자바스크립트 암호 프로그래밍

JAVASCRIPT CRYPTOGRAPHY PROGRAMMING

2017. 3.

중부대학교 정보보호학과 이병천 교수

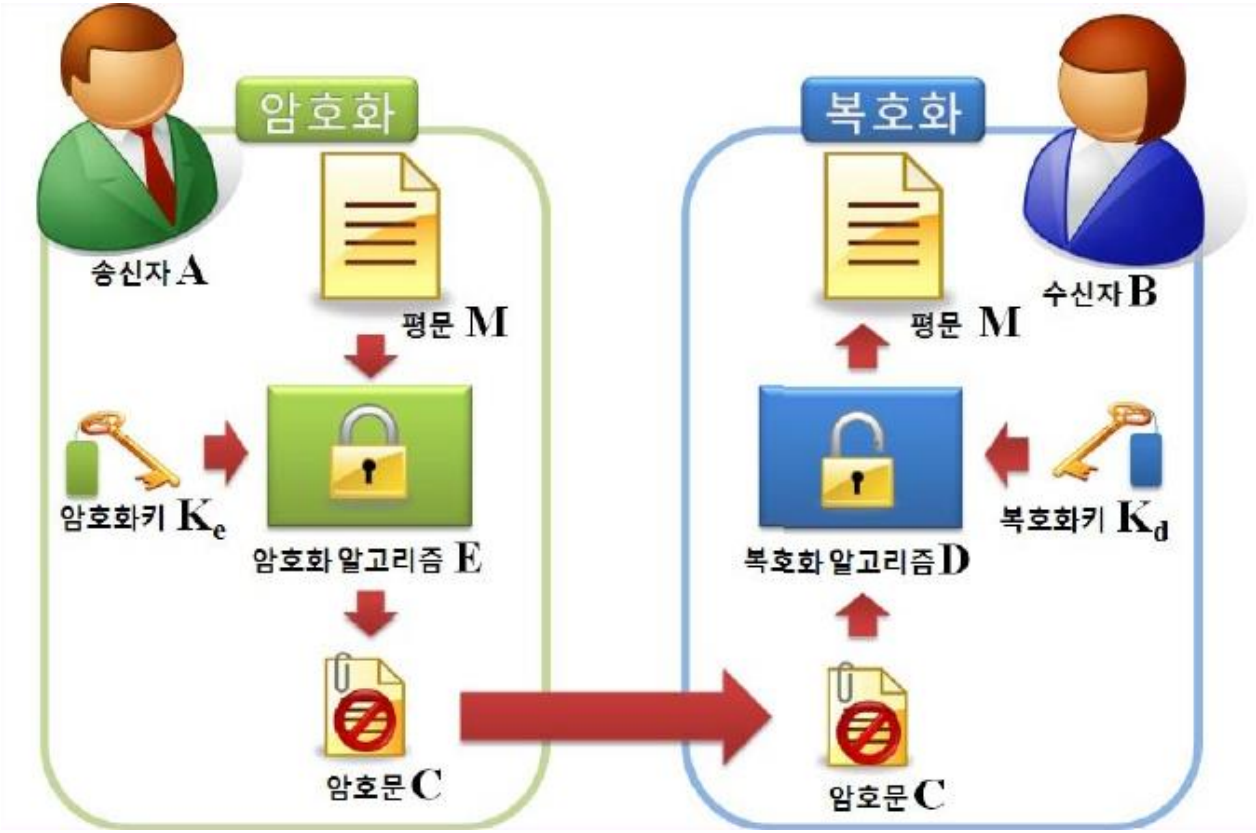
차례

1. 암호와 정보보호

1.1 암호의 개념

- 암호(cryptography)
  - 암호의 어원: 그리스어 Cryptos에서 유래
  - 평문을 해독 불가능한 형태로 변형하거나 또는 암호화된 통신문을 원래의 해독 가능한 상태로 복구하는 것
- 현대 암호가 제공해야 하는 정보보호 기능
  - 기밀성: 정보를 타인이 보지 못하도록 감춤 (암호화)
  - 무결성: 정보의 불법 변조 여부를 확인 (해쉬함수, 메시지인증코드)
  - 부인방지: 사용자의 행위를 부인하지 못함 (전자서명)
  - 인증, 접근통제: 사용자의 신분을 확인하고 정보에 대한 적법한 권한을 부여
- 암호학
  - 암호학이란 이와 같은 기능들을 구현하기 위한 모든 수학적인 원리, 수단, 방법 등의 기반기술을 말함
  - 암호를 사용하지 않고 궁극적인 정보보호를 성취하는 것은 불가능

