情報保護學會論文誌
第12券 第6號, 2002. 12

패스워드 기반 인증 키 공유 프로토콜에서의 효율성*

황 정 연**, 홍 석 희**, 박 혜 영**, 장 상 운**, 박 영 호***, 류 희 수****

Efficiency in the Password-based Authenticated Key Exchange

Jung Yeon Hwang**, Seok Hie Hong**, Hye-Young Park**,
Jang Woon Jang**, Young-Ho Park***, Heuisu Ryu****

요 약

지금까지 연구된 패스워드 기반 인증 키 공유 프로토콜에 대한 제안은 대부분이 증명 가능한 안전성 논의에 초점이 맞추어져 있었다. 하지만 모바일(mobile) 환경과 같은 실제적인 환경에서는 안전성만큼이나 효율성은 매우 중요한 논의사항이다. 본 논문에서는 랜덤 오라클(random oracle) 모델에서 안전성이 증명된 PAK[1]의 효율성에 대해 논의한다. PAK을 구성하는데 쓰이는 4개의 해쉬 함수 H_i $(1 \le i \le 4)$ 가운데 패스워드의 증명자를 생성하는 첫 번째 해쉬 함수는 PAK의 효율성에 가장 중요한 영향을 미친다. [1]에서 제시된 H_1의 구성에 대한 두 가지 방법을 분석하고, 위수 q인 또 다른 생성원을 사용하는 H_{1q} 방법이 효율성에 장점을 가짐을 보인다. [2]에서 제안과는 다르게, 패스워드에 대한 해쉬 함수 출력 값을 타원곡선 위의 점 또는 XTR 부분군의 원소로 변환시키는 부가적인 절차를 요구하지 않는 PAK2-EC와 PAK2-XTR을 제시한다. 마지막으로, PAK2 프로토콜을 SPEKE, AMP 그리고 SRP와 같은 패스워드 기반 인증 키 공유 프로토콜들과 계산량을 비교한다.

ABSTRACT

Proposals for a password-based authenticated key exchange protocol that have been published so far almost concentrated on the provable security. But in a real environment such as mobile one, efficiency is a critical issue as security. In this paper we discuss the efficiency of PAK which is secure in the random oracle model [1]. Among 4 hash functions in PAK, the instantiation for H_1, which outputs a verifier of the password, has most important effect on the computational efficiency. We analyze two different methods for H_1 suggested in [1] and we show that H_{1q} has merits in transforming to EC or XTR variants as well as in the efficiency. As an efficient variant, we propose PAK2-EC and PAK2-XTR which do not require any additional step converting a hash output into a point of elliptic curve or XTR subgroup when compared to the previous work on the PAK[2]. Finally we compare PAK2 with the password-based authenticated key exchange protocols such as SPEKE, SRP, and AMP.

Keyword : *Password-based authenticated key exchange*

* 본 연구는 한국전자통신연구원 연구과제(0701-2002-0043) 지원으로 수행하였습니다.
** 고려대학교 정보보호 기술센터(videmot, hsh, hypark, jangsw @cist.korea.ac.kr)
*** 세종 사이버 대학교(youngho@cybersejong.ac.kr)
**** 한국전자통신 연구원(hsryu@etri.re.kr)

Ⅰ. 서 론

1.1 배경

통신에서 사용자의 인증과 키 공유는 매우 주요한 요소들이다. 최근에는 위의 두 가지 목적을 사용자들에게 알려진 매우 단순한 패스워드만을 사용하여 이루려는데 중요한 관심이 모아져 왔다. 단지 복잡도가 매우 낮은(인간이 기억할 수 있을 정도의) 패스워드만을 사용하여, 프로토콜에 관여된 참가자의 신원에 대한 어떤 확신을 제공하고 동시에 참가자들 사이에 암호학적으로 강한 공유키를 생성하는 프로토콜을 *패스워드 기반 인증 키 공유 프로토콜*이라고 한다.

이 프로토콜은 공개키 기반 구조(PKI)의 사용이 불가능하거나 이용되는 데 어려움이 많은 다양한 환경에서 유용할 것이다. 특별히, 최근 네트워크의 폭발적인 증가로 인해서 무선 환경에서와 같은 작은 자원만이 허락되는(low resource) 환경에서의 원거리 접근 문제는 중요하다. 이런 환경에서 사용되는 안전성이 보장된 프로토콜의 주요한 관심사는 효율성에 대한 고려이다.

1.2 관련된 연구들

Bellovin과 Merritt[3]에 의해 제기된 패스워드 기반 세션 키 생성 문제 이후로 PKI(Public Key Infrastructure)의 사용 없이 위의 문제를 해결하려는 프로토콜에 대한 많은 연구가 진행되어 왔다. 최근에는 안전성 검증에 대해 몇 가지 논의가 제시되어 왔다. 예를 들자면 [4,5,6]의 프로토콜은 랜덤 오라클 모델에서 안전성이 증명되었고 [7,8]의 프로토콜들은 표준 모델에서 안전성이 증명되었다. 사실상, 지금까지 랜덤 오라클의 이상적인 가정이 필요 없는 표준 모델에서 안전성이 증명된 프로토콜은 Goldreich, Lindell에 의해 제시된 [8]와 Katz, Ostrovsky, 그리고 Yung에 의해서 제시된 [7], 두 가지 뿐이다.

그러나 표준 모델에서의 안전성 증명에도 불구하고 [7,8]에서 제안된 프로토콜은 높은 계산과 통신 복잡도가 요구된다. [7]의 프로토콜은 비상수적인 횟수의 라운드와 다자간 계산[9]이 요구된다. 따라서 이론적인 면에서만 가치를 갖는 모습이다. [7]의 프로토콜은 선택 암호문 공격에 대해 안전성이 증명된 Cramer-Shoup 암호 스킴의 연장된 암호화 기법을 바탕으로 구성되었다. 이것은 [8]의 프로토콜보다 실제적이기는 하지만 안전성 보장을 위해 역시 많은 지수승 연산을 요구함으로써 작은 자원만이 허락되는(low-resource) 환경에서 사용되기에는 적합하지 않다.

이렇듯 효율성의 관점에서 랜덤 오라클 모델에서 안전성이 증명된 PAK[4] 프로토콜을 주목하는 것은 가치가 있다. 랜덤 오라클 모델에서 안전성이 증명된 암호학적 스킴들이 실제로 매우 유용하다는 것은 이미 잘 알려져 있다. PAK 프로토콜은 패스워드를 이용하여 DH 형태의 키 교환 트랜스크립트(transcript)를 패스워드의 인증자로 곱하는 매우 단순한 방식의 암호화를 한다. 더욱이, 최근에는 PAK-EC와 PAK-XTR과 같이 타원 곡선 군 또는 XTR 부분군과 같은 보안 상수의 크기가 작고 원소의 처리에 대해 이점을 갖는 군들을 이용하는 PAK의 변형 프로토콜들도 제시되었다[2]. 이 프로토콜들 또한 랜덤 오라클 모델에서 안전성이 증명되었다.

