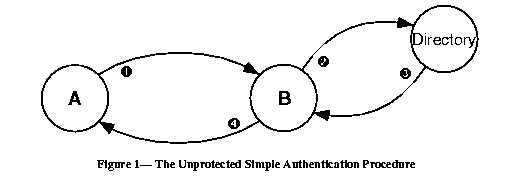
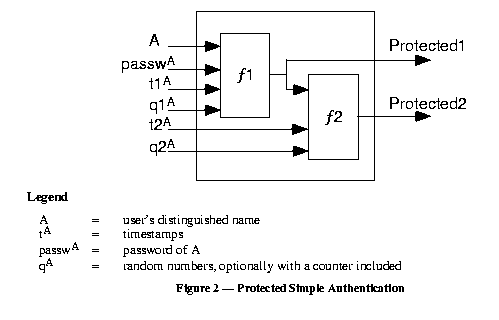
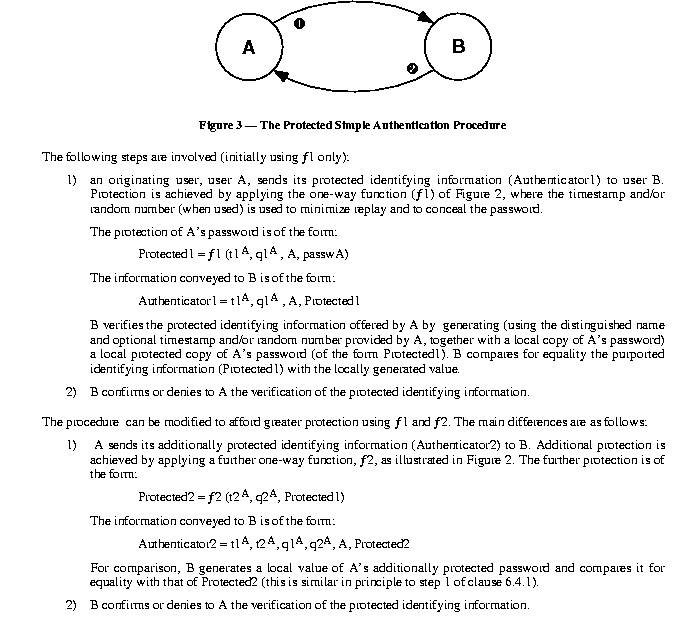
# Kimlik Doğrulama Prosedürleri

X.509 ayrıca çeşitli uygulamalarda kullanılmak üzere tasarlanmış üç alternatif kimlik doğrulama prosedürünü de içerir. Tüm bu prosedürler açık anahtar imzalarından yararlanır. İki tarafın birbirlerinin açık anahtarını, ya birbirlerinin sertifikalarını dizinden alarak ya da sertifika her iki taraftan gelen ilk mesajda yer aldığı için bildiği varsayılır. Bu prosedürler, bir istemcinin kimliğini ve parolasını bir zaman damgası ve rastgele sayıyla birlikte açık veya karma olarak göndererek bir sunucuya doğruladığı basit kimlik doğrulama prosedürlerinin aksine güçlü olarak kabul edilir ( [www.dante.net/np/ds/osi/9594-8-X.509.A4.ps](http://www.dante.net/np/ds/osi/9594-8-X.509.A4.ps) ):





# Kimlik Doğrulama Prosedürleri (Devamı 1)



# Tek Yönlü Kimlik Doğrulama

Tek yönlü kimlik doğrulama, bir kullanıcıdan (A) diğerine (B) tek bir bilgi aktarımını içerir ve aşağıdakileri oluşturur:

1. A'nın kimliği ve mesajın A tarafından üretildiği
2. Mesajın B'ye yönelik olduğu
3. Mesajın bütünlüğü ve özgünlüğü (birden fazla gönderilmemiş olması)

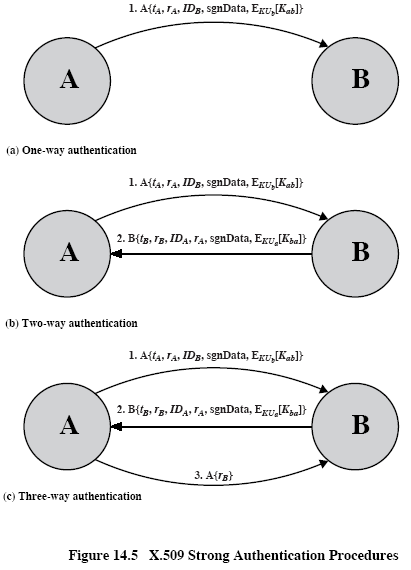
Bu süreçte yalnızca başlatan varlığın kimliğinin doğrulandığını, yanıt veren varlığın kimliğinin doğrulanmadığını unutmayın.

