HMAC_SHA256加密方法

Javascript HMAC SHA256

Run the code online with this jsfiddle. Dependent upon an open source js library called http://code.google.com/p/crypto-js/.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/crypto-js.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/hmac-sha256.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/enc-base64.min.js"></script>

<script>
  var hash = CryptoJS.HmacSHA256("Message", "secret");
  var hashInBase64 = CryptoJS.enc.Base64.stringify(hash);
  document.write(hashInBase64);
</script>
```

PHP HMAC SHA256

PHP has built in methods for hash_hmac (PHP 5) and base64_encode (PHP 4, PHP 5) resulting in no outside dependencies. Say what you want about PHP but they have the cleanest code for this example.

```
$s = hash_hmac('sha256', 'Message', 'secret', true);
echo base64_encode($s);
```

Java HMAC SHA256

Dependent on Apache Commons Codec to encode in base64.

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;

public class ApiSecurityExample {
  public static void main(String[] args) {
    try {
     String secret = "secret";
     String message = "Message";

     Mac sha256_HMAC = Mac.getInstance("HmacSHA256");
     SecretKeySpec secret_key = new SecretKeySpec(secret.getBytes(), "HmacSHA256");
     sha256_HMAC.init(secret_key);

     String hash = Base64.encodeBase64String(sha256_HMAC.doFinal(message.getBytes()));
     System.out.println(hash);
    }
    catch (Exception e){
     System.out.println("Error");
    }
   }
}
```

Groovy HMAC SHA256

It is mostly java code but there are some slight differences. Adapted from Dev Takeout - Groovy HMAC/SHA256 representation.

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.security.InvalidKeyException;

def hmac_sha256(String secretKey, String data) {
 try {
    Mac mac = Mac.getInstance("HmacSHA256")
    SecretKeySpec secretKeySpec = new SecretKeySpec(secretKey.getBytes(), "HmacSHA256")
    mac.init(secretKeySpec)
    byte[] digest = mac.doFinal(data.getBytes())
    return digest
   } catch (InvalidKeyException e) {
    throw new RuntimeException("Invalid key exception while converting to HMac SHA256")
  }
}

def hash = hmac_sha256("secret", "Message")
encodedData = hash.encodeBase64().toString()
log.info(encodedData)
```

C# HMAC SHA256

```
using System.Security.Cryptography;

namespace Test
{
  public class MyHmac
  {
    private string CreateToken(string message, string secret)
    {
      secret = secret ?? "";
      var encoding = new System.Text.ASCIIEncoding();
      byte[] keyByte = encoding.GetBytes(secret);
      byte[] messageBytes = encoding.GetBytes(message);
      using (var hmacsha256 = new HMACSHA256(keyByte))
      {
        byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
        return Convert.ToBase64String(hashmessage);
      }
    }
  }
}
```

Objective C and Cocoa HMAC SHA256

Most of the code required was for converting to bae64 and working the NSString and NSData data types.

```
#import "AppDelegate.h"
#import <CommonCrypto/CommonHMAC.h>

@implementation AppDelegate

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
 NSString* key = @"secret";
 NSString* data = @"Message";

 const char *cKey = [key cStringUsingEncoding:NSASCIIStringEncoding];
 const char *cData = [data cStringUsingEncoding:NSASCIIStringEncoding];
 unsigned char cHMAC[CC_SHA256_DIGEST_LENGTH];
 CCHmac(kCCHmacAlgSHA256, cKey, strlen(cKey), cData, strlen(cData), cHMAC);
 NSData *hash = [[NSData alloc] initWithBytes:cHMAC length:sizeof(cHMAC)];

 NSLog(@"%@", hash);

 NSString* s = [AppDelegate base64forData:hash];
 NSLog(s);
}

+ (NSString*)base64forData:(NSData*)theData {
 const uint8_t* input = (const uint8_t*)[theData bytes];
 NSInteger length = [theData length];

 static char table[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";

 NSMutableData* data = [NSMutableData dataWithLength:((length + 2) / 3) * 4];
 uint8_t* output = (uint8_t*)data.mutableBytes;