1.3 논문의 결과

PAK 프로토콜은 4개의 해쉬 함수들 H_i $(1 \le i \le 4)$을 이용한다. 이 중 H_j $(2 \le j \le 4)$에 대해서는 랜덤한 출력 값을 얻기 위해서 SHA-1과 같이 속도가 매우 빠른 XOR과 같은 연산을 수반하는 해쉬 함수가 이용된다. 반면에 DH 형태의 키 교환을 위해 클라이언트 쪽에서 DH-트랜스크립트(transcript) g^x를 패스워드로 암호화하기 필요한 패스워드의 인증자를 생성하는 H_1의 계산에서는 유한체의 순환 부분군에 대한 지수승 연산이 수반된다. 따라서 H_1의 초기 설정은 PAK 프로토콜의 계산 효율성에 매우 중요한 영향을 미친다.

현재, H_1의 구성에는 두 가지가 제시되어 있다[1]. 첫 번째 방법은 $|r|$-비트 지수승이 필요하고 나머지 방법은 $|q|$-비트 지수승이 필요하다. 이런 두 가지 해쉬 방법을 각각 H_{1r}과 H_{1q}로 나타내기로 한다. 본 논문에서는 두 가지 해쉬 방법들을 분석하고 효율성을 비교한다. [1]에서 H_{1q}의 구성에 대한 언급에도 불구하고 그 장점은 구체적이지 않다. (본 논문에서 제시한 H_{1q}의 방법은 [1]의 H_{1q}와는 약간 다른 형태이다.) 사실, H_{1r}에서 서로 다른 위수를 갖는 두 원소들(위수 q인 생성원과 패스워드에 대한 위수 $p-1$인 해쉬 함수 출력 값)의 사용으로 인한 여러

가지 결점들은 H_{1q}에서 나타나지 않는다.

먼저, H_{1q}의 사용(PAK2)은 PAK 프로토콜의 효율성을 매우 증가시킨다. 중요한 아이디어는 또 다른 임의의 위수 q인 생성원을 이용하는 것이다. H_{1q}에서 클라이언트와 서버는 단지 $|q|=160$-비트 지수승에 대한 계산만 하므로 전체적인 계산량은 H_{1r}의 경우보다 매우 작아진다. 어떤 경우에서는 클라이언트가 같은 위수 q를 갖는 두 생성원에 대해 효율적인 동시 지수승 방법을 사용할 수도 있다. KOY 프로토콜과 비교해서는 PAK2의 계산과 통신 비용은 KOY의 1/3정도에 불과하다.

그리고, 1024비트의 해쉬 출력 값이 필요한 H_{1r}과는 대조적으로 H_{1q}는 단지 160비트의 해쉬 출력 값만 요구된다. 실제적으로, 160비트 해쉬 출력 값을 갖는 SHA-1과 같은 충돌 회피 함수를 사용하여 프로토콜을 설계할 수 있을 것이다.

H_{1q}방법을 사용함으로써 PAK2는 간단하게 효율적인 프로토콜들인, PAK2-EC 또는 PAK2-XTR로 변형되어진다. 이전의 PAK에 대한 이런 연구와는 대조적으로 PAK2-EC 또는 PAK2-XTR에서는 패스워드의 해쉬 출력 값을 타원 곡선 위의 점이나 XTR 부분군의 원소가 되도록 변형시키는 부가적인 작업이 필요 없어진다. 따라서 이런 프로토콜들은 모바일 환경과 같은 작은 자원이 허락되는 환경에 유용할 것이다.

1.4 논문의 구성

본 논문의 구성은 다음과 같다. 2절에서는 [1]의 PAK 프로토콜을 간단하게 살펴본다. 3절에서는 PAK에서 H_1의 구성에 대한 효율성을 고려하고 PAK2를 PAK과 KOY 프로토콜들과 비교한다. 4와 5절에서는 새로운 PAK2-EC와 PAK2-XTR을 제시하고 마지막으로 6절에서는 PAK2 프로토콜을 SPEKE와 AMP 그리고 SRP와 같은 패스워드 기반 인증 키 공유 프로토콜들과 효율성을 비교한다.

Ⅱ. PAK 프로토콜

PAK은 효율적인 3 라운드 패스워드 기반 인증 키 공유 프로토콜이다. 이 절에서는 본래의 스킴[1]과 해쉬 함수의 입력 값에 작은 차이를 가지는 PAK 프로토콜을 간략히 기술한다. 차이점은 공유된 정보를 확인하는 인증자를 만드는 해쉬 함수와 키 생성 해쉬 함수 H_i $(2\le i\le 4)$에 γ 대신 γ^{-1}값이 입력 값으로 사용된다. 이것은 비록 작은 양이지만 클라이언트 쪽의 계산량을 줄여준다. 보다 세부적인 내용은 [1]을 참조하기로 한다.

2.1 시스템 파라미터

x와 ℓ을 시스템의 암호학적 보안 상수(security parameter)들이라 하자. 여기서 x는 해쉬 함수와 세션 키들을 위한 보안 상수이고 ℓ ($\ell > x$)은 이산대수 기반의 공개키를 위한 보안 상수이다. 예를 들어, $x=160$와 $\ell=1024$인 경우, 현실적으로 안전한 값으로 여겨진다. p와 q를 다음을 만족하는 소수라고 하자. G_q를 위수 q인 $GF(p)$의 부분군으로 나타낸다. 주어진 $GF(p)$와 이의 부분군 G_q에 대해 G_q의 생성원 g_1는 임의로 선택된 공개파라미터이다. 공개파라미터는 모든 참가자에게 이용 가능하다고 가정한다. 다음에는 해쉬 함수를 정의한다.

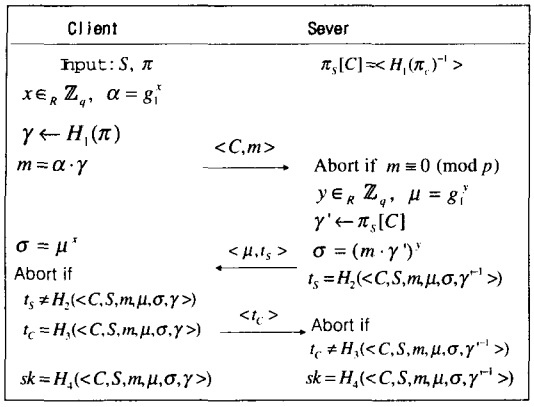
- $H_1 : \{0,1\}^* \rightarrow G_q$.
- $H_i : \{0,1\}^* \rightarrow \{0,1\}^x$, $(2\le i\le 4)$.

H_i $(1\le i\le 4)$는 독립적인 랜덤 함수들이다. H_1은 패스워드의 증명자(verifier), 즉 위수 q인 G_q의 원소를 만들기 위해 이용된다. H_2와 H_3은 클라이언트와 서버가 공유된 정보를 확인하기 위한 인증자(authenticator)를 만들기 위해 필요하다. H_4는 공유된 세션 키를 만들기 위해 이용된다.

