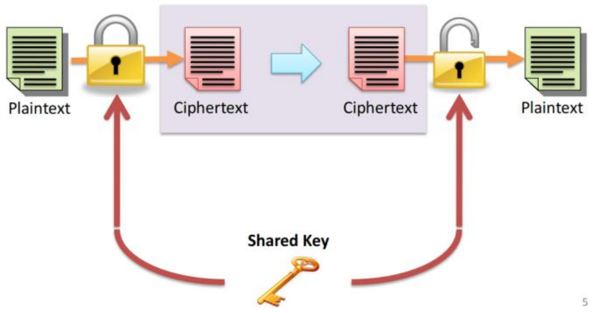
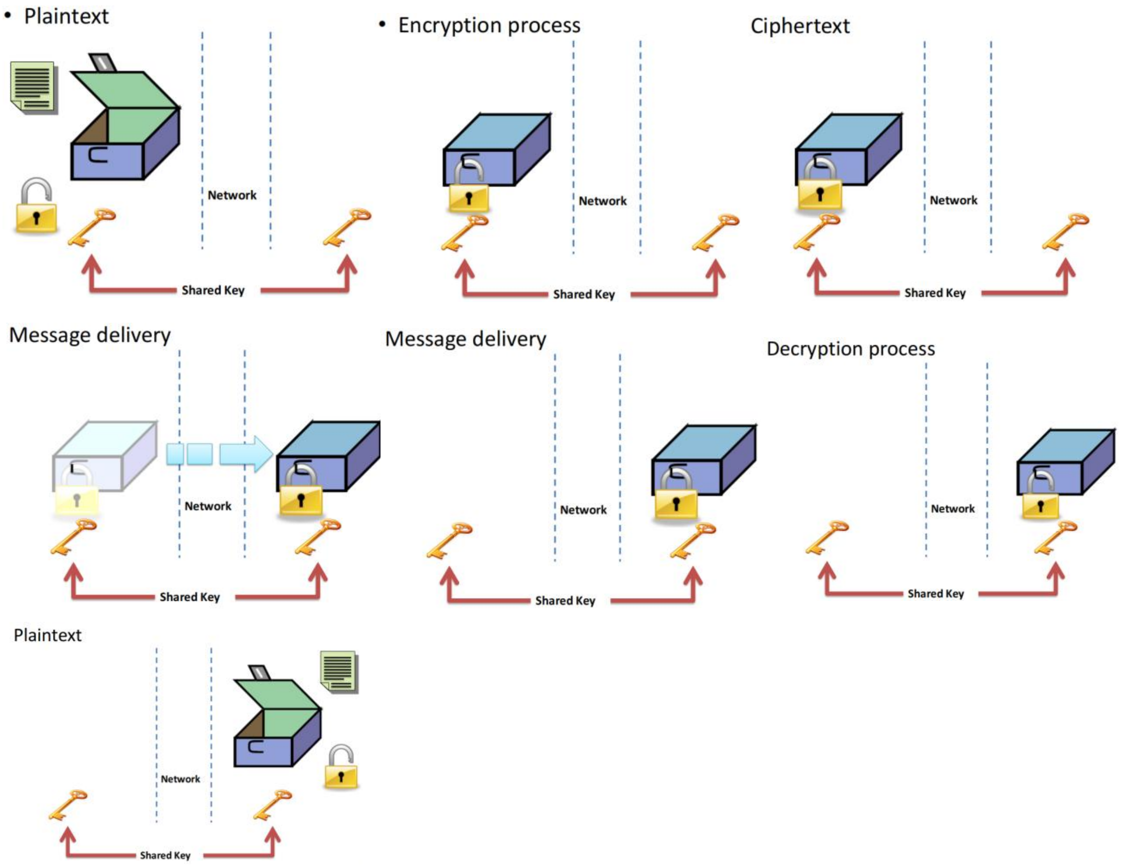
其他内容可访问博客：[either fight | or die (yst-10.github.io)](https://yst-10.github.io)

