위임 메커니즘을 이용한 연산 효율적인 PKI 기반 Single Sign-On 프로토콜 설계 및 분석

박기웅†, 임상석†, 송호성‡, 박규호†

한국과학기술원
컴퓨터공학 연구실†
{woongbak,sslim,kpark}@core.kaist.ac.kr

University of Wisconsin-Green Bay
Computer Science Dept.‡
songh@uwgb.edu

요 약

본 논문에서는 유비쿼터스 환경에서의 초소형 보안단말기를 위한 연산 효율적인 PKI 기반 Single Sign-On 프로토콜을 제안하고 분석한다. 본 논문에 따른 초소형 보안 단말기를 위한 프로토콜은 사용자에게 투명한 보안 메커니즘과 끊김없는 인증 서비스를 제공하며, 제안된 Single Sign-On 메커니즘은 사용자의 인증 연산을 위임 메커니즘과 사용자 이동 예측에 의한 선인증을 위임 서버가 수행하도록 함으로서 기존의 보안 수준과 기능을 유지하되 인증 지연시간과 유비쿼터스 환경에 존재하는 서비스 장치의 보안 인프라의 구축비용을 획기적으로 줄이고 병목현상을 해결한다. 또한 기존의 위임 메커니즘은 위임한 기간에 사용자의 인증 요청에 대한 부인 방지 기능을 제공하지 못한다. 본 논문에서 제안한 프로토콜은 심판서버를 도입하여 인증 메시지와 장치간의 바인딩 연산을 통해 부인 방지 기능을 제공하여 적은 연산량으로 전자 상거래 및 인증시에 PKI와 동일한 안전도를 보장한다.

1. 서론

본 논문에서는 유비쿼터스 컴퓨팅 환경에서 안전하고 원활한 보안 메커니즘을 제공하기 위한 프로토콜을 제시하고 분석한다. 유비쿼터스 컴퓨팅이 현실화 되면 사용자는 인비저블 형태의 편재된 컴퓨팅 서비스를 제공 받을 수 있을 것이다[6]. 이러한 서비스를 제공하기 위해 유비쿼터스 컴퓨팅에서는 기존의 컴퓨팅 환경에 비해 많은 변화가 생긴다[7]. 기존의 서버-클라이언트 형태의 인증구조에서 환경에 편재된 서비스 디바이스들과 사용자가 상호 인증을 요구하는 구조가 되고 서비스 디바이스의 수가 늘어남에 따라 사용자가 인증해야 하는 횟수도 늘어나게 되며 대칭키 기반 인증을 수행할 경우 특정 도메인에 종속된 보안 인프라가 되고 관리되어야 할 키의 개수가 기하급수적으로 증가하게 되어 유비쿼터스 환경을 위한 보안인프라로 적합하지 않다.

상호 인증 및 전자 상거래에 필수기능인 인증, 권한, 과금 기능을 제공하기 위해서 PKI를 적용시키는 것이 가장 적합한 솔루션이다[9]. PKI는 비대칭키 기반의 인증 방식으로서 디지털 서명, 부인방지 등의 기능을 제공하여 유비쿼터스 환경에 있어 인증 및 과금에 대한 메커니즘이 제공될 수 있으며 권한 인증을 위하여 PMI와의 연동이 용이하다는 장점이 있다[10].

하지만 PKI에서 사용되는 RSA 알고리즘은 대칭키에 비하여 연산 오버헤드가 매우 높다는 단점이 있으며 PKI를 유비쿼터스 환경에 구축하기 위해서는 각각의 서비스 디바이스마다 공개키와 개인키를 생성하고 관리를 해야 하며, 각 디바이스가 RSA 연산을 수행해야 하기 때문에 비용적인 측면과, 연산 오버헤드 측면에 있어서 단점으로 작용된다.

안전한 유비쿼터스 서비스를 제공하기 위하여 보안 인프라를 구축하기 위한 기존의 방법으로 각각의 서비스 디바이스와 사용자마다 공개키와 개인키를 생성하고 인증을 위하여 RSA 알고리즘을 수행할 수 있는 고속 프로세서를 내장시켜야 한다면 시스템을 구축하는데 있어 비용적인 측면이 단점이 될 수 있다.

본 논문에서는 이와 같은 단점을 극복하여 연산 능력이 좋지 않은 초소형 단말기를 이용하는 환경에서 유비쿼터스 환경에 필요한 보안 기능을 제공하는 기법과, 유비쿼터스 환경에 적합한 SSO 프로토콜을 제안하고 분석한다. 본 논문에 따른 SSO 프로토콜은 사용자에게 투명한 보안 메커니즘과 끊김없는 인증 기능을 제공하며, 제시된 위임서버는 사용자의 인증 연산을 위임 메커니즘과 사용자의 위치 정보를 이용하여 기존의 보안 수준과 기능을 유지하되 인증 지연시간과 유비쿼터스 환경의 보안 인프라의 구축비용을 획기적으로 줄이고 병목현상을 해결한다.

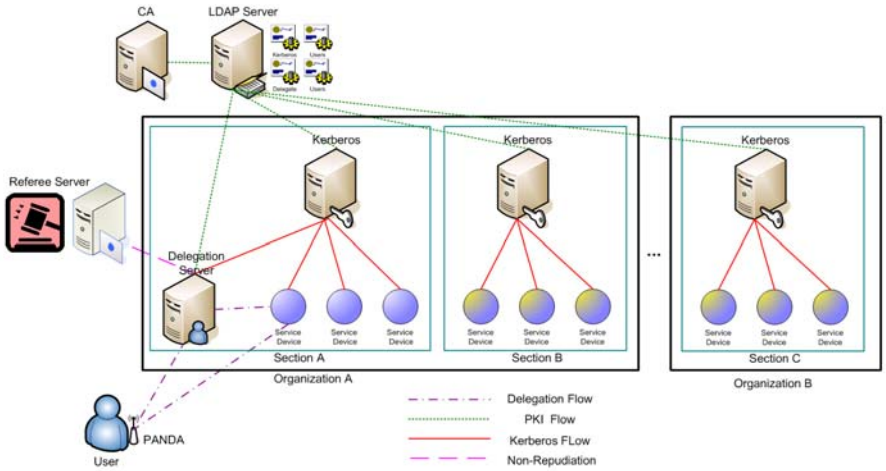
본 논문의 구성은 다음과 같다. 2장은 유비쿼터스 환경에 보안 인프라를 구축하기 위한 프로토콜 요구사항을 분석한다. 3장은 제안한 보안 인프라와 프로토콜을 제공하기 위한 구성요소와 기능을 설명한다. 4장은 제안한 프로토콜을 기술하였으며, 5장은 제안한 프로토콜과 인프라를 기존의 방식과 비교 분석하였으며, 6장에서 결론을 맺는다.