 NSInteger i;
 for (i=0; i < length; i += 3) {
 NSInteger value = 0;
 NSInteger j;
 for (j = i; j < (i + 3); j++) {
 value <<= 8;

 if (j < length) {  value |= (0xFF & input[j]);  }  }  NSInteger theIndex = (i / 3) * 4;  output[theIndex +
0] = table[(value >> 18) & 0x3F];
 output[theIndex + 1] = table[(value >> 12) & 0x3F];
 output[theIndex + 2] = (i + 1) < length ? table[(value >> 6) & 0x3F] : '=';
 output[theIndex + 3] = (i + 2) < length ? table[(value >> 0) & 0x3F] : '=';
 }

 return [[NSString alloc] initWithData:data encoding:NSASCIIStringEncoding]; }

@end
```

Go programming language - Golang HMAC SHA256

- Try it online in your browser with Play GoLang
- crypto/hmac package

```go
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
    "fmt"
)

func ComputeHmac256(message string, secret string) string {
    key := []byte(secret)
    h := hmac.New(sha256.New, key)
    h.Write([]byte(message))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}

func main() {
    fmt.Println(ComputeHmac256("Message", "secret"))
}
```

Ruby HMAC SHA256

Requires openssl and base64.

```ruby
require 'openssl'
require "base64"

hash  = OpenSSL::HMAC.digest('sha256', "secret", "Message")
puts Base64.encode64(hash)
```

Python (2.7) HMAC SHA256

```python
import hashlib
import hmac
import base64

message = bytes("Message").encode('utf-8')
secret = bytes("secret").encode('utf-8')

signature = base64.b64encode(hmac.new(secret, message, digestmod=hashlib.sha256).digest())
print(signature)
```

Tested with Python 2.7.6. Also, be sure not to name your python demo script the same as one of the imported libraries.

Python (3.7) HMAC SHA256

```python
import hashlib
import hmac
import base64

message = bytes('Message', 'utf-8')
secret = bytes('secret', 'utf-8')

signature = base64.b64encode(hmac.new(secret, message, digestmod=hashlib.sha256).digest())
print(signature)
```

Tested with Python 3.7.0. Also, be sure not to name your python demo script the same as one of the imported libraries. Thanks to @biswap

anda.

Perl HMAC SHA256

See Digest::SHA documentation. By convention, the Digest modules do not pad their Base64 output. To fix this you can test the length of the hash and append equal signs "=" until it is the length is a multiple of 4. We will use a modulus function below.

```
use Digest::SHA qw(hmac_sha256_base64);
$digest = hmac_sha256_base64("Message", "secret");

# digest is currently: qnR8UCqJggD55PohusaBNviGoOJ67HC6Btry4qXLVZc

# Fix padding of Base64 digests
while (length($digest) % 4) {
    $digest .= '=';
}

print $digest;
# digest is now: qnR8UCqJggD55PohusaBNviGoOJ67HC6Btry4qXLVZc=
```

Dart HMAC SHA256

Dependent upon the Dart crypto package.

```
import 'dart:html';
import 'dart:convert';
import 'package:crypto/crypto.dart';

void main() {

  String secret = 'secret';
  String message = 'Message';

  List<int> secretBytes = UTF8.encode('secret');
  List<int> messageBytes = UTF8.encode('Message');

  var hmac = new HMAC(new SHA256(), secretBytes);
  hmac.add(messageBytes);
  var digest = hmac.close();

  var hash = CryptoUtils.bytesToBase64(digest);

  // output to html page
  querySelector('#hash').text = hash;
  // hash => qnR8UCqJggD55PohusaBNviGoOJ67HC6Btry4qXLVZc=
}
```

Swift HMAC SHA256

I have not verified but see this stackOverflow post

Rust

Take a look at the alco/rust-digest repository for Rust (lang) guidance. I have not verified yet.

Powershell (Windows) HMAC SHA256

Mostly wrapping of .NET libraries but useful to see it in powershell's befuddling syntax. See code as gist

```
$message = 'Message'
$secret = 'secret'

$hmacsha = New-Object System.Security.Cryptography.HMACSHA256
$hmacsha.key = [Text.Encoding]::ASCII.GetBytes($secret)
$signature = $hmacsha.ComputeHash([Text.Encoding]::ASCII.GetBytes($message))
$signature = [Convert]::ToBase64String($signature)

echo $signature

# Do we get the expected signature?
echo ($signature -eq 'qnR8UCqJggD55PohusaBNviGoOJ67HC6Btry4qXLVZc=')
```

Shell (Bash etc) HMAC SHA256

Using openssl. Credit to @CzechJiri

```
MESSAGE="Message"
SECRET="secret"

echo -n $MESSAGE | openssl dgst -sha256 -hmac $SECRET -binary | base64
```

Delphi HMAC SHA256

https://stackoverflow.com/a/40182566/215502