一. *Conventional Encryption*

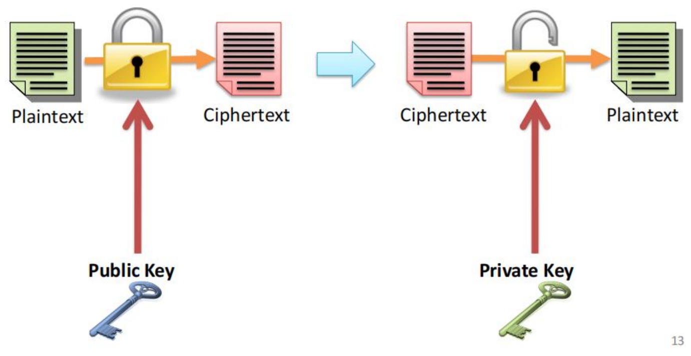
• Encryption and decryption share the same key





二. *Public Key Encryption* (Asymmetric Cryptography)

• Encryption and decryption use different keys



(1)Advantages vs Disadvantages

<1>Advantages

– No need to share the same key 不需要共享相同的密钥

<2>Disadvantages

– How to trust who's the lock's holder?如何信任谁是锁的持有人？
– Who originated the document?谁发起了这份文件？

(2)Requirements:

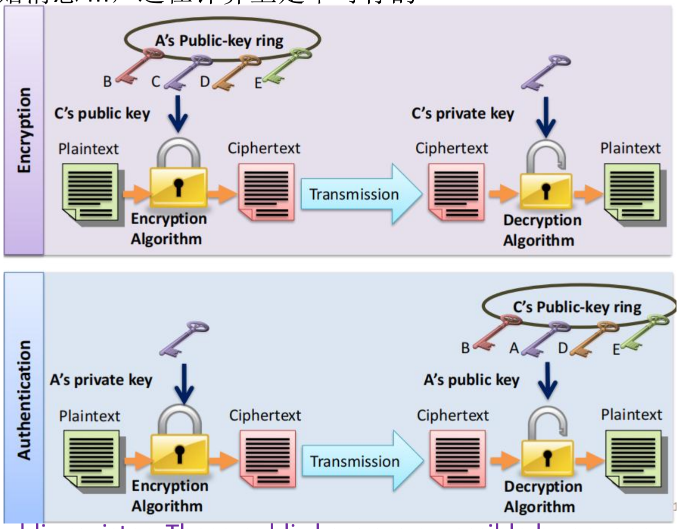
– Easy to generate pair of key(public key 和 private key).

– If *A* knows the public key of *B* it is easy to generate the ciphertext *c* from the original plaintext message *m*. 如果 A 知道 B 的公钥，那么就很容易从原始明文消息 m 中生成密文 c。

– It is easy for *B* to decrypt the message using the private key. B 很容易使用私钥解密消息。

– It is computationally infeasible for an opponent, knowing the ciphertext and public key to recover the original message *m*.

如果对手知道密文和公钥来恢复原始消息 m，这在计算上是不可行的



(3)Basic elements:

– Plaintext
– Encryption Algorithm
– Public and Private keys
– Ciphertext
– Decryption Algorithm

(4)Operation principle

<1>. Each user generate two keys, public and private.两个 Key
<2>. The user put their public key in public register. These public keys are accessible by anyone.
用户将公钥放入公共注册器。任何人都可以访问这些公钥。
<3>. If User1 wishes to send a message to User2, User1 encrypts the message using User2's public key.
如果用户 1 希望向用户 2 发送消息，则用户 1 使用用户 2 的公钥对该消息进行加密。

<4>. User2 decrypts it using his/her own private key. No one else could decrypt the message because it's kept privately.用户 2 使用自己的私钥解密。没有任何人可以解密这个消息，因为它是保密的。

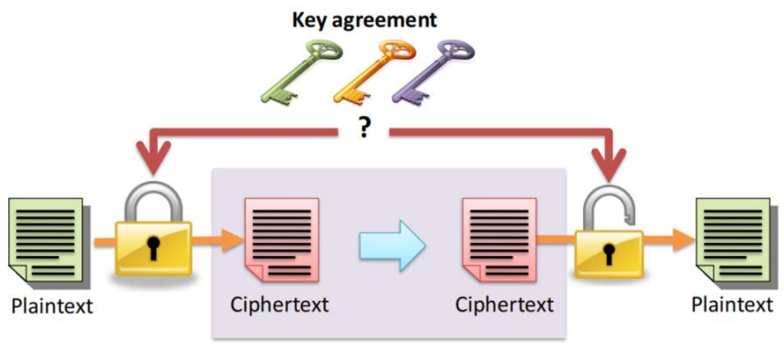
(5)Public-Key Algorithms（大框）

• Diffe-Hellman key exchange
• RSA public-key encryption algorithm
• Digital Signature Standard (DSS)
• Elliptic-Curve Cryptography

