Bu, sıkıştırılmış mesaj bloklarına çok fazla fazlalık ve karşılıklı bağımlılık getirir, bu da aynı sıkıştırma işlevi çıktısına eşlenen farklı bir mesaj bloğu bulma görevini zorlaştırır. Şekil 11.13, SHA-512 mantığını özetlemektedir.

SHA-512 algoritması, karma kodun her bitinin girdinin her bitinin bir işlevi olduğu özelliğine sahiptir. F temel fonksiyonunun karmaşık tekrarı, iyi karışmış sonuçlar üretir; yani, benzer düzenlilikler sergileseler bile rastgele seçilen iki mesajın aynı hash koduna sahip olması olası değildir. Şimdiye kadar yayınlanmamış olan SHA-512'de bazı gizli zayıflıklar olmadıkça, aynı mesaj özetine sahip iki mesaj bulmanın zorluğu 2256 işlem mertebesindeyken, belirli bir mesaja sahip bir mesajı bulmanın zorluğu özet 2512 işlem sırasıdır .

Örnek

Buraya FIPS 180'deki bir örneği temel alan bir örnek ekliyoruz. Üç ASCII karakterinden oluşan tek blokluk bir mesajı hashlemek istiyoruz: aşağıdaki 24 bitlik ikili diziye eşdeğer olan "abc":

01100001 01100010 01100011

SHA algoritmasının 1. adımından, mesajın 896 modulo 1024'e uygun bir uzunluğa kadar doldurulduğunu hatırlayın. Bu tek blok durumunda, doldurma 896 - 24 = 872 bitten oluşur ve ardından "1" bit gelir 871 "0" bit ile.
Ardından, orijinal mesajın bit cinsinden uzunluğunu (dolgudan önce) içeren, mesaja 128 bitlik bir uzunluk değeri eklenir. Orijinal uzunluk 24 bit veya onaltılık değer 18'dir. Bunların hepsini bir araya getirdiğimizde, 1024 bitlik mesaj bloğu onaltılık olarak şu şekildedir:

6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018

Bu blok W0, kelimelerine atanır. . . , aşağıdaki gibi görünen mesaj programının W15'i.

W0 = 6162638000000000	W8 = 0000000000000000
W1 = 0000000000000000	W9 = 0000000000000000
W2 = 0000000000000000	W10 = 0000000000000000
W3 = 0000000000000000	W11 = 0000000000000000
W4 = 0000000000000000	W12 = 0000000000000000
W5 = 0000000000000000	W13 = 0000000000000000
W6 = 0000000000000000	W14 = 0000000000000000
W7 = 0000000000000000	W15 = 0000000000000018

Dolgulu mesaj M1, M2, c , MN bloklarından oluşur. Her mesaj bloğu Mi , 16 adet 64-bit kelime Mi,0, Mi,1, c , Mi,15'ten oluşur . Tüm ekleme modulo 264 gerçekleştirilir .

H0,0 = 6A09E667F3BCC908 H0,4 = 510E527FADE682D1
H0,1 = BB67AE8584CAA73B H0,5 = 9B05688C2B3E6C1F
H0,2 = 3C6EF372FE94F82B H0,6 = 1F83D9ABFB41BD6B
H0,3 = A54FF53A5F1D36F1 H0,7 = 5BE0CD19137E2179

i = 1 ila N için

1. t = 0 ila 15 için W mesaj planını hazırlayın

t = 16 ila 79 için Wt = Mi,t

2. s1 512(Wt-2) + Wt-7 + s0 512(Wt-15) + Wt-16 Wt =

Çalışma değişkenlerini sıfırlayın a =

Yüksek-1, 0 e = Yüksek-1, 4 b =

Yüksek-1 , 1 f = Yüksek-1, 5 c =

Yüksek-1, 2 g = Yüksek-1, 6 d =

Yüksek-1, 3 h = Yüksek-1, 7 3. t =

0 ila 79 için ana hash hesaplamasını gerçekleştirin

T1 = h + Ch(e, f, g) + ¢Σ512 1 e + Wt + Kt

T2 = ¢Σ512 0 a + Maj(a, b, c)

h =

gg =

ff =

ee = d + T1

d = c

c = b

b = bir

a = T1 + T2 4.

Ara hash değerini hesaplayın Hi, 4 = e + Hi-1,4 Hi,

Merhaba, 0 = bir + Yüksek-1, 0 5 = f + Hi-1, 5 Hi, 6 =

Merhaba, 1 = b + Yüksek-1, 1 g + Hi-1, 6 Hi, 7 = h +

Merhaba, 2 = c + Yüksek-1, 2 Merhaba-1, 7

Merhaba, 3 = d +

Yüksek-1, 3 dönüş {HN, 0 } HN, 1 } HN, 2 } HN, 3 } HN, 4 } HN, 5 } HN, 6 } HN, 7}

Şekil 11.13 SHA-512 Mantığı

Şekil 11.13'te gösterildiği gibi, a'dan h'ye sekiz 64 bitlik değişken, H0,0 ila H0,7 değerlerine sıfırlanır . Aşağıdaki tablo, bu değişkenlerin başlangıç değerlerini ve ilk iki turun her birinden sonraki değerlerini göstermektedir.