암호 시스템의 구성



송신자
(sender)

도청자(eavesdropper)
공격자(attacker)

수신자
(receiver)

암호 관련 용어 설명

- 도청자(eavesdropper)
- 공격자(attacker)
- 암호설계(cryptography)
- 암호해독(cryptanalysis)
- 수동공격(passive attack)
- 능동공격(active attack)

암호의 여러가지 방식

- 대칭키 암호 (비밀키 암호)
  - 암호화와 복호화 알고리즘에 동일한 키가 사용되는 방식의 암호
  - 비밀키는 제3자에게 알려지면 안되므로 송신자와 수신자는 사용되는 키를 비밀리에 공유하고 안전하게 보관해야 한다.
- 블록 암호
  - 대칭키 암호의 일종으로 암호화와 복호화시 특정 크기의 블록 단위로 암호화/복호화 연산을 하는 방식의 암호
  - 각 블록의 암호화와 복호화에는 동일한 키가 사용됨
- 스트림 암호
  - 블럭의 크기를 1로 하여 블록 마다 각각 다른 키를 사용하여 암호문을 생성하는 방식
  - 암호화와 복호화시 키스트림 생성기를 이용하여 키스트림을 생성하며 이것을 평문과 연산하여 암호화하고(송신자) 거꾸로 이것을 암호문과 연산하여 평문을 얻어낸다(수신자).

암호의 여러가지 방식

- 비대칭키 암호(공개키 암호)
  - 하나의 쌍이 되는 두 개의 키를 생성하여 하나는 암호화에 사용하고 다른 하나는 복호화에 사용한다.
  - 암호화에 사용하는 키는 공개할 수 있어서 공개키라고 부르고 복호화에 사용하는 키는 사용자만이 안전하게 보관해야 하는 키로 개인키(비밀키)라고 부른다.
  - 두 개의 키가 서로 다르므로 비대칭키 암호라고 부르며 하나의 키를 공개하므로 공개키 암호라고도 부른다.
  - 공개키를 공개하더라도 이것으로부터 개인키를 계산해내는 것은 수학적으로 매우 어려운 문제이다.
  - 공개키와 쌍이 되는 개인키는 반드시 존재하므로 이것을 찾아내는 것이 불가능한 것은 아니다. 찾아내기 어려울 뿐이다.

암호의 여러가지 방식

- 해쉬함수
  - 임의의 길이의 정보(비트스트링)를 입력으로 하여 고정된 길이의 출력값인 해쉬코드를 생성해내는 함수
  - 해쉬값은 입력정보에 대한 변조할 수 없는 특징값을 나타내며 통신 중에 정보의 변조가 있었는지 여부를 확인하는 용도에 사용된다.
  - 이런 용도로 사용될 수 있기 위해서 해쉬함수는 같은 해쉬값을 가지는 두 개의 입력 메시지를 찾는 것이 계산적으로 불가능해야 한다.
  - 해쉬함수에는 키를 사용하지 않는다. 주어진 정보에 대한 해쉬값은 누구나 계산할 수 있다.

암호의 여러가지 방식

- 메시지인증코드 (MAC: Message Authentication Code)
  - 데이터가 변조(수정, 삭제, 삽입 등)되었는지를 여부를 검증할 수 있도록 데이터에 덧붙이는 코드
  - 원래의 데이터로만 생성할 수 있는 값을 데이터에 덧붙여서 확인하도록 하는 것
  - 송신자와 수신자는 비밀키를 공유하고 있으며 MAC 계산에 비밀키를 사용한다. 그러므로 송신자와 수신자만이 MAC을 계산하고 검증할 수 있다.

