2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

안전성을 위한 JOTP(Java One Time Password) 보안 알고리즘의 설계 및 구현

윤세미*, 조익성*, 임재홍*
*한국해양대학교 전자통신공학과
semihi@hanmail.net

Design and Implementation of an Extended JOTP Security Algorithm for the Safety

Se-Mi Yoon*, Ik-Sung Cho*, Jae-Hong Yim*
*Dept. of Electronics & Communication Eng.,
Korea Maritime University

요약

본 논문은 클라이언트가 서버에 로그인을 요청할 때 사용자를 인증하기 위해 사용되는 패스워드를 매번 생성하여 한번 사용한 패스워드는 다시 사용하지 않는 안전한 JOTP 알고리즘의 설계 및 구현에 대해 설명한다. 확장된 JOTP에서는 클라이언트가 난수 발생에 의해 생성한 패스워드로 로그인할 때 사용하고, 패스워드의 저장과 관리가 클라이언트 측에서 이루어져 서버의 침입 또는 패스워드의 도난에 대해서도 안전하다. 또 웹 기반의 관리자 인터페이스와 클라이언트가 서버에 접속하기 위한 절차가 기존의 OTP 알고리즘보다 간단하다는 특징을 가진다.

1. 서론

오늘날 인터넷은 수천의 네트워크와 수만의 컴퓨터를 상호 연결하는 수단으로 거대한 하나의 단일화된 네트워크를 형성하여 정보의 교환을 자유롭게 하고 있다. 인터넷의 발달은 정보의 공유와 교환을 자유롭게 하는 이점이 있지만 사용권한이 없는 사용자가 정보를 악용하거나, 공개되어서는 안 되는 정보가 침해당할 수도 있다는 취약점을 가진다. 이러한 네트워크 환경에서의 약점을 보완하기 위해서는 컴퓨터 시스템과 정보통신망 전반에 대한 안전성과 신뢰성 확보가 선행되어야 한다.

지금까지 분산 환경에서의 인증 시스템에 관해서는 많은 연구가 있었다[1][2][3]. 그 중에서 일회용 패스워드 시스템은 네트워크 도청에 의한 패스워드 도용이나 패스워드 추측 등 분산 컴퓨팅 환경에서 발생할 수 있는 여러 가지 위협에 대처하기 위한 방법 중 한 가지이다. 일회용 패스워드 시스템은 호스트에 로그인 할 때마다 다른 패스워드를 사용함으로 도청의 위험을 해결하고 패스워드의 재사용이 불가능해서 침입자가 네트워크 도청을 통해서 패스워드를 알아내어도 더 이상 사용할 수 없게 된다[4].

본 논문에서는 앞에서 기술한 바와 같이 클라이언트가 서버에 로그인을 요청할 때 사용자를 인증하기 위해 사용되는 패스워드를 매번 생성하여 한번 사용한 패스워드는 다시 사용하지 않는 안전한 JOTP 알고리즘의 설계 및 구현에 대해 설명한다. 논문의 구성은 다음과 같다. 2장에서는 IETF에서 제안된 패스워드 보안 알고리즘의 기능과 특징에 대해 설명한다. 3장은 확장된 JOTP 알고리즘의 설계 및 구현에 관하여, 4장에서는 실제 구현된 JOTP 알고리즘의 실험결과 및 고찰에 관하여 설명하고 마지막 5장에서는 결론으로 본 논문을 끝맺는다.

2. 일회용 패스워드 시스템

일회용 패스워드 시스템은 크게 S/Key라 불리는 방식과 Challenge/Response 방식이 있는데 본 장에서는 최근의 일회용 패스워드에 대하여 고찰하고 그 문제점을 진단한다[5].

2.1 S/Key 방식

S/Key 방식은 클라이언트 측면에 일회용 패스워드가 생성되어져야 하고, 다른 하나는 서버 측면으로 일회용 패스워드가 검사되어져야 한다. 일회용 패스워드는 MD(Message Digest) 4 one-way hash 함수를 이용해서 생성되고 검사되어진다. 이 시스템은 8 바이트의 입력을 받아 그 출력을 얻도록 제작되었으며 최근에는 MD4의 취약성을 보완한 MD5로 업데이트되었다.

