BÖLÜM 9

HASH FONKSİYONLARI

Hash fonksiyonları $h : \{1, 2, \ldots, 2^m\} \to \{1, 2, \ldots, 2^n\}$ ve asağıdaki özelliklere sahip olan fonksiyonlardır:

1. **sıkıştırma:** h fonksiyonu, uzunluğu sonlu ve değişken olabilen girdiyi alıp sabit bir uzunlukta çıktı vermelidir,
2. **kolay hesaplanabilirlik:** herhangi bir girdi için $h(x)$ değerini hesaplamak kolay olmalıdır.

Hash fonksiyonları anahtarsız hash fonksiyonları ve anahtarlı hash fonksi-yonları olmak üzere ikiye ayrılır:

1. **Anahtarsız hash fonksiyonları** $h : \{0, 1\}^* \to \{0, 1\}^n$
   - Blok şifreleme sistemleri tabanlı
   - Modüler aritmetik tabanlı
   - Customized (MD4,MD5,SHA-1,RIPE-MD,HAVAL)
2. **Anahtarlı hash fonksiyonları** $h_k : \{0, 1\}^* \to \{0, 1\}^n$
   - Blok şifreleme sistemleri tabanlı

- Anahtarsız hash fonksiyonları tabanlı
- Customized (MAA,MD5-MAC)
- Akan şifreler için üretilen

Customized hash fonksiyonları sadece hash için kullanılan anahtarlı veya anahtarsız olarak üretilen hash fonksiyonlarıdır. Ayrıca güvenilir-liği teorik olarak ispatlanan fakat pek pratik olmayan evrensel hash fonksiyonlarıda farklı bir grup olarak görülebilir.

Anahtarsız hash fonksiyonlarının üç temel özelliği aşağıda belirtilmiştir(h bir hash fonksiyonu, x ve $x^{'}$ girdileri, y ve $y^{'}$ çıktıları göstermektedir):

1. **preimage resistance:** $h(x) = y$ değeri bilindiğinde, x'i hesaplamak sonlu zamanda mümkün degil. y biliniyor, $h(x^{'}) = y$ olacak bir $x^{'}$ bulmak zor(hesaplamak sonlu zamanda mümkün degil).
2. **2nd-preimage resistance:** $h(x) = y$ biliniyor, $h(x^{'}) = y$ olacak farklı bir mesaj $x \neq x^{'}$ bulmak zor.
3. **collision resistance:** $h(x) = h(x^{'})$ olacak şekilde iki farklı mesaj x ve $x^{'}$ bulmak zor.

Örnek 1 *Mod-32 checksum (Mod 32 kontrol toplamları). Mesajın içerisindeki bütün 32-bit'lik parçaların toplamı alınarak kullanılan fonksiyon. Hesaplaması kolay, sıkıştırma var, fakat preimage resistant değil.*

Örnek 2 *$g(x) = x^2 \ mod \ n = pq$ p, q büyük asal sayılar (n'nin çarpanları bilinmiyorsa tek yönlü fonksiyondur.) Hesaplaması kolay, sıkıştırma yok, preimage resistant (çünkü preimage bulmak n'yi çarpanlarına ayırmaya denk), fakat 2nd preimage ve collision var $(x, -x)$.*

Örnek 3 *DES tabanlı tek yönlü fonksiyon. $f(x) = E_k(x) \oplus x$,sabit bir anahtar(k) için. E rasgele bir permütasyon olarak kabul edilirse f fonksi-yonu tek yönlü olur. y bilindiğinde $y = E_k(x) \oplus x$ olacak şekilde x ve k bulmak zor (E'nin rasgele olamasından dolayı), $E_k^{-1}(x \oplus y) = x$ bulmak zor, Dolayısıyla f tek yönlü bir fonksiyon. Fakat fonksiyon belli mesaj uzunlukları için çalışıyor.*

- collision resistant ise 2nd preimage resistantdır:

  Fonksiyonumuzun collision resistant olduğunu kabul edelim. 2nd preimage resistant degilse $\Rightarrow$ Sabit $x, h(x)$ için $h(x) = h(x')$ olan $x \neq x'$ bulabiliriz, fakat bu collision resistant olamadığını gösteririr, kabulümüzle çelişir.

