情報保護學會論文誌
第13券 第4號, 2003. 8

상이한 도메인에서 검증 가능한 자체 인증 공개키

정 영 석*, 한 종 수*, 오 수 현*, 원 동 호**

Inter-Domain Verifiable Self-certified public keys

Young-Seok Chung*, Jong-Su Han*, Sooh-Hyun Oh*, Dongho Won**

요 약

자체 인증 공개키는 공개키 생성시 사용자와 신뢰기관이 함께 참여하기 때문에 다른 사용자가 검증하는 데에 있어서 별도의 인증서를 필요로 하지 않는다. 이에 공개키를 서명 등의 어플리케이션에 사용하여 서명이 제대로 검증되지 않은 경우 서명 과정에 문제가 있는지 사용자의 공개키에 문제가 있는지 확인할 수 있는 검증 가능한 자체 인증 공개키가 제안되었으며, 이러한 공개키의 검증 과정에는 사용자가 신뢰하는 키 발급 기관(KGC : Key Generation Center)의 공개키가 사용된다. 공개키 검증시 사용자들이 동일한 키 발급 기관을 신뢰하는 경우에는 간단하게 상대방의 공개키를 검증할 수 있지만, 상이한 도메인 내의 사용자간에는 키 발급 기관간 신뢰 관계가 구축되어야만 상대방의 공개키를 검증할 수 있게 된다. 본 논문에서는 상호 신뢰 관계가 구축된 두 키 발급 기관하의 사용자간 인증서의 사용 없이 상대방 공개키의 정당성을 검증할 수 있는 자체 인증 공개키를 제안한다. 또한, 서로 다른 공개키 파라미터를 사용하는 키 발급 기관 하의 사용자간 서명 생성·검증과 키 분배를 수행하는 과정에 대하여 알아본다.

ABSTRACT

Self-certified public keys need not be accompanied with a separate certificate to be authenticated by other users because the public keys are computed by both the authority and the user. At this point, verifiable self-certified public keys are proposed that can determine which is wrong signatures or public keys if public keys are used in signature scheme and then verification of signatures does not succeed. To verify these public keys, key generation center's public key trusted by users is required. If all users trust same key generation center, public keys can be verified simply. But among users in different domains, trusty relationship between two key generation centers must be accomplished. In this paper we propose inter-domain verifiable self-certified public keys that can be verified without certificate between users under key generation centers whose trusty relationship is accomplished. Also we present the execution of signature and key distribution between users under key generation centers use different public key parameters.

keyword : *Self-certified, trusty relationship, inter-domain*

Ⅰ. 서 론

최근 빠르게 확산되고 있는 전자 상거래 및 전자 서명 서비스를 안전하게 제공하기 위해 각 국가별로 PKI(Public Key Infrastructure)[1,2]가 구축되고 있다. 이러한 PKI 환경에서는 많은 수의 사용자들을 수용하고 효율적인 인증서 관련 서비스를 제공하기 위해 여러 개의 인증기관을 다양한 형태로 두고 있다. 이

* 성균관대학교 정보통신공학부 정보통신보호연구실({yschung, jshan, shoh}@dosan.skku.ac.kr)
** 성균관대학교 정보통신공학부 정교수(dhwon@dosan.skku.ac.kr)

처럼 각 나라마다 여러 인증기관이 존재하기 때문에 세계적으로 상당히 많은 수의 공인 및 사설 인증기관이 존재한다.

따라서 사용자들이 다른 인증기관하의 사용자들과 전자 상거래 등을 수행하기 위해서는 상대방이 신뢰하고 있는 인증기관이 발행한 인증서가 필요하다. 그러나 각 사용자들은 자신이 속한 도메인의 인증기관만을 신뢰하기 때문에 다른 도메인의 사용자 인증서를 검증하기 위해서는 두 사용자의 인증기관간 상호연동[3,4]이 이루어져야 하고, 이를 통한 인증 경로 검색 및 발견 서비스를 이용하여 상대방의 인증서를 검증하게 된다.

한편, 사용자 공개키의 정당성을 검증하는 방식에는 위와 같은 PKI 환경에서 제공되는 인증서를 사용하는 방식과 인증서 없이 사용자의 개인식별정보를 이용하는 방식이 있다. Shamir[5]가 제안한 개인식별정보를 그대로 공개키로 사용하는 방식에서는 누구에게나 알려져 있는 개인식별정보를 공개키로 사용하기 때문에 별도의 공개키 검증 과정이 필요 없다. 하지만 사용자의 개인키가 키 발급 기관에 노출되는 단점이 있다. Girault[6]가 제안한 자체 인증 공개키 방식에서는 공개키 내에 개인식별정보가 포함되어 있고, 그것의 생성과정에 사용자와 키 발급 기관이 함께 참여하기 때문에 공개키가 인증서의 역할을 하게 된다. 따라서 공개키 자체만으로 인증이 가능하고, 발급하고 검증해야 할 인증서가 필요하지 않으며, 인증서 기반 방식보다 계산량이나 저장공간 측면에서 부담이 적다. 또한 Shamir 방식과 달리 사용자의 개인키가 키 발급 기관에 노출되지 않는다. 그러나 자체 인증 공개키가 키 분배나 서명, 암·복호화 과정에 사용되기 전까지는 공개키의 정당성을 명시적으로 확인할 수 없는 단점이 있다. 또한, 실제 어플리케이션에 사용되어 문제가 발생한 경우 즉, 두 사용자간 동일한 세션키가 생성되지 않거나 검증자가 서명의 검증 또는 복호화를 정상적으로 수행하지 못한 경우 자체 인증 공개키에 이상이 있는지 키 분배 또는 서명 등의 과정에서 오류가 발생한 것인지 알 수 없게 된다. 이러한 단점을 보완하여 검증자가 공개키 자체만으로도 정당성을 확인할 수 있는 검증 가능한 자체 인증 공개키[7]의 개념이 등장하였다.

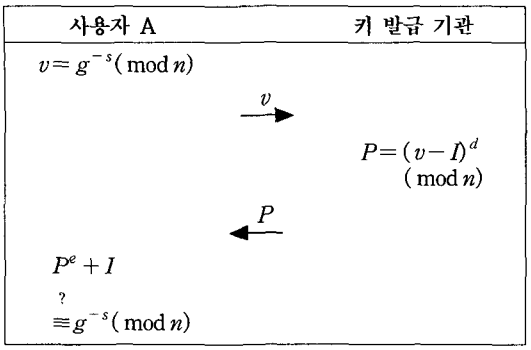
본 논문에서는 인증서의 검증이 필요 없는 자체 인증 공개키의 활용에 있어서, 상호 연동이 체결된 PKI 환경에서와 같이 서로 다른 도메인에 속한 사용자간에도 상대방 공개키의 정당성을 검증할 수 있는 자체 인증 공개키를 제안한다. 제안하는 상이한 도메인에서 검증 가능한 자체 인증 공개키는 상호 신뢰관계가 구축된 키 발급 기관하의 사용자간 사용할 수 있으며, 자신이 신뢰하는 키 발급 기관을 통해 상대방 키 발급 기관하의 사용자 공개키를 검증하게 된다. 또한, 제안하는 공개키를 실제 어플리케이션에 사용하기 위해서는, 사용자의 자체 인증 공개키를 생성할 때 사용되는 공개키 파라미터가 키 발급 기관마다 다르다는 점을 고려하여 수학적 계산이 올바르게 이루어지도록 해야한다. 이에 대해 본 논문에서는 서로 다른 키 발급 기관을 신뢰하는 사용자들이 정상적으로 서명의 생성 및 검증과 키 분배를 수행할 수 있는 방안에 대해 언급한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 개념의 바탕이 되는 자체 인증 공개키의 개념에 대해 소개하고, 3장에서 상이한 도메인의 사용자간에도 검증할 수 있는 자체 인증 공개키 및 어플리케이션으로의 적용 방안을 제안한다. 그리고 4장에서 제안한 공개키의 특징을 살펴보고 안전성을 분석하며, 마지막으로 5장에서 결론을 맺는다.

