OTP 기반의 웹서비스 인증 메커니즘 설계 및 구현

Design and Implementation of OTP Based Authentication Mechanism for Web Service

송유진(Song You Jin)*, 이동혁(Lee Dong Hyeok)*

초 록

SOAP 표준 규약은 보안에 대한 특별한 언급을 하지 않고 있으며 이는 정보보호의 측면에서 상당한 위험 요소를 안고 있다. 특히 사용자 인증 시 SOAP Header내의 Username과 Password Element로 텍스트 값을 전송하게 되면 악의를 가진 자에게 스니핑(Sniffing) 및 재연(Replay) 공격을 받게 될 가능성이 있다. 본 논문에서는 SOAP 메시지의 인증 정보에 대한 여러 공격들을 효율적으로 대응할 수 있는 새로운 사용자 인증 메커니즘을 제시한다. 제안한 메커니즘은 S/KEY 시스템의 단점을 보완하여 사용횟수의 제한이 없으며 초기화 과정을 여러 번 수행할 필요가 없다. 또한 오버헤드에 대한 부담이 없으며 seed의 노출에 대한 위험 부담도 상대적으로 적다. 제안한 메커니즘은 SOAP 메시지의 사용자 인증을 보다 효율적이고 안전하게 제공한다.

ABSTRACT

The SOAP specifications are not provided a functions of information security, especially authentication function. In case of user authentication, delivery of the username and password elements can be exposed to sniffing/replay attack by malicious attacker. In this paper, we propose a new mechanism to protect authentication attacks for the SOAP messages. The proposed mechanism is compensated for weakness of S/KEY system. Our mechanism has no limitation for time and overhead and also provide a more effective and secure delivery.

키워드 : 웹서비스, 일회용 패스워드, 인증 메커니즘, S/KEY

Web Services, One-Time Password, Authentication Mechanism, S/KEY

이 연구는 2003학년도 동국대학교 연구년 지원에 의하여 이루어졌음.

* 동국대학교 대학원 전자상거래학과

1. 서 론

최근 웹서비스가 각광을 받게 되면서 웹서비스 보안에 대한 여러 논의도 함께 진행되어 왔다. 특히 웹서비스에서 사용되는 SOAP 메시지는 불특정 다수에게 노출된 인터넷을 통과하며 이는 정보보호의 관점에서 상당한 위험요소를 안고 있다.

SOAP 표준 규약 자체에는 보안에 대한 어떠한 언급도 되어 있지 않다[8]. SOAP는 사용자 인증시 SOAP Header내의 Username과 Password Element로 인증정보를 텍스트로 전송하여 사용자를 인증한다. 그러나 이러한 경우 악의를 가진 자가 사용자의 인증정보를 스니핑(sniffing)하게 된다면 패스워드를 쉽게 유추할 수 있게 된다.

이에 대한 보안 표준으로 WS-Security가 제시되었으나, 기존의 WS-Security에 있는 UsernameToken Element에는 패스워드에 대한 다이제스트 값을 보낼 수 있는 옵션만을 제공하고 있다[9]. 그러나 다이제스트 값 역시 재사용될 수 있다는 문제점이 남아 있으며 재연(Replay) 공격의 가능성이 존재한다. 한편 WS-Security에 SAML, Keberos 등의 보안 토큰을 내장할 수 있으나 이는 성능 저하를 가져올 수 있고 처리 방법도 복잡한 편이다[9,10].

본 논문에서는 이러한 메커니즘의 필요성으로부터 일회용 패스워드(OTP:One-Time Password) 기반의 웹서비스 인증 메커니즘을 연구한다. 일회용 패스워드 시스템은 수동적 공격(Passive Attack)에 대해 사용자의 패스워드를 보호하기 위한 메커니즘으로써 네트워크상의 스니핑(Sniffing), 재연(Replay) 및 서버 침해 공격 등으로부터 안전하고 효율적인 인증을 제공한다.

그러나 S/KEY 시스템에는 몇가지 단점이 있다. 특히 사용횟수에 제한이 있고 여러 번 초기화 과정을 수행해야 하는 점은 트랜잭션이 빈번하게 이루어지는 웹서비스 환경에서는 적합하지 않다. 따라서 본 논문에서는 S/KEY 시스템을 기반으로 효율적이고 안전한 인증 메커니즘을 새롭게 제안한다. 제안한 메커니즘은 S/KEY 시스템의 단점을 보완하여 사용횟수의 제한이 없으며 초기화 과정을 여러 번 수행할 필요가 없다. 또한 해쉬 연산의 횟수가 5회로 고정되어 오버헤드에 대한 부담이 없다. 아울러 seed의 노출에 대한 위험부담도 상대적으로 적은 편이다. 제안한 메커니즘은 SOAP 메시지의 사용자 인증을 보다 효율적이고 안전하게 제공한다. 또한 이러한 메커니즘을 웹서비스에 적용하여 사용자와 서버간 인증 절차에 따라 SOAP 메시지를 주고받음으로써 웹서비스에서의 사용자 인증이 보다 안전하고 효율적으로 이루어지는 것을 검증하였다.

본 논문의 구성은 다음과 같다. 2장에서는 웹서비스 관련 인증 메커니즘 및 RFC-1760 상에 기술된 S/KEY 시스템을 연구한다. 그리고 3장에서는 S/KEY 시스템을 보완할 수 있는 새로운 인증 메커니즘을 제안하고 4장에서 안전성과 효율성을 분석하였다. 5장에서는 실제 웹서비스 시스템을 구현하여 제안한 방법에 대한 정상적인 작동 여부를 확인하였으며 6장에서는 결론 및 향후 연구방향을 제시하였다.

2. 관련 인증 메커니즘 연구

2.1 OASIS 인증 메커니즘

WS-Security는 보안 관련 데이터를 전달하기 위한 SOAP Header 요소의 내용을 정의하고 있다. XML 전자서명이 사용되는 경우, 헤더에는 메세지가 서명되는 방법, 사용된 키, 서명이 끝난 데이터를 나타내는 정보가 포함된다. 이들은 XML 전자서명 규격으로 정의되고 있다. 동일한 메세지 요소가 암호화되는 경우, XML 암호 규격에서 규정되는 암호화 정보가 WS-Security 의 헤더에 포함된다[10]. 이와 같이 WS-Security는 전자서명이나 암호화의 형식은 지정하지 않는다. WS-Security에는 다른 규격에서 규정된 보안 정보를 SOAP 메세지에 넣는 방법을 규정하고 있다.