2. 유비쿼터스 환경에 적합한 보안 인프라 및 SSO 프로토콜 요구사항

2.1 보안적 측면

- 기밀성 : 사용자와 서비스 디바이스 사이에 - User와 Service사이에 안전한 통신을 제공해야 한다. [12]
- 인증, 권한, 과금(AAA[1]) : 디바이스가 서비스를 제공하는데 있어서 상호인증과 사용자의 권한, 과금 등의 기능이 제공되는 인프라이어야 한다. 이를 위

[1] Authentication, Authorization, Accounting



<그림1> : 본 논문에서 제시된 보안 인프라의 전반적인 구조

하여 부인방지(Non-Repudiation) 및 디지털 서명(Digital Signature) 등의 기능이 제공되어야 한다. [13]

2.2 사용자 측면

- 인증 지연 최소화 : 인증을 수행하는데 있어서 인증 지연 시간을 최소화 시켜야 한다.
- 사용자 개입 최소화 : 인증 및 권한을 검증하는데 있어서 사용자의 편리성을 위하여 보안을 유지하되 사용자의 개입을 최소화 시켜야 한다.

2.3 시스템 측면

- 구현 비용 최소화 : 시스템을 구현하는데 있어서 구축비용 측면에 있어서 효율적인 시스템이어야 한다.
- 확장성(Scalability) : 인프라의 규모에 따라 확장이 될 수 있는 인프라 구조를 가져야 한다.
- 상호 연산성(Interoperability) : 특정 도메인에 관계없이 보안 인증이 가능해야 한다.

3. 시스템 디자인과 보안 인프라의 구성요소

3.1 시스템 요구사항에 따른 시스템 디자인

논의한 시스템 요구사항을 종합하여 보안적 측면, 사용자 측면, 시스템 측면의 요구사항을 만족하고 유비쿼터스 환경에 적합한 효율적이고 강력한 보안 인프라에 대한 구성을 제안한다.

첫째, 보안적 측면에 있어서 확장성과 상호 연산성을 제공하고 서비스 제공에 필수적인 기능인 AAA와 이에 따른 디지털 서명기능 부인방지 기능을 제공하기 위하여 PKI를 적용시킨다. PKI는 비대칭키 기반의 보안 인프라로서 RSA 알고리즘을 이용하여 인증을 수행하며, 대칭키에 비해 효율적인 키 관리가 가능하다[14]. 하지만 RSA 알고리즘은 소수(Prime Number) 생성과 대수에 대한 승수 연산으로 사용자가 가지고 다니는 초소형 단말기나, 서비스 디바이스에 적용시키기에는 오버헤드가 커지며 구축비용이 증가하게 된다.

둘째, 사용자 측면에 있어서 인증 연산을 하는데 있어서 지연시간과 사용자의 개입을 최소화하기 위하여 대행 인증서(Proxy Certificate)[15,16]를 이용한 위임기법을 적용시키고 Single Sign-On 프로토콜을 제안한다. 본 논문에서는 위임기법을 이동성이 있는 사용자와 유비쿼터스 환경에 편재되어 있는 여러 서비스 디바이스간의 인증 연산을 효율적으로 수행하기 위하여 사용자의 위치를 예측하여 기존 PKI의 보안성을 유지시키되 사용자의 인증 지연시간을 최소화 시키는 지능적인 위임서버를 제안한다.

셋째, 시스템 측면에 있어서 시스템의 병목현상을 최소화 시키고 인프라 구축비용을 최적화하기 위하여 비대칭키 기반 보안 인프라에 연산적으로 효율적인 대칭키 인증 기법을 적용한다. 이를 위하여 시스템의 병목현상을 제거하기 위하여 유비쿼터스 환경을 섹션으로 나누어 보안 연산에 대한 트래픽을 분산시키고 PKI와 동일한 보안 기능을 제공하는 위임 메커니즘과 대칭키 기반 인증 메커니즘을 접목시킨 새로운 인프라를 제안한다.

3.2 제안된 보안 인프라의 구성요소

본 논문에서는 유비쿼터스 환경에 존재하는 서비스 디바이스를 섹션으로 나누고 각 섹션마다 커버로스 서버를 두고 각 커버로스 서버는 PKI의 개체가 되는 구조로 설계된다. 이와 같은 구조에서는 여러 서비스 디바이스의 공개키 개인키 쌍들을 커버로스가 대표하여 하나의 개인키와 공개키 쌍을 사용하게 되므로 생성되는 공개키, 개인키 쌍의 수가 현격히 줄어들게 되어 키 관리 문제가 용이하게 되며, 보안 인프라에서 발생되는 인증 트래픽을 분산시켜 병목현상을 제거시킬 수 있다. 각 서비스 디바이스들은 커버로스에 필요한 대칭키 암호연산을 수행할 수 있는 연산 능력만 갖추면 사용자의 인증이 가능하므로 구축비용 측면에서도 매우 효율적이다. 또한, 사용자의 연산 오버헤드를 낮추기 위하여 위임 서버를 도입하였다. 본 논문에서 위임 서버는 사용자를 대신하여 인증에 관련된 연산을 수행하는 서버를 의미한다. 또한 기존의 위임 메커니즘은 위임한 기간에 사용자의 인증 요청에 대한 부인 방지 기능을 제공하지 못한다. 위임이 된 환경에서도 사용자의 부인방지를 위하여 심판 서버를 도입하였다. 심판 서버는 부인방지를 위한 증거자료를 수집하는 서버로서 인증 연산에 대한 부인을 방지할 수 있는 메커니즘을 제공한다.

그림1은 제안된 보안 인프라의 전반적인 구성요소를 보여준다. 사용자, 위임 서버, 커버로스 서버의 공개키를 관리하는 CA와 인증서를 배포하는 LDAP 디렉토리 서버가 있으며, 각 사용자와 위임서버, 커버로스 서버들은 각각 개인키와 공개키를 생성하여 가지고 있다.

· **사용자:** 각 사용자는 서비스를 받는데 필요한 인증, 권한, 과금 등의 기능을 초소형 보안 단말기를 이용하여 받게 된다. 본 논문에서 제시하는 초소형 보안 단말기는 저사양의 구현비용이 매우 저렴한 장치로서 본 논문에 실제 적용시킨 단말기는 PANDA[2][17,18]이다. PANDA는 지그비통신이[19] 가능한 초소형 단말기로서 내부에 통신모듈과 프로세서가 장착되어 있다.

· **서비스 디바이스:** 서비스를 제공하는 장치로서 서비스를 제공하는 메인 디바이스에 부착되는 형태로 구성된다. 서비스 디바이스 역시 사용자가 가지고 있는 보안 단말기와 비슷한 사양의 장비로 구성될 수 있다.

· **CA:** 인증서를 관리하고 유비쿼터스 환경에 존재하는 각 개체의 인증서에 대한 서명을 하는 기능을 담당한다.

· **LDAP 디렉토리 서버:** 환경에 존재하는 각 개체의 인증서를 저장하고 각 개체가 인증시 필요한 인증서를 응답하게 된다.

· **커버로스 서버:** 전형적인 대칭키 기반의 인증 서버로서 각 섹션별로 위치하며 하나의 커버로서 서버는 섹션의 인증연산을 위한 TGT(Ticket Grant Ticket), SGT(Session Grant Ticket)을 생성하고 관리하게 된다.

