공개키 알고리듬을 이용한 분산 컴퓨터망에서의 인증 방식

염 흥 열*, 백 종 현*, 김 창 련**, 김 석 우**

순천향대학교 전자공학과*, 한국전자통신연구소**

Authentication for Distributed Computer Network Using Public Key Algorithm

Heung-Youl,Youm*, Jong-Hyun,Baek*, Chang-Ryeon,Kim**, Seok-Woo,Kim**

Dept. of Electronic Eng., Soonchunhyang Univ.*,
Electronics and Telecommunications Research Institute**

-요약-

분산 컴퓨터망에서의 고객과 서버간의 서비스는 인증이 완료된 후 제공되어야 한다. 분산망에서의 지금까지 알려진 대표적인 인증 시스템은 Kerberos 인증 기법이다. 현재 Kerberos 인증 방식에서는 대칭형 알고리듬인 DES 알고리듬과 패스워드에 바탕을 두고 있다. 그러나 여기서는 DES 의 채용으로 인한 안전성 문제와 영역(Realm) 간 인증 정보교환시 복잡한 키 관리가 요구되는 단점이 있다. 본 논문에서는 Kerberos 인증 방식이 갖는 문제점을 분석하고, 이를 바탕으로 공개키 알고리듬을 이용한 인증 구조를 제안한다. 또한 IDEA 를 이용한 인증 방식과 공개키 암호 알고리듬을 이용한 인증 방식을 시뮬레이션한다. 시뮬레이션에 이용된 C 루틴은 64 비트 IDEA 루틴, 128 비트 MD5 루틴, 고속의 지수 연산 루틴, 768 비트 지수 연산 루틴, 그리고 RSA 암호키 생성 루틴 등이다. 본 논문의 결과는 분산 컴퓨터 망에서의 인증 시스템 설계시 유용하게 활용될 수 있을 것이다.

제1장 서론

컴퓨터 통신망에서의 서버는.다수의 원격 사용자에게 서비스를 제공해야 하고 서비스 요청이 통신망을 통해 수행되므로, 서비스를 요구하는 사용자의 신원을 정확히 확인할 수 있는 능력을 가져야 한다. 인증(Authentication) 은 사용자의 정체(Identity) 을 확인하는 과정이다. 지금까지 널리 이용되었던 인증 방식은 패스워드를 이용하여 사용자의 신원을 인증받는 방법이 있으나, 이는 중간의 불법 사용자가 패스워드를 도청하여 추후에 적법 사용자를 가장하는 공격이 가능해진다. 따라서 좀더 안전하고 강력한 인증 방식이 요구되어 지고 있다. 공개된 통신망에서 고객(Client) 과 서버간의 대표적인 인증 서비스는 MIT 에서 개발된 Kerberos 이다.[1] 공개 분산 환경은 고객의 워크스테이션, 공개된 통신망, 그리고 서비스를 제공하는 서버로 구성된다. 공개 분산망에서는 안전성 측면에서 세 가지 위협이 존재한다. 첫째, 불법의 사용자가 특정 워크스테이션 액세스를 얻어 워크스테이션내의 다른 적법의 사용자를 가장하는 위협이다. 둘째, 사용자가 자신의 워크스테이션 IP 주소를 변경하고, 변경된 워크스테이션에서 다른 워크스테이션을 가장하는 공격이다. 셋째, 사용자가 다른 사람의 인증 정보를 도청하고 추후에 도청된 인증 정보를 이용하여 서버의 서비스 권한을 얻는 재생 공격이다. 분산망에서의 인증 시스템은 위의 세가지 위협 요소에 대한 대비책이 강구되어야 한다. Kerberos 는 대칭형 암호 방식인 DES를 이용하고, 중앙 집중화된 인증 서버(Centralized Authentication Server) 를 제공하며, 인증 서버는 서버가 서비스를 요청하는 사용자를 인증할 수 있게 하는 인증 정보를 제공하는 역할을 수행한다. Kerberos 에는 버전 4와 버전 5 의 두 가지 버전이 발표되어 있다. 버전 4는 현재 널리 적용되고 있는 방식이고 버전 5는 RFC 1510으로 드래프트 표준화되고 있다. Kerberos 인증 방식은 기본적으로 DES 와 비밀키 분배 방식을 이용하고 있다는 것에 있다. DES 는 많은 암호학적 취약점이 발표되고 있고, 영역이 다른 사용자가 다른 영역에 있는 서버를 이용하고 N 개의 영역이 있을 경우 ${}_NC_2$ 개의 비밀키의 사전 교환이 인증 시스템간에 요구되어 이들 교환된 비밀키의 분배 및 관리가 큰 문제로 대두되었다.[2,3,4,5]

본 논문에서는 2장에서 기존의 Kerberos 인증 시스템을 분석하고 기존의 Kerberos 인증 시스템의 문제점을 분석한 후, 이를 해결하기 위한 공개키 암호 및 서명 알고리듬을 이용한 분산망에서의 인증 방식을 제안

하고 관련 특성을 분석한다. 3장에서는 DES의 암호학적 단점을 극복하기 위하여 IDEA 를 이용한 분산망에서의 인증 방식을 시뮬레이션하고, 또한 공개키 암호 및 서명 알고리듬을 이용한 분산망에서의 인증 방식을 시뮬레이션한다.

제2장 분산망에서의 인증 방식 및 공개키 알고리듬을 이용한 인증 방식

2.1 인증 방식

인증은 개체의 신분 확인(Verification of Identity) 과 개체가 생성한 데이터의 무결성(Integrity) 을 확인하는 과정으로 구분된다. 신분 확인은 신원을 검증받는 인증자(Prover) 와 인증자의 신원을 확인하는 검증자(Verifier) 간에 수행된다. 데이타의 무결성은 수신된 데이타가 생성된 데이타와 동일하다는 것을 확인하는 과정으로서, 이는 디지틀 서명 알고리듬을 이용하여 실현된다. 본 논문에서의 인증은 신분 확인만을 의미한다. 인증 메커니즘은 다시 보장 정도, 검증자의 수, 그리고 부인 봉쇄를 제공하는가에 따라 구분된다. 보장 정도는 데이타가 과거 어떤 시점에서 인증자에 의해 생성됐다는 사실만을 나타내거나, 인증자가 메시지를 보내고 있는 시점에 존재하고 있다는 것을 나타내거나, 또는 데이타가 인증자에 의해 매번 새롭게 생성되고 있다는 사실을 나타내는 정도에 따라 구분된다. 검증자의 수에 따른 분류는 메세지당 단일 검증자 인증 방식과 메세지당 다중 검증자 인증 방식으로 구분된다. 또한 메카니듬이 부인 봉쇄를 지원하는지의 여부에 따라 구분된다.

인증 방식에서 이용되는 기밀성 서비스는 불법의 도청자로 부터 정보를 보호하는 데 이용된다. 액세스 제어(Authorization) 는 동작 수행이 허용되었는가를 판단하는 과정이고, 일반적으로 인증 후에 수행되며, 지역 검증 정보, 또는 다른 인증 정보에 기초를 두고 수행된다. 대화식 서비스에서의 실시간 인증은 원격 로그인, 파일 시스템 R/W, 그리고 응용을 위한 정보 검색 시스템에서 이용될 수 있다.

2.2 Kerberos 인증 방식

Kerberos 인증은 패스워드 도청에 의한 누설과 사용자가 매번 패스워드를 입력해야 하는 문제점을 해결하기 위하여 MIT 에서 개발된 일반적인 인증 시스템이다. Kerberos 는 분산망에서의 인증 시스템으로서, 프로세스인 고객(Client) 이 특정 사용자를 대신하여 검증자에게 사용자의 신분을 확인하기 위한 일련의 암호화된 메시지를 교환하는 과정이다. 현재 많은 시스템에서 Kerberos 버젼 4(V4) 를 사용하고 있으나, 1989년 연구를 개시하여 현재 Draft 표준화 상태인 Kerberos 버젼 5 (V5)의 적용이 점차 증가할 것이다.[4,6,7]

Kerberos 가 가져야 할 요구 조건은 다음과 같다. 첫째, 네트워크 도청자가 사용자를 가장해서 필요한 정보를 얻을 수 없어야 한다. 둘째, Kerberos 서비스의 유용성 결여는 전체 서비스의 결여를 의미하므로 충분히 신뢰성 있게 동작해야 한다. 셋째, 사용자에 인증 과정의 부가로 인해 패스워드 입력 외에 별도의 정보를 요구하지 않아야 하는 투명성이 제공되어야 한다는 것이다. 넷째, 많은 고객과 서버를 지원할 수 있는 규모의 가변성에 대한 요구이다.