일회용 패스워드는 단방향 함수를 여러 번 적용함으로 계속해서 생성되어진다. 첫 번째 일회용 패스워드는 사용자의 비밀패스워드(s)를 정해진 특정수(n)만큼 단방향 함수를 수행함으로 생성되어진다.

예를 들어 n=4라고 가정하면

p(1) = f(f(f(f(s)))) 이며,

다음 일회용 패스워드는 사용자의 패스워드를 단방향 함수에 n-1번 수행함으로 생성되어진다.

p(2) = f(f(f(s)))

처음에 호스트 컴퓨터는 수신한 일회용 패스워드의 사본을 저장하고, 그것을 단방향 함수에 적용한 후 그 결과가 패스워드 파일 안에 저장된 사본과 일치하지 않으면, 인증 요구는 실패하게 된다. 만약 그들이 일치하면, 패스워드 파일 안에 있는 사용자의 엔트리는 단방향 함수의 마지막 실행전에 저장되어 있던 일회용 패스워드의 사본으로 갱신되어진다.

2.2 Challenge-Response 방식

Challeng-Response 방식은 사용자가 로그인을 요구했을 때, 서버는 난수에 의해 생성된 challenge라고 하는 메시지를 클라이언트에게 전달하고 이 challenge를 이용해 생성된 패스워드를 서버로 전송함으로써 사용자를 인증하는 일회용 패스워드 방식이다.

각 사용자에 대해 사용자 식별번호(PIN)와 비밀키를 부여하고, 클라이언트와 서버의 데이터베이스에는 비밀키가 저장되어 있어서 로그인할 때 발생되는 난수를 암호화하는데 사용된다. 클라이언트가 로그인을 요청할 때 PIN을 전송하면, 서버는 난수를 발생하여 이 난수를 challenge 메시지로 클라이언트에게 전송하고, 서버의 데이터베이스에서 PIN에 해당하는 비밀키를 찾아 발생한 난수를 암호화한다. 클라이언트는 challenge 메시지로 받은 난수를 자신이 저장하고 있는 비밀키를 이용하여 암호화하고 이를 response 메시지로 서버에게 보내면, 서버는 response 메시지를 받아 자신이 난수를 암호화한 값과 일치하면 로그인을 허락한다.

3. 확장된 JOTP 보안 알고리즘

3.1 알고리즘 설계

안전한 보안을 제공하기 위한 확장된 JOTP 보안 알고리즘의 흐름도는 그림 1과 같다.

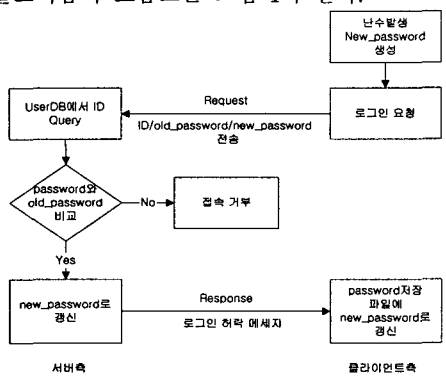

그림 1. JOTP 알고리즘 흐름도
Fig. 1. JOTP Algorithm Flow Diagram

본 논문에서 제안한 알고리즘의 전체적인 구성을 살펴보면 다음과 같다. 사용자는 서버에서 제공하는 서비스를 사용하기 위해 계정등록을 요청하면 로그인에 사용될 ID와 패스워드를 등록하여 서버의 데이터베이스에 이들을 저장한다. 서버에 등록된 패스워드와 같은 패스워드를 클라이언트의 하드디스크나 플로피디스켓에 저장하여 LAN이나 인터넷을 통하여 서버로 로그인을 요청할 때 사용한다. 클라이언트는 서버와 접속하기 전에 항상 난수발생에 의해 새로운 패스워드를 생성하여 서버로 전송한다. 결과적으로 클라이언트가 로그인할 때는 등록된 계정인 ID와 이미 저장된 패스워드인 old_password와 난수발생에 의해 생성된 패스워드인 new_password를 서버에게 보내면, 서버는 클라이언트로부터 로그인 요청을 받은 후, 전송된 old_password와 자신이 보관하고 있던 패스워드를 비교하여 서로 일치하면 로그인을 허락한다. 접속이 성립되면 서버와 클라이언트는 각각 new_password로 기존의 패스워드의 내용을 갱신한다.