II. 연구 배경

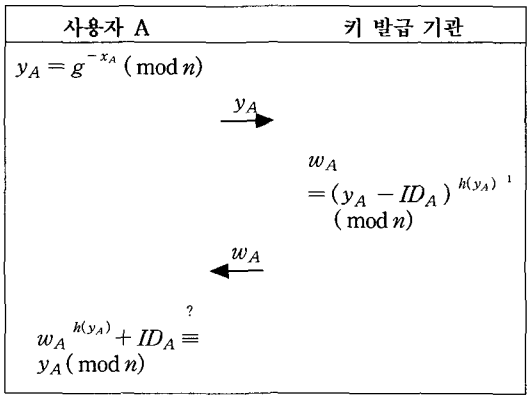
1991년 Girault[6]에 의해 처음 자체 인증 공개키의 개념이 소개된 이후 많은 연구가 이루어졌다. 그 결과 자체 인증 공개키의 사용 없이 정당성 검증이 가능한 자체 인증 공개키가 제안되었으며, 이를 기반으로 자체 인증이라는 개념이 확장되어 자체 인증 서명,[8] 자체 인증서[9] 등이 소개되었다.

자체 인증 공개키는 사용자가 직접 개인키를 선택하고 그로부터 계산된 공개키를 키 발급 기관에 전송하여 키 발급 기관으로부터 공개키에 대해 서명을 받는 방식으로 생성된다. 이 때, 키 발급 기관은 공개키와 함께 사용자의 개인식별정보에 서명을 하게 된다. 이와 같은 형태로 자체 인증 공개키가 생성되기 때문에 이를 검증하기 위해서는 사용자의 개인식별정보와 키 발급 기관의 공개키가 필요하게 된다. 따라서 키 발급 기관의 공개키로 검증된 사용자의 공개키는 그 사용자의 개인식별정보와 연관되어 자체 인증성을 갖게 된다. Girault는 이와 같은 과정을 소인수 분해 문제 기반의 RSA/Rabin 서명을 이용한 방식과 이산 대수 문제 기반의 ElGamal 서명을 이용한 방식으로 제안하였다. [그림 1]은 RSA/Rabin 서명을 이용한 방식을 나타낸 것이다.



- $n = p \cdot q$ (p, q는 큰 소수)
- d : 키 발급 기관의 개인키
- e : 키 발급 기관의 공개키
  $e \cdot d = 1 \bmod (p-1)(q-1)$
- g : $(Z/nZ)^*$상에서 위수가 최대인 정수
- s : 사용자 A의 개인키
- v : 사용자 A의 공개키
- P : 사용자 A의 자체 인증 공개키
- I : 사용자 A의 개인식별정보

〔그림 1〕 RSA/Rabin 서명을 이용한 자체 인증 공개키



- $n = p \cdot q$ (p, q는 큰 소수)
- g : $p = 2p' + 1$, $q = 2q' + 1$
  일 때, 위수가 $r = p'q'$인 원소
  $(g^r = 1 (\mathrm{mod}\, n))$
- x_A: 사용자 A의 개인키
- y_A: 사용자 A의 공개키
- w_A: 사용자 A의 검증 가능한 자체 인증 공개키
- ID_A: 사용자 A의 개인식별정보

〔그림 2〕 RSA 서명을 이용한 검증 가능한 자체 인증 공개키

반면, 검증 가능한 자체 인증 공개키[7]에서는 키 발급 기관이 사용자에게 공개키를 생성해 줄 때, 일방향 해쉬 함수의 지수승 연산을 사용함으로써 사용자가 이를 확인할 수 있도록 하였다. 이는 [그림 2]와 같은 소인수 분해 문제 기반의 RSA 서명을 이용한 방식과 이산 대수 문제 기반의 Schnorr 서명을 이용한 방식으로 제안되었다.

Ⅲ. 상이한 도메인에서 검증 가능한 자체 인증 공개키

3.1 개요

제안하는 상이한 도메인에서 검증 가능한 자체 인증 공개키(이하 자체 인증 공개키)는 검증할 때 있어서 사용자의 개인식별정보를 사용하기 때문에 자체 인증성을 갖는다. 또한, 키 발급 기관에서 사용자의 자체 인증 공개키 생성시 사용자의 공개키 및 키 발급 기관의 공개키, 공개키 파라미터와 각각의 개인식별정보에 대한 해쉬값을 사용한다. 따라서 검증자 입장에서는 자체 인증 공개키의 정당성 및 발급 주체의 신원을 명시적으로 확인할 수 있다.

한편, 키 발급 기관간에는 신뢰 관계가 구축되어 있어야 하며, 이를 통해 상대방 키 발급 기관의 공개키 또는 공개키 파라미터에 대한 사용자의 요청이 있을 때 키 발급 기관은 이의 해쉬값에 서명하여 전송해 주는 형식으로 사용자에게 타 키 발급 기관의 공개키 또는 공개키 파라미터를 인증해 준다. 따라서 사용자는 자신이 신뢰하는 키 발급 기관의 서명의 정당성을 검증함으로써 다른 도메인의 키 발급 기관을 신뢰하게 되고, 이렇게 하여 다른 도메인 사용자의 자체 인증 공개키를 검증할 수 있게 된다.

제안하는 방식은 키 발급 기관으로부터 자체 인증 공개키를 발급 받는 과정과 이를 서명 및 키 분배 등의 어플리케이션에 사용하는 과정, 사용 후 문제가 발생한 경우 자체 인증 공개키에 이상이 있는지, 서명이나 키 분배 과정에서 오류가 발생하였는지 다른 도메인의 사용자도 검증할 수 있는 과정으로 구성되며, 소인수 분해 및 이산 대수 문제를 기반으로 하고 있다.

3.2 소인수 분해 문제 기반의 자체 인증 공개키

3.2.1 정의 및 요구사항

- A, B
  - 각 사용자

- $KGC_{(A/B)}$
  - 각각 A와 B가 신뢰하는 키 발급 기관
  - 두 키 발급 기관은 서로 다른 인증 도메인을 형성하고 있으며, 상호 신뢰 관계가 구축되어 있어 상대방 기관의 공개키 및 공개키 파라미터를 저장하고 있음
- ID_A, ID_{KGC_A}
  - 각각 A와 KGC_A의 개인식별정보
- $p_{(A/B)}$
  - 사용자가 개인키, 공개키 생성 및 서명 등의 어플리케이션에 사용하도록 $KGC_{(A/B)}$에서 공개한 소수
  - 그룹의 위수가 작은 소수의 곱으로만 이루어져 있을 때 각각의 서브 그룹 상에서 이산 대수를 구한 후, 중국인의 나머지 정리를 이용하여 전체 그룹에서의 이산 대수를 계산하는 방법인 Pohlig-Hellman 알고리즘[10]에 의한 이산 대수 계산을 어렵게 하기 위해 $p_{(A/B)}$는 상당히 큰 소수이어야 하며, $p_{(A/B)} = 2p_{(A/B)}' + 1$ 일 때, p' 역시 큰 소수이어야 함
  ($e.g.$ $|p'| \geq 2^{160}$, $|p| = 2^{1024}$)
- $n_{(A/B)}$
  - 각각 KGC_A와 KGC_B의 공개키 파라미터
  - $n_{(A/B)} = q_{(A/B)} \cdot r_{(A/B)}$
  이 때, $q_{(A/B)}, r_{(A/B)}$은 Pohlig-Hellman 알고리즘에 의한 이산 대수 계산을 어렵게 하기 위해 각각 비슷한 크기의 상당히 큰 소수이어야 하며 $q_{(A/B)} = 2q_{(A/B)}' + 1$, $r_{(A/B)} = 2r_{(A/B)}' + 1$ 일 때, q', r' 역시 큰 소수이어야 함
  ($e.g.$ $|q'|, |r'| \geq 2^{160}$, $|q|, |r| \geq 2^{1024}$)
- 정수 3
  - KGC_A, KGC_B 등 키 발급 기관의 공개키
  - 공개키를 이용한 지수 연산의 효율성을 높이기 위해 사용되는 두 가지 형태($100\cdots001_{(2)}$, $\cdots000011_{(2)}$) 중 본 방식에서는 $\cdots000011_{(2)} = 3_{(10)}$을 공개키로 사용함
- $d_{(A/B)}$
  - 각각 KGC_A와 KGC_B의 개인키
  - $3 \cdot d_{(A/B)} = 1 \pmod{\phi(n_{(A/B)})}$
  이 때, $\phi(n_{(A/B)}) = (q_{(A/B)} - 1)(r_{(A/B)} - 1)$
- $g_{(A/B)}$
  - $Z_{p_{(A/B)}}^*$ 상에서의 원시 원소
- $h()$
  - $\{0,1\}^* \rightarrow Z_{2^t} = \{0, \cdots, 2^t - 1\}$ ($e.g.$ $t = 128$) 인 해쉬 함수
  - 메시지를 M, 해쉬값을 H라 할 때, $h()$는 다음 3가지 특성을 만족해야 함
    · 일방향성(약)
    : H로부터 $h(M) = H$가 되는 M을 찾는 것은 계산상 불가능
    · 일방향성(강)
    : M과 $H = h(M)$이 주어졌을 때, $h(M') = H$이 되는 $M \neq M'$을 찾는 것은 계산상 불가능
    · 충돌 회피성
    : $h(M) = h(M')$이 되는 $M \neq M'$를 찾는 것은 계산상 불가능

3.2.2 자체 인증 공개키 발급 과정

제안하는 소인수 분해 문제 기반 자체 인증 공개키의 발급은 오프라인 상에서 다음과 같은 과정으로 이루어진다.

