論文 98-23-7-23

# 패스워드 기반 인증 프로토콜 K1P의 확장

정회원 권 태 경*, 송 주 석*

# Extending the Password-based Authentication Protocol K1P

Taekyoung Kwon*, Jooseok Song* *Regular Members*

요 약

본 논문에서는 선행 논문인 [1]과 [2]에서 제안한 바 있는 패스워드 기반 인증 프로토콜 K1P의 특징을 요약하고 세가지의 확장된 프로토콜을 제안한다. 제안된 프로토콜들은 K1P의 설계 개념인 안전성과 효율성을 유지하도록 하였으며, 각기 다양한 목적에 따라 사용될 수 있도록 하였다. 새롭게 제안된 프로토콜은 일시 키-K1P, 사용자 공개키-K1P, 그리고 지수적 키 교환-K1P 등이다.

ABSTRACT

We summarize the password-based authentication protocol K1P which was introduced in our earlier papers [1,2] and then propose three more extended protocols. These protocols preserve a design concept of K1P, i.e., security and efficiency, and can be used for various purposes. They are a One-time key K1P, a Client public key K1P, and an Exponential key exchange K1P.

## I. 서 론

컴퓨터 통신망 환경에서 사용자의 신분을 확인하기 위한 인증(authentication) 기술은 정보보호 연구의 매우 중요한 분야이다. 특히 인증은 통신망에서의 불법적인 신분 위장을 다양한 암호학적 기반 기술을 이용하여 막을 수 있도록 하기 위한 것으로 반드시 필요하다. 이와 같은 관점에서 암호화 알고리즘을 기반으로 하는 인증 프로토콜은 통신망 사용자의 신분 확인과 불법 위장 방지를 위해서 필요한 메시지와 절차를 정의하는 규약이라고 할 수 있겠다. 실제로 인증 프로토콜은 1978년에 Needham-Shroeder 프로토콜[4]이 처음으로 제안된 이래 Denning-Sacco 프로토콜[5], Otway-Rees 프로토콜[6] 등을 비롯하여 무척 다양한 프로토콜들이 제안되어 여러 컴퓨터 시스템 환경에서 적용되고 있다. 이와 같은 인증 프로토콜은 통신 참여자간에 공유하고 있는 비밀값을 통해서 참여자의 신분을 확인토록 하며, 시간스탬프(time-stamp)나 논스(nonce)값을 이용하여 인증 메시지의 유효성을 인정하게 된다. 따라서 참여자간의 공유 비밀은 반드시 안전하게 보호될 수 있는 값이어야 하며 인증 프로토콜 설계시 공유 비밀의 안전을 가정하는 것이 일반적이었다. 그러나 사용자가 임의로 선택할 수 있는 패스워드(password)가 이와 같은 공유 비밀로 사용될 경우, 이 값은 추측되기 쉽다는 문제점을 갖는다[7, 8].

* 연세대학교 컴퓨터과학과
論文番號 : 98084-0226
接受日字 : 1998年 2月 26日

패스워드가 추측되기 쉬운 이유를 한마디로 요약하면, 선택시 키보드나 키패드가 사용되고 사용자가 일반적으로 기억하기 쉬운 문자열을 선택하기 때문에 패스워드의 실제 비트열이 갖는 엔트로피가 극도로 낮아지기 때문이다. 따라서 이와 같이 추측되기 쉬운 공유 비밀을 약한 비밀(weak secret)이라고 한다. 결국 이와 같이 약한 비밀값이 공유 비밀로 이용될 경우 인증 프로토콜은 패스워드 추측 공격(password guessing attack)에 노출된다[1, 2, 10, 12]. 인증 프로토콜에서의 패스워드 추측 공격에 대해서는 [1], [2], [10], [12], [16] 등에서 자세히 다루고 있으며, 기본적인 패스워드 인증의 안전성 문제에 대해서는 [7]과 [8]에서 자세하게 다루고 있다.

실제로 패스워드 추측 공격을 막기 위한 프로토콜로서 패스워드 기반 인증 프로토콜이 1989년에 발표된 LGSN 프로토콜[9] 이래 다양하게 제안되고 있다. 특히 1992년에는 Bellovin과 Merrit가 EKE(Encryption Key Exchange) 프로토콜을 제안하였으며[11], LGSN 프로토콜을 제안한 바 있는 Gong 등은 자신들의 결과를 더욱 확장하여 GLNS 프로토콜을 1993년에 발표하였다[12]. 그러나 이와 같은 패스워드 기반 프로토콜들은 인증에 필요한 난수(random value)의 생성 횟수, 메시지의 크기, 암호화 횟수, 프로토콜의 단계 횟수 등에 있어서 기존의 인증 프로토콜보다 더 많은 계산량과 통신량을 요구하게 된다는 단점이 있다. 따라서 이러한 문제를 해결하기 위한 노력도 시도된 바 있다.