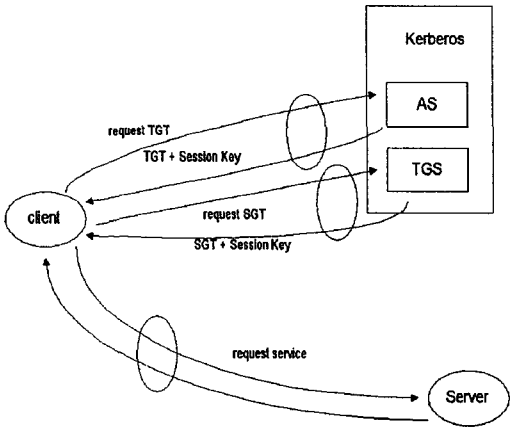

그림 2.1 전체적인 인증 구조

Kerberos 프로토콜은 근본적으로 Needham과 Schroder 인증 프로토콜에 기초를 두고 있고, 기본적으로 인증을 2회 인증 정보 교환으로 인증이 완료되기 위하여 타임스탬프 방식을 적용하였다. 또한 영역이 다른 사용자 및 서버 사이에 수행되는 영역간 상호 인증을 허용하고 있다. Kerberos 인증 프로토콜은 다음의 사항을 가정하고 구성된다. 첫째, AS는 모든 사용자의 패스워드들을 알고 있고, 둘째, AS는 중앙 집중화된 데이타베이스에서 모든 사용자의 패스워드에 대한 해쉬값을 안전하게 저장해야 하며, 셋째, AS는 각 서버와 인증을 위한 별도의 비밀키를 공유해야 한다. 이는 물리적으로 분배되거나 스마트 카드 형태로 분배될 수 있다. Kerberos 버전 4의 전체적인 인증 구조는 그림 2.1과 같고, 이를 구체적으로 실현하기 위한 인증 프로토콜은 그림 2.2와 같다.

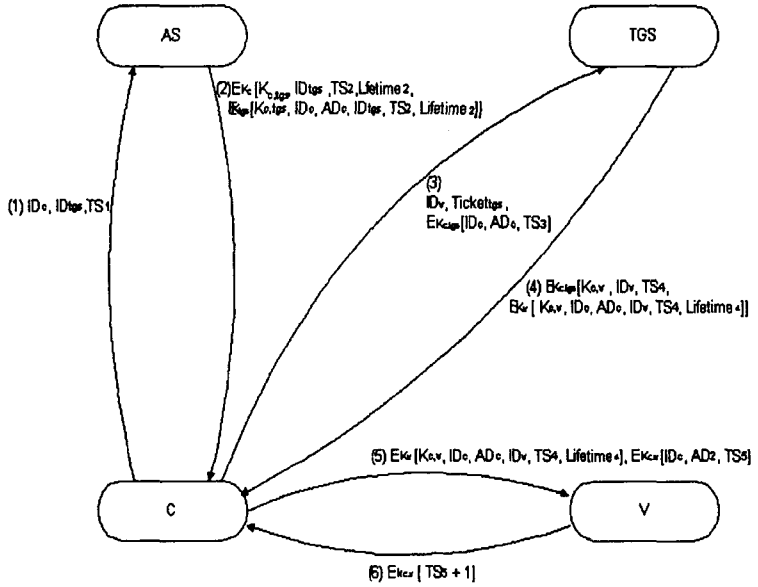

그림 2.2 V4 인증 프로토콜

그림 2.1과 같은 Kerberos 전체 인증 구조는 고객이 포함되어 있는 WS(Workstation), TGT(Ticket Granting Ticket) 를 생성하는 AS(Authentication Server), SGS(Service Granting Ticket) 를 발행하는 TGS(Ticket Granting Server), 그리고 구체적으로 서비스를 제공하는 서버로 구성된다. 이의 전체 흐름도는 다음과 같다. ① 사용자가 WS 에 로그인하면서 파일 전송 서비스를 요구한다. 그러면 WS 는 AS 로 TGT 를 요청한다. ② AS는 데이타 베이스에서 사용자의 액세스 권한을 검증하고, TGT($Ticket_{tgs}$) 와 고객과 TGS 세션키를 생성하여, 사용자의 패스워드를 일방향 해쉬 함수에 적용하여 구한 사용자 키로 암호화하여 전달한다. ③ WS 는 사용자 키를 이용하여 TGS로 부터 온 암호문을 복구한 후, 고객과 TGS간의 세션키와 TGT를 복구한다. 이는 하루에 한 번 수행된다. WS는 TGT 와 인증자를 TGS 로 전송한다. ④ TGS는 이를 수신하여 TGT 의 유효성을 검증한 후, TGT 에서 구한 세션키를 이용하여 인증자의 유효성을 검증한다. 둘의 유효성 검증이 완료되면 해당 서버를 확인한 후, SGT(Service Granting Ticket, $Ticket_v$) 와 서버와 고객간의 세션키를 고객과 TGS 간의 세션키로 암호화하여 고객으로 전송한다. ⑤ 이를 수신한 고객은 TGS 와의 세션키를 이용하여 암호문으로 부터 SGT 와 고객과 서버간의 세션키를 복구한다. 그리고 SGT 와 고객과 서버간의 세션키를 이용하여 구한 인증자를 서버로 전송한다. ⑥ 이를 수신한 서버는 SGT 에서 서버와 고객간의 세션키를 복구하고 SGT 의 유효성을 확인한 후, 세션키를 이용하여 인증자의 유효성을 확인한다. 이런 과정을 통해 서버는 고객의 정체를 확인하는 인증 과정을 완료한다. 그리고 고객의 서버의 인증을 위하여 서버의 인증자를 생성하여 SGS 로 이를 전송한다. ⑦ 이를 수신한 고객은 수신된 서버의 인증자를 검증함으로서, 고객에 의한 서버의 정당성을 확인한다.

그림 2.2와 같은 인증 프로토콜에서 인증 시스템의 구성 요소간의 교환되어야 할 인증 메시지는 여섯 가지 인증 메시지가 있다. 첫 번째 메시지는 고객이 AS로 전달하는 인증 요구(AR:Authentication Request) 정보

로서, TGS에 TGT를 요구하기 위한 메시지이다. 이는 인증 요구 정보는 식 (2.1) 과 같다.

$$ID_c,\ ID_{tgs},\ TS_1 \qquad (2.1)$$

여기서, ID_c 는 사용자 신분을 알리기 위한 ID 이고, ID_{tgs} 는 액세스를 원하는 TGS의 ID 이며, TS_1 은 고객의 클럭이 AS의 클럭과 동기되어 있는 정도를 알리기 위한 클럭 정보이다. 이를 이용하여 AS는 AR 정보가 시기 적절하다는 것을 확인한다. 그리고 AS 는 식 (2.2)와 같은 인증 응답(AR:Authentication Response) 정보를 고객으로 전송한다.

$$E_{K_c}[K_{c,tgs}, ID_{tgs}, TS_2, Lifetime_2, E_{K_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, Lifetime_2]] \qquad (2.2)$$

$K_{c,tgs}$ 는 고객과 TGS간의 세션키이며, 고객과 TGS 간의 안전한 데이타의 교환을 가능케 하기 위해 AS 에 의해 선택된다. $K_{c,tgs}$ 는 K_c 로 암호화된 메시지 내부에 있기 때문에 고객만이 $K_{c,tgs}$ 를 알 수 있다. $K_{c,tgs}$ 는 TGT($Ticket_{tgs}$) 내에도 포함된다. K_c 는 사용자의 패스워드로부터 유도된 암호키로서, 패스워드를 일방향 해쉬 함수로 적용하여 구한 결과이다. 이는 고객과 AS 만이 공유하고 있다. ID_{tgs} 는 서비스를 제공받기 위한 TGS의 ID 이다. TS_2 는 TGT 가 발행된 시간을 의미한다. $Lifetime_2$ 는 TGT 의 수명을 의미한다. AD_c 는 TGT 를 사용할 WS 의 망 주소이다. 식 (2.2) 의 후반부는 TGT 를 의미한다.

이를 수신한 고객은 AR 로부터 고객과 TGS 간의 세션키 $K_{c,tgs}$ 와 TGT 를 복구한다. 고객은 이를 이용하여 식 (2.3)과 같은 티켓 요구 (TR:Ticket Request) 정보를 생성하여 이를 TGS 로 전송한다.

$$ID_v,\ Ticket_{tgs} = E_{Ktgs}\ [K_{c,tgs}\ ,\ ID_c\ ,\ AD_c\ ,\ ID_{tgs}\ ,\ TS_2\ ,\ Lifetime_2\ ],$$
$$Authenticator_c\ =\ E_{K_{c,tgs}}[\ ID_c\ ,\ AD_c\ ,\ TS_3\ ] \qquad (2.3)$$

여기서 ID_v 는 요구하는 서버의 ID 이며, 고객의 인증자 $Authenticator_c$ 에는 고객의 ID 인 ID_C, 고객의 망 주소인 AD_C, 그리고 인증자가 발급된 시간인 TS_3 를 포함한다. TGT 는 $K_{c,tgs}$를 분배하기 위한 정보이다. TGT 는 고객이 AS에 의해 인증 받았음을 TGS 로 알리기 위한 정보이다. 고객의 인증자, $Authenticator_c$ 는 고객이 생성하며, TGT 의 유효성을 검증하기 위한 정보이며, 고객과 TGS 간의 세션키 $K_{c,tgs}$ 로 ID_c, AD_c, TS_3 를 암호화한 암호문이다. TS_2 는 인증자가 생성된 시간을 TGS로 알리기 위한 시간 정보이다. 인증자는 재생 공격을 방지하기 위하여 매우 짧은 수명을 갖는다.