암호의 여러가지 방식

- 패스워드 기반 키생성
  - 사용자가 입력하는 패스워드를 직접 비밀키로 사용하는 것은 고정된 키를 사용하게 되어 사전공격 등의 방법이 가능하므로 보안성에 문제가 많다.
  - 서버에 사용자의 패스워드를 직접 저장하게 되면 관리자가 사용자의 패스워드를 알게 되고 외부 공격자의 해킹에 대한 위협도 있어서 피해야 한다.
  - 사용자가 입력하는 패스워드에 의존하면서도 난수 특성을 갖는 키를 생성하여 사용할 필요가 있다.
  - 패스워드기반 키생성함수 PBKDF2() 이용

암호의 여러가지 방식

- 전자서명
  - 전자문서에 대한 전자적인 방식에 의한 서명으로 서명자만이 생성할 수 있고 누구나 서명의 유효성을 검증할 수 있다.
  - 전자서명은 공개키 암호 방식의 일종이다.
  - 개인키로 서명을 생성하고 공개키로 서명을 검증한다.
  - 개인키는 해당 서명자만이 가지고 있으므로 다른 사람이 서명을 위조할 수 없으며, 전자서명은 서명자의 정당한 서명으로 인정된다. 유효한 서명에 대해 서명자는 자신이 서명한 사실을 부인할 수 없다(부인방지 기능).

암호의 여러가지 방식

- 키합의
  - 공개키 암호는 속도가 느리기 때문에 실제의 데이터를 암호화, 복호화하는 용도에는 사용하지 않는다.
  - 송신자와 수신자가 비밀키 암호를 사용하기 위해서는 미리 비밀키를 공유하거나 안전한 통신 채널을 사용하여 세션키를 전송하는 것이 필요하다.
  - 송신자와 수신자가 직접 만나지 않고도 공개된 통신채널을 통해서 특정한 방법으로 세션키를 안전하게 공유하는 방식을 키합의라고 한다.

1.2 정보보호를 위한 암호의 역할

- 정보화 사회란?
  - 정보의 축적, 처리, 전송 능력이 획기적으로 증대되면서 정보의 가치가 물질이나 에너지 이상으로 중요해지는 사회
  - 정보시스템, 정보서비스에 크게 의존하는 사회
  - 정보가 상품으로서의 가치를 인정받아 시장에서 유통되는 사회

- 정보보호의 중요성
  - 정보시스템의 오류, 마비로 인한 피해
  - 정보의 불법적 노출시 피해, 개인정보를 이용한 사기
  - 해킹공격에 의한 피해
  - 산업스파이
  - 정보전