1993년에 Tsudik과 Van Herreweghen이 LGSN 프로토콜에서 암호화 횟수를 크게 줄인 효율적인 LGSN 프로토콜을 제안하였으며[13], 1995년에는 Steiner, Tsudik, Waidner 등이 프로토콜의 단계를 줄인 개선된 EKE 프로토콜을 3자 EKE(three party EKE) 프로토콜과 함께 제안하였다[14]. 또한 같은 해에 Gong은 GLNS 프로토콜에서 프로토콜의 단계와 라운드 수를 줄인 Optimal 프로토콜을 제안하였다[15]. 그러나 이와 같이 오버헤드를 줄이는 과정에서 [13]과 [14]의 프로토콜들은 발견할 수 없는 온라인 추측 공격(undetectable on-line guessing attack)에 노출된다는 연구결과[16]가 발표되었으며, [15]의 프로토콜들은 난수의 생성 횟수, 암호화 횟수 등에서 오히려 오버헤드가 더 커진다는 단점을 갖는다[1]. 따라서 1997년에는 이와 같은 패스워드 기반 프로토콜들에 비해서 크게 효율적인 한편, 발견할 수 없는 온라인 추측 공격에도 노출되지 않도록 설계된 안전한 패스워드 기반 프로토콜인 K1P(k-won protocol)를 권태경 등이 제안하였다[1, 2]. K1P는 일시 패드(onetime-pad)와 일방향 해쉬 함수(oneway hash function)를 적절히 사용하여 안전성과 효율성을 동시에 도모하였다. 선행 논문인 [1]에서는 K1P의 기본 개념을 제안하고 GNY 로직[17]을 이용한 형식적 분석과 효율성 평가를 하였고 [2]에서는 ISDN과 WWW에 대한 적용 모델을 제안하였다.

본 논문에서는 패스워드 기반 인증 프로토콜 K1P에 대해서 살펴본 후 직접 인증 K1P를 기반으로 확장된 세 종류의 프로토콜을 제안한다. 먼저 II장에서는 K1P 프로토콜의 특징을 요약한 후 그 구조를 살펴본다. III장에서는 확장된 K1P를 제안하고 IV장에서는 프로토콜을 평가한다. 그리고 V장에서 결론을 맺는다.

## II. K1P 프로토콜의 개요

본 장에서는 패스워드 기반 인증 프로토콜인 K1P

표 1. 표기법 요약

| | | | |
|---|---|---|---|
| $A$, $B$ | 통신 참가자의 시스템 주체 | $S$ | 서버의 시스템 주체 |
| $na1$ | $A$가 생성한 첫 난수, 즉 논스값 | $P_A$ | $A$가 선택한 패스워드 |
| $K_A$ | $A$의 공개키 | $K_S$ | $S$의 공개키 |
| $K$ | 세션키 | $\{M\}_K$ | 메시지 $M$을 암호키 $K$로 암호화한 결과 |
| $h(M)$ | 메시지 $M$의 일방향 해쉬값 | $f(M)$ | 미리 정의된 간단한 함수 (예) $f(M)=M+1$ |
| $\|X\|$ | $X$의 비트 길이 | | |
| $k(M)$ | 입력값 $M$으로부터 정해진 크기의 세션키를 생성하기 위한 키 변환 함수 | | |
| $F_A$ | $A$가 전송한 메시지의 부분값 | $\oplus$ | XOR 연산자 |
| , | concatenator | $A \rightarrow B: X$ | $A$가 메시지 $X$를 $B$에게 전송 |

의 특징 및 구조에 대해서 고찰하도록 한다. 먼저 프로토콜 기술을 위해서 사용될 표기법을 요약하면 표 1과 같다.

### 2.1 K1P의 설계 개념 및 특징

K1P의 기본적인 설계 개념은 일시 패드와 강한 성질의 일방향 해쉬 함수를 사용하여 패스워드 기반 인증시의 안전성과 효율성을 도모하도록 하는 것이다 [1, 2]. 참고 문헌에서 설계 개념과 구조를 살펴 보면, [1]에서는 프로토콜 설계의 동기와 방법, 검증에 대해서 자세히 다루고 있으며 [2]에서는 프로토콜 설계의 요구 사항과 그 응용을 중점적으로 다루고 있다. 이것을 요약하면 다음과 같다.

(1) 일시 패드의 생성

K1P에서 인증을 얻고자 하는 주체는 먼저 난수 두 개를 생성한 후, 이 두 값을 XOR 연산하여 일시 패드를 생성한다. 이 난수들은 프로토콜의 논스값에 해당한다. 예를 들면 주체 $A$는 임의로 $na1$과 $na2$를 생성하고 일시 패드 $na1 \oplus na2$를 생성하게 된다. $na1$과 $na2$는 서로에 대해서도 일시 패드의 역할을 하게 되므로 그 크기가 같아야($|na1| = |na2|$)하며 또한 충분히 커야 한다. 정보 암호화를 위한 일시 패드 개념의 안전성은 이미 널리 알려져있다[18]. 이 일시 패드는 실제로 세션키를 안전하게 분배하기 위한 효율적인 암호화 기법으로 이용되므로 반드시 그 크기가 암호화 대상인 세션키 $K$와 같거나 커야 ($|na1 \oplus na2| \geq |K|$) 한다.

(2) 논스값의 분배

논스값은 일시 패드의 생성뿐만 아니라 메시지의 유효성을 증명하는데도 사용된다. 따라서 이 논스값은 프로토콜 참여자 사이에 안전하게 분배될 수 있어야 하며 특히 검증가능문 공격(verifiable-text attack) [10]에 대해서 안전하도록 분배될 수 있어야 한다. 따라서 공개키 암호화 기법을 이용하여 상대방의 공개키로 암호화한 논스값을 상대방에게 전달하도록 한다. 즉, 논스값을 평문 형태로 분배하지 않을 뿐만 아니라 약한 공유 비밀로 암호화하지도 않는다. 이 때 공개키는 사전에 안전하게 공개되었다고 가정한다.

(3) 패스워드 기반 인증

K1P의 인증은 패스워드를 통해서 이루어지므로 패스워드가 추측 공격에 노출되지 않도록 메시지를 구성한다. 먼저 공개키로 암호화되는 인증 요구 메시지에서는 패스워드 $P_A$를 마스킹하여 전송한다. 즉, 패스워드를 랜덤값 하나와 XOR 연산한 결과를 메시지에 포함하도록 한다. 따라서 인증을 위해서는 $\{na1, na2, P_A \oplus na1\}_{K_s}$과 같은 메시지가 전달될 수 있으며, 이 메시지를 복호화한 후 마스킹된 패스워드 $P_A$를 복원하여 인증을 이룰 수 있다.