이를 수신한 TGS는 AS와 TGS 간의 비밀키인 K_{tgs} 로 TGT 를 복호하여 $K_{c,tgs}$, ID_c, AD_c, ID_{tgs}, TS_2, 그리고 $Lifetime_2$ 를 복구한다. TGS 는 $K_{c,tgs}$ 를 알고있는 객체는 유효한 고객 만이라는 것을 확인 할 수 있다. TGS 는 $K_{c,tgs}$ 를 사용하여 인증자를 복호한다. TGS 는 TGT 에서 복구된 ID_C, AD_C 와 인증자로 부터 복구된 ID_c, AD_c 를 검사하여 일치하면, 고객의 정당성을 확인한다. TR 을 검증한 TRS 는 이에 응하여 티켓 응답 정보인 TRPS(Ticket Response) 을 식 (2.4)와 같이 생성하여 고객으로 전달한다.

$$E_{K_{c,tgs}}[K_{c,v}\ ,\ ID_v\ ,\ TS_4\ ,\ E_{K_v}\ [K_{c,v}\ ,\ ID_c\ ,\ AD_c,\ ID_v\ ,\ TS_4\ ,\ Lifetime_4\ ]] \qquad (2.4)$$

$K_{c,tgs}$ 는 고객과 TGS 간의 세션키로서, AS에 의해 생성된다. ID_v 는 서버 V의 ID 이다. TS_4 는 식 (2.4) 의 후반부인 SGT 가 발행된 시점을 나타낸다. Lifetime 4 는 TGT 의 수명을 의미한다. SGT 는 $K_{c,v}$, ID_c, AD_v, IDv, IS_4, $Lifetime_4$ 을 TGS 와 서버가 사전에 공유하고 있는 비밀키인 K_v 로 암호화한 암호문이다. 이를 수신한 고객은 고객과 서버간의 비밀키인 $K_{c,v}$ 와 SGT 를 복구한다. 이를 이용하여 고객은 서비스 요구(SR:Service Request) 정보인 메시지 5을 식 (2.5)와 같이 생성하여 서버로 전달한다.

$$SGT\ =\ E_{K_v}\ [K_{c,v}\ ,\ ID_c\ ,\ AD_c,\ ID_v\ ,\ TS_4\ ,\ Lifetime_4\ ],$$
$$Authenticator_v\ =\ E_{K_{c,v}}\ [ID_C\ ,\ AD_C\ ,\ TS_5\ ] \qquad (2.5)$$

여기서, SGT 는 식 (2.4)에서 복구된 정보이고, $Authenticator_v$ 는 $K_{c,v}$ 로 ID_C, AD_C , TS_6 를 암호화한 결과이다. TGT는 고객이 AS에 의해 인증 받았음을 확인하기 위한 정보이고, 재사용이 가능하다. K_v 는 단지 TGS와 서버로 알고 있는 비밀키이다. $K_{c,v}$ 는 고객과 서버간의 세션키이다. ID_c 는 TGT의 적법한 소유자의

ID 이고, AD_c 는 고객의 주소로서, 다른 주소를 갖는 WS 으로부터 TGT 의 사용을 방지하기 위한 주소이다. ID_v 는 서버의 ID 이다. TS_4 는 TGT 가 발행된 시간을 의미한다. $Lifetime_4$ 는 TGT 의 수명이다. 인증자는 현재 TGT 를 제시하고 있는 고객이 TGT 를 발행 받은 고객임을 입증하기 위한 정보이다. TS_5 는 인증자가 생성된 시간이다.

이를 수신한 서버는 자신의 비밀키를 이용하여 $K_{c,v}$, ID_c, AD_c, ID_v, TS_4, $Lifetime_4$ 를 복구한다. $K_{c,v}$ 는 고객과 서버간의 세션키이며, TS_5 는 인증자가 생성된 시간을 의미한다. 그리고 인증자로 부터 ID_c, AD_c, TS_5 를 복구한다. 서버는 SGT 에서 복구된 ID_c 와 AD_c가 인증자로 부터 복구된 ID_c 와 AD_c 와 동일한 가를 확인함으로서, 고객의 정체를 확인하고 서비스를 제공한다. 이를 확인한 서버는 고객이 서버의 신분을 확인하기 위하여 식 (2.6) 과 같은 서비스 응답 정보인 식 (2.6) 가 같은 메시지 6을 고객으로 전달한다.

$$E_{K_{c,v}}[TS_5 + 1] \tag{2.6}$$

서버는 선택적으로 TS_5 보다 "1" 증가된 값을 $K_{c,v}$ 로 암호화한 결과를 고객으로 전달한다. 이를 수신한 고객은 $K_{c,v}$ 로 $E_{K_{c,v}}[TS_5 + 1]$ 를 복호하여 $TS_5 + 1$ 를 확인하여 서버의 신분을 확인한다. 여기서 세션키 $K_{c,v}$ 는 유효한 고객과 서버만이 알고 있다.

2.3 다중 영역간 상호 인증

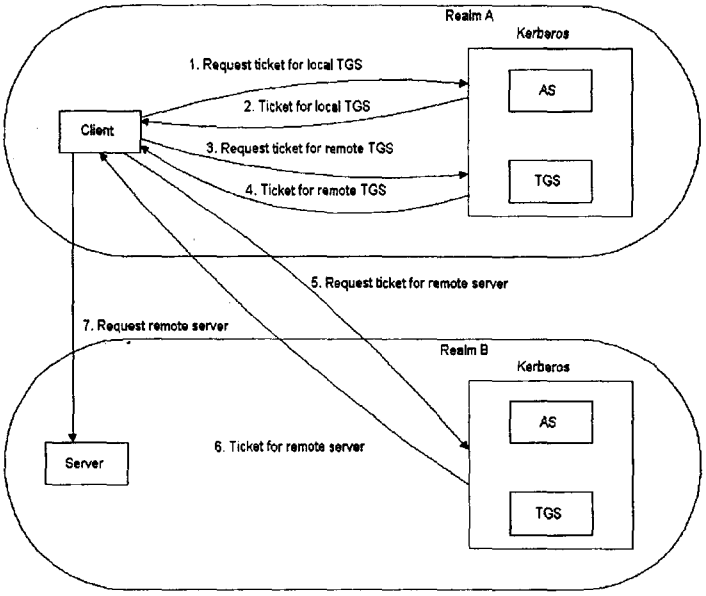

그림 2.3 영역간 인증을 위한 인증 구조

영역(Realm) 은 특정한 인증 서버로 등록된 고객과 서버의 집합으로서, 동일한 관리 기관하에 있는 모든 고객과 서버로 구성된 망이다. 완전한 Kerberos 시스템은 여러 개의 영역으로 이루어진 AS들, 고객들, 그리고 많은 응용 서버들로 구성된다. 단일 영역 Kerberos 인증 시스템에서의 기본 요구 사항은 다음 두 가지만이 요구되나 다중 영역 Kerberos 에서는 하나가 더 추가된다. ① AS는 자신의 관할 하에 있는 모든 사용자의 해쉬된 패스워드와 사용자 ID를 유지해야 한다. ② AS는 각 서버와 비밀키를 사전에 미리 공유하고 이를 비밀스럽게 간직해야 한다. 만약 다중 영역인 경우, 여기에 AS 가 다른 영역의 AS 와 사전에 추가의 비밀키를 공유해야 한다는 것이다. 영역이 다른 서버와 고객간의 인증 절차는 그림 2.3과 같은 기본 인증 구조와 그림 2.4와 같은 인증 프로토콜로 실현된다.

그림 2.3과 같은 영역간 인증을 위한 기본 구조는 기본적으로 원격 TGS에서 발행하는 TGT를 획득하는 절차로 구성된다. 다른 영역에 있는 서버는 다른 영역의 AS 를 신뢰해야 하는 가정 하에서 동작된다.

