

اصرار اصلت پیام و رمزگاری کلید عمومی

(انشگاه صنعتی امیرکبیر

(انشگاه مهندسی کامپیوتر و فناوری اطلاعات

توسط: حمید، خا شهریاری

فهرست

- ♦ روش‌های احراز اصالت پیام
- ♦ HMAC و Secure Hash Function
- ♦ اصول رمزنگاری کلید عمومی
- ♦ الگوریتم‌های رمزنگاری کلید عمومی
- ♦ امضای رقمی
- ♦ مدیریت کلید

امراز اصالت

- ♦ نیازمندیها: باید قادر به بررسی موارد زیر باشد:
 - پیام از سوی منبع یا نویسنده ادعا شده می آید
 - محتوا تغییر نکرده است
 - در برخی موارد، آیا در زمان معین یا در ترتیب خاصی آمده است
- ♦ حفاظت در برابر حملات فعال
 - (falsification of data and transactions) ▪

رهیافت‌های احراز اصالت

- ♦ احراز اصالت با رمزنگاری معمولی
 - تنها فرستنده و گیرنده باید کلید را داشته باشند.
- ♦ احراز اصالت بدون رمز کردن پیام
 - یک برچسب احراز اصالت تولید و به هر پیام پیوست می شود.
 - کد احراز اصالت پیام

(Message Authentication Code)

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

$$MAC = F(K_{AB}, M)$$
 •

خواست می شود که تنها **A** و **B** از کلید اطلاع دارند.

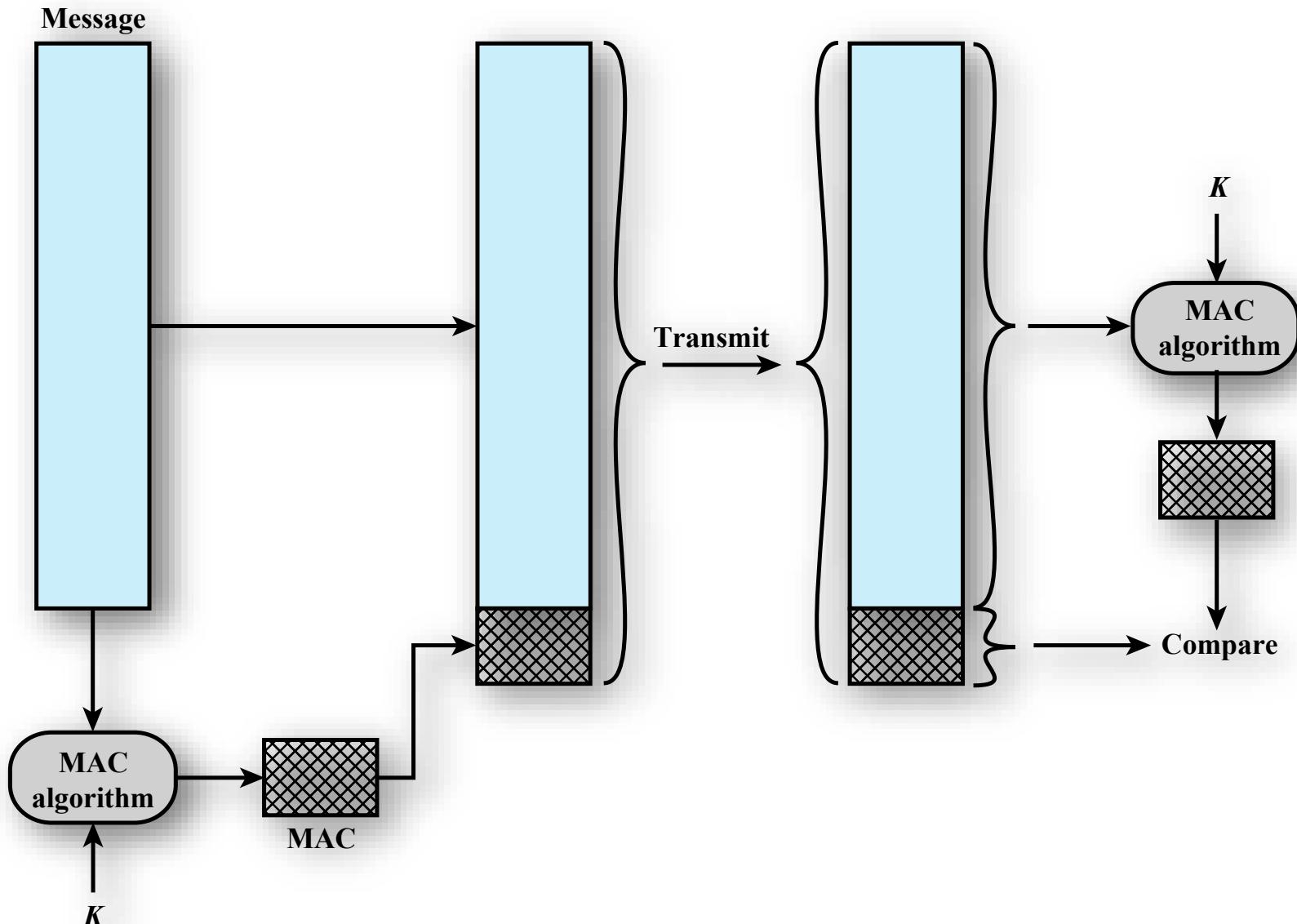


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

امراز اصالت پیام

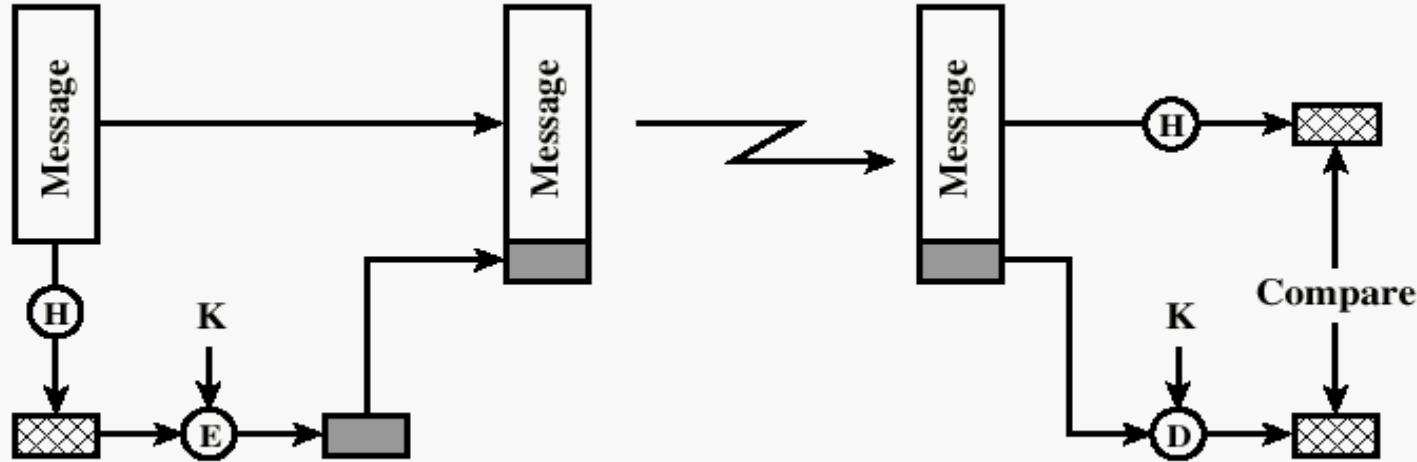
- MAC نیازمندیهای زیر را برأورده می کند:
 - پیام از سوی منبع یا نویسنده ادعا شده می آید،
 - محتوا تغییر نکرده است،
 - در برخی موارد، پیام در زمان معین یا در ترتیب خاصی آمده است.

اصرار اصولت پیام

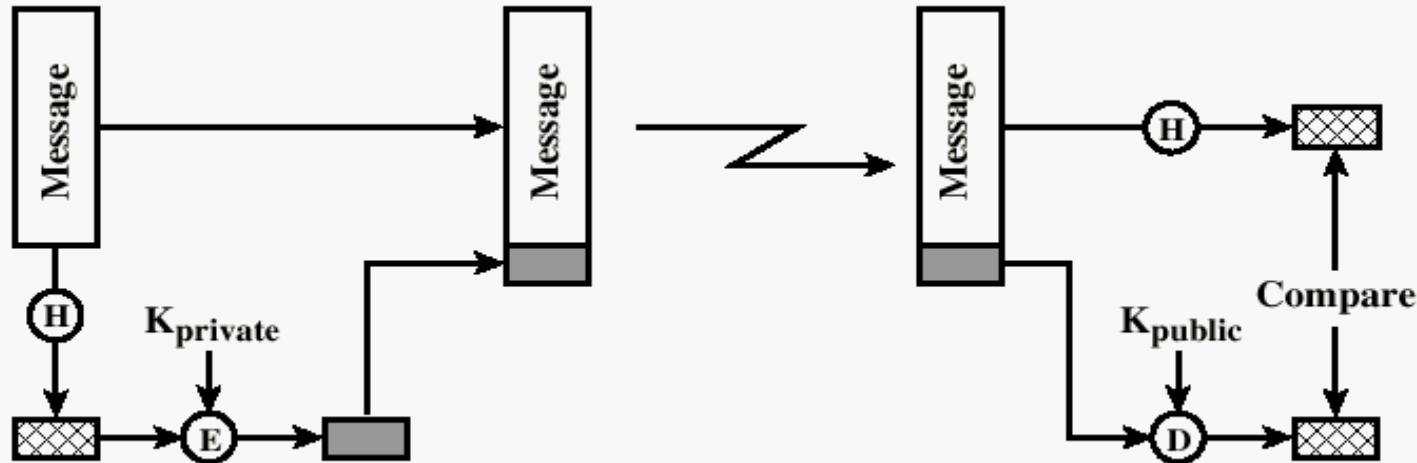
♦ چرا با رمزنگاری معمولی انجام ندهیم؟
■ موارد زیر را در نظر بگیرید:

- 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 اصولت برنامه ها برای جلوگیری از تغییر غیر مجاز برنامه (مثلا توسط ویروسها)