(4) 인증에 대한 확인 및 서버 인증

K1P에서는 인증에 대한 확인을 해쉬 함수를 통해서 이루게 된다. 즉, 인증된 경우 인증 서버는 메시지 $na1 \oplus na2$, $h(P_A \oplus na1, na2)$를 전송하여 참여자가 자신의 인증에 대한 확인을 할 수 있도록 한다. 한편, 서버에 대한 인증은 서버의 공개키에 대한 신뢰 및 인증 확인 메시지를 통해서 이룰 수 있다. 즉, 서버가 공개키로 암호화된 메시지를 복호화 했다는 사실과 패스워드를 공유하고 있다는 사실을 통해서 서버를 인증하게 된다.

(5) 세션키의 분배

세션키는 일시 패드로 암호화하여 분배되며, 키의 무결성 검사에는 해쉬값이 이용된다. 즉, 메시지 $na1 \oplus na2 \oplus K$, $h(P_A \oplus na1, K, na2)$로 세션키를 분배하고 키의 무결성을 확인한다.

(6) 세션키 분배에 대한 확인

세션키는 분배 사실에 대한 확인이 확실하게 이루어져야만 그 가치를 발휘할 수 있다. 따라서 해쉬 값만으로 구성된 메시지 $h(P_A \oplus na2, K, na1)$가 세션키를 생성한 인증 서버에게 전달되어야만 그 세션키가 암호화를 위해서 사용될 수 있으며 비로소 모든 인증 절차를 마치게 된다.

### 2.2 K1P의 프로토콜 구조

기본적으로 K1P는 두 주체간의 직접적인 인증을 위한 직접 인증 프로토콜과 두 주체가 제 3자를 통해서 간접적인 인증을 하게 되는 상호 인증 프로토콜로 구성된다[1]. 두 프로토콜 모두 2.1절에서 요약한 설

계 개념과 특징에 근간하고 있다. 본 절에서는 K1P의 구조에 대해서 간략히 설명하도록 한다. 아래의 프로토콜들은 선행 논문 [1]에서 제안된 K1P를 조금씩 개선한 결과이며, 그 개선점에 대해서도 각각 설명하도록 한다.

(1) 직접 인증 프로토콜

K1P 직접 인증 프로토콜은 패스워드를 기반으로 하는 클라이언트-서버 환경에서 적용하기에 적합하며 다음과 같이 두가지 종류가 있다.

(i)

$$\begin{aligned}&1.\ A \rightarrow S:\ A,\ \{na1, na2, P_A \oplus na1\}_{K_s}\\&2.\ S \rightarrow A:\ na1 \oplus na2 \oplus K,\ h(P_A \oplus na1, K, na2)\\&3.\ A \rightarrow S:\ h(P_A \oplus na2, K, na1)\end{aligned} \tag{2.1-i}$$

(ii)

$$\begin{aligned}&1.\ A \rightarrow S:\ A,\ \{na, P_A \oplus na\}_{K_s}\\&2.\ S \rightarrow A:\ na \oplus ns,\ h(P_A \oplus na, ns)\\&3.\ A \rightarrow S:\ h(P_A \oplus ns, K, na)\end{aligned}$$

$$\begin{aligned}&(a)\ K = k(h(na, ns))\\&(b)\ K = k(g^{xy} \bmod n)\\&\quad (na = g^x \bmod n,\ ns = g^y \bmod n)\end{aligned} \tag{2.1-ii}$$

이 프로토콜의 수행 절차에 대한 설명 및 안전성과 효율성에 대한 분석은 선행 논문 [1]에서 자세하게 다루고 있다. 실제로 프로토콜 (2.1-i)에서는 패스워드 $P_A$를 통해서 $S$가 $A$를 쉽게 인증하고 세션키 $K$를 안전하게 분배할 수 있으며, 패스워드 추측 공격에 대해서 안전하다. 프로토콜 (2.1-ii)에서는 해쉬 함수나 Diffie-Hellman 방식[3], 즉 지수적 키 교환(exponential key exchange) 방법을 통해서 $A$와 $S$가 서로 합의하에 세션키를 공동으로 생성할 수 있다.

참고 문헌 [1]에서와의 차이점은 패스워드에 대한 표기법이 바뀌었다는 점과 (2.1-ii) 프로토콜에서 세션키의 크기 변환을 위한 함수 $k()$가 추가되었다는 점이다. $k()$는 암호화를 위한 알고리즘에 따라 알맞은 크기의 세션키를 생성하도록 하기 위한 함수로서, 입력값의 크기가 해당 알고리즘의 암호키 크기에 적합할 경우에는 아무 연산도 하지않고 입력값을 그대로 출력하도록 한다.

(2) 상호 인증 프로토콜

K1P 상호 인증 프로토콜은 패스워드를 기반으로 하고 안전성 센터의 설치가 가능한 통신망 환경에서 적용하기에 적합하며 다음과 같이 구성된다.

$$\begin{aligned}&1.\ A \rightarrow B:\ \{A, B, na1, na2, P_A \oplus na1\}_{K_s}\\&2.\ B \rightarrow S:\ \{A, B, na1, na2, P_A \oplus na1\}_{K_s},\\&\quad\ \{B, A, nb1, nb2, P_B \oplus nb1\}_{K_s}\\&3.\ S \rightarrow B:\ na1 \oplus na2 \oplus K, h(P_A \oplus na1, K, na2, A \oplus B),\\&\quad\ nb1 \oplus nb2 \oplus K, h(P_B \oplus nb1, K, nb2, A \oplus B)\\&4.\ B \rightarrow A:\ na1 \oplus na2 \oplus K, h(P_A \oplus na1, K, na2, A \oplus B),\\&\quad\ \{f(F_A), nb1 \oplus nb2\}_K\\&5.\ A \rightarrow B:\ h(f(nb1 \oplus nb2) \oplus K)\end{aligned} \tag{2.2}$$