- collision resistant ise preimage resistant olmak zorunda degildir:

  $g : (0,1)^* \rightarrow (0,1)^n$ collision resistant olsun, h fonksiyonunu aşağıdaki şekilde tanımlanırsa preimage resistant olmaz;

  $$h(x) = \begin{Bmatrix} 1 \| x & if \ |x|=n \\ 0 \| g(x) & if \ |x| \neq n \end{Bmatrix},$$

  $h : (0,1)^* \rightarrow (0,1)^{n+1}$ $n+1$ bit hash fonksiyon.

- preimage resistant ise 2nd preimage resistant olmak zorunda degildir:

  Örnek 2'de görülebilir.

Ekstra Şartlar:

1. **Non-correlation:** Girdi ve çıktı bitleri arasında korelasyon olmamalı, blok şifre sistemlerindeki gibi avalanche özelliği sağlanmalı(bütün girdi bitleri bütün çıktı bitlerini etkilemeli),

2. **Near-collision resistance:** $w(h(x) \oplus h(x^{'})$ küçük olacak farklı x ve $x^{'}$ çiftlerini bulmak zor olmalı (w:hamming agırlığı),

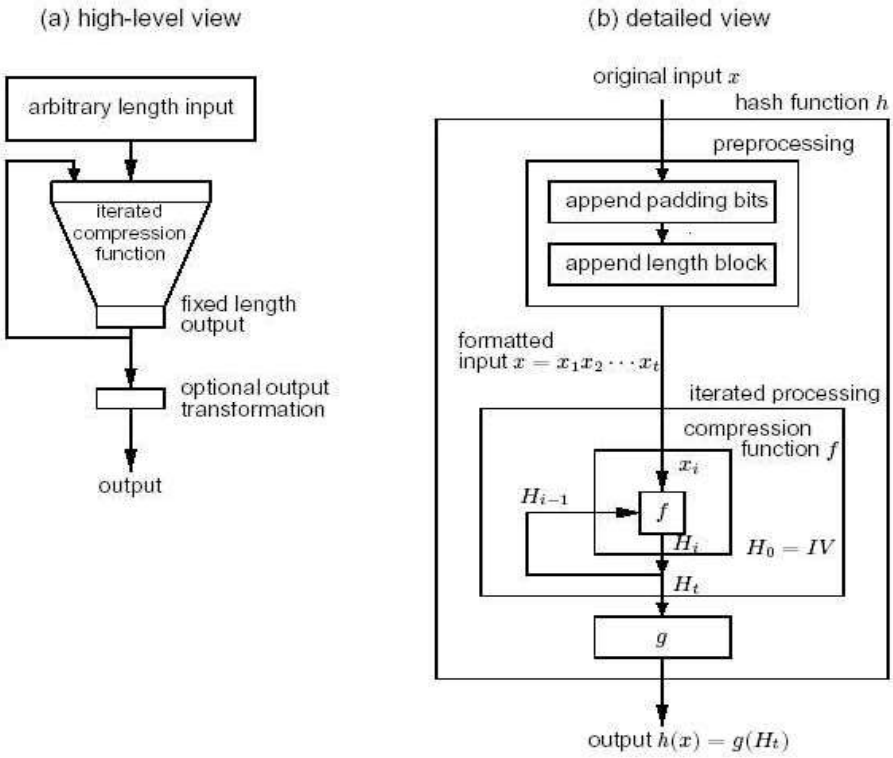
3. **Partial-preimage resistance(local one-wayness):** Girdi bitlerinin bir kısmını dahi bulmak zor olmalı, girdinin t uzunluğundaki kısmını bulmak için yaklaşık $2^t - 1$'lik hesaplama yapmak gerekmeli (girdinin belli bir kısmı bilinse dahi diger kısmını bulmak zor olmalı).

Anahtarsız hash fonksiyonlarının çogu girdi ve çıktı uzunluğu sabit olan bir f hash fonksiyonunun tekrarlı olarak uygulanmasıyla elde edilir. Bu fonksiyonlara **Iterative hash fonksiyonları**(h) adı verilir. Herhangi bir uzunluktaki x girdisi, sabit r-bit uzunluklara bölünür(x_i), x'in uzunluğunun r'nin katı olması için belli bir kurala bağlı olarak x'e padding (bit ekleme) yapılır. Girdi parçaları x_i'ler sırasıyla f'ye sokulur, f'nin çıktısı ve x_{i+1} tekrar f'nin girdisi olarak kullanılır ve son girdi bloğuna kadar bu işlem tekrarlanır. Bu durumda aşağıdaki işlemler yapılmış olur: $x = x_1 x_2 \dots x_t$,

$H_0 = IV$; $H_i = f(H_{i-1}, x_i)$, $1 \leq i \leq t$; $h(x) = g(H_t)$,

IV :başlangıç değeri

Iterative hash fonksiyonlarının genel ve detaylı yapıları aşağıdaki şekillerde verilmiştir:



NOT: f fonksiyonunun collision resistant olması h fonksiyonunun collision resistant olmasını garantiler.