① 사용자 A로부터 자체 인증 공개키 발급 신청을 받은 키 발급 기관 KGC_A는 직접 대면 등의 방법으로 사용자 A의 신원을 확인한다. KGC_A는 A에게 발급한 자체 인증 공개키와 ID_A의 목록을 유지함으로써, 후에 사용자 A 또는 제 3자가 ID_A로 또 다른 자체 인증 공개키를 발급 받는 것을 방지한다.

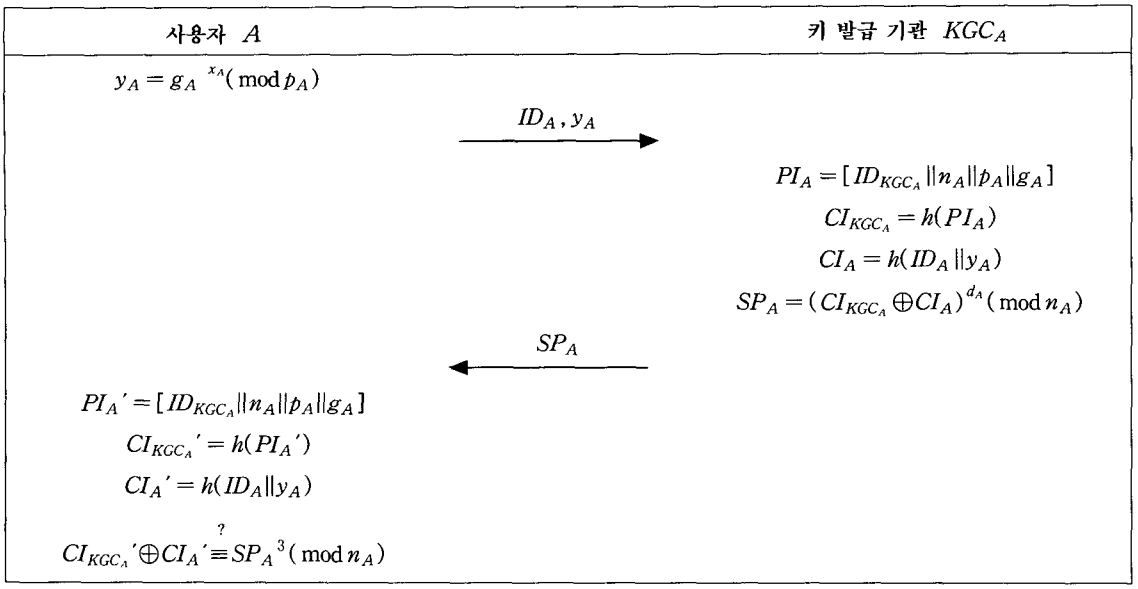
② 사용자 A는 자신의 개인키 $x_A \in Z_{\frac{p_{(A/B)}-1}{2}}^*$를 선택하고, 자신이 신뢰하는 키 발급 기관 KGC_A의 공개키 파라미터 p_A, g_A를 이용하여 공개키 y_A를 계산하고, ID_A와 함께 KGC_A에게 전송 한다.

$$y_A = g_A^{\,x_A} \pmod{p_A}$$

③ 키 발급 기관 KGC_A는 자신의 개인식별정보 ID_{KGC_A}와 사용자들에게 공개하고 있는 공개키 파라미터 n_A, p_A, g_A를 연접하여 공개 정보 PI_A를 생성한다.

$$PI_A = [ID_{KGC_A} \| n_A \| p_A \| g_A]$$

④ PI_A와 y_A에 대한 위조를 막고 무결성을 제공하기 위해 키 발급 기관 KGC_A는 일방향 해쉬 함



〔그림 3〕 소인수 분해 문제 기반 자체 인증 공개키의 발급 과정

수 $h()$를 이용하여 자신의 인증 정보 CI_{KGC_A}와 A의 인증 정보 CI_A를 생성한다.

$$CI_{KGC_A} = h(PI_A)$$
$$CI_A = h(ID_A \| y_A)$$

⑤ 키 발급 기관 KGC_A는 자신의 비밀키 d_A를 이용해 CI_{KGC_A}, CI_A에 서명을 하여 자체 인증 공개키 SP_A를 생성하고, 이를 A에게 전송한다.

$$SP_A = (CI_{KGC_A} \oplus CI_A)^{d_A} (\mathrm{mod}\, n_A)$$

⑥ 사용자 A는 자신의 공개키를 계산할 때 사용했던 KGC_A의 공개키 파라미터와 ID_{KGC_A}, n_A로 PI_A'를 생성하고, 일방향 해쉬 함수 $h()$를 이용하여 CI_{KGC_A}' CI_A'를 계산한다. 그리고 SP_A가 y_A에 대한 자체 인증 공개키임을 확인한다.

$$CI_{KGC_A}' = h(PI_A')$$
$$CI_A' = h(ID_A \| y_A)$$
$$CI_{KGC_A}' \oplus CI_A' \stackrel{?}{\equiv} SP_A{}^3 (\mathrm{mod}\, n_A)$$

[그림 3]은 소인수 분해 문제 기반 자체 인증 공개키 발급 과정을 그림으로 나타낸 것이다.

3.2.3 서명 생성 및 검증 과정

자체 인증 공개키를 실제 어플리케이션에 사용하는 예로, 사용자 A가 ElGamal 서명 방식을 이용하여 생성한 서명을 사용자 B가 검증하는 과정을 보면 다음과 같다.

① 사용자 A는 $Z_{\frac{p_A-1}{2}}{}^*$상에서 임의로 선택한 원소 k를 이용하여 중간값 R을 계산한다.

$$R = g_A{}^k (\mathrm{mod}\, p_A)$$

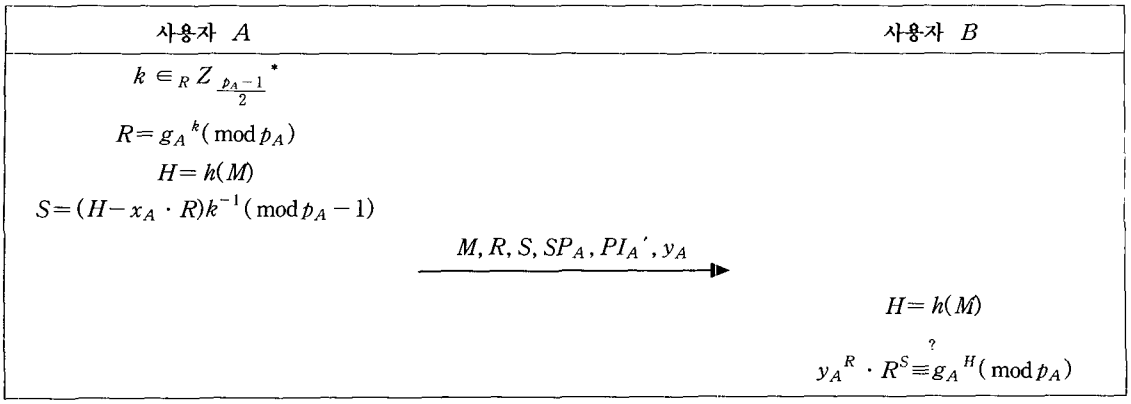
② 사용자 A는 임의의 서명문 M에 대한 해쉬 값 $H = h(M)$에 자신의 개인키 x_A를 이용하여 서명 S를 생성하고, M, R 그리고 SP_A, PI_A', y_A를 사용자 B에게 전송한다.

