

# اصرار اصلت پیام و رمزگاری کلید عمومی

(انشگاه صنعتی امیرکبیر

(انشگاه مهندسی کامپیوتر و فناوری اطلاعات

توسط: حمید، خا شهریاری

# فهرست

- ♦ روشهای اثبات احیان پیام
- ♦ HMAC , Secure Hash Function
- ♦ اصول, مزگاری کلید عمومی
- ♦ الگوریتمهای, مزگاری کلید عمومی
- ♦ امنیتی, قمی
- ♦ مدیریت کلید

# امراز اصالت

- ♦ نیازمندیها: باید قادر به بررسی موارد زیر باشد:
  - پیام از سوی منبع یا نویسنده ادعای شده می‌آید
  - محتوا تغییر نکرده است
  - در برخی موارد، آیا در زمان معین یا در ترتیب خاصی آمده است
  - ♦ حفاظت از حملات فعال

(falsification of data and transactions) ■

# رهیافت‌های اهراز اصالت

- ♦ اهراز اصالت با رمزگاری معمولی
  - تنها خرستنده و گیرنده باید کلید را داشته باشند.
- ♦ اهراز اصالت بدون رمزگردان پیام
  - یک برعسب اهراز اصالت تولید و به هر پیام پیوست می‌شود.
  - کد اهراز اصالت پیام

## (Message Authentication Code)

بر حسب تابعی از پیام و کلید تولید می‌شود: **MAC** •

$$MAC = F(K_{AB}, M)$$
 •

خوب می‌شود که تنها **A** و **B** از کلید اطلاع دارند.

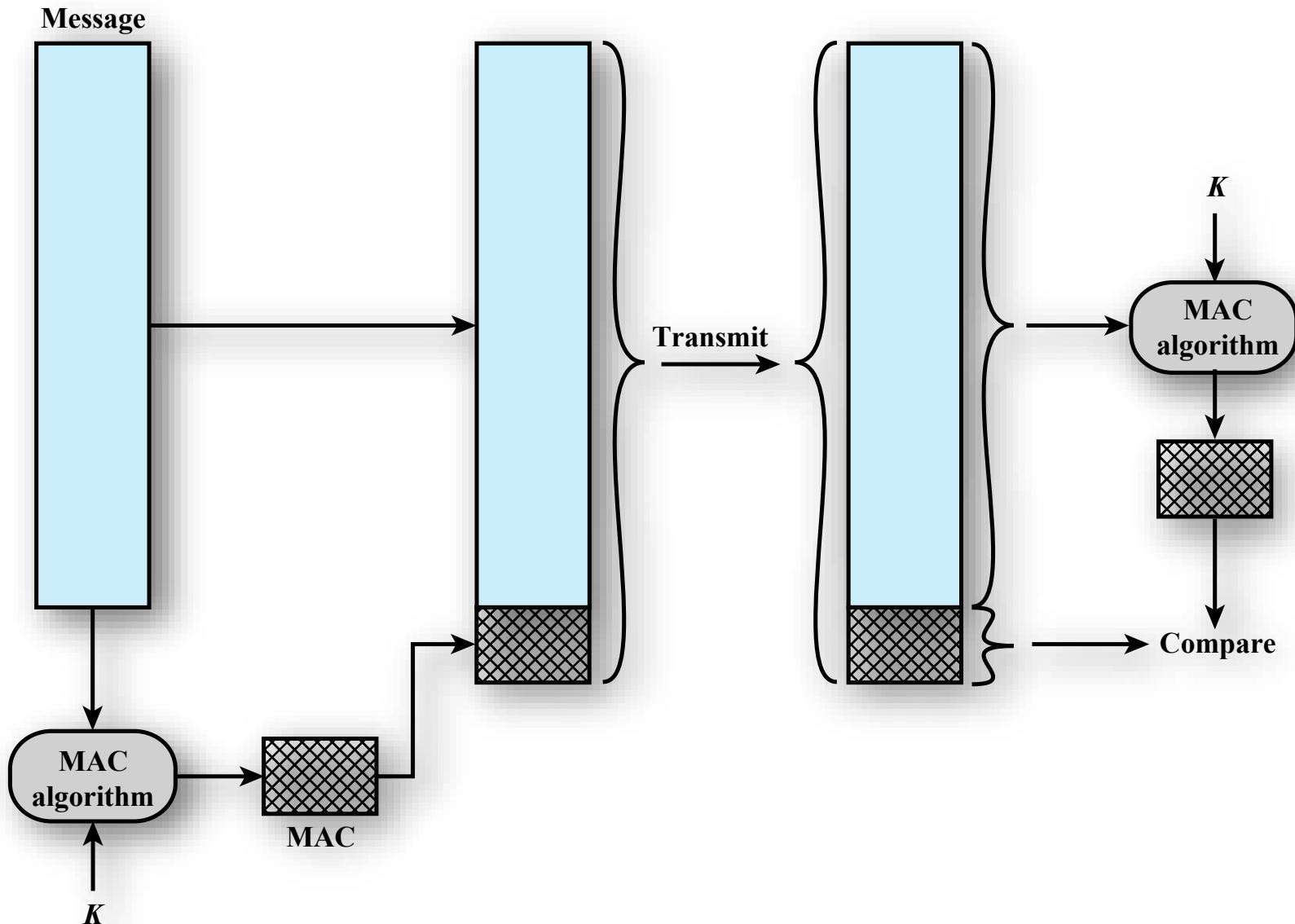


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

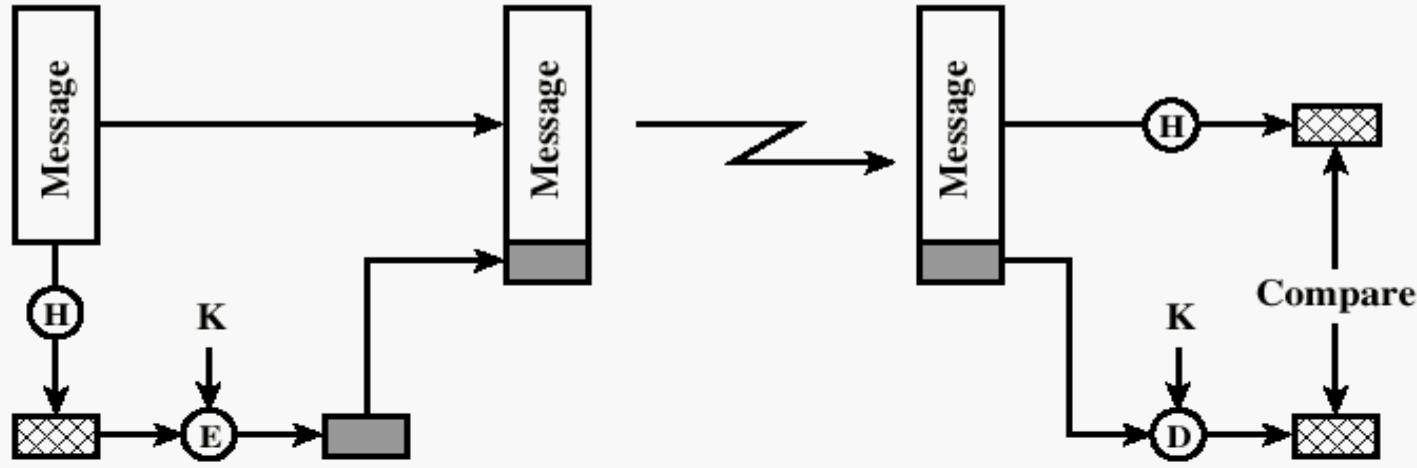
# امراز اصالت پیام

- ♦ **MAC** نیازمندیهای زیر را برآورده می‌کند:
  - پیام از سوی منبع یا نویسنده ادعای شده می‌آید،
  - محتوا تغییر نکرده است،
  - در برابر موارد، پیام در زمان معین یا در ترتیب خاصی آمده است.

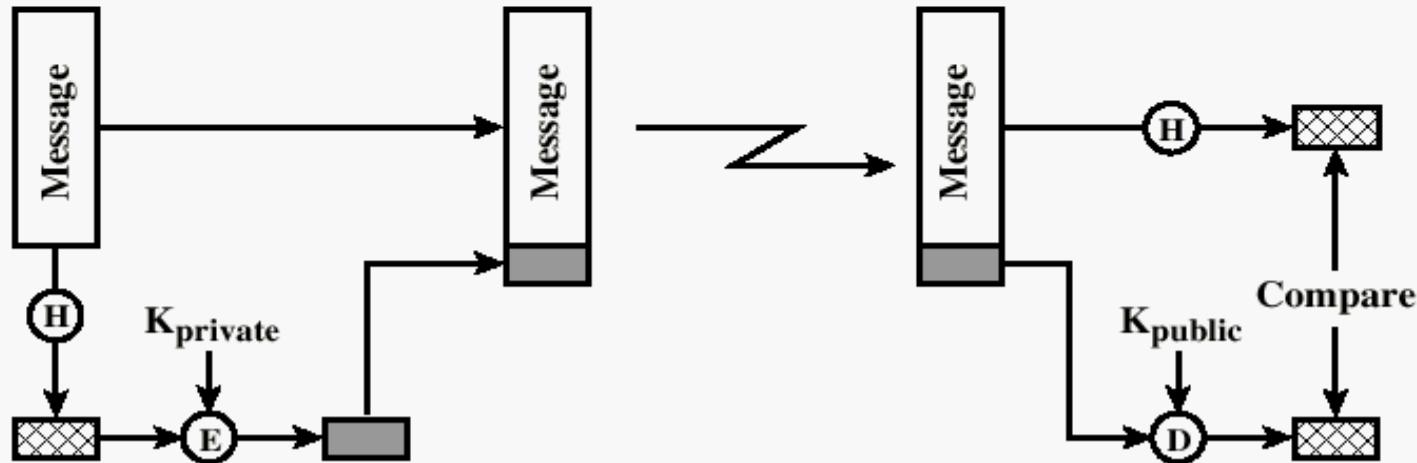
# اصرار اصالت پیام

- ◆ پرا با، مزنگاری معمولی انجام ندهیم؟!
- موارد زیر را در نظر بگیرید:
  - تعدادی برنامه کاربردی که پیامها را به همه می خرستند (**broadcast**). مثلا هشدار به کاربران یک سایت.
  - گیرنده پیام بار پردازشی سنگینی دارد و نمی تواند رمزگشایی کند.
  - اصرار اصالت برنامه ها برای جلوگیری از تغییر غیر مجاز برنامه (مثلاً توسط ویروسها)

