BLM5102 Bilgisayar Sistemleri ve Ağ Güvenliği

Prof. Dr. Hasan Hüseyin BALIK

(10. Hafta)

İçerik

- 3.Şifreleme Algoritmaları
  - 3.1 Şifreleme Araçları
  - 3.2 Simetrik Şifreleme ve Mesaj Gizliliği
  - 3.3 Açık Anahtarlı Şifreleme ve İleti Kimlik Doğrulaması

3.3 Açık Anahtarlı Şifreleme ve İleti Kimlik Doğrulaması

3.3.İçerik

- Güvenli Hash/Özet İşlevleri
- HMAC
- Kimliği Doğrulanmış Şifreleme
- RSA (Ron Rivest, Adi Shamir ve Leonard Adleman) Açık Anahtarlı Şifreleme Algoritması
- Diffie-Hellman ve Diğer Asimetrik Algoritmalar

	Bit 1	Bit 2	• • •	Bit n
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
Block m	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

Figure 21.1 Simple Hash Function Using Bitwise XOR

Güvenli Hash/Özet Algoritması (SHA)

- SHA ilk olarak NIST tarafından geliştirilmiştir.
- 1993'te FIPS 180 olarak yayınlanmıştır (SHA-0)
- 1995 yılında SHA-1 olarak revize edilmiştir.
  - 160 bitlik hash değerleri üretir
- NIST, 2002'de revize edilmiş FIPS 180-2'yi yayınladı
  - SHA'nın 3 ek sürümünü ekler
  - SHA-256, SHA-384, SHA-512
  - 256/384/512-bit hash değerleri ile
  - SHA-1 ile aynı temel yapı ancak daha fazla güvenlik
- En yeni sürüm, 224 bit ve 256 bit özet boyutlarına sahip iki SHA-512 varyantı ekleyen FIPS 180-4'tür.

SHA Parametrelerinin Karşılaştırılması

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512	SHA-512/224	SHA-512/256
Mesaj boyutu	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$
Kelime boyutu	32	32	32	64	64	64	64
Blok boyutu	512	512	512	1024	1024	1024	1024
Mesaj özeti boyutu	160	224	256	384	512	224	256
Adım sayısı	80	64	64	80	80	80	80
Güvenlik	80	112	128	192	256	112	128

Notlar: 1. Tüm boyutlar bit cinsinden ölçülür.
2. Güvenlik, n boyutundaki bir mesaj özetine yapılan doğum günü saldırısının, yaklaşık $2^{n/2}$'lik bir iş faktörü ile bir çarpışma üretmesi gerçeğini ifade eder.