روش‌های مختلف اثبات اصالت پیام



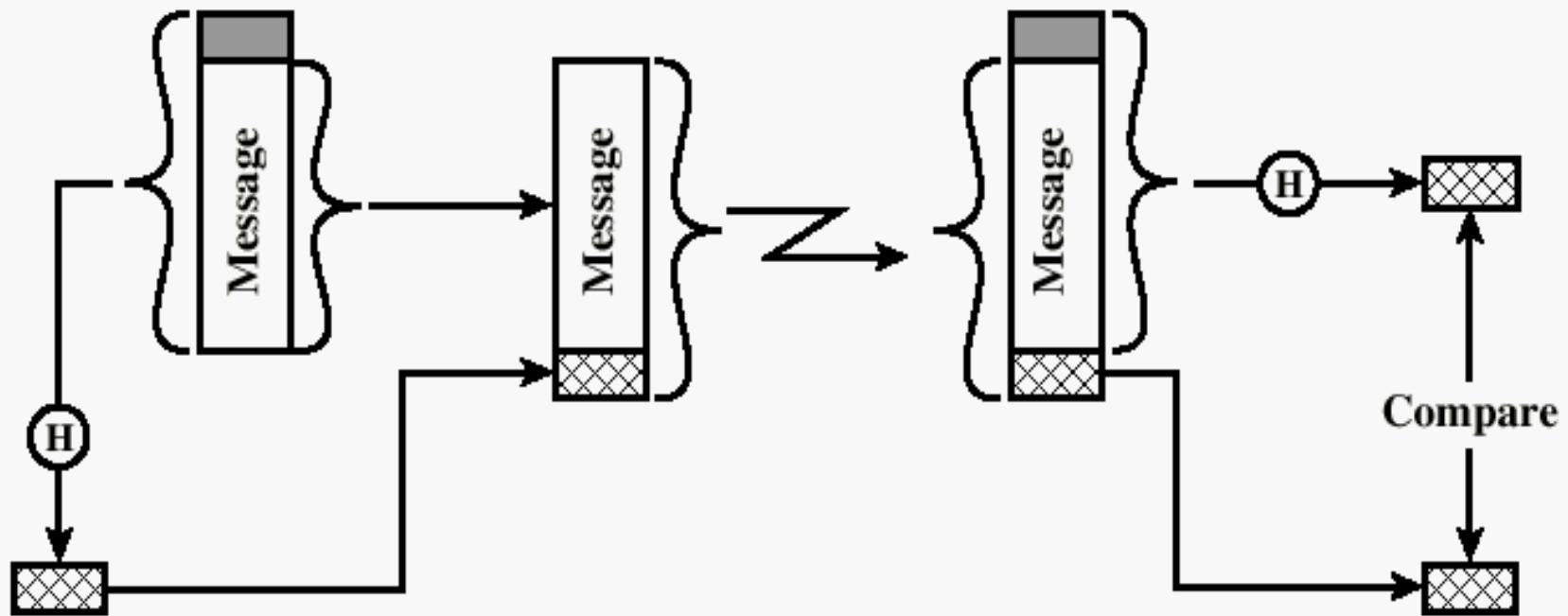
(a) Using conventional encryption



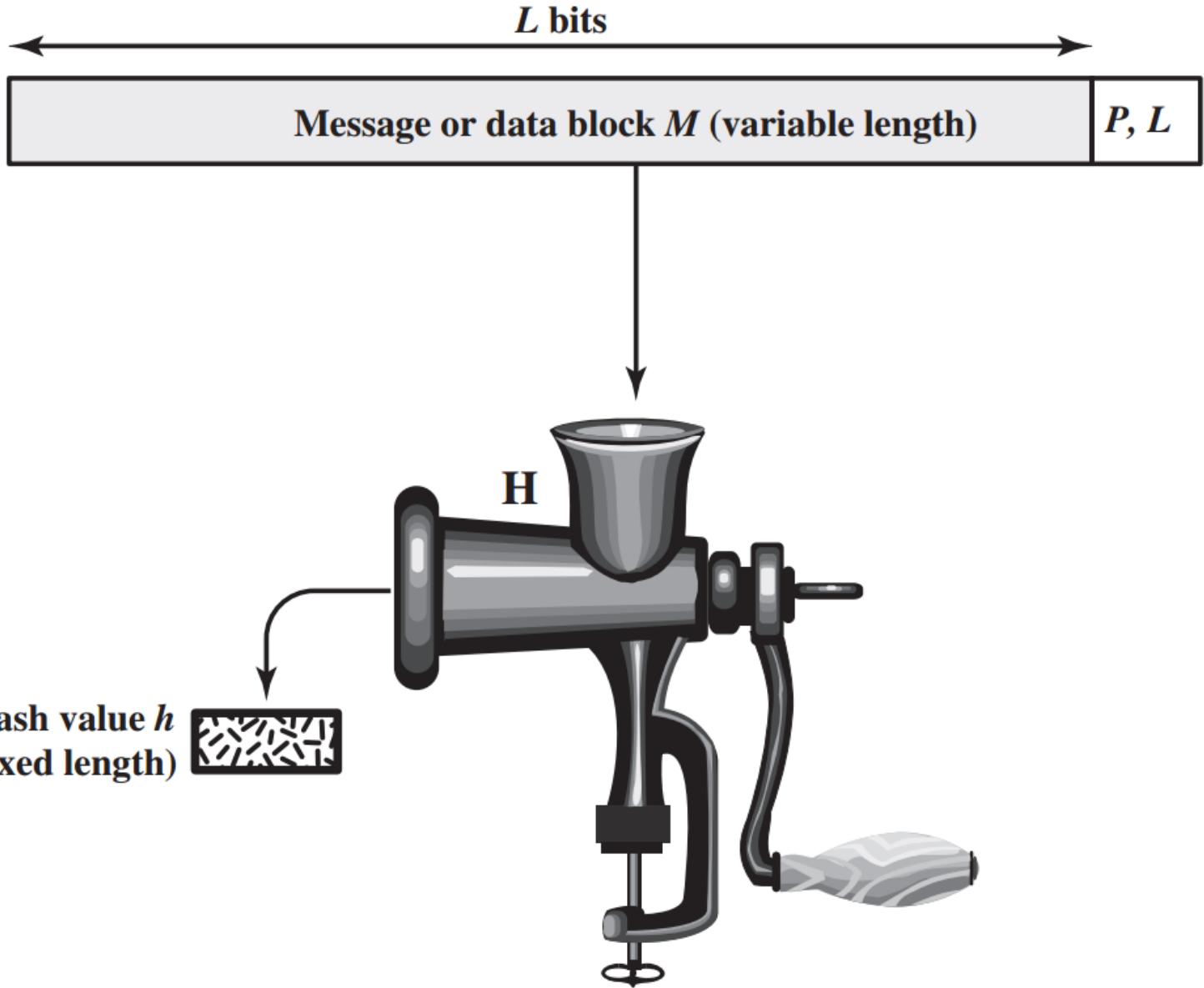
(b) Using public-key encryption

روش‌های مختلف اثراز اصالت پیام

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



(c) Using secret value

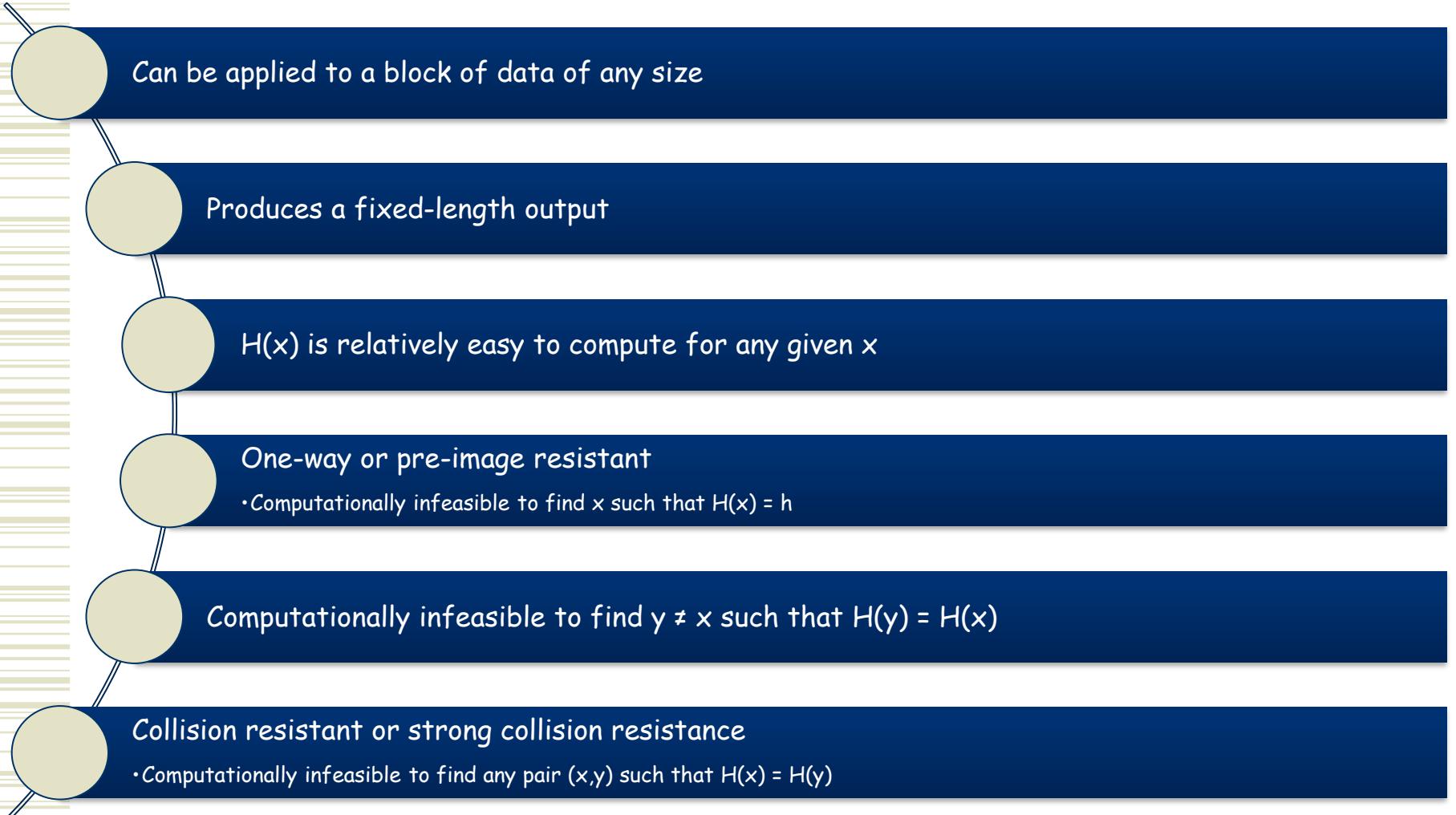


P, L = padding plus length field

تابع 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:
 - .1 قابلیت اعمال به هر اندازه از داده ها
 - .2 خروجی طول ثابتی داشته باشد
 - .3 محاسبه $H(x)$ برای هر مقدار x ساده باشد
 - .4 برای هر بلوک x ، بدست آوردن x از $H(x)$ غیر ممکن باشد
 - .5 برای هر بلوک x ، بدست آوردن بلوک دیگر مانند y که $H(x)=H(y)$ باشد غیر ممکن باشد.
 - .6 پیدا کردن زوج هایی مانند (x,y) که $H(x)=H(y)$ شود غیر ممکن باشد.