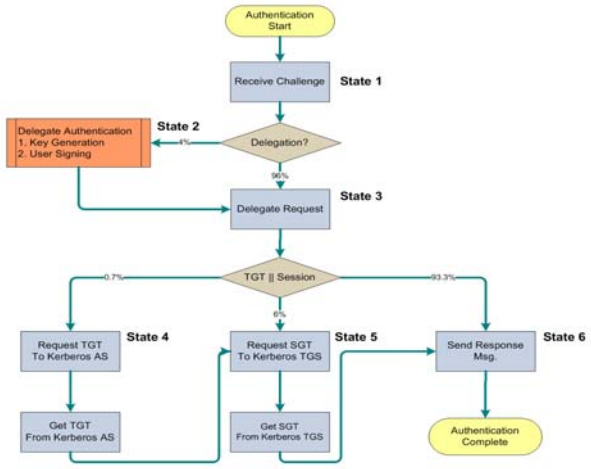
· **위임 서버:** 위임서버는 위임받은 사용자의 대리 개인키와 공개키를 가지고 있으며 위임된 개인키는 사용자의 개인키를 이용하여 서명이 되어 있으며 이를 대행 인증서라 한다. 처음 보안 인프라에 들어온 사용자는 먼저 위임 서버에게 자신의 인증을 위임하게 된다. 위임에 관련된 연산은 RFC3820[21]을 따른다. 대행 인증서에는 위임 기간 및 권한 등을 기재하여 위임을 하게 된다. 위임을 한 이후에는 사용자가 위임을 해지하거나, 위임 기간이 지나지 않은 이상 사용자의 요청에 따라 인증 연산을 대행하여 처리하게 된다. 여기서 위임 서버는 인증 지연시간을 줄이기 위하여 위치 서버를 이용하여 사용자의 위치 정보를 이용하여 사용자가 사용할 서비스 디바이스를 예측하여 해당하는 서비스에 해당하는 TGT과 SGT을 미리 받아 사용자의 인증에 필요한 응답 메시지를 신속하게 생성하여 서비스 디바이스에게 전달해주는 역할을 수행한다.

· **심판 서버:** 본 논문에서 제시하는 심판서버는 사용자의 부인 방지를 위한 기능을 가지고 있다. 위임 서버에서 사용되는 대행 인증서를 이용한 인증 메커니즘에서는 위임서버가 사용자의 인증을 대행하게 되므로 공개키 기반의 PKI를 사용하더라도 실제 사용자에 대한 부인 방지 기능을 제공할 수 없게 된다. 제시한 심판서버는 보안 인프라에서 각 개체만이 알고 있는 키를 이용하여 인증되었다는 사실에 대한 증거를 저장하여 사용자의 인증 증명 시 저장되어 있는 증거 자료를 이용하여 부인을 방지하는 메커니즘을 제공한다.

4. 제안하는 Single Sign-On프로토콜

4.1 Single Sign-On프로토콜 흐름도

그림2는 제안하는 Single Sign-On 프로토콜의 흐름도를 나타낸다. 프로토콜은 6개의 상태로 구성되어 실행된다.



<그림2> Single Sign-On프로토콜 흐름도

(State1): 사용자가 서비스 디바이스에 접근하여 해당 서비스를 받으려는 경우 사용자는 챌린지 메시지를 받게 된다.

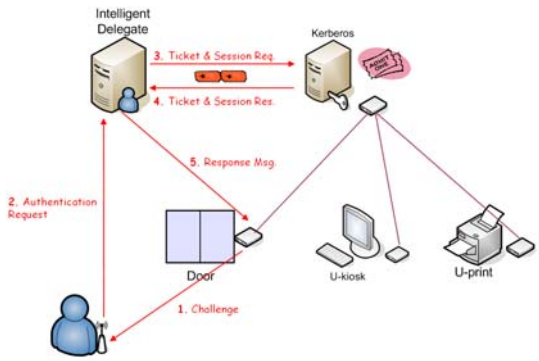
(State2): 사용자가 보안 인프라에 처음 접근할 시에는 위임 서버에게 자신의 인증 연산을 위임시키게 된다. 이를 위해서는 위임 서버가 생성한 공개키와 개인키 중 공개키를 사용자에게 보내고 사용자는 해당 공개키를 자신의 개인키로 서명을 하여 대행 인증서를 생성시켜 이를 다시 위임 서버에게 보내게 된다. 이로서 사용자는 자신이 수행할 RSA 연산을 위임 서버에게 위임을 시킴으로써 다른 서비스에 대한 인증 연산을 수행할 시에 필요한 RSA 연산을 자신이 수행하지 않고 사용자가 위임한 위임 서버가 수행하게 된다.

(State3): 사용자가 위임 서버로 위임을 한 후 서비스에 대한 인증을 수행하기 위해서는 서비스 디바이스가 보낸 챌린지 메시지를 받은 사용자는 수신한 메시지를 자신이 위임한 위임 서버에게 전달을 하게 된다.

(State4, 5, 6): 챌린지 메시지를 수신한 위임서버는 해당하는 서비스의 커버로스 서버에 접속을 하여 서비스 디바이스에 해당하는 티켓과(State 4) 세션을(State5) 얻은 후에 사용자 인증에 필요한 응답 메시지를 서비스에 전송하여 인증이 이루어지게 된다. (State6)

그림3은 프로토콜 흐름도에 기술된 위임 및 인증 메커니즘을 나타내고 있다. 사용자가 위임 서버에게 자신의 권한을 위임 시켰다고 가정하면 다음과 같은 흐름으로 인증 절차가 이루어진다. 먼저 서비스 디바이스는 사용자에게 챌린지 메시지를 보낸다. 사용자는 수신한 메시지를 위임서버에게 전송하면 위임서버는 해당하는 서비스에 해당하는 커버로스 서버를 통하여 TGT를 얻고 이를 통해 SGT를 얻게 된다. SGT를 이용하여 응답 메시지를 서비스

[2] PANDA : Personal Authentication Network Device Architecture



<그림3> SSO 프로토콜 흐름에 따른 인증 및 위임 메커니즘

디바이스에게 보내 인증이 완료가 된다. 이때 위임 서버가 위치 서버로부터 받은 정보를 이용하여 사용자가 앞으로 사용할 서비스 디바이스를 예측할 수 있다면 TGT와 SGT을 미리 수신하여 보다 빠른 인증 시간을 제공할 수 있게 된다.

4.2 Single Sign-On 프로토콜 명세

제안하는 Protocol을 명시하는데 사용되는 개체(Entity)와 메시지(Message)에 대한 정의는 다음과 같다.

◆ 개체 기호 정의
· 사용자 : Alice
· 위임 서버 : Delg
· 커버로스 인증 서버(AS) : Kerb_AS
· 커버로스 티켓허가서버(TGS) : Kerb_TGS
· 서비스 디바이스 : Serv
◆메시지 기호 정의
· IDx : X의 ID
· PUx : X의 공개키
· PRx : X의 개인키
· PUx,y : Y에 의해 생성된 X의 위임된 공개키
· PRx,y : Y에 의해 생성된 X의 위임 개인키
· CRx,y : Y에 의해 서명된 X의 인증서(대행 인증서)
· Nonce : Replay 공격에 대비한 무작위 생성 데이터
· Kx,y : X와 Y사이에 공유하는 공유키(대칭키)
· Delg_X : X의 위임 프로세스

4.2.1 State 1

서비스 디바이스는 자신의 서비스 ID와 서비스 캡슐을 계속해서 폴링하게 된다. 이때 서비스 캡슐은 랜덤한 데이터와 고유번호(SN)의 해쉬값으로 한 명의 사용자(Alice)가 인증될 때 마다 또는 일정 주기로 내부 내용은 바뀌게 된다. Alice가 서비스 디바이스로부터 인증을 받고 싶으면 (State1)으로 이동한다.