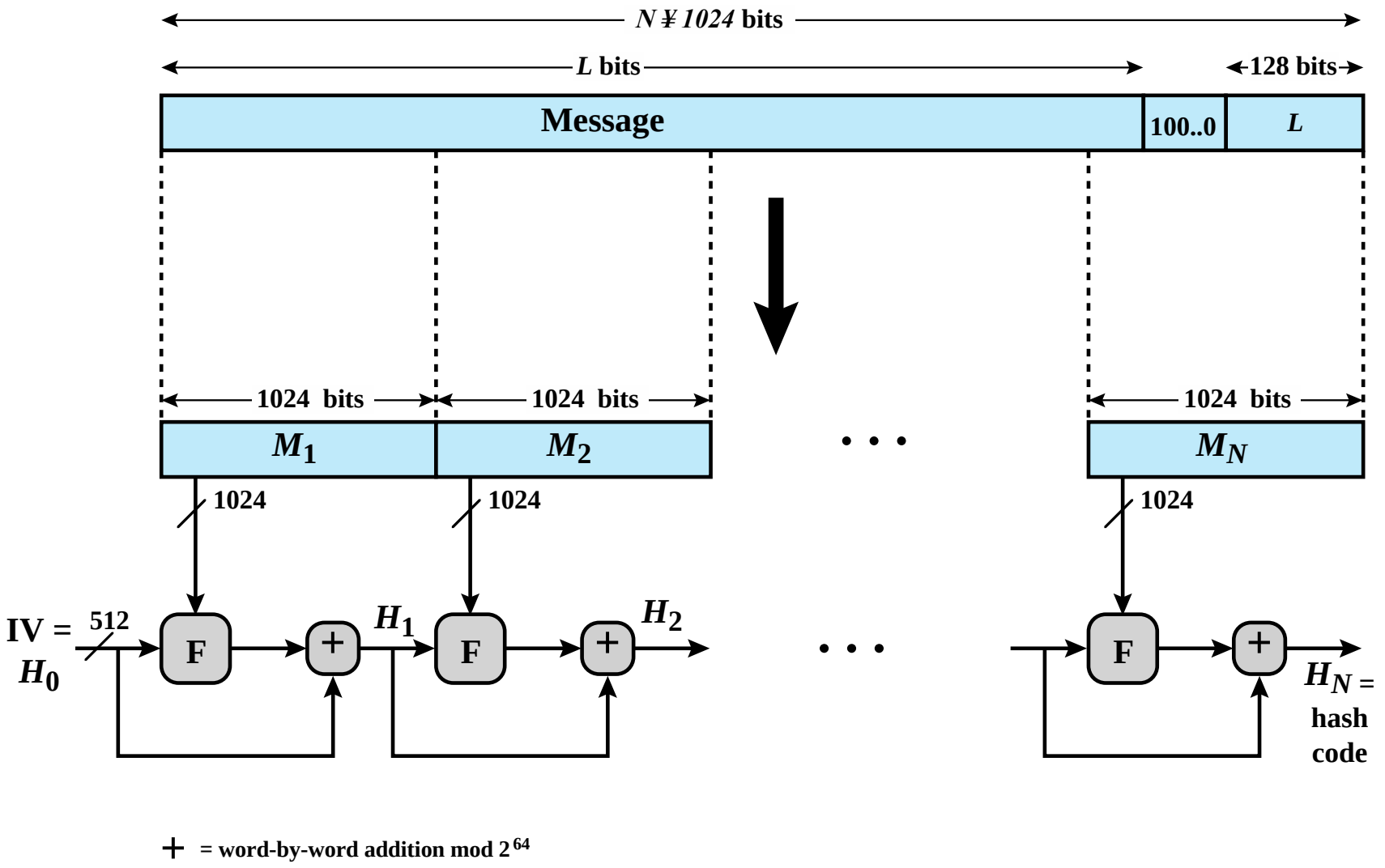


Figure 21.2 Message Digest Generation Using SHA-512

- İşlem aşağıdaki adımlardan oluşur:
  - **Adım 1:** Dolgu bitlerini ekleyin: böylece mesaj uzunluğu 896 modulo 1024 [length ◎ 896 (mod 1024)] ile uyumlu olur. Dolgu, tek bir 1 bit ve ardından gerekli sayıda 0 bitten oluşur.
  - **Adım 2**: Ek uzunluk: orijinal mesajın işaretsiz 128 bit tamsayı uzunluğu olan 128 bitlik bir blok olarak (doldurmadan önce).
  - **3. Adım**: Hash arabelleği başlatın: Hash işlevinin ara ve nihai sonuçlarını tutmak için 512 bitlik bir arabellek kullanılır. Tampon, sekiz adet 64 bitlik kayıt (a, b, c, d, e, f, g, h) olarak temsil edilebilir.
  - **4. Adım**: Mesajı 1024 bitlik (128 kelimelik) bloklar halinde işleyin, Algoritmanın kalbi, 80 döngüden oluşan bir modüldür; bu modül Şekil 21.2'de F olarak etiketlenmiştir.
  - **Adım 5**: Çıktı. Tüm N 1024 bitlik blok işlendikten sonra, N'inci aşamadaki çıktı, 512 bitlik mesaj özetidir.