사용자가 다른 영역 서버의 서비스를 받기 원하면, 해당 서버를 위한 SGT($Ticket_{v\_rem}$) 가 요구된다. 이는 기본적으로 원격의 TGS 에 의해 발행되어야 한다. 고객은 같은 영역내의 서버를 이용하기 위해서는 기본적으로 2.2절의 절차를 따르지만 다른 영역의 서버를 이용하기 위해서는 다른 영역 TGS에서 발행된 SGT 이 요구된다. 따라서 다른 영역에 있는 서버를 이용하기 위한 고객은 자신의 TGS 에서 얻은 TGT 를 원격의 TGS 로 제출하여 원격 TGS 에서 발행한 SGT 를 얻는 절차가 추가되어야 한다. 이는 기본적으로 그림 2.3 에서 알 수 있듯이 자국의 AS 와 고객간에 수행되는 인증 요구 및 응답 과정, 고객과 자국의 TRS 간에 수행되는 원격국용 TGT($Ticket_{tgs\_rem}$)를 구하기 위한 원격국 TGT 요구 및 응답 과정, 자국의 TGS 에서 발행된 원격 TGT 를 이용하여 원격 TGT 와 고객간에 수행되는 원격 SGT($Ticket_{v\_rem}$) 요구 및 응답 과정, 그리고 마지막으로 원격국 서버와 고객간에 수행되는 원격국 서비스 요구 및 응답 과정으로 구성된다.

그림 2.4와 같은 영역간 인증을 위한 구체적인 프로토콜에서 이용되는 구체적인 인증 정보는 식 (2.7) 과 같다.

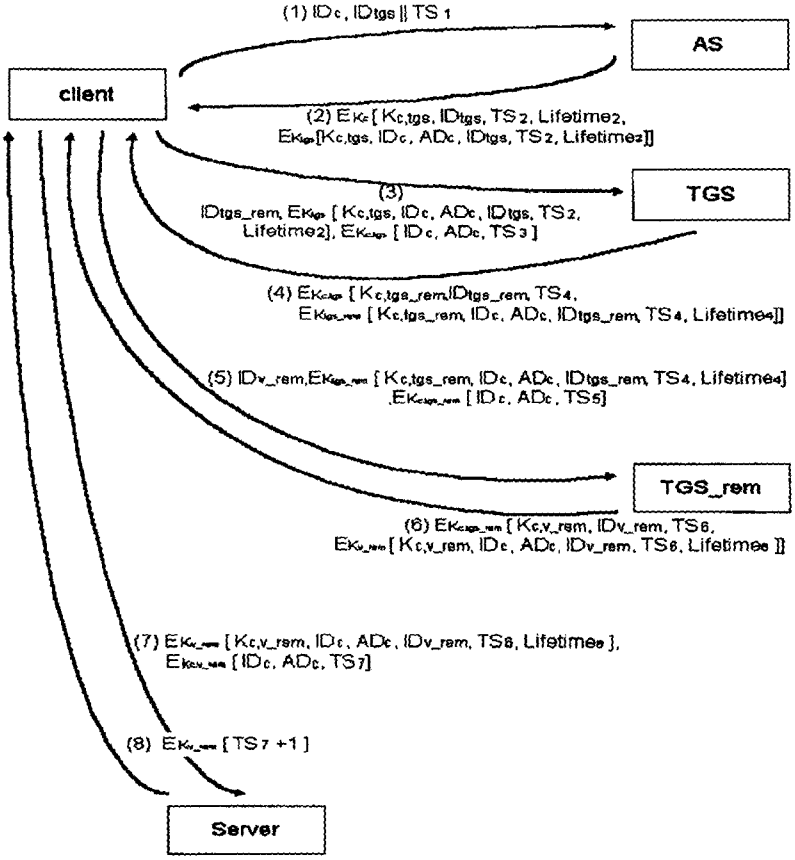

그림 2.4 영역간 인증을 위한 인증 프로토콜

. C -> AS : ID_C , ID_{tgs} , TS_1

. AS -> C : $E_{Kc}[K_{c,tgs}$, ID_{tgs} , TS_2 , $Lifetime_2$, $Ticket_{tgs}]$

여기서, $Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, Lifetime_2]$

. C -> TGS : ID_{tgs_rem} , $Ticket_{tgs}$, $Authenticator_c$

여기서, $Authenticator_c = E_{K_{c,tgs}}[ID_c, AD_c, TS_3]$

. TGS -> C : $E_{Kc,tgs}[K_{c,tgs\_rem}$, ID_{tgs_rem} , TS_4 , $Ticket_{tgs\_rem}]$

여기서, $Ticket_{tgs\_rem} = E_{K_{tgs\_rem}}[K_{c,tgs\_rem}, ID_c, AD_c, ID_{tgs\_rem}, TS_4, Lifetime_4]$

. C -> TGS_{rem} : ID_{v_rem} , $Ticket_{tgs\_rem}$, $Authenticator_c$

여기서, $Authenticator_c = E_{K_{c,tgs\_rem}}[ID_c, AD_c, TS_5]$

. TGS_{rem} -> C : $E_{Kc,tgs\_rem}[K_{c,v\_rem} , ID_{v\_rem} , TS_6 , Ticket_{v\_rem}]$

여기서, $Ticket_{v\_rem} = E_{K_{v\_rem}}[K_{c,v\_rem} , ID_c , AD_c , ID_{v\_rem} , TS_6 , Lifetime_6]$,

. C -> V_{rem} : $Ticket_{v\_rem}$, $Authenticator_v$

여기서, $Authenticator_v = E_{K_{c,v\_rem}}[IDc , ADc , TS_7]$

. V_{rem} -> C : $E_{K_{c,v\_rem}}[TS_7 + 1]$ (2.7)

이와 같은 인증 방법은 N 개의 영역이 있을 경우, 사전에 미리 교환되어야 할 AS 간의 비밀키가 [N(N-1)]/2 가 되어 키 분배 및 관리가 어렵다는 문제점이 있다.

2.4 공개키 알고리듬에 바탕을 둔 분산망에서의 인증 구조 및 프로토콜

본 절에서는 공개키 알고리듬을 이용한 인증 구조를 제안하고, 이를 위한 구체적인 인증 프로토콜을 제시하며, 제안된 방식의 특징을 기술한다. 본 방식은 기본적으로 Kerberos 방식에 바탕을 두고 구성된다. 공개키 알고리듬을 이용한 인증 구조는 기본적으로 다음과 같은 원칙이 적용된다. 첫째, 고객과 서버는 공개키 암호 시스템의 계산적 부하를 줄이기 위하여 기존의 Kerberos 의 인증 방식에서 이용한 대칭형 암호 알고리듬만(DES 또는 IDEA) 을 적용한다. 둘째, TGT 와 SGT 얻는 과정에서만 공개키 기밀성 알고리듬, 공개키 서명 알고리듬, 그리고 일방향 해쉬 함수를 이용한다. 셋째, 영역간 인증을 위하여 각 AS 간의 믿음의 연결점은 공개키 증명서를 이용하여 해결한다. 넷째, 암호학적 안전성에 문제점이 대두되고 있는 DES 암호 방식 대신 IDEA 암호 알고리듬을 적용한다.

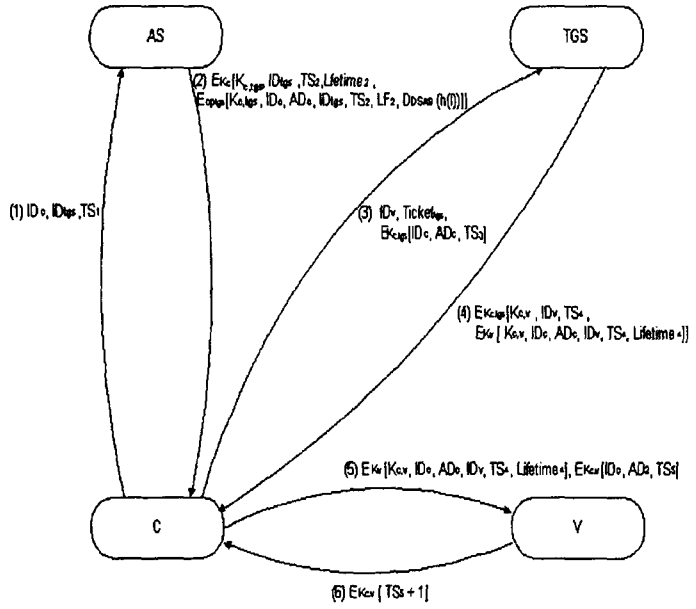

그림 2.5 영역내 인증을 위한 인증 프로토콜

이를 위하여 각 개체가 준비해야 할 변수는 다음과 같다. AS 는 공개키 방식의 서명용 공개키와 비밀키 (DP_{AS}, DS_{AS}) 를 준비한다. 이는 적법한 AS 가 TGT 를 발행했음을 TGS 에 알리기 위해 이용된다. TGS 는 공개키 서명용 공개키 및 비밀키(DP_{tgs}, DS_{tgs})와 기밀성용 공개키 및 비밀키(CP_{tgs}, CS_{tgs})를 준비해야 한다. 기밀성용 공개키 알고리듬은 AS 에서 TGS 로 TGT 를 보낼 때 이용되는 기밀성 알고리듬이다. 서명용

공개키 알고리듬은 자국의 TGS 가 원격국의 TGS 로 원격 TGT 를 전송할 때 이용되는 알고리듬이다. 채용되어야 할 공개키 서명 및 기밀성 알고리듬은 임의의 알고리듬이 선택될 수 있다.