·**메시지 1-1.** Serv→Alice: ID_{Serv}||Serv_Capsule
Serv_Capsule = SN||Hash(SN||$Nonce_{Serv}$)

※ 서비스 디바이스 (메시지 1-1)

①서비스 디바이스는 자신의 서비스 디바이스 ID와 Serv_Capsule 을 폴링하게 된다.

②만약 사용자 Alice가 서비스 디바이스에 인증을 하기 원하게 된다면 해당 메시지를 수신하게 된다.

4.2.2 State 2

만약 Alice가 위임 서버에 자신의 인증을 위임하지 않았다면 위임 메커니즘을 수행하는 (State2)로 이동하여 처리한 후 (State3)으로 이동한다. 이미 Alice가 인증을 위임하였다면 (State3)으로 이동한다.

·**메시지 2-1.** Alice→Delg: E{PU_{Delg}, Delg Request}
Delg Request = ID_{Delg}||ID_{Alice}||$K_{Alice,Delg}$||$Nonce_{Alice}$||E{PR_{Alice},$K_{Alice,Delg\_A}$||E{$PU_{Referee}$, K_{Repu_Alice}}}

·**메시지 2-2.** Delg→Alice: E{$K_{Alice,Delg\_A}$,$PU_{Alice,Delg\_A}$||$Nonce_{Alice}$}

·**메시지 2-3.** Alice→Delg: E{$K_{Alice,Delg\_A}$, $CR_{Delg\_A,Alice}$}

·**메시지 2-4.** Delg→Referee: E{$PU_{Referee}$,$K_{Delg\_A,Referee}$||$CR_{Delg\_A,Alice}$|| $Nonce_{Referee}$|| E{PR_{Delg},$CR_{Delg\_A,Alice}$||E{$PU_{Referee}$,K_{Repu_Alice}||ID_{Alice}||ID_{Delg_A}}}

·**메시지 2-5.** Referee→Delg: E{PU_{Delg},Seq_{Alice}||$Nonce_{Referee}$}

※ Alice (메시지 2-1)

①Alice는 부인 방지를 위한 키를 무작위수 생성기로 생성한다. : K_{Repu_Alice}

②생성한 Key (K_{Repu_Alice}), ID_{Alice}, ID_{Delg}를 심판 서버의 공개키 $PU_{Referee}$를 이용하여 암호화 시킨다.

③위임 요청(Delegation Request) 메시지 생성

④생성된 위임 요청 메시지를 위임 서버 공개키로 암호화

⑤생성된 메시지(E{PU_{Delg},Delg Request})를 위임 서버로 전송

※ 위임 서버 (메시지 2-2)

①Alice로부터 위임 요청(Delegation Request) 메시지 수신 (E{PU_{Delg}, Delg Request})

②위임 요청 메시지를 위임 서버의 개인키 (PR_{Delg})를 이용하여 복호화

③Alice의 공개키를 이용하여 E{PR_{Alice},$K_{Alice,Delg\_A}$||E{$PU_{Referee}$,K_{Repu_Alice}}를 복호화.

④복호화된 $K_{Alice,Delg_A}$와 위임 요청 메시지 내에 있는 $K_{Alice,Delg_A}$가 일치하면 Alice 인증 완료.

⑤Alice의 위임을 위한 Key pair ($PU_{Alice,Delg\_A}$,$PR_{Alice,Delg\_A}$) 생성

⑥Alice로부터 Signing을 받기 위해 생성된 공개키($PU_{Alice,Delg\_A}$)를 Alice로부터 수신한 $K_{Alice,Delg_A}$로 암호화 후 Alice에게 $Nonce_{Alice}$와 함께 전송

※ Alice (메시지 2-3)

①위임 서버로부터 받은 메시지(E{ $K_{Alice,Delg\_A}$, $PU_{Alice,Delg\_A}$ || $Nonce_{Alice}$})를 복호화

②수신한 메시지에서 $Nonce_{Alice}$를 이용하여 위임 서버인증

②수신한 $PU_{Alice,Delg\_A}$, Expire time, 위임할 권한을 자신의 개인키(PR_{Alice})를 이용하여 서명하여 Proxy 인증서($CR_{Delg\_A,Alice}$)생성

③생성된 $CR_{Delg_A,Alice}$을 $K_{Alice,Delg_A}$로 암호화하여 위임 서버로 전송

④Expire Time 까지 위임 완료

※ 위임 서버(메시지 2-4)

①Delegation Request 메시지 중 E { $PU_{Referee}$, K_{Repu_Alice}}와 Alice로부터 받은 Proxy 인증서 ($CR_{Delg\_A,Alice}$)를 위임 서버의 개인키로 서명.

②암호화된 메시지를 $K_{Delg\_A,Referee}$, $CR_{Delg\_A,Alice}$, $Nonce_{Referee}$와 함께 심판 서버의 공개키로 암호화.

③위에서 생성된 메시지를 심판 서버로 보낸다.

※ 심판 서버 (메시지 2-5)

①위임 서버로부터 받은 메시지를 자신의 개인키로 복호화.

②복호화된 메시지를 위임 서버의 공개키를 이용하여 복호화하고 $CR_{Delg_A,Alice}$를 비교하여 위임 서버에 대한 인증 완료.

③위임 서버로부터 수신한 Proxy 인증서를 이용하여 Alice 인증 완료.

④심판 서버의 개인키를 이용하여 E{$PU_{Referee}$,K_{Repu_Alice}}를 복호화 시킨 후 Alice에 대한 Sequence Number 부여.

⑤Sequence Number, $Nonce_{Referee}$를 $K_{Delg_A,Referee}$를 이용하여 암호화하여 위임 서버에 전송.

⑥Alice의 부인 방지를 위한 자료 완성.

4.2.3 State 3

위임 서버가 (State2)연산을 통해 위임되었거나 이미 되었다면 위임 서버가 Service에 대한 인증을 수행할 수 있도록 Request 메시지를 보내기 위하여 (State3)으로 이동하려 처리 후 (State4)로 이동한다.

·**메시지 3-1.** Alice→Delg: ID_{Alice}||E{$K_{Alice,Delg\_A}$,ID_{Alice}||ID_{Serv}|| Serv_Capsule ||$Subkey_{Alice,Serv}$||E{K_{Repu_Alice}, Serv_Capsule}}

·**메시지 3-2.** Delg→Referee: $ID_{Delg\_A}||E\{K_{Delg\_A,Referee}, ID_{Serv}||Seq_{Alice}||Nonce_{Referee}||Serv\_Capsule||E\{PR_{Alice,Delg\_A}, E\{K_{Repu\_Alice}, Serv\_Capsule\}\}\}$

·**메시지 3-3.** Referee→Delg: $E\{ K_{Delg\_A,Referee}, OK \text{ or } BAD \| Nonce_{Referee} \}$

※ Alice (메시지 3-1)

①Alice는 자신의 ID와 함께 서비스 디바이스로부터 수신한 서비스 디바이스 ID와 Serv_Capsule과 $Subkey_{Alice,Serv}$를 $K_{Alice,Delg_A}$를 이용하여 암호화한다.

②생성된 메시지를 사용자의 ID (ID_{Alice})와 함께 위임 서버로 전송시킨다.