En azından, mesaj bir zaman damgası, , bir nonce ve B'nin kimliğini içerir ve A'nın özel anahtarıyla imzalanır. Zaman damgası, isteğe bağlı bir oluşturma zamanı ve bir sona erme zamanından oluşur. Bu, mesajların gecikmeli teslimini önler. Nonce, tekrarlama saldırılarını tespit etmek için kullanılabilir. Nonce değeri, mesajın sona erme zamanı içinde benzersiz olmalıdır. Bu nedenle, B, nonce'u saklayabilir

# Tek Yönlü Kimlik Doğrulama (Devamı 1)

süresi dolana kadar devam eder ve aynı nonce'a sahip yeni mesajları reddeder.

Saf kimlik doğrulaması için, mesaj yalnızca B'ye kimlik bilgilerini sunmak için kullanılır. Mesaj ayrıca iletilecek bilgileri de içerebilir. Bu bilgi, sgnData, imzanın kapsamına dahil edilir ve imzanın gerçekliğini ve bütünlüğünü garanti eder. Mesaj ayrıca B'ye, B'nin genel anahtarıyla şifrelenmiş bir oturum anahtarı iletmek için de kullanılabilir.



# Çift Yönlü Kimlik Doğrulama

Az önce sıralanan üç öğeye ek olarak, çift yönlü kimlik doğrulama aşağıdaki öğeleri oluşturur:

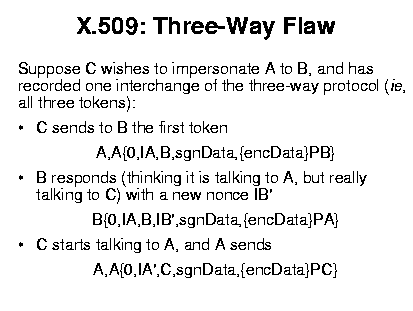
1. B'nin kimliği ve yanıt mesajının B tarafından oluşturulduğu
2. Mesajın A'ya yönelik olduğu
3. Cevabın bütünlüğü ve özgünlüğü

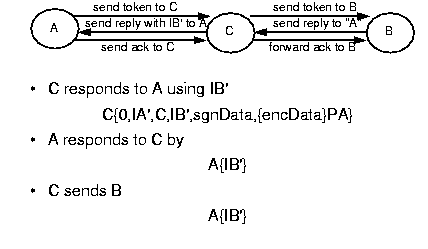
Cevap mesajı, cevabı doğrulamak için A'dan gelen bir nonce içerir. Ayrıca B tarafından üretilen bir zaman damgası ve bir nonce içerir. Daha önce olduğu gibi, mesaj imzalanmış ek bilgiler ve A'nın genel anahtarıyla şifrelenmiş bir oturum anahtarı içerebilir.

# Üç Yönlü Kimlik Doğrulama

Burada, A'dan B'ye imzalanmış bir kopya içeren son bir mesaj eklenmiştir . Bu tasarımın amacı, zaman damgalarının kontrol edilmesine gerek olmamasıdır: Her iki nonce da diğer taraf tarafından yankılandığından, her taraf tekrarlama saldırısını tespit etmek için döndürülen nonce'u kontrol edebilir. Bu yaklaşım, senkronize saatler mevcut olmadığında gereklidir. Şekil 14.5'te gösterilmemiştir, ancak A'dan gelen yanıt, rakip C, A ve B arasındaki mesajları engellediğinde ortada buluşma saldırısına karşı koymak için B'yi de içerir.

( [http:// nob.cs.ucdavis.edu/classes/ecs153-1997-01/Postscript/kerbiso.ps](http://nob.cs.ucdavis.edu/classes/ecs153-1997-01/Postscript/kerbiso.ps) ).





Yukarıdaki protokolde zaman damgaları sıfırdır çünkü üçlü kimlik doğrulamada saatler kullanılmaz.