$$S = (H - x_A \cdot R) k^{-1} (\mathrm{mod}\, p_A - 1)$$

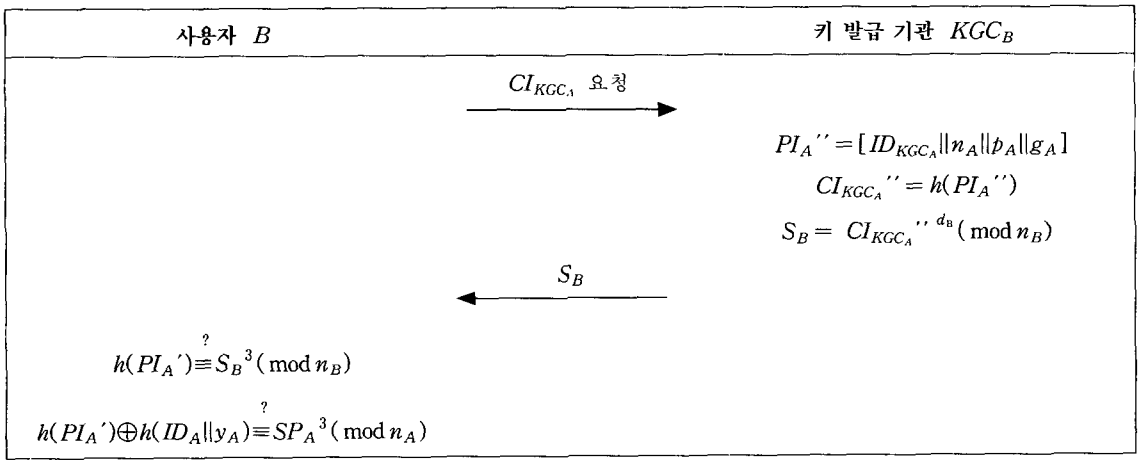
③ 사용자 B는 전송 받은 정보로 서명을 검증한다.

$$y_A{}^R \cdot R^S \stackrel{?}{\equiv} g_A{}^H (\mathrm{mod}\, p_A)$$

[그림 4]는 서로 다른 도메인의 사용자간 서명을 생성하고 검증하는 과정을 나타낸 것이다.



〔그림 4〕 서명 생성 및 검증 과정



〔그림 5〕 소인수 분해 문제 기반 자체 인증 공개키 검증 과정

3.2.4 소인수 분해 문제 기반 자체 인증 공개키의 검증 과정

3.2.3절에서의 서명 검증식이 성립하지 않는 경우, 서명값이 변경·위조되었는지 공개키에 이상이 있는지 알아보기 위해, 다음과 같은 검증 과정을 수행한다.

① 사용자 B는 사용자 A의 서명을 검증하기 위해 키 발급 기관 KGC_B에게 KGC_A의 인증 정보 CI_{KGC_A}를 요청한다.

② 키 발급 기관 KGC_B는 정당성을 검증하여 보관 중인 KGC_A의 공개키 파라미터로 공개 정보 PI_A''를 생성하고, KGC_A의 인증 정보 CI_{KGC_A}''를 생성한다.

$$PI_A'' = [ID_{KGC_A} \| n_A \| p_A \| g_A]$$
$$CI_{KGC_A}'' = h(PI_A'')$$

③ 키 발급 기관 KGC_B는 자신의 개인키로 CI_{KGC_A}''에 서명을 하여 사용자 B에게 전송한다.

$$S_B = CI_{KGC_A}''{}^{d_B} \pmod{n_B}$$

④ 사용자 B는 KGC_B의 공개키로 S_B를 검증한 값과 사용자 A로부터 전송 받은 PI_A'의 해쉬 값이 같은지 비교함으로써 KGC_A의 공개키 파라미터를 신뢰하게 된다.

$$h(PI_A') \stackrel{?}{\equiv} S_B{}^3 \pmod{n_B}$$

⑤ 사용자 B는 KGC_A의 공개키 파라미터와 사용자 A가 전송한 SP_A를 이용하여 A의 공개키 y_A를 검증한다. 검증식이 성립하는 경우 서명값에 이상이 생겼음을, 식이 성립하지 않는 경우 A의 공개키에 이상이 생겼음을 알 수 있다.

$$h(PI_A')\oplus h(ID_A\|y_A)\stackrel{?}{\equiv} SP_A{}^3 (\bmod n_A)$$

[그림 5]는 소인수 분해 문제 기반 자체 인증 공개키의 검증 과정을 나타낸 것이다.

3.3 이산 대수 문제 기반의 자체 인증 공개키

3.3.1 정의 및 요구사항

- $p_{(A/B)}$, $q_{(A/B)}$
  - $q_{(A/B)}|p_{(A/B)}-1$을 만족하는 큰 소수
- $g_{(A/B)}$
  - 위수가 $q_{(A/B)}$인 $Z_{p_{(A/B)}}{}^*$의 서브 그룹에서의 원시원소
- $x_{KGC_{(A/B)}}$, $y_{KGC_{(A/B)}}$
  - $KGC_{(A/B)}$의 개인키 및 공개키
- $x_{KGC_A} \in Z_{q_A}{}^*$
- $y_{KGC_{(A/B)}} = g_{(A/B)}{}^{x_{KGC_{(A/B)}}} (\bmod p_{(A/B)})$

※ A, B, KGC_A, KGC_B, ID_A, ID_{KGC_A}, $h()$는 3. 2.1과 동일

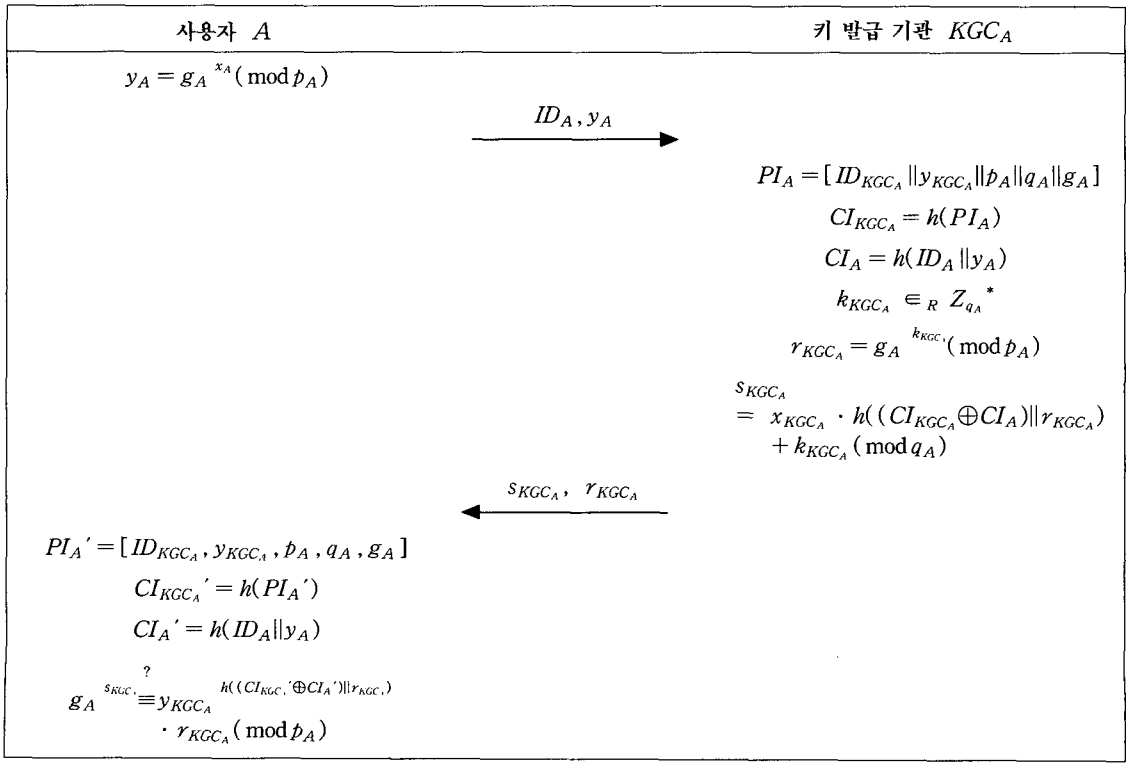
3.3.2 자체 인증 공개키 발급 과정

제안하는 이산 대수 문제 기반 자체 인증 공개키의 발급은 오프라인 상에서 다음과 같은 과정으로 이루어진다.