※ 위임 서버(메시지 3-2)

①Alice로부터 수신된 메시지중 $E\{K_{Repu_Alice}, Serv_Capsule\}$를 $PR_{Alice,Delg_A}$를 이용하여 암호화한다.

②서비스 디바이스 ID와 심판 서버 등록 시 생성되었던 Seq_{Alice}, $Nonce_{Referee}$와 ①번에서 생성되었던 내용을 $K_{Delg_A,Referee}$로 암호화시킨다.

③메시지를 심판 서버에 전송시킨다.

※ 심판 서버 (메시지 3-3)

①수신된 메시지를 $K_{Delg_A,Referee}$를 이용하여 복호화한다.

②심판 서버는 ID_{Serv}, Seq_{Alice}에 따라 분류하여 Serv_Capsule와 $E\{PR_{User,Delg_A}, E\{K_{Repu_User}, Serv_Capsule\}$를 $CR_{User,Delg_A}$에 저장되어 있는 $PU_{User,Delg_A}$를 이용하여 복호화 시킨다.

③K_{Repu_User}를 이용하여 $E\{K_{Repu_User}, Serv_Capsule\}$를 복호화시킨다.

④Serv_Capsule을 비교하여 같을 경우에는 OK 메시지를 보내고 그러지 않을 경우에는 BAD 메시지를 $Nonce_{Referee}$와 함께 전송시킨다.

4.2.4 State 4

만약 위임 서버가 해당하는 Service에 대한 TGT (Ticket Grant Ticket)을 가지고 있지 않다면 다음과 같이 TGT를 얻는 작업을 수행하기 위하여 (State4)로 이동한다. 만약 이미 해당하는 서비스 디바이스에 대한 TGT를 가지고 있다면 (State5)로 이동한다.

·**메시지 4-1.** Delg→Kerb_AS: $E\{PU_{Kerb}, TGT\text{-}REQ||CR_{Alice,Delg\_A}||E\{PR_{Delg}, CR_{Alice,Delg\_A}||K_{Kerb\_AS,Delg\_A}\}\}$

TGT-REQ = Ticket Grant Ticket Request
= PA_Data ‖ ID_{Alice} ‖ ID_{TGS} ‖ Time ‖ $Nonce_{TGT}$

PA_Data = Pre-Authentication Data
= $E\{K_{Kerb\_AS,Delg\_A}$, System Time$\}$

Time = Key-Expiration||Auth-Time||Start-Time‖ End-Time

·**메시지 4-2.** Kerb_AS→Delg: $ID_{Alice}||TGT_{Alice,TGS}||E\{K_{Delg\_A}, K_{Kerb\_AS,Delg}||Nonce_{TGT}||Times\}$

$TGT = ID_{Kerb\_TGS}||E\{K_{AS,TGS}, K_{Delg\_A,Kerb\_TGS}||Times||ID_{Alice}\}$

※ 위임 서버(메시지 4-1)

①TGT-REQ 생성 및 PR_{Delg}를 이용하여 $CR_{Alice,Delg_A}$와 $K_{Kerb,Delg_A}$를 암호화시킨다.

②생성된 메시지를 커버로스 서버로 전송시킨다.

※ 커버로스 AS(Authentication Server) (메시지 4-2)

①위임 서버로부터 받은 메시지를 PR_{Kerb}를 이용하여 복호화한다.

②수신된 메시지 중 $E\{PR_{Delg}, CR_{Alice,Delg_A}||K_{Kerb,Delg_A}\}$부분을 위임 서버의 공개키 ($PU_{Delg}$)를 이용하여 복호화 시킨다.

③복호화된 $CR_{Alice,Delg_A}$와 메시지에 내장된 $CR_{Alice,Delg_A}$를 비교하여 일치하면 위임 서버에 대한 인증 완료.

④$CR_{Alice,Delg_A}$의 Trust Relation (Proxy Path)를 이용하여 검증 (Proxy 인증서 검증 완료)

⑤Ticket Grant Ticket (TGT_{Alice})를 발부하고 TGS와 통신을 위해 필요한 정보($K_{Delg\_A,Kerb\_TGS}$, $Nonce_{TGT}$, Times)를 $K_{Delg_A,Kerb_AS}$를 암호화하여 보내준다.

4.2.5 State 5

만약 Alice가 이미 TGT를 가지고 있거나, (State4)와정을 통해 TGT를 얻었다면 해당하는 서비스 디바이스에 대한 SGT(Session Grant Ticket)을 얻기 위하여 (State5)으로 넘어간다. 만약 위임 서버가 해당하는 서비스 디바이스에 대한 SGT을 가지고 있다면 (State6)으로 넘어간다.

·**메시지 5-1.** Delg→Kerb_TGS: $TGS_{Alice\text{-}REQ}$

$TGS_{Alice\text{-}REQ} = ID_{Serv}||Times||Nonce_{TGS}||TGT_{Alice}||Authenticator(Alice, Kerb\_TGS)$

$TGT_{Alice} = ID_{Kerb\_TGS} \| E\{ K_{AS,TGS}, K_{Delg\_A,Kerb\_TGS} \| Times \| ID_{Delg\_A} \}$

$Authenticator(Alice, Kerb\_TGS) = E\{K_{Delg\_A,Kerb\_TGS}||ID_{Alice}||K_{Delg\_A,Ser}\}$

·**메시지 5-2**. Kerb_TGS → Delg: $TGS_{Alice\text{-}REP}$

$TGS_{Alice\text{-}REP} = ID_{Alice}||SGT_{Alice}||E\{K_{Delg\_A,TGS}, K_{Delg\_A,Serv}||Nonce_{TGS}||Times\| ID_{Serv}\}$

$SGT_{Alice} = ID_{Kerb\_TGS}||E\{K_{Delg\_A,Serv}||ID_{Alice}||Times\}$

※ 위임 서버(메시지 5-1)

①Kerb_AS로부터 받은 메시지 중 $Nonce_{TGT}$를 이용하여 TGT에 대한 검증을 수행한다.

②수신된 TGT를 이용하여 Alice가 원하는 Service에 대한 TGS 발급을 요청한다.

※ 커버로스 TGS(Ticket Grant Server) (메시지 5-2)

① 위임 서버로부터 받은 $TGS_{Alice\text{-}REQ}$중 TGT_{Alice}를 이용하여 위임 서버를 검증한다.

② 요청한 서비스 디바이스에 대한 $TGS_{Alice\text{-}REP}$를 생성한다.

② 생성된 $TGS_{Alice\text{-}REP}$ 메시지를 위임 서버로 전송한다.

4.2.6 State 6

위임 서버가 이미 SGT를 가지고 있거나 (State5) 과정을 통해 얻었다면 다음 연산과정을 수행한다.

·**메시지 6-1.** Delg→Serv: $SGT_{Alice}||Nonce_{Serv}||E\{K_{Delg\_A,Serv}, ID_{Alice}||Subkey_{Alice,Serv}\}$

·**메시지 6-2.** Serv→Delg: $E\{K_{Delg\_A,Serv}, Subkey_{Alice,Serv}||Nonce_{Serv}\}$

·**메시지 6-3.** Alice→Serv: $ID_{Alice}||E\{Subkey_{Alice,Serv}, Serv\_Capsule\}$

※ 위임 서버(메시지 6-1)

① Alice가 보내준 $Subkey_{Alice,Serv}$와 ID_{Alice}를 $K_{Delg_A,Serv}$를 이용하여 암호화시킨다.