Hash Function Requirements



یک تابع درهم سازی ساده

- ❖ فرض کنید که داده ورودی (مانند فایل) دنباله ای از بلوک های n بیتی باشد.
 - ❖ C_i : بیت i ام کد درهم سازی
 - ❖ m : تعداد بلوکها
 - ❖ b_{ij} : بیت i ام 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 ₁₁	b ₂₁			b _{<i>n</i>1}
block 2	b ₁₂	b ₂₂			b _{<i>n</i>2}
	•	•	•	•	•
	•	•	•	•	•
	•	•	•	•	•
block <i>m</i>	b _{1<i>m</i>}	b _{2<i>m</i>}			b _{<i>n</i><i>m</i>}
hash code	C ₁	C ₂			C _{<i>n</i>}

Figure 3.3 Simple Hash Function Using Bitwise XOR

Secure Hash Algorithm (SHA)

- ♦ توسط موسسه NIST در سال ۱۹۹۳ اعلان 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)

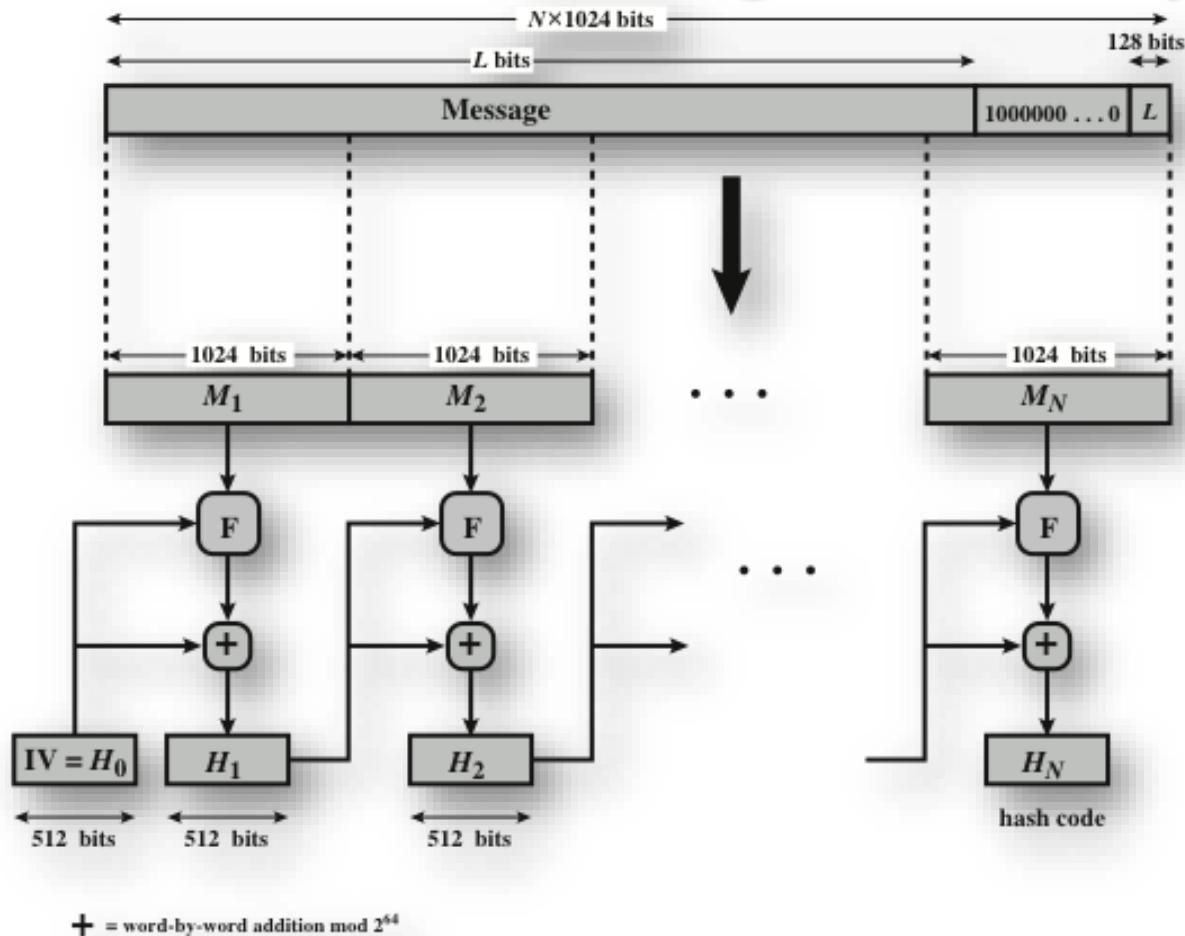


Figure 3.4 Message Digest Generation Using SHA-512
<http://www.aut.ac.ir/shahriari>

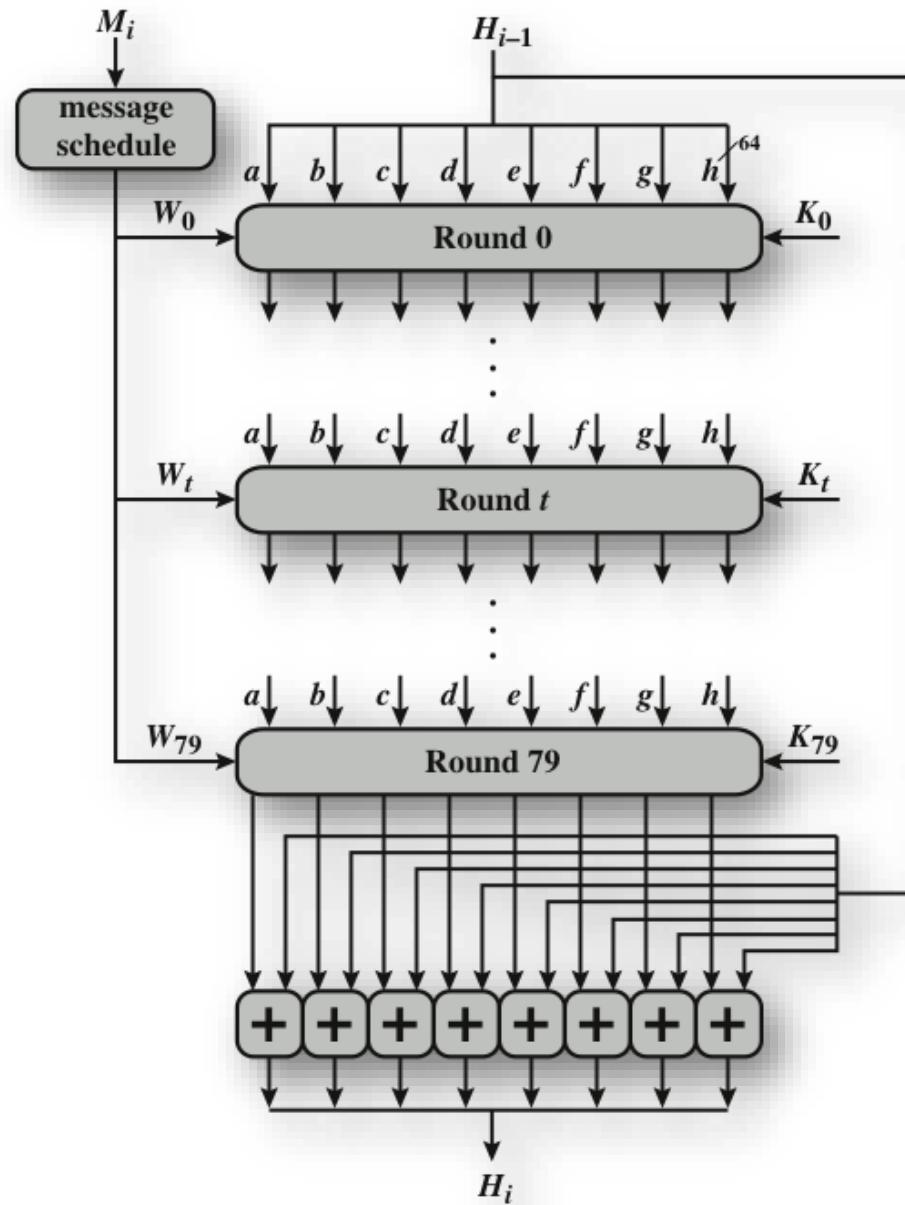


Figure 3.5 SHA-512 Processing of a Single 1024-Bit Block

SHA-3

- ♦ NIST در سال 2007 مسابقه ای را جهت انتخاب الگوریتم جدید SHA-3 اعلام کرد.
- ♦ در سال 2012 الگوریتم Keccak انتخاب شد و به عنوان SHA-3 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 پسکایده پیام

MD5 ◆

- توسط (RFC 1321) Rivest
- ورودی به طول دلخواه و خروجی 128 بیت
- در سال 2004 شکسته شد!

RIPMED-160

- ♦ طی پروژه RIPE در اتحادیه اروپا در سال 1997 ارائه شد.
- ♦ ورودی بلوکهای 512 بیتی
- ♦ در ابتدا 128 بیتی بود که با یافتن حملاتی به برخی دورهای آن به 160 بیت ارتقا یافت.

