T.C. SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM465 - KRİPTOLOJİYE GİRİŞ
PROJE

MD5 Şifreleme

Hazırlayan

Ferdi Sönmez

B211210375

İçindekiler

MD5 Şifreleme Nedir?

Message Digest 5 (MD5) algoritması, verilen dosyanın veya mesajın (şifre vb.) kendine has “parmak izi” nin oluşturulmasını "hash" fonksiyonlarına dayalı olarak sağlayan bir algoritmadır. Bir veritabanı yönetimi (database management) tekniğidir. 1991 yılında MIT (Massachusetts Institute of Technology)’de görev yapan Profesör Ron Rivest tarafından geliştirilmiştir. Profesör Rivest MD5’i MD4’ün bir üst sürümü olarak tasarlamıştır.

Özellikleri:

1) MD5 algoritması tek yönlü çalışır. Şifreleme yapılır, ancak şifre çözüm işlemi yapılamaz.
2) MD5 algoritması, üzerinde işlem yapılan dosyada (aktarma vb.) herhangi bir değişiklik olup olmadığını tespit eder. Eğer bir değişiklik yapılmışsa, yeni dosyanın MD5 algoritmasından geçilmesinden çıkan sonuç ile ilk dosyanın MD5 sonucu birbirinden farklı olacaktır.
3) MD5 algoritması bir alt sürümü olan MD4’e göre yavaş çalışır, ancak şifrelendirme sistemi çok daha karışık ve çözülmesi güçtür.
4) Genel olarak 4 farklı aşamalı bir sisteme sahiptir. Her aşama birbirinden farklı işleyişe sahip olup 16’şar basamaktan oluşmuştur. Bir MD5 şifreleme işleminde aşağıdaki resimdeki sistemden 64 tane gerçekleşmektedir.
5) Boyutu fark etmeksizin algoritmaya girişi yapılan dosyanın çıkışı olarak 128-bit uzunluğunda 32 karakterli 16'lık sayı sisteminde bir dizi elde edilir.

Kullanıldığı Yerler:

1)İnternet trafiğinde. "SSL (Secure Sockets Layer - Güvenli Yuva Katmanı)" gibi.
2)Özel bilgisayar ağlarında. "VPN (Virtual Private Network - Sanal Özel Ağ)" gibi.
3)Güvenli uzaktan ulaşım uygulamalarında. "SSH (Secure Shell - Güvenli Kabuk)" gibi.
4)Kimlik belirleme uygulamalarında.

Dezavantajları:

Kullanıcı adı ve şifre ile giriş yapılan sitelerde, kullanıcı şifresini unuttuğu takdirde sistem eski şifreyi veremez. Şifre, MD5 algoritmasından geçirilmiş halde saklandığı için şifre çözülemez. Sistem kullanıcıya yeni şifre atar.
MD4 'e göre daha uzun bir şifre ürettiğinden çalışması daha uzun zaman alır.

Çakışmalar (Collisions):

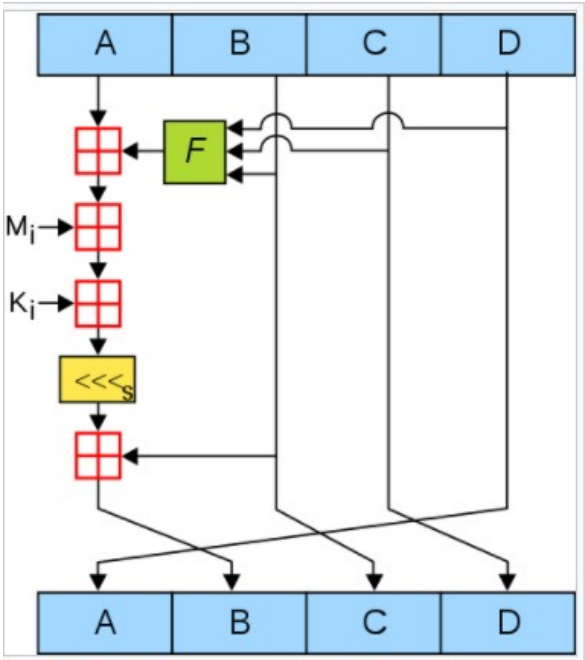
Çakışma olduğunda yani 2 veri aynı şifreye sahip olduğunda, verinin "hash" tablosundaki yeni yerinin hesaplanabilmesi için doğrusal sınama (linear probing), ikinci dereceden sınama (quadratic probing) yada ikili sınama (double probing) yöntemlerinden biri uygulanır. Doğrusal sınamada; veri "hash" tablosunda hemen bir sonraki yere yerleştirilir. İkili sınamada; şifrenin bulunduğu yerin nümerik karesi alınarak yeni yer belirlenir. İkinci dereceden sınamada ise; 2 "hash" fonksiyonu iç içe kullanılır. Eğer belli bölgelerde birikme olmuşsa buna kümelenme (cluster) denir. Zaten sınama yöntemlerindeki amaç da kümelenmeyi önlemektir. Ayrıca homojen dağılım olması için "hash" tablolarının büyüklüğü asal sayı tercih edilmelidir.