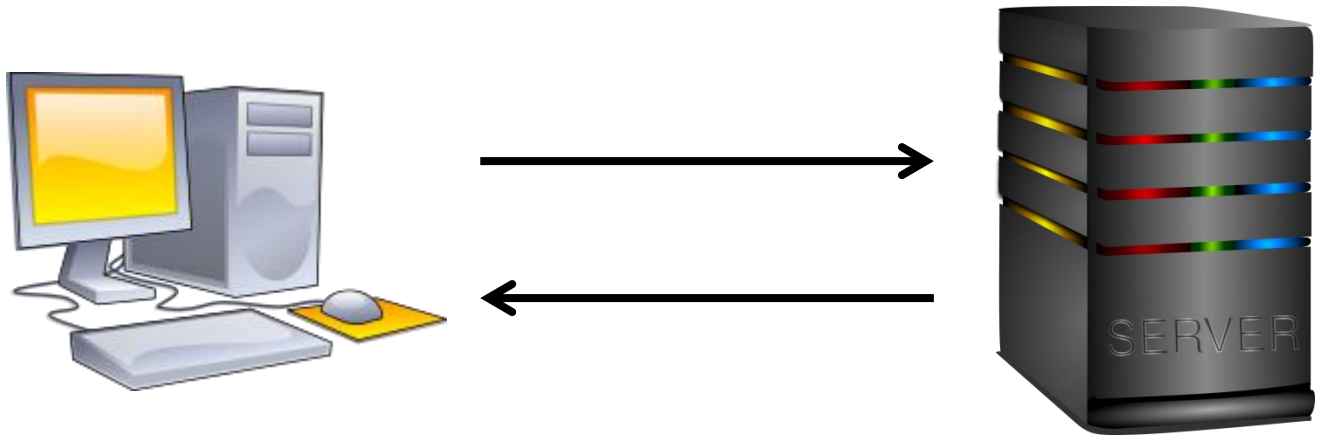
암호와 정보보호의 관계

- 정보시스템이 제공해야 하는 정보보호 기능
  - 기밀성(Confidentiality) : 정보를 볼 수 있는 권한을 가지지 않은 사람이 정보에 접근하지 못하도록 제한을 가함.
  - 무결성(Integrity) : 정보가 권한이 없는 사람에 의해서 조작되거나 훼손되지 않도록 함.
  - 가용성(Availability) : 권한을 가진 사람은 원할 때 정보를 사용할 수 있어야 함.
  - 인증, 접근제어: 사용자의 신분을 확인하고 그에 맞는 권한을 부여함
- 암호기술은 정보보호를 제공하기 위한 수학적 기반기술
- 인터넷과 같은 공개된 통신망을 사용하면서 암호기술 없이 이런 정보보호의 목표를 제공하는 것은 불가능

2. Javascript Cryptography

- 여러 가지 라이브러리들이 존재
  - http://cryptojs.altervista.org/test/simulate-threading-speed_test.html
- CryptoJS
  - 구글에서 개발
  - https://github.com/sytelus/CryptoJS
- Forge
  - 인증서 기반 TLS 기능을 제공
  - http://digitalbazaar.github.io/forge/
- Web Cryptography API
  - 웹브라우저에 기본 내장된 웹 암호 표준
  - 웹 기술 표준화 기구인 W3C에서 표준화
  - https://www.w3.org/TR/WebCryptoAPI/ 참조

웹 암호 기술



클라이언트(브라우저)측 암호
- 자바스크립트

웹서버측 암호
- PHP
- Javascript (node.js)
- Java (JSP)
- 파이썬 (Django)

해쉬함수

- 해쉬함수란?
  - 임의의 길이의 입력메시지에 대하여 고정된 길이의 특징값(해쉬값)을 계산해내는 함수. 키가 사용되지 않으므로 입력메시지가 같으면 동일한 해쉬값을 출력.
- 해쉬함수는 다음과 같은 특성을 만족시켜야 한다.
  - 일방향성: 입력메시지로부터 해쉬값을 계산하는 것은 쉽지만 출력 해쉬값으로부터 그 해쉬값을 출력하는 입력메시지를 찾는 것은 어렵다.
  - 충돌회피성: 같은 해쉬값을 출력하는 두개의 입력메시지를 찾아내는 것은 어렵다.

해쉬함수 예제


```
<script src="CryptoJS/rollups/md5.js"></script>
<script src="CryptoJS/rollups/sha1.js"></script>
<script src="CryptoJS/rollups/sha256.js"></script>
<script src="CryptoJS/rollups/sha512.js"></script>
<script src="CryptoJS/rollups/sha3.js"></script>
<script src="CryptoJS/ripemd160.js"></script>

<script>
    function hash()
    {
        var msg = document.getElementById("message").value;
        var rst = document.getElementById("result");
        var select = document.getElementById("item");
        var option_value = select.options[select.selectedIndex].value;
        if (option_value==1) // MD5
            rst.value = CryptoJS.MD5(msg);
        else if (option_value==2) // SHA1
            rst.value = CryptoJS.SHA1(msg);
        else if (option_value==3) // SHA256
            rst.value = CryptoJS.SHA256(msg);
        else if (option_value==4) // SHA512
            rst.value = CryptoJS.SHA512(msg);
        else if (option_value==5) // SHA3
            rst.value = CryptoJS.SHA3(msg);
        else if (option_value==6) // RIPEMD160
            rst.value = CryptoJS.RIPEMD160(msg);
    }
</script>
```

CryptoJS.SHA1(msg);