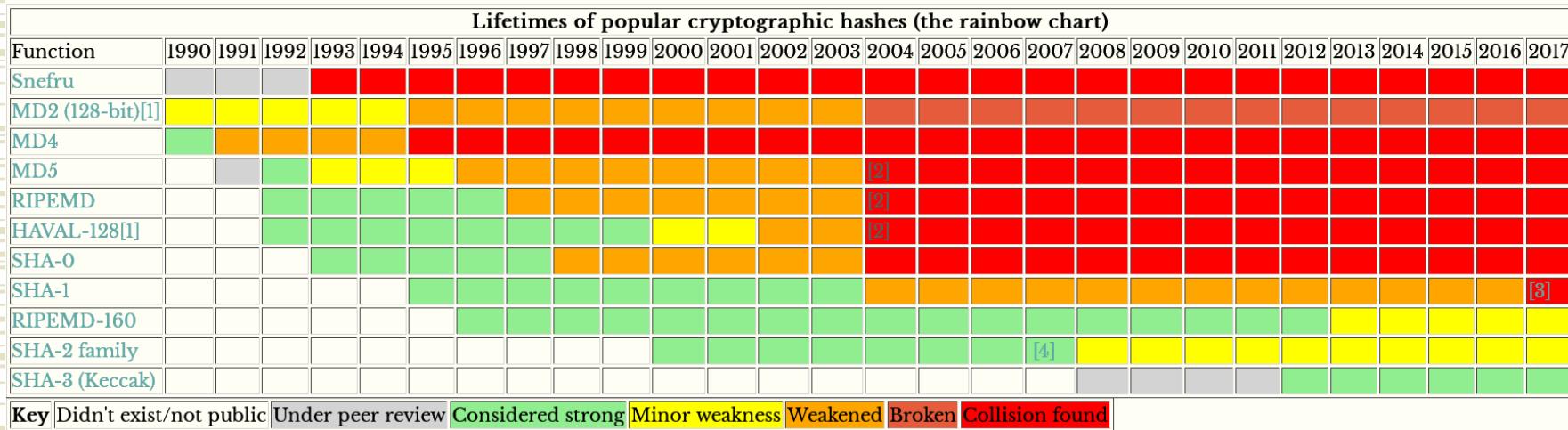
rules

	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	∞	∞

SHA مفاسی کانوارده

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

مقایسه چرخه میان توابع درهم سازی



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

SHA1 vulnerability

♦ گوگل در سال 2017 توانست با منابع زیر آسیب پذیری SHA1 را نشان دهد: پیدا کند:

- 6500 CPU years + 110 GPU years:
 - Nine quintillion (9,223,372,036,854,775,808) SHA1 computations in total
 - 6,500 years of CPU computation to complete the attack first phase
 - 110 years of GPU computation to complete the second phase
- Ref: <https://shattered.io>

HMAC

- ♦ استفاده از یک روش MAC مبتنی بر یک کد درهم سازی مانند SHA-1
- ♦ انگیزه ها:
 - توابع درهم سازی از توابع رمزنگاری مانند DES سریعتر اجرا می شوند.
 - توابع کتابخانه ای برای توابع درهم 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 درهم سازی
 - جایگزینی آسان توابع درهم سازی
 - حفظ کارآیی اولیه توابع درهم سازی
 - استفاده ساده از کلیدها
 - امکان تحلیل آسان