※ 3.2.2에서와 동일하거나 비슷한 과정은 생략하거나 간략하게 설명

① 키 발급 기관 KGC_A는 직접 대면 등의 방법으로 사용자 A의 신원을 확인한 후, 발급한 자체 인증 공개키와 A의 개인식별정보 목록을 유지한다.

② 사용자 A는 자신의 개인키 $x_A \in Z_{q_A}{}^*$를 선택하고, 자신이 신뢰하는 키 발급 기관 KGC_A의 공개키 파라미터 p_A, g_A를 이용하여 공개키 y_A



〔그림 6〕 이산 대수 문제 기반 자체 인증 공개키의 발급 과정

를 계산하여 ID_A와 함께 KGC_A에게 전송한다.

$$y_A = g_A{}^{x_A} (\mathrm{mod}\, p_A)$$

③ 키 발급 기관 KGC_A는 공개 정보 PI_A를 생생성하여 자신의 인증 정보 CI_{KGC_A}와 사용자의 인증 정보 CI_A를 생성한다. 그리고 $Z_{q_A}{}^*$ 상에서 임의의 원소 k_{KGC_A}를 선택하여 r_{KGC_A}를 계산한다.

$$PI_A = [ID_{KGC_A} \| y_{KGC_A} \| p_A \| q_A \| g_A]$$
$$CI_{KGC_A} = h(PI_A)$$
$$CI_A = h(ID_A \| y_A)$$
$$r_{KGC_A} = g_A{}^{k_{KGC_A}} (\mathrm{mod}\, p_A)$$

④ 키 발급 기관 KGC_A는 자신의 개인키 x_{KGC_A}와 k_{KGC_A}를 이용하여 s_{KGC_A}를 생성한다. 그리고 s_{KGC_A}, r_{KGC_A}를 사용자 A에게 전송한다.

$$s_{KGC_A} = x_{KGC_A} \cdot h((CI_{KGC_A} \oplus CI_A) \| r_{KGC_A}) + k_{KGC_A} (\mathrm{mod}\, q_A)$$

⑤ 사용자 A는 자신의 공개키를 계산할 때 사용했던 KGC_A의 공개키 파라미터로 PI_A'를 생성하고, 일방향 해쉬 함수 $h()$를 이용하여 CI_{KGC_A}' CI_A'를 계산한다.

$$CI_{KGC_A}' = h(PI_A')$$
$$CI_A' = h(ID_A \| y_A)$$

⑥ 사용자 A는 자신의 공개키 y_A가 포함된 CI_A'를 이용하여 s_{KGC_A}, r_{KGC_A}를 검증함으로써 s_{KGC_A}, r_{KGC_A}가 y_A에 대한 자체 인증 공개키임을 확인한다.

$$g_A{}^{s_{KGC_A}} \overset{?}{\equiv} y_{KGC_A}{}^{h((CI_{KGC_A}' \oplus CI_A') \| r_{KGC_A})} \cdot r_{KGC_A} (\mathrm{mod}\, p_A)$$

[그림 6]은 이산 대수 문제 기반 자체 인증 공개키의 발급 과정을 그림으로 나타낸 것이다.

3.3.3 키 분배 과정

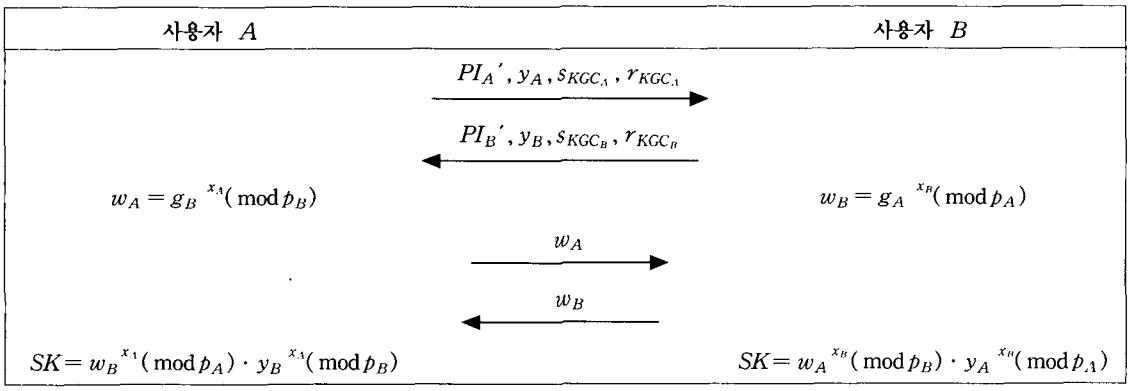
자체 인증 공개키를 실제 어플리케이션에 사용하는 예로, 사용자 A, B가 Diffie-Hellman 키 분배 프로토콜을 수행하는 과정을 보면 다음과 같으며, 상대방의 자체 인증 공개키의 검증 과정은 두 사용자가 동일하므로 B가 A의 자체 인증 공개키를 검증하는 과정만을 보도록 한다.

① 사용자 A와 B는 각각 공개 정보 $PI_{(A/B)}'$, 공개키 $y_{(A/B)}$, 자체 인증 공개키 $s_{KGC_{(A/B)}}$, $r_{KGC_{(A/B)}}$를 교환한다.

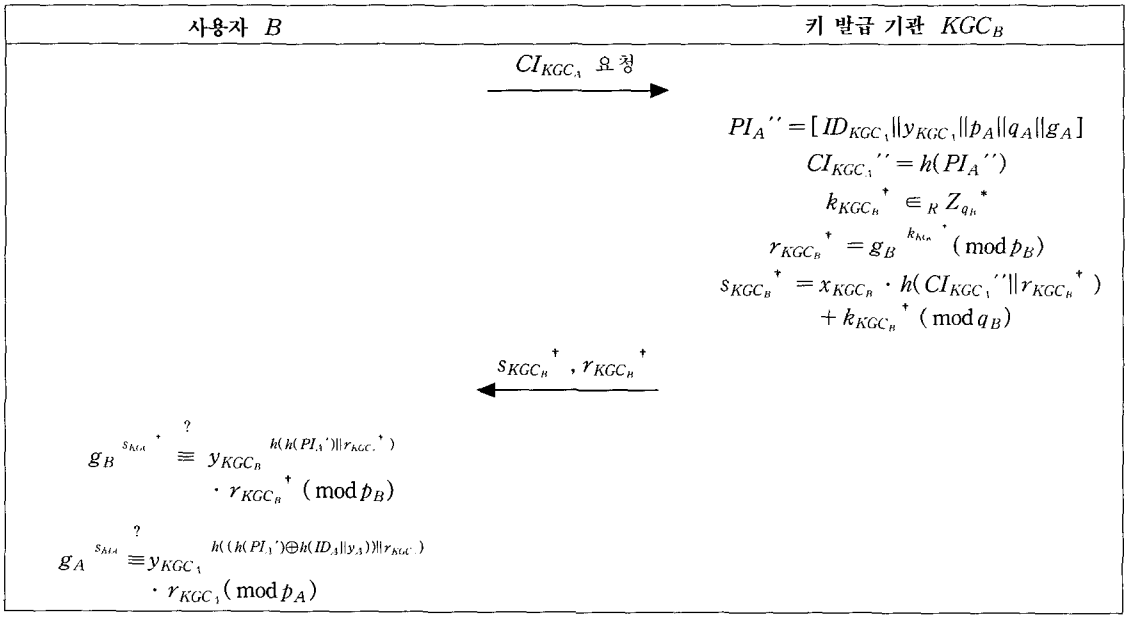
② 각 사용자는 자신의 비밀키로 세션키 중간값 $w_{(A/B)}$를 계산하여 상대방에게 전송하고, 세션키 SK를 계산한다.

$$w_{(A/B)} = g_{(B/A)}{}^{x_{(A/B)}} (\mathrm{mod}\, p_{(B/A)})$$
$$SK = w_{(B/A)}{}^{x_{(A/B)}} (\mathrm{mod}\, p_{(A/B)}) \cdot y_{(B/A)}{}^{x_{(A/B)}} (\mathrm{mod}\, p_{(B/A)})$$

[그림 7]은 상이한 도메인의 사용자간 키 분배를



〔그림 7〕 키 분배 과정



〔그림 8〕 이산 대수 문제 기반 자체 인증 공개키 검증 과정

수행하는 과정을 나타낸 것이다.