http://cris.joongbu.ac.kr/course/2017-1/wp/crypto/hash.html

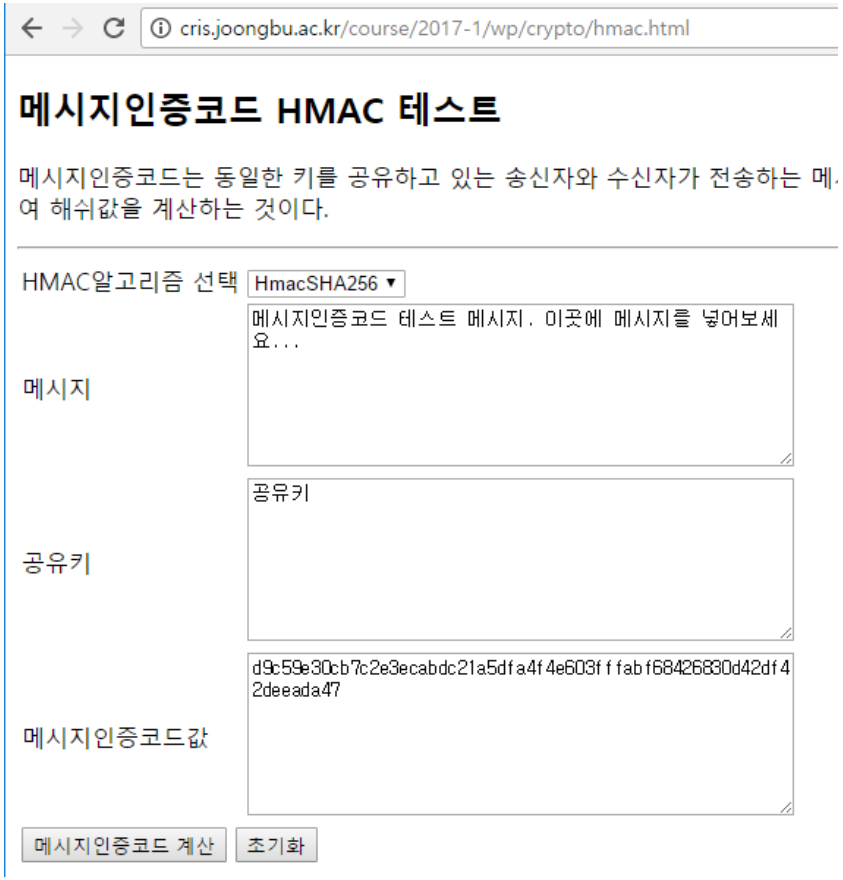
메시지인증코드

- 메시지인증코드
  - 메시지인증코드는 동일한 키를 공유하고 있는 송신자와 수신자가 전송하는 메시지의 인증성을 상대방에게 확인시키기 위해서 계산하는 값.
  - 전송하는 메시지와 공유한 키를 입력으로 하여 해쉬값을 계산
  - HMAC – 해쉬함수를 이용한 메시지인증코드 표준

메시지인증코드 예제


```
<script src="CryptoJS/rollups/hmac-md5.js"></script>
<script src="CryptoJS/rollups/hmac-sha1.js"></script>
<script src="CryptoJS/rollups/hmac-sha256.js"></script>
<script src="CryptoJS/rollups/hmac-sha512.js"></script>

<script>
    function hmac()
    {
        var msg = document.getElementById("message").value;
        var key = document.getElementById("key").value;
        var rst = document.getElementById("result");
        var select = document.getElementById("item");
        var option_value = select.options[select.selectedIndex].value;
        if (option_value==1) // HmacMD5
            rst.value = CryptoJS.HmacMD5(msg,key);
        else if (option_value==2) // HmacSHA1
            rst.value = CryptoJS.HmacSHA1(msg,key);
        else if (option_value==3) // HmacSHA256
            rst.value = CryptoJS.HmacSHA256(msg,key);
        else if (option_value==4) // HmacSHA512
            rst.value = CryptoJS.HmacSHA512(msg,key);
    }
</script>
```



CryptoJS.HmacSHA1(msg,key);