# روش‌های مختلف امراز اصلاح پیام



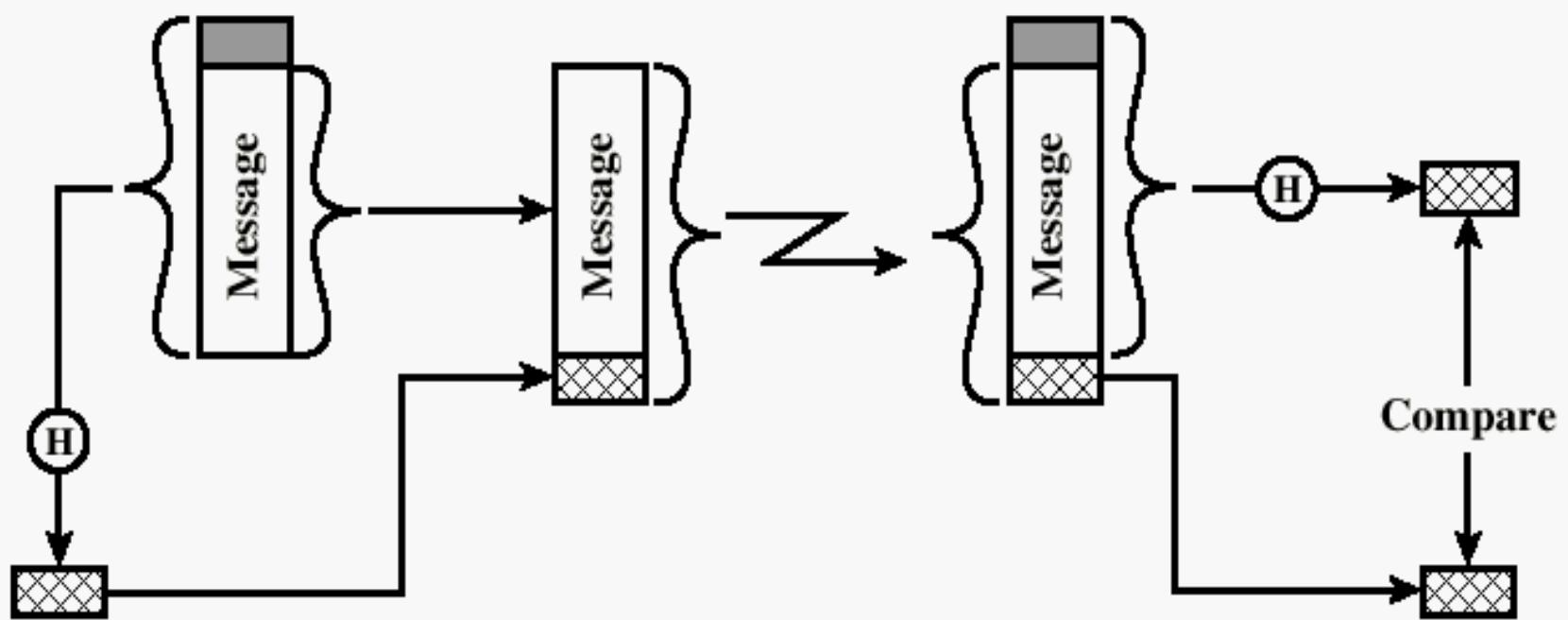
(a) Using conventional encryption



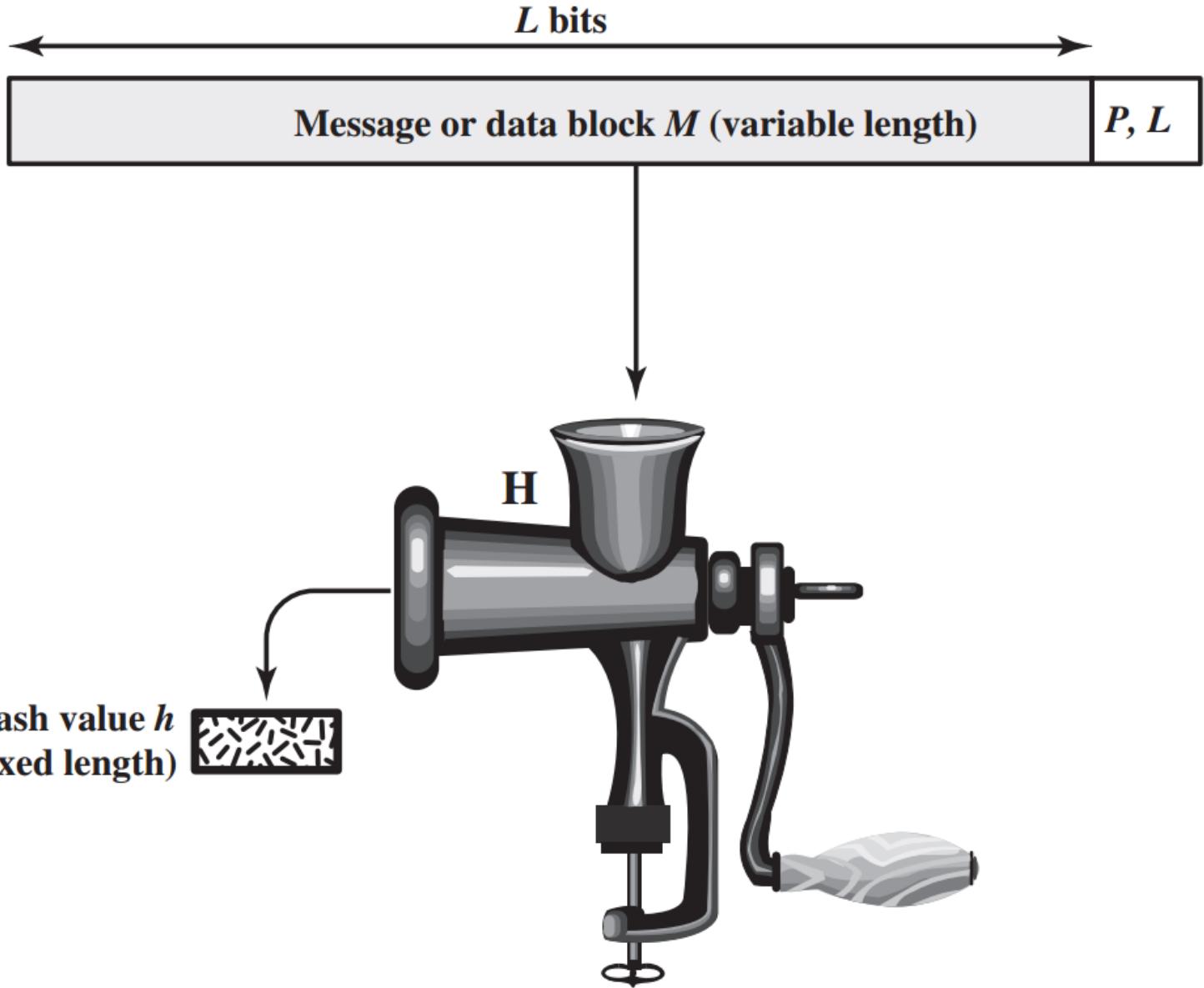
(b) Using public-key encryption

# روش‌های مختلف اثبات اصالت پیام

- ♦ روش سوم بدون همزگاری: داده مخفی قبل از درهم سازی اضافه شده و قبل از فرستادن حذف می‌شود.



(c) Using secret value

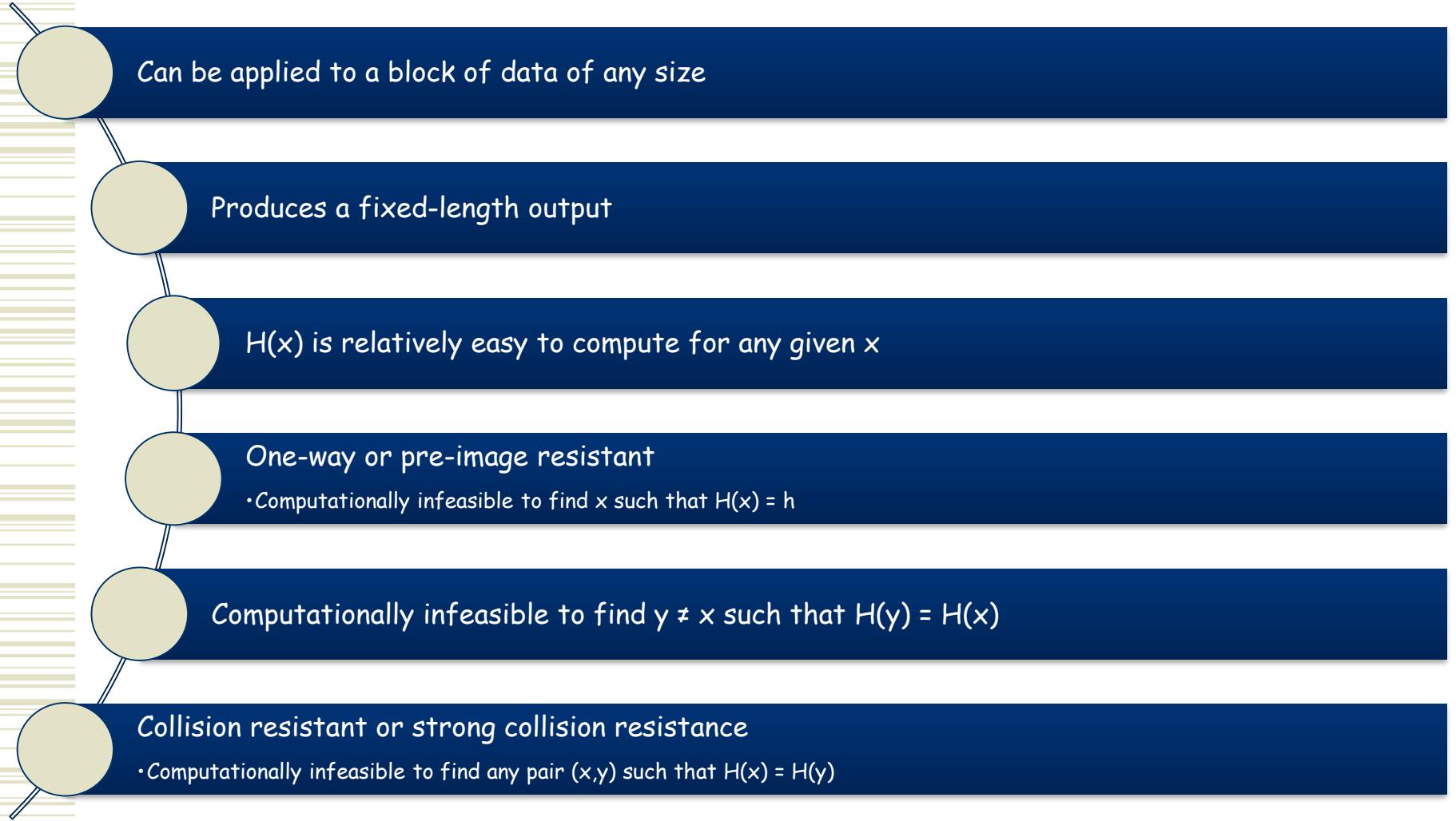


$P, L$  = padding plus length field

# توابع درهم سازی امن

- ◆ هدف از تابع درهم سازی تولید یک "اثر انگشت" منحصر به فرد است.
- ◆ بنابراین تابع درهم سازی باید ویژگی های زیر داشته باشد:
  ۱. قابلیت اعمال به هر اندازه از داره ها
  ۲. خروجی طول ثابتی داشته باشد
  ۳. مهاسبه  $H(x)$  برای هر مقدار  $x$  ساده باشد
  ۴. برای هر بلوک  $x$ ، بدست آوردن  $x$  از  $H(x)$  غیر ممکن باشد
  ۵. برای هر بلوک  $x$ ، بدست آوردن بلوک دیگر مانند  $y$  که  $H(x)=H(y)$  باشد غیر ممکن باشد.
  ۶. پیدا کردن زوج هایی مانند  $(x,y)$  که  $H(x)=H(y)$  شود غیر ممکن باشد.