WS-Security의 목적은 XML 기반 보안·메타 데이타의 컨테이너를 규정하는 것으로서 신원의 확인과 액세스 제어의 개념을 SOAP 메시지에 반영하고 있다. SOAP 메세지를 사용해 보안 기능을 구현하기 위해서는 메세지에 다음과 같은 정보가 포함되어야 한다.

- 메세지에 포함되는 개체의 식별
- 개체가 멤버로서 소속하는 그룹의 증명
- 개체가 보관 유지하고 있는 액세스 권한의 증명
- 메세지가 개조되어 있지 않은 것의 증명
- 허가되지 않은 경우, 정보를 공개하지 않도록 하는 메커니즘

이러한 WS-Security는 SOAP 메세지를 사용해 메시지 송신자 신원의 식별과 그 권한의 표시에 보안 토큰을 사용하는 방법을 규정하고 있다.

WS-Security는 무한의 사용자 인증 방식을 지원하고 있다. 규격에서는 특히 3 종류의 방법에 대해 다루고 있다.

- Username/Password 메커니즘
- X. 509 증명서를 사용한 PKI 메커니즘
- Kerberos 메커니즘

여기서는 사용자명/패스워드 메커니즘에 대해 살펴본다.

1) 사용자명/패스워드

송신자의 인증 정보를 전달하는 방법으로서 사용자명과 패스워드의 쌍을 이용할 수 있다. HTTP 의 기본인증이나 다이제스트 인증에서도 이 구조가 사용되고 있다. 실제, HTTP 다이제스트 인증의 구조를 알고 있으면, 이 인증 메커니즘을 이해하는 것은 어렵지 않다. 사용자명과 패스워드의 쌍으로 구성된 인증 정보를 송신하기 위해서 WS-Security에서는 UsernameToken 요소가 정의되고 있는데 이 요소의 스키마는 다음과 같다.

```
<xs:element name="UsernameToken">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Username"/>
      <xs:element ref="Password" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Id" type="xs:ID"/>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
</xs:element>
```

이 스키마에서는 2개의 형태(Username과 Password)를 참조하고 있다. Password 요소에는 패스워드가 어떠한 형태로 송신되고 있는지를 나타내는 Type 라는 이름의 속성이

정의되고 있다. 패스워드는 평문 또는 다이제스트의 형식으로 송신할 수 있다. SOAP 메세지에 UsernameToken를 넣어 송신하는 경우, 그 XML은 다음과 같은 형태가 된다.

```
〈wsse:UsernameToken〉
  〈wsse:Username〉scott〈/wsse:Username〉
  〈wsse:Password Type="wsse:PasswordText〉
                  password〈/wsse:Password〉
〈/wsse:UsernameToken〉
```

이러한 형태의 예에서 패스워드는 평문으로 송신되고 있어 보안상의 취약점이 있다. 따라서보다 안전한 형태로 패스워드를 송신하기 위해 다이제스트를 송신한다.

```
〈wsse:UsernameToken〉
  〈wsse:Username〉scott〈/wsse:Username〉
  〈wsse:Password Type="wsse:PasswordDigest"〉
    KE6QugOpkPyT3Eo0SEgT30W4Keg=
  〈/wsse:Password〉
  〈wsse:Nonce〉5uW4ABku/m6/S5mE+L7vg==
  〈/wsse:Nonce〉
  〈wsu:Created xmlns:wsu=
    "http://schemas.xmlsoap.org/ws/2002/07/utility"〉
      2002-08-19T00:44:02Z
  〈/wsu:Created〉
〈/wsse:UsernameToken〉
```

SHA1 해쉬함수를 사용하여 패스워드를 해독하기 어렵게 함으로써 보안 기능을 향상시키고 있다[5]. 패스워드의 다이제스트는 nonce와 작성 일시와 패스워드를 연결한 것에 해쉬함수 처리한 것이다. nonce는 16 바이트의 데이터로 Base64 로 인코드(encode)된다. 이 방법을 이용하는 경우, 우선 송신측이 이러한 정보로부터 패스워드의 다이제스트를 작성한다. 수신측에서는 평문으로 보관되고 있는 패스워드를 해쉬함수로 처리하여 다이제스트를 다시 작성한다. 결과가 일치하면, 송신된 패스워드가 정확한 것으로써 판단한다.

한편, 이 방식은 재연(replay) 공격으로부터 안전하지 않다. 이 방법에서는 wsu:Timestamp 요소를 함께 사용해서 유효기간을 짧게 설정한다. 더우기 wsu:Timestamp 요소에 서명함으로써 타임스탬프의 변조를 검출할 수 있도록 한다. 이와 같이 악의의 공격자가 정당한 UsernameToken를 추출하여 웹서비스 공격이 가능하게 될 수 있다. 재연 공격으로부터 Web 서비스를 지키려면 수신한 메세지를 유일하게 식별할 수 있는 특징을 메세지의 유효기간중에 캐쉬에 보관하여 추적할 수 있는 메커니즘이 필요하다.

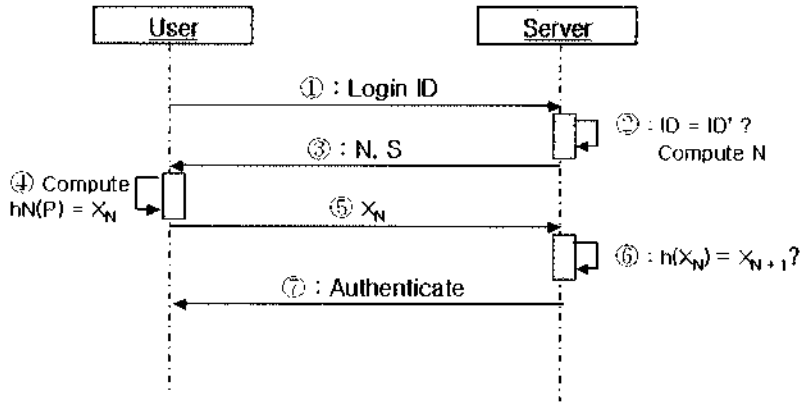
2.2 S/KEY 일회용 패스워드 시스템

2.2.1 S/KEY OTP 시스템

S/KEY OTP(One-Time Password) 시스템은 Passive attack에 대해 사용자의 패스워드를 보호하기 위한 메커니즘으로써 네트워크상의 스니핑(Sniffing)과 재연(Replay) 공격으로부터 안전한 인증을 제공한다. 이 시스템은 기존의 인증 시스템들에 비해 여러 가지 장점을 가지고 있다.