HMAC جیکوب

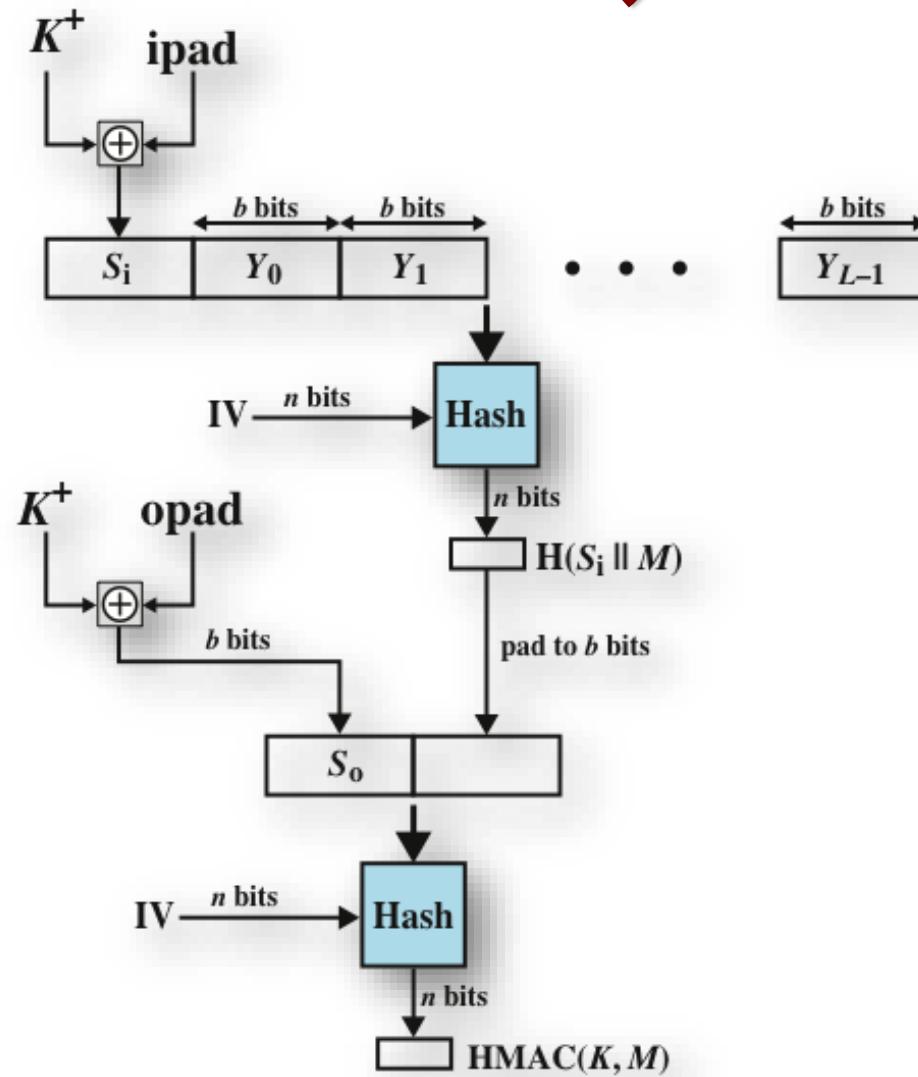


Figure 3.6 HMAC Structure
<http://www.aut.ac.ir/shahriari>

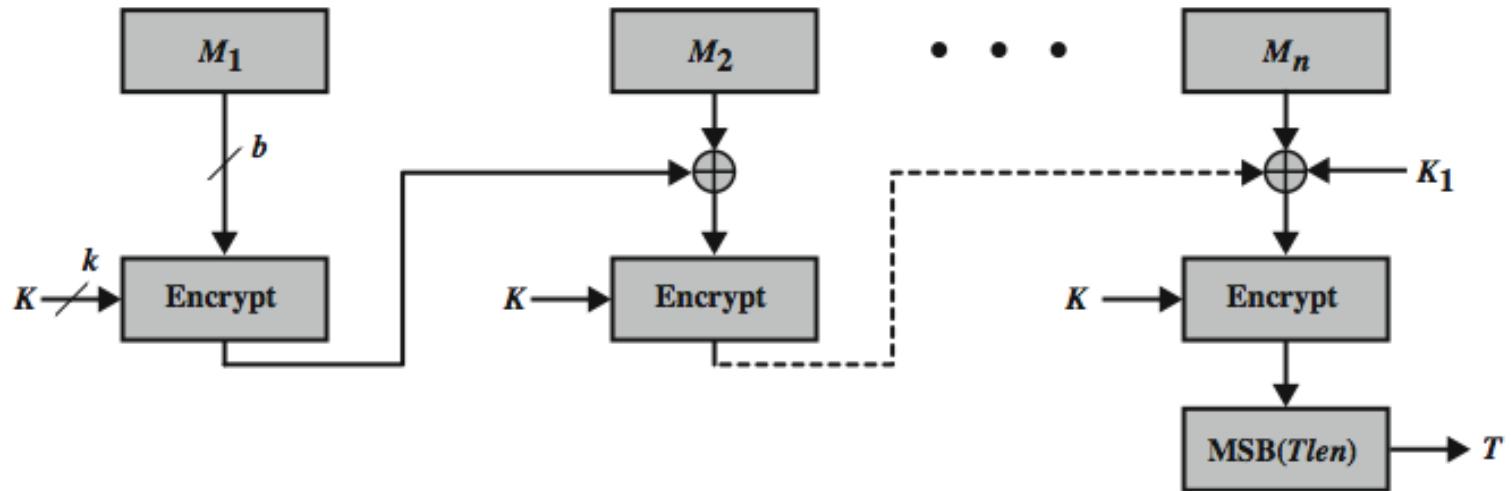
HMAC Security

- ◆ 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

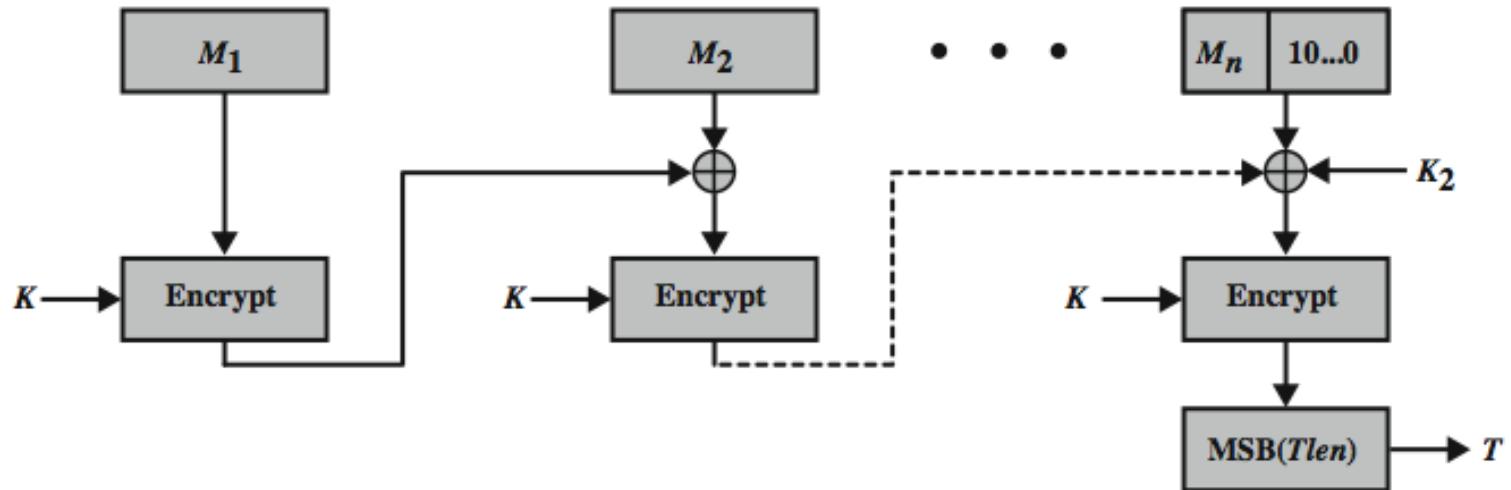
CMAC

- ◆ 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



(a) Message length is integer multiple of block size



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

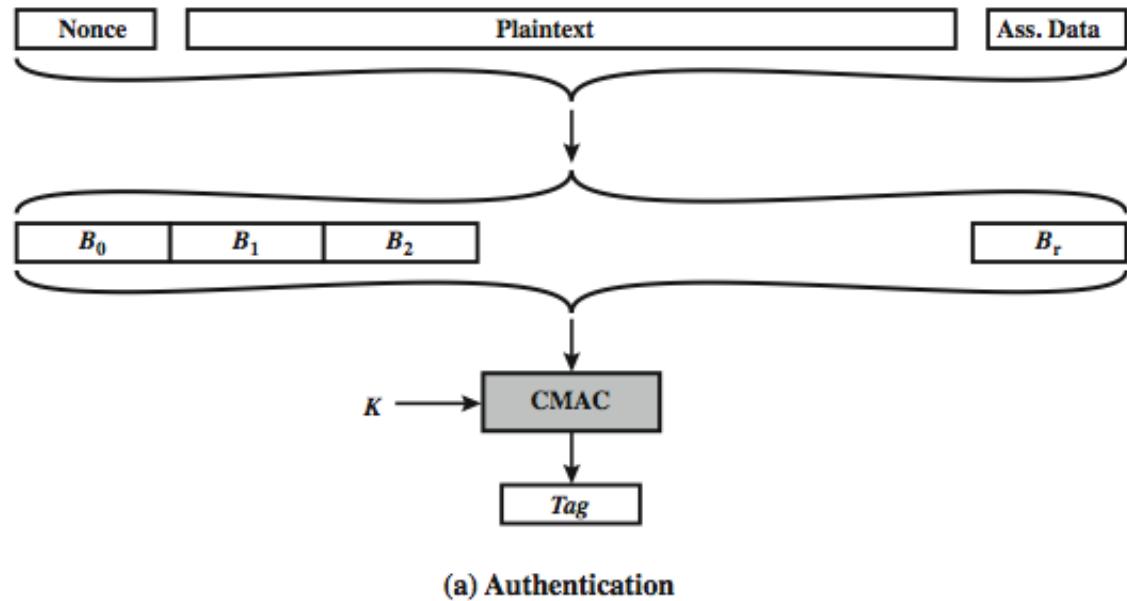
Authenticated Encryption

- 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

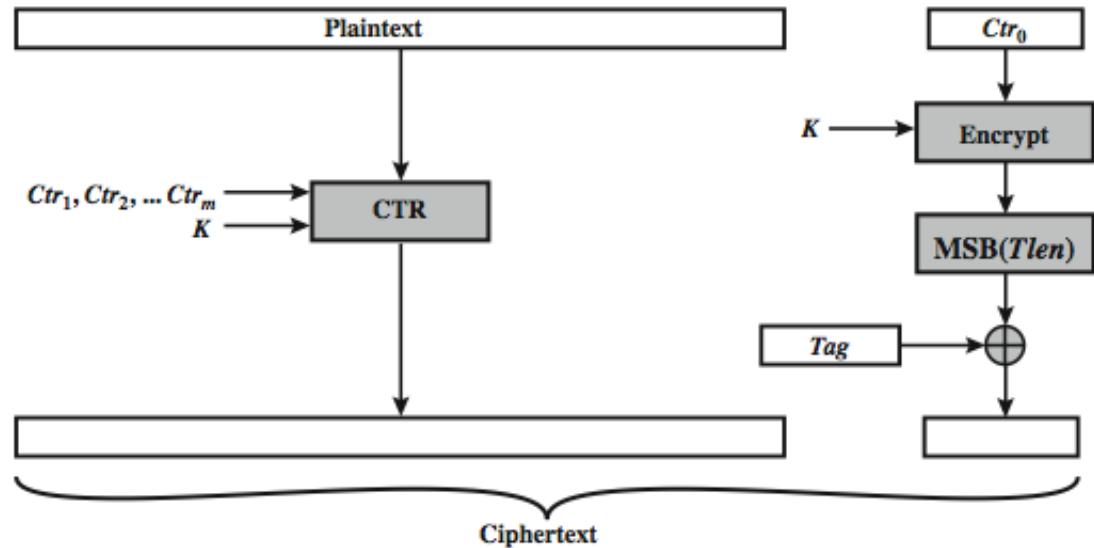
Counter with Cipher Block Chaining-Message Authentication Code (CCM)

- ◆ 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

هزنگاری نامتقاض (کلید عمومی)

مبانی رمزنگاری کلید عمومی

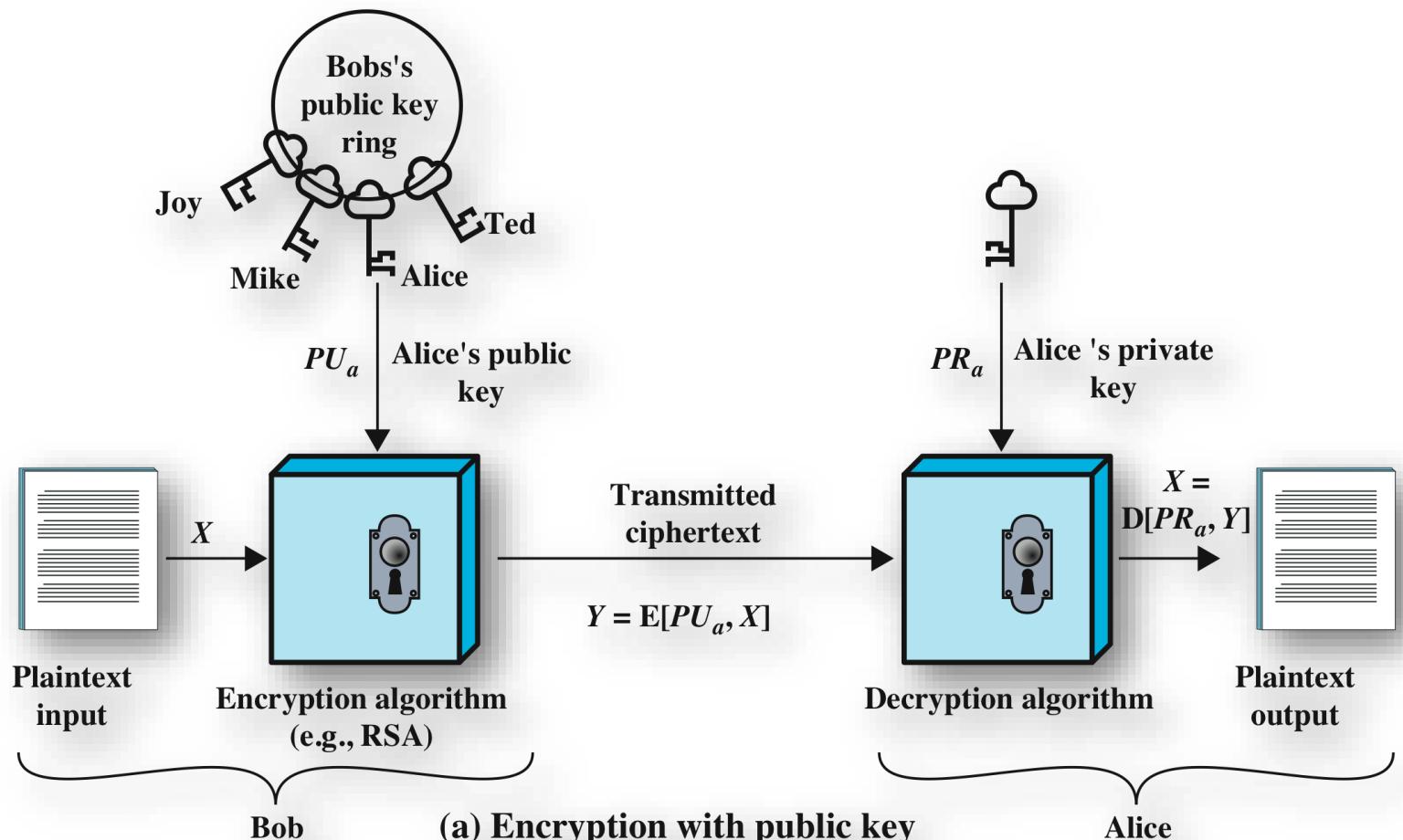


- ♦ رمزنگاری کلید عمومی اساساً با انگیزه رسیدن به دو هدف طراحی شد:
 - حل مساله توزیع کلید در روشهای رمزنگاری متقارن
 - امضای دیجیتال
- ♦ دیفی و هلمن اولین راه حل را در 1976 ارایه دادند.