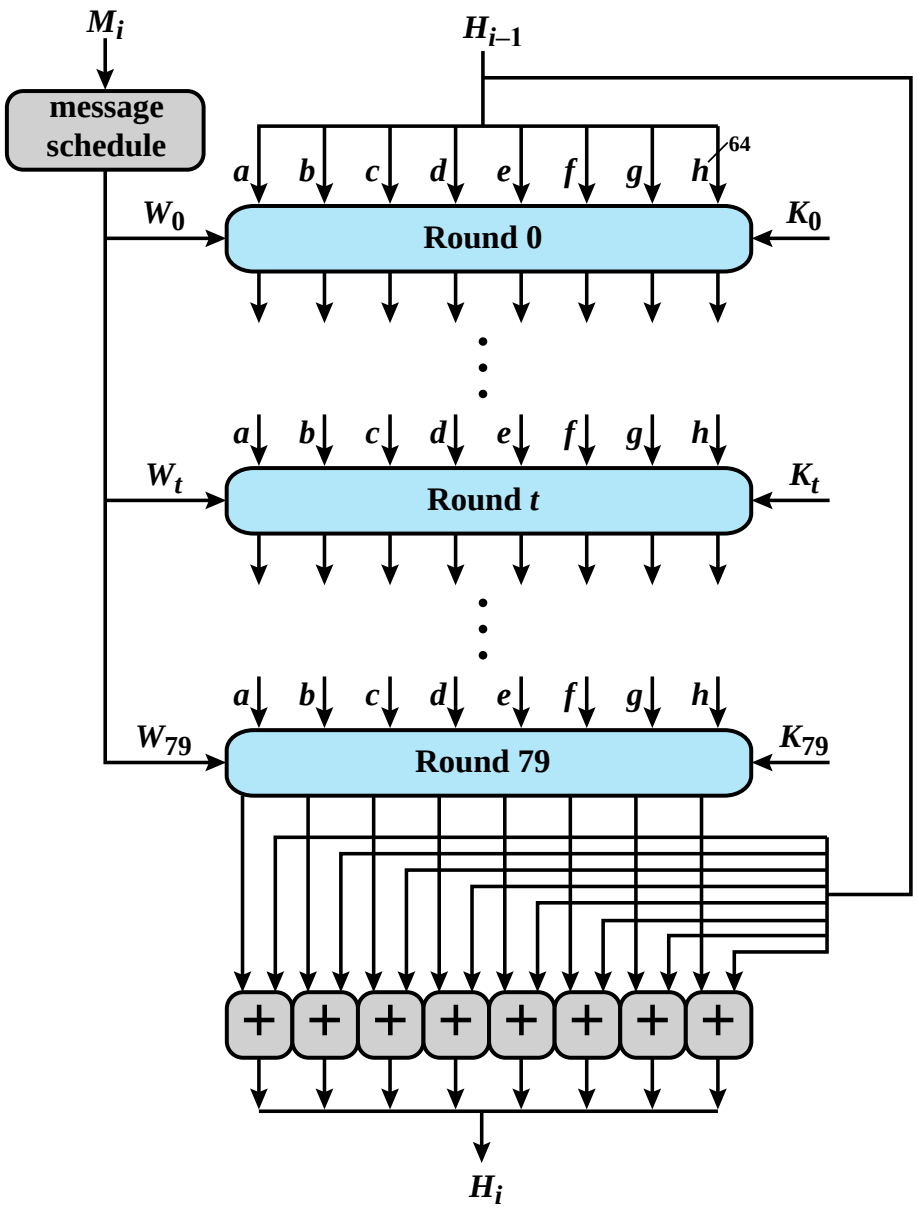


Figure 21.3 SHA-512 Processing of a Single 1024-Bit Block

SHA-3

- SHA-2, öncekilerle aynı yapıyı ve matematiksel işlemleri paylaşır ve endişeye neden olur
- Savunmasız hale gelmesi durumunda SHA-2'nin değiştirilmesi için gereken süre nedeniyle, NIST 2007'de SHA-3'ü üretmek için bir yarışma duyurdu.

Gereksinimler:

- 224, 256,384 ve 512 bitlik özet değer uzunluklarını desteklemelidir
- Algoritma, tüm mesajın işlenmeden önce bellekte arabelleğe alınmasını gerektirmek yerine küçük blokları bir seferde işlemelidir.

- NIST gönderilenlerden birini seçti ve SHA-3'ü FIPS 202 olarak resmen yayınladı (SHA-3 Standardı: Permutation-based Hash and Extendable-Output Functions, Ağustos 2015).
- SHA-3, SHA-2'nin yerini almaktan ziyade tamamlayıcısıdır.