PAK에서 H_j $(2\le j\le 4)$에 대해서는 랜덤한 출력 값을 얻기 위해서 SHA-1과 같이 속도가 매우 빠른 XOR과 같은 연산을 수반하는 해쉬 함수가 이용된다. 반면에 DH 형태의 키 교환을 위해 클라이언트 쪽에서 DH-트랜스크립트(transcript) g^x를 패스워드로 암호화하기 위해 필요한 패스워드의 인증자를 생성하는 H_1의 계산에서는 유한체의 순환 부분군에 대한 지수승 연산이 수반된다. 따라서 H_1의 초기 설정은 PAK 프로토콜의 계산 효율성에 매우 중요한 영향을 미친다.

2.2 프로토콜

프로토콜은 두 참가자인 클라이언트와 서버가 관계된다. 클라이언트와 서버의 아이디들은 각각 C, S로



〔그림 1〕 PAK2 프로토콜

나타낸다. 만일 $GF(p)$상의 연산이 분명하다면 제시 된 식에서 mod p에 대한 표현을 생략한다.(〔그림 1〕 참조)

[정리 1]

([1,4]) 만일 DDH 문제의 어려움을 가정한다면 PAK 프로토콜은 안전한 패스워드 기반의 인증 키 공유 프로토콜이다.

Ⅲ. PAK의 해쉬 함수 H_1에 관한 효율성

앞 절에서 살펴본 바와 같이 PAK 프로토콜의 효율성은 주요하게 H_1의 구성에 의해 영향을 크게 받는다. H_1은 패스워드의 인증자(verifier) (위수 q인 원소)를 생성하는 해쉬 함수이다. 이 절에서는 H_1의 두 가지 구성에 따라 PAK의 효율성은 매우 큰 차이를 가짐을 보인다. 이렇게 구성된 PAK 프로토콜을 KOY 프로토콜과 비교한다.

3.1 H_1의 구성

여기서는 〔1〕에서 제시된 H_1의 구성에 관한 두 가지 방법을 기술한다. 설명의 단순성을 위해서 두 가지 해쉬 함수를 H_{1r}과 H_{1q}로 표현하고 H_{1r}과 H_{1q}를 사용하는 PAK 프로토콜을 각각 PAK1과 PAK2로 표현하기로 한다. 여기서 PAK1은 〔4〕의 PAK 프로토콜과 동일함을 알 수 있다.

[방법 1] $H_{1r}(x) = H(x)^{(p-1)/q}$.

여기서 $H(x)$는 매우 높은 확률로 위수 $p-1$인 1024 비트 난수를 출력한다.

[방법 2] $H_{1q}(x) = g_2^{H'(x)}$

여기서 g_2는 위수 q인 또 다른 임의의 공개 생성원이다. $H'(x)$는 매우 높은 확률로 160 비트 난수를 출력한다.

해쉬 함수 $H(x)$의 정의는 〔1〕을 참조한다. 위에서의 H_{1q}의 기술은 〔1〕에서 제시한 방법과 조금 차이가 있다. 〔1〕에서 기술한 H_{1q}의 정의는 다음과 같다.

$$H_{1q}(x) = g_1 \cdot g'^{H(x)} \bmod q$$

여기서 $g' = H(ASCII(1))^{(p-1)/q}$ 이고 $H(ASCII(1))$는 임의의 난수 값을 출력한다. 〔1〕에서 정의한 H_{1q}의 정의에서는 여분의 g_1의 곱셈을 한 번 더 수행한다.

사실상 효율적인 H_{1q}을 구성하는 중요 아이디어는 위수 q인 또 다른 생성원 g_2을 이용하는 것이다. 이 경우에 단지 두 생성원 g_1와 g_2는 시스템의 초기화 단계에서 임의로(randomly) 그리고 독립적으로(independently) 선택되어 공개된 파라미터라고 가정한다. 다시 말해, 어느 누구도 한 생성원에 대한 다른 생성원의 이산 대수를 알지 못한다. 이것은 안전성을 위해 매우 중요한 가정이다. 만일 $\log_{g_1} g_2$ 또는 $\log_{g_2} g_1$의 값이 알려진다면 PAK2 프로토콜에서 패스워드 공간에 대한 오프라인 전수조사 공격(off-line dictionary attack)이 가능함을 쉽게 알 수 있다.

3.2 효율성

계산량에 있어서 PAK2는 전체적으로 PAK1 보다 적은 양이 소요된다. 이것은 PAK1의 경우에 클라이언트가 $|r|$-비트 지수승을 해야 하는 반면에 PAK2에서는 클라이언트와 서버 모두 $|q|$-비트 지수승만 하면 되기 때문이다. 지금부터는 PAK1, PAK2 그리고 KOY의 효율성을 구체적으로 비교한다. KOY 프로토콜은 표준 모델(standard model)에서 안전성이 증명된 실제적인 패스워드 기반 인증 키 공유 프로토콜이다.

이 후 제시된 표에서, $GF(p)$에 대한 몇 가지 연산들을 다음과 같이 나타낸다.

- Exp_n : n비트 지수승 연산
- $sExp_{(n,k)}$: k개의 고정된 밑으로 계산되는 n비트

동시 지수승 연산
- M : 곱셈 연산
- $Hash_m$: m비트 해쉬 출력 값을 갖는 해쉬 함수의 사용

그리고 연산에 관한 전체 계산량을 유도한 가정은 다음과 같다.

(1) 기본적인 이진 방법이 Exp_{160}, $sExp_{(160,2)}$ 그 리고 $sExp_{(160,5)}$의 연산 횟수 계산에 이용되었다.
(2) 일반적으로 $GF(p)$상의 제곱에 대한 연산은 곱셈에 대한 연산의 최대 80%가 소요된다고 가정한다. 그러면 $\mathrm{Exp}_{|q|} \approx (\log q$번의 제곱 연산과 $1/2\log q$번의 곱셈 연산) $\approx (1.3\log q$번의 곱셈 연산)이므로 Exp_{160}은 $GF(p)$상에서 208번의 곱셈 연산이 필요하다.
(3) $sExp_{(|q|,2)}$의 연산은 $2\,\mathrm{Exp}_{|q|}$의 60%에 해당 한다고 가정한다. 위의 가정에 대한 이유는 $sExp_{(|q|,2)} \approx (\log q$번의 제곱 연산+$3/4\log q$번의 곱셈 연산+두 기저의 곱셈에 대한 사전 계산량) 이기 때문이다. 따라서 $sExp_{(|q|,2)} \approx 0.6 \cdot 2\mathrm{Exp}_{|q|}$ 이므로 $sExp_{(|q|,2)}$는 $GF(p)$상에서 250번의 곱셈연 산이 필요하다.
(4) $sExp_{(|q|,5)}$의 연산은 $5\,\mathrm{Exp}_{|q|}$의 29%에 해당한다고 가정한다. 위의 가정에 대한 이유는 $sExp_{(|q|,5)} \approx (\log q$번의 제곱 연산+$31/32\log q$ 번의 곱셈 연산+5개 기저의 곱셈 조합에 대한 사전 계산량) 이기 때문이다. 따라서 $sExp_{(|q|,5)} \approx 0.29 \cdot 5\mathrm{Exp}_{|q|}$ 이므로 $sExp_{(|q|,5)}$는 $GF(p)$상에서 309번의 곱셈 연산이 필요하다.

