, 1988)

[5] B. Tung, B.C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky "Public Key Cryptography for Cross-Realm Authentication in Kerberos". draft-ietf-cat-kerberos-pk-cross-08.txt

[6] A. Medvinsky, M. Hur, S. Medvinsky, C. Neuman. "Public Key Utilizing Tickets for Application Servers (PKTAPP)

[7] A. Harbitter and D. Menasce, "Performance of Public Key Enabled Kerberos Authentication in Large Networks." Proc. 2001. IEEE Symposium on Security and Privacy, Oakland, CA, May 13-16, 2001

[8] R. ganesan, "Yaksha : Augmenting Kerberos with Public Key Cryptography," Proc. of ISOC Symposium on Network and Distributed System Security, pp. 132-143, 1995

◐ 저 자 소 개 ◑

신 광 철 (Kwang-Cheul Shin)
성균관대학교 대학원 정보공학과 졸업(공학박사)
前 육군전산소 프로그램 및 시스템분석장교
前 육군대학 전쟁연습프로그램 및 전산실장
前 벽성대학 소프트웨어 개발과 교수
現 성결대학교 e-비즈니스 IT학부 교수
E-mail : skcskc12@sungkyul.edu

조 성 제 (Sung-Je Cho)
1997년 : 홍익대학교 대학원 전자계산학과 졸업(이학박사)
1995년-2000년: 前 경주대학교 컴퓨터전자공학부 교수
2005 3월~현재 : 現 성결대학교 e-비즈니스 IT학부 교수
E-mail : chosj@sungkyul.edu