HMAC

- Kryptografik hash kodundan türetilen bir MAC geliştirmeyle ilgilidir
  - Kriptografik hash fonksiyonları genellikle daha hızlı yürütülür
  - Kütüphane kodu yaygın olarak bulunur
  - SHA-1, gizli bir anahtara dayanmadığı için MAC olarak kullanılmak üzere tasarlanmamıştır.
- RFC2014 olarak yayınlandı
- IP güvenliği için uygulanması zorunlu MAC olarak seçilmiştir
  - Aktarım Katmanı Güvenliği (TLS) ve Güvenli Elektronik İşlem (Secure Electronic Transaction-SET) gibi diğer İnternet protokollerinde kullanılır

HMAC Tasarım Hedefleri

Mevcut hash fonksiyonlarını değişiklik yapmadan kullanmak

Önemli bir bozulma olmadan hash fonksiyonunun orijinal performansını korumak

Daha hızlı veya daha güvenli sağlama işlevlerinin bulunması veya gerekli olması durumunda gömülü sağlama işlevinin kolayca değiştirilebilmesini sağlamak

Anahtarları basit bir şekilde kullanmak ve işlemek

Katıştırılmış hash işlevine ilişkin makul varsayımlara dayalı olarak kimlik doğrulama mekanizmasının gücünün iyi anlaşılmış bir kriptografik analizine sahip olmak