주어진 프로토콜의 계산량은 주요하게 $GF(p)$의 곱셈의 횟수에 대한 관점으로 다루어진다. 제시된 표 1에서 알 수 있듯이 PAK1(최초 제안된 본래의 PAK 프로토콜[4])의 클라이언트 쪽 계산량은 KOY 프로토콜의 클라이언트 쪽과 비슷하다. 그러나 PAK2의 경우에는 1/3정도에 불과하다. 서버 쪽의 계산량에서는 PAK1과 PAK2 모두 KOY의 1/3이다. 따라서 전체적으로 PAK2의 경우에는 대략 KOY의 1/3 정도이고 PAK1의 경우에는 KOY의 7/10이다. PAK2가 KOY 보다 매우 효율적임을 알 수 있다.

〔표 1〕 PAK, PAK2 그리고 KOY의 계산량 비교

(a) PAK, PAK2

연산	PAK1		PAK2	
지수승 $GF(P)$	1 Exp_{864} 2 Exp_{160}	2 Exp_{160}	1 Exp_{160} 1 $sExp_{(160,2)}$	2 Exp_{160}
합계	1331 M	416 M	458 M	416 M
해쉬	1 $Hash_{1024}$ 1 $Hash_{160}$	3 $Hash_{160}$	4 $Hash_{160}$	3 $Hash_{160}$
서 명				

(합계 : $GF(P)$에서의곱셈, $|p|=1024$ 비트)

(b) KOY

연산	KOY	
지수승 $GF(P)$	4 Exp_{160} 1 $sExp_{(160,2)}$ 2 $sExp_{(160,5)}$	3 Exp_{160} 1 $sExp_{(160,2)}$ 2 $sExp_{(160,5)}$
합계	1391 M	1154 M
해쉬	1 $Hash_{160}$	1 $Hash_{160}$
서 명	1 서명생성	1 서명증명

KOY에서는 Cramer-Shoup 암호 시스템[11]의 연장된 형태와 일회용 전자서명 기법[10]을 이용한다. 따라서 이런 기법들은 계산 및 통신 복잡도를 증가시키는 요인으로 작용한다. 이 복잡도는 Cramer-Shoup 암호 시스템이 안전성을 위해 기본적으로 요구하는 양이므로 요구된 복잡도는 피할 수 없는 양으로 여겨진다. [7]로부터 쉽게 알 수 있듯이 통신 복잡도는 PAK2에 있어서 약 5배 이상이 소요됨을 알 수 있다. 그러므로 PAK2 프로토콜이 효율성이 특별히 중요한 환경에서 더 적합함을 알 수 있다.

[참고]
- [4]에서와 같이, 해쉬 함수 H_i $(2 \le i \le 4)$의 입력 값으로 γ 대신 패스워드 π가 이용된다면 $m = a \cdot \gamma = g_1^{\gamma} g_2^{H(\pi)}$를 계산하기 위해서 같은 위수 q를 갖는 고정된 두 생성원에 대한 동시 지수승 연산 기법을 이용할 수 있다.
- 해쉬 함수는 계산량에 있어서 중요하게 고려되지 않는다.
- 전체 계산량에 있어서 KOY 프로토콜의 경우 서명연산은 반영하지 않았다. 왜냐하면 사용된 서명 기법은 매우 효율적으로 계산할 수 있는 일회용 서명 스킴이기 때문이다.

3.3 PAK2-Z 프로토콜

PAK2의 구성과 비슷하게, PAK의 비대칭 형태의 프로토콜인 PAK-Z[1]의 프로토콜도 H_{1q}의 해쉬 방법을 이용하여 효율성을 향상시킬 수 있다. PAK2-Z의 세부적인 기술은 생략하기로 한다.

Ⅳ. PAK2-EC

이 장에서는 타원곡선을 이용하는 PAK2 프로토콜을 제시한다. 본래의 PAK 프로토콜에서는 패스워드의 해쉬 출력 값을 타원 곡선 위의 점으로 변환시키는 과정이 필요했으나 PAK2의 경우는 이런 부가적인 과정이 필요하지 않다.

4.1 타원곡선 군

타원곡선은 유한체 F_{p^m}상에서 정의된다. 여기서 p는 소수이고 m은 임의의 양의 정수이다. 타원곡선 방정식을 간단한 Weierstrass의 형태로 표현하면 다음과 같다.

$E_{a,b}: Y^2 = X^3 + aX + b$ (p가 홀수일 때).
$E_{a,b}: Y^2 + XY = X^3 + aX^2 + b$ ($p=2$일 때).

[12,13,14]에서 보여진 바와 같이, 타원곡선 상의 점들과 무한원점을 원소로 갖는 집합은 덧셈에 관한 군을 이룬다.

여기서는 이 그룹을 $E(F_{p^m})$으로 나타내기로 한다. 그리고 $\#E(F_{p^m}) = r \cdot q$이라고 가정한다. 여기서 r은 $GCD(r,q)=1$인 코팩터(cofactor)이다. 마지막으로 G_q를 위수 q인 $E(F_{p^m})$의 부분군으로 나타낸다.

4.2 PAK2-EC 프로토콜

PAK2-EC 프로토콜은 PAK2 프로토콜과 구조적으로 매우 비슷하다. PAK2에서의 모든 연산들은 단순히 어떤 부가적인 단계 없이 타원곡선 그룹에 대응되는 연산으로 대치되어진다. 즉, 유한체 상의 곱셈은 타원곡선 군에서 덧셈이 된다.[그림 2 참조]

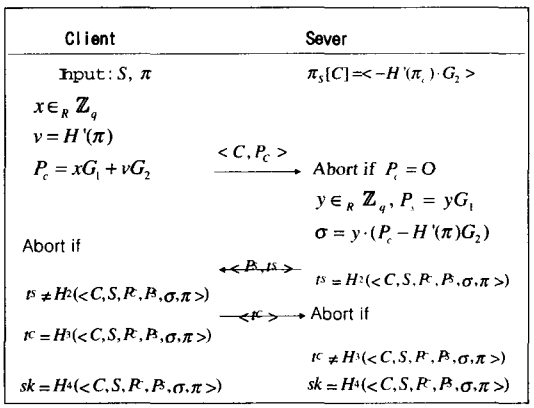
[초기화]

주어진 타원곡선 군 $E(F_{p^m})$과 이의 부분군 G_q에 대하여 G_q의 두 생성원 G_1, G_2이 임의로 그리고 독립적으로 선택되고 공개된다.

[프로토콜]

패스워드를 통해 인증 및 공유된 키를 생성하기 위하여 클라이언트와 서버는 다음과 같이 진행한다.