3.2 알고리즘 구현

본 논문에서는 알고리즘을 구현하기 위한 GUI 및 프로그래밍 개발툴로써 JDK(Java Development Kits)를 사용하고, JDK와 연동될 데이터 베이스 엔진으로는 SQL 7.0을 사용하였다.

확장된 JOTP 보안 알고리즘의 구현은 클라이언트측 개체와 서버측 개체로 나누어 작성하였는데, 그림 2는 클라이언트 측에서 패스워드 파일에 저장된 패스워드를 읽고, 새로운 패스워드를 난수 발생에 의해 생성하는 메소드를 나타낸 것으로써, 클라이언트가 서버에 로그인을 요청할 때 발생한다.

```
file=new File("pass.txt");
fReader=new FileReader(file);
int size=(int)file.length();
oldPass=new char[size];
for(int i=0; i<size-1; i++)
      byPass=fReader.read(oldPass, i, size-i);
fReader.close();
oldPassword=new String(oldPass, 0, size);
Random randomPass = new Random(Integer.parseInt(oldPassword));
newPassword=String.valueOf(randomPass.nextInt());
```

그림 2. 새로운 패스워드를 발생하는 메소드
Fig. 2. The Method of Generating New Passwords

```
rs=stmt.executeQuery("SELECT * FROM password WHERE id='"+id+"'");
rs.next();
dbpw=rs.getString(2);
if(dbpw==old_password)
      stmt2.executeUpdate("UPDATE password SET
            pw='" + new_password + "' WHERE id='" + id + "'");
```

그림 3. 사용자를 확인하는 메소드
Fig. 3. The Method of Verifing Users

그림 3에서 나타난 메소드는 서버의 데이터베이스에서 사용자 ID를 질의하고 패스워드의 동일성 여

부를 판단하여 등록된 사용자일 경우 데이터베이스에 새로운 패스워드로 갱신하는 기능을 가진다.

로그인을 요청한 클라이언트가 등록된 사용자임을 확인한 뒤, 클라이언트측의 패스워드 파일에 새로운 패스워드로 갱신하는 이벤트를 처리하는 부분의 프로그램 코드는 그림 4와 같다.

```
FileWriter fWriter=new FileWriter(file);
fWriter.write(newPassword);
fWriter.close();
```

그림 4. 패스워드를 갱신하는 메소드
Fig. 4. The Method of Updating Password

4. 실험 결과 및 고찰

그림 5, 6은 클라이언트와 서버의 연결 전의 모습이다. 클라이언트 개체에서 보듯이 패스워드를 기입하는 텍스트 필드는 없다. 연결 버튼을 눌러서 서버에 접속을 시도하면서 내부적으로 난수발생에 의해 새로운 패스워드를 발생한다.

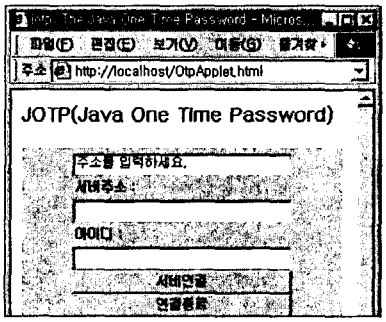

그림 5. 연결전 클라이언트 실행 화면
Fig. 5. Client Screen before Connection

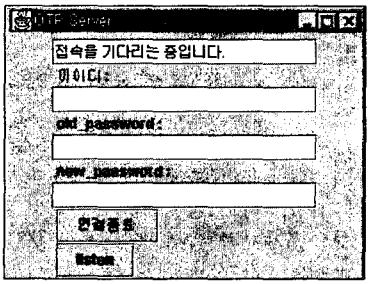

그림 6. 연결 전 서버 실행 화면
Fig. 6. Server Screen before Connection

그림 7, 8는 접속 후의 클라이언트와 서버의 모습이다. 서버 개체에 나타난 test-id, -1454133806, -939569004는 클라이언트가 연결버튼을 누르면서 서버에 보낸 것으로 ID/old_password/new_password로 구성되어 있다. 물론 실제 사용에 있어서 패스워드를 노출할 필요는 없겠지만 접속 성공여부를 쉽게 판단하기 위해서 출력시킨 것이다.