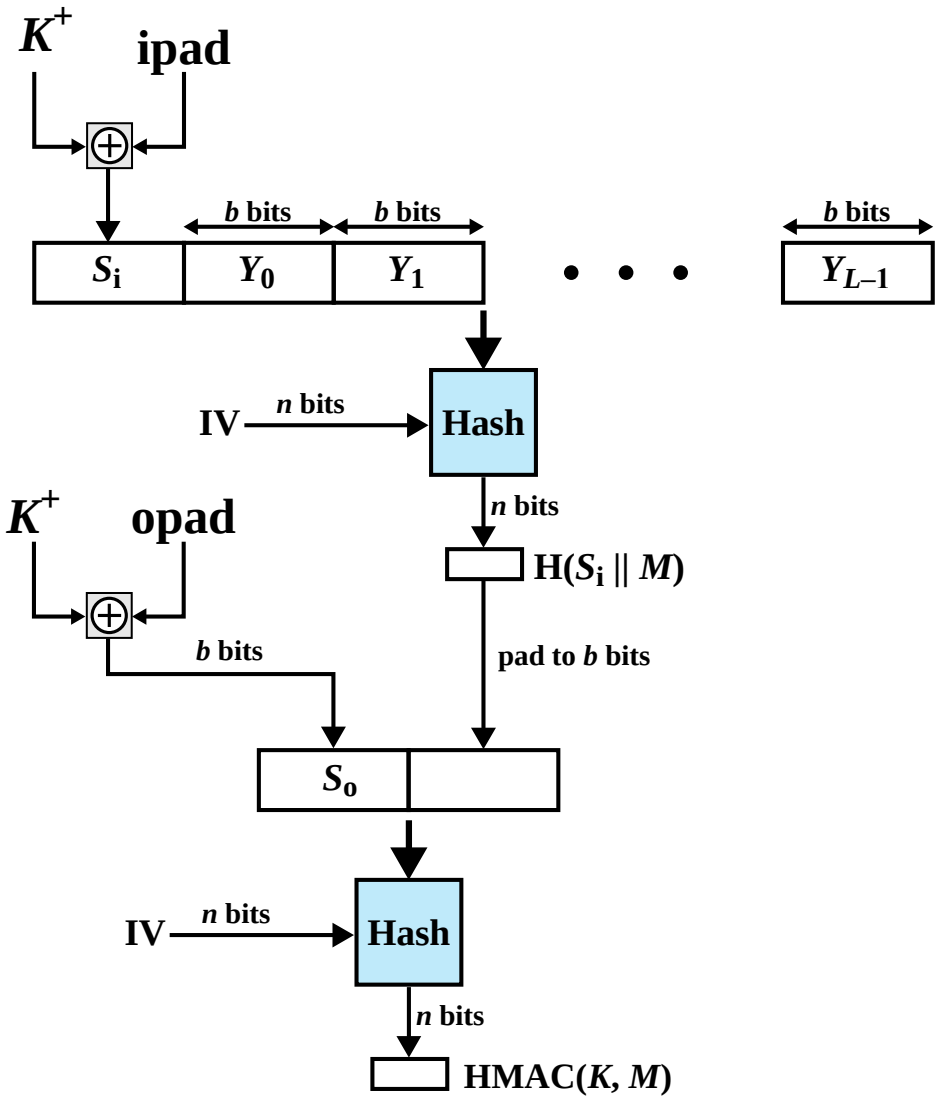
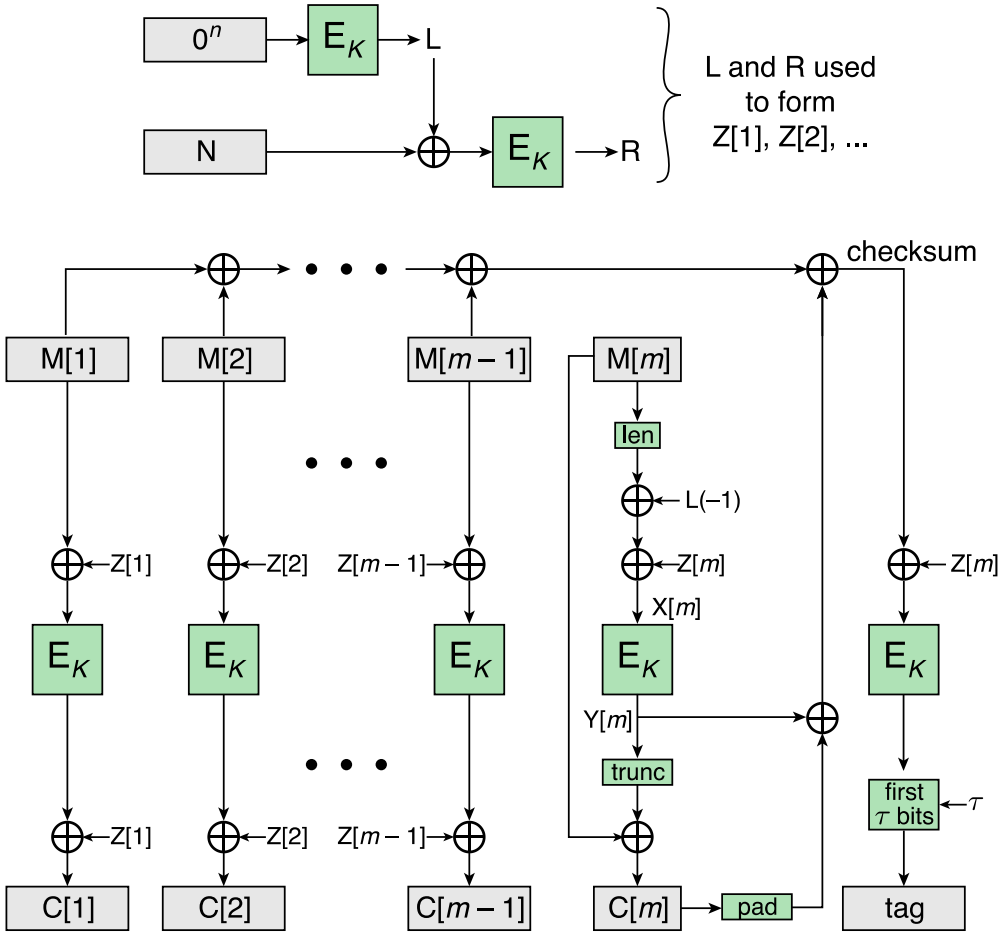


Figure 21.4 HMAC Structure

HMAC'ın Güvenliği

- Güvenlik, temeldeki hash işlevinin kriptografik gücüne bağlıdır.
- HMAC'ın çekiciliği, tasarımcılarının gömülü hash fonksiyonunun gücü ile HMAC'ın gücü arasındaki kesin ilişkiyi kanıtlayabilmesidir.
- Meşru bir kullanıcı tarafından oluşturulan ve saldırgan tarafından görülen iletiler üzerinde belirli bir çaba düzeyi için, HMAC'a başarılı saldırı olasılığı, gömülü (embedded) hash işlevine yapılan aşağıdaki saldırılardan birine eşdeğerdir:
    - Saldırgan, rastgele, gizli ve saldırgan tarafından bilinmeyen bir IV ile bile sıkıştırma işlevinin bir çıktısını hesaplayabilir.
    - Saldırgan, IV rastgele ve gizli olsa bile karma işlevinde çarpışmalar bulur.