1. 먼저 클라이언트는 다음과 같은 작업을 한다.
   (가) 클라이언트는 임의의 수 $x \in_R Z_q$를 선택하고 $v = H'(\pi)$를 계산한다. 여기서 π는 클라이언트의 패스워드이다.
   (나) 클라이언트는 $P_c = xG_1 + vG_2$를 계산한다.
   (다) 클라이언트는 점 $P_c = (X_c, Y_c)$를 (X_c, y_c)의 형태로 압축한다. 여기서 y_c는 Y_c를 압축된 형태로 나타낸 1비트 정보이다.(통신상의 효율성을 위 해 타원곡선의 점에 대해 압축된 형태를 이용한다.) 그리고 클라이언트는 서버에게 C, P_c를 보낸다.
2. C, P_c를 수신하고 난 후,
   (가) 서버는 임의의 수 $y \in_R Z_q$를 선택하고 $P_s = yG_1$를 계산한다.
   (나) 서버는 $\sigma = y \cdot (P_c - H'(\pi) \cdot G_2)$를 계산한다. 여기서 π는 클라이언트의 패스워드이다.
   (다) 서버는 인증자 $t_s = H_2(\langle C, S, P_c, P_s, \sigma, \pi \rangle)$를 계산한다. 그리고 P_s, t_s를 클라이언트에게 보낸다.
3. 클라이언트는 $\sigma = x \cdot P_s$를 계산하고 난 후 $t_s = H_2(\langle C, S, P_c, P_s, \sigma, \pi \rangle)$인지를 검증한다. 만일 검증이 유효하면 클라이언트는 서버에게 $t_c = H_3(\langle C, S, P_c, P_s, \sigma, \pi \rangle)$를 보낸다.
4. 서버는 $t_c = H_3(\langle C, S, P_c, P_s, \sigma, \pi \rangle)$인지를 검증한다.
5. 공유된 키 값은 $K = H_4(\langle C, S, P_c, P_s, \sigma, \pi \rangle)$이다.



[그림 2] PAK2-EC 프로토콜

4.3 PAK2-EC의 안전성

PAK2-EC의 안전성 증명은 직접적으로 PAK-EC[2]의 안전성 증명을 따른다.

4.4 PAK2-EC의 효율성

본 절에서는 PAK-EC[2]와 PAK2-EC의 계산 복잡도를 비교한다. 여기서는 $\#E(F(p))=r\cdot q$이고 $GCD(r,q)=1$인 $E(F(p))$의 부분군을 고려한다. 현실적으로, $|p|\approx 160$와 $|q|\approx 160$ 그리고 $|r|\leq 2$인 경우가 안전한 보안 상수의 크기로 여겨진다. 두 프로토콜 PAK-EC와 PAK2-EC의 주요한 차이점은 해쉬 출력 값의 타원곡선의 원소로의 변환 여부에 있다. 즉, PAK-EC에서는 H_1에 대한 패스워드의 해쉬 출력 값을 G_q의 임의의 점으로 변환시키는 작업이 필요하지만 PAK2-EC에서는 그런 변환이 필요하지 않다. 따라서, 우선 이 점에서 PAK2-EC는 프로토콜의 설계에 대한 단순성을 갖는다.

[표 2]에서 타원곡선상의 몇 가지 연산들은 다음과 같이 표현된다.

- Ml_n : n 비트 점 곱셈 연산(point multiplication).
- $sMul_{(n,k)}$: k개의 고정된 밑으로 계산되는 n비트 동시 곱셈 연산.
- Add : 점 덧셈 연산 (point addition).

그리고 [표 2]에서 연산에 관한 프로토콜의 전체 계산량을 유도한 가정은 다음과 같다.

(1) 기본적인 이진 방법이 Ml_{160}, $sMul_{(160,2)}$의 연산 횟수 계산에 이용되었다.

[표 2] PAK-EC와 PAK2-EC의 계산량 비교

연 산	PAK-EC		PAK2-EC	
	client	sever	client	sever
점 곱 셈	2 Ml_{160} 1 Ml_2	2 Ml_{160}	1 $sMul_{(160,2)}$ 1 Ml_{160}	2 Ml_{160}
점 덧 셈	1 Add	1 Add	1 Add	1 Add
해쉬 값의 변환	1 inversion 1 square root 1 gcd			
합 계	4741 M	4320 M	4752 M	4320 M
해 쉬	4	3	4	3

(2) 점 2배 (point doubling) 연산은 8번의 체 곱셈과 점 덧셈 (point addition)은 11번의 체 곱셈이 소요된다 (Jacobian 좌표를 갖는 경우에서 와 같다).[15]

(3) $Ml_{|q|}\approx \log q$번의 2배 연산+ $1/2\log q$번의 덧셈 연산 $\approx 13.5\log q$번의 $GF(p)$상의 곱셈 $\Rightarrow Ml_{160}\approx$ 2160번의 $GF(p)$상의 곱셈.

(4) $sMul_{(|q|,2)}$의 연산은 $2\,Ml_{|q|}$의 60%에 해당한다고 $\log q$ 가정한다. 위의 가정에 대한 이유는 $sMul_{(|q|,2)}\approx$ ($\log q$번의 2배 연산+$3/4\log q$번의 덧 셈 연산+두 기저의 곱셈에 대한 사전 계산량)이기 때문이다. 따라서 $sMul_{(|q|,2)}\approx 0.6\cdot 2\mathrm{Exp}_{|q|}\Rightarrow sMul_{(|q|,2)}\approx GF(p)$상의 2592번의 덧셈 연산.

(5) PAK-EC 상에서 해쉬 출력 값의 변환 작업은 확률적으로 평균 2번이 필요하다.

(6) PAK-EC의 연산량 계산에서 1번의 법 p에 대한 역수 계산과 1번의 최대 공약수 계산은 생략되었다.

만일 PAK-EC에서 1번의 법 p에 대한 역수 계산과 1번의 최대 공약수 계산이 고려된다면 PAK2-EC는 PAK-EC보다 약간 더 효율적임을 알 수 있다.

통신 복잡도에 있어서 PAK-EC와 PAK2-EC는 PAK 또는 PAK2의 1/6 정도에 해당한다. 이것은 타원곡선의 안전성이 160-비트 크기의 유한체에서도 보장되기 때문이다.

V. PAK2-XTR

이 장에서는 XTR 부분군을 이용하는 PAK2 프로토콜을 제시한다. 본래의 PAK 프로토콜에서는 패스워드의 해쉬 출력 값을 XTR 부분군의 원소로 변환시키는 과정이 필요했으나 PAK2의 경우는 이런 부가적인 과정이 필요하지 않다.

5.1 XTR 암호 시스템

XTR[16]이란 확장체의 부분군의 원소들을 트레이스를 이용하여 표현하고 지수승을 계산하는 방법이다. XTR은 $GF(p^6)$에 대한 명백한 구성없이 $GF(p^6)$의 안전성을 이루기 위해 $GF(p^2)$의 연산을 이용한 최초 방법이다. 다음에서는 XTR의 시스템 파라미터

에 대해 간단히 살펴보기로 하자. p는 $p\equiv 2 \bmod 3$와 $p\neq 8 \bmod 9$를 만족하는 170 비트 소수이다. p에 대해 계산된 6차 싸이클로토믹(cyclotomic) 다항식, 즉, $\phi_6(p)=p^2-p+1$는 길이 160비트의 소수 q를 인수로 갖지만 q^2은 약수로 갖지 않는다. G_q를 위수 q인 $GF(p^6)$의 부분군이라 하고 $g\in GF(p^6)$를 G_q의 생성원이라 하자. $Tr(g)$는 임의의 k에 대해 $Tr(g^k)$를 계산하기 위해 사용된다. 여기서는 $Tr(g)$를 XTR 생성원으로 부르기로 한다.