# Tek Seferlik Şifre (OTP)

**L. Lamport, Güvenli olmayan iletişimde parola kimlik doğrulaması. – ACM İletişimleri, 1981, v. 24, No 11, 770-772,** [**http://cmpe.emu.edu.tr/chefranov/cmpe552-06/Lecture%20Notes/Lamport81.pdf'yi**](http://cmpe.emu.edu.tr/chefranov/cmpe552-06/Lecture%20Notes/Lamport81.pdf) takip ediyoruz.].

Parolalar bir ağdan veya bir sunucunun veritabanından geçtiklerinde tehlikeye girebilir. Önceki değerlendirmelerde, güvenilir bir üçüncü taraf olduğunu ve/veya sunucu tarafındaki parola veritabanının güvenli olduğunu varsaydık. Ancak bu doğru olmayabilir ve OTP şemaları bu gibi durumlar için çözüm sunar.

Çözüm büyük ölçüde tek yönlü karma fonksiyonlarına dayanmaktadır.

Lamport'un şeması, başlatma prosedürü gerekmeden önce bir kullanıcının sunucuya kimlik doğrulamasının sonlu sayıda olmasına izin verir. Başlatma prosedüründe, kullanıcı ve sunucu güvenli bir şekilde gizli bilgi alışverişinde bulunur (bazı özel kanallar aracılığıyla, şahsen, sipariş edilen posta, kurye veya başka bir güvenli yolla). Şema, bir parolanın asla güvenli olmayan bir ağdan geçmediğini ve sunucunun parola veritabanının tehlikeye girebileceğini, ancak bir davetsiz misafir tarafından değiştirilemeyeceğini varsayar. Sunucu ve kullanıcı, kimlik doğrulama prosedüründe aynı karma işlevini, , kullanır. Sunucu, veritabanındaki geçerli bir "parola" değerine karma işlevini uygulayarak kullanıcıyı doğrular. Aslında, bu değer pasaporttan türetilir ve bir kimlik doğrulamadan diğerine değişen geçerli bir parola olarak kullanılır. Bu nedenle şemaya "tek seferlik parola" denir. Geçerli parola, kimlik doğrulama numarasına bağlıdır ve değerini aşamaz . OTP şemaları "meydan okuma-yanıt" şemalarını temsil eder.

# Başlatma Prosedürü

İstemci bir şifre , bir sayı , hesaplar seçer

,

Neresi

.

İstemci güvenli bir şekilde sunucuya ( ) teslim eder ve sunucu bunu ( ) tuple'ına kaydeder.

# Kimlik Doğrulama Prosedürü

İstemci C, sunucu S'den kimlik doğrulaması istediğinde aşağıdakiler gerçekleşir:

1. C -> S: C\_ID //müşteri kimliğini gönderir

2. S -> C: Counter(C\_ID) //sunucu ilgili Counter değerine göre yanıt verir

3. C -> S: C\_Kimliği,

4. S: Eğer öyleyse {

S, C'yi doğrular ve ( )=( ) değerini ayarlar

}

Aksi halde C doğrulanmaz

sonra , bire eşit olacak ve bir sonraki kimlik doğrulamasında, C şifresini açık bir şekilde iletmelidir. Dolayısıyla, gizli istemcinin şifresini göndermeden yapılabilecek en fazla kimlik doğrulama sayısı 'dir . Şemadaki hile, karma işlevinin tek yönlü olması nedeniyle, karmasının S tarafından saklanan değere eşit olacağı bir değer bulmanın mümkün olmamasıdır. Bu değer bir kimlik doğrulamasından diğerine değiştiği sürece, mevcut şifrenin bilinmesi, kırılma oranı şifre güncelleme oranından daha az olabileceğinden, şifrenin kullanılmasına izin vermez. Şema, büyükse (sık başlatmayı önlemek için) C'nin ortalama olarak büyük miktarda hesaplama yapması gerektiği (kimlik doğrulama prosedürünün 3. Adımı) sorununa sahiptir.

# MD5 Mesaj Özeti Algoritması

MD5 ( <http://www.faqs.org/rfcs/rfc1321.html>) 1991 yılında MIT'de Ron Rivest tarafından geliştirildi. 1996 yılında bir kusur bulunana kadar MD5 en yaygın kullanılan güvenli karma algoritmasıydı. Açıklamada Stallings, Kriptografi ve Ağ Güvenliği ders kitabını takip ediyoruz.

# MD5 Mantığı

Algoritma girdi olarak keyfi uzunlukta bir mesaj alır ve çıktı olarak 128 bitlik bir mesaj özeti üretir. Giriş 512 bitlik bloklarda işlenir.