n **= block length in bits**
N = nonce
len(M[m]) = length of M[m] represented as an n-bit integer
trunc(Y[m]) = deletes least significant bits so that result is same length as M[m]
pad = pad with least significant 0 bits to length n
τ = length of authentication tag

Figure 21.5 OCB Encryption and Authentication

algorithm OCB-Encrypt$_K$(N, M)	**algorithm** OCB-Decrypt$_K$(N, M)
Partition M into M[1]…M[m]	Partition M into M[1]…M[m]
$L \leftarrow L(0) \leftarrow E_K(0^n)$	$L \leftarrow L(0) \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$	$R \leftarrow E_K(N \oplus L)$
for i ← 1 **to** m **do** $L(i) \leftarrow 2 \cdot L(i-1)$	**for** i ← 1 **to** m **do** $L(i) \leftarrow 2 \cdot L(i-1)$
$L(-1) = L \cdot 2^{-1}$	$L(-1) = L \cdot 2^{-1}$
$Z[1] \leftarrow L \oplus R$	$Z[1] \leftarrow L \oplus R$
for i ← 2 **to** m **do** $Z[i] \leftarrow Z[i-1] \oplus L(\text{ntz}(i))$	**for** i ← 2 **to** m **do** $Z[i] \leftarrow Z[i-1] \oplus L(\text{ntz}(i))$
for i ← 1 **to** m – 1 **do**	**for** i ← 1 **to** m – 1 **do**
$C[i] \leftarrow E_K(M[i] \oplus Z[i]) \oplus Z[i]$	$M[i] \leftarrow D_K(C[i] \oplus Z[i]) \oplus Z[i]$
$X[m] \leftarrow \text{len}(M[m]) \oplus L(-1) \oplus Z[m]$	$X[m] \leftarrow \text{len}(M[m]) \oplus L(-1) \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$	$Y[m] \leftarrow E_K(X[m])$
$C[m] \leftarrow M[m] \oplus$ (first len(M[m]) bits of Y[m])	$M[m] \leftarrow$ (first len(C[m]) bits of Y[m]) $\oplus$ C[m]
Checksum ←	Checksum ←
$M[1] \oplus \ldots \oplus M[m-1] \oplus C[m]0^* \oplus Y[m]$	$M[1] \oplus \ldots \oplus M[m-1] \oplus C[m]0^* \oplus Y[m]$
Tag ← E_K(Checksum ⊕ Z[m]) [first t bits]	Tag' ← E_K(Checksum ⊕ Z[m]) [first t bits]

Figure 21.6 OCB Algorithms

RSA Açık Anahtarlı Şifreleme

- MIT'den Rivest, Shamir ve Adleman tarafından 1977'de
- En iyi bilinen ve yaygın olarak kullanılan açık anahtar algoritması
- Şifrele: $C = M^e \bmod n$
- *Şifre çözme:* $M = C^d \bmod n = (M^e)^d \bmod n = M$
- Hem gönderici hem de alıcı n ve e değerlerini bilir.
- Sadece alıcı d'nin değerini bilir
- Genel anahtar $PU = \{e, n\}$ ve özel anahtar $PR = \{d, n\}$ ile açık anahtar şifreleme algoritmasıdır

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Figure 21.7 The RSA Algorithm