5.2 PAK2-XTR 프로토콜

타원곡선에 대한 변형에서와 같이 PAK2의 연산은 XTR 부분군의 대응되는 연산으로 쉽게 대치된다.[그림 3 참조]

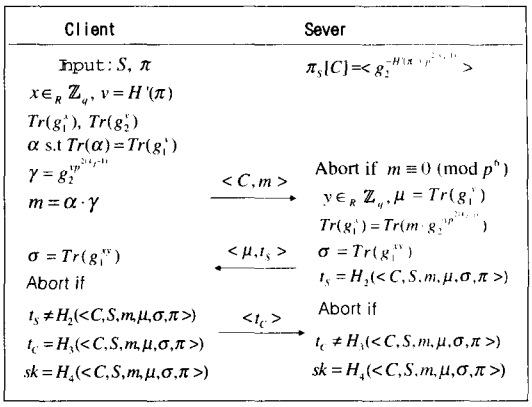
[초기화]

주어진 XTR 부분군에 대하여 두 XTR-생성원 $Tr(g_1)$과 $Tr(g_2)$가 임의로 그리고 독립적으로 선택하고 공개된다.

[프로토콜]

패스워드를 통해 인증 및 공유된 키를 생성하기 위하여 클라이언트와 서버는 다음과 같이 진행한다.

1. 먼저, 클라이언트는 다음과 같은 작업을 한다.
   (가) 클라이언트는 임의의 수 $x\in_R Z_q$를 선택하고 $v=H(\pi)$를 계산한다. 여기서 π는 클라이언트의 패스워드이다.
   (나) 클라이언트는 [16]의 알고리즘 2.3.7과 $Tr(g_1)$, $Tr(g_2)$, x 그리고 v를 이용하여 $Tr(g_1^x)$, $Tr(g_2^v)$을 계산한다.
   (다) 클라이언트는 $Tr(g_1^x)=Tr(\alpha)$를 만족하는 α를 계산한다. 사실상, $\alpha=g_1^{xp^{2(i-1)}}$ $(0\le i\le 2)$이다.
   (라) 사전에 미리 정해진 순서 인덱스 j에 대하여, 집합 $\{g_2^v, g_2^{vp^2}, g_2^{vp^4}\}$에 대한 자연적인 순서를 이용하여 $Tr(g_2^v)=Tr(\gamma=g_2^{vp^{2k-1}})$를 만족하는 γ를 계산한다. 예를 들어, 만일 $j=1$이면, $\{g_2^v, g_2^{vp^2}, g_2^{vp^4}\}$의 원소 가운데 절대 값이 가장 작은 값을 찾는다.
   (마) 클라이언트는 $m=\alpha\cdot\gamma$을 계산하고 난 후, 서버에게 C, m을 보낸다.



[그림 3] PAK2-XTR 프로토콜

2. C, m을 수신하고 난 후,
   (가) 서버는 m에 대한 유효성 검증을 한다. 즉, 만일 $m\equiv 0 \bmod p^6$이면 세션을 끝낸다.
   (나) 서버는 임의의 수 $y\in_R Z_q$를 선택하고 [16]의 알고리즘 2.3.7을 이용하여 $Tr(g_1^y)$를 계산한다.
   (다) 서버는 $\alpha=m\cdot g_2^{vp^{2(k-1)}}$를 계산한다. 여기서 j는 사전에 미리 정해진 값이다. 서버는 $g_2^{vp^{-2(k-1)}}$값을 직접 저장하고 있다고 가정할 수 있다. 그리고 $Tr(\alpha)=Tr(g_1^x)$를 계산한다.
   (라) 서버는 $Tr(g_1^x)$, y 그리고 [16]의 알고리즘 2.3.7을 이용하여 $\sigma=Tr(g_1^{xy})$를 계산한다.
   (마) 서버는 인증자 $t_s=H_2(\langle C, S, m, \mu, \sigma, \pi\rangle)$를 계산한다. 그리고 μ, t_s를 클라이언트에게 보낸다.
3. 클라이언트는 $\mu=Tr(g_1^y)$, x, 그리고 [16]의 알고리즘 2.3.7을 이용하여 $\sigma=Tr(g_1^{xy})$를 계산하고 난 후 $t_s=H_2(\langle C, S, m, \mu, \sigma, \pi\rangle)$인지를 검증한다. 만일 검증이 유효하면 클라이언트는 서버에게 $t_s=H_3(\langle C, S, m, \mu, \sigma, \pi\rangle)$를 보낸다.
4. 서버는 $t_s=H_3(\langle C, S, m, \mu, \sigma, \pi\rangle)$인지를 검증한다.
5. 공유된 키 값은 $K=H_4(\langle C, S, m, \mu, \sigma, \pi\rangle)$이다.

5.3. PAK2-XTR 안전성

PAK2-XTR의 안전성 증명은 직접적으로 PAK 2의 안전성 증명을 따른다. 단지, PAK2 프로토콜과의 차이는 두 번째 라운드의 통신 메시지에 있다. 하지만 PAK2-XTR의 경우는 PAK2의 경우보다 공격자에게 주어지는 정보가 더 불확실함을 쉽게 알 수 있다.

5.4 PAK2-XTR 효율성

본 절에서는 PAK-XTR과 PAX2-XTR의 효율성에 대하여 비교한다. XTR 시스템의 보안 상수의 크기는 $|p|=170$이고 $|q|=160$이다. PAK2-EC에서와 비슷하게 PAK2-XTR에서는 해쉬 출력 값을 XTR 군의 랜덤한 값의 트레이스로 변환시키는 과정이 필요하지 않다.

〔표 3〕에서, $GF(p)$에 대한 몇 가지 연산들을 다음과 같이 나타낸다.

- $XExp_n$: n비트 XTR 지수승 연산 (〔17〕의 알고리즘 2.3.7을 이용하는 방법)
- $Ml_{GF(p^6)}$: $GF(p^6)$내에서의 곱셈 연산
- $Inv_{GF(p)}$: $GF(p)$내에서의 역수 계산

그리고, 연산에 관한 전체 계산량을 유도한 가정은 다음과 같다.

(1) $XExp_q$의 계산. 주어진 $x\in Z_q$에 대한 $Tr(g^x)$은 〔17〕의 알고리즘 2.3.7에 의해 $GF(p)$에서 $8\log q$번의 곱셈을 요구한다. 따라서 $XExp_{170}$과 $XExp_{160}$은 각각 $8\cdot 170=1360$번과 $8\cdot 160=1280$번의 곱셈을 요구한다.

(2) 〔16〕의 알고리즘 5.6에 의해서 $v=Tr(v')$을 만족하는 v'을 계산하는 데에는 $GF(p)$에서 $5.3\log q$번의 곱셈을 요구한다. $5.3\cdot 170=901$번의 곱셈이 필요하다.