이 프로토콜의 수행 절차에 대한 설명 및 안전성과 효율성에 대한 분석 역시 논문 [1]에서 자세하게 다루고 있다. 이 프로토콜의 특징은 메시지 1의 부분값을 정해진 함수로 처리한 결과 $f(F_A)$와 이미 구성한 일시 패드 $nb1 \oplus nb2$를 세션키 $K$로 암호화하여 세션키 분배 확인을 위한 도전-응답(challenge-response)값으로 이용한다는 점이다[1].

참고 문헌 [1]에서의 개선점은 메시지 5에서 세션키 $K$에 대한 확인을 위해서 암호화하는 대신 해쉬 함수를 사용하도록 했다는 점이다. 따라서 해쉬 함수 연산이 1회 늘어나는 반면 암호화 연산이 1회 줄어든다는 장단점이 있게 되며, 결국 해쉬 함수 알고리즘과 암호화 알고리즘의 성질에 따라 효율적인 쪽을 선택하는 것이 바람직하다고 할 수 있다.

## III. 확장된 K1P의 설계

K1P의 직접 인증 프로토콜은 클라이언트-서버 환경에서 패스워드를 통한 인증을 수행하기에 적합한 프로토콜로서 두 개체간에 공유하고 있는 패스워드를 이용해서 안전한 인증을 이룰 수 있도록 한다. 그러나 프로토콜의 구조상 다음과 같은 제한을 갖는다.

(a) 반드시 프로토콜에서 사용되는 논스값의 크기가 세션키의 크기보다 같거나 커야 한다.

(b) 반드시 서버측의 공개키가 사용되어야 한다.

따라서 본 절에서는 이와 같은 제한을 극복하도록 K1P의 직접 인증 프로토콜을 확장하여 보다 다양한 환경에서 적용될 수 있는 세가지 종류의 확장된 프로토콜을 제안한다. 각 프로토콜은 그 구조와 안전성을

상호 인증 K1P에 근간하고 있다. 따라서 선행 논문 [1]에서 상호 인증 K1P에 대한 형식적 분석을 GNY 로직을 통하여 수행하였기 때문에 본 논문에서는 형식적 분석을 생략하도록 하며, 패스워드 추측 공격에 대한 안전성을 중점적으로 설명하도록 한다.

### 3.1 일시 키-K1P

일시 패드가 갖는 특징중 하나는 평문과 암호키, 그리고 암호문의 크기가 같다는 점이다. 일시 패드를 사용하는 직접 인증 K1P가 (a)와 같은 제한을 갖게 되는 이유는 세션키를 암호화하는 일시 패드가 메시지 1에 포함된 두 논스값을 XOR 연산한 결과이기 때문이다. 따라서 사용하는 암호화 알고리즘의 암호키 크기에 따라 프로토콜의 논스값 크기가 영향을 받게 되는 것이다. 예를 들어서 [표 2]와 같이 암호키의 크기가 256 비트인 GOST[18] 알고리즘과 160 비트의 해쉬값을 만드는 SHA[18] 알고리즘을 사용할 경우, 이에 따라 논스값의 크기도 각각 256 비트씩으로 지나치게 커지게 되며 메시지 2의 크기도 416 비트로 역시 커지게 된다.

표 2. 논스값과 메시지 크기 비교

| | 논스값의 크기 | 메시지 2의 크기 |
|---|---|---|
| 직접 인증 K1P (2.1-i) | 256 비트 | 416 비트 |
| 일시 키-K1P (3.1) | 64(가변) 비트 | 320(가변) 비트 |

(암호키 256 비트, 해쉬값 160 비트)

그러나 해쉬값 $h(P_A \oplus na1, na2)$를 일시 키로 사용하면 논스값의 크기 뿐만 아니라 메시지 2의 크기도 줄일 수 있다. 이 예에서도 논스값은 64비트 정도의 적당한 크기로 줄일 수 있으며, 이 때 메시지 2의 크기는 320 비트가 된다. 해쉬 값은 임의성과 일방향성을 가지므로 키로 사용될 경우에도 여전히 큰 정보량을 갖는다. 특히 이 키는 인증 결과의 검증과 키 분배를 위한 짧은 시간 동안에만 일시적으로 사용되므로 안전하다. K1P에서의 일시 키 사용에 대해서는 선행 논문 [1]과 [2]에서도 잠시 언급한 바 있다. 일시 키를 사용하여 확장된 일시 키-K1P는 다음과 같다.

$$\begin{aligned} &1.\ A \rightarrow S:\ A, \{na1, na2, P_A \oplus na1\}_{K_S} \\ &2.\ S \rightarrow A:\ \{na1, K\}_{h(P_A \oplus na1, na2)} \\ &3.\ A \rightarrow S:\ h(P_A \oplus na2, K, na1) \end{aligned} \tag{3.1}$$

이 프로토콜은 프로토콜 (2.1-i)와 같은 구조를 갖지만 단계 2에서 일시 패드 대신 일시 키로 암호화하여 키를 분배하는 차이가 있다. 프로토콜 (2.1-i)에서는 해쉬값 $h(K_A \oplus na1, K, na2)$에 새로운 세션키 $K$가 포함되었지만 프로토콜 (4.1)에서 일시키로 사용되는 $h(K_A \oplus na1, na2)$에는 세션키 $K$가 포함되지 않는다. 메시지 2에서 논스값 $na1$은 메시지의 무결성과 유효성을 위한 논스값으로 사용된다. 이 프로토콜의 안전성은 같은 구조를 갖는 근간 프로토콜 (2.1-i)와 같다.