3.3.4 이산 대수 문제 기반 자체 인증 공개키의 검증 과정

3.3.3절에서 올바른 세션키가 생성되지 않은 경우, 세션키에 이상이 있는지 공개키에 이상이 있는지 알아보기 위해, 다음과 같은 검증 과정을 수행한다.

① 사용자 B로부터 KGC_A의 인증 정보 CI_{KGC_1}의 요청을 받은 KGC_B는 PI_A''로부터 CI_{KGC_1}''를 생성하고, $Z_{q_B}^*$상에서 임의의 원소 $k_{KGC_B}^{\dagger}$를 선택하고 $r_{KGC_B}^{\dagger}$, $s_{KGC_B}^{\dagger}$를 생성하여 B에게 전송한다.

$$r_{KGC_B}^{\dagger} = g_B^{k_{KGC_B}^{\dagger}} \pmod{p_B}$$
$$s_{KGC_B}^{\dagger} = x_{KGC_B} \cdot h(CI_{KGC_1}'' || r_{KGC_B}^{\dagger}) + k_{KGC_B}^{\dagger} \pmod{q_B}$$

② 사용자 B는 $r_{KGC_B}^{\dagger}$, $s_{KGC_B}^{\dagger}$를 이용하여 A로부터 전송 받은 PI_A'를 정당성을 확인한다.

$$g_B^{s_{KGC_B}^{\dagger}} \stackrel{?}{\equiv} y_{KGC_B}^{h(h(PI_A') || r_{KGC_B}^{\dagger})} \cdot r_{KGC_B}^{\dagger} \pmod{p_B}$$

③ 사용자 B는 정당성이 검증된 KGC_A의 공개키 y_{KGC_1}를 가지고 A의 자체 인증 공개키 r_{KGC_1}, s_{KGC_A}로부터 세션키 생성에 사용된 y_A의 정당성을 확인한다. 식이 성립하는 경우 세션키 생성 과정에 이상이 생겼음을, 식이 성립하지 않는 경우 A의 공개키에 이상이 생겼음을 알 수 있다.

$$g_A^{s_{KGC_A}} \stackrel{?}{\equiv} y_{KGC_1}^{h((h(PI_A') \oplus h(ID_A || y_A)) || r_{KGC_A})} \cdot r_{KGC_1} \pmod{p_A}$$

[그림 8]은 이산 대수 문제 기반 자체 인증 공개키의 검증 과정을 그림으로 나타낸 것이다.

Ⅳ. 제안하는 자체 인증 공개키의 특징 및 안전성

제안하는 상이한 도메인에서 검증 가능한 자체 인증 공개키의 특징 및 여러 형태의 공격에 대한 안전성을 분석해 보면 다음과 같다.

4.1 자체 인증성 및 제 3자에 의한 사용자 공개키의 대체 공격에 대한 안전성

키 발급 기관에서 자체 인증 공개키를 발급할 때 사용자의 개인식별정보가 공개키 안에 포함된다. 그

리고 키 발급 기관의 개인키로 서명을 하게 된다. 따라서, 이러한 자체 인증 공개키를 검증하기 위해서는 발급 대상이 되는 사용자의 개인식별정보를 사용해야 한다. 이와 같은 과정을 통해 키 발급 기관에 의한 자체 인증 공개키와 사용자간의 연계성이 이루어지게 되고, 자체 인증 공개키는 자체 인증성을 갖게 된다.

4.1.1 소인수 분해 문제 기반의 자체 인증 공개키

3.2절에서의 자체 인증 공개키의 경우 제 3의 공격자 C가 정당한 사용자 A의 자체 인증 공개키 SP_A에 포함되어 있는 y_A를 자신 또는 다른 사람의 공개키 y_C로 대체하는 것은 식 1)에서 모듈러 합성수 n_A 상에서 3의 역원 d_A를 구하기 위해 식 2)와 같은 형태로 n_A를 소인수 분해하는 것과 동일하므로 계산상 어렵다.

$$3 \cdot d_A = 1 (\mathrm{mod}\, \phi(n_A)) \quad (1)$$

$$\phi(n_A) = (q_A - 1)(r_A - 1) \quad (2)$$

한편, 자신이 신뢰하는 키 발급 기관으로부터 자체 인증 공개키를 발급 받고자 하는 사용자는 공개키 자체의 인증성 및 무결성을 검증하기 위해 키 발급 기관에서 공개한 공개키 파라미터 n_A를 사용한다. 그런데 자체 인증 공개키 SP_A를 발급 받고자 할 경우 식 3)에서와 같이 자신의 공개키 y_A를 먼저 모듈러 p_A상에서 계산하고, 이 값을 키 발급 기관에 전송하여 SP_A를 발급 받는다. 그리고 y_A의 생성시 사용된 파라미터가 식 4), 5), 6)을 통해 식 7)에 포함되고, SP_A는 식 7)에서와 같이 검증된다. 따라서 사용자는 SP_A가 y_A에 대한 정당한 자체 인증 공개키임을 확인하게 된다.

$$y_A = g_A{}^{x_A} (\mathrm{mod}\, p_A) \quad (3)$$

$$PI_A' = [ID_{KGC_A} \| n_A \| p_A \| g_A] \quad (4)$$

$$CI_{KGC_A}' = h(PI_A') \quad (5)$$

$$CI_A' = h(ID_A \| y_A) \quad (6)$$

$$CI_{KGC_A}' \oplus CI_A' \stackrel{?}{\equiv} SP_A{}^3 (\mathrm{mod}\, n_A) \quad (7)$$

4.1.2 이산 대수 문제 기반의 자체 인증 공개키

3.3절에서의 자체 인증 공개키 역시 제 3의 공격자 C가 s_{KGC_A}, r_{KGC_A}를 자신 또는 다른 사람의 공개키 y_C에 대한 자체 인증 공개키로 위조하기 위해서는 키 발급 기관 KGC_A의 개인키 x_{KGC_A}가 필요하고, 이를 계산하는 것은 식 8)에서 모듈러 p_A 상에서 이산 대수 문제를 푸는 것과 동일하므로 제 3의 공격자에 의한 위조는 계산상 어렵다.

$$y_{KGC_A} = g_A{}^{x_{KGC_A}} (\mathrm{mod}\, p_A) \quad (8)$$

또한, 사용자 A는 식 9)에서와 같이 먼저 키 발급 기관에서 공개한 파라미터로 자신의 공개키 y_A를 계산하고, 자체 인증 공개키 s_{KGC_A}, r_{KGC_A}를 발급 받는다. 그리고 y_A의 생성시 사용된 파라미터가 식 10), 11), 12)를 통해 식 13)에 포함되고, s_{KGC_A}, r_{KGC_A}를 식 13)에서와 같이 검증하기 때문에 s_{KGC_A}, r_{KGC_A}가 y_A에 대한 자체 인증 공개키임을 확인하게 된다.

$$y_A = g_A{}^{x_A} (\mathrm{mod}\, p_A) \quad (9)$$

$$PI_A' = [ID_{KGC_A} \| y_{KGC_A} \| p_A \| q_A \| g_A] \quad (10)$$

$$CI_{KGC_A}' = h(PI_A') \quad (11)$$

$$CI_A' = h(ID_A \| y_A) \quad (12)$$

$$g_A{}^{s_{KGC}} \stackrel{?}{\equiv} y_{KGC_A}{}^{h((CI_{KGC}' \oplus CI_A') \| r_{KGC})} \cdot r_{KGC_A} (\mathrm{mod}\, p_A) \quad (13)$$

그러므로 제안한 자체 인증 공개키 SP_A와 s_{KGC_A}, r_{KGC_A}는 사용자 A의 공개키 y_A에 대한 제 3자의 위·변조 공격에 안전하다.

4.2 검증 가능성 및 공개키 사용자의 부인 공격에 대한 안전성

자체 인증 공개키가 검증 가능성을 갖는다는 것은 서명 등의 어플리케이션에 사용되어 검증식이 성립하지 않는 경우, 서명 등의 과정에 이상이 있는 것인지 자체 인증 공개키에 이상이 있는 것인지 확인할 수 있음을 의미한다.