〔표 3〕 PAK-XTR과 PAK2-XTR의 계산량 비교

연산	PAK-XTR		PAK2-XTR	
	client	sever	client	sever
지수승	2 $XExp_{160}$	1 $XExp_{160}$ 2 $XExp_{160}$	3 $XExp_{160}$	2 $XExp_{160}$
트레이스 역 상	2	1	2	
곱 셈	4 $M_{GF(p^6)}$	3 $M_{GF(p^6)}$	1 $M_{GF(p^6)}$	1 $M_{GF(p^6)}$
해쉬 값의 변환	1 $Hash_{1024}$ 4 $M_{GF(p^6)}$ 1 $Inv_{GF(p)}$ 1 $XExp_{170}$			
합 계	6026 M	6467 M	5900 M	2738 M
해 쉬	3 $Hash_{160}$	3 $Hash_{160}$	4 $Hash_{160}$	3 $Hash_{160}$
통신량	$2\cdot 340+2\cdot 160+1\cdot 1$ = **1002 bits**		$1\cdot 1024+1\cdot 340+2\cdot 160$=**1684 bits**	

(3) $GF(p^6)$의 원소들의 곱셈을 계산하기 위해서 Karatsuba 방법[18]을 이용했다. $GF(p^6)$의 두 원소의 곱셈은 $GF(p)$에서 18번의 곱셈으로 수행된다.

(4) $GF(p^6)$의 원소들에 대해서 p^2 또는 ap^4의 지수승은 어떤 곱셈 연산 없이 쉽게 계산할 수 있다. 따라서 $GF(p^6)$의 원소에 대한 트레이스도 쉽게 계산된다.

주어진 프로토콜의 계산량은 주요하게 $GF(p)$에서의 곱셈의 횟수로 다루어진다.

결과적으로, PAK2-XTR에서 클라이언트는 총 $34.6\log p+O(c)$번의 곱셈이 요구되고 서버는 총 $16\log p+O(c)$번의 곱셈이 요구된다. 클라이언트는 $Tr(g_1^x)$과 $Tr(g_2^y)$를 계산하기 위해 $2\cdot 8\log q$번의 곱셈이 필요하다. 미리 정해진 인덱스 j에 대해 γ는 쉽게 계산된다. $Tr(g_2^y)$으로부터 γ의 후보들 가운데 하나를 계산하기 위해 $5.3\log p$의 곱셈이 필요하다. 마지막으로 σ를 계산하기 위해 $8\log q$번의 곱셈이 필요하다. 서버는 μ와 σ를 계산하기 위해 $2\cdot 8\log q$번의 곱셈이 필요하다.

클라이언트에 대한 계산량은 PAK2-XTR의 경우가 PAK-XTR의 경우보다 약간 더 작다. 하지만 서버의 경우에는 PAK2-XTR이 PAK-XTR보다 훨씬 효율적이다. PAK2-XTR은 PAK-XTR보다 계산량에 있어서 약 1/2.4에 불과한 반면에 통신량은 1.6 정도 더 늘어나게 된다.

PAK2-XTR에서는 클라이언트가 패스워드의 해쉬 출력 값을 XTR 부분군의 임의의 원소로 변환시키는 작업이 필요하지 않으므로 프로토콜 설계의 단순성은 물론 계산량도 줄일 수 있다.

Ⅵ. 비교와 결론

본 논문에서는 〔1〕에서 제시된 해쉬 함수 H_1에 관한 PAK 프로토콜의 효율성을 논의 하였다. H_{1q}방법을 사용하는 PAK2 프로토콜이 PAK〔4〕보다 매우 적은 계산량이 요구됨을 보였다. 뿐만 아니라 타원곡선 또는 XTR 부분군을 이용하는 PAK2의 변형된 형태를 제시하여 이전의 작업과는 다르게 패스워드의 해쉬 값을 타원곡선 위의 점이나 XTR 부분군의 원소로 변환시키는 부가적인 작업 없이 쉽게 위의 군을 사용하는 프로토콜들(PAK2-EC, PAK2-

〔표 4〕 대칭형 패스워드 기반 인증 키 공유 프로토콜들의 계산량 비교

프로토콜	Pass	해쉬 함수		지수승		곱셈		안전성 증명여부
		Client	Sever	Client	Sever	Client	Sever	
SPEKE	3	1 $Hash_{1024}$ 1 $Hash_{160}$	3 $Hash_{160}$	2 Exp_{1023}	2 Exp_{1023}	**2660**	**2660**	증명됨
PAK2	3	4 $Hash_{160}$	3 $Hash_{160}$	1 $sExp_{160}$ 1 Exp_{160}	2 Exp_{160}	**458**	**4162**	증명됨
KOY	3	1 $Hash_{160}$	1 $Hash_{160}$	2 $sExp_{(160,5)}$ 1 $sExp_{(160,2)}$ 4 Exp_{160}	2 $sExp_{(160,5)}$ 1 $sExp_{(160,2)}$ 3 Exp_{160}	**1391**	**1154**	증명됨 (표준모델)

〔표 5〕 비대칭형 패스워드 기반 인증 키 공유 프로토콜들의 계산량 비교

프로토콜	Pass	해쉬 함수		지수승		곱셈		안전성 증명여부
		Client	Sever	Client	Sever	Client	Sever	
B-SPEKE	4	1 $Hash_{1024}$ 2 $Hash_{1023}$	2 $Hash_{1023}$	1 Exp_{1} 2 Exp_{1023}	4 Exp_{1023}	2660	5320	증명됨
SRP	4	4 $Hash_{1023}$	3 $Hash_{160}$	3 Exp_{1023}	3 Exp_{1023}	3990	3990	증명안됨
AMP	4	5 $Hash_{160}$	4 $Hash_{160}$	2 Exp_{160}	2 $sExp_{(160,2)}$	416	500	증명안됨
PAK2-Z	3	1 $Hash_{1024}$ 3 $Hash_{160}$ 1 $Hash_{x'}$	2 $Hash_{160}$ 1 $Hash_{x'}$	1 $sExp_{(160,2)}$ 2 Exp_{160}	2 Exp_{160}	666	416	증명됨

XTR)로 변형됨을 보였다. 이러한 프로토콜들은 실제적으로 모바일 장비나 스마트 카드와 같이 낮은 자원이 요구되는 환경에서 유용할 것이다.

이 절에서는 마지막으로 PAK2 프로토콜을 다른 패스워드 기반 인증 공유키 프로토콜들과 비교한다. 일반적으로, 패스워드 기반 인증 공유키 프로토콜은 패스워드의 직접 사용 또는 서버 공격에 대응하는 패스워드의 인증자의 사용에 따라 대칭형과 비대칭형의 2가지로 구분된다. 대칭형인 경우에는 PAK2를 SPEKE[20], KOY와 비교하고 비대칭형인 경우에는 PAK2-Z를 B-SPEKE[21], SRP[22], AMP[23]과 비교한다. 테이블4에서 사용된 비교방법은 3절에서 이용한 가정들을 사용했다.