TGS 를 위한 서명용 및 기밀성용 공개키는 믿을 수 있는 제3자인 CA(Certificate Authority) 에 의해 발행되는 공개키 증명서를 영역내의 모든 AS 로 전달하거나 별도의 CEP(Certificate Exchange Protocol) 로 전달되어야 한다. AS 와 고객간에 공유되는 비밀키는 사용자의 패스워드를 일방향 해쉬함수에 적용하여 구한 결과이다. AS 는 사용자의 ID 에 대응되는 해쉬된 비밀키를 데이타 베이스에 비밀스럽게 보관해야 한다. TGS 와 서버간에 공유되어야 할 비밀키는 사전에 비밀스럽게 분배되어야 한다. TGS 을 위한 공개키 증명서는 ITU X.509 구조를 따라 구성되며, 공개키 증명서 내의 공개키는 서명용 공개키와 기밀성용 공개키 등 2가지가 있다. 대표적인 기밀성 알고리듬은 RSA 이고, 대표적인 서명 알고리듬은 RSA, DSS 등이다. 공개키 증명서를 생성하기 위한 CA 의 서명 알고리듬과 해쉬 함수도 준비해야 한다.

영역내 인증을 위한 인증 구조는 그림 2.1과 같이 AS, TGS, C, 그리고 서버로 구성되며, 영역간 인증 구조는 그림 2.3과 같이 AS, 자국 TGS, 원격국 TGS, 서버, 그리고 고객 등으로 구성된다. 영역내 인증을 위한 인증 프로토콜은 그림 2.5와 같은 프로토콜을 이용한다.

그림 2.5와 같은 인증 프로토콜에서 인증 시스템의 구성 요소간의 교환되어야 할 인증 메시지는 여섯 가지 인증 정보이다. 고객이 AS로 전달하는 인증 요구(AR:Authentication Request) 정보로서, TGS에 TGT를 요구하기 위한 메시지이다. 이는 인증 요구 정보는 식 (2.8) 과 같다.

$$ID_c \ , \ ID_{tgs} \ , \ TS_1 \qquad (2.8)$$

이에 응하여 AS 는 인증 응답(AR:Authentication Response) 정보를 식 (2.9)'와 같이 고객으로 전송한다.

$$E_{K_c}[K_{c,tgs}, ID_{tgs}, TS_2, Lifetime_2, E_{CP_{as}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, Lifetime_2, D_{DS_{AS}}(h(I))]] \qquad (2.9)$$

$K_{c,tgs}$ 는 사용자의 패스워드에서 얻을 수 있는 비밀키이다. $K_{c,tgs}$ 는 K_c 로 암호화된 메시지 내부에 있기 때문에 적법한 고객만이 $K_{c,tgs}$ 를 알 수 있다. TGT 는 식 (2.9) 의 후반부로서, $I=(K_{c,tgs} \ , \ ID_c \ , \ AD_c \ , \ ID_{tgs} \ , \ TS_2 \ , \ Lifetime_2)$를 해쉬하여 AS 의 서명키로 서명한 서명문과 I 를 TGS 의 기밀성용 공개키로 암호화한 암호문이다. AS 의 서명문은 해당 데이타가 AS 에 의해 생성됐음을 나타내기 위한 정보이고, 이를 TGS 의 기밀성용 공개키로 암호화함으로서 TGS 만이 이를 복호할 수 있게 하였다.

AR 을 수신한 고객은 AR 로부터 고객과 TGS 간의 세션키 $K_{c,tgs}$ 와 TGT 를 복구한다. 고객은 이를 이용하여 식 (2.10)과 같은 티켓 요구 (TR:Ticket Request) 정보를 TGS 로 전송한다.

$$ID_v \ , \ Ticket_{tgs} = E_{CP_{as}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, Lifetime_2, D_{DS_{AS}}(h(I))],$$
$$Authenticator_c = E_{K_{c,tgs}}[\ ID_c, \ AD_c, \ TS_3\ ] \qquad (2.10)$$

ID_v는 요구하는 서버의 ID이며, 고객의 인증자 $Authenticator_c$ 에는 고객의 ID 인 ID_C, 고객의 망 주소인 AD_C, 그리고 인증자가 발급된 시간인 TS_3 를 포함한다. TGT는 $K_{c,tgs}$를 분배하기 위한 정보이다. TGT 는 고객이 AS에 의해 인증 받았음을 TGS 로 알리기 위한 정보이다. 고객의 인증자, $Authenticator_c$ 는 고객이 생성하며, TGT 의 유효성을 검증하기 위한 정보이며, 고객과 TGS 간의 세션키 $K_{c,tgs}$ 로 ID_c, AD_c, TS_3 를 암호한 암호문이다. TS_2 는 인증자가 생성된 시간이다.

이를 수신한 TGS는 자신의 기밀성용 비밀키로 TGT 를 복호하여 $K_{c,tgs}$, ID_c , AD_c , ID_{tgs}, TS_2, $Lifetime_2$, 그리고 서명문를 복구한 후, $K_{c,tgs}$, ID_c , AD_c , ID_{tgs}, TS_2, $Lifetime_2$ 를 해쉬 함수에 적용한 결과와, 서명문을 AS 의 서명용 공개키로 복호한 결과가 동일한가를 확인한다. 동일하면 해당 정보가 AS 에 의해 생성되었음을 확인한다. TGS 는 $K_{c,tgs}$ 를 아는 객체는 유효한 고객 만이라는 것을 확인 할 수 있으므로 TGS 는 $K_{c,tgs}$를 사용하여 인증자를 복호한다. TGS 는 TGT 에서 복구된 ID_C, AD_C 와 인증자로 부터 복구된 ID_c , AD_C 를 검사하여 일치하면, 고객의 정당성을 확인한다. TR 을 검증한 TRS 는 이에 응하여 티켓 응답 정보인 TRS(Ticket Response) 을 식 (2.11) 과 같이 생성하여 고객으로 전달한다.

$$E_{K_{c,tgs}}[K_{c,v} \ , \ ID_v \ , \ TS_4 \ , \ E_{K_v}\ [K_{c,v} \ , \ ID_c \ , \ AD_C \ , \ ID_v \ , \ TS_4 \ , \ Lifetime_4 \ ]] \qquad (2.11)$$

$K_{c,tgs}$ 는 고객과 TGS 간의 세션키로서, AS에 의해 생성된다. ID_v 는 서버 V의 ID 이다. TS_4 는 식 (2.11)의 후반부인 SGT 가 발행된 시점을 나타낸다. Lifetime 4 는 TGT 의 수명을 의미한다. SGT 는 $K_{c,v}$, ID_c, AD_C, ID_v, TS_4, $Lifetime_4$ 을 TGS 와 서버가 사전에 공유하고 있는 비밀키인 K_v 로 암호화한 암호문이다. 이를 수신한 고객은 고객과 서버간의 비밀키인 $K_{c,v}$ 와 SGT 를 복구한다. 이를 이용하여 고객은 서비스 요구(SR:Service Request) 정보를 식 (2.12)와 같이 생성하여 서버로 전달한다.

$$SGT = E_{K_v}[K_{c,v}, ID_c, AD_C, ID_v, TS_4, Lifetime_4],$$
$$Authenticator_v = E_{K_{c,v}}[ID_C, AD_C, TS_5] \quad (2.12)$$

여기서, SGT 는 식 (2.11)에서 복구되고, $Authenticator_v$ 는 $K_{c,v}$ 로 ID_C , AD_C , TS_5 를 암호화한 결과이다. TGT는 고객이 AS에 의해 인증 받았음을 확인하기 위한 정보이고, 재사용이 가능하다. K_v 는 단지 TGS와 서버로 알고 있는 비밀키이다. $K_{c,v}$ 는 고객과 서버간의 세션키이다. ID_c 는 TGT의 적법한 소유자의 ID 이고, AD_c 는 고객의 주소로서, 다른 주소를 갖는 WS 으로부터 TGT 의 사용을 방지하기 위한 주소이다. ID_v 는 서버의 ID 이다. TS_4 는 TGT 가 발행된 시간을 의미한다. $Lifetime_4$ 는 TGT 의 수명이다. 인증자는 현재 TGT 를 제시하고 있는 고객이 TGT 를 발행받은 고객임을 입증하기 위한 정보이다. TS_5 는 인증자가 생성된 시간이다.