OTP생성 알고리즘은 일방향함수를 여러번 적용함으로 계속해서 생성되어진다. 즉, 첫번째 OTP는 사용자의 비밀 패스워드(s)를 정해진 특정 수(n)만큼의 일방향함수를 수행함으로 생성되어진다. 예를 들어 n=4 라고 가정하면,

$$p(1) = f(f(f(f(s))))$$



〈그림 1〉 S/KEY 시스템을 이용한 인증 방식

N, S : OTP생성시 사용되는 seed와 숫자
h : 일방향 해쉬 함수

다음 OTP는 사용자의 패스워드를 일방향함수에 n-1번 수행함으로 생성되어진다. OTP p(i)의 사용을 모니터하고 있는 도청자는 다음 패스워드 p(i+1)를 생성해낼 수 없을 것이다.

처음에 호스트 컴퓨터는 수신한 OTP의 복사본을 저장하고, 그것을 일방향함수에 적용한다. 만약 그 결과가 시스템의 패스워드 파일 안에 저장된 복사본과 일치하지 않으면, 그 인증 요구는 실패하게 된다[3, 4].

〈그림 1〉은 RFC-1760에서 명시하고 있는 S/KEY의 방식으로 인증하게 될 경우의 메시지 흐름을 나타내고 있다.

S/KEY의 인증 절차는 다음과 같다[2].

① 사용자는 자신의 아이디로 신분을 밝힌다.
② 시스템은 기대되는 OTP의 일련 번호와 시스템 특유의 secd로 challenge한다.
③ 사용자는 일련 번호와 seed를 이용하여 OTP를 생성한다.
④ 사용자는 계산된 일회성 패스워드를 전송한다.
⑤ 서버는 미리 저장된 패스워드로 OTP를 생성한다.
⑥ 서버는 전송되어진 OTP와 5번에서 계산된 OTP가 동일하면 사용자를 인증한다.
⑦ 서버는 전송된 OTP를 저장한다.

2.2.2 S/KEY OTP 시스템 분석

1) 장점

S/KEY 기반 인증 메커니즘은 일반적인 패스워드 인증 방식에서의 다음과 같은 문제점들을 해결해 줄 수 있다.

- 패스워드가 노출되는 경우 : 한번 사용된 OTP는 다시는 재사용되지 않는다. 따라서 공격자가 패스워드를 가로채더라도 그 값은 무의미하므로 스니핑 및 재연 공격을 방지한다.

- 서버의 정보가 노출되는 경우 : 패스워드(Password)는 사용자만 알고 있으며 서버에는 패스워드가 저장되지 않는다. 서버에는 X_{N+1}이 저장되어 있으며 N과 seed만으로 X_N을 계산할 수 있는 사람은 사용자뿐이다. 따

라서 공격자가 서버의 데이터베이스에 접근을 하고 그 데이터를 알아내더라도 그 결과는 무의미해진다.

2) 문제점 분석

① 사용 횟수의 제한

S/KEY 시스템은 초기화 과정에서 적절한 숫자 N을 생성하게 되며, 이에 따라 사용횟수가 N회로 제한된다. 따라서 N회의 인증 이후에 별도의 초기화 과정이 필요하다. 만약, 초기화 과정에서 충분한 사용횟수를 고려하여 큰 수의 N을 선택한다면 그만큼 OTP 생성에 필요한 해쉬연산의 부담이 증가된다. 이러한 방식은 웹서비스 환경에 적합하지 않으므로 본 논문에서는 S/KEY방식을 보완한 새로운 메커니즘을 제시하였다. 제안한 메커니즘은 사용횟수의 제한이 없으며 어떤 경우에도 해쉬연산은 5회만 거치게 되어 일련번호의 증가에 따른 해쉬연산에 대한 부담이 없다.

② 초기화 과정에서의 패스워드 노출

S/KEY 시스템은 사용 횟수에 따라 반복적인 초기화 과정이 필요하며 이러한 초기화 과정에서 비밀 패스워드가 노출되게 된다. 트랜잭션이 빈번히 이루어지는 웹서비스 환경을 고려해 볼 때 이러한 방식은 적합하지 않다. 제안한 메커니즘은 최초 1회의 초기화 과정을 거치면 더 이상 초기화가 필요하지 않으므로 전송 과정에서 비밀정보가 노출되지 않는다.

③ 사전 참조식 공격

S/KEY 시스템으로 인증하는 과정에서 일회용 패스워드와 seed가 함께 노출되는 경우에는 오프라인 사전 참조식 공격(dictionary attack)의 위험성이 존재한다. S/KEY 시스템에 대한 사전 참조식 공격의 일반적인 시나리오는 〈그림 2〉에 나타나 있다.

〈그림 2〉에서 공격자는 특정 사용자의 정보로 인증하기 위해 적절한 단어로 여러 번 로그인을 시도하고 있다. 이와 같이 공격자가

```
username : jdoe
challenge : 99 k113355
response : WELD GUY CHIMP SWING GONE
jdoe's real password : ???

dictionary word 1 : love
challenge : 99 k113355
response : BAD LOST CRUMB HIDE KNOT
( well that's not it ,,,, )

dictionary word 1 : you
challenge : 99 k113355
response : FORT HARD BIKE HIT SWING
( not it either,,, )

dictionary word 1 : secret
challenge : 99 k113355
response : WELD GUY CHIMP SWING GONE
(bingo,,,!!)
```

〈그림 2〉 사전 참조식 공격의 시나리오[7]

challenge와 response를 가로채는 경우, 공격자는 사전에 있는 적절한 단어들과 비교하여 패스워드를 유추해 낼 수 있다[7]. 따라서 사용자 고유의 패스워드뿐만 아니라 seed에 대한 적절한 보호도 필요함을 알 수 있다. 본 논문에서 제안한 메커니즘은 한 쌍의 패스워드가 사용되어 단일 패스워드 방식보다 사전 참조식 공격에 상대적으로 안전하다. 또한 seed 값이 고정되지 않고 매 인증마다 변경되므로 seed 노출에 따른 위험 부담이 상대적으로 적다.

3. 제안 메커니즘

본 장에서는 S/KEY OTP 시스템의 문제점을 개선한 새로운 메커니즘을 제안한다. 그리고 제안한 메커니즘에 대한 안전성 및 효율성을 검토한다.