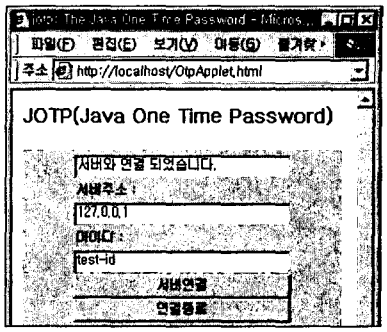

그림 7. 연결 후 클라이언트 실행 화면
Fig. 7. Client Screen after Connection

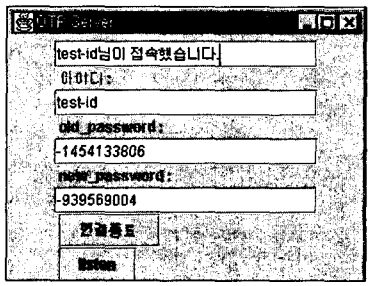

그림 8. 연결 후 서버 실행 화면
Fig. 8. Server Screen after Connection

접속이 성공적으로 이루어진 후 클라이언트의 패스워드 파일과 서버의 데이터베이스에서 새 패스워드로 갱신한 모습을 그림 9, 10에 나타내었다. 이해를 돕기 위해 새 패스워드로 바뀌기 전과 후의 그림을 합성하여 한 눈에 비교할 수 있도록 하였다.

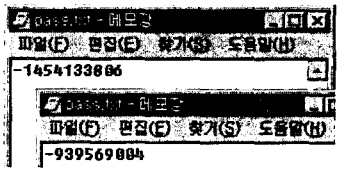

그림 9. 클라이언트의 패스워드 파일
Fig. 9. Client's Password File

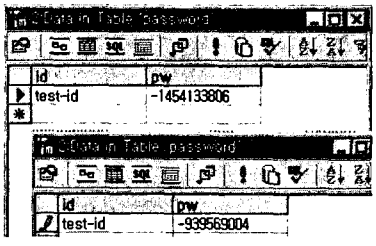

그림 10. 서버의 데이터베이스
Fig. 10. Server's DataBase

표 1은 'test-id'라는 사용자의 ID로 서버에 접속을 5번을 시도했을 때 old_password와 new_pass-

word의 값을 나타낸 것이다. 현재의 접속한 횟수를 n이라고 했을 때 n-1번째의 new_password와 n번째의 old_password가 같음을 알 수 있다.

표 1. Old_password와 New_password 비교
Table 1. Comparison Old_password with New_password

횟수	old_password	new_password
1	-1454133806	-939569004
2	-939569004	2042232660
3	2042232660	-1557241232
4	-1557241232	1024999232
5	1024999232	87967194

기존의 일회용 패스워드 알고리즘은 그림 11과 같이 5 단계를 거친 후에 클라이언트의 접속 여부를 결정하며, 그 과정 속에 서버와 클라이언트 사이의 메시지를 주고받는 절차가 있다. 이것은 그만큼 네트워크의 사용이 증가되면 네트워크의 전송상태에 따라 클라이언트의 대기시간이 길어질 수 있으며, 또한 서버에서 challenge 생성과 암호화의 수행은 서버의 부하를 증가시키는 직접적인 요인이 된다.

그러나 확장된 알고리즘에서는 그림 12와 같이 기존의 방법보다 2 단계를 줄여서 접속여부를 결정한다. 클라이언트는 로그인 요청과 동시에 자신이 생성한 암호화된 패스워드를 서버로 보낸다. 그러면 서버에서는 확인 절차를 거쳐 정당한 사용자로 확인되면 기존의 패스워드를 새로이 보내온 패스워드로 갱신하고, 로그인을 허락한다.

본 알고리즘은 서버의 부하를 현저히 줄일 수 있고, 네트워크의 사용횟수도 절반정도로 줄일 수 있다. 그러므로 결과적으로 클라이언트의 대기시간을 상당히 단축시킬 수 있게된다.

