SHA1 Hash hesaplanması

Bilgilerin güvenliği için iletişimde kullanılan parametrelere bir anahtar ilave edilerek SHA1 hash değeri hesaplanmaktadır.

İşlem başlatma için hash hesaplanması

Bu işlem için
SHA1(clientid + oid + amount + okUrl + failUrl + islemtipi + taksit + rnd + işyerianahtarı)
fonksiyonu kullanılmaktadır.

Örnek:

clientid	31234
oid	gl3p5uy
amount	9.95
okUrl	https://magazam.com.tr/odemesayfasi
failUrl	https://magazam.com.tr/hatasayfasi
islemtipi	Auth
taksit	4
rnd	yHK84HjUd0
işyeri anahtarı	123456

olduğu durumda hash alınacak değer, bu alanların arka arkaya eklenmiş hali

31234gl3p5uy9.95https://magazam.com.tr/odemesayfasihttps://magazam.com.tr/hatasayfasiAuth4yHK84HjUd0123456

oluşan bu metinin hash değeri de Base64 kodlu olarak
Gin7L8Q/PUfTesQKG2WILXjlghE=
olacaktır.

.NET için örnek kaynak kod

```
SHA1 sha = new SHA1CryptoServiceProvider();
byte[] inputBytes = sha.ComputeHash(hashstr);
ToBase64Transform base64Transform = new ToBase64Transform();
base64Transform.TransformBlock(
                    inputBytes,
                    0,
                    inputBytes.Length,
                    outputBytes,
                    0);
```

Java için örnek kaynak kod

```
java.security.MessageDigest sha1 =
      java.security.MessageDigest.getInstance("SHA-1");

(new sun.misc.BASE64Encoder.BASE64Encoder())
      .encode(sha1.digest(hashstr.getBytes()))ş
```

PHP için örnek kaynak kod

```
PHP ((PHP 4 >= 4.3.0, PHP 5) )
base64_encode (pack('H*',sha1($hashstr)))
```

SHA1 hash hesaplama genel kaynak kod

```
Initialize variables:
h0 := 0x67452301
h1 := 0xEFCDAB89
h2 := 0x98BADCFE
h3 := 0x10325476
h4 := 0xC3D2E1F0
Pre-processing:
append a single "1" bit to message
append "0" bits until message length in bits ≡ 448 ≡ -64 (mod 512)
append length of message (before pre-processing), in bits as 64-bit big-endian
integer to message
Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w(i), 0 ≤ i ≤ 15
    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w(i) := (w(i-3) xor w(i-8) xor w(i-14) xor w(i-16)) leftrotate 1
    Initialize hash value for this chunk:
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
    Main loop:
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f := (b and c) or ((not b) and d)
            k := 0x5A827999
        else if 20 ≤ i ≤ 39
            f := b xor c xor d
            k := 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f := (b and c) or (b and d) or (c and d)
            k := 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f := b xor c xor d
            k := 0xCA62C1D6

        temp := (a leftrotate 5) + f + e + k + w(i)
        e := d
        d := c
        c := b leftrotate 30
        b := a
        a := temp

    Add this chunk's hash to result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e

digest = hash = h0 append h1 append h2 append h3 append h4 (expressed as big-
endian)
```