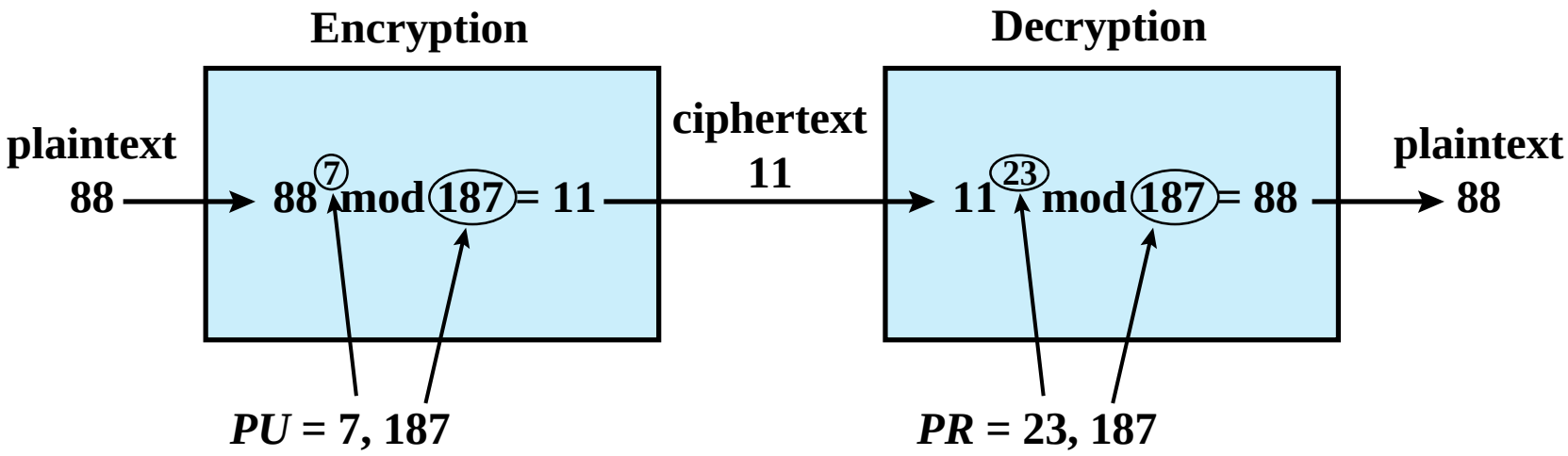


Figure 21.8 Example of RSA Algorithm

RSA'nın güvenliği

Kaba Kuvvet

- Tüm olası özel anahtarları denemeyi içerir

Matematiksel saldırılar

- İki asal sayının çarpımını çarpanlarına ayırma çabasında hepsi eşdeğer olan birkaç yaklaşım vardır

Zamanlama saldırıları

- Bunlar, şifre çözme algoritmasının çalışma süresine bağlıdır

Seçilmiş şifreli metin saldırıları

- Bu tür bir saldırı, RSA algoritmasının özelliklerinden yararlanır

Faktorizasyonda İlerleme

Ondalık Hane Sayısı	Bit Sayısı	Elde Edilme Tarihi
100	332	Nisan 1991
110	365	Nisan 1992
120	398	Haziran 1993
129	428	Nisan 1994
130	431	Nisan 1996
140	465	Şubat 1999
155	512	Ağustos 1999
160	530	Nisan 2003
174	576	Aralık 2003
200	663	Mayıs 2005
193	640	Kasım 2005
232	768	Aralık 2009

Zamanlama Saldırıları

- Bir kriptografik danışman olan Paul Kocher, bir meraklının, bir bilgisayarın mesajları deşifre etmesinin ne kadar sürdüğünü takip ederek özel bir anahtarı belirleyebileceğini gösterdi.
- Zamanlama saldırıları yalnızca RSA için değil, aynı zamanda diğer açık anahtarlı şifreleme sistemleri için de geçerlidir.
- Bu saldırı iki nedenden dolayı endişe vericidir:
    - Tamamen beklenmedik bir yönden gelir
    - Saldırı için sadece bir tane şifreli metin yeterlidir

Zamanlama Saldırısına Karşı Tedbirler

Sabit üs alma süresi

- Bir sonuç döndürmeden önce tüm üslerin aynı süreyi aldığından emin olunur
- Bu basit bir düzeltmedir ancak performansı düşürür

Ratgele gecikme

- Zamanlama saldırısının kafasını karıştırmak için üs alma algoritmasına rastgele bir gecikme eklenerek daha iyi performans elde edilebilir
- Savunmacılar yeterince gürültü eklemezse, saldırganlar rastgele gecikmeleri telafi etmek için ek ölçümler toplayarak yine de başarılı olabilir

Körleştirme

- Üs alma işlemini gerçekleştirmeden önce şifreli metni rastgele bir sayı ile çarpılır
- Bu işlem, saldırganın bilgisayarda hangi şifreli metin bitlerinin işlendiğini bilmesini engeller ve bu nedenle zamanlama saldırısı için gerekli olan bit-bit analizini engeller.