3.1 인증 절차

본 절에서는 최초 사용자와 서버가 상호간 신뢰하고 있다는 가정 하에 상호간 인증 정보를 확립하는 방법과 인증에 필요한 절차들에 대하여 기술한다.

본 장에서 사용되는 기호들은 〈표 1〉과 같다.

3.1.1 초기화 단계

초기화 단계는 사용자와 서버 상호간 기본 인증 정보를 확립하기 위한 단계이며 최초 1회만 실행된다. 이 절차의 실행에 앞서 사용자는 먼저 로그인에 필요한 ID와 한 쌍의 적절한 패스워드를 생성한다. 여기에서 생성된 패스워드는 사용자만 알고 있으며 어느 곳에도 노출되거나 저장되지 않는다. 초기화 단계에서 필요한 절차는 다음과 같다. 사용자가 서버에 로그인하게 되면 서버는 사용자에게 초기 seed를 사용자에게 전달하고, 사용자는 seed와 한 쌍의 패스워드를 기반으로 K_3를 계산하여 서버에 전달한다. 그리고 서버는 사용자로부터 전송된 K_3와 초기 seed를 보관한다. 이러한 과정을 간단히 표시하면 다음과 같다.

① Message : U → H : ID

② Message : U ← H : S_{Init} (S_{Init}은 임의

〈표 1〉 기 호

기 호	의 미
P_N	패스워드 (N번째)
U	Client
H	Host
S	seed(S_S: 저장된 값, S_N : 신규, S_{Init} : 최초값)
K_N	OTP (N번째)
‖	합성 연산자
$H(Str)_{Key}$	Str을 Key로 해쉬연산
K_S	Host에 보관된 OTP

로 생성한 값)

③ Message : U → H : K_3

($K_3 = H(K_1)_{K_2}$;

$K_1 = H(P_1 | S_{Init})_{P_2}$,

$K_2 = H(P_2 | S_{Init})_{P_1}$)

④ Store : H Store $K_3 \rightarrow K_S$

$S_{Init} \rightarrow S_S$

3.1.2 인증 단계

여기에서는 초기화 단계 이후 사용자가 서버로부터 인증받는 과정에 대해 기술한다. 인증에 필요한 메시지의 전달 과정을 간단히 표현하면 〈그림 3〉과 같다.

① Message : U → H : ID

② H : Generate S_N

③ Message : U ← H : S_S, S_N

④ Compute : U : $K_1 = H(P_1 | S_S)_{P_2}$,

$K_2 = H(P_2 | S_S)_{P_1}$, $K_3 = H(K_{N1})_{K_{N2}}$

($K_{N1} = H(P_1 | S_N)_{P_2}$, $K_{N2} = H(P_2 | S_N)_{P_1}$)

⑤ Message : U → H : K_1, K_2, K_3

⑥ Compare : $H(K_2)_{K_1} = K_S$?

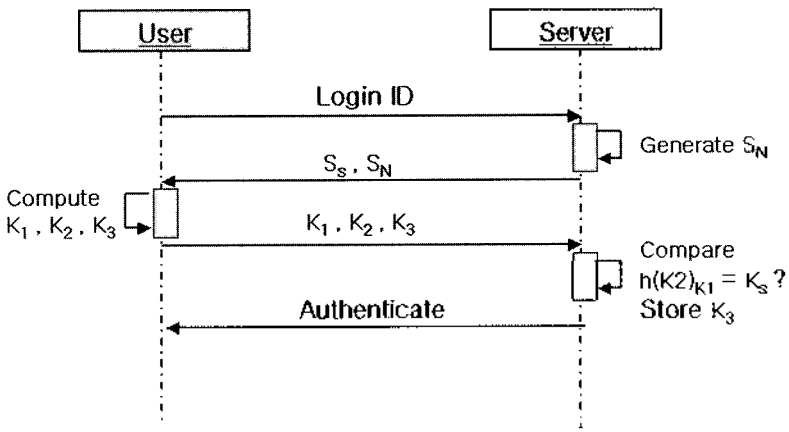
⑦ Store : H Store $K_3 \rightarrow K_S$

$S_N \rightarrow S_S$

⑧ Authenticate : U ← H : Authenticate

사용자는 한 쌍의 패스워드를 가지고 있다. 그리고 사용자가 서버에게 ID를 전달하면 서버는 기존에 저장된 S_S와 임의로 생성한 S_N을 사용자에게 전달한다. 사용자는 서버에게 전달받은 두 쌍의 seed를 이용하여 K_1, K_2, K_3를 각각 계산하고 다시 서버에 전달하게 된다. (각 키들은 패스워드와 seed의 조합으로 생성된다. 인증 후 이러한 값들은 더 이상 재사용되지 않으므로 공격자의 스니핑 및 재연 공격과 사전 참조식 공격을 미연에 방지할 수 있게 된다.) 서버는 K_2와 K_1으로 다시 해쉬연산을 수행하며 그 결과가 미리 보관해 두었던 K_S와 일치하는 경우 사용자를 인증하고 K_3와 S_N을 각각 K_S와 S_S로 저장하게 된다. 다음 로그인에는 S_N을 seed로 사용하게 될 것이다.

앞서 초기화 단계에서 K_3를 $H(K_1)_{K_2}$로 생



〈그림 3〉 제안한 메커니즘

〈표 2〉 각 인증 방식들의 안전성 비교

공격유형 \ 방 식	일반 패스워드 방식	S/KEY 시스템	제안한 메커니즘
스니핑 공격 방지	X	△	○
재연 공격 방지	X	△	○
사전 참조식 공격 방지	X	X	△
서버 침해 공격 방지	X	○	○

X:안전하지 않음 △:부분적으로 안전함 ○:안전함

성하였으며 여기에서 $K_1 = H(P_1 | |S_{Init})_{P_2}$ 이고, $K_2 = H(P_2 | |S_{Init})_{P_1}$ 이다. 이 단계 이후 서버는 S_{Init}을 S_S로 저장하고 K_3를 K_S로 저장하였다. 인증 단계에 들어와서 서버는 사용자에게 S_S를 보내며 사용자는 이를 근거로 K_1과 K_2를 각각 $H(P_1 | |S_S)_{P_2}$와 $H(P_2 | |S_S)_{P_1}$로부터 생성하였다. 앞서 언급하였듯 초기화 단계에서의 S_{Init}는 인증 단계에서 S_S가 된다. 따라서 인증 단계에서 $H(K_2)_{K_1}$를 K_S와 비교하여 같을 경우 전송하는 사용자를 인증할 수 있게 된다. 인증 과정에서 전송받은 K_3는 다음 로그인에서 인증의 근거로 사용하기 위해 저장한다.