### 3.2 사용자 공개키-K1P

직접 인증 K1P는 프로토콜의 구조상 단계 1에서 클라이언트에 해당하는 $A$가 서버에 해당하는 $S$의 공개키로 인증 요구 메시지를 암호화하기 때문에 (b)와 같은 제한을 갖는다. 결국 $S$의 공개키가 알려지지 않은 경우에는 프로토콜을 수행할 수 없으며 $S$의 공개키 갱신시 부담이 크다. 한편 $A$는 반드시 $S$의 공개키를 보관하거나 공개키 디렉토리를 참조해야 한다. 이와 같은 제한을 극복하기 위해서 클라이언트측인 사용자 $A$의 공개키를 사용하도록 K1P를 확장할 수 있다. 여기서 $A$의 공개키는 언제든지 임의로 선택되어질 수 있으며 따라서 공개키 갱신의 부담도 없을 뿐만 아니라 공개키 디렉토리에서 관리될 필요가 없다. 사용자 공개키-K1P는 다음과 같다.

$$\begin{aligned} &1.\ A \rightarrow S:\ A, \{K_A\}_{P_A} \\ &2.\ S \rightarrow A:\ \{ns, K, P_A \oplus ns\}_{K_A} \\ &3.\ A \rightarrow S:\ h(P_A \oplus ns, K, ns) \end{aligned} \tag{3.2}$$

단계 1에서 사용자 $A$는 임의의 공개키 $K_A$를 선택한 후 이것을 패스워드로 암호화한 메시지 1을 서버 $S$에게 전송한다.

단계 2에서는 $S$가 논스값 $ns$와 세션키 $K$를 생성한 후 $A$의 공개키 $K_A$로 암호화하여 응답한다. $A$는 이 메시지를 복호화한 후 $ns$와 마스크 값 $P_A \oplus ns$에서 패스워드 $P_A$가 정상적으로 나올 경우 $S$를 인증하며 세션키 $K$의 유효성을 확인한다.

단계 3에서는 $A$가 해쉬값을 계산하여 $S$에게 응답한다. $S$ 역시 자신이 갖고 있는 값들로 같은 해쉬

연산을 한 후 전송된 해쉬값과 자신이 계산한 해쉬값을 비교한다. 두 값이 일치할 경우 $S$는 $A$를 인증할 수 있으며, 또한 $A$가 세션키 $K$를 받았음을 확인할 수 있다.

이 프로토콜의 각 메시지는 근간 프로토콜인 직접 인증 K1P의 메시지 구조를 따르고 있다. 또한 메시지 2에서는 공개키의 유효성을, 그리고 메시지 3에서는 논스값 $ns$의 유효성을 통해서 각 메시지의 유효성이 확인되며 따라서 메시지 재전송에 대해서 안전하다. 패스워드를 추측한 공격자가 메시지 1을 추측한 패스워드로 복호화하더라도 메시지 2와 3에서 키와 패스워드에 대한 확인이 불가능하므로 이 프로토콜은 역시 안전한 패스워드 기반 인증 프로토콜이다.

### 3.3 지수적 키 교환-K1P

참고 문헌 [3]의 지수적 키 교환 방식은 인증 기능을 갖고 있지 않지만, 프로토콜 참여자가 함께 키를 생성할 수 있다는 장점이 있다. 따라서 이 방식은 직접 인증 K1P[1], EKE[11] 등에서도 응용된 바 있다. 그러나 직접 인증 K1P (2.1-ii)에서는 공개키 암호 시스템이 사용되어야 한다는 제한이 있다. 따라서 본 절에서는 공개키 암호 시스템을 사용하지 않도록 확장된 K1P를 제안한다. 이 프로토콜은 Diffie-Hellman 방식[3]의 이산 대수 문제에 근간하여 키를 생성하도록 한다. 따라서 이산대수 문제 기반의 지수적 키 교환 방식에 안전성을 근간하므로 지수적 키 교환-K1P라고 하며, 3.2절에서 제안한 사용자 공개키-K1P의 구조를 바탕으로 다음과 같이 설계된다.

$$
\begin{aligned}
&1.\ A \rightarrow S:\ A,\ \{na\}_{P_A}\\
&2.\ S \rightarrow A:\ \{ns\}_{P_A},\ h(na,\ na^{y} \bmod n)\\
&3.\ A \rightarrow S:\ h(ns^{x} \bmod n,\ ns)\\
&(*)\ \ K = k(h(g^{xy} \bmod n))\\
&\qquad (na = g^{x} \bmod n,\ ns = g^{y} \bmod n)\\
&\qquad (na^{y} \equiv ns^{x} \equiv g^{xy} \pmod n)
\end{aligned}
\tag{3.3}
$$

단계 1에서 $A$는 임의의 비밀값 $x$를 선택한 후 공개값인 $g$와 $n$을 이용하여 지수 계산을 한다. 그리고 그 결과 $na(=g^{x} \bmod n)$를 패스워드로 암호화하여 $S$에게 전송한다.

단계 2에서는 $S$ 역시 임의로 선택한 비밀값 $y$의 지수계산결과 $ns(=g^{y} \bmod n)$를 $A$의 패스워드로 암호화한다. 그리고 메시지 1을 복호화한 결과인 $na$와 함께 이 값과 자신의 비밀값 $y$의 지수계산 결과인 $na^{y} \bmod n$을 입력값으로 하여 해쉬값을 계산한다. $S$는 이 두가지 메시지 블록으로 $A$에게 응답한다.