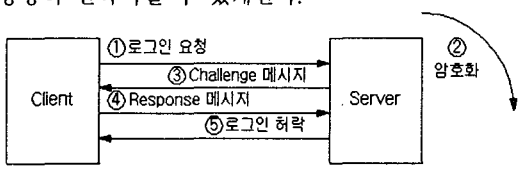

그림 11. 기존의 일회용 패스워드 시스템
Fig. 11. Legacy One Time Password System

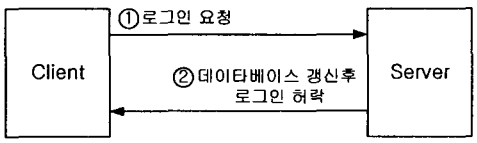

그림 12. 확장된 알고리즘
Fig. 12. Extended Algorithm

본 연구에서 실험해 본 결과 클라이언트가 서버에 접속을 시도할 때마다 새로운 패스워드를 생성하고 이를 XOR에 의해 암호화하여 전송 중에 패스워드의 노출을 방지하며, 사용자 인증이 확인되면 새로운 패스워드로 갱신하여 한번 사용한 패스워드는 다시 사용하지 않아 보안성을 높일 수 있다는 결론을 얻을 수 있으며, 새로운 패스워드 발생이 클라이언트 측에서 이루어지기 때문에 로그인할 때 새로운 패스워드를 함께 전송함으로서 서버에 접속하기 위한 절차가 기존의 OTP 알고리즘에 비해 절반 가량 줄어들어 네트워크 사용이 감소하며 클라이언트의 대기 시간도 줄어든다는 것을 확인할 수 있다.

5. 결론

보안을 위하여 사용자 인증 방법이 가장 광범위하게 쓰이고, 사용자를 인증하는 수단으로 가장 손쉽게 사용하는 방법이 사용자에 따른 패스워드를 이용하는 것이다[9]. 패스워드를 이용한 사용자 인증의 단점을 보완하기 위한 방안으로 클라이언트가 서버로 로그인 할 때마다 사용자 인증에 필요한 패스워드를 발생하여 한번 사용한 패스워드는 다시 사용하지 않는 JOTP 보안 알고리즘을 제안하였다.

본 연구에서 구현된 알고리즘은 기존의 OTP 보안 알고리즘보다 접속 절차가 간단하여 네트워크 사용을 적게 하고, 패스워드 발생과 관리가 클라이언트측에서 이루어져 서버의 부하를 경감시키는 장점을 가지는 반면 클라이언트의 패스워드 파일을 플로피디스켓에 저장할 경우 저장매체로서의 안정성이 떨어질 수 있으며, 하드디스크에 저장할 경우 보안성이 떨어질 수 있다는 단점을 가진다.

향후 연구 · 보완되어야 할 사항으로는 첫째 클라이언트의 패스워드 파일을 저장하는 매체에 대한 연구와, 둘째 전송되는 패킷의 포맷이 ID, old_password, new_password의 순서로 전송되고 있으나 이 순서가 무작위로 위치하여 보완성을 더욱 높일 수 있는 방안이 연구되어야 할 것으로 사료된다.

참고문헌

[1] D. Borman, "Telnet Authentication : Kerberos Version 4", RFC1411, January, 1993.
[2] John Kohl, Clifford Neuman, "The Kerberos Network Authentication Service V5", RFC1510.
[3] Neil Haller, "The S/KEY One-Time Password System", In Proceedings of the Internet Society Symposium on Network and Distributed System Security, February 1994, pp151-157.(RFC1760)
[4] Leslie Lamport, "Password Authentication wi th Insecure Communication", Communications of the ACM 24.11 (November 1981), pp770-772.
[5] N. Haller, "A One-Time Password System", RFC2289, February 1998.
[6] National Bureau of Standards, Password Usage Standards, NBS publication, Washington, 1984.
[7] Karfman Chalrie and Perlman Radia, Network security, Prentice Hall, 1995.
[8] J. Klensin, R. Catoe, P. Krumviede, "IMAP/-POP AUTHorize Extension for Simple Challenge-/Response", RFC2195, September 1997.
[9] Haller, N., and R. Atkinson, "On Internet-Authentication", RFC 1704, October 1994.