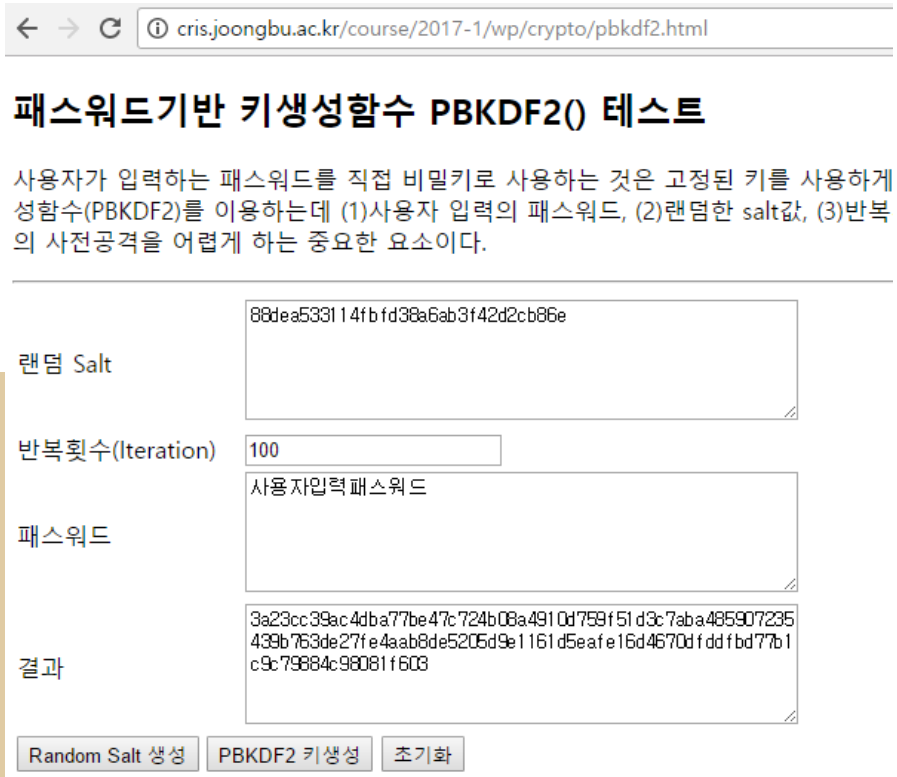
http://cris.joongbu.ac.kr/course/2017-1/wp/crypto/hmac.html

패스워드기반 키생성

- 필요성
  - 사용자가 입력하는 패스워드를 직접 비밀키로 사용하는 것은 고정된 키를 사용하게 되어 사전공격 등의 방법이 가능하므로 보안성에 문제가 많다.
  - 사용자가 입력하는 패스워드에 의존하면서도 난수 특성을 갖는 키를 생성하여 사용할 필요가 있다.
- 패스워드기반 키생성함수 PBKDF2()
  - Password-based Key Derivation Function
  - (1)사용자 입력 패스워드, (2)랜덤한 salt값, (3)반복횟수(iteration) 값을 이용하여 난수처럼 보이는 암호키를 생성하여 사용
  - salt값과 반복횟수값은 공격자의 사전공격을 어렵게 하는 중요한 요소이다.

패스워드기반 키생성 예제


```
<script src="CryptoJS/rollups/pbkdf2.js"></script>
<script>
    function randomSalt()
    {
        var s = document.getElementById("salt");
        s.value = CryptoJS.lib.WordArray.random(128/8);
    }
    function PBKDF2()
    {
        var s = document.getElementById("salt").value;
        var l = document.getElementById("iteration").value;
        var p = document.getElementById("pass").value;
        var r = document.getElementById("result");
        r.value = CryptoJS.PBKDF2(p, s, { keySize: 512/32, iterations: l });
    }
</script>
```