4.2.1 소인수 분해 문제 기반의 자체 인증 공개키

3.2절에서의 자체 인증 공개키를 검증자가 검증하는 과정 중에는 식 14)에서와 같이 y_A를 일방향 해쉬 함수에 입력하고, 이 값을 식 15)와 같은 검증식

을 통해 검증함으로써 SP_A로부터 y_A를 명시적으로 확인할 수 있다.

$$h(ID_A \| y_A) \tag{14}$$

$$CI_{KGC_A}'' \oplus h(ID_A \| y_A) \stackrel{?}{\equiv} SP_A{}^3 \pmod{n_A} \tag{15}$$

또한, 사용자 A가 $y_A \neq y_A''$인 y_A''를 이용하여 생성한 $CI_A'' = h(ID_A \| y_A'')$를 자신의 인증 정보, y_A''을 자신의 공개키라 주장함으로써 기존에 키 발급 기관으로부터 자체 인증 공개키를 발급 받은 공개키 y_A를 부인하는 경우, y_A''에 대한 자체 인증 공개키 SP_A''를 생성하기 위해 사용자 A는 식 1)에서의 키 발급 기관의 개인키 d_A가 필요하게 되고, 이를 아는 것은 식 2)와 같은 형태로 합성수 n_A를 소인수 분해하는 것과 동일하므로 y_A가 자신의 공개키가 아니라고 부인하거나 y_A''가 자신의 공개키라 주장하는 것은 계산상 어렵다.

4.2.2 이산 대수 문제 기반의 자체 인증 공개키

3.3절에서의 자체 인증 공개키를 검증자가 검증하는 과정에서는 정당성이 검증된 y_{KGC_A}와 4.1.2절에서 본 바와 같이 위조가 불가능한 자체 인증 공개키 s_{KGC_A}, r_{KGC_A}를 이용하여 A의 공개키 y_A의 정당성을 확인한다. 이 때, 식 16)에서와 같이 y_A를 역방향 연산이 불가능한 해쉬 함수에 입력하고, 이를 이용하여 검증식이 성립하는지를 명시적으로 확인한다.

$$g_A{}^{s_{KGC_A}} \stackrel{?}{\equiv} y_{KGC_A}{}^{h((h(PI_A')\oplus h(ID_A \| y_A)) \| r_{KGC_A})} \cdot r_{KGC_A} \pmod{p_A} \tag{16}$$

사용자 A가 $y_A \neq y_A''$인 y_A''를 이용하여 생성한 $CI_A'' = h(ID_A \| y_A'')$를 자신의 인증 정보, y_A''을 자신의 공개키라 주장함으로써 기존에 키 발급 기관으로부터 자체 인증 공개키를 발급 받은 공개키 y_A를 부인하는 경우 역시, y_A''에 대한 자체 인증 공개키 s_{KGC_A}'', r_{KGC_A}''를 생성하기 위해 사용자 A는 식 8)에서 모듈러 p_A 상에서 이산 대수 문제를 풀어야 하므로 y_A가 자신의 공개키가 아니라고 부인하거나 y_A''가 자신의 공개키라 주장하는 것은 계산상 어렵다.

4.3 상이한 도메인에서의 검증 가능성 및 제 3자에 의한 키 발급 기관의 공개키 파라미터의 대체 공격에 대한 안전성

자체 인증 공개키가 상이한 도메인에서의 검증 가능성을 갖는다는 것은 서로 다른 키 발급 기관을 신뢰하고 있는 사용자간 상대방의 자체 인증 공개키를 검증할 수 있음을 의미한다.

4.3.1 소인수 분해 문제 기반의 자체 인증 공개키

3.2절에서 사용자 B가 사용자 A의 자체 인증 공개키 SP_A를 검증하기 위해서는 KGC_A의 공개키 파라미터 n_A가 필요하다. 그러나 B는 KGC_B만을 신뢰할 뿐 KGC_A는 신뢰하지 않기 때문에 공개된 KGC_A의 공개키 파라미터를 그대로 사용할 수 없다. 이에 KGC_B가 정당성 검증을 마친 식 17)의 값에 대한 해쉬값 CI_{KGC_A}''(식 18))에 서명(식 19))하여 B에게 전송해 줌으로써, B로 하여금 식 20)과 같은 검증을 통해 PI_A'를 신뢰할 수 있게 한다. B는 기존에 KGC_B의 공개키 인자 PI_B를 사용하여 3.2.2에서와 같은 과정으로 자체 인증 공개키 SP_B를 발급 받았으므로, PI_B는 신뢰하고 있는 상태이다. 그리고 이러한 PI_B를 이용하여 CI_{KGC_A}''에 대한 KGC_B의 서명 S_B를 검증하므로, PI_A''를 신뢰하게 된다.

$$PI_A'' = [ID_{KGC_A} \| n_A \| p_A \| g_A] \tag{17}$$

$$CI_{KGC_A}'' = h(PI_A'') \tag{18}$$

$$S_B = CI_{KGC_A}''{}^{d_B} \pmod{n_B} \tag{19}$$

$$h(PI_A') \stackrel{?}{\equiv} S_B{}^3 \pmod{n_B} \tag{20}$$

또한, 제 3의 공격자 C가 KGC_A의 공개키 파라미터를 임의의 값 $CI_{KGC_C} = h(PI_C)$로 대체하는 경우 식 20)의 과정에서 $CI_{KGC_C} \neq S_B{}^3 \pmod{n_B}$임을 쉽게 알 수 있으며, CI_{KGC_C}에 대한 서명 S_C로 대체하기 위해서는 식 1), 2)에서와 같이 소인수 분해 문제를 풀어 KGC_B의 개인키 d_B를 알아야 하므로, C에 의한 이러한 공격은 계산상 어렵다.

4.3.2 이산 대수 문제 기반의 자체 인증 공개키

3.3절에서 사용자 A의 자체 인증 공개키 s_{KGC_A}, r_{KGC_A}를 검증하기 위해 사용자 B는 자신이 신뢰하

는 KGC_B로부터 A의 인증 정보(식 21), 22))에 대한 서명값(식 24), 25))을 전송 받아 검증식(식 26))이 성립하는지 확인함으로써 PI_A'를 신뢰하게 된다.

$$PI_A'' = [ID_{KGC_A} \| y_{KGC_A} \| p_A \| q_A \| g_A] \quad (21)$$

$$CI_{KGC_A}'' = h(PI_A'') \quad (22)$$

$$k_{KGC_B}^{+} \in_R Z_{q_B}^{*} \quad (23)$$

$$r_{KGC_B}^{+} = g_B^{k_{KGC_B}^{+}} (\bmod p_B) \quad (24)$$

$$s_{KGC_B}^{+} = x_{KGC_B} \cdot h(CI_{KGC_A}'' \| r_{KGC_B}^{+}) + k_{KGC_B}^{+} (\bmod q_B) \quad (25)$$

$$g_B^{s_{KGC}^{+}} \stackrel{?}{\equiv} y_{KGC_B}^{h(h(PI_A'') \| r_{KGC_B}^{+})} \cdot r_{KGC_B}^{+} (\bmod p_B) \quad (26)$$

또한, 제 3의 공격자 C가 KGC_A의 공개키 파라미터를 임의의 값 PI_C로 대체하는 경우 식 26)의 과정에서 $g_B^{s_{KGC}^{+}} \neq y_{KGC_B}^{h(h(PI_C) \| r_{KGC_B}^{+})} \cdot r_{KGC_B}^{+} (\bmod p_B)$ 임을 쉽게 알 수 있으며, $CI_{KGC_C} = h(PI_C)$에 대한 서명 S_C로 대체하기 위해서는 식 8)에서 이산 대수 문제를 풀어 KGC_B의 개인키 x_{KGC_B}를 알아야 하므로, C에 의한 이러한 공격은 계산상 어렵다.

4.4 상이한 도메인의 사용자간 어플리케이션으로의 적용 가능성 및 안전성

서로 다른 공개키 파라미터를 사용하는 사용자간에도 자체 인증 공개키를 사용하여 서명 생성 및 검증과 키 분배를 성공적으로 수행할 수 있다.