کاربردهای رمزنگاری کلید عمومی

- ♦ سه دسته کاربرد:
 - **محرمانگی:** فرستنده پیام را با کلید عمومی گیرنده رمز می کند.
 - **احراز اصالت یا امضای رقمی:** فرستنده 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

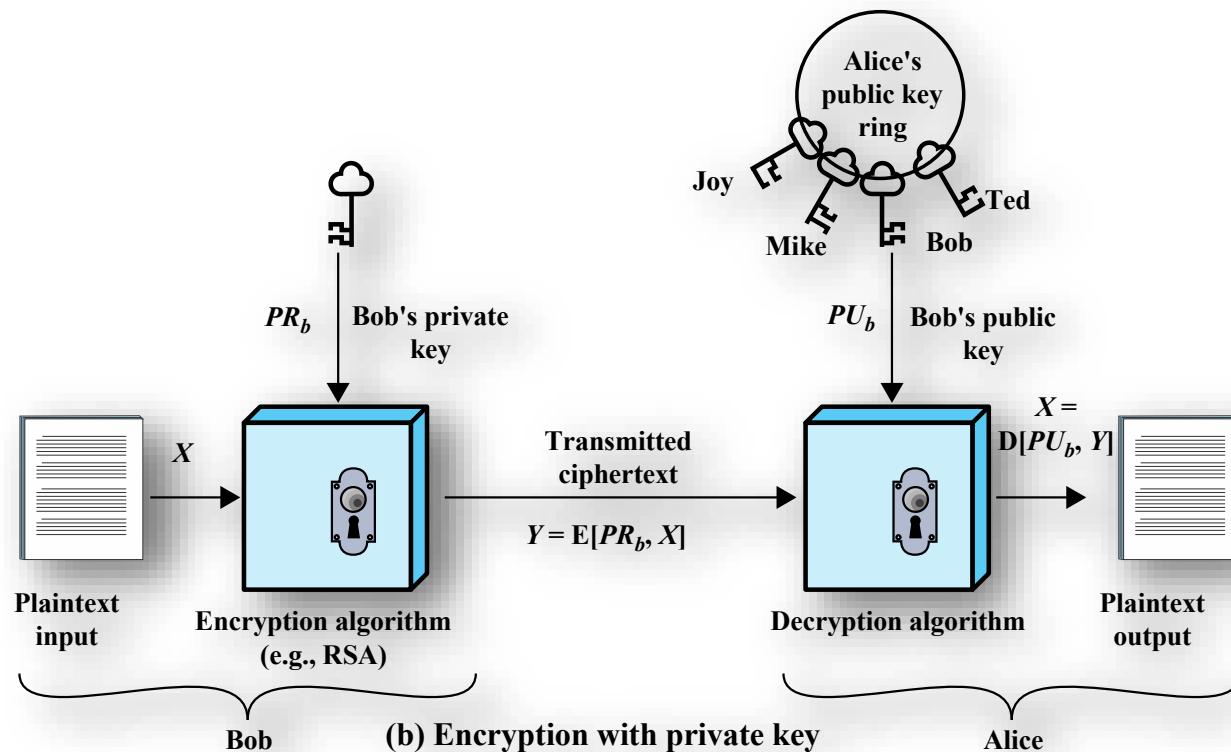


Figure 2.6 Public-Key Cryptography

رمزگاری کلید عمومی

- ♦ کلید های رمزگذاری و رمزگشایی متفاوت اما مرتبط هستند.
- ♦ رسیدن به کلید رمزگشایی از کلید رمزگذاری از لحاظ محاسباتی ناممکن می باشد.
- ♦ درکاربرد محترمانگی:
 - رمزگذاری امری همگانی می باشد و اساساً نیازی به اشتراک گذاشتن اطلاعات محترمانه ندارد.
 - رمزگشایی از طرف دیگر امری اختصاصی بوده و محترمانگی پیامها محفوظ می ماند.

نمادها و قراردادها

- ♦ کلید عمومی:
 - این کلید را برای شخص A با KU_A نشان می‌دهیم.

- ♦ کلید خصوصی:
 - این کلید را برای شخص A با KR_A نشان می‌دهیم.

نیازمندیهای رمزگاری کلید عمومی

1. از نظر محاسباتی برای طرف B تولید یک زوج کلید (کلید عمومی KU_b و کلید خصوصی KR_b) آسان باشد.
2. (در کاربرد محرمانگی) برای فرستنده تولید متن رمز آسان باشد:
$$C = E_{KU_b}(M)$$
3. (در کاربرد محرمانگی) برای گیرنده رمزگشایی متن با استفاده از کلید خصوصی آسان باشد:

$$M = D_{KR_b}(C) = D_{KR_b}[E_{KU_b}(M)]$$

نیازمندیهای رمزگاری کلید عمومی

4. از نظر محاسباتی تولید کلید خصوصی (KR_b) با دانستن کلید عمومی (KU_b) غیر ممکن باشد.
5. بازیابی پیام M که با کلید عمومی رمز شده، با دانستن فقط KU_b و C غیر ممکن باشد.
6. (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

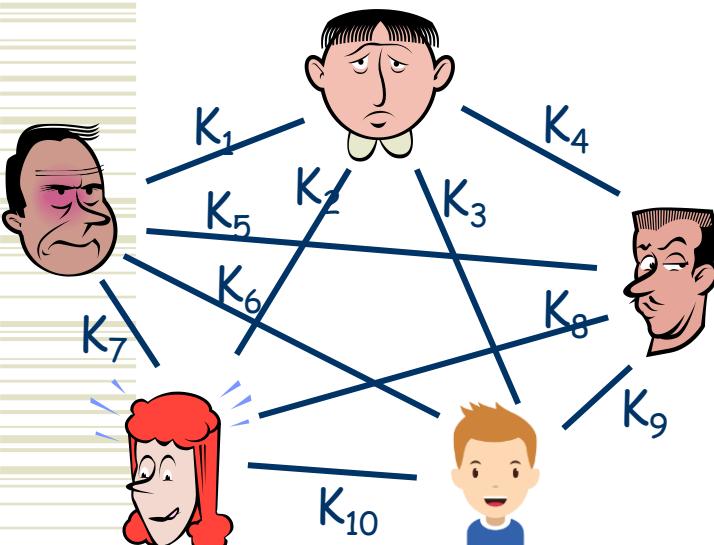
- ♦ مشکل مدیریت کلیدها

- 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 پیام رمز شده برای تعیین کلید کافی نباشد.

مقایسه رمزگذاری مرسوم و رمزگذاری کلید عمومی-۳

- ♦ مزایه 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 ناممکن باشد.
 - اطلاع از الگوریتم، داشتن یکی از کلیدها و نیز در اختیار داشتن نمونه پیام‌های 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 آن را برای حل مشکلات توزیع کلید بازی می‌کند.

دو تصور غلط ا



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

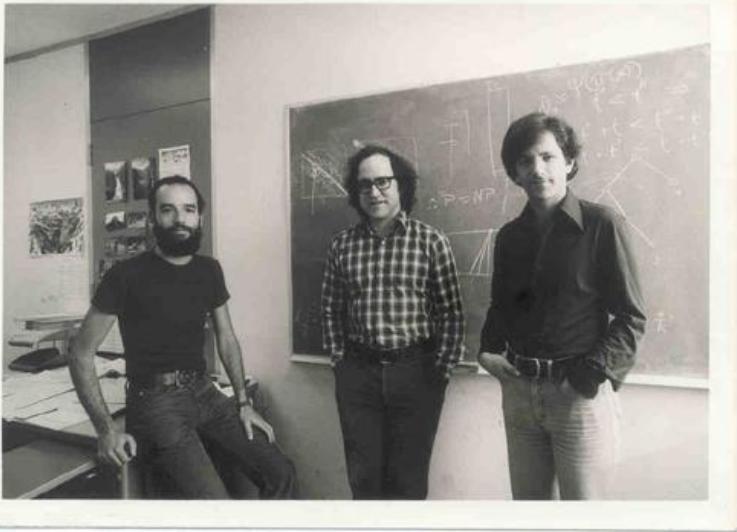
● در هر دو روش رمزنگاری امنیت به طول کلید وابسته است.

■ 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 پیمانه‌ای
- استفاده از اعداد طبیعی خیلی بزرگ
- امنیت آن ناشی از دشوار بودن تجزیه اعداد بزرگ، که حاصل ضرب دو عامل اول بزرگ هستند، می‌باشد.
- 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 $x \rightarrow x^e \bmod N$:RSA
- ♦ تابع معکوس: $x \rightarrow x^d \bmod N$
- ♦ دریچه تابع همان d میباشد.

تولید کلید RSA

$$N = p \times q$$

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

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

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

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

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

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.

$1 < e < \varphi(N)$ را انتخاب کن که
 $\gcd(\varphi(N), e) = 1$
 $d = e^{-1} \pmod{\varphi(N)}$ را حساب کن

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

چرا RSA کار می‌کند؟

- ♦ به دلیل قضیه اولر:
- اگر a و n نسبت به هم اول باشند یعنی $\text{gcd}(a, n) = 1$ آنگاه:
- $a^{\phi(n)} \mod n = 1$
- ♦ در RSA داریم:

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

بنابراین:

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

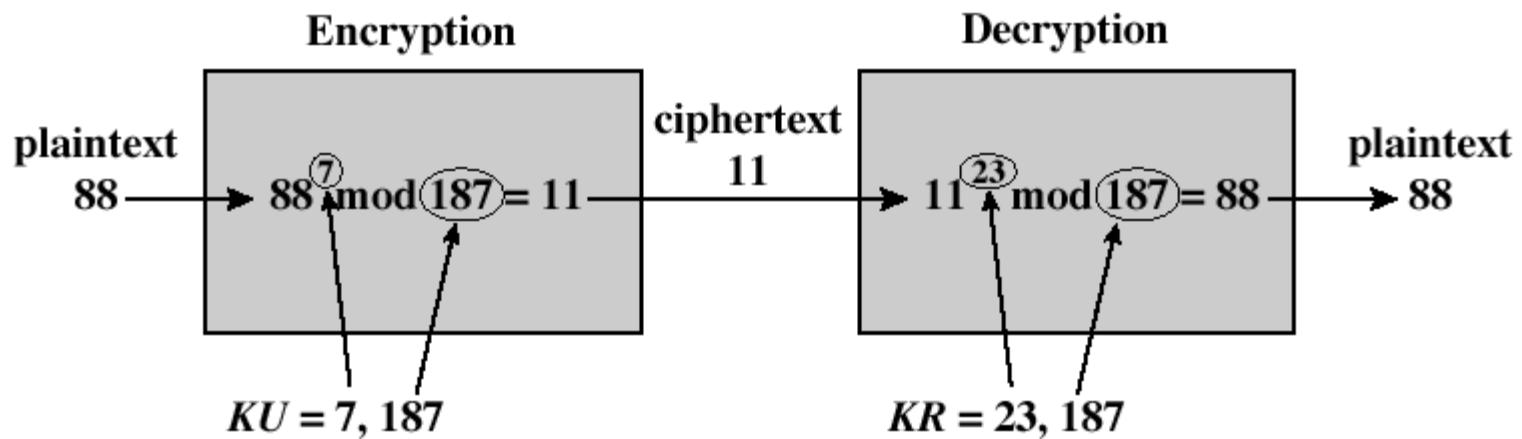
قراردادها و پرتکل RSA

- ♦ 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 بیتهاي 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 واگشاپی رمز می تواند اطلاعاتی را در مورد کلید افشا کند.