http://cris.joongbu.ac.kr/course/2017-1/wp/crypto/pbkdf2.html

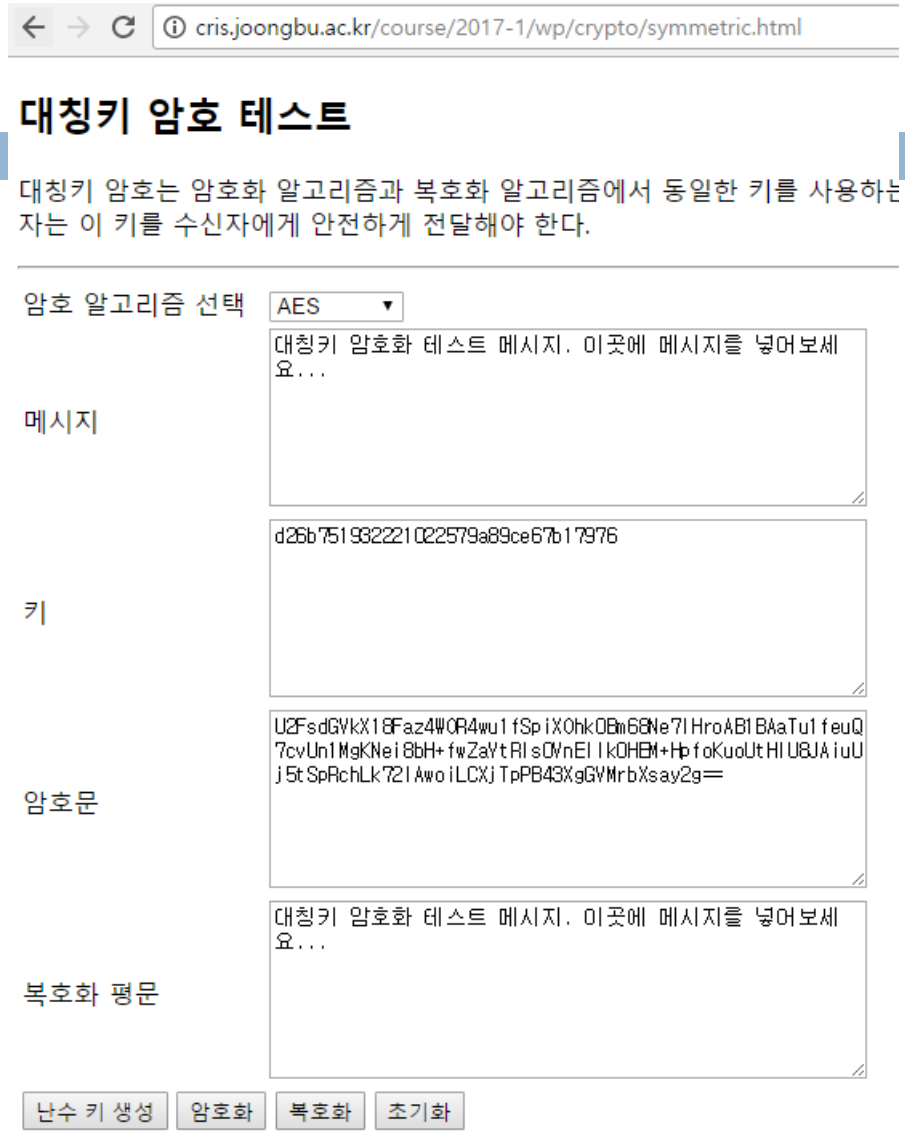
대칭키 암호

- 대칭키 암호
  - 암호화 알고리즘과 복호화 알고리즘에서 동일한 키를 사용하는 알고리즘
  - 송신자는 일반적으로 난수생성함수를 이용하여 임의로 생성한 키를 사용하여 암호화하며 송신자는 이 키를 수신자에게 안전하게 전달해야 함
- AES
  - 블록암호 국제 표준

대칭키 암호 예제

암호화
CryptoJS.AES.encrypt(msg,key);

복호화
CryptoJS.AES.decrypt(encrypted,key)