② Kerb_TGS로부터 받은 SGT_{Alice}와 생성된 $Nonce_{Serv}$를 서비스 디바이스로 전송시킨다.

※ 서비스 디바이스 (메시지 6-2)

①수신된 메시지 중 SGT_{Alice}를 검증하고 $K_{Delg_A,Serv}$를 이용하여 $E\{K_{Delg,Serv}, ID_{Alice}||Subkey_{Alice,Serv}\}$를 복호화한다.

③위임 서버로부터 받은 $Subkey_{Alice,Serv}$와 $Nonce_{Serv}$를 $K_{Delg_A,Serv}$를 이용하여 암호화시켜 전송시킨다.

④생성된 메시지를 위임 서버로 전송시킨다.

⑤서비스 디바이스는 Alice로부터 Confirm메시지가 수신될 때까지 대기한다.

※ Alice (메시지 6-3)

① Alice는 $Subkey_{Alice,Serv}$를 이용하여 Serv_Capsule를 암호화 시킨다.

② 서비스 디바이스에게 생성한 메시지를 전송한다.

③ 상호 인증 완료

4.2.7 User에 대한 부인방지 검증 메커니즘

※ 심판 서버에 사용자 Alice에 대하여 저장되어 있는 값은 다음과 같다.

· ID_{Alice}
· Seq_{Alice}
· Serv_Capsule
· $E\{ K_{Repu\_Alice}, Serv\_Capsule \}$

①심판 서버는 ID로부터 Non Repudiation 검증 Request 메시지를 받으면 저장되어 있는 $E\{PR_{Alice,Delg_A}, E\{K_{Repu_Alice}, Serv_Capsule\}\}$를 위임 서버의 Proxy 인증서를 이용하여 복호화 시켜 $E\{K_{Repu_Alice}, Serv_Capsule\}$를 얻는다.

②저장되어 있는 K_{Repu_Alice}를 이용하여 위임 서버로부터 수신된 Serv_Capsule 메시지와 비교한다.

③서비스 디바이스는 심판 서버에게 SN와 $Nonce_{Serv}$를 제출.

④SN와 $Nonce_{Serv}$를 해쉬 함수에 입력하여 Nonce α를 얻는다.

⑤얻은 Nonce α 값이 서비스 디바이스가 보낸 것과 일치하면 사용자가 수신한 Challenge 메시지를 위임 서버

로 포워딩한 것이 증명이 되므로 부인 방지 기능을 제공할 수 있게 된다.

5. 제안한 프로토콜의 분석 및 효과

제안한 프로토콜은 유비쿼터스 환경에 존재하는 다수의 서비스 디바이스와 사용자간의 인증연산을 수행하기 위하여 기존의 PKI에서 사용되는 RSA알고리즘, 대행 인증서 발행을 위한 위임 메커니즘, 대칭키 기반 인증시스템인 커버로스를 도입하였다. 기존의 PKI와 위임 메커니즘, 커버로스 메커니즘을 사용하는 프로토콜을 안정성, 시스템, 인프라 측면을 고려하여 분석하였다.

5.1 안전성 측면

사용자가 인프라를 처음 접근 할 경우 PKI를 통한 인증을 통해 사용자의 위임서버간의 인증을 수행하고 위임 연산을 위한 키 교환을 수행한다. 본 논문에서 제시한 프로토콜의 한 요소인 위임 서버는 PKI와의 안전한 연동을 위하여 위임 기간을 사용자가 설정하고 일정 시간동안 위임서버에게 자신의 인증 연산을 위임시키도록 한다. 그리고 위임 요구 메시지와 인증에 필요한 모든 메시지에는 송신자가 첨부한 Nonce를 추가시키고 위임기간 동안 제한적으로 사용되는 공유키를 사용하게 한다. 이는 연산 오버헤드를 획기적으로 줄이는 반면 송신자의 Nonce 메시지를 통해 메시지 재연 공격(Replay Attack)을 방지하고 사용자와 인증서버 간의 초기 인증 시 PKI를 통한 인증을 수행하도록 하여 MTM 공격(Man in the middle Attack)에 안전하도록 설계하였다.

[정리 1] 본 제안 기법은 메시지 재연 공격에 안전하다.

증명 : 전체 인증 경로를 사용자(Alice) - 위임서버(Delg) - 커버로스서버(Kerb) - 서비스 디바이스(Serv)로 가정한 경우 다음의 경우에 대한 안정성 증명을 한다.

(A) 악의 사용자가 위임 요청 메시지(Delegation Request)를 재연한 경우

(B) 악의 사용자가 위임 응답 메시지(메시지 2-2)를 재연한 경우

(C) 인증 요청 메시지를 재연한 경우

(D) 커버로스 TGT 및 SGT 요청 메시지를 재연한 경우

(E) 위임서버와 서비스 간 Response 메시지(메시지 6-2)를 재연한 경우

· (A)의 경우 공격자는 사용자가 위임서버에 전송한 위임 요청 메시지를 캡처하여 이를 다시 사용할 수 있다. 그러나 위임 요청메시지(메시지 2-1)에 의해 전송되는 메시지에는 위임서버가 무작위로 생성한 위임용 공개키 $PU_{Alice,Delg_A}$가 들어있고 사용자는 수신한 $PU_{Alice,Delg_A}$를 자신의 개인키인 PR_{ALice}로 Encrypt시켜 주어야 한다. 하지만 $PU_{Alice,Delg_A}$는 위임 시에 무작위로 생되고 위임기간이 끝나면 버려지므로 위와 같은 메시지 재연 공격은 성공할 수 없다.

· (B)의 경우 공격자는 위임 응답메시지를 캡처하여 이를 다시 사용할 수 있다. 그러나 위임 응답 메시지에는 사용자가 무작위로 생성시킨 $Nonce_{Alice}$가 들어있고 이는 메시지가 전송될 때마다 한번 사용되고 버려지므로 위와 같은 메시지 재연 공격은 성공할 수 없다.

· (C)(E)의 경우 공격자는 사용자의 인증 요청 메시지 또는 위임서버와 서비스 디바이스 간 Response 메시지(메시지 6-2)를 캡처하여 이를 다시 사용할 수 있다. 그러나 인증 요청 메시지에는 서비스 디바이스가 생성한 Serv_Capsule이 들어 있고 이는 인증시마다 내부의 내용이 바뀌게 되어 한번 사용되고 버려지므로 위와 같은 메시지 재연 공격은 성공할 수 없다.

· (D)의 경우 공격자는 위임서버가 커버로스 서버에 대한 메시지 또는 커버로스 서버가 위임서버에게 보내지는 메시지를 공격자가 캡처하여 이를 다시 사용할 수 있다. 하지만 커버로스와 위임서버간의 메시지에는 $Nonce_{TGT}$, $Nonce_{TGS}$가 있고 각 Nonce 값은 한번사용이 되고 매번 바뀌게 되므로 메시지 재연 공격은 성공할 수 없다. 그러므로 본 제안 기법은 메시지 재연 공격에 안전하다.

[정리 2] 본 제안 기법은 MTM 공격에 안전하다.

증명 : [정리 1]과 동일한 인증 경로의 경우 다음의 경우에 대한 안정성 증명을 한다.