$A$는 메시지 2에서 복호화한 값 $ns$와 자신의 비밀값 $x$의 지수계산 결과 $ns^{x} \bmod n$과 함께 자신이 단계 1에서 생성한 $na$를 입력으로 한 해쉬값 $h(na,\ ns^{x} \bmod n)$을 메시지 2의 해쉬값 $h(na,\ na^{y} \bmod n)$과 비교한다. 두 값이 일치하면 $A$는 $S$가 $na$를 그리고 자신이 $ns$를 정상적으로 복호화했음을 확인할 수 있다. 다시 말해서 두 해쉬값이 일치하기 위해서는 $na^{y} \equiv ns^{x} \equiv g^{xy} \pmod n$의 조건을 반드시 만족해야 하므로, 두 값의 비교에 따라 $A$는 $S$를 인증하는 한편 서로 공통된 값 $g^{xy} \bmod n$을 공유하고 있음을 확신할 수 있다.

단계 3에서는 $A$가 메시지 2에서 얻은 값 $ns$와 함께 이 값과 자신의 비밀값 $x$의 지수계산 결과인 $ns^{x} \bmod n$를 입력로 한 해쉬값 $h(ns^{x} \bmod n,\ ns)$를 전송한다. $S$는 이 값을 앞서 계산한 $na^{y} \bmod n$과 $ns$의 해쉬값인 $h(na^{y} \bmod n,\ ns)$와 비교한다. 역시 두 해쉬값이 일치하면 $S$는 패스워드로 복호화한 값 $na$의 유효성 확인을 통해서 $A$를 인증하게 되며, 또한 서로 공통된 값 $g^{xy} \bmod n$을 공유하고 있음을 확신할 수 있다.

결국 $A$와 $S$는 공유하고 있는 값인 $g^{xy} \bmod n$으로부터 같은 세션키를 얻을 수 있으며 상대방이 같은 세션키를 생성하는 것을 확신할 수 있는데 이것은 서로 상대방과 $g^{xy} \bmod n$를 공유하고 있다는 사실을 프로토콜을 통해서 확인했기 때문이다.

이 프로토콜에서 특별히 주의할 점은 세션키 $K$를 생성하는 방법이다. 이 프로토콜에서는 세션키 $K$의 생성을 위해서 먼저 $g^{xy} \bmod n$의 일방향 해쉬값을 얻은 후 이 값을 키 변환 함수 $k()$에 입력하도록 한다. 이와 같이 공유 값인 $g^{xy} \bmod n$을 직접 키 변환 함수에 입력하지 않고 일방향 해쉬 함수를 한번 거치는 이유는, 세션키 $K$가 유출되었을 때의 오프라인 추측 공격을 막기 위해서이다. 그러나 이것은 키 변환 함수 $k()$에 대한 가정에 따라서 생략될 수도

있다. 즉, 키 변환 함수가 일방향 함수일 경우에는 해쉬 함수가 사용될 필요가 없지만 그렇지 않을 경우에는 필요하다.

예를 들어서, 키 변환 함수 $k()$가 일방향 함수가 아니고 또한 일방향 해쉬 함수가 사용되지 않았을 경우를 살펴보자. 먼저 공격자는 유출된 세션키가 만들어진 프로토콜 세션의 메시지를 이미 도청해서 저장했을 것이다. 이와 같이 세션키 $K$가 유출되었을 경우 공격자는 이 키에 $k()$의 역함수를 적용하여 $g^{xy} \bmod n$을 계산해낼 수 있다. 그리고 추측한 패스워드로 메시지 1과 메시지 2의 암호문을 복호화하여 $na'$와 $ns'$를 구한다. 그리고 $g^{xy} \bmod n$과 함께 복호화한 각각의 값을 입력으로 한 해쉬값을 구하여 도청한 메시지와 다음과 같이 비교할 수 있다.

$$\begin{aligned} &h(na', g^{xy} \bmod n) \quad \text{vs.} \quad h(na, na^{y} \bmod n) \\ &h(g^{xy} \bmod n, ns') \quad \text{vs.} \quad h(ns^{x} \bmod n, ns) \end{aligned} \tag{3.4}$$

만약 이 값들의 쌍이 각각 일치한다면 추측한 패스워드의 유효성이 인정되는 것이며, 결국 오프라인 패스워드 공격이 성공하게 되는 것이다. 그러나 키 변환 함수 $k()$가 일방향 함수이거나 일방향 해쉬 함수가 먼저 사용되었을 경우에는 공격자가 알려진 세션키로부터 $g^{xy} \bmod n$을 얻을 수 없으므로 이와 같은 공격에 대해서 안전하게 된다. 따라서 지수적 키 교환-K1P에서는 일방향 해쉬함수를 $g^{xy} \bmod n$에 적용한 결과로부터 세션키 $K$를 유도하도록 하였다. 이렇게 하면 $K$가 알려진 경우에도 $g^{xy} \bmod n$가 공개되지 않으며 따라서 해쉬값을 이용한 오프라인 추측 공격을 막을 수 있다. 또한 메시지 2와 3에서 인증이 가능하므로, 공개키 암호 시스템을 이용하지 않는 지수적 키 교환-K1P는 패스워드 추측 공격에 대해서 안전하다.

## IV. 확장된 K1P의 효율성 평가

본 장에서는 본 논문에서 제안한 확장된 K1P를 안전하다고 평가되는 다른 패스워드 기반 프로토콜들과 효율성면에서 비교하도록 한다. 이와 같이 안전한 프로토콜과의 선택 비교 사유는 참고문헌 [1]에서 설명하고 있다. 비교 항목은 프로토콜의 전체 단계수, 각 참여자의 난수 생성 횟수, 참여자간의 암호화 연산 횟수로 구분되며, 특히 암호화 연산은 공개키 암호화, 관용 암호화, 해쉬 함수 등의 세가지 연산에 대해서 비교하였다.