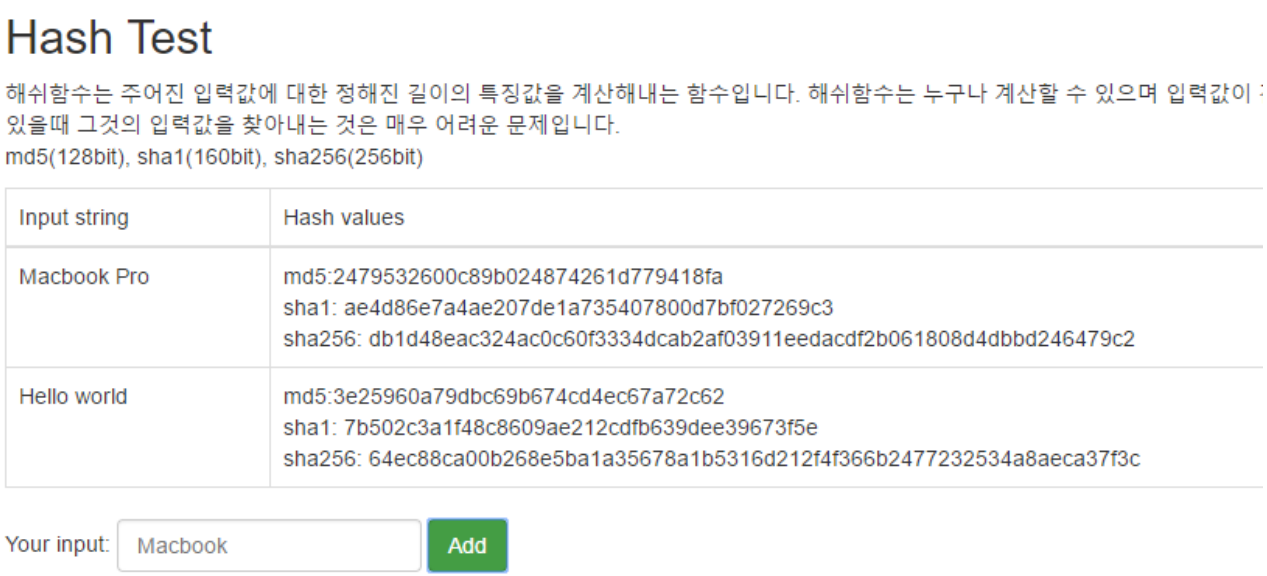


http://cris.joongbu.ac.kr/course/2017-1/wp/crypto/symmetric.html

Forge

- Forge
  - 인증서 기반 TLS 기능을 제공
  - http://digitalbazaar.github.io/forge/
- 예제

  http://cris.joongbu.ac.kr/course/2017-1/wp/crypto/forge/index.html



Hash Test

해쉬함수는 주어진 입력값에 대한 정해진 길이의 특징값을 계산해내는 함수입니다. 해쉬함수는 누구나 계산할 수 있으며 입력값이
있을때 그것의 입력값을 찾아내는 것은 매우 어려운 문제입니다.
md5(128bit), sha1(160bit), sha256(256bit)

Input string	Hash values
Macbook Pro	md5:2479532600c89b024874261d779418fa sha1: ae4d86e7a4ae207de1a735407800d7bf027269c3 sha256: db1d48eac324ac0c60f3334dcab2af03911eedacdf2b061808d4dbbd246479c2
Hello world	md5:3e25960a79dbc69b674cd4ec67a72c62 sha1: 7b502c3a1f48c8609ae212cdfb639dee39673f5e sha256: 64ec88ca00b268e5ba1a35678a1b5316d212f4f366b2477232534a8aeca37f3c

Your input: Macbook Add

Forge

RSA Self-certificate Generation

공인인증서는 인증기관이 발급하는 것으로 사용자의 신분을 확인한 후 사용자가 제출하는 공개키에 대해 인증기관의 개인키를 이
자체인증서는 사용자가 스스로 생성하는 인증서로 자신의 키쌍을 생성한 후 자신의 개인키로 자신의 공개키에 대해 서명하여 인증
인증서를 발급하기 위해서는 다음 정보를 입력하시오.

Field	Input Your Information
Country Name	KR
Province Name	경기도
Locality Name	고양시
Organization Name	중부대학교
Organization Unit Name	정보보호학과
Common Name	홍길동
Email Address	hgd@joongbu.ac.kr

Key Length: 1024 bits ▾ Certificate Generation

Public Key

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCzexF1xynh3T85oHJimfLTDGT8
321IDkrKcnUBWAeMVzn4IP5wP39qRne5h6fxfX4kM2ys3hWMa3QIegtFXJOAaZdQ
Zuw5PlDoBtHOamT3ryFOgguc2EEhuCK8/ohWwHh1TzjI3MHEaQ+1ItwGOgPqApm4
WH7HhInMqv6AwnpiCwIDAQAB
-----END PUBLIC KEY-----
```

Web Cryptography API

- Web Cryptography API
  - 웹브라우저에 기본 내장된 웹 암호 표준
  - 웹 기술 표준화 기구인 W3C에서 표준화
  - https://www.w3.org/TR/WebCryptoAPI/ 참조
  - https 환경에서만 사용
- 브라우저에 기본 내장된 window.crypto 객체 이용

```
// SHA-1 - digest
window.crypto.subtle.digest(
    { name: "SHA-1",  },
    u8a
    // new Uint8Array([1,2,3,4]) //The data you want to hash as an ArrayBuffer
)
.then(function(hash){
    //returns the hash as an ArrayBuffer
    result_hash += "plaintext: "+str+"<br>";
    var dv = buf2hex(hash);
    console.log(new Uint8Array(hash));
    result_hash += "SHA-1: "+dv+"<br>";
})
.catch(function(err){
    console.error(err);
});
```