(A) 사용자와 위임서버간의 MTM 공격

(B) 사용자와 서비스간의 MTM 공격

(C) 위임서버와 커버로스 서버간의 MTM공격

(D) 위임 서버와 서비스 디바이스간의 MTM공격

· (A)(C)의 경우 공격자는 각 개체 사이에서 양방향 공격을 수행할 수 있다. 하지만 각 개체는 상호 인증을 위하여 각 개체의 개인키 PR_{Delg}, PR_{Alice}, PR_{Kerb}를 사용하여 인증을 하므로 각 인증에 대한 Challenge 메시지에 대한 Response 메시지를 생성할 수 없으므로 위와 같은 MTM 공격은 성공할 수 없다.

· (B)(D)의 경우 사용자의 인증을 위하여 서비스 디바이스는 서비스 캡슐(Serv_Capsule)을 생성하여 보낸다. 서비스 캡슐의 내용은 매번 바뀌게 되고 서비스 개체는 서비스 캡슐의 유효성 여부를 이용하여 인증을 하며, 서비스 개체는 커버로스 서버와 이미 공유된 공유키로 암호화된 Response 메시지가 전송되므로 위와 같은 MTM 공격은 성공할 수 없다.

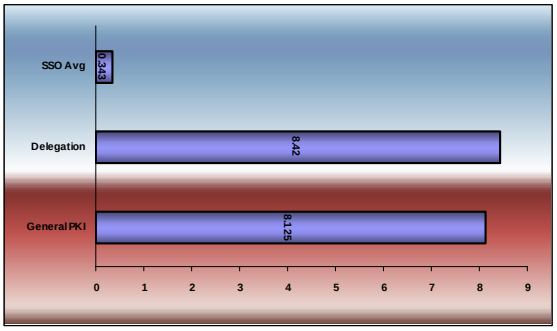
또한, 사용자는 위임서버에게 자신의 인증 연산을 위임을 시킬 경우 재판 서버에게 사용자의 비밀 개인키를 재판 서버의 공개키와 사용자의 개인키로 암호화 시켜 전송을 시키고 각 재판 서버는 인증 시 서비스 디바이스가 발행한 서비스 캡슐을 위임 서버로부터 저장을 시킴으로서 검증 청구시 사용자의 인증 연산이 위임된 환경에서도 부인방지 기능이 제공되어 위임을 시키지 않은 PKI와 동일한 보안성과 기능성을 제공할 수 있도록 설계 되었다.

5.2 연산량 및 인증 지연시간 측면

본 논문에서 제시한 프로토콜은 사용자가 보안 인프라에 처음 접근 시에 위임서버에게 위임을 시키게 되면 그 이후에는 사용자의 인증 요청에 따라 인증이 수행하게 된다. <그림3>에서 제시된 Single Sign-On 프로토콜 흐름도에 따라 위임 서버와 사용자의 위임 여부에 따라 다음과 같이 인증 흐름이

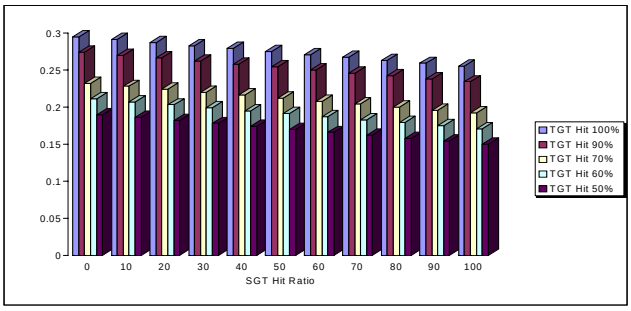
바뀌게 된다.

· 사용자가 처음 보안 인프라에 접근하여 인증을 받을 경우
  State1 → State2 → State3 → State4 → State5 → State6
· 위임 후 사용자가 인증을 받을 경우
  State1 → State3 → State4 → State5 → State6
· 위임 서버가 사용자가 사용하려는 서비스에 대한 TGT를 이미 갖고 있는 경우
  State1 → State3 → State5 → State6
· 위임 서버가 사용자가 사용하려는 서비스에 대한 TGT와 SGT를 이미 갖고 있는 경우
  State1 → State3 → State6



<그래프 1> 일반적인 PKI 인증과 위임연산, 제안된 SSO 프로토콜 인증 시간 비교

사용자가 위임 시간동안 25번의 인증을 수행을 한다고 가정하면 평균 인증 지연시간은 <그래프1>과 같다. <그래프1>에서는 제시된 환경에서 동일한 Device를 이용하여 인증을 받을 시에 한번 인증을 받는데 소요되는 시간이다. PKI의 RSA 1024bit를 사용자의 보안 단말기에서 구동이 될 경우 평균 8.125 sec의 시간이 소요 된다[23]. 또한, 사용자가 위임을 하는데 소요되는 시간은 평균 8.42 sec 일반적인 PKI 인증보다 오랜 시간이 소요되나 위임 연산은 사용자가 한번만 수행을 하게 되면 그 이후의 모든 서비스에 대한 인증은 줄어들게 되어 평균 인증 시간은 0.343 sec로 줄어들게 된다. <그래프2>는 사용자가 이미 자신의 인증 연산을 위임 서버에게 위임을 하였을 경우 위임 서버의 적중률에 따른 인증 지연시간을 보여준다.



<그래프 2> 위임서버의 TGT, SGT 적중률에 따른 인증 지연시간

사용자가 접속하는 TGT와 SGT에 대한 적중률에 따라 최장 0.402sec에서 최단 0.150sec까지 감소되는 것을 볼 수 있다.[23] 이에 관련된 연구는 로 사용자 패턴에 따른 알고리즘을 적용시켜 이를 구현할 수 있게 된다.

5.3 키관리 측면

유비쿼터스 환경에 PKI를 적용하여 사용자와 서비스 디바이스가 개인키와 공개키를 생성하여 관리해야 할 경우에는 키관리 측면과 구축 비용적인 측면에서 매우 비효율 적이다. <수식1>은 기존의 일반적인 PKI 방식에서 생성되고 관리되어야 할 공개키와 개인키 쌍의 개수와 제안된 방식에서 생성되고 관리되어야 할 쌍의 개수를 나타낸다. 좌항(M+N)은 기존의 방법에서 필요로 하는 키 개수를 의미한다. 이는 서비스 디바이스 수와 사용자의 수의 합으로 나타낼 수 있다. 제안된 방법에서 서비스 디바이스는 각각 개인키와 공개키를 가지고 있는 것이 아니라 섹션으로 통합되고 각 섹션에서는 커버로스 Server가 대표하여 하나의 키를 생성하게 된다. 그러므로 전체 서비스 디바이스에서 필요로 하는 Key의 개수는 M에서 $\frac{M}{\alpha}$로 줄어들게 된다. 또한 사용자의 Key의 개수는 기존의 N개에서 $2N+\frac{N}{\beta}$으로 늘어나게 된다. 이는 각 위임 서버에서 개인키와 공개키가 생성되어야 하며, 각 사용자의 위임되는 Key가 위임 서버에서 생성이 되기 때문이다. 하지만 일시적으로 생성된 Key는 사용자에 의해 서명이 될 뿐 CA에 의해 관리가 되지 않으므로 Key관리 측면에 있어서 시스템에 큰 영향을 주지 않는다. 그러므로 실제 생성되고 관리되어야 할 Key는 $N+\frac{N}{\beta}$가 된다. 그러므로 α와 β의 수가 비슷하고 $M(\alpha-1)>N$ 을 만족하면 $M-\frac{M+N}{\alpha}$ 만큼 관리해야 할 키가 줄어드는 것을 확인 할 수 있다.