# Hash Function Requirements



# یک تابع درهم سازی ساده

- ❖ خرض کنید که داده ورودی (مانند فایل) (نباله ای از بلوک های  $n$  بیتی باشد.
  - ❖  $C_i$ : بیت  $i$ م ک درهم سازی
  - ❖  $m$  : تعداد بلوکها
  - ❖  $b_{ij}$ : بیت  $i$ م در بلوک  $j$
- $$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

# یک تابع درهم سازی ساده

	bit 1	bit 2	•	•	bit <i>n</i>
block 1	b <sub>11</sub>	b <sub>21</sub>			b <sub><i>n</i>1</sub>
block 2	b <sub>12</sub>	b <sub>22</sub>			b <sub><i>n</i>2</sub>
	•	•	•	•	•
	•	•	•	•	•
	•	•	•	•	•
block <i>m</i>	b <sub>1<i>m</i></sub>	b <sub>2<i>m</i></sub>			b <sub><i>n</i><i>m</i></sub>
hash code	C <sub>1</sub>	C <sub>2</sub>			C <sub><i>n</i></sub>

Figure 3.3 Simple Hash Function Using Bitwise XOR

# *Secure Hash Algorithm (SHA)*

- ♦ توسط موسسه NIST در سال ۱۹۹۳ اعلام شد و به عنوان استاندارد FIPS-180 منتشر شد.
- ♦ نسخه جدیدتر آن با نام SHA-1 در سال ۱۹۹۵ به عنوان استاندارد FIPS-180-1 معرفی شد.
- ♦ بر اساس الگوریتم درهم سازی MD4
- ♦ ورودی هر پیام با طول کمتر از  $2^{64}$  بیت و خروجی ۱۶۰ بیت
- ♦ SHA-2 در ۲۰۰۵ کنار گذاشتن SHA-1 و حرکت به سمت NIST تا ۲۰۱۰ را اعلام کرد.
- ♦ در ۲۰۰۲ نسخه جدید FIPS-180-2 منتشر شد که خانواده SHA-2 شامل SHA-256, SHA-384, SHA-512 معرفی شد.

# Secure Hash Algorithm (SHA)

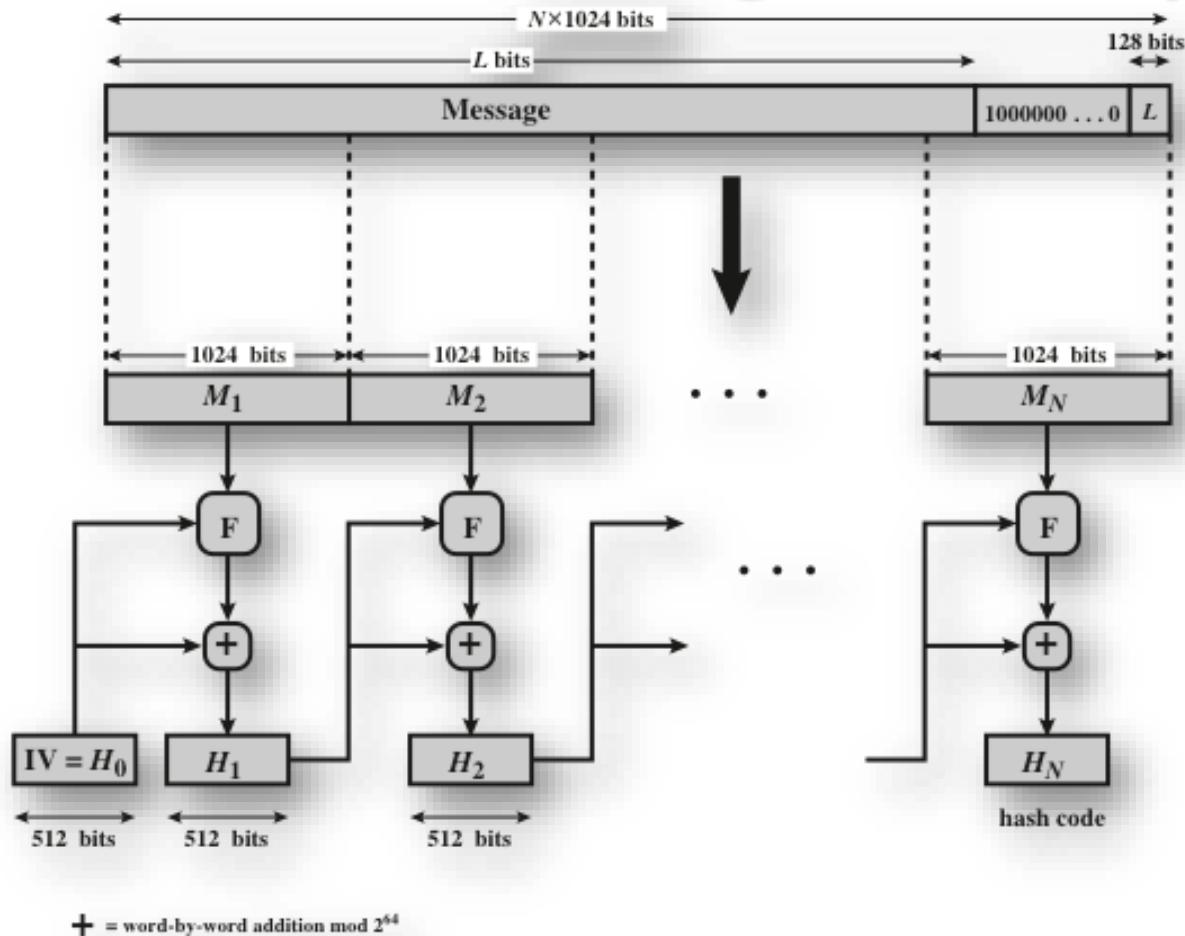


Figure 3.4 Message Digest Generation Using SHA-512  
<http://ceit.aut.ac.ir/~shahriari>

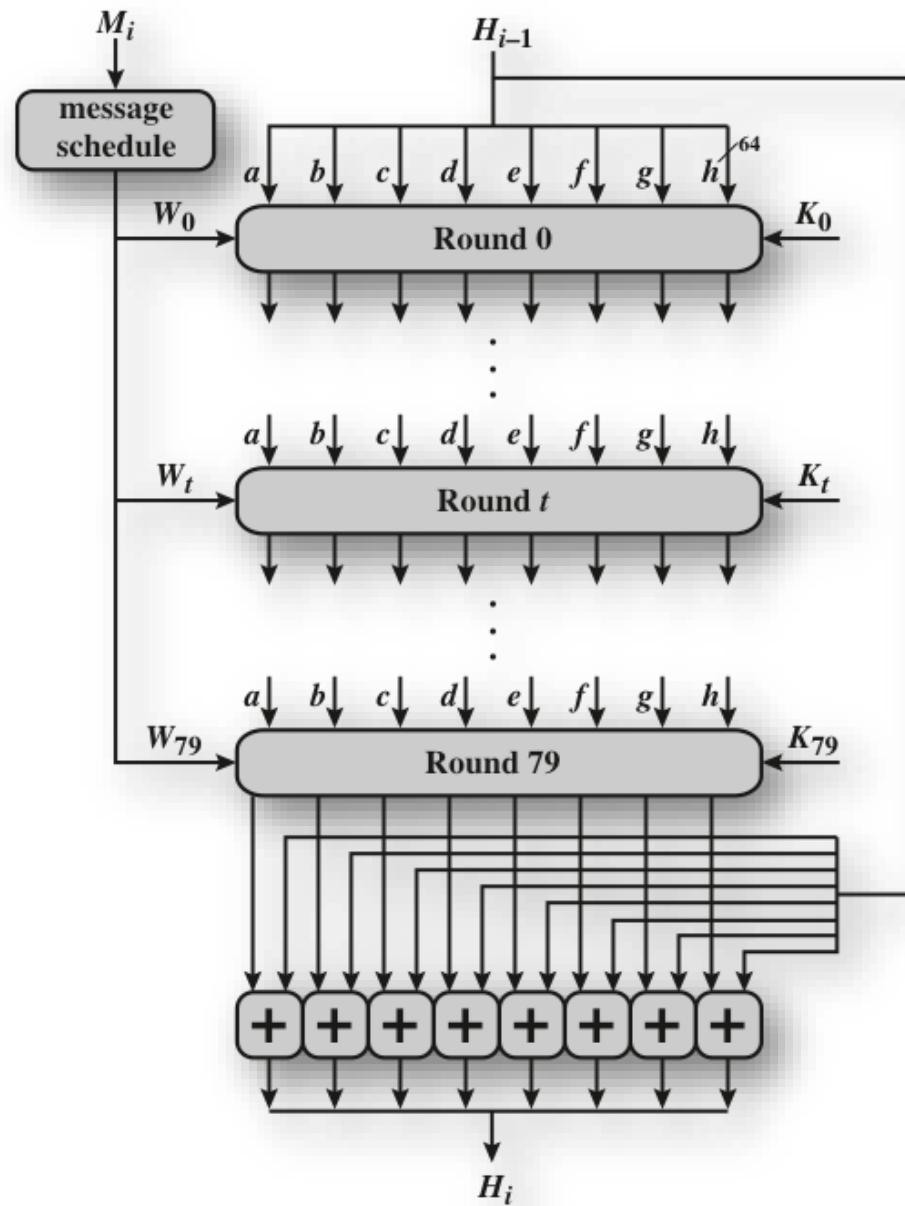


Figure 3.5 SHA-512 Processing of a Single 1024-Bit Block

# **SHA-3**

- ♦ NIST ◆  
● مسابقه‌ای را جهت انتخاب الگوریتم جدید  
● اعلام کرد.
- ♦ Keccak ◆  
● سال ۲۰۱۳ الگوریتم انتخاب شد و به عنوان  
● اعلان شد.

# **SHA-3**

1. It must be possible to replace SHA-2 with SHA-3 in any application by a simple drop-in substitution. Therefore, SHA-3 must support hash value lengths of 224, 256, 384, and 512 bits.

2. SHA-3 must preserve the online nature of SHA-2. That is, the algorithm must process comparatively small blocks (512 or 1024 bits) at a time instead of requiring that the entire message be buffered in memory before processing it.

Basic requirements that must be satisfied by any candidate for SHA-3

# دیگر الگوریتمهای تولید پسکده پیام

MD5 ◆

- (RFC 1321) Rivest توسعه
- ورودی به طول دلفواه و خروجی ۱۲۸ بیت
- در سال ۲۰۰۴ شناسته شد!

# **RIPMED-160**

- ♦ طی پروژه RIPE در اتحادیه اروپا در سال ۱۹۹۷ ارائه شد.
- ♦ ورودی بلوکهای ۵۱۲ بیتی
- ♦ در ابتدا ۱۲۸ بیتی بود که با یافتن حملاتی به برخی دورهای آن به ۱۶۰ بیت ارتقا یافت.