MD5 Şifreleme Algoritması

MD5 değişken uzunluktaki bir mesajı 128 bitlik bir sabit uzunlukta çıktı olarak işler.Giriş mesajı 512-bitlik blok parçalarına ayrılır (on altı tane 32-bitlik kelimeler halinde). İleti, uzunluğu 512 ile bölünebilecek şekilde doldurulur. Bu doldurma işlemi şu şekilde işler: İlk olarak mesajın sonuna bir bit 1 eklenir. Sonrasında mesajın uzunluğu 512'nin katından 64 bit eksik olacak şekilde 0'larla doldurulur. Geriye kalan 64 bite de orijinal mesajın uzunluğu mod 2^{64} 'de yazılır.
Ana MD5 algoritması, A, B, C ve D olarak adlandırılan dört adet 32 bitlik kelimeye ayrılmış 128 bitlik parçalar üzerinde çalışır. Bunlar belirli sabit değerlerle başlatılır.Daha sonra ana algoritma, her 512-bit ileti bloğunu durumunu(128 bit) değiştirmek için kullanır.Bir mesaj bloğunun işlenmesi, tur denilen dört benzer aşamadan oluşur; Her tur, doğrusal olmayan bir fonksiyon , modüler toplama işlemi ve bit bazında sola kaydırma işlemlerinden oluşur. Toplamda 16 tur vardır. Figür 1'de her tur içinde yapılan işlemler gösterilmiştir.4 olası F fonksiyonu vardır; her turda farklı bir fonksiyon kullanılır.

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \neg D)$$

$\oplus, \wedge, \vee, \neg$ sırasıyla XOR, AND, OR veNOToperasyonlarının yerine kullanılmıştır.



Figür 1.Bir MD5 fonksiyonudur.MD5 4 round içinde 16 kere olmak üzere bu işlemden 64 tane içerir.F lineer olmayan bir fonksiyondur. M_j 32 bitlik mesaj inputu, K_i her işlem için farklı bir constanttır. . $\lll_s$ s kadarlık sola kaydırma demektir. $\boxplus$ mod 2^{32}'de toplama için kullanılmıştır