먼저 패스워드 기반의 직접 인증 프로토콜들을 비교한 결과는 [표 3]과 같다. 직접 인증, 일시 키, 사용자 공개키 등으로 분류되는 K1P는 모두 프로토콜 단계가 3단계인 반면, Gong의 Optimal direct 프로토콜을 제외한 나머지, 즉 Strengthened EKE와 GLNS nonce direct 프로토콜은 5단계로 이루어진다. 또한 난수 생성 횟수와 암호화 관련 연산 면에서도 K1P가 월등히 경제적이며, 전반적으로 K1P가 효율적이라는 것을 알 수 있다. 참고로 [표 3]의 난수 생성 횟수에서 $S$가 세션키를 생성하는 횟수나, 사용자 공개키-K1P, Strengthened EKE, 그리고 Optimal direct 프로토콜 등에서 $A$가 공개키를 생성하는 횟수는 포함하지 않았다.

표 3. 패스워드 기반의 직접 인증 프로토콜 비교

| 프로토콜 | 프로토콜 단계 | 난수 생성 횟수 | | 암호화 관련 연산 횟수 | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Pub. | Conv. | Hash. |
| 직접 인증 K1P-i [1, 2] | 3 | A | 2 | A-S | 1 | 0 | 2 |
| | | S | 0 | | | | |
| 일시 키-K1P (3.1) | 3 | A | 2 | A-S | 1 | 1 | 2 |
| | | S | 0 | | | | |
| 사용자 공개키-K1P (3.2) | 3 | A | 0 | A-S | 1 | 1 | 1 |
| | | S | 1 | | | | |
| Strengthened EKE [11] | 5 | A | 2 | A-S | 1 | 3 | 2 |
| | | S | 2 | | | | |
| GLNS nonce direct [12] | 5 | A | 1 | A-S | 1 | 5 | 0 |
| | | S | 4 | | | | |
| Gong Optimal direct [15] | 3 | A | 1 | A-S | 1 | 3 | 0 |
| | | S | 2 | | | | |

지수적 키 교환 방식을 이용하는 패스워드 기반 인증 프로토콜을 비교한 결과는 [표 4]와 같다. 직접 인증 K1P와 지수적 키 교환-K1P의 프로토콜 단계는 3단계인 반면 DH based EKE는 4단계로 이루어진다. 또한 난수 생성 횟수에서도 EKE가 참가자마다 각각 1회씩 더 많다. 암호화 관련 연산 횟수에서는 직접

인증 K1P가 공개키 암호화를 필요로 하는 반면, 지수적 키 교환-K1P와 EKE는 공개키 암호화를 사용하지 않는다는 특징이 있다. 그러나 이 두 프로토콜은 대칭키 암호화 횟수를 부가적으로 늘리게 되는데, 특히 EKE가 총 5회의 대칭키 암호화를 필요로 하는 반면, 확장된 K1P는 2회의 대칭키 암호화와 3회의 일방향 해쉬 계산을 필요로 한다. 전반적으로 K1P가 더욱 효율적인 것을 알 수 있다.

표 4. 패스워드 기반의 지수적 키 교환 인증 프로토콜 비교

| 프로토콜 | 프로토콜 단계 | 난수 생성 횟수 | | 암호화 관련 연산 횟수 | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Pub. | Conv. | Hash. |
| 지수적 키 교환-K1P (3.3) | 3 | A | 1 | A-S | 0 | 2 | 3 |
| | | S | 1 | | | | |
| 직접 인증 K1P-ii [1] | 3 | A | 1 | A-S | 1 | 0 | 2 |
| | | S | 1 | | | | |
| DH based EKE [11] | 4 | A | 2 | A-S | 0 | 5 | 0 |
| | | S | 2 | | | | |

이와 같이 패스워드 기반 프로토콜 K1P는 안전하다고 평가되는 기타 패스워드 기반 프로토콜과 비교할 때 월등히 효율적인 프로토콜이다. 또한 본 논문에서 확장한 K1P, 즉 일시 키, 사용자 공개키, 지수적 키 교환 등의 프로토콜들도 같은 종류의 타 프로토콜들에 비해서 기존의 K1P가 갖던 효율성의 비교적 우위를 그대로 유지하고 있다.

## V. 결 론

본 논문에서는 선행 논문인 [1]과 [2]에서 제안한 바 있는 패스워드 기반 인증 프로토콜 K1P의 특징을 분석 요약하고, 이것을 기반으로 하여 일시 키-K1P, 사용자 공개키-K1P, 지수적 키 교환-K1P 등의 세가지 확장된 프로토콜들을 제안하였다. 이 프로토콜들 역시 사용자가 임의로 선택한 패스워드를 통해서 안전한 인증을 가능하게 하는 패스워드 기반 인증 프로토콜로서 기존 K1P의 설계 개념인 안전성과 효율성을 더욱 다양화 시킨 것이다.

일시 키-K1P는 기존의 직접 인증 K1P가 갖던 제한점인 논스값과 세션키 크기의 관계를 해결하는 한편 메시지 2의 크기를 줄이는 특징을 갖고 있다. 사용자 공개키-K1P는 클라이언트-서버 모델로 프로토콜을 고려할 경우 기존의 직접 인증 K1P가 서버측의 공개키만을 요구했던 반면, 이 프로토콜은 클라이언트측의 공개키 사용을 가능하게 하므로써 K1P를 더욱 다양화 시켰다. 한편 지수적 키 교환-K1P는 공개키 알고리즘을 사용하지 않고 단지 Diffie-Hellman 방식, 즉 이산대수 문제에 근간한 지수적 키 교환 방식만을 사용한 패스워드 기반 인증 프로토콜이다. 따라서 기존의 직접 인증 K1P-ii가 패스워드 인증을 통한 지수적 키 교환을 위해서 공개키를 사용했던 것과 비교된다.