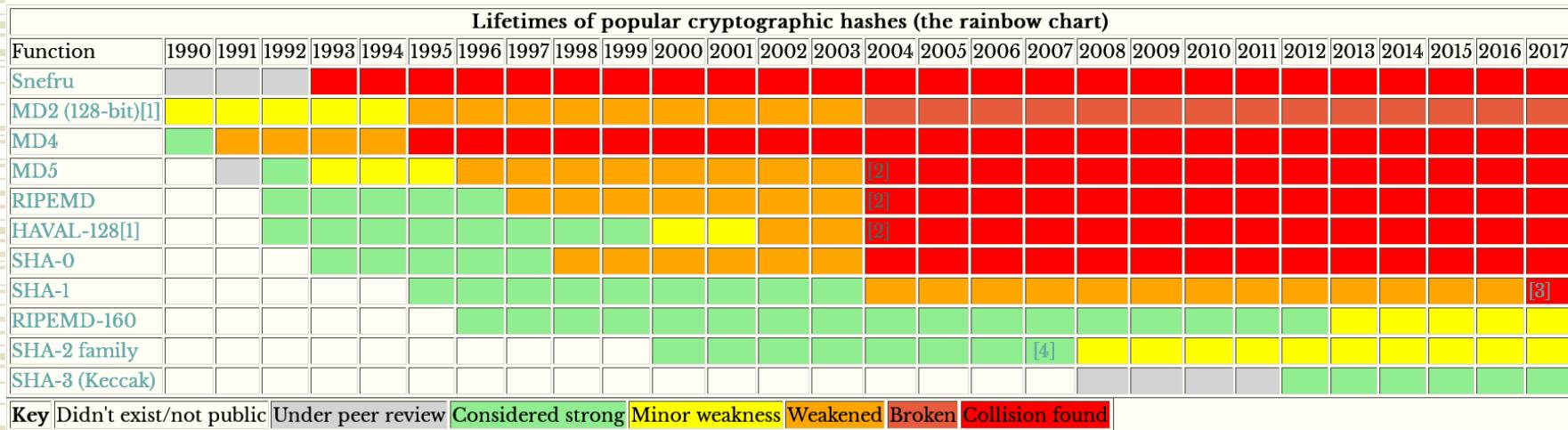
# rules

	SHA-1	MD5	RIPEMD-160
Digest length	160 bits	128 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	64 (4 rounds of 16)	160 (5 paired rounds of 16)
Maximum message size	$2^{64}-1$ bits	$\infty$	$\infty$

# مقایسه فانکواده SHA

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
<b>Message Digest Size</b>	160	224	256	384	512
<b>Message Size</b>	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
<b>Block Size</b>	512	512	512	1024	1024
<b>Word Size</b>	32	32	32	64	64
<b>Number of Steps</b>	80	64	64	80	80

# مقایسه چرخه میان توابع درهم سازی



Source: <http://valerieaurora.org/hash.html>

# HMAC

♦ استفاده از یک روش MAC مبتنی بر یک کد در هم سازی مانند SHA-1

♦ انگلیزه ها:

- توابع در هم سازی از توابع مزنگاری مانند DES سریعتر اجرا می شوند.
- توابع کتابخانه ای برای توابع در هم سازی به وفور در دسترس است.
- عدم وجود محدودیتهای صادرات از طرف آمریکا

# HMAC

- ♦ در حال حاضر HMAC بخش لازم در پیاده سازی IP Security است.
- ♦ اهداف طراحی HMAC (که در RFC 2104 لیست شده):
  - استفاده بدون تغییر توابع در هم سازی
  - جایگزینی آسان توابع در هم سازی
  - حفظ کارآیی اولیه توابع در هم سازی
  - استفاده ساده از کلیدها
  - امکان تحلیل آسان

# HMAC جیوه

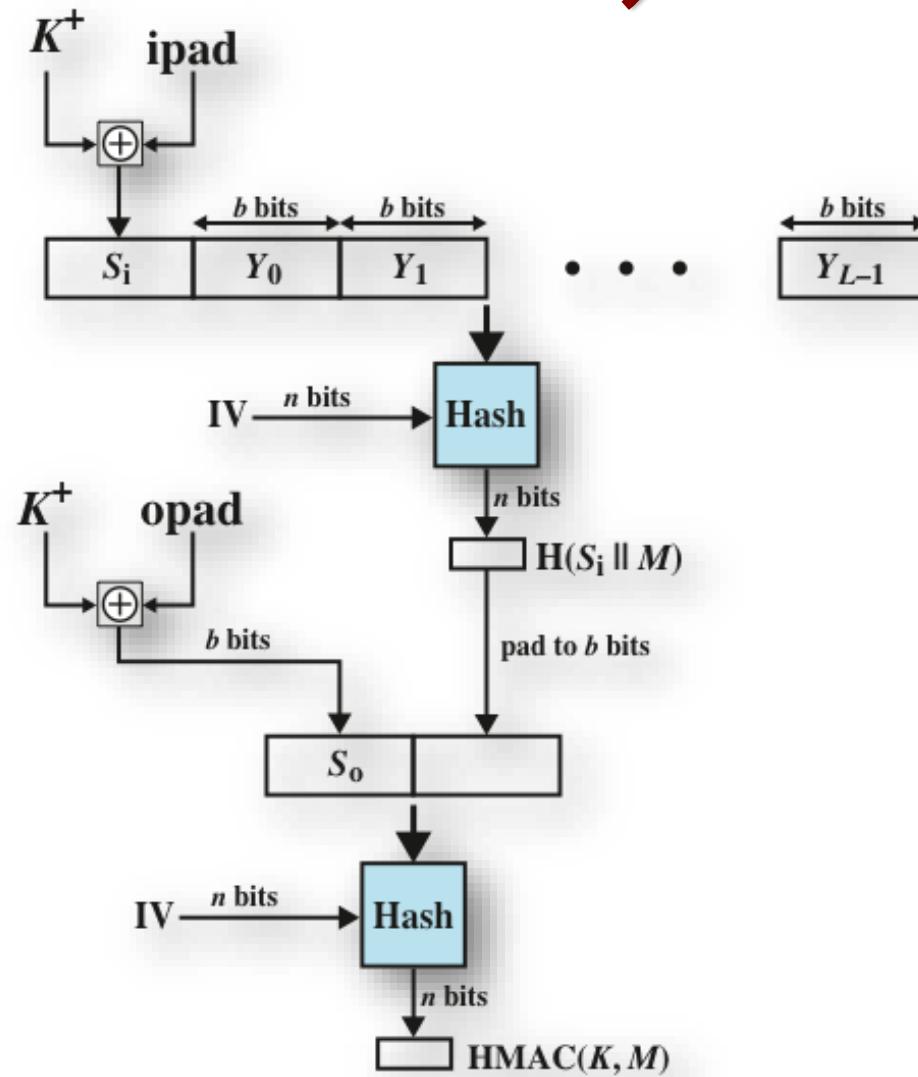


Figure 3.6 HMAC Structure  
<http://ceit.aut.ac.ir/~shahriari>

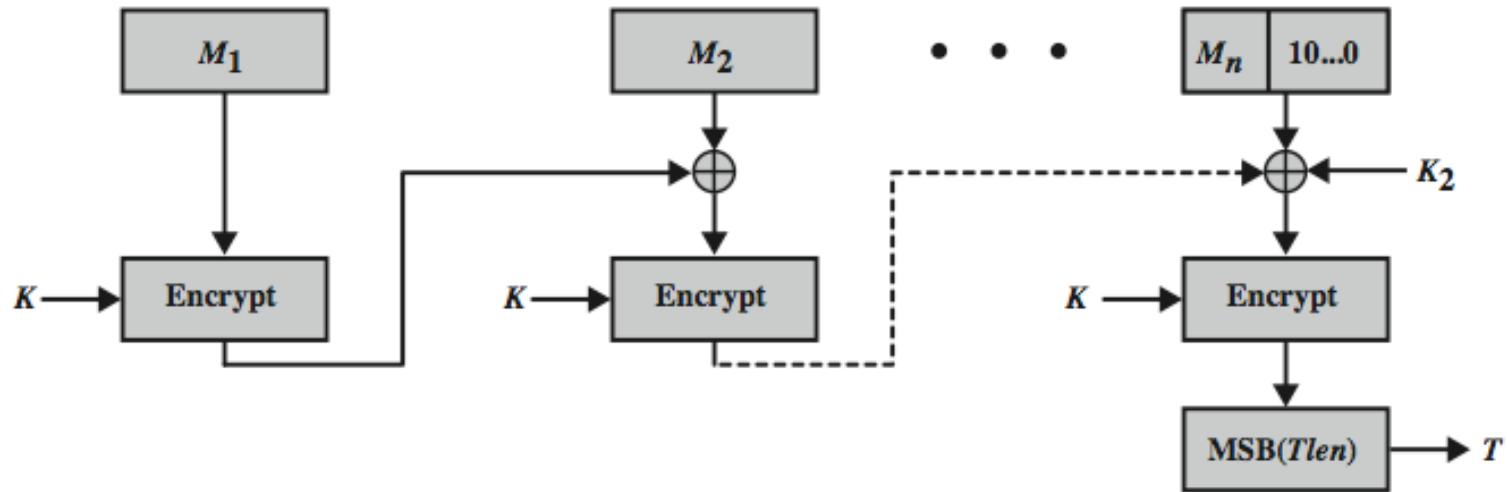
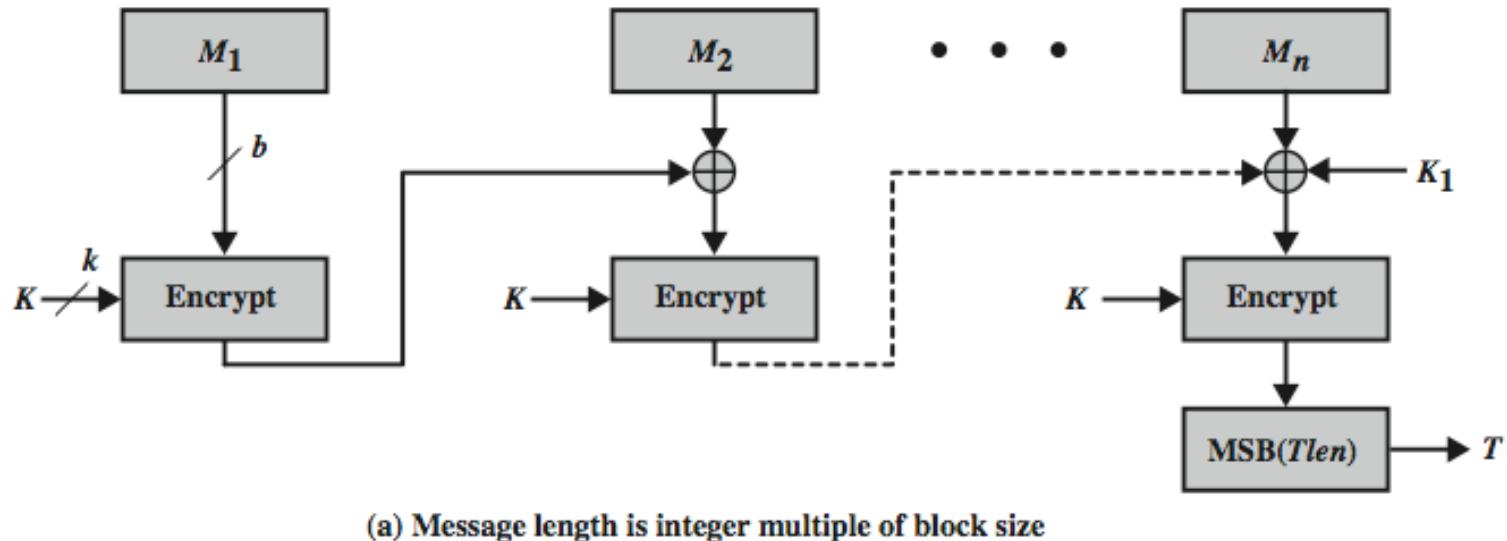
# **HMAC Security**

- ◆ proved security of HMAC relates to that of the underlying hash algorithm
- ◆ attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- ◆ choose hash function used based on speed verses security constraints

# **CMAC**

- ◆ widely used in govt & industry
- ◆ but has message size limitation
- ◆ can overcome using 2 keys & padding
- ◆ thus forming the Cipher-based Message Authentication Code (**CMAC**)
- ◆ adopted by NIST SP800-38B