이를 수신한 서버는 자신의 비밀키를 이용하여 $K_{c,v}$, ID_c, AD_c, ID_v, TS_4, $Lifetime_4$ 를 복구한다. 그리고 인증자로 부터 ID_c, AD_c, TS_5 를 복구한다. 서버는 SGT 에서 복구된 ID_c 및 AD_c와 인증자로 부터 복구된 ID_c 및 AD_c 가 동일한 가를 확인함으로서, 고객의 정체를 확인하고 서비스를 제공한다. 이를 확인한 서버는 고객이 서버의 신분을 확인하기 위하여 식 (2.13) 과 같은 서비스 응답 정보인 식 (2.6)과 같은 메시지 6을 고객으로 전달한다.

$$E_{K_{c,v}}[TS_5 + 1] \quad (2.13)$$

서버는 선택적으로 TS_5 보다 "1" 증가된 값을 $K_{c,v}$ 로 암호화한 결과를 고객으로 전달한다. 이를 수신한 고객은 $K_{c,v}$ 로 $E_{K_{c,v}}[TS_5 + 1]$ 를 복호하여 $TS_5 + 1$ 를 확인하여 서버의 신분을 확인한다. 여기서 세션키 $K_{c,v}$ 는 유효한 고객과 서버만이 알고 있다.

사용자가 다른 영역의 서버의 서비스를 받기 원하면, 해당 서버를 위한 SGT 가 필요하다. 이는 기본적으로 원격의 TGS 에 의해 발행되어야 한다. 원격 인증을 위한 기본 구조는 그림 2.3과 같은 인증 구조에 바탕을 두고 있다. 이는 기본적으로 그림 2.3 에서 알 수 있듯이 자국의 AS 와 고객간에 수행되는 인증 요구 및 응답 과정, 고객과 자국의 TRS 간에 수행되는 원격국용 TGT 를 구하기 위한 원격국 TGT 요구 및 응답 과정, 자국의 TGS 에서 발행된 원격 TGT 를 이용하여 원격 TGT 와 고객간에 수행되는 원격 SGT 요구 및 응답 과정, 그리고 마지막으로 원격국 서버와 고객간에 수행되는 원격국 서비스 요구 및 응답 과정으로 구성된다. 그림 2.6과 같은 영역간 인증을 위한 구체적인 프로토콜에서 이용되는 구체적인 인증 정보는 식 (2.14)와 같다.

. C -> AS : ID_C , ID_{tgs} , TS_1

. AS -> C : $E_{Kc}[K_{c,tgs}$, ID_{tgs} , TS_2 , $Lifetime_2$, $Ticket_{tgs}]$,

여기서, $Ticket_{tgs} = E_{CP_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, TS_2, Lifetime_2, D_{DS_{AS}}(h(I))]$

$I = K_{c,tgs}$, ID_c , ADc , IDtgs , TS_2 , $Lifetime_2$

. C -> TGS : ID_{tgs_rem} , $Ticket_{tgs}$, $Authenticator_c$

여기서, $Authenticator_c = E_{K_{c,tgs}}[ID_c , AD_c , TS_3]$

. TGS -> C : $E_{Kc,tgs}[K_{c,tgsrem}$, ID_{tgsrem} , TS_4 , $Ticket_{tgs\_rem}]$

여기서, $Ticket_{tgs\_rem} = E_{CP_{tgs\_rem}}[K_{c,tgs\_rem}, ID_c, AD_c, ID_{tgs\_rem}, TS_4, Lifetime_4, D_{DS_{AS}}(h(I))]$

. C -> TGS_{rem}: ID_{vrem} , $Ticket_{tgs\_rem}$, $Authenticator_v$

여기서, $Authenticator_v = E_{K_{c,tgs\_rem}}[ID_c , AD_c , TS_5]$

. TGS_{rem} -> C: $E_{Kc,tgs\_rem}[K_{c,v\_rem}, ID_{vrem}, TS_6, Ticket_{v\_rem}]$

여기서,$Ticket_{v\_rem} = E_{K_{v\_rem}}[K_{c,v\_rem}, ID_C, AD_C, ID_{v\_rem}, TS_6, Lifetime_6]$

. C -> V_{rem} : $Ticket_{v\_rem}$, $Authenticator_c$

여기서, $Authenticator_v = E_{K_{c,v\_rem}}[IDc, ADc, TS_7]$

. V_{rem} -> C : $E_{K_{c,v\_rem}}[TS_7 + 1]$

(2.14)

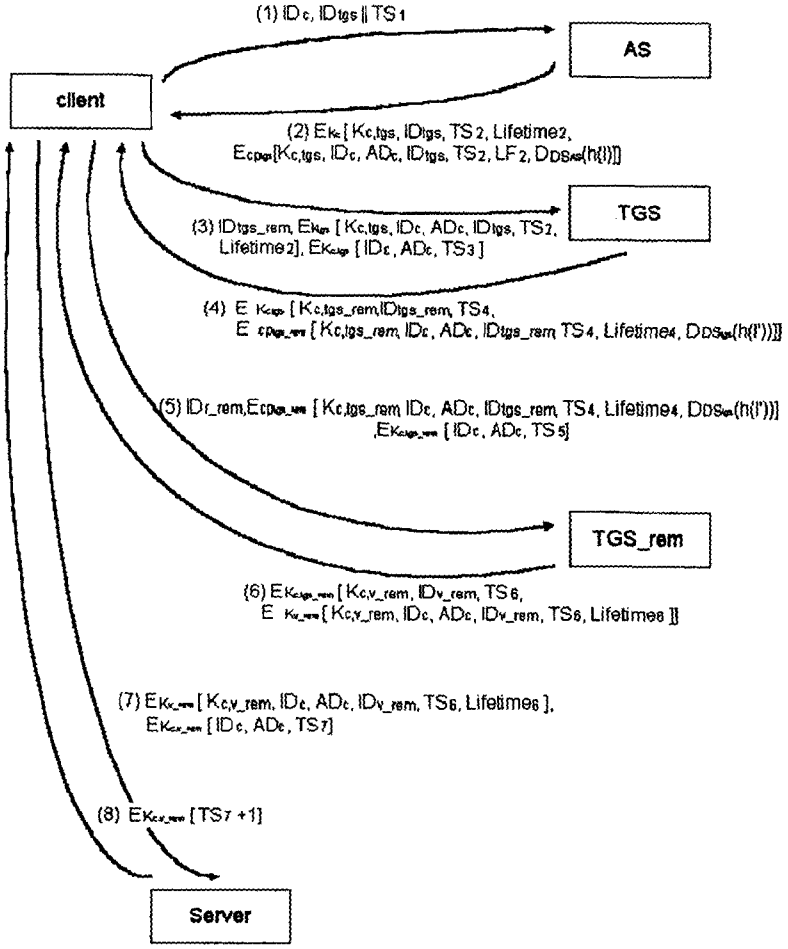

그림 2.6 영역간 인증을 위한 인증 프로토콜

제안된 인증 방식의 특징은 다음과 같다. 첫째, 영역간 상호 인증을 위하여 각 AS는 비밀키를 미리 상호 공유하지 않고 공개키 증명서만을 교환한다. 따라서 안전하게 비밀키를 저장하기 위한 복잡한 키 분배 및 유지 시스템이 요구되지 않는다. 둘째, DES 대신 IDEA 를 채용함으로서, DES 의 암호학적 안전성에 대한 불신을 제거하였다. 셋째, AS 와 TGS 만이 공개키 서명 및 기밀성 알고리듬을 이용하고, 고객과 서버는 기존의 방식과 마찬가지로 오직 대칭형 암호 알고리듬 만을 사용함으로서, AS와 TGS 에만 공개키 암호 알고리듬의 도입으로 인한 속도 및 복잡도의 증가를 부가하고 고객과 서버에는 그대로 비밀키 암호 방식만을 사용한다. 넷째, 공개키 증명서를 도입하였으므로 믿음의 확장이 용이하다. 다섯째, 공개키 증명서를 발행할 CA 가 분산망 내에 반드시 추가로 요구된다.

제3장 분산망에서의 인증 방식 시뮬레이션

3.1 IDEA 를 이용한 분산망에서의 인증 방식 시뮬레이션

본 절에서는 먼저 IDEA 을 이용한 분산망에서의 인증 방식을 시뮬레이션하였다. 시뮬레이션은 기본적인 파라메타에 대해서만 수행하였다. AS 와 고객간에 교환되는 패스워드 및 패스워드를 MD5 해쉬 함수로 통과한 128 비트의 IDEA 비밀키는 식 (3.1)과 같다.

BJH100
1568 4920 178c 65ec ad71 a957 b096 43d5 (3.1)

고객과 서버간에 공유되어야 할 128 비트의 IDEA 비밀키는 식 (3.2)와 같다.

47fa d16e 0b79 4641 25be 8612 bc82 0382 (3.2)

AR 의 정보는 식 (3.3)과 같다.

ID_c, ID_{tgs} = jonghyun@asan.sch.ac.kr, tgs@asan.sch.ac.kr (3.3)

ARS 는 식 (3.4)과 같다.