(6)One Way Functions

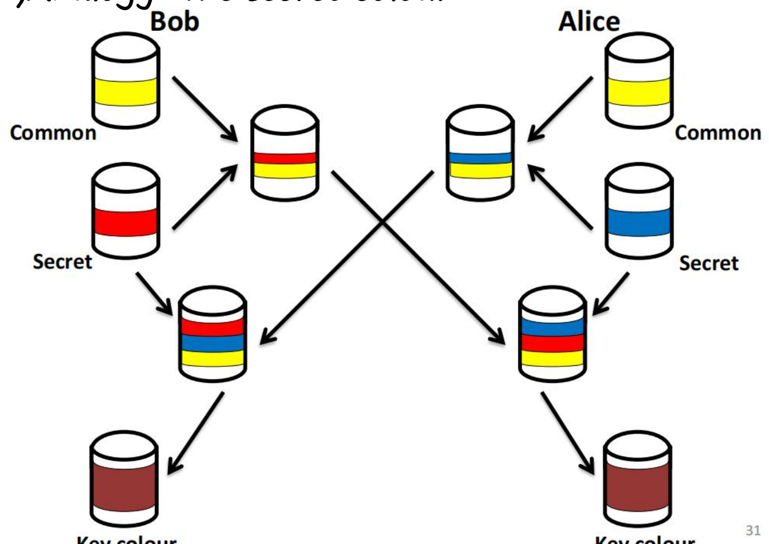
A one-way mathematical function is very easy to do, but very difficult to reverse.
单向数学函数很容易做，但很难逆转。
Public-key encryption can be thought of as a function that anyone should be able to compute, since the encryption key is public, and It should be very difficult for an attacker to efficiently determine a plaintext from knowledge of a ciphertext and the public encryption key.
公密钥加密可以被认为是一个任何人都应该能够计算的函数，因为加密密钥是公共的，而且攻击者应该很难有效地使用它，根据对密文和公共加密密钥的知识来确定明文。

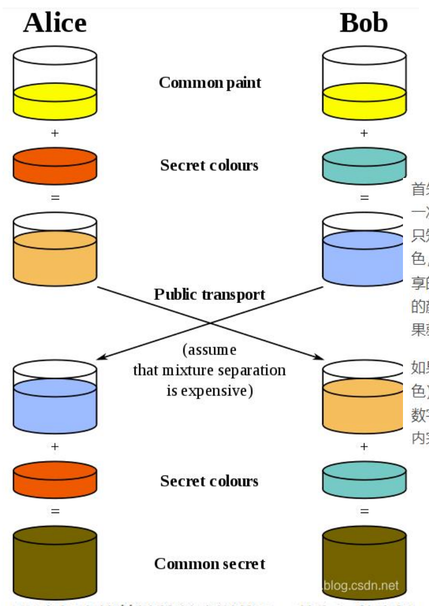
三. Diffie-Hellman Key Exchange



• The algorithm was developed to enable two users to exchange a secret key securely. The algorithm itself is limited to the exchange of keys.
该算法是为了使两个用户能够安全地交换一个密钥。该算法本身仅限于密钥的交换。
• Based on the difficulty to compute discrete logarithms.基于计算离散对数的难度。

(1)Analogy: The secret colour





首先假设双方分别是Alice和Bob，他们就一个随机的初始的颜色达成了一致。这个颜色不需要被保密，但是每一次要不一样。在这个例子中，这个初始的颜色是黄色。然后他们两个人每一个人选择一个秘密的颜色，Alice只知道Alice的颜色，其他任何人都不知道Alice选的是什么颜色，Bob同理。在这个例子中，Alice选择的是红色，Bob选择的是蓝绿色。这个过程的关键部分是Alice和Bob每个人将他们自己的秘密颜色和他们对方彼此共享的颜色（初始的颜色即黄色）混合在一起，导致了分别是orang-tan色，和浅蓝色。他们公开的将混合之后的颜色进行交换。最终，他们每一个人将他们从对方收到的颜色和他们自己的private color进行混合。这个结果就是最终的混合颜色。在这个例子中是黄褐色。这个颜色和对方的颜色是一致的。