# CMAC Overview



(b) Message length is not integer multiple of block size  
<http://ceit.aut.ac.ir/~shahriari>

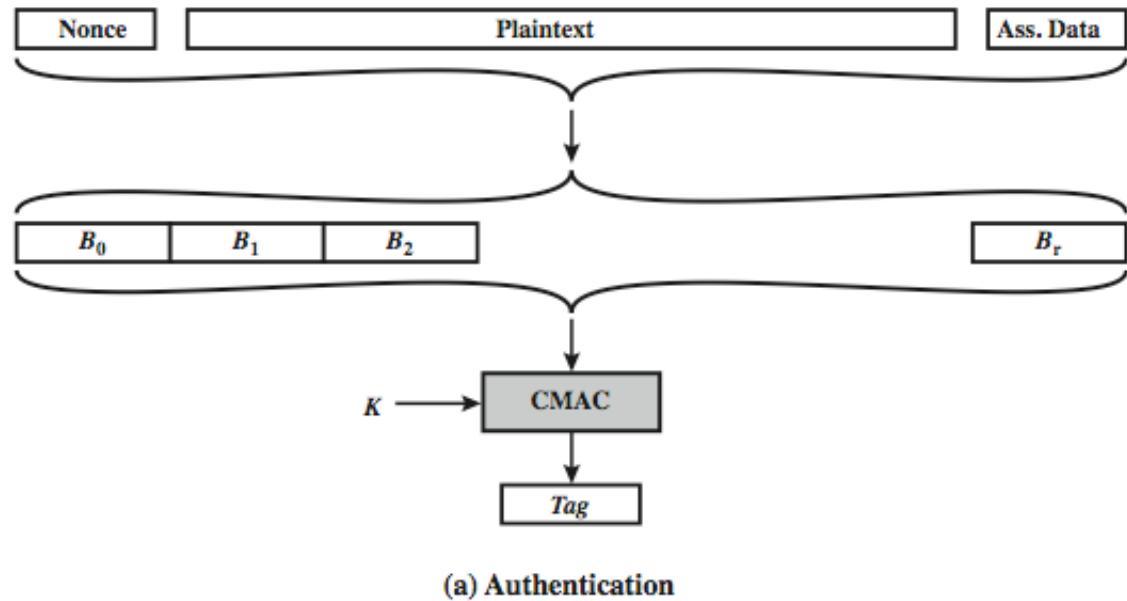
# *Authenticated Encryption*

- simultaneously protect confidentiality and authenticity of communications
  - often required but usually separate
- approaches
  - Hash-then-encrypt:  $E(K, (M \parallel H(M)))$
  - MAC-then-encrypt:  $E(K_2, (M \parallel MAC(K_1, M)))$
  - Encrypt-then-MAC: ( $C=E(K_2, M)$ ,  $T=MAC(K_1, C)$ )
  - Encrypt-and-MAC: ( $C=E(K_2, M)$ ,  $T=MAC(K_1, M)$ )
- decryption /verification straightforward
- but security vulnerabilities with all these

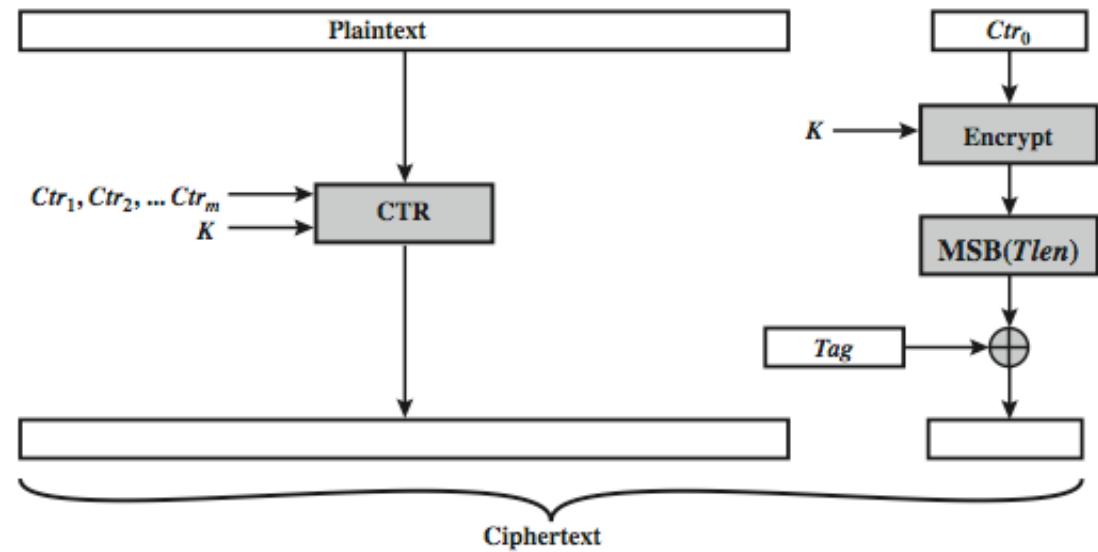
# *Counter with Cipher Block Chaining-Message Authentication Code (CCM)*

- ◆ NIST standard SP 800-38C for WiFi
- ◆ variation of encrypt-and-MAC approach
- ◆ algorithmic ingredients
  - AES encryption algorithm
  - CTR mode of operation
  - CMAC authentication algorithm
- ◆ single key used for both encryption & MAC

# CCM Operation



(a) Authentication



(b) Encryption  
<http://ceit.aut.ac.ir/~shahriari>

هزنگاری نامتقاض (کلید عمومی)

# مبانی رمزنگاری کلید عمومی

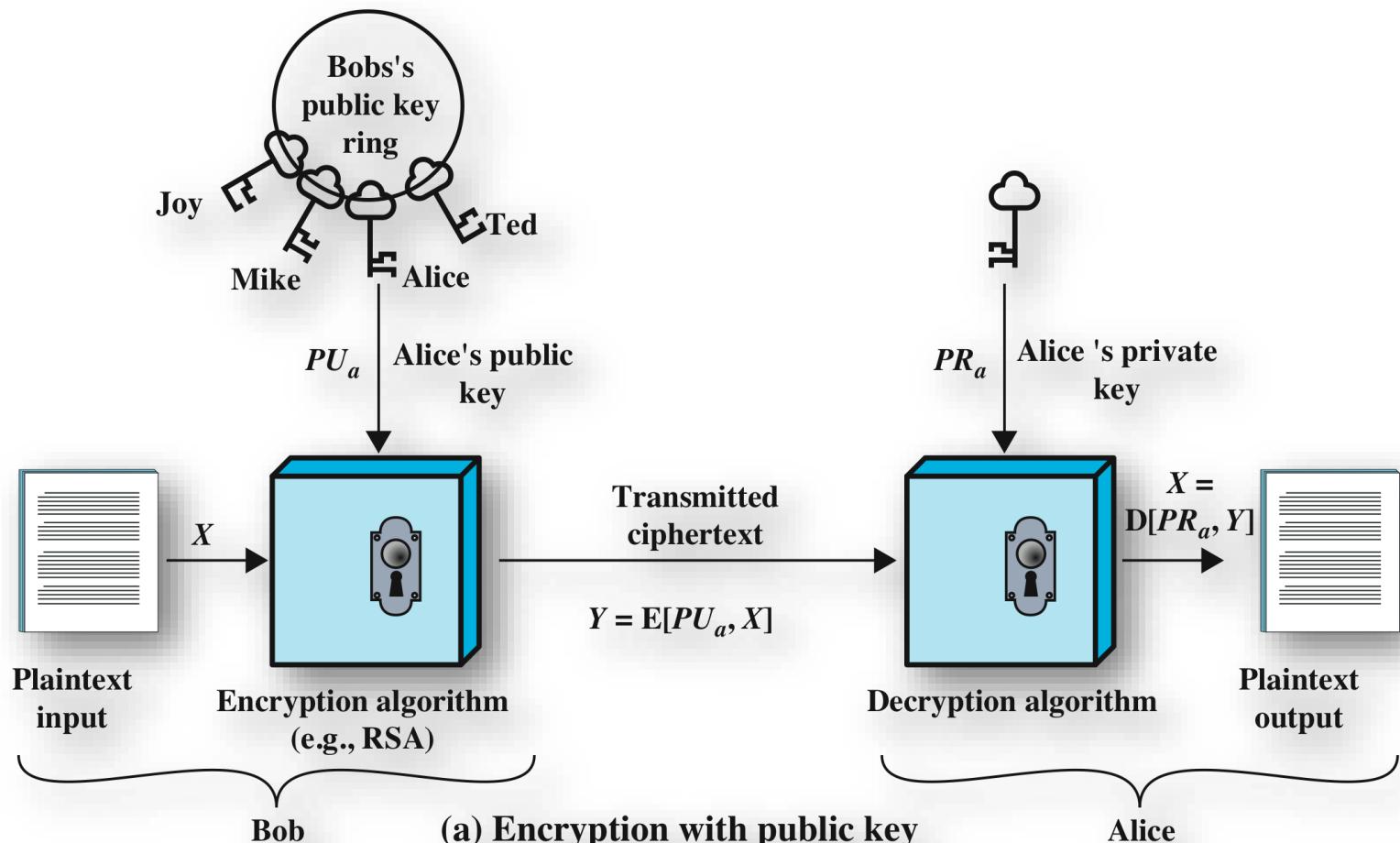


- ♦ رمزنگاری کلید عمومی اساساً با انگیزه سیدن به دو هدف طراحی شد:
  - حل مساله توزیع کلید در روشای رمزنگاری متقارن،
  - امنیتی دیجیتال
- ♦ دیفی و هلمن اولین راه حل را در ۱۹۷۶ ارایه دادند.

# کاربردهای رمزگاری کلید عمومی

- ♦ سه دسته کاربرد:
  - **مهمانگی:** خرستنده پیام را با کلید عمومی گیرنده رمز می کند.
  - **احراز احیالت یا امضا:** خرستنده پیام را با کلید خصوصی خود امضا می کند.
  - **تبادل کلید:** دو طرف با همکاری هم کلید یک نشست را مبادله می کنند.