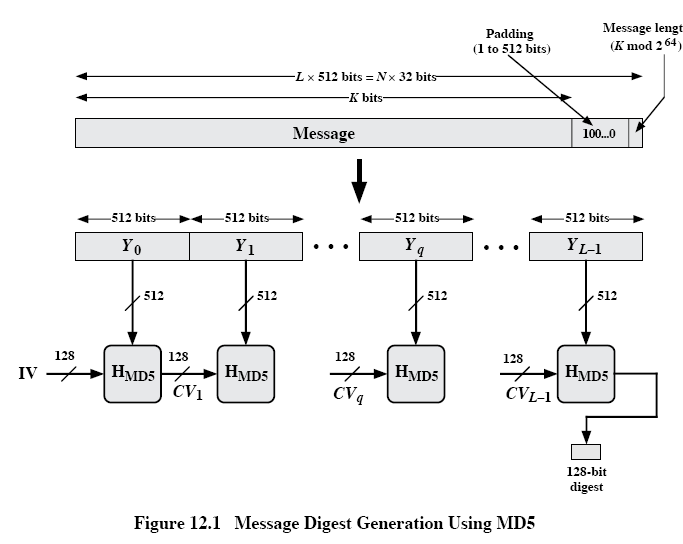
Şekil 12.1, bir özet üretmek için bir mesajın genel işlenmesini göstermektedir. İşleme aşağıdaki adımlardan oluşur:

1. **Dolgu bitlerini ekle** : Mesaj, bit cinsinden uzunluğu 512 ( ) modülünde 448'e denk olacak şekilde doldurulur . Yani, doldurulan mesajın uzunluğu 512 bitin bir tam sayı katından 64 bit daha azdır. Mesaj zaten istenen uzunlukta olsa bile, dolgu her zaman eklenir. Örneğin, mesaj 448 bit uzunluğundaysa, 960 bit uzunluğa ulaşana kadar 512 bit ile doldurulur. Böylece, dolgu bitlerinin sayısı 1 ila 512 aralığında olur. Dolgu, gerekli sayıda 0 biti izleyen tek bir 1 bitten oluşur.
2. **Ek uzunluğu** : Orijinal mesajın bit cinsinden uzunluğunun 64 bitlik bir gösterimi (dolgudan önce) Adım 1'in sonucuna eklenir (önce en az anlamlı bayt). Orijinal

length 'den büyükse , yalnızca length'in alt sıradaki 64 biti kullanılır. Bu nedenle, alan orijinal mesajın uzunluğunu, modulo olarak içerir .

İlk iki adımın sonucu, 512 bit uzunluğunda bir tam sayı katı olan bir mesaj üretir. Şekil 12.1'de,

# MD5 Mantığı (Devamı 1)

genişletilmiş mesaj, 512 bitlik blokların dizisi olarak temsil edilir , böylece genişletilmiş mesajın toplam uzunluğu bit olur. Eşdeğer olarak, sonuç 16 adet 32 bitlik kelimenin bir katıdır. M[0..N-1]'in, N'nin 16'nın tam sayı katı olduğu sonuçtaki mesajın kelimelerini göstermesine izin verin. Böylece, .

1. **MD tamponunu başlat** : Karma işlevinin ara ve son sonuçlarını tutmak için 128 bitlik bir tampon kullanılır. Tampon dört adet 32 bitlik kayıt (A, B, C, D) olarak temsil edilebilir. Bu kayıtlar aşağıdaki 32 bitlik tam sayılara (onaltılık değerler) başlatılır:

A=67452301

B=EFCDAB89

C=98BADCFE

D=10325476

Bu değerler, düşük adresli bayt konumundaki bir kelimenin en az anlamlı baytı olan küçük uçlu biçimde saklanır. 32 bit dizeler olarak, başlatma değerleri (onaltılık biçimde) aşağıdaki gibi görünür:

Kelime A: 01 23 45 67

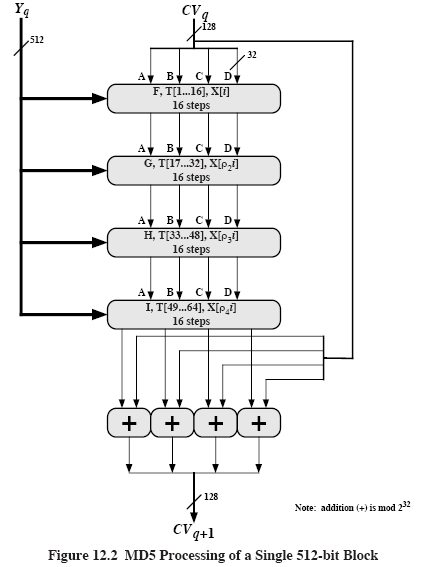
Kelime B: 89 AB CD EF

Kelime C: FE DC BA 98

Kelime D: 76 54 32 10

# MD5 Mantığı (Devamı 2)

1. **512 bitlik (16 sözcük) bloklarda işlem mesajı** : Algoritmanın kalbi, dört işlem "turundan" oluşan bir sıkıştırma işlevidir; bu modül Şekil 12.1'de etiketlenmiştir ve mantığı Şekil 12.2'de gösterilmiştir. Dört tur benzer bir yapıya sahiptir, ancak her biri spesifikasyonda F, G, H ve I olarak adlandırılan farklı bir ilkel mantıksal işlev kullanır.



Her tur, işlenen geçerli 512 bitlik bloğu ( ) ve 128 bitlik arabellek değeri ABCD'yi girdi olarak alır ve arabelleğin içeriklerini günceller. Her tur ayrıca, sinüs fonksiyonundan oluşturulmuş 64 elemanlı bir tablonun T[1..64] dörtte birini kullanır. T'nin *i- inci elemanı,*

T[i], 'nin tam sayı kısmına eşit değere sahiptir , burada *i* radyan cinsindendir. Çünkü , T'nin her bir elemanı 32 bitte temsil edilebilen bir tam sayıdır. Tablo, giriş verilerindeki herhangi bir düzenliliği ortadan kaldırması gereken 32 bitlik desenlerin "rastgele" bir kümesini sağlar. Tablo 12.1b, T değerlerini listeler.

# MD5 Mantığı (Devamı 3)

Dördüncü tur çıktısı, ilk turdaki girdiye ( ) eklenerek üretilir . Toplama, tampondaki her dört kelime için, karşılık gelen kelimelerin her biri için , modüler toplama kullanılarak bağımsız olarak yapılır .

1. Çıktı: Tüm L 512 bitlik bloklar işlendikten sonra L'inci aşamadan gelen çıktı 128 bitlik mesaj özetidir.

MD5’in davranışını şu şekilde özetleyebiliriz:



Nerede

- Adım 3'te tanımlanan ABCD tamponunun başlangıç değeri

- Mesajın q'uncu 512 bitlik *bloğu*

*L* – mesajdaki blok sayısı (dolgu ve uzunluk alanları dahil)

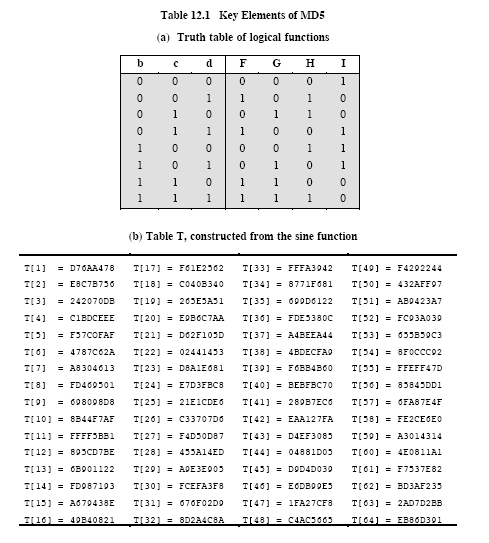
Mesajın *q'uncu* bloğuyla işlenen zincirleme değişkeni

- ilkel mantık fonksiyonu *x'i kullanarak yuvarlama fonksiyonu*

- son mesaj özeti değeri

giriş çiftinin her bir kelimesi için ayrı ayrı gerçekleştirilen modülo eklemesi

# MD5 Mantığı (Devam 4)



# MD5 Sıkıştırma Fonksiyonu

Bir 512-bit bloğun işlenmesinin dört turunun mantığına daha detaylı bakalım. Her tur, ABCD tamponunda çalışan 16 adımdan oluşan bir diziden oluşur. Her adım şu biçimdedir