A	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
C	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
D	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
G	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
H	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

Turların her birinde altı değişkenin doğrudan kopyalandığını unutmayın. önceki turdaki değişkenler.

İşlem 80 tur boyunca devam eder. Son turun çıktısı

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9
d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

Hash değeri daha sonra şu şekilde hesaplanır:

H1,0 = 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba
H1,1 = bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131
H1,2 = 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2
H1,3 = a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a
H1,4 = 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8
H1,5 = 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd
H1,6 = 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e
H1,7 = 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f

Ortaya çıkan 512 bit mesaj özeti şu şekildedir:

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

Şimdi giriş mesajını "abc"den "abc"ye bir bit değiştirdiğimizi varsayalım. “cbc.” Ardından, 1024 bitlik mesaj bloğu

6362638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018

Ve ortaya çıkan 512 bitlik mesaj özeti şu şekildedir:

531668966ee79b70 0b8e593261101354 4273f7ef7b31f279 2a7ef68d53f93264
319c165ad96d9187 55e6a204c2607e27 6e05cdf993a64c85 ef9e1e125c0f925f

İki hash değeri arasında farklılık gösteren bit konumlarının sayısı 253'tür, bu da bit konumlarının neredeyse tam yarısıdır, bu da SHA-512'nin iyi bir çığ etkisine sahip olduğunu gösterir.

11.6 SHA-3

Bu yazı itibariyle, Güvenli Karma Algoritma (SHA-1) henüz "kırılmadı".
Yani, hiç kimse pratik bir sürede çarpışmalar üretmek için bir teknik göstermedi.
Bununla birlikte, SHA-1 yapı ve kullanılan temel matematiksel işlemler açısından, her ikisi de kırılmış olan MD5 ve SHA-0'a çok benzer olduğundan, SHA-1 güvensiz kabul edilir ve SHA-2 için aşamalı olarak kaldırılmıştır.

SHA-2, özellikle 512-bit versiyonu, tartışılmaz bir güvenlik sağlıyor gibi görünüyor. Ancak SHA-2, öncekilerle aynı yapıyı ve matematiksel işlemleri paylaşır ve bu bir endişe nedenidir. SHA-2 için uygun bir yedek bulmak yıllar alacağından, savunmasız hale gelmesi durumunda NIST yeni bir hash standardı geliştirme sürecini başlatmaya karar verdi.

Buna göre NIST, 2007'de SHA-3 olarak adlandırılacak yeni nesil NIST karma işlevini üretmek için bir yarışma duyurdu. SHA-3 için kazanan tasarım, Ekim 2012'de NIST tarafından duyuruldu ve Ağustos 2015'te FIP 102 olarak yayınlandı.
SHA-3, çok çeşitli uygulamalar için onaylanmış standart olarak SHA-2'yi tamamlaması amaçlanan bir kriptografik özet işlevidir.

Ek V, NIST tarafından AES adayları arasından seçim yapmak için kullanılan değerlendirme kriterlerine ve ayrıca kazanan aday olan Keccak'ı seçme gerekçesine bakar. Bu materyal, yalnızca SHA-3 tasarımını anlamakta değil, aynı zamanda herhangi bir kriptografik hash algoritmasını yargılamak için kullanılan kriterleri anlamakta da yararlıdır.

Sünger Yapı

SHA-3'ün temel yapısı, tasarımcıları tarafından sünger yapı [BERT07, BERT11] olarak anılan bir şemadır. Sünger yapısı, diğer yinelemeli hash fonksiyonlarıyla aynı genel yapıya sahiptir (Şekil 11.8). Sünger işlevi bir giriş mesajını alır ve onu sabit boyutlu bloklara ayırır. Her blok sırayla işlenir ve her yinelemenin çıktısı bir sonraki yinelemeye beslenir ve sonunda bir çıktı bloğu üretilir.

Sünger işlevi üç parametre ile tanımlanır: f = her giriş bloğunu işlemek için kullanılan dahili işlev[3] r = bit hızı pedi olarak adlandırılan giriş bloklarının bit cinsinden boyutu = doldurma algoritması

Sünger işlevi, hem değişken uzunluklu girdiye hem de çıktıya izin vererek onu bir hash işlevi (sabit uzunluklu çıktı), sözde rastgele sayı üreteci (sabit uzunluklu girdi) ve diğer kriptografik işlevler için kullanılabilen esnek bir yapı haline getirir. Şekil 11.14 bu noktayı göstermektedir. n bitlik bir giriş mesajı, her biri r bitlik k sabit boyutlu bloğa bölünür . Mesaj, r bitin tamsayı katı olan bir uzunluğa ulaşmak için doldurulur . Ortaya çıkan bölüm, k * r uzunluğundaki Pk-1 blok dizisidir . Tekdüzelik için dolgu her zaman eklenir, yani P0, P1, c

[3]Keccak belgeleri, f'yi bir permütasyon olarak ifade eder. Göreceğimiz gibi, hem permütasyonları hem de ikameleri içerir. Yineleme işlevi olarak f'ye atıfta bulunuyoruz çünkü bu, her yineleme için bir kez, yani işlenen mesajın her bloğu için bir kez yürütülen işlevdir.