# کاربرد محرمانگی: رمزگذاری با کلید عمومی



# کاربرد احراز اصالت: رمزگذاری با کلید خصوصی

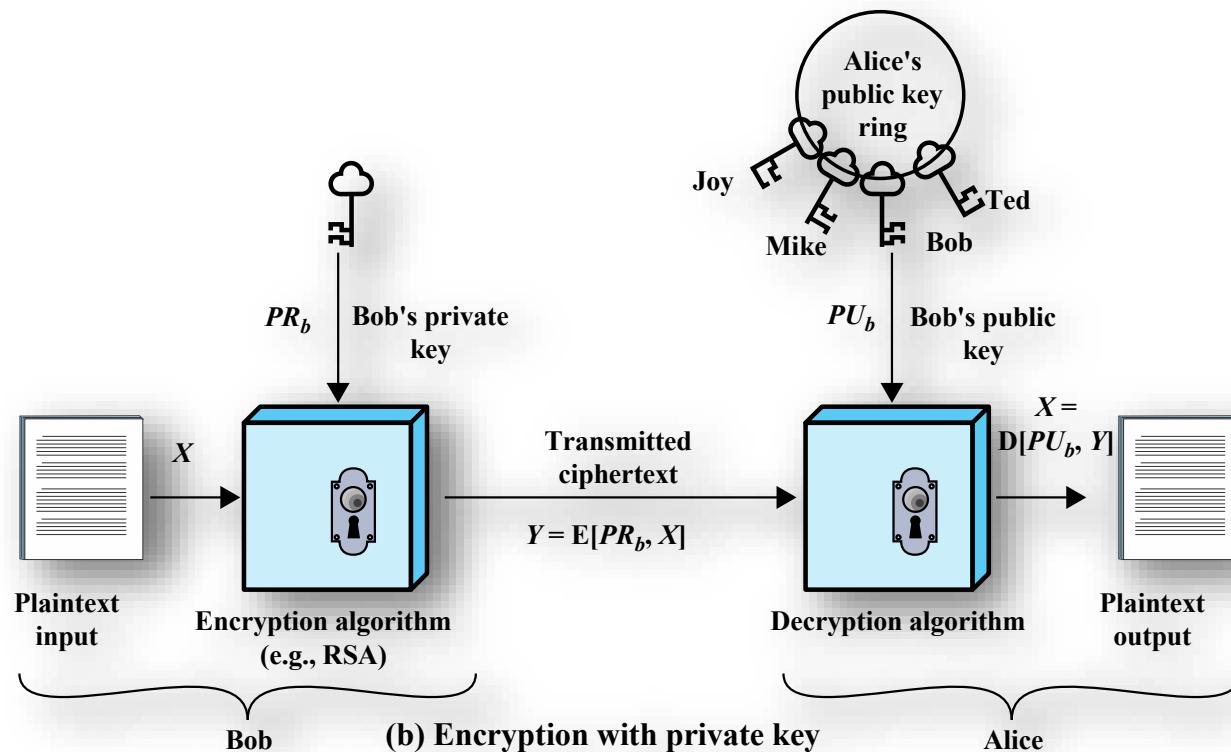


Figure 2.6 Public-Key Cryptography

# رمزنگاری کلید عمومی

- ♦ کلید های رمزگذاری و رمزگشایی متفاوت اما مرتبط هستند.
- ♦ رسیدن به کلید رمزگشایی از کلید رمزگذاری از لحاظ محاسباتی ناممکن می باشد.
- ♦ رمزگذاری امری همگانی میباشد و اساساً نیازی به اشتراک گذاشتن اطلاعات محرمانه ندارد.
- ♦ رمزگشایی از طرف دیگر امری اختصاصی بوده و محرمانگی پیامها محفوظ میماند.

## نمادها و قراردادها

- ◆ کلید عمومی: کلید، مزگزاری (در کاربرد مهندسی)
  - این کلید را برای شخص **A** با **KU<sub>a</sub>** نشان میدهیم.
- ◆ کلید خصوصی: کلید، مزگشتایی (در کاربرد مهندسی)
  - این کلید را برای شخص **A** با **KR<sub>a</sub>** نشان میدهیم.

# نیازمندیهای رمزنگاری کلید عمومی

۱. از نظر محاسباتی برای طرف B تولید یک زوج کلید (کلید عمومی  $KU_b$  و کلید خصوصی  $KR_b$ ) آسان باشد.
۲. برای فرستنده تولید متن رمز آسان باشد:

$$C = E_{KUb}(M)$$

۳. برای گیرنده رمزگشایی متن با استفاده از کلید خصوصی آسان باشد:

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

## نیازمندیهای رمزگاری کلید عمومی

۴. از نظر محاسباتی تولید کلید خصوصی ( $KR_b$ ) با دانستن کلید عمومی ( $KU_b$ ) غیر ممکن باشد.
۵. بازیابی پیام  $M$  با دانستن  $KU_b$  و  $C$  غیر ممکن باشد.
۶. (ویژگی تقارنی) از هر یک از کلیدها میتوان برای رمز کردن استفاده کرد. در این صورت از کلید دیگر برای رمزگشایی استفاده میشود.

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

# مقایسه رمزنگاری مرسوم و رمزنگاری کلید عمومی-۱

رمزنگاری مرسوم (Conventional Cryptography)

- ♦ استفاده از یک کلید یکسان و مخفی برای رمزگذاری و رمزگشایی

معایب

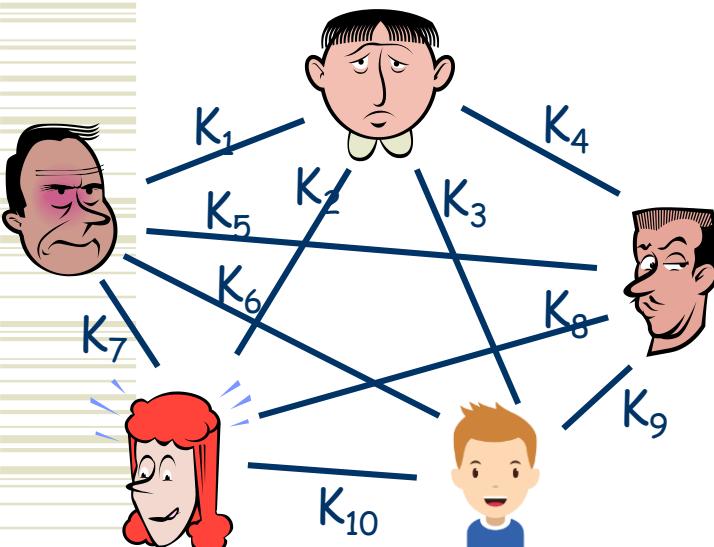
- ♦ مشکل مدیریت کلیدها

- نیاز به توافق بر روی کلید پیش از برقراری ارتباط
- برای ارتباط  $n$  نفر باهم به  $\frac{n(n-1)}{2}$  کلید احتیاج داریم

مزایا

- ♦ عدم پشتیبانی از امضاء الکترونیکی

با این وجود از الگوریتمهای رمزنگاری با کلید عمومی سریع‌تر است



# مقایسه رمزنگاری مرسوم و رمزنگاری کلید

## عمومی - ۲

رمزگذاری مرسوم

♦ برای امن بودن باید:

- کلید سری، مخفی نگه داشته شود.
- رسیدن به پیام واضح از روی متن رمز شده از نظر محاسباتی ناممکن باشد.
- اطلاع از الگوریتم و داشتن نمونه‌هایی از پیغام رمز شده برای تعیین کلید کافی نباشد.

# مقایسه رمزگذاری مرسوم و رمزگذاری کلید عمومی-۳

- ♦ مزایهای امنیتی (رمزگذاری با کلید عمومی)
  - تنها یکی از دو کلید باید مخفی بماند
  - با داشتن یک کلید، رسیدن به پیام واضح از روی متن رمز شده توسط همان کلید، از نظر محاسباتی نا ممکن باشد.
  - اطلاع از الگوریتم، داشتن یکی از کلیدها و نیز در اختیار داشتن نمونه پیغام‌های رمز شده برای تعیین کلید دوم کافی نباشد.

# جایگزینی یا تکمیل؟

از نظر کاربردی، رمزگذاری با کلید عمومی بیش از آن که جایگزینی برای رمزگذاری مرسوم باشد، نقش مکمل آنرا برای حل مشکلات توزیع کلید بازی می کند.

## دو تصور غلط ا



دو تصور اشتباه دیگر درباره الگوریتمهای کلید عمومی  
■ رمزنگاری با کلید عمومی امن تر است!

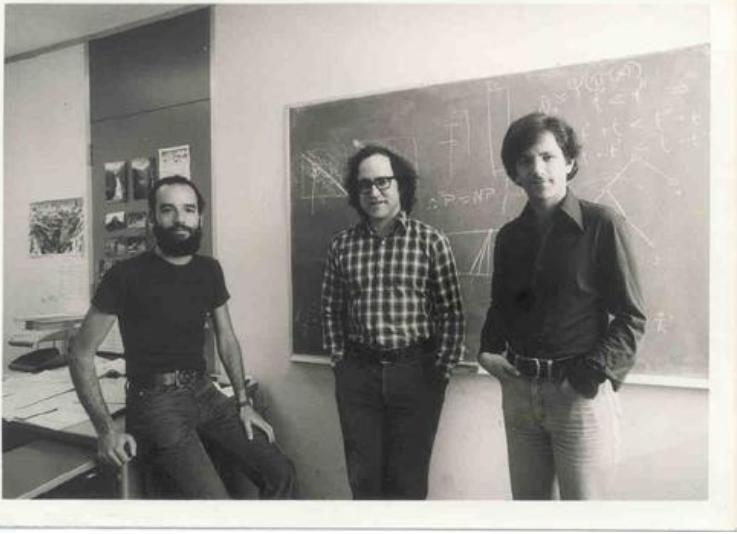
● در هر دو روش رمزنگاری امنیت به طول کلید وابسته است.

■ مسئله توزیع کلید در رمزنگاری با کلید عمومی برطرف شده  
است

● چگونه مطمئن شویم کلید عمومی لزوماً متعلق به شخص ادعائند  
است؟!

● پس توزیع کلید عمومی آسانتر است، ولی بدیهی نیست.

# الگوریتم رمزگذاری RSA



◆ کلیات

- توسط Rivest-Shamir-Adleman در سال ۱۹۷۷ در MIT ارائه شد
- مشهورترین و پرکاربردترین الگوریتم رمزگذاری کلید عمومی
- مبتنی بر توان رسانی پیمانه ایی
- استفاده از اعداد طبیعی خیلی بزرگ
- امنیت آن ناشی از دشوار بودن تجزیه اعداد بزرگ، که حاصل ضرب دو عامل اول بزرگ هستند، می باشد.
- مستندات مربوط به آن تحت عنوان PKCS استاندارد شده است.

# نمادگذاری RSA

- ♦  $N$ : پیمانه محاسبات
- ♦  $e$ : نمای رمزگذاری
- ♦  $d$ : نمای رمزگشایی
- ♦  $M$ : پیام ، عدد صحیح متعلق به  $Z_N^*$
- ♦ تابع  $RSA$ :  
$$x \rightarrow x^e \text{ mod } N$$
- ♦ تابع معکوس:  
$$x \rightarrow x^d \text{ mod } N$$
- ♦ دریچه تابع همان  $d$  میباشد.

# RSA تولید کلید

$$N = p \times q$$

$$\varphi(N) = (p-1) \times (q-1)$$

$$\gcd(\varphi(N), e) = 1$$

$$d \times e \equiv 1 \pmod{\varphi(N)}$$

$$C = M^e \pmod{N}$$

$$M = C^d \pmod{N} = (M^e)^d \pmod{N}$$

دو عدد اول  $p$  و  $q$  را انتخاب کن.

$N$  و  $\varphi(N)$  را حساب کن  
 $\varphi(N)$ : تعداد اعداد (کوچکتر از  $N$ ) که نسبت به  $N$  اول است.