Nerede

*a,b,c,d* – tamponun dört kelimesi, adımlar arasında değişen belirli bir sırada

*g* – F,G,H,I ilkel fonksiyonlarından biri

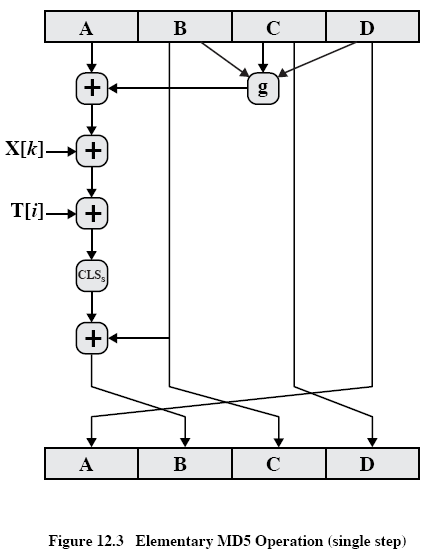
*<<<s* – 32 bitlik argümanın *s* bitlik dairesel sola kaydırılması (döndürülmesi)

*X[k] – M[q* *16+k] –* mesajın *q -inci 512 bitindeki k* -inci 32 bitlik kelime

*T[i]* – *T* matrisindeki *i -inci 32 bitlik kelime*

+ - modül ekleme

Şekil 12.3 adım işlemini göstermektedir



*a,b,c,d ) kullanılma* sırası, her adımda kelime düzeyinde dairesel bir sağa kaydırma üretir.

Algoritmanın dört turunun her biri için dört ilkel mantıksal fonksiyondan biri kullanılır. Her ilkel fonksiyon girdi olarak üç 32 bitlik kelime alır ve 32 bitlik bir çıktı üretir. Her

# MD5 Sıkıştırma Fonksiyonu (Devamı 1)

fonksiyon bir dizi mantıksal işlem gerçekleştirir; yani çıktının *n'inci* biti üç girdinin bir fonksiyonudur. Fonksiyonlar aşağıdaki gibi özetlenebilir:

|  |  |  |
| --- | --- | --- |
| Yuvarlak | İlkel fonksiyon g | g(b,c,d) |
| 1 | F(b,c,d) |  |
| 2 | G(b,c,d) |  |
| 3 | H(b,c,d) |  |
| 4 | Ben(b,c,d) |  |

*X[0..15]* sözcük dizisi, işlenen geçerli 512 bitlik giriş bloğunun değerini tutar. Bir tur içinde, *X[i] 'in 16 sözcüğünün her biri* , bir adım sırasında tam olarak bir kez kullanılır; bu sözcüklerin kullanıldığı sıra, turdan tura değişir. İlk turda, orijinal sırayla kullanılırlar. 2 ila 4 arasındaki turlar için aşağıdaki permütasyonlar tanımlanmıştır:

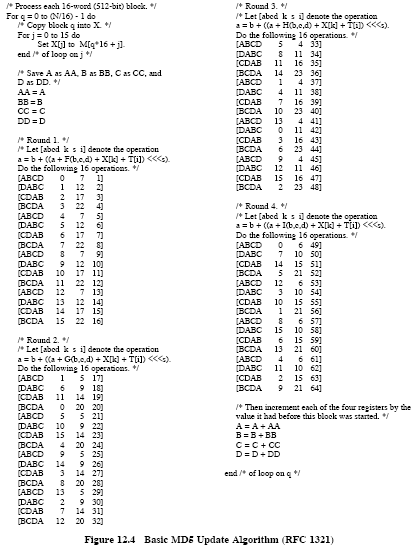


Permutasyonu düşünün :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 6 | 11 | 0 | 5 | 10 | 15 | 4 | 9 | 14 | 3 | 8 | 13 | 2 | 7 | 12 |

*T'nin* 64 32 bitlik sözcük öğelerinin her biri, bir turun bir adımında tam olarak bir kez kullanılır. Ayrıca, her adım için *ABCD* tamponunun 4 baytından yalnızca birinin güncellendiğine dikkat edin. Dolayısıyla, tamponun her baytı tur sırasında dört kez ve ardından bu bloğun nihai çıktısını üretmek için son bir kez güncellenir. Son olarak, her turda dört farklı dairesel sola kaydırma miktarının kullanıldığına ve turdan tura farklı olduğuna dikkat edin. Tüm bu karmaşıklığın amacı, çarpışmaların (aynı çıktıyı üreten iki 512 bitlik blok) üretilmesini çok zorlaştırmaktır.

# MD5 Sıkıştırma Fonksiyonu (Devamı 2)



2. Turdaki *k değerlerinin* sırasının yukarıda belirtilen permütasyonu izlediğini görüyoruz .