如果某个第三方听到了这个交换，他就会知道公共的颜色（黄色）和第一个混合的颜色(orange-tan和浅蓝色），但是对这个第三方来说，确定最终的秘密颜色（黄褐色）在计算上是很困难的。事实上，当在使用大的数字而不是颜色时，这个行为需要大量的计算。这个行为即使是现代超级计算机都是不可能在一个合理的时间内完成的。

DH密钥交换算法简单来说就是一共有三种密钥，一个仅自己知道的私钥，一个可以公开的公钥，一个相同的共享密钥，甲乙双方都需要用对方私钥计算出来的一个公钥，再结合自己的私钥计算出一个相同的共享密钥。来达到安全传输密钥的目的。

(2)primitive root

<1>Modular Arithmetic

a* = *n* x *q* + *r
where *r* is the remainder or *r* = *a* mod *q* & *n*∈*I*

e.g. 14 mod 12 = 2 → 14 = 1x12+2
26 mod 12 = 2 → 26 = 2x12+2

12 is *congruent* not only to 12 itself, but also to 0, so the time called "12:00" could also be

called "0:00", since 0 ≡ 12 mod 12,and 14 mod 12≡26 mod 12

<2>Discrete Logarithm

Definition: If *a* is a primitive root of the prime number *p* then *a* mod *p*, *a* 2 mod *p*, … *a* *p*-1

mod *p*, are distinct and consist of the integers 1 through *p*-1 in some permutation.

如果 a 是素数 p 的原始根，那么 a mod p，一个 a^2 mod p，…a^p-1 mod p 是不同的，由某些排列中的整数 1 到 p-1 组成。

– Evaluate 7^i mod 71 i=1-p-1(70)

7	49	59	58	51	2	14	27	47	45	31	4	28	54	23	19
62	8	56	37	46	38	53	16	41	3	21	5	35	32	11	6
42	10	70	64	22	12	13	20	69	57	44	24	26	40	67	43
17	48	52	9	63	15	34	25	33	18	55	30	68	50	66	36
39	60	65	29	61	1										

7 是 71 的原根，因为 7^i mod 71 在 1 到 70 之间且每个都不同但都有
8 不是，会有重复的

(3)Diffie-Hellman algorithm

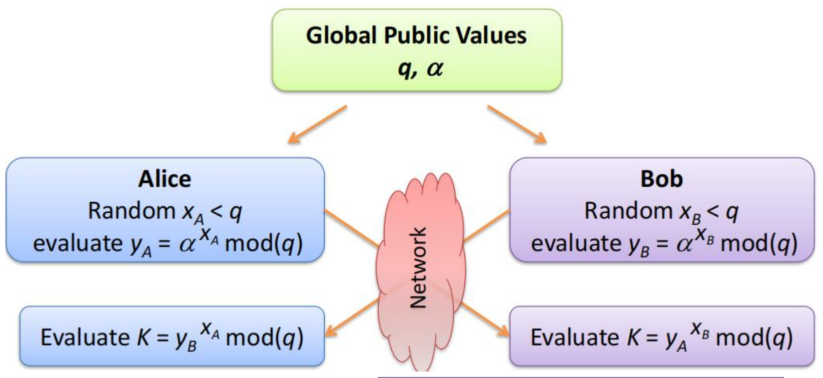
Global Public Elements	
q	Prime number
α	$\alpha < q$, α a primitive root of q

- Alice generates a private key *xA* , evaluates *yA* and sends that to Bob.
- Bob responds by generating a private key *xB* , evaluates *yB* and sends that to Alice.
- Both users generate the key *k*.