4.4.1 서명 생성 및 검증

본 논문에서는 소인수 분해 문제 기반 자체 인증 공개키를 사용하여 서명을 생성하고 검증하는 과정을 보았다. 키 발급 기관 KGC_A의 공개키 · 개인키 쌍은 소인수 분해 문제를 기반으로 하고 있기 때문에 이 때 사용되는 모듈러 n_A는 식 27)에서와 같이 큰 소수의 곱으로 이루어져야 한다. 그리고 사용자 A는 식 28)에서와 같이 p_A를 모듈러로 하여 공개키를 생성하고, p_A는 검증자에게 공개함으로써 식 29), 30), 31)을 통해 생성된 서명을 식 32)와 같이 검증할 수 있도록 한다. p_A를 선택할 때 Pohlig-Hellman 알고리즘을 이용하여 이산 대수 문제를 풀 수 없도록 하기 위해 큰 소수를 사용하므로 y_A, p_A로부터 x_A를 계산하는 것은 어렵다.

$$n_A = q_A \cdot r_A \quad (27)$$

$$y_A = g_A^{x_A} (\bmod p_A) \quad (28)$$

$$k \in_R Z_{p_A - 1}^{*} \quad (29)$$

$$R = g_A^{k} (\bmod p_A) \quad (30)$$

$$S = (M - x_A \cdot R) k^{-1} (\bmod p_A - 1) \quad (31)$$

$$y_A^{R} \cdot R^{S} \stackrel{?}{\equiv} g_A^{M} (\bmod p_A) \quad (32)$$

4.4.2 키 분배

본 논문에서 이산 대수 문제 기반 자체 인증 공개키는 키 분배 과정에 사용되었다. 사용자 A는 세션키 중간값을 계산하는 식 33)과 세션키를 생성하는 식 34)에서 자신의 개인키 x_A승 연산을 함으로써 본인만이 정당한 세션키를 생성할 수 있도록 한다. 또한, 사용자 B의 공개키를 사용함으로써 세션키를 이용한 통신 상대방이 B임을 알 수 있다.

$$w_A = g_B^{x_A} (\bmod p_B) \quad (33)$$

$$SK = w_B^{x_A} (\bmod p_A) \cdot y_B^{x_A} (\bmod p_B) \quad (34)$$

4.5 기존 방식과의 비교

제안하는 자체 인증 공개키는 기존의 방식에서 제공하던 자체 인증성 및 검증 가능성 외에 서로 다른 도메인에 속한 사용자간에 자체 인증 공개키를 검증할 수 있는 기능과 다른 도메인의 사용자간의 어플리케이션으로의 적용 가능성을 제공한다. 다음 [표 1]은 이상의 과정을 요약한 것이다.

[표 1] 기존 방식과 제안하는 방식의 특징 비교

특징 \ 방식	SC	VSC	PSC
자체 인증성	○	○	○
검증 가능성	×	○	○
상이한 도메인에서의 검증 가능성	×	×	○
상이한 도메인의 사용자간 어플리케이션으로의 적용성	×	×	○

SC(Self-Certified) : 자체 인증 방식
VSC(Verifiable Self-Certified) : 검증 가능한 자체 인증 방식
PSC(Proposed Self-Certified) : 제안하는 자체 인증 방식

V. 결 론

최근 유·무선 환경에 적합한 PKI가 구축되고 사용자가 늘어남에 따라, 상이한 도메인 하의 사용자간 신뢰 관계를 구축하기 위해 상호 연동의 개념이 활발히 연구되고 있다.

이와 같이 인증서 기반 방식에서는 서로 다른 도메인의 사용자간 상대방의 공개키를 검증할 수 있는 여러 방법이 존재하는 반면, 기존의 자체 인증 공개키는 동일한 도메인 내의 사용자간에만 검증이 가능했다.

이에 본 논문에서는 상호 신뢰 관계가 구축된 키 발급 기관하의 사용자간 검증할 수 있는 자체 인증 공개키를 제안하였으며, 이를 서명과 키 분배 등의 어플리케이션에 사용하여 서로 다른 파라미터를 사용하는 사용자간에도 이러한 과정을 수행할 수 있는 방안에 대하여 제안하였다.

이는 암호학적으로 잘 알려진 문제인 소인수 분해 문제와 이산 대수 문제를 기반으로 하여, 제 3의 공격자가 사용자의 공개키를 위·변조하거나 키 발급 기관의 공개키 파라미터를 대체하는 공격에 대해 안전함을 증명하였다. 또한, 사용자가 자신의 공개키를 부인하거나 다른 공개키를 자신의 공개키라 주장하는 공격에 대해 안전함을 증명하였다.

자체 인증 공개키는 자체 인증성을 제공하기 때문에 인증서를 사용할 필요가 없지만, 향후 최근 활발히 연구되고 있는 유·무선 PKI 환경과의 접목을 통해 최대한 적은 수의 인증서를 사용하면서 효율적인 암호 통신을 가능하게 하는 자체 인증 공개키 방식과 이를 이용한 효율적인 상호 연동 방안에 대한 연구가 이루어져야 할 것이다.

참 고 문 헌

[1] R. Housley, W. Ford, W. Polk, D. Solo, "RFC 2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile.", 1999.

[2] R. Housley, W. Ford, W. Polk, D. Solo, "RFC 3280, Internet X.509 Public Key Infrastructure Certificate and CRL Profile.", 2002.

[3] PKI forum White Paper, "PKI Interoperability Framework", PKI forum, 2001.

[4] PKI forum White Pater, "CA-CA Interoperability", PKI forum, 2001.

[5] A. Shamir, "Identity-based cryptosystems and signature schemes", Advances in Cryptology : Crypto '84, LNCS 196, Springer-Verlag, pp.47~53, 1985.

[6] M. Girault, "Self-certified public keys", Advances in Cryptology : Eurocrypt '91, LNCS 547, Springer-Verlag, pp.490~497, 1991.

[7] Seungjoo Kim, Soo-Hyun Oh, Sang-joon Park and Dongho Won, "Verifiable Self-Certified Public Keys", Proc. of WCC'99 : INRIA Workshop on Coding and Cryptography, pp.139~148, 1999.

[8] Byoungcheon Lee and Kwangjo Kim, "Self-certified Signatures", INDO-CRYPT 2002, LNCS 2551, Springer-Verlag, pp.200~214, 2002.

[9] Byoungcheon Lee and Kwangjo Kim, "Self-Certificate : PKI using Self-Certified Key", Proc. of Conference on Information and Cryptology 2000, Vol. 10, No.1, pp.65~73, 2000.

[10] A. Menezes, P. van Oorschot and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, Inc., 1997.

[11] H. Petersen and P. Hoster, "Self-certified keys-Concepts and Applications", Proc. of Communications and Multimedia Security'97, Champman & Hall, pp.102~116, 1997.

[12] W. Diffie, M. E. Hellman, "New directions in cryptography", *IEEE Trans. Inform. Theory, vol. 22*, pp.644~654, 1976.

[13] R. Rivest, A. Shamir and L. Adleman, "A method of obtaining digital signature and public key cryptosystem", ACM Communication 21, No.2, pp.120~126, 1978.

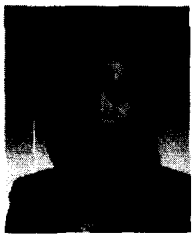
[14] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Trans. Inform. Theory, vol.31, pp.469~472, 1985.

[15] T. Beth, "Efficient Zero-Knowledge Identification Scheme for Smart Cards", Advances in Cryptology : Eurocrypt'88, LNCS 330, Springer-Verlag, pp.77~86, 1988.

〈著者紹介〉

정 영 석 (Young-Seok Chung) 학생회원
2002년 2월 : 성균관대학교 정보공학과 졸업(공학사)
2002년 3월~현재 : 성균관대학교 정보통신공학부 석사 과정

한 종 수 (Jong-Su Han) 학생회원
2002년 2월 : 성균관대학교 정보공학과 졸업(공학사)
2002년 3월~현재 : 성균관대학교 정보통신공학부 석사 과정



오 수 현 (Soo-Hyun Oh) 학생회원
1998년 2월 : 성균관대학교 정보공학과 졸업(공학사)
2000년 2월 : 성균관대학교 전기전자 및 컴퓨터 공학부 졸업(공학석사)
2000년 3월~현재 : 성균관대학교 정보통신공학부 박사 과정

원 동 호 (Dongho Won) 정회원
성균관대학교 전자공학과 졸업(학사, 석사, 박사)
1978년~1980년 : 한국전자통신연구원 전임연구원
1985년~1986년 : 일본 동경공업대 객원연구원
1988년~1999년 : 성균관대학교 교학처장, 전기전자 및 컴퓨터공학부장, 정보통신대학 원장, 정보통신기술연구소장
1996년~1998년 : 국무총리실 정보화추진위원회 자문위원
2002년~2003년 : 한국정보보호학회장
2003년~현재 : 성균관대학교 정보통신공학부 교수, 정통부 지정 정보보호 인증기술 연구센터장, 성균관대학교 연구처장