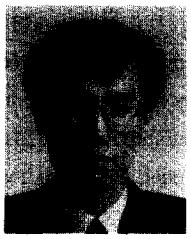
이미 선행 논문 [1]에서는 K1P에 대한 경험적인 분석과 GNY 로직[17]을 이용한 형식적인 분석을 하였으며, [2]에서는 ISDN과 WWW을 예로 K1P의 적용 모델을 제시하였다. 이와 같이 선행 논문 [1], [2]와 본 논문에서 제안된 패스워드 기반 인증 프로토콜인 K1P는 다양한 종류로 구성되어 있으며 패스워드를 통한 인증과 안전한 통신이 요구되는 다양한 환경에 적용될 수 있을 것이다.

## 참 고 문 헌


1. 권태경, 송주석, "추측 공격에 대해서 안전하고 효율적인 패스워드 기반 인증 프로토콜의 설계 및 검증," 한국정보과학회 논문지(A), 제24권 제8호, pp. 795-806, Aug. 1997.
2. T.K.Kwon, M.H.Kang, J.S.Song, "An Adaptable and Reliable Authentication Protocol for Communication Networks," Proceedings of IEEE INFOCOM'97, pp. 738-745, Apr. 1997.
3. W.Diffie, M.Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, vol. 22, no. 3, pp. 644-654, Nov. 1976.
4. R.Needham, M.Schroeder, "Using Encryption For Authentication in Large Networks of Computers," Communications of the ACM, vol. 21, no. 12, pp.993-999, Dec. 1978.
5. D.Denning, G.Sacco, "Timestamps in Key Distribution Protocols," Communications of ACM, vol. 24, no. 8, pp. 533-536, Aug. 1981.
6. D.Otway, O.Rees, "Efficient and Timely Mutual Authentication," ACM Operating Systems Review, vol. 21, no. 1, pp. 8-10, Jan. 1987.

7. R.Morris, K.Thompson, "Password Security: A Case History," Communications of the ACM, vol. 22, no. 11, pp. 594-587, Nov. 1979.
8. D.C.Feldmeier, P.R.Karn, "UNIX Password Security -Ten Years Later," Proceedings of Crypto'89, published as Lecture Notes in Computer Science, No. 435, pp. 44-63, 1989.
9. M.Lomas, L.Gong, J.Saltzer, R.Needham, "Reducing Risks from Poorly Chosen Keys," Proceedings of the 12th ACM Symposium on Operating System Principles, ACM Operating Systems Review, vol. 23, no. 5, pp. 14-18, Dec. 1989.
10. L.Gong, "Verifiable-text Attacks in Cryptographic Protocols," Proceedings of IEEE INFOCOM'90, pp. 686-693, June 1990.
11. S.Bellovin, M.Merrit, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," Proceedings of the IEEE Symposium on Security and Privacy, pp. 72-84, 1992.
12. L.Gong, M.Lomas, R.Needham, J.Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks," IEEE Journal on Selected Areas in Communications, vol. 11, no. 5, pp. 648-656, June 1993.
13. G.Tsudik, E.Van Herreweghen, "Some Remakrs on Protecting Weak Keys and Poorly-Chosen Secrets from Guessing Attacks," 1993 IEEE Symposium on Reliable Distributed Systems, pp. 136-142, 1993.
14. M.Steiner, G.Tsudik, M.Waidner, "Refinement and Extension of Encrypted Key Exchange," ACM Operating System Review, vol. 29, no. 3, pp. 22-30, 1995.
15. L.Gong, "Optimal Authentication Protocols Resistant to Password Guessing Attacks," Proceedings of the 8th IEEE Computer Security Foundations Workshop, pp. 24-29, June 1995.
16. Y.Ding, P.Horster, "Undetectable On-line Password Guessing Attacks," ACM Operating Systems Review, vol. 29, no. 4, pp. 77-86, Oct. 1995.
17. L.Gong, R.Needham, R.Yahalom, "Reasoning about Belief in Cryptographic Protocols," Proceedings of the IEEE Symposium on Research in Security and Privacy, pp.234-248, 1990.
18. B.Schneier, Applied Cryptography, 2nd Ed., John Wiley & Sons, 1996.

**권 태 경**(Taekyoung Kwon) 정회원
1970년생
1988년 3월~1992년 2월: 연세대학교 컴퓨터과학과 (이학사)
1993년 3월~1995년 2월: 연세대학교 컴퓨터과학과 (이학석사)
1995년 3월~현재: 연세대학교 컴퓨터과학과 박사과정 재학중
※주관심분야: 컴퓨터 통신망 보안, 암호학, ATM Traffic Management, PCS Mobility Management
e-mail : ktk@emerald.yonsei.ac.kr



**송 주 석**(Jooseok Song) 정회원
1953년생
1976년 2월: 서울대학교 전기공학과 졸업(공학사)
1979년 2월: 한국과학원 전기 및 전자공학과 졸업(공학석사)
1979년 2월~1982년 2월: 한국전자통신 연구소 전임연구원
1988년 8월: University of California at Berkeley 전산과학과 졸업(공학박사)
1988년 9월~1989년 2월: Naval Postgraduate School Information System Department 조교수
1989년 3월~현재: 연세대학교 컴퓨터과학과 교수
※주관심분야: 프로토콜 엔지니어링, 통신망 보안, PCS
e-mail : jssong@emerald.yonsei.ac.kr