4. 제안 메커니즘의 분석

본 장에서는 제안한 메커니즘을 기존의 S/KEY 시스템 및 일반 패스워드 방식과 비교하여 안전성과 효율성을 검증한다.

4.1 안전성 분석

제안한 메커니즘은 기존의 S/KEY 시스템의 여러 취약점을 개선하였으며, 또한 기존 시스템의 장점을 그대로 가지고 있다. RFC-1760상에 기술된 S/KEY 시스템은 스니핑과 재연공격에 대한 1차적인 방어는 가능하나, 오프라인 사전 참조식 공격과 같은 능동적인 공격자에게는 취약점이 있다. 본 논문에서 제안한 메커니즘은 S/KEY 시스템과 달리 한 쌍의 패스워드를 사용하며 인증 과정에서의 정보가 노출되더라도 seed로부터 K_1,K_2를 생성하는 것은 매우 어렵다. 이러한 방법으로 한층 더 강력한 웹서비스에서의 인증 메커니즘을 구현할 수 있다.

다음 〈표 2〉는 일반 패스워드 방식과 OTP 시스템 및 제안한 메커니즘의 안전성을 비교하고 있다.

(1) 스니핑 공격

제안한 메커니즘의 인증 과정에서는 어떠한 비밀 정보도 전달되지 않는다. 두개의 seed는 매 인증마다 변경되며 S_S와 K_1, K_2로부터 P_1, P_2를 유추하기는 어렵다. 특히 S_N과 K_3로 다음 인증시의 일회용 패스워드(OTP)가 되는 K_{N1}, K_{N2}를 유추하기는 매우 어렵다[5].

(2) 재연 공격

S_S와 S_N은 매 인증마다 변경되며 그에 따

라 K_1, K_2의 값도 매번 달라지므로 이러한 모든 인자를 공격자가 가로채어도 재사용 할 수 없다. 본 논문에서 제안한 메커니즘의 특징으로 매번 다른 패스워드를 요구하여 재연 공격에 안전하며 여러 번 인증한 후에도 별도의 초기화 과정이나 키 교환 등의 절차 없이 영구적으로 사용할 수 있는 장점이 있다.

(3) 사전 참조식 공격

제안한 메커니즘은 S/KEY OTP 시스템과 달리 두개의 패스워드를 사용한다. 따라서 〈그림 2〉에 나타난 방식의 사전 참조식 공격이 성공할 확률은 S/KEY 시스템보다 훨씬 낮다. 아울러 패스워드에 대한 무작위 공격에도 기존의 S/KEY 시스템보다 상대적으로 안전함을 알 수 있다.

(4) 서버 침해 공격

서버에는 어떠한 비밀정보도 보관되어지지 않는다. 일방향 함수의 특성상 서버에 저장된 K_S와 S_S를 사용하여 $H(K_2)K_1 = K_S$가 될 수 있는 K_1, K_2를 계산하는 특별한 방법은 알려져 있지 않다[5]. 따라서 서버의 정보를 알아내더라도 사용자의 패스워드를 유추하는 것은 매우 어렵다.

4.2 효율성 분석

4.2.1 사용자명/패스워드 방식과의 비교분석

전통적인 패스워드 방식은 1회의 메시지 전송만으로도 인증을 받을 수 있어 사용이 편리하고 절차가 간단하다는 장점이 있으나 정보가 노출될 경우 공격자가 모든 권한을 획득할 수 있다는 치명적인 단점을 안고 있다. RFC-1760상에 기술된 S/KEY 시스템은 이러한 부분을 고려하여 여러 횟수의 해쉬 연산을 거친 OTP를 전송함으로써 이러한 위험성을 줄이고자 하였다. 그러나 S/KEY는 사용 횟수가 초기 설정한 N회로 제한이 되며, N번의 인증이 끝나면 다시 초기화가 필요하다는 불편함이 있다. 또한 충분한 사용 횟수를 위해 N을 적당한 큰 수로 잡는다면 그만큼 연산에 필요한 오버헤드가 많아지게 된다.

제안한 알고리즘은 〈표 3〉에 나타난 것과 같이 일반 패스워드 방식과 동일한 1회의 초기화 과정만 필요하며, 사용 횟수에 제한 없이 사용할 수 있다는 장점이 있다. 또한 해쉬 연산 횟수도 5회로 고정됨으로써 오버헤드에 대한 부담이 없음을 알 수 있다. 아울러 인증

〈표 3〉 각 인증 방식들의 효율성 비교

	일반 패스워드 방식	S/KEY 시스템	제안한 메커니즘
사용 횟수	제한 없음	n회	제한 없음
해쉬연산 횟수	없음	n-1회	5회
메시지 전송 횟수	1회	3회	3회
초기화 횟수	1회	다수	1회

(n : S/KEY의 일련번호)

에 필요한 메시지 전송 횟수도 S/KEY 시스템과 동일한 3회로써 추가적인 메시지 전송이 필요하지 않다. 따라서 본 논문에서 제안한 메커니즘이 기존의 S/KEY 시스템을 대치할 수 있으며, 더욱 효율적으로 작동될 수 있음을 나타내고 있다.

(1) 사용 횟수 및 초기화 횟수

S/KEY 시스템은 일련번호를 사용하여 OTP를 생성하므로 사용횟수가 초기화 과정에서 설정한 n회로 제한된다. 따라서 사용 범위를 초과하면 다시 초기화 과정을 거치게 되는 번거로움이 있으며 초기화 과정에서의 비밀 패스워드 노출에 따른 위험이 존재한다. 제안한 메커니즘은 일련번호를 사용하지 않고 인증 과정에서 임의로 생성되는 seed값으로 OTP를 연산하므로 1회의 초기화 과정만 필요하다. 따라서 제안한 메커니즘은 사용 횟수의 제한이 없으며, 재초기화 과정에 따른 비밀 정보 노출의 위험이 없다.