برهی نقاط ضعف RSA

- ♦ ویژگی RSA: رمز شده حاصلضرب دو پیام برابر حاصلضرب رمز شده دو پیام است:

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

- ♦ مهاجمی که بخواهد $c \equiv m^e \pmod{n}$ را رمزگشایی کند، اگر بتواند صاحب کلید خصوصی را وادار به رمزگشایی یک پیامی مانند $c' \equiv c r^e \pmod{n}$ که r یک مقدار انتخاب شده توسط مهاجم است بکند، می‌تواند m را بدست آورد:

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

- مقدار mr را بدست آورده در معکوس r ضرب می‌کند.

Factoring Problem

- ◆ This problem is hard, yet probably not NP-complete!
- ◆ For $N=2^n$ a straight algorithm is $O(\frac{2^{n/2}}{n})$
- ◆ The fastest is $\exp(1.91 \cdot n^{1/3} (\log n^{\frac{2}{3}}))$
 - Factoring 1024 bit number is order of 2^{70}
 - Factoring 2048 bit number is order of 2^{90}
- ◆ 4096 bit number is needed to reach 128-bit security

ویرگی دیگر RSA

$$RSA_{KR}(RSA_{KU}(m)) = m = RSA_{KU}(RSA_{KR}(m))$$

اول استفاده از کلید عمومی
و سپس استفاده از کلید
خصوصی

اول استفاده از کلید
خصوص و سپس استفاده
از کلید عمومی

نتیجه یکسان است!

RSA ... ویزگی دیگر

دلیل:

$$\begin{aligned}(m^e \bmod N)^d \bmod N &= m^{ed} \bmod N \\&= m^{de} \bmod N \\&= (m^d \bmod N)^e \bmod N\end{aligned}$$

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 تأخیرهای 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



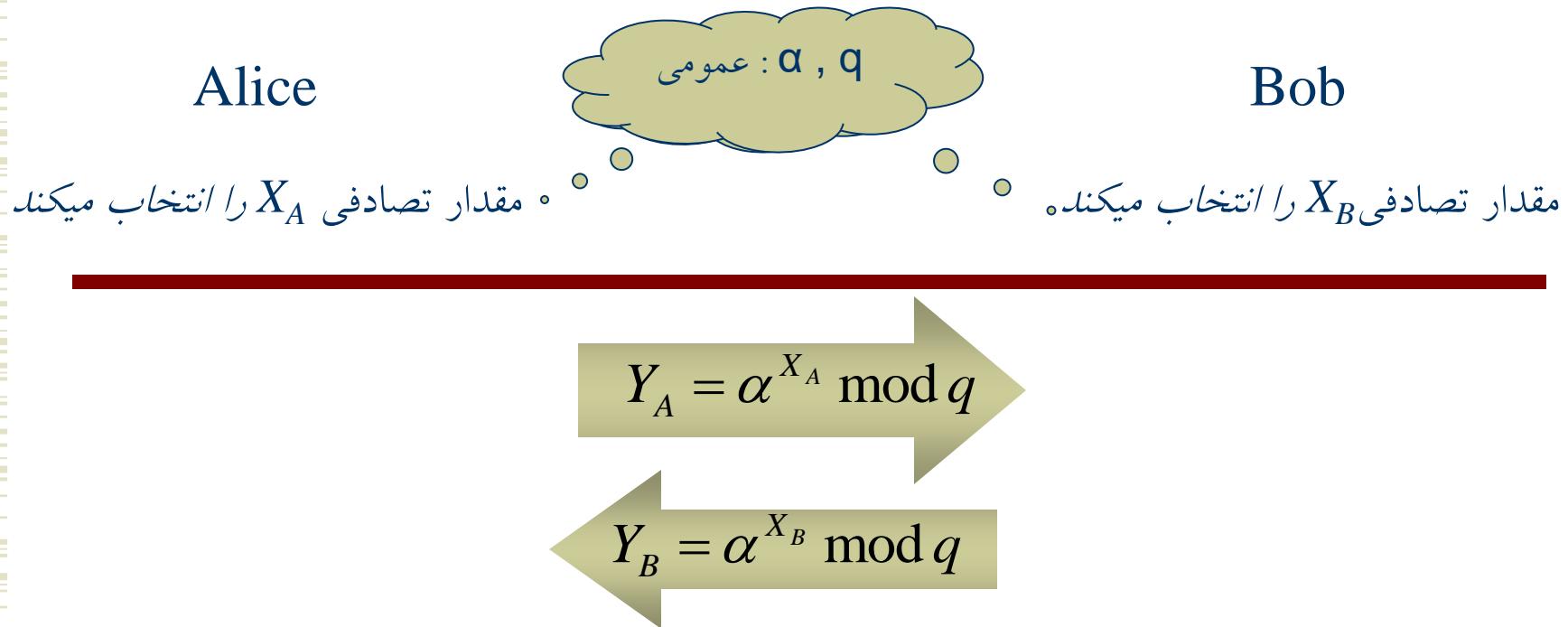
الگوریتم Diffie Hellman

- ♦ برای تبادل کلید مورد استفاده قرار می گیرد
- ♦ طرفین بر روی مقادیر q و a توافق می کنند.
- ♦ q یک عدد اول و a یک مولد (Primitive root) برای این عدد است:
 $\{a \bmod q, a^2 \bmod q, \dots, a^{q-1} \bmod q\} = \{1, \dots, q-1\}$
- ♦ امنیت روش مبتنی بر مشکل بودن لگاریتم گستته است.

Table 8.3 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

الگوریتم Diffie - Hellman



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

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

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

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

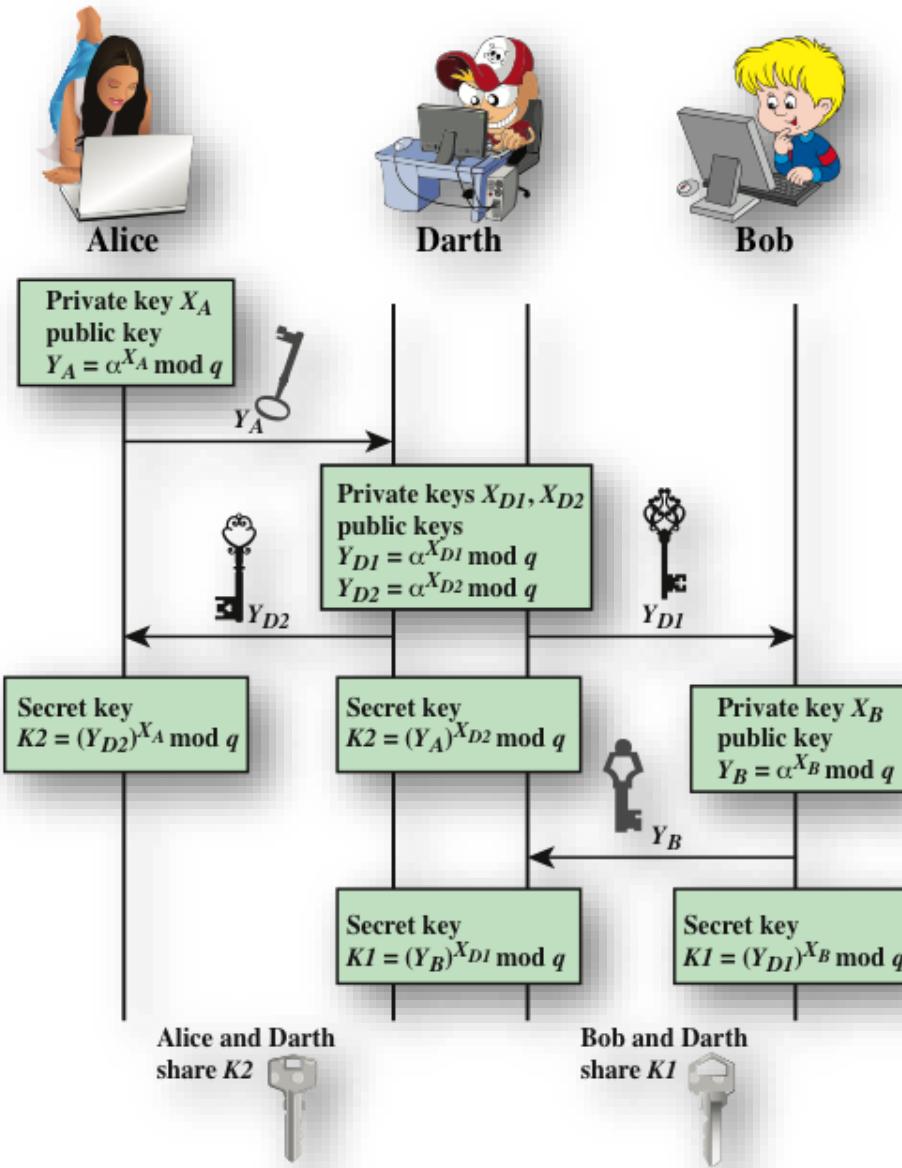
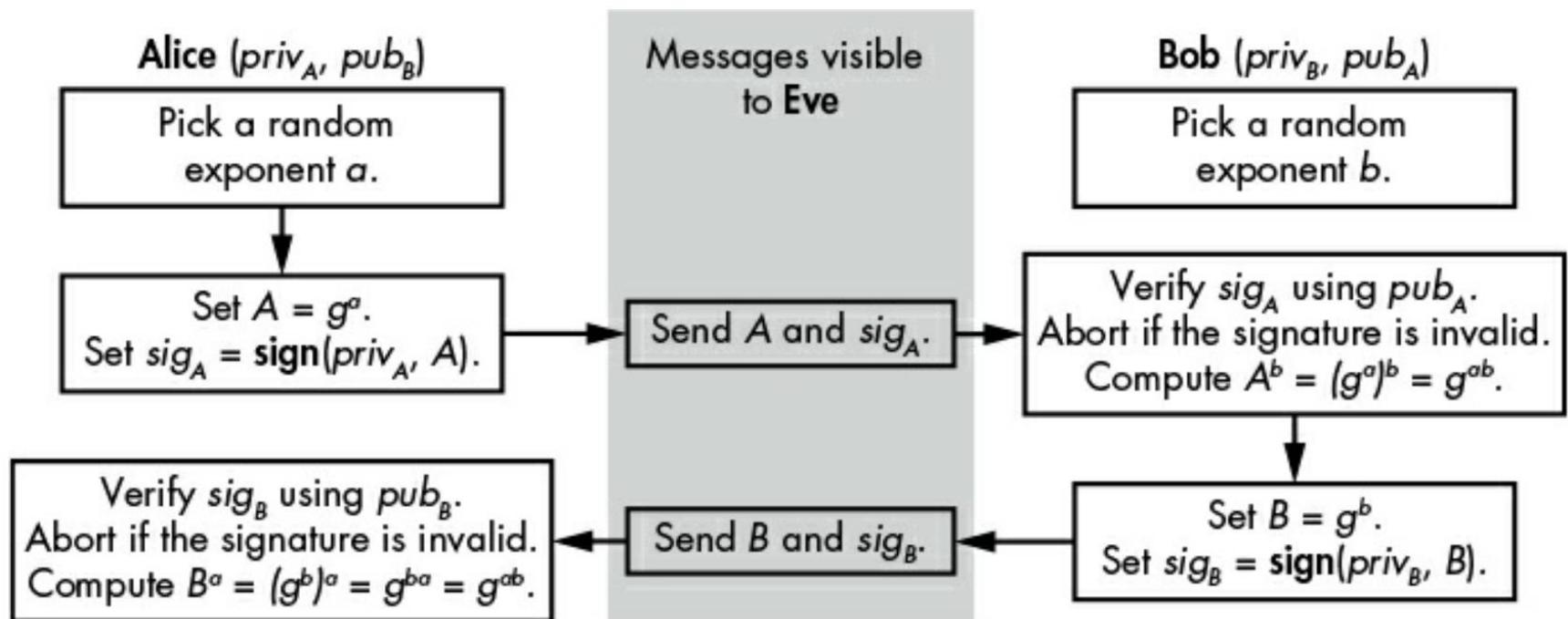