$$M+N>\frac{M}{\alpha}+N+\frac{N}{\beta}$$
$$\text{if } \alpha \fallingdotseq \beta \text{ then } M(\alpha-1)>N$$

α = Section별 존재하는 평균 서비스 디바이스 개수
β = 위임 서버에서 Cover될 수 있는 사용자 수
M = 서비스 디바이스 수
N = User 수

<수식 1> : 기존의 방법과 제안된 방법의 키 개수 비교

6. 결 론

본 논문에서는 유비쿼터스 환경에서의 초소형 보안단말기를 위한 PKI 기반 Single Sign-On 프로토콜을 제안하고 분석하였다. 본 논문에서 제시한 프로토콜 및 인프라는 다음과 같은 특성을 갖는다. 1)연산능력이 좋지 않은 초소형 보안 단말기에서 강력한 보안성을 제공하기 위하여 PKI를 도입하였으며, 인증시 필요한 RSA연산의 오버헤드를 줄이기 위하여 대행 인증서를 통한 위임 메커니즘을 도입하였다. 2)환경에 편재되어 있는 각 서비스 디바이스와 사용자간의 인증을 위하여 대칭키 기반 프로토콜인 커버로스를 도입하여 환경을 커버로스 섹션으로 나누어 보안영역을 분할 할 수 있도록 설계하였고 이를 통해 각 서비스 디바이스는 PKI를 수행할 연산능력이 없더라도 커버로스가 대표하여 PKI를 통하

여 위임서버와 인증을 맺어 사용자와 서비스 디바이스 간 안전한 상호 인증을 제공할 수 있게 되었다. 3)기존의 위임 메커니즘에서 사용자가 다른 개체에게 위임을 시킨 후에는 모든 인증에 대하여 실제 사용자에 대한 부인방지 기능을 제공할 수 없는 문제가 있으나 심판 서버를 두어 인증에 필요한 증거자료를 서비스 디바이스와 사용자를 통해 받아 저장시킴으로서 사용자의 부인을 방지 할 수 있는 메커니즘을 제공하였다. 4)시스템 측면에서도 병목현상을 제거시키기 위하여 커버로스를 섹션별로 나누고 하나의 커버로스 서버가 PKI의 개체가 되어 인증이 될 수 있도록 설계하도록 함으로서 시스템의 병목현상을 해결하고 확장성과 상호 연산성을 갖춘 보안 인프라를 설계하였고 키 관리 측면에서도 기존의 커버로스나 PKI형태의 키 관리 구조에서 PKI와 커버로스를 계층적으로 배치하여 효율적인 키 관리가 가능해 졌으며 각 서비스 디바이스와 사용자는 성능이 좋지 않더라도 PKI와 동일한 보안성과 안전성을 제공해 줌으로서 인프라 구축비용 측면에서도 장점을 보인다.

본 논문은 사용자와 서비스 디바이스들이 무수히 많은 유비쿼터스 환경에 PKI와 위임 메커니즘, 섹션별 커버로스를 이용한 효율적이고 강력한 인증 메커니즘을 설계하였으며 앞으로 실제 설계된 인프라를 U-Campus환경에 적용시켜 활용할 수 있을 것이다.

참고문헌

[1] Kyu Ho Park, UFC Project Group* "UFC: A Ubiquitous Fashionable Computer", Next Generation PC 2005,October 2005.

[2] M. Satyanarayanan, Carnegie Mellon University and Intel Research Pittsburgh "A Catalyst for Mobile and Ubiquitous Computing" IEEE Pervasive Computing 2002, Feb 2002

[3] http://core.kaist.ac.kr/UFC.html KAIST Computer Engineering Research Laboratory, IT-839 Project Home Page.

[4] R. Morales-Salcedo, H. Ogata, and Y. Yano “Using RFID and Dynamic Metadata in an Educational Digital Library”, IASTED International Conference on WEB-BASED EDUCATION, pp.323Security and privacy rights management for mobile and ubiquitous computing-331

[5] Michael Fahrmair, Wassiou Sitou, and Bernd Spanfelner "Security and privacy rights manaement for mobile and ubiquitous computing", Ubicomp 2005

[6] Gregory D. Abowd and Elizabeth D. Mynatt. "Charting past, present and future research in ubiquitous computing. ACM Transactions on Computer-Human Interaction", Special issue on HCI in the new Millenium, 7 (1):29−58, March 2000.

[7] Stephen S. Intille "Change Blind Information Display for Ubiquitous Computing Environments", Ubicomp 2000

[8] IAN F. AKYILDIZ AND SHANTIDEV MOHANTY, "A Ubiquitous Mobile Communication Architecture for Next-Generation Heterogeneous Wireless Systems" IEEE Radio Communications 2005

[9] Cisco Systems "PKI AAA Authorization Using the Entire Subject Name" Cisco Systems, Technical Documentation

[10] Wei Zhouand Christoph Meinel "Implement role based access control with attribute certificates" Proceedings of the 6th International Conference on Advanced Communication Technology (ICACT2004)

[11] Mike Fraser "Mobile and Ubiquitous Computing" COMSM0106 − Mobile and Ubiquitous Computing

[12] P. Horster, M, Michels, "Hidden signature schemes based on the descrete logarithm problem and related concepts", Proc. of Communications and Multimedia Security 1995 Chapman & Hall 1995

[13] D. Chaum and H. van Antwerpen, "Undeniable signatures", Advances in Cryptology-CRYPTO 1989 Proceedings, Springer - Verilag 1990.

[14] Shashi Kiran,Patricia Lareau,Steve Lloyd "PKI Basics - A Technical Perspective" PKI Forum 2002

[15] Von Welch "X.509 Proxy Certificates for Dynamic Delegation" PKI Forum 2004

[16] S. Tuecke, D. Engert "Internet X.509 Public Key Infrastructure Proxy Certificate Profile" Global Grid Forum 2002

[17] Ki-Woong Park, Sang-Seok Lim, Kyu-Ho Park "PANDA: An Interoperable Mobile Security Card for biquitous Services", KAIST CORE Lab. Technical Report,2006.

[18] Ki-Woong Park, H.-J. Choi, and K. H. Park, “An interoperable authentication system using zigbee-enabled tiny portable device and pki,” in Internation Conference on Next Generation PC, October 2005.

[19] Z. A. B. of Directors, ZigBee Specification v1.0. ZigBee Alliance,2005.

[20] J. Linn "The Kerberos Version 5 GSS-API Mechanism" RFC1964, 1996

[21] S. Tuecke, V. Welch "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile" RFC3820, 2004

[22] Eun-Hwan Kim, Moon-Seog Jun "Design of a Strong Authentication Mechanism using Public-Key based on Kerberos" **정보과학학회지** 2002.

[23] Ki-Woong Park,“Tiny-Terminal Development Technical Report" KAIST CORE Lab.

[24] Woo-Min Hwang, S. Lim, K. Park "A Context-Aware Replacement Page Cache for Wearable Computer" Next Generation PC, 2006