(2) 해쉬연산 횟수

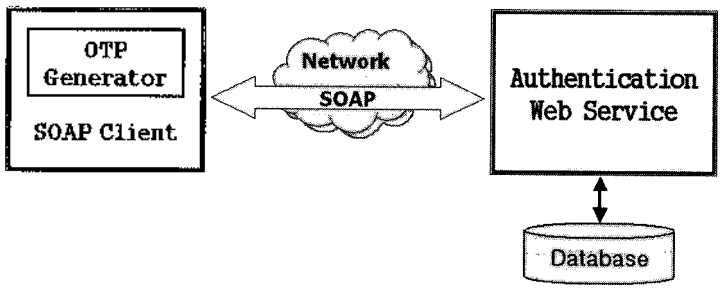
S/KEY 시스템으로 OTP를 생성하는데 필요한 연산 횟수는 n-1회이며 충분한 사용횟수를 고려한다면 큰 숫자의 n값이 필요하므로 오버헤드의 부담이 존재한다. 제안한 메커니즘은 사용횟수에 관계 없이 해쉬연산의 횟수가 5회로 고정되어 있어 오버헤드에 대한 부담이 없다.

5. XML 웹서비스 인증 메커니즘 구현

5.1 시스템 구성도

인증 메커니즘 구현을 위한 웹서비스 시스템의 구성도는 〈그림 4〉와 같다.

〈그림 4〉에 나타난 구성도는 OTP 방식의 인증 메커니즘을 구현하기 위한 웹서비스의 환경을 나타내고 있다. 사용자는 SOAP 클라이언트를 통하여 서버로부터 인증을 받기를 원하고 있으며, 인증 서버는 제안한 인증 메커니즘을 통하여 인증 절차를 수행한다. SOAP 클라이언트는 인증에 필요한 절차 및 OTP를 생성할 수 있는 메소드가 포함되어 있으며, 인증 서버에는 인증을 수행하는 메소드와 사용자를 인증하기 위해 필요한 정보를



〈그림 4〉 웹서비스 시스템의 구성도

가지고 있다. 웹서비스는 〈그림 4〉와 같이 웹을 통하여 특정 서비스를 제공하고 있으며, SOAP 클라이언트는 인증 웹서비스의 메소드를 호출하여 인증 절차에 필요한 각종 메시지를 보내고, 인증 서버는 전송받은 메시지로 사용자를 인증한다.

5.2 시스템 설계

5.2.1 클래스 다이어그램

다음 〈그림 5〉는 시스템의 클래스 다이어그램을 나타내고 있다.

각 클래스의 역할은 다음과 같다.

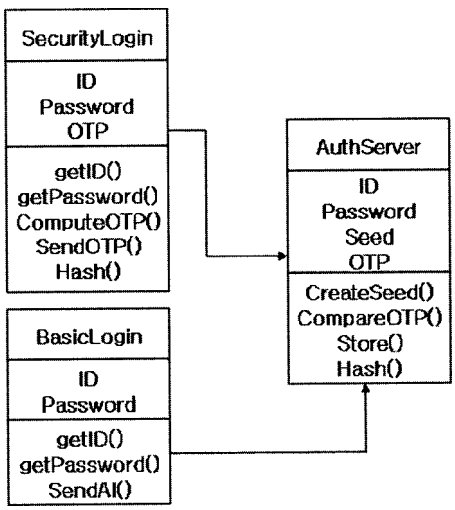
SecurityLogin : 보안 로그인을 담당하며 이 클래스 내에 OTP생성에 필요한 메소드를 포함한다.

BasicLogin : 기본 로그인을 담당하며 패스워드와 아이디를 전송하는 메소드를 포함한다.

AuthServer : 보안 로그인에서의 OTP에 대한 검증 및 기본 로그인에 대한 인증을 수행한다.

클래스 다이어그램에서 사용되는 모듈은 다음과 같다.

- getID() : 사용자의 아이디를 입력받는다.
- getPassword() : 사용자의 패스워드를 입력받는다.
- ComputeOTP() : 일회용 패스워드 메커니즘을 사용하여 OTP를 생성한다.
- BasicLogin() : 웹서비스 Authentication()을 호출하여 기본인증을 받는다.
- SecurityLogin() : 웹서비스 Authentication()을 호출하여 보안인증으로 신원을 확인한다.
- SendOTP() : 웹서비스 Authentication()에 계산된 OTP를 전달한다.
- SendAI() : 기본인증에서 인증 정보를 전달한다.



〈그림 5〉 클래스 다이어그램

- Hash() : SHA-1 : 해쉬 함수를 적용하여 그 결과를 리턴한다.
- CompareOTP(): 전송된 OTP를 저장된 값과 일치하는지 여부를 판단한다.
- Createseed() : OTP의 계산에 사용될 seed를 생성한다.
- Store() : K_3, S_N정보를 저장한다.

5.2.2 시퀀스 다이어그램

다음 〈그림 6〉은 인증 메커니즘의 시퀀스 다이어그램을 나타내고 있다.

각각의 절차는 다음과 같다.

① 사용자는 ID를 입력하고 보안 인증을 서버에 전달한다.

② 서버는 ID를 확인하고 임의의 S_N을 생성한다.

③ 생성한 S_N과 함께 S_S를 사용자에게 전달한다.

④ 사용자는 전송받은 S_N, S_S로 K_1, K_2, K_3를 계산한다.

⑤ 사용자는 서버에 K_1, K_2, K_3를 전달한다.

⑥ 서버는 미리 저장된 K_S와 K_1, K_2의 해쉬값의 동일 여부를 판단한다.

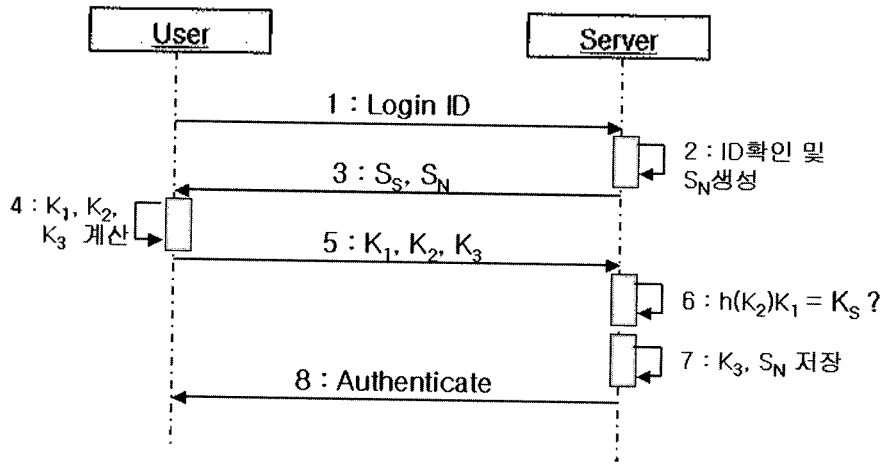
⑦ 값이 동일할 경우 K_3를 S_N과 함께 서버에 저장한다.