$E_{K_c}[k_{c,tgs}, ID_{tgs}, E_{K_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}]]$

= 004f 683f 2a6d 8f5b 9dcf 2bcf e961 d934 490e 4d9f 49ed e87d 1ec4 cbd7 b36d 3e70 9077 1bdc b50e 8b04 5ca3 d2bc 1955 b182 26de 72ba a021 2aa9 a5a7 cf51 dab3 5442 28da c531 7abd 9882 df3f 02ea f6bc be81 7f26 92f2 4ce1 0fbb 6ced 9b95 3a50 d642 6635 1044 a950 7173 4868 7bd8 a646 5ec5 8cd0 ffb8 7644 4194 cdc8 3bfb 8424 ac66 0620 ade6 ba00 61bd 2208 7b19 ecac 326c f3c7 0716 e1c5 26a2 44d4 92eb 7216 6017 0d1e 534c d2e7 8c67 f74c f24f c6dd 8251 475c 1377 3693 0b48 5897 7d75 2d5d 5262 02d4 85ae d88e 69ca 8020 c8bb bc49 ceac e963 1c42 a35c 1354 443b 622a 4bab 429f 10ff 206a fdc3 f0f0 5c13 e129 b480 da82 392f 03a2 4ae2 f110 2561 0452 d358 5278 fadd d020 18fe 2845 1bee 99e9 5dc0 39f0 (3.4)

TR 은 식 (3.5) 과 같다.

ID_v , $Ticket_{tgs}$ = $E_{K_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}]$

= server@asan.sch.ac.kr, 00a2 5453 17ac 635e ff90 1d4e 2c0a b347 8b2e 495b c662 42ec c71a ad1d 146c 8b90 ebf4 9e28 a2e8 e03c b7c3 b818 9fec 2805 bffd 5770 203b bb61 beb2 26d4 2a55 1593 8c0b 3114 8574 0d3e bd8d adfd 1327 7814 8de2 bc58 d82e 204b ac20 9a4d 8b97 17f8 6c04 75d7 e97b 1d7e

$Authenticator_c$ = $E_{K_{c,tgs}}[ID_c , AD_c]$

= 00b6 fd42 4acd 7fff 3a16 6c91 fb5d 139d 06db 4c61 f28a faf5 5330 3a43 4bf7 96b1 aef7 84d8 1ef1 79c2 5fc6 360b 7b01 03d5 0e69 a537 609d 3c35 (3.5)

TRS는 식 (3.6) 과 같다.

$E_{K_{c,tgs}}[K_{c,v} , ID_v , E_{K_c}[K_{c,v} , ID_c , AD_v , ID_v]]$

= 00eb cc4f 7de7 b45a 6819 e280 05fe 0e87 5eb3 fa56 231f e5bb afaf a800 b647 22df 1294 8ca0 b82d 4b2a 1677 f189 6cdd 85fd f18d 2dd3 6ac2 6f0e 6952 3e5b f3c8 465f c0ee 9c65 617d deb6 7552 4ebf 745a ed45 c608 5d98 c017 c845 a5c3 93c8 19c0 6424 09cb 44c3 b136 f644 2236 f6b1

9de0 23d1 aead 21d0 42fb 2946 d809 f828 0f4b b6fb 28f1 c7ca 1242 3bee 9e0e 4245 4cba 729d
3573 259c 39de 4312 c1c2 89e8 7fee 3fc0 2abc 05c7 a94d 6841 c50e 236c 4bf8 654e 1522 f2a7
d98b 4b7a 53fc 1da3 4fa4 5a61 1423 fc08 fb24 84cd 875b 3ec0 22c9 e10b 0c4f 071f 98ac 6722
2e07 20a1 e0e4 c16f ffba 39fb b88c e664 316e 9adc e9f1 d57f 6b1b 9bc3 0ac8 6d66 1ace 915e
ca7d 85ec dad1 8717 473f 90a0 d4ae eace ba47 d2b4 (3.6)

SR 은 식 (3.7) 과 같다.

SGT = E_{K_v}[$K_{c,v}$, ID_c, AD_v, ID_v]

= 0029 e20f 8889 761a 0302 717e 7358 eb52 4ae1 46ee 2587 4482 dd17 867e 28f9 af1d 3b82
9514 2117 594b 8eb3 0286 3e8e 94aa fcb4 2a2d b1bb 88b5 d659 9fc5 35c0 2f9b 68a4 7152
74c9 aa9a b99e b460 9a01 160b b894 c7d4 0a81 233e 0380 8463 45bc a5a4 526a ad55
9270 fe5b

Authenticator$_v$ = $E_{K_{c,v}}$[ID_c, AD_c]

= 001e 9235 3486 282d 3e89 b2e5 c0c8 a939 019a 3669 fb0c 30cf d566 c5c2 a079
8741 7206 5a10 de56 216a e86c f94b 81a0 021d 63dc 926c d605 7cff (3.7)

시뮬레이션 결과 고속 동작이 가능함을 확인하였다. .

3.2 공개키 알고리듬을 이용한 인증 방식 시뮬레이션

RSA 서명 및 기밀성 알고리듬을 이용하여 인증 방식을 시뮬레이션 하였다. 모듈러 지수 연산 알고리듬은 768 비트이며, 윈도우 크기가 6인 윈도우 방식을 이용했으며, 모듈러 곱셈 알고리듬은 고전적인 방식을 적용하여 C 언어로 구현하였다. 모듈러 지수 연산 알고리듬은 IBM PC 586 100MHz 에서 MS-C 를 이용하여 컴파일하여 0.7초 이내에 동작됨을 확인하였다. 또한 안전한 768 비트 크기의 RSA 키 생성기를 C 언어로 구현하였다. 인증에 이용된 C 루틴은 64 비트 IDEA 루틴, 128 비트 MD5 루틴, 고속의 지수 연산 루틴, 768 비트 지수 연산 루틴, 그리고 RSA 암호키 생성 루틴 등이다. 시뮬레이션은 영역내 인증 방식 만을 수행하였다. 인증을 위한 AS 의 서명용 공개키 및 비밀키는 식 (3.8) 과 같다.

n = f085 5d29 cb4e 4ab1 c3db f340 302b b9f2 aa04 2a5a 23a1 a2ee ce2e c308 7989 3137 e894 8876
6ae5 7054 202f 7de9 a58f 3a52 6f35 8690 2ae8 e06d a3d2 6184 f4b9 fdd9
e = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0001
d = 252e 4897 9461 16f4 e1f5 1766 5dcb f4a3 c99a 793d 93f2 81b0 6ed9 15b7 32aa 7d9b fd34 31e4
ae6b cd6f be3e 1d71 3cd9 28f8 489f 8521 2acd ddb3 ccb0 6691 fcb3 9901 (3.8)

TRS 의 기밀성용 공개키 및 비밀키는 식 (3.9) 과 같다.

n = d12a fb16 c43e a524 b190 dbc7 92ce dcb0 7e29 54a2 f0c0 fd5a 277c 7d7e 186a b955 92d4 6cae
3bb1 648f 421e 2478 c3c1 5cf9 a0f7 3a98 3191 f27a 347b 67aa 1bf6 ed39
e = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0001
d = 97a9 617c 0579 22c6 f355 0476 b594 f661 a668 936c bdc2 4131 4056 3cab d44a 75ec 2d2c 1a1a
ea2c 3a29 ea8e 1b17 3dcc 5525 415a f6cf f9de c87c 7eef a8bf 62c6 f05d (3.9)

인증 요구(AR:Authentication Request) 정보는 식 (3.10) 와 같다.

ID_c , ID_{tgs} = jonghyun@asan.sch.ac.kr, tgs@asan.sch.ac.kr (3.10)

인증 응답(AR:Authentication Response) 정보은 식 (3.11) 과 같다.