〔표 4〕에서 알 수 있듯이, AMP 이외에 B-SPEKE, SRP와 비교하여 PAK-Z는 매우 적은 계산 복잡도를 갖는다. 하지만 PAK-Z는 3라운드이고 랜덤 오라클 모델에서 안전성을 제공해 주고 있지만 AMP는 4라운드이며 안전성에 관한 형식적인 증명을 아직까지 제시하지 못했다. 대칭인 경우에, PAK2는 매우 단순한 프로토콜인 SPEKE보다 훨씬 적은 계산량이 요구됨을 보여 주고 있다.

참 고 문 헌

〔1〕 P. MacKenzie, "The PAK suites : Protocols for Password-Authenticated Key Exchange", 2002.

〔2〕 P. MacKenzie, "More Efficient Password-Authenticated Key Exchange". In RSA Conference, Cryptographer's Track, pp. 361~377, 2001.

〔3〕 S. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks", In Proceedings of the Symposium on Security and Privacy, pp. 72~84 IEEE, 1992.

〔4〕 V. Boyko, P. MacKenzie and S. Patal, "Provably secure password-authenticated key exchange using Diffie-Hellman", In B. Prenel, editor, Advances in Cryptology Eurocrypt'00, Lecture Notes in Computer Science Vol. 1807, Springer-Verlag, pp. 156~171, 2000.

〔5〕 M. Bellare, D. Pointcheaval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks", Advances in Cryptology Eurocrypt'00, Lecture Notes in Computer Science, Vol. 1807, Springer-Verlag, pp. 139~155, 2000.

〔6〕 P. MacKenzie, S. Patal and R. Swami-

nathan, "Password-authenticated key exchange based on RSA", Advances in Cryptology Asiacrypt'00, Lecture Notes in Computer Science Vol. 1976, Springer-Verlag, pp. 599~613, 2000.

[7] J. Katz, R. Ostrovsky and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords", Advances in Cryptology Eurocrypt'01, Lecture Notes in Computer Science Vol. 2045, Springer-Verlag, pp. 475~494, 2001.

[8] O. Goldreich and Y. Lindell, "Session-key generation using human passwords only", In J. Killian editors, Advances in Cryptology, Crypto'01, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 408~432, 2001.

[9] O. Goldreich, S. Micali and A. Wigderson, "How to Play Any Mental Game, or a Completeness Theorem for protocols with an Honest Majority", STOC '87.

[10] S. Even, O. Goldreich and S. Micali, "On-Line/Off-Line Digital Signatures", Advances in Cryptology, Crypto'89, Lecture Notes in Computer Science Vol. 435, Springer-Verlag, pp. 263~277, 1998.

[11] R. Cramer and V. Shoup, "A Practical Public Key Cryptosystem Provably Secure Against Chosen Ciphertext Attack", Advances in Cryptology, Crypto'98, Lecture Notes in Computer Science Vol. 1462, Springer-Verlag, pp. 13~25, 1998.

[12] N. Koblitz, "Elliptic curve cryptosystems", Math. Comp., 48 pp. 203~209, 1987.

[13] V. Miller, "Use of elliptic curves in cryptography", Advances in Cryptology Crypto'85, Lecture Notes in Computer Science Vol. 218, Springer-Verlag, pp. 417~426, 1986.

[14] I. F. Blake, G. Seroussi and N. P. Smart, "Elliptic Curves in Cryptography", LMS Lecture Note Series 265, CUP, 1999.

[15] R. P. Gallant, J. L. Lambert, and S. A. Vanstone, "Faster Point Multiplication in Cryptology", Advances in Cryptology, Crypto'01, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 190~200, 2001.

[16] A. Lenstra and E. Verheul, "The XTR public key system", Advances in Cryptology Crypto'00, Lecture Notes in Computer Science Vol. 1807, Springer-Verlag, pp. 156~171, 2000.

[17] A. Lenstra and E. Verheul, "Key improvements to XTR", Advances in Cryptology Asiacrypt'00, Lecture Notes in Computer Science Vol. 1807, Springer-Verlag, pp. 156~171, 2000.

[18] D. E. Knuth, "The art of computer programming", Volumn 2, Seminumerical Algorithms, second edition, Addison-Wesley, 1981.

[19] A. Lenstra and E. Verheul, "Key improvements to XTR", Advances in Cryptology Asiacrypt'00, Lecture Notes in Computer Science, Vol. 1807, Springer-Verlag, pp. 156~171, 2000.

[20] D. Jablon, "Strong Password-Only Authenticated Key Exchange", Computer Communication Review, ACM SIGCOMM, Vol. 26, No. 5, pp. 5~26, October 1996.

[21] D. Jablon, Extended Password Key Exchange Protocols Immune to Dictionary Attacks, Proceedings of the Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '97) IEEE Computer Society, June 18~20, 1997, Cambridge, MA, pp. 248~255.

[22] T. Wu, The Secure Remote Password Protocol , Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium, San Diego, March 1998, pp. 97~111.

[23] T. Kwon, submission to P1363, July 23, 2000.

〈著者紹介〉

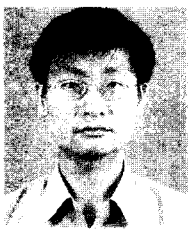
황 정 연 (Jung Yeon Hwang) 학생회원
1999년 2월 : 고려대학교 수학과 학사
2001년 3월~현재 : 고려대학교 정보보호대학원 석사과정
〈관심분야〉 공개키 암호 알고리즘, 암호 프로토콜

홍 석 희 (Seok Hie Hong) 정회원
1995년 2월 : 고려대학교 수학과 학사
1997년 2월 : 고려대학교 수학과 석사
2001년 2월 : 고려대학교 수학과 박사
〈관심분야〉 정보보호, 암호 알고리즘, 비밀키 암호 설계 및 분석, 패스워드 기반 프로토콜

장 상 운 (Sang Woon Jang) 학생회원
2002년 2월 : 고려대학교 수학과 학사
2002년 3월~현재 : 고려대학교 정보보호대학원 석사과정
〈관심분야〉 공개키 암호 알고리즘, 암호 프로토콜

박 혜 영 (Hye-Young Park) 학생회원
2001년 2월 : 고려대학교 수학과 학사
2001년 3월~현재 : 고려대학교 정보보호대학원 석사과정
〈관심분야〉 정수론, 공개키 암호, 암호 프로토콜

박 영 호(Young-Ho Park) 정회원
1990년 2월 : 고려대학교 수학과 학사
1993년 2월 : 고려대학교 수학과 석사
1997년 2월 : 고려대학교 수학과 박사
2002년 3월~현재 : 세종 사이버 대학교 조교수
〈관심분야〉 정수론, 공개키 암호, 암호 프로토콜



류 희 수 (Heuisu Ryu) 정회원
1990년 2월 : 고려대학교 수학과 학사
1992년 2월 : 고려대학교 수학과 석사
1999년 5월 : Johns Hopkins University 수학과 박사
2000년 7월~현재 : 한국전자통신연구원
〈관심분야〉 정보보호, 타원곡선 암호, 이동통신 보안