⑧ 사용자에게 인증되었음을 알린다.

5.3 분석 및 구현화면

5.3.1 시스템의 작동 과정

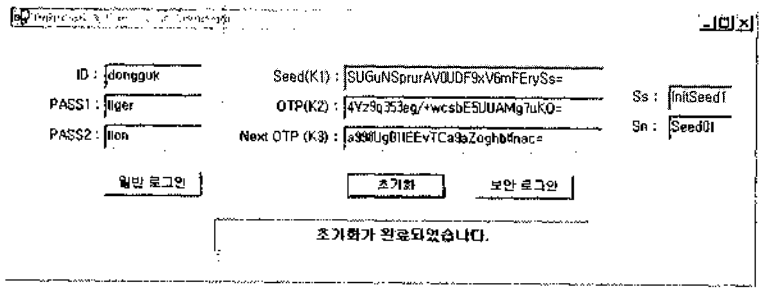
〈그림 8〉은 웹서비스 인증 시스템을 구현하기 위한 클라이언트 프로그램으로 웹서비스로부터 인증을 받고 그 결과를 출력해 주는 프로그램이다. 인증의 방법은 두 가지가 있는데 일반 로그인은 기존의 텍스트 기반으로 암호를 전달하는 방식이며, 초기화 및 보안 로그인은 제안한 메커니즘의 알고리즘을 수행하기 위한 부분이다. 사용자의 아이디와 패스



〈그림 6〉 제안한 인증 메커니즘의 시퀀스 다이어그램

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <soap:Header>
    - <AuthHeader xmlns="http://tempuri.org/">
        <UserName>dongguk</UserName>
        <Password>tiger,lion</Password>
      </AuthHeader>
    </soap:Header>
  - <soap:Body>
    - <Authentication xmlns="http://tempuri.org/">
        <act>Basic_Login</act>
      </Authentication>
    </soap:Body>
  </soap:Envelope>
```

〈그림 7〉 아이디와 비밀번호가 노출된 SOAP메세지



〈그림 8〉 초기화가 완료된 후의 화면

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <soap:Header>
    - <AuthHeader xmlns="http://tempuri.org/">
        <UserName>dongguk</UserName>
        <Password>GR3Kg5e/O5KzKiCgniDsf4oYUq1=</Password>
      </AuthHeader>
    </soap:Header>
  - <soap:Body>
    - <Authentication xmlns="http://tempuri.org/">
        <act>Initialize_Key</act>
      </Authentication>
    </soap:Body>
  </soap:Envelope>
```

〈그림 9〉 초기화 과정에서 전송된 OTP

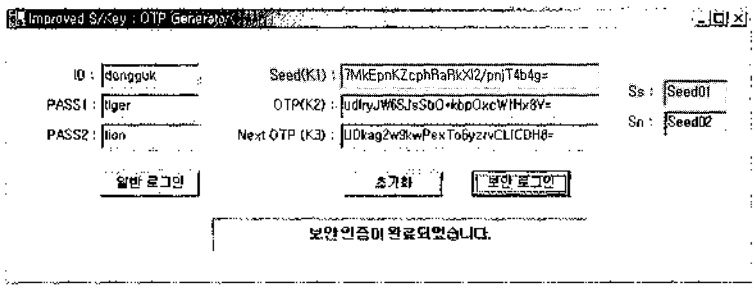
워드를 입력하고 기본 인증을 선택하면 패스워드를 그대로 전송하는 기존의 방식으로 인증하게 된다. 인증 절차가 끝난 후 클라이언트는 로그 파일을 생성하게 된다. 로그 파일상에 나타난 클라이언트와 인증 웹서비스와의 SOAP 메시지 내역은 〈그림 7〉과 같다. 이러한 경우 사용자의 아이디와 비밀번호가 노출되어 쉽게 알 수 있다.

이번에는 제안한 메커니즘을 적용하여 전송해 본다. 우선 제안한 메커니즘이 원활히 이루어질수 있도록 초기화 단계를 수행한다. 초기화를 위해서는 먼저 자신의 ID를 전송하고, 최초 seed를 전달받은 후 초기 OTP값을 계산하여 전송한다.

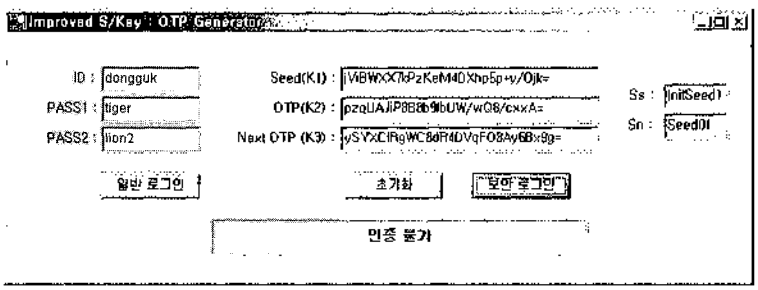
초기화 단계에서 서버에 전송한 초기 OTP는 〈그림 9〉와 같다. 서버는 Password Element 내의 패스워드 값을 임의로 생성한 최초 seed와 함께 저장한다.

보안 로그인을 클릭하면 앞서 제안한 메커니즘의 절차대로 인증 과정을 수행한다. 모든 절차를 종료한 후 〈그림 10〉과 같이 정상적으로 인증 과정이 수행되었음을 알 수 있다.

만약 유효하지 않는 OTP가 전송되었을 경우는 〈그림 11〉과 같은 화면을 만나게 된다.