Anahtarsız hash fonksiyonları:

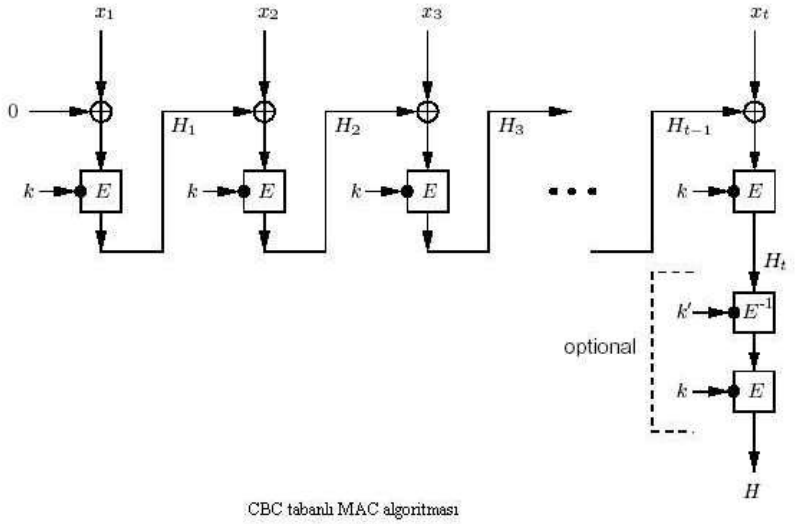
- **Blok şifre sistemleri tabanlı:** İterasyonda kullanılan f fonksiyonu herhangi bir blok şifre sistemi olarak seçilir. Kullanılan makinanın içinde bir blok şifreleme sistemi varsa hash fonksiyonu olarakta kullanılabilir.

- **Moduler Aritmetik tabanlı:** İterasyon fonksiyonu (f) mod M aritmetiğini baz alan bir fonksiyon olarak seçilir, çarpanlara ayırma ve discrete logaritim problemlerini temel alan sistemler seçilebilir.

- **Customized:** Özel olarak hash için tasarlanmış ve optimize hıza sahip olan fonksiyonlardır. Pratik olarak kullanılmaktadır, MD ailesi ve SHA örnek olarak verilebilir. Güvenilirlikleri hesaplama gücüne dayalı olarak ispatlanır, matemetiksel olarak güvenilir oldukları ispatlanmamıştır.

Anahtarlı hash fonksiyonları:

- **Blok şifreleme sistemleri tabanlı:** CBC tabanlı MAC'lar örnek olarak verilebilir.



CBC tabanlı MAC algoritması

- **Anahtarsız hash fonksiyonları tabanlı:** Gizli bir anahtarın anahtarsız hash

fonksiyonlarının girdisinin bir parçası olarak kullanılmasıyla üretilen hash fonksiyonlarıdır.

Aşağıdaki yöntemler örnek olarak verilebilir(k anahtar ve h bir anahtarsız hash fonksiyonu olmak üzere):

1. secret prefix metod: $M(x) = h(k||x)$,
2. secret suffix metod: $M(x) = h(x||k)$,
3. envelope metod with padding: $h_k(x) = h(k||p||x||k)$ p :padding $k||p$ bir blok uzunluğunda olacak şekilde padding yapılıyor,
4. hash tabanlı: $HMAC(x) = h(k||p_1||h(k||p_2||x))$, p_1, p_2 :padding, $k||p_1$ ve $h(k||p_2||x)$ birer tam blok uzunluğunda olacak şekilde padding yapılıyor.

- **Customized MAC'lar:** Sadece hash yapmak için tasarlanmış ve içerisinde gizli anahtar barındıran hash fonksiyonlarıdır. Örnek olarak MAA ve MD5-MAC verilebilir.

Hash fonksiyonları ile ilgili detaylı bilgiler aşağıdaki kaynaklarda bulunabilir:

- Handbook of Applied Cryptography,Chapter 9, by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac
- Cryptographic Hash Functions: A Survey, by S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk
- Hash functions based on block ciphers: a synthetic approach, by B. Preneel,R. Govaerts, and J. Vandewalle