Figure 3.13. Man-in-the-Middle Attack
<http://www.aut.ac.ir/shahriari>

Authenticated Diffie-Hellman



Digital Signature standard (DSS)

- ◆ FIPS PUB 186
- ◆ Makes use of the SHA-1 and presents a new digital signature technique, the Digital Signature Algorithm (DSA)
- ◆ Originally proposed in 1991 and revised in 1993 and again in 1996
- ◆ Uses an algorithm that is designed to provide only the digital signature function
- ◆ Unlike RSA, it cannot be used for encryption or key exchange

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

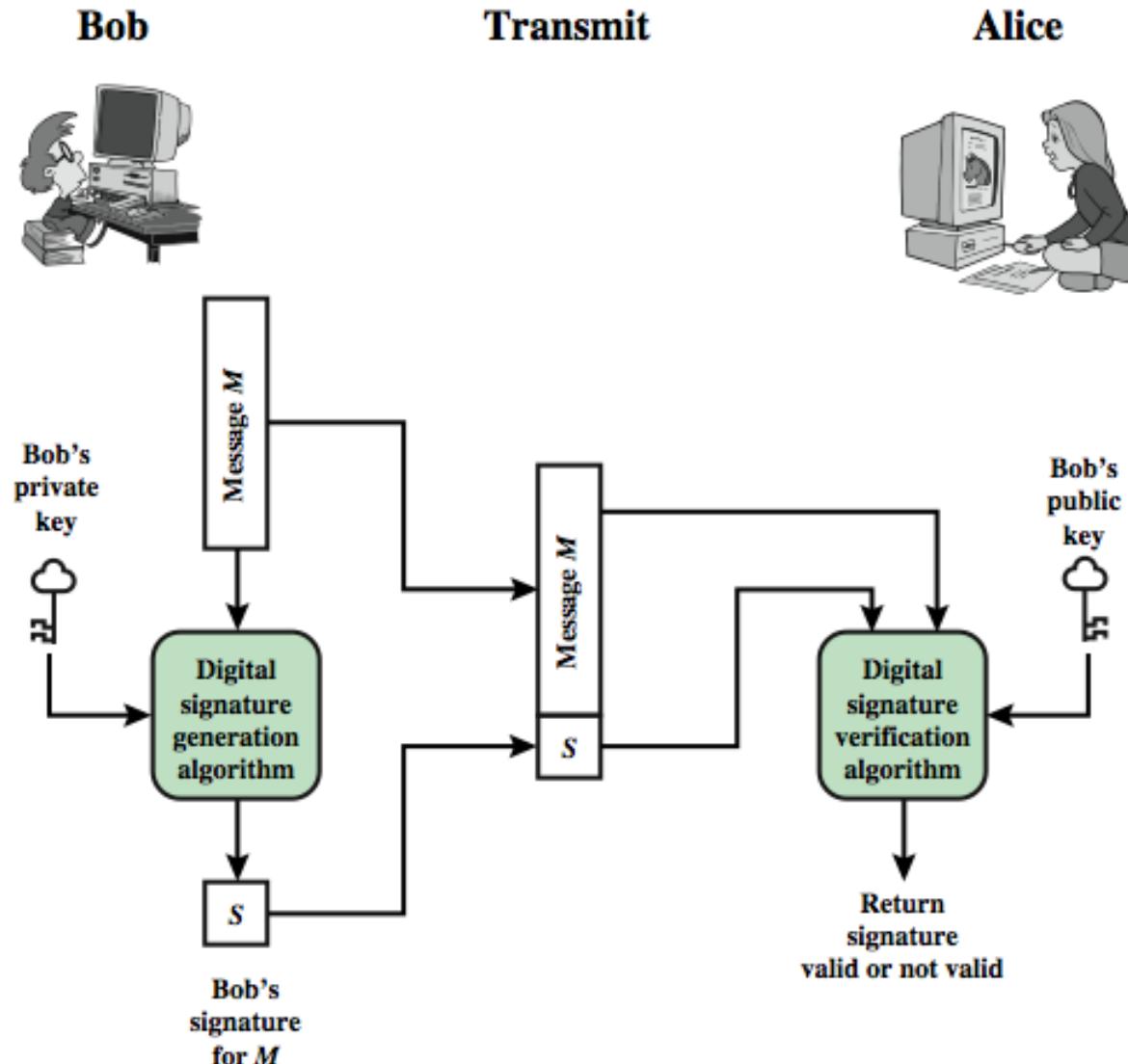
امضا دیجیتال

DIGITAL SIGNATURE

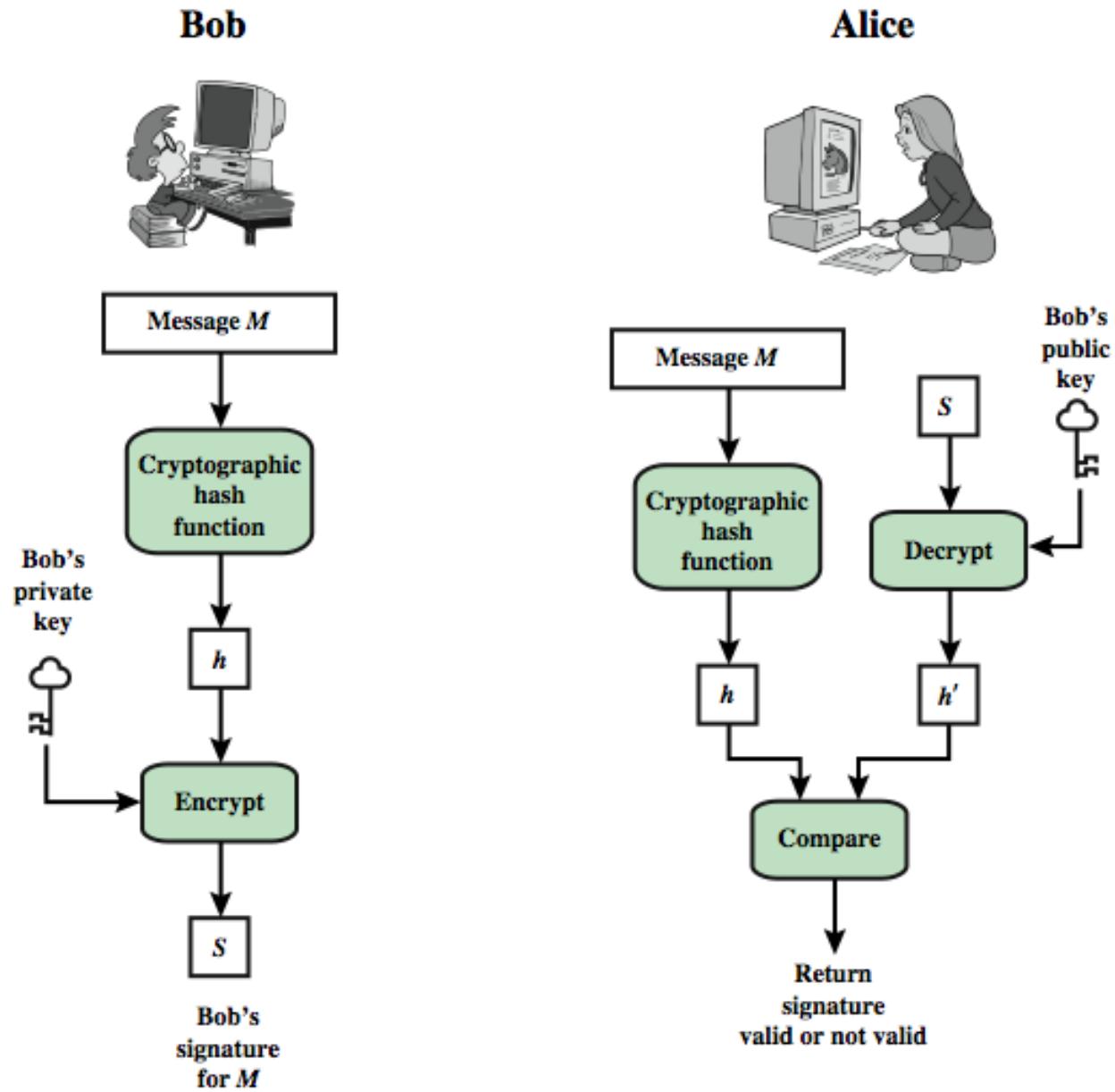
Digital Signatures

- ◆ 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



مقایسه کاربردهای الگوریتم‌های مختلف کلید عمومی

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