Diffie-Hellman Anahtar Değişimi

- İlk yayınlanan açık anahtar algoritması
- Diffie ve Hellman tarafından 1976'da açık anahtar kavramlarının açıklanmasıyla birlikte sunulmuştur
- Bir dizi ticari üründe kullanılır
- Mesajların daha sonra şifrelenmesi için kullanılabilecek bir gizli anahtarı güvenli bir şekilde değiş tokuş etmenin pratik yöntemidir
- Güvenlik, ayrık logaritmaların hesaplanmasının zorluğuna dayanır

Global Public Elements

q	prime number
a	$a < q$ and a a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = a^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = a^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Figure 21.9 The Diffie-Hellman Key Exchange Algorithm

Diffie-Hellman Örneği

Diyelim ki

- Asal sayımız q = 353
- Primitif kökü α = 3

A ve B'nin her biri ortak anahtarlarını hesaplar

- A, YA = 3^{97} mod 353 = 40'ı hesaplar
- B, YB = 3^{233} mod 353 = 248'i hesaplar

Ardından gizli anahtarı değiştir ve hesapla:

- A için: K = $(Y_B)^{XA}$ mod 353 = 248^{97} mod 353 = 160
- B için: K = $(Y_A)^{XB}$ mod 353 = 40^{233} mod 353 = 160

Saldırganın çözmesi gerekenler::

- 3^a mod 353 = 40 ki bu zor
- İstenen cevap 97'dir, ardından anahtarı B'nin yaptığı gibi hesaplayın

Alice **Bob**

Alice	Bob
Alice and Bob share a prime q and , such that $< q$ and is a primitive root of q	Alice and Bob share a prime q and , such that $< q$ and is a primitive root of q
Alice generates a private key X_A such that $X_A < q$	Bob generates a private key X_B such that $X_B < q$
Alice calculates a public key $Y_A = {}^{X_A} \bmod q$	Bob calculates a public key $Y_B = {}^{X_B} \bmod q$
Alice receives Bob's public key Y_B in plaintext	Bob receives Alice's public key Y_A in plaintext
Alice calculates shared secret key $K = (Y_B)^{X_A} \bmod q$	Bob calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

Y_A

Y_B

Figure 21.10 Diffie-Hellman Key Exchange

Ortadaki Adam Saldırısı

- Saldırı:
  1. Darth X_{D1}veX_{D2}özel anahtarlar ve ortak anahtarlarıY_{D1}veY_{D2} üretir
  2. Alice Y_A ′yı Bob′a iletir
  3. Darth, Y_A'yı yakalar ve Y_{D1}'i Bob'a iletir. Darth ayrıca K2'yi de hesaplar
  4. Bob Y_{D1} ′i alır ve K1'i hesaplar
  5. Bob X_A ′i Alice′e iletir
  6. Darth, X_A'yı yakalar ve Y_{D2}'yi Alice'e iletir. Darth K1'i hesaplar
  7. Alice Y_{D2}′i alır ve K2'yi hesaplar
- Sonraki tüm iletişimler tehlikeye girer

Diğer Açık Anahtar Algoritmaları

Elektronik imza Standardı (DSS)

- FIPS PUB 186
- SHA-1 ve Dijital İmza Algoritmasını (DSA) kullanır
- İlk olarak 1991'de önerildi, güvenlik endişeleri nedeniyle 1993'te revize edildi ve 1996'da başka bir küçük revizyon geçirdi
- Şifreleme veya anahtar değişimi için kullanılamaz
- Yalnızca dijital imza işlevini sağlamak için tasarlanmış bir algoritmadır

Eliptik Eğri Kriptografisi (ECC)

- RSA'dan daha küçük bit boyutu için eşit güvenlik sağlar
- IEEE P1363 gibi standartlarda görülür
- ECC'deki güven düzeyi henüz RSA'daki kadar yüksek değildir
- Eliptik eğri olarak bilinen matematiksel bir yapıya dayanır