〈그림 10〉 보안 로그인 상태



〈그림 11〉 인증이 되지 않는 경우

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <soap:Header>
    - <AuthHeader xmlns="http://tempuri.org/">
        <UserName>dongguk</UserName>
      </AuthHeader>
    </soap:Header>
  - <soap:Body>
    - <Authentication xmlns="http://tempuri.org/">
        <act>Initialize_Login</act>
      </Authentication>
    </soap:Body>
  </soap:Envelope>
```

〈그림 12〉 로그인 과정

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <soap:Body>
    - <AuthenticationResponse xmlns="http://tempuri.org/">
        <AuthenticationResult>TRUE,Seed02,Seed03</AuthenticationResult>
      </AuthenticationResponse>
    </soap:Body>
  </soap:Envelope>
```

〈그림 13〉 한 쌍의 seed값 전달

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <soap:Header>
    - <AuthHeader xmlns="http://tempuri.org/">
        <UserName>dongguk</UserName>
        <Password>H1s25aVGCs7vAkB/ScVE0Ap0bNQ=,ncLQ5VBd85NSUWvcrcWDPTT5B88=,
          yjF6QfBnRpc1uo6Xod4wJDKUfLM=</Password>
      </AuthHeader>
    </soap:Header>
  - <soap:Body>
    - <Authentication xmlns="http://tempuri.org/">
        <act>Security_OTP</act>
      </Authentication>
    </soap:Body>
  </soap:Envelope>
```

〈그림 14〉 클라이언트가 생성하여 전송되는 OTP

5.3.2 SOAP 메시지에 나타난 인증 절차

사용자는 〈그림 12〉와 같이 ID를 전달하게 되며, 서버는 ID로 사용자의 신원을 확인한다.

신원 확인 후 서버는 사용자에게 아래와 같은 SOAP 메시지를 보내게 된다. 사용자의 신원을 확인했음을 알리고, 이번 단계에서 사용될 두개의 seed를 함께 전송한다.

사용자는 전송받은 두개의 seed를 이용하여 K_1, K_2, K_3를 각각 계산하고 서버에 전달한다. 만약 공격자가 〈그림 14〉의 정보를 가로채더라도 사용자의 패스워드를 유추하는 것은 매우 어렵다.

서버가 전송받은 OTP를 이용하여 사용자를 최종 인증하게 되면 K_3를 저장하고 〈그림 15〉과 같이 정상적으로 인증되었음을 사용자에게 알린다.

위의 절차에서 사용자의 비밀 정보는 어느 곳에도 노출되지 않았음을 알 수 있으며, 공격자에게 어느 경로가 노출되더라도 사용자는 인증 정보에 대한 안전을 보장받을 수 있다. 또한 인증에 필요한 메시지의 전송 횟수도 S/KEY의 절차와 같은 3회로써 연산 및 메시지의 전송에 관한 부담이 적고 효율적인 메커니즘인 것을 알 수 있다.

6. 결 론

XML 웹서비스는 분산 어플리케이션의 특성상 강력한 정보 보호가 필요하다. 현재 XML 웹서비스에 필요한 여러 보안 표준들이 나와 있으나 SOAP 표준 규격에는 보안에 대한 언급이 되어 있지 않아 제3자에 의해 비밀번호와 같은 중요한 개인정보가 유출될 가능성이 존재한다. 본 논문에서는 인증에 관한 취약점을 보완할 수 있는 방법을 제안하였다. 즉, S/KEY 시스템의 단점을 보완한 새로운 인증 메커니즘을 제안하고 웹서비스의 핵심 프로토콜인 SOAP에 적용하였으며 실제 시스템을 구현하였다. 아울러 해당 시스템의 클라이언트와 서버간의 통신 과정에서 제안한 메커니즘에 대한 동작 및 인증 과정을 SOAP 메시지를 통하여 살펴보았다.

본 논문의 연구결과로 SOAP Header내의 인증정보를 새롭게 제안한 메커니즘을 적용하여 전송함으로써 한층 더 간편하고 안전한 인증을 제공할 수 있다. 제안한 메커니즘은 RFC-1760상에 언급된 S/KEY 시스템의 장점을 그대로 수용하면서도 취약점들을 보완하여 웹서비스 상의 안전하고 효율적인 인증을 가능하게 한다.

본 논문의 향후 과제로서 SOAP 프로토콜에 대한 보안상 문제점의 검토 및 WS-Security 규격상에 기술된 인증 방식들에 대한 비교 평가를 진행하고, 또한 제안한 방법을 표준 규격에 구체적으로 적용하는 방법에 대해 연구하고자 한다.

참 고 문 헌

[1] 정지훈, "웹서비스", 한빛미디어, 2002

[2] N. Haller, "The S/KEY One-Time Password System", RFC 1760, 1995

[3] KISA, "일회용 패스워드 기술", http://www.kisa.or.kr/technology/sub4/password.htm

[4] N. Haller/Philip R. Karn, "Description of The S/KEY One-Time Password System", http://www.irisa.fr/atelier/skey.txt

[5] Val Henson/Richard Henderson, "Guidelines for Using Compare-by-hash", http://infohost.nmt.edu/~val/review/hash2.pdf

[6] NIST, "Announcing the Secure Hash Standard", FIPS 180-1, 1995

[7] Mudge, "Vulnerabilities in the S/KEY one time password system", http://www.edelweb.fr/skeyflaws.html

[8] W3C, "Simple Object Access Protocol (SOAP) 1.1", http://www.w3.org/TR/SOAP/

[9] IBM, "Web Services Security (WS-Security)", http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

[10] IBM, "WS-Security AppNotes", http://www-106.ibm.com/developerworks/library/ws-secapp/

저 자 소 개

송유진 (E-mail : song@dongguk.ac.kr)

1982. 한국항공대학교 졸업(학사)

1987. 경북대학교 대학원 졸업(석사)

1995. 일본 Tokyo Institute of Technology 졸업(박사)

1988. ~ 1996. 한국전자통신연구원 선임연구원

2003. 12~2005. 2 미국 University of North Carolina at Charlotte 연구교수

1996. ~ 현재 동국대학교 전자상거래학과/대학원 교수

2005. ~ 현재 동국대학교 부설 전자상거래연구소 소장

1998. ~ 현재 한국정보보호학회 이사

1997. ~ 현재 한국정보시스템학회 이사

2001. ICISC2001 운영위원장 역임

2003. 하계CISC2003 프로그램 위원장

관심 분야 전자상거래응용 보안 (Ubiquitous/Web Service Privacy, Location Privacy, 디지털컨텐츠 보호, XML보안, SCM/CRM 보안 등), Context Aware Application Security

이동혁 (E-mail: jazzbop@korea.com)

2004. 동국대학교 전자상거래학과 학사

2005. ~ 현재 동국대학교 전자상거래학과 석사과정

관심 분야 XML 보안, 유비쿼터스/웹서비스 프라이버시 보호, 전자상거래 보안