$1 < e < \varphi(N)$   
 $\gcd(\varphi(N), e) = 1$   
کلید عمومی:  $d = e^{-1} \pmod{\varphi(N)}$

کلید خصوصی:  
 $\{e, N\}$ :  
 $\{d, N\}$ :

# پرا RSA کار میکند؟

به دلیل قضیه اول را:

- $a^{\phi(n)} \mod n = 1$  where  $\gcd(a, n) = 1$

برایم RSA،

- $n=p \cdot q$
- $\phi(n) = (p-1)(q-1)$
- carefully chose  $e$  &  $d$  to be inverses mod  $\phi(n)$
- hence  $e \cdot d = 1 + k \cdot \phi(n)$  for some  $k$

بنابراین:

$$\begin{aligned} C^d &= M^{e \cdot d} = M^{1+k \cdot \phi(n)} = M^1 \cdot (M^{\phi(n)})^k \\ &= M^1 \cdot (1)^k = M^1 = M \mod n \end{aligned}$$

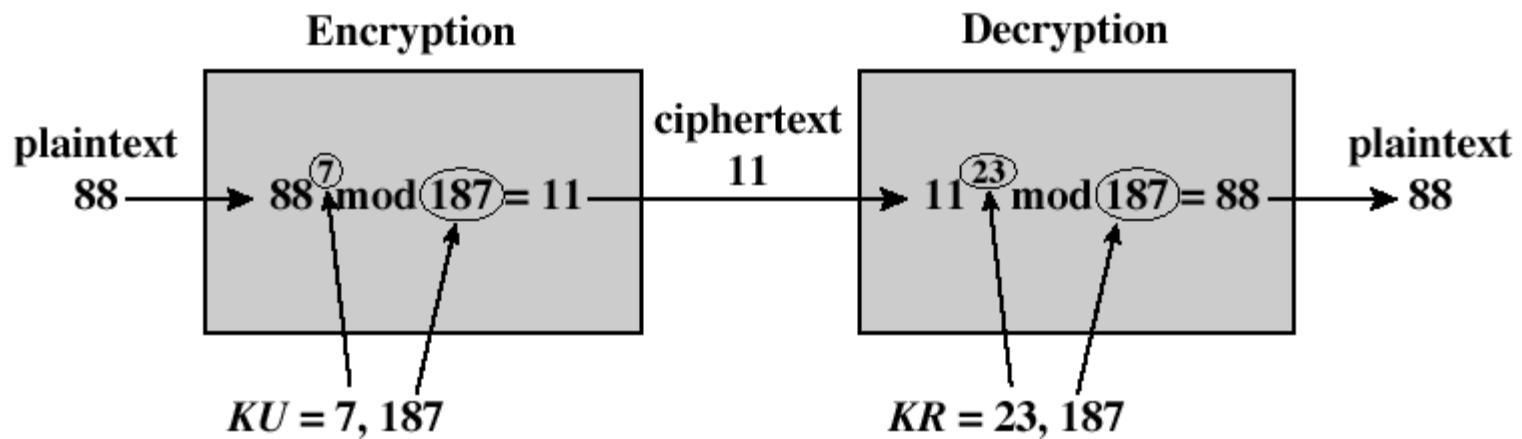
# فراردادها و پر تکل RSA

- ♦ هم فرستنده و هم گیرنده مقدار  $N$  را می‌دانند
- ♦ فرستنده مقدار  $e$  را می‌داند
  - کلید عمومی:  $(N, e)$
- ♦ تنها گیرنده مقدار  $d$  را می‌داند
  - کلید خصوصی:  $(N, d)$
- ♦ نیازمندیها:
  - محاسبه  $M^e$  و  $C^d$  آسان باشد
  - محاسبه  $d$  با دانستن کلید عمومی غیرممکن باشد

# مثال-RSA

$$p = 17, q = 11, n = p * q = 187$$

$$\Phi(n) = 16 * 10 = 160, \text{ pick } e = 7, d.e \equiv 1 \pmod{\Phi(n)} \rightarrow d = 23$$



# حملات ممکن بر RSA

- ♦ حمله آزمون جامع (Brute Force)
  - طول کلید با پیدایش هر نسل جدید از پردازنده ها افزایش می یابد، ضمن این که قدرت پردازشی هکرها زیاد می شود!
  - طول کلید معادل تعداد بیتهای پیمانه محاسبات ( $N$ ) می باشد.
- ♦ حملات ریاضی
  - تجزیه پیمانه  $N$  و در نتیجه محاسبه  $\varphi(N)$
  - محاسبه  $\varphi(N)$  به صورت مستقیم
  - محاسبه  $d$  بدون استفاده از  $\varphi(N)$
- ♦ حمله زمانی
  - زمان اجرای عملیات رمزگذاری یا واگنشایی رمز میتواند اطلاعاتی را در مورد کلید افشا کند.

# برقی نقاط ضعف RSA

♦ ویژگی RSA: مز شده ها صلبخرب دو پیام برابر ها صلبخرب، مز شده دو پیام است:

$$m_1^e m_2^e \equiv (m_1 m_2)^e \pmod{n}$$

♦ مواجهی که بخواهد  $c \equiv m^e \pmod{n}$  را، مزگشایی کند، اگر بتواند صاحب کلید خصوصی را واردار به، مزگشایی یک پیامی مانند  $r$  که  $c' \equiv c r^e \pmod{n}$  یک مقدار انتخاب شده توسط مواجه است بگذراند، میتواند  $m$  را بدست آورد.

$$c r^e \pmod{n} \equiv (mr)^e \pmod{n}$$

♦ مقدار  $mr$  را بدست آورده، معلوس  $r$  خرب میکند.

# حملات ریاضی RSA

- ♦ مقاله : Stanford در دانشگاه Dan Boneh
- Twenty Years of Attacks on the RSA Cryptosystem 1999

# راههای مقابله با حمله زمانی به RSA

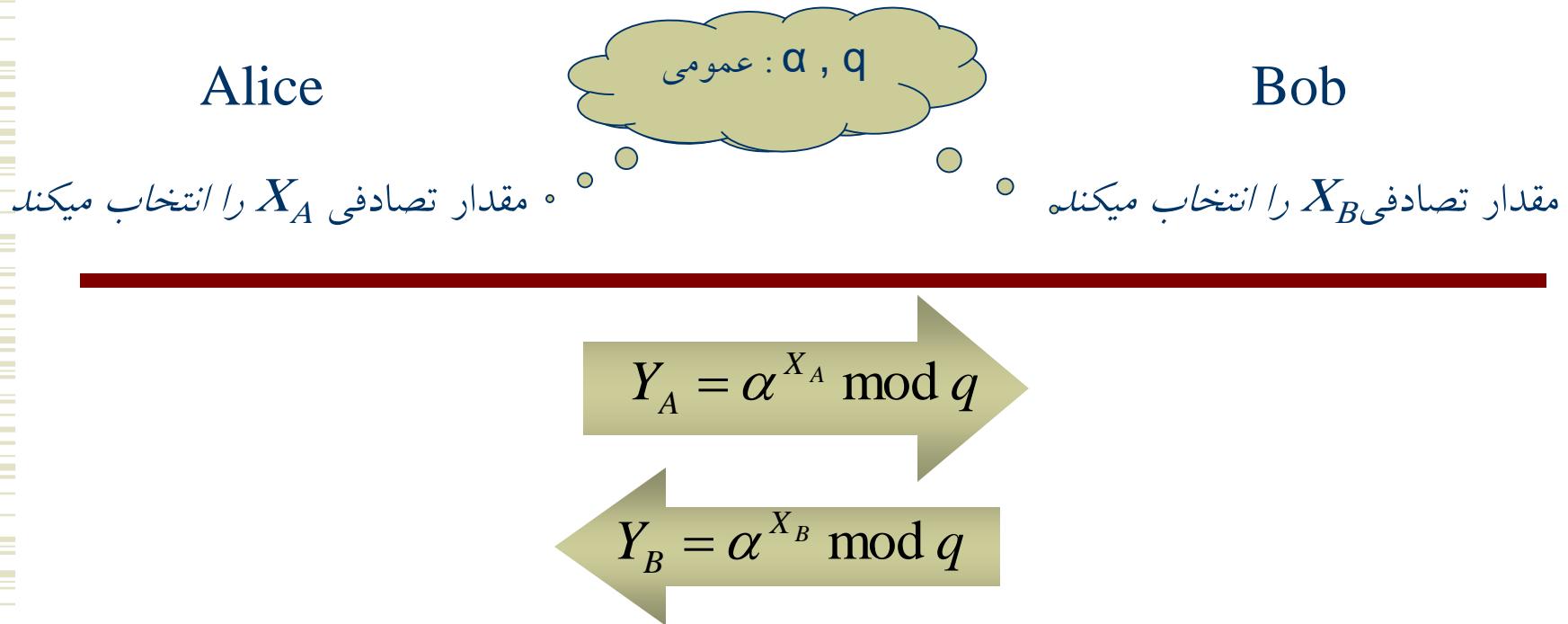
- ◆ استفاده از توان رساندن با زمان ثابت محاسباتی.
- تابع باید به ازای همه ورودیها زمان ثابتی به طول بیانجامد
- ◆ اضافه کردن تاخیرهای تصادفی
- ◆ قرار دادن اعمال اضافی و گمراه کننده در بین محاسبات
- ضرب کردن متن رمزشده در یک عدد تصادفی قبل از عملیات به توان رسانی
- تحلیلگر از ساختار بیتی متنی که به توان می‌رسد، مطلع نیست و این حمله زمانی را برای او غیرممکن می‌سازد



# الگوریتم *Diffie Hellman*

- ♦ برای تبادل کلید مورد استفاده قرار می‌گیرد
- ♦ طرفین بر روی مقادیر  $q$  و  $a$  توافق می‌کنند.
- ♦  $q$  یک عدد اول و  $a$  یک مولد برای این عدد می‌باشد.  
 $\{a \bmod q, a^2 \bmod q, \dots, a^{q-1} \bmod q\} = \{1, \dots, q-1\}$
- ♦ امنیت روش مبتنی بر مشکل بودن لگاریتم گستته است.

# الگوریتم Diffie - Hellman



$$K_{AB} = (Y_B)^{X_A} \text{ mod } q$$

$$K_{AB} = (Y_A)^{X_B} \text{ mod } q$$

$\alpha^{(X_A \times X_B)} \text{ mod } q$  کلید مشترک عبارت است از

# *Diffie-Hellman Example*

- ◆ users Alice & Bob who wish to swap keys:
- ◆ agree on prime  $q=353$  and  $a=3$
- ◆ select random secret keys:
  - A chooses  $x_A=97$ , B chooses  $x_B=233$
- ◆ compute respective public keys:
  - $y_A = 3^{97} \text{ mod } 353 = 40 \text{ (Alice)}$
  - $y_B = 3^{233} \text{ mod } 353 = 248 \text{ (Bob)}$
- ◆ compute shared session key as:
  - $K_{AB} = y_B^{x_A} \text{ mod } 353 = 248^{97} \text{ mod } 353 = 160 \text{ (Alice)}$
  - $K_{AB} = y_A^{x_B} \text{ mod } 353 = 40^{233} \text{ mod } 353 = 160 \text{ (Bob)}$

# *Key Exchange Protocols*

- ◆ users could create random private/public D-H keys each time they communicate
- ◆ users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- ◆ both of these are vulnerable to a meet-in-the-Middle Attack
- ◆ authentication of the keys is needed

# **Man-in-the-Middle Attack**

1. Darth prepares by creating two private / public keys
  2. Alice transmits her public key to Bob
  3. Darth intercepts this and transmits his first public key to Bob.  
Darth also calculates a shared key with Alice
  4. Bob receives the public key and calculates the shared key  
(with Darth instead of Alice)
  5. Bob transmits his public key to Alice
  6. Darth intercepts this and transmits his second public key to Alice. Darth calculates a shared key with Bob
  7. Alice receives the key and calculates the shared key (with Darth instead of Bob)
- Darth can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob

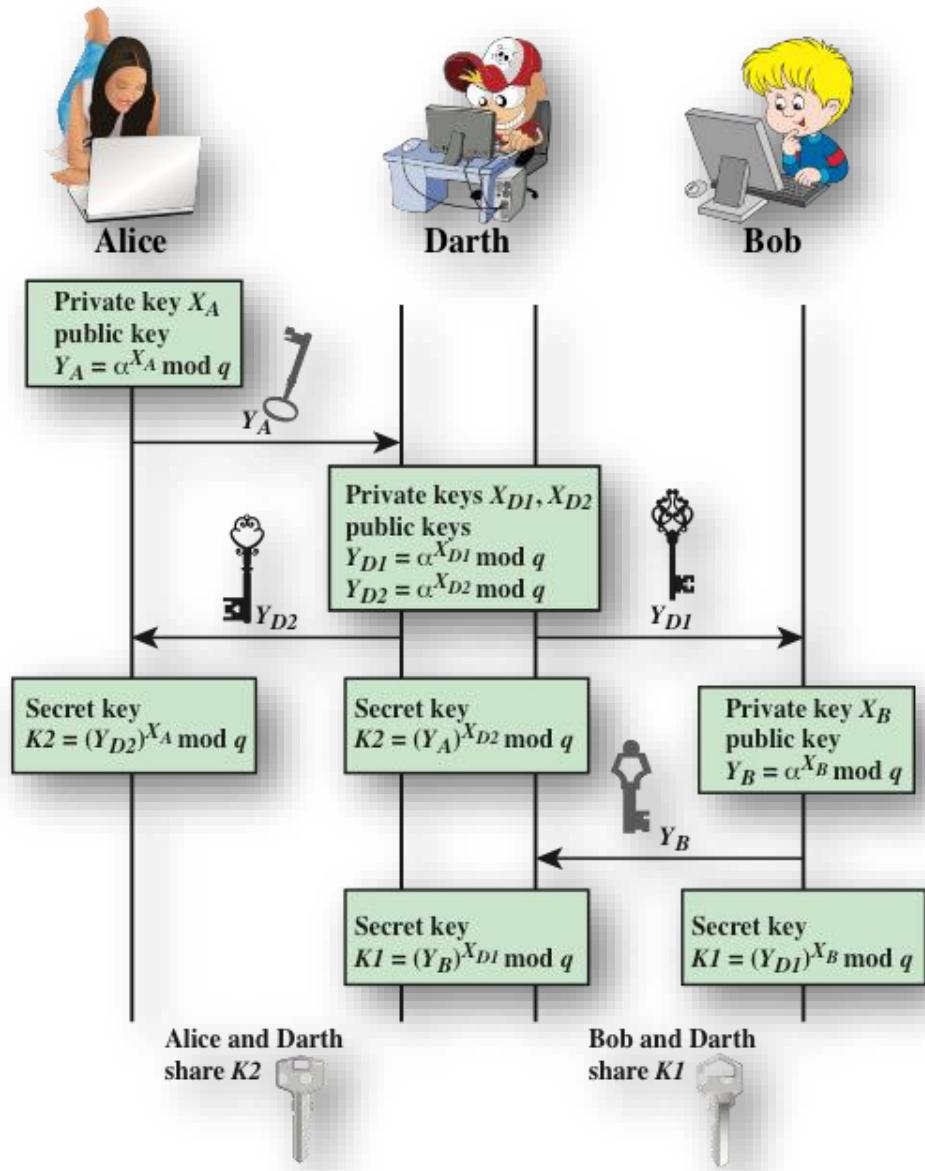


Figure 3.13 Man-in-the-Middle Attack  
<http://ceit.aut.ac.ir/~shahriari>

# *Elliptic-curve cryptology (ECC)*

- ◆ Technique is based on the use of a mathematical construct known as the elliptic curve
- ◆ Principal attraction of ECC compared to RSA is that it appears to offer equal security for a far smaller bit size, thereby reducing processing overhead
- ◆ The confidence level in ECC is not yet as high as that in RSA

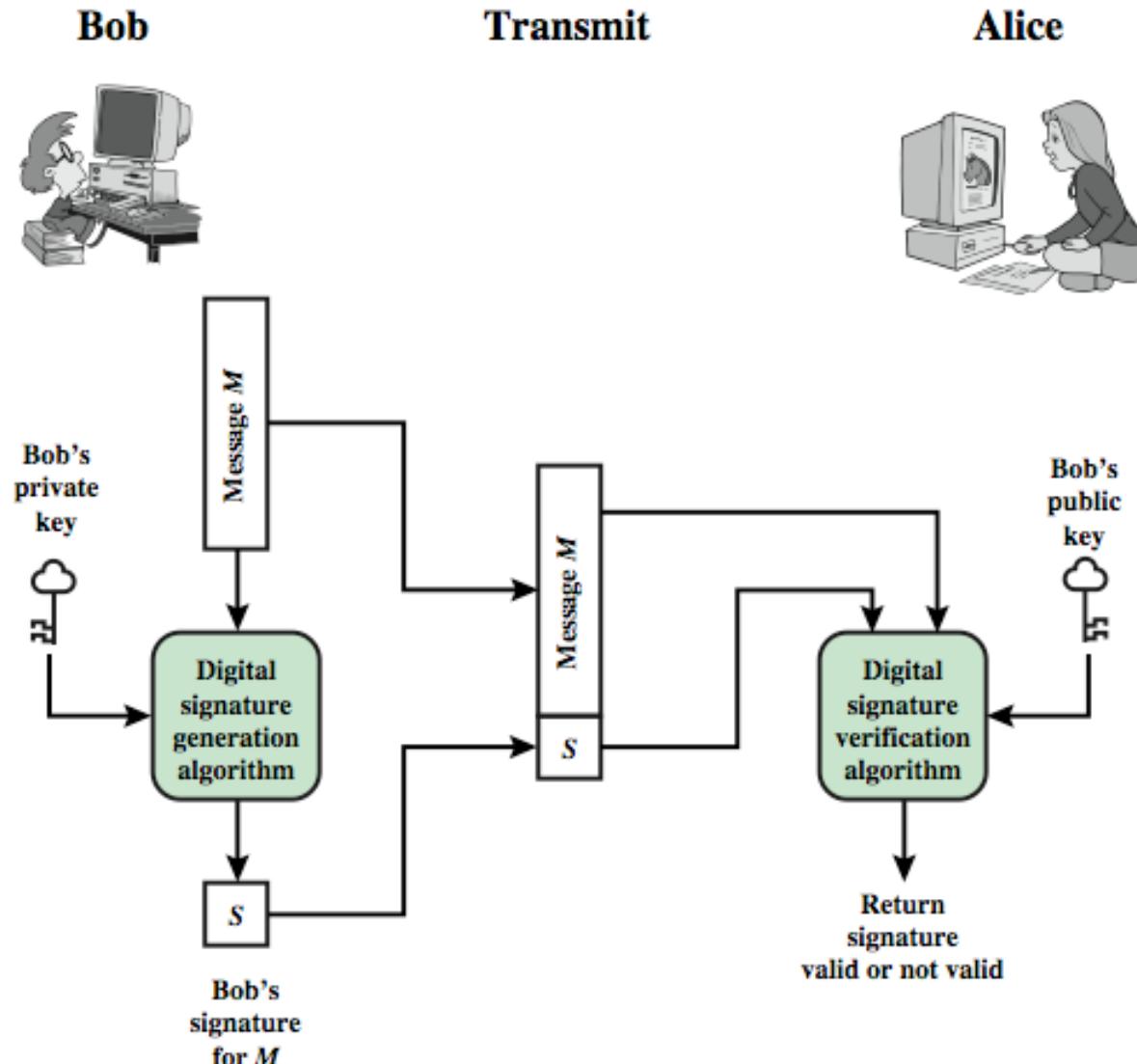
# امنیت دیجیتال

## *DIGITAL SIGNATURE*

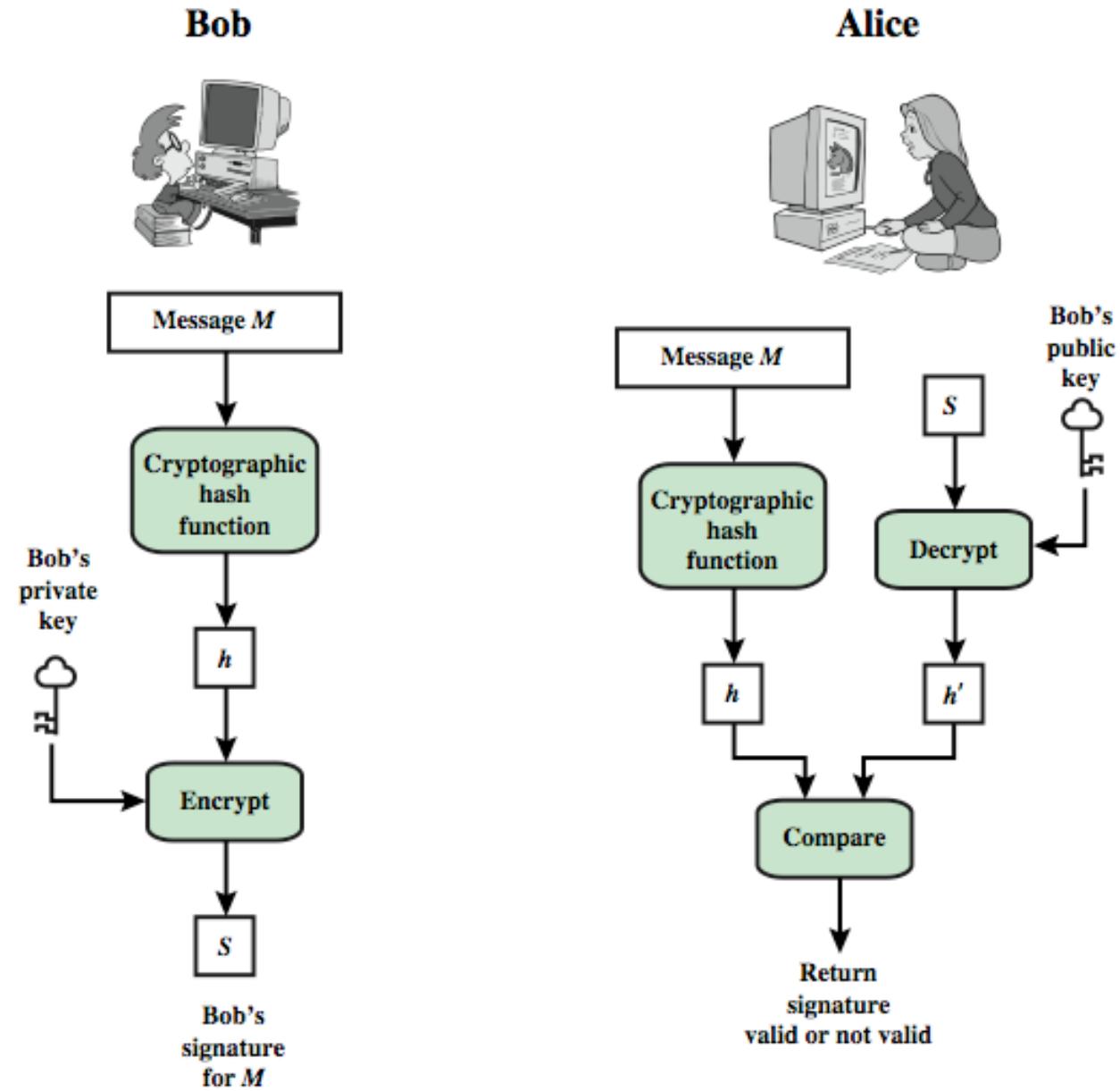
# *Digital Signatures*

- ◆ have looked at message authentication
  - but does not address issues of lack of trust
- ◆ digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes
- ◆ hence include authentication function with additional capabilities

# Digital Signature Model



# Digital Signature Model



Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic Curve	Yes	Yes	Yes