Sözde Kod

```
//Not: Bütün değerler işaretsiz 32 bittir hesaplama yaparken 2^32
üzerinden mod alın
var int[64] s, K

//s her tur için kaydırma miktarlarını belirtir
s[ 0..15] := { 7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7,
12, 17, 22 }
s[16..31] := { 5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,
9, 14, 20 }
s[32..47] := { 4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4,
11, 16, 23 }
s[48..63] := { 6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6,
10, 15, 21 }

for i from 0 to 63
    K[i] := floor(2^32 × abs(sin(i + 1)))
end for
K[ 0.. 3] := { 0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee }
K[ 4.. 7] := { 0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501 }
K[ 8..11] := { 0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be }
K[12..15] := { 0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821 }
K[16..19] := { 0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa }
K[20..23] := { 0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8 }
K[24..27] := { 0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed }
K[28..31] := { 0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a }
K[32..35] := { 0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c }
K[36..39] := { 0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70 }
K[40..43] := { 0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x04881d05 }
K[44..47] := { 0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665 }
K[48..51] := { 0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039 }
K[52..55] := { 0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1 }
K[56..59] := { 0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1 }
K[60..63] := { 0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391 }

//değişkenlere değer ata:
var int a0 := 0x67452301   //A
var int b0 := 0xefcdab89   //B
var int c0 := 0x98badcfe   //C
var int d0 := 0x10325476   //D

//Pre-processing: Tek bir 1 bit ekleme
append "1" bit to message
// Notice: the input bytes are considered as bits strings,
//  where the first bit is the most significant bit of the
byte.[46]
```

```
append "0" bit until message length in bits ≡ 448 (mod 512)
append original length in bits mod (2 pow 64) to message

//İletiyi ardışık 512-bit parçalar halinde işleyin
for each 512-bit chunk of message
    break chunk into sixteen 32-bit words M[j], 0 ≤ j ≤ 15
//Bu parça için hash değşkenlerine değer ata:
    var int A := a0
    var int B := b0
    var int C := c0
    var int D := d0
//Ana Döngü:
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            F := (B and C) or ((not B) and D)
            g := i
        else if 16 ≤ i ≤ 31
            F := (D and B) or ((not D) and C)
            g := (5×i + 1) mod 16
        else if 32 ≤ i ≤ 47
            F := B xor C xor D
            g := (3×i + 5) mod 16
        else if 48 ≤ i ≤ 63
            F := C xor (B or (not D))
            g := (7×i) mod 16
//a,b,c,d'nin tanımlarına dikkat edin
        dTemp := D
        D := C
        C := B
        B := B + leftrotate((A + F + K[i] + M[g]), s[i])
        A := dTemp
    end for
//Bu parçanın şimdiye kadarki hash değerini ekle:
    a0 := a0 + A
    b0 := b0 + B
    c0 := c0 + C
    d0 := d0 + D
end for

var char digest[16] := a0 append b0 append c0 append d0 //(Output
is in little-endian)

//sola kaydırma fonksiyonu tanımı
leftrotate (x, c)
    return (x << c) binary or (x >> (32-c));
```

Program Çıktıları

MD5 algoritması uygulanması için verilen metin:

```
int main( int argc, char *argv[] )
{
    int j,k;
    const char *msg = "Ferdi Sonmez";   //Sifrelenmesi icin verilen metin
    unsigned *d = md5(msg, strlen(msg));
    WBunion u;

    printf("=0x");
    for (j=0;j<4; j++){
        u.w = d[j];
        for (k=0;k<4;k++) printf("%02x",u.b[k]);
    }
    printf("\n");

    return 0;
}
```

Programın Çalıştırılması:

```
f@ubuntu:~/Desktop/Deneme$ make
gcc -O2 -Wall -pedantic -c Kripto.c
gcc -O2 -Wall -pedantic -lm -lnsl  -o Kripto Kripto.o -lm
f@ubuntu:~/Desktop/Deneme$ ./Kripto
=0xc69088c01ec9fbc02c50014316ad47bb
```

Programın Ürettiği Çıktı:

```
f@ubuntu:~/Desktop/Deneme$ ./Kripto
=0xc69088c01ec9fbc02c50014316ad47bb
```

İnternet Sitesi Üzerinden Kontol: (https://www.hesapmatik.com/hesapla/md5-sifreleme/)

Decryption

Hash fonksiyonları verilen text üzerinde sıkılaştırma işlemi yaptığından dolayı şifrelenen metinler tam olarak verilen önceki text haline dönemezler.MD5 işleminde yapılan verilen metin 128 bit aralığına sıkıştırılmaktadır. Bu sebepten dolayı verilen metin şifrelendikten sonra tam olarak geri döndürülemez.Bu sebepten dolayı bu hash işlemleri kriptolonması gereken birçok yerde kullanılmaktadır.Örneğin veri tabanı sistemlerinde şifresi veya kullanıcı adı istenilen kullanıcının şifresi ve kullanıcı adı MD5 veya diğer hashleme algoritmalarından geçirilerek tutulmaktadır. Bu hashleme işlemlerinin geri döndürülmemesinden dolayı veritabanı güvenliği sağlanmaktadır.

Kaynakça

1) https://tr.wikipedia.org/wiki/MD5
2) "RFC 6151 – Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms". Internet Engineering Task Force. Mart 2011.
3) "Marc Stevens – Research – Single-block collision attack on MD5". Marc-stevens.nl. 2012
4) https://www.sibervatan.org/makale/md5-sifreleme/39
5) https://www.cozumpark.com/community/security-4/1025/
6) https://hackpress.org/makale/guvenlik/md5-nedir
7) https://sibersaldirilar.com/genel-siber-guvenlik/kriptoloji/md5-nedir-nasil-kirilir/
8) https://www.hesapmatik.com/hesapla/md5-sifreleme/