$E_{K_c}[K_{c,tgs}, ID_{tgs}, E_{CP_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, D_{DS_{AS}}(h(I))]]$

= 00dd 0e8a 9ff7 59b3 a0db 4691 3eef fd1c a6ff 206a fdc3 f0f0 5c13 e129 b480 da82 392f 03a2 4ae2 f110 2561 0452 d358 5278 fadd d020 18fe 2845 1bee 99e9 5dc0 39f0 35d6 0351 6d8b c21e bdf9 f4bf a24a cad2 7fb4 cd80 87bf b6c4 3fa4 54c6 fbea 003d 527d b4a8 0173 de3b b150 6e1c d11d d57f 995a 17cd 7323 ad55 56ed 12ed 724d 0a88 d2eb af47 a37c d519 c66e b1d7 b421 7825 855b 5bed 6edd 9668 5283 0565 2384 7767 aedd 0491 9707 60da 1603 1351 7620 84bb 39f3 d6f1 6995 3da4 6535 7505 0ffb c3be f08c 29c6 b95a 31af 8dd7 977b 59ee 64c2 4074 2be9 6bac fdf0 e8ec b5f8 bfc5 2831 e117 e82c 431e 9cde a78f 629b 8a73 eb59 6a06 5ef4 13a5 7fc6 d5b1 f2ce b440 7537 5b44 6a52 0dc9 663b f040 0cf6 78dd 4a45 0910 22cd 8e86 97d3 6542 1afd b39e 560c 216b e92d f086 89cf 0004 f93b 202e 0f0e 3573 7879 7f31 e5ec e183 2385 b6e3 8728 b869 2422 3475 fbae f7a7 85cc 3ea3 46b0 dee9 5ada 18da 2493 e427 cb39 4eb2 1230 0283 d8e9 a7c3 1c74 2604 bb6b 752f 9575 44f0 cf10 fc58 79b5 df3b ae75 72c1 304f 5fed 9324 f3d5 2be6 a30b 6079 a057 93b7 449a 69b3 63bb 9867 4bb0 c5e1 02f7 027a a7a7 3396 d58a f465 1158 ce1f 62a3 0b6d 034c fba7 463f 67b1 0296 9758 3c76 ed7f c838 5375 b03d 9e2b 2d5e 26e6 47d9 c131 7c76 fe67 b6d3 99da 0cc7 4e5a 90b5 050d 9352 9474 5007 6326 ae82 b6b0 76b5 ac26 2a32 5d81 68c6 7c9c 8ddc 9881 45b4 2603 cd67 7cf6 8c7b 11af ca95 f0c1 7258 bb26

(3.11)

티켓 요구 (TR:Ticket Request) 정보는 식 (3.12) 과 같다.

ID_v , $Ticket_{tgs}$ = $E_{CP_{tgs}}[K_{c,tgs}, ID_c, AD_c, ID_{tgs}, D_{DS_{AS}}(h(I))]$,

= server@asan.sch.ac.kr, 0004 144b 01fa 5c06 f496 19c0 0616 9cb2 ebb3 99d2 0635 123c a748 4d48 5c4a e3e2 de53 1a39 af0e d38b c499 ab4e a91d 66ce 8d29 9821 60d8 52d2 371f 9178 0919 3893 99d5 fe21 2c97 97a8 8477 ceb2 3313 4b7f 8175 d13e 3009 48d7 b5a4 e7c5 7f96 d16b 846e 3291 8ffa 5556 48ad 2cee 1d4e 71e7 ef33 894e f2f8 b0c8 015b 2c56 d817 20cf 932a 1f7e a10c 175c 7943 2a09 59ea b38a 3ed2 9c5c 5d4c a590 acdd c396 191d 7608 ea96 3245 167b 6678 ffe6 0e9f d261 e2b8 ec24 864b 38ca 798d 1878 e7da 9e62 cc5b 1f60 36e2 9df5 2483 9255 a3ef cc39 83d0 0a16 1cf0 d642 068a bb4a 1cbd 4e10 f4ab cda3 9806 fdd5 b2c8

$Authenticator_c$ = $E_{K_{c,tgs}}[ID_c , AD_c]$

= 00b6 fd42 4acd 7fff 3a16 6c91 fb5d 139d 06db 4c61 f28a faf5 5330 3a43 4bf7 96b1 aef7 84d8 1ef1 79c2 5fc6 360b 7b01 03d5 0e69 a537 609d 3c35

(3.12)

TRS(Ticket Response) 는 식 (3.13) 와 같다.

$E_{K_{c,tgs}}[K_{c,v} , ID_v , E_{k_v}[K_{c,v} , ID_c , AD_v , ID_v]]$

= 00eb cc4f 7de7 b45a 6819 e280 05fe 0e87 5eb3 fa56 231f e5bb afaf a800 b647 22df 1294 8ca0 b82d 4b2a 1677 f189 6cdd 85fd f18d 2dd3 6ac2 6f0e 6952 3e5b f3c8 465f c0ee 9c65 617d deb6 7552 4ebf 745a ed45 c608 5d98 c017 c845 a5c3 93c8 19c0 6424 09cb 44c3 b136 f644 2236 f6b1 9de0 23d1 aead 21d0 42fb 2946 d809 f828 0f4b b6fb 28f1 c7ca 1242 3bee 9e0e 4245 4cba 729d 3573 259c 39de 4312 c1c2 89e8 7fee 3fc0 2abc 05c7 a94d 6841 c50e 236c 4bf8 654e 1522 f2a7 d98b 4b7a 53fc 1da3 4fa4 5a61 1423 fc08 fb24 84cd 875b 3ec0 22c9 e10b 0c4f 071f 98ac 6722 2e07 20a1 e0e4 c16f ffba 39fb b88c e664 316e 9adc e9f1 d57f 6b1b 9bc3 0ac8 6d66 1ace 915e ca7d 85ec dad1 8717 473f 90a0 d4ae eace ba47 d2b4

(3.13)

SR:Service Request) 정보는 식 (3.14) 과 같다.

SGT = $E_{K_c}[K_{c,v}, ID_c , AD_c , ID_v]$

= 0029 e20f 8889 761a 0302 717e 7358 eb52 4ae1 46ee 2587 4482 dd17 867e 28f9 af1d 3b82 9514 2117 594b 8eb3 0286 3e8e 94aa fcb4 2a2d b1bb 88b5 d659 9fc5 35c0 2f9b 68a4 7152

74c9 aa9a b99e b460 9a01 160b b894 c7d4 0a81 233e 0380 8463 45bc a5a4 526a ad55 9270 fe5b

Authenticator$_v$ = $E_{K_{c,t}}$[IDc , ADc]

= 001e 9235 3486 282d 3e89 b2e5 c0c8 a939 019a 3669 fb0c 30cf d566 c5c2 a079 8741 7206 5a10 de56 216a e86c f94b 81a0 021d 63dc 926c d605 7cff

(3.14)

시뮬레이션 결과 제안된 공개키 알고리듬을 이용한 분산망에서의 인증 방식이 정상적으로 동작될 수 있음을 확인하였다.

제4장 결론

분산 컴퓨터망에서의 서버는 다수의 사용자를 인증한 후 자신의 서비스를 제공해야 한다. 본 논문에서는 기존의 대표적인 분산망에서의 인증 방식인 Kerberos 인증 시스템의 동작을 분석하고, 이를 바탕으로 기존의 분산망에서의 문제점을 제거할 수 있는 공개키 알고리듬을 이용한 인증 구조 및 프로토콜을 제안하였다. 제안된 인증 방식에서는 AS 의 서명용 공개키 및 비밀키, TGS 의 서명용 공개키 및 비밀키, TGS 의 기밀성용 공개키 및 비밀키, 그리고 TRS 의 서명용 및 기밀성용 공개키를 위한 공개키 증명서가 필요함을 확인하였다. 또한 공개키 알고리듬을 이용한 분산망에서의 인증 방식의 정당성을 확인하기 위하여 인증 시스템을 시뮬레이션하였다. 시뮬레이션된 공개키 서명 및 기밀성 알고리듬은 RSA 이며 해쉬 함수는 MD5 해쉬 함수이다. 모듈러 지수 연산 알고리듬은 768 비트 크기의 윈도우 크기가 6인 윈도우 방식을 이용했으며, 모듈러 곱셈 알고리듬은 고전적인 방식을 적용하여 C 언어로 구현하였다. 모듈러 지수 연산 알고리듬은 IBM PC 586 100MHz 에서 0.7초 이내에 동작됨을 확인하였다. 또한 안전한 768 비트 크기의 RSA 키 생성기를 C 언어로 구현하였다. 시뮬레이션에 이용된 C 루틴은 64 비트 IDEA 루틴, 128 비트 MD5 루틴, 고속의 지수 연산 루틴, 768 비트 지수 연산 루틴, 그리고 RSA 암호키 생성 루틴 등이다. 본 논문은 버젼 4 Kerberos 인증 방식에 적용될 수 있으나, 버젼 5에도 무리없이 적용할 수 있다. 추후의 연구는 버젼 5에 적용 가능한 공개키 알고리듬을 이용한 인증 방식에 대한 연구를 수행할 예정이다. 본 논문의 결과는 분산 컴퓨터망에서의 인증 시스템 설계시 유용하게 활용될 수 있을 것이다.

-참고문헌-

[1] W. Stallings, Network and Internetwork Security, IEEE Press, 1995.

[2] Man Young Rhee, Cryptography and Secure Communications, Mcgraw-Hill, 1992.

[3] ISO/IEC IS 9798-3, Entity Authentication Mechanisms-Part 3 : Entity Authentication Using a Public-key Algorithm, ISO, Geneva, Switzerland, 1993.

[4] B.C.Neuman, Theodore Ts's, "Kerberos : An Authentication Service for Computer Networks," IEEE Communications Magazine, Vol.32, No.9, pp.33-38, Sept. 1994.

[5] ITU Rec. X.509, The Directory-Authentication Framework, ITU, Geneva, Switzerland, 1993.

[6] NBS, Data Encryption Standard, FIPS Pub-46, 1977.

[7] J.T.Kohl, and B.C. Neuman, The Kerberos Network Authentication Service, Internet RFC 1510, September 1993.