User Key generation

Alice	
Select a private x_A	$x_A < q$
Calculate public y_A	$y_A = \alpha^{X_A} \bmod q$

Bob	
Select a private x_B	$x_B < q$
Calculate public y_B	$y_B = \alpha^{X_B} \bmod q$


Global Public Values
q, α
Alice
Random $x_A < q$
evaluate $y_A = \alpha^{X_A}$ mod(q)
Bob
Random $x_B < q$
evaluate $y_B = \alpha^{X_B}$ mod(q)
Network
Evaluate $K = y_B^{X_A}$ mod(q)
Evaluate $K = y_A^{X_B}$ mod(q)

Generation of Secret Key:

Alice	$y_B^{X_A} \bmod q$
Bob	$y_A^{X_B} \bmod q$

– It works because *k* = *yA^XB* and *k* = *yB^XA* produce identical results.

- **Example**
  - q = 353 and α =3.

  - Alice's secret key X_A = 97,
  - Bob's secret key X_B = 233

  - Alice: $Y_A = 3^{97}$ mod 353 = 40
  - Bob: $Y_B = 3^{233}$ mod 353 = 248

- **Key:**
  - Alice: 248^{97} mod 353 = 160
  - Bob: 40^{233} mod 353 = 160

(4)Man-in-the-middle attack

<1>Eve prepares for the attack by generating two random private keys XE1 and XE2 and then computing the corresponding public keys YE1 and YE2.   Eve 通过生成两个随机的私钥 XE1 和 XE2，然后计算相应的公钥 YE1 和 YE2 来准备攻击。
<2>Alice transmits YA to Bob.

<3> Eve intercepts YA and transmits YE1 to Bob. Eve also calculates *K2* = yA ^XE2 mod(*q*).

EVE 拦截了 YA，把 YE1 传给了 Bob。Eve 还计算了 K2 = yA ^XE2 mod (q)。

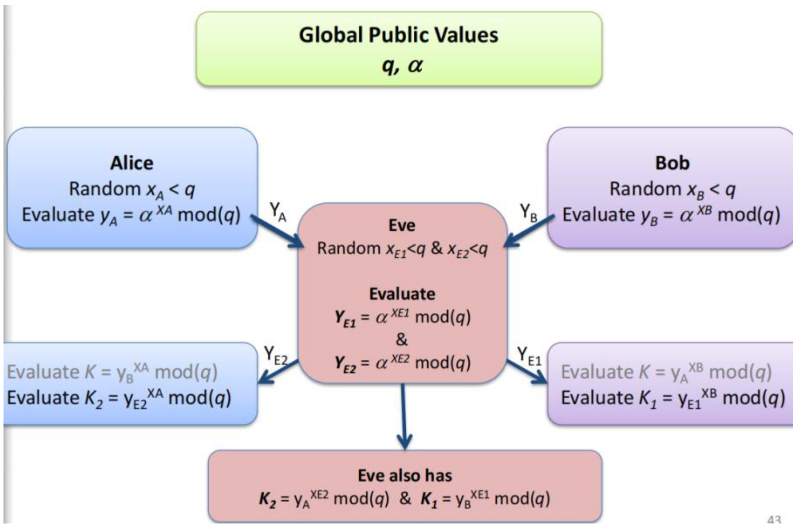
4. Bob receives YE1 and calculates K1 = yE1^ XB mod(*q*).

5. Bob transmits YB to Alice.

6. Eve intercepts YB and transmits YE2 to Alice. Eve calculates *K1* = yB ^XE1 mod(*q*).

7. Alice receives YE2 and calculates K2 = yE2^ XA mod(*q*).

At this point, Bob and Alice think that they share a secret key, but instead Bob and Eve share secret key and Alice and Eve share secret key. 在这一点上，鲍勃和爱丽丝认为他们共享了一个密钥，但鲍勃和 EVE 共享了密钥，爱丽丝和 EVE 共享了密钥。



Global Public Values
q, α
Alice
Random $x_A < q$
Evaluate $y_A = \alpha^{XA}$ mod(q)
Y_A
Bob
Random $x_B < q$
Evaluate $y_B = \alpha^{XB}$ mod(q)
Y_B
Eve
Random $x_{E1}<q$ & $x_{E2}<q$
Evaluate
$Y_{E1} = \alpha^{XE1}$ mod(q)
&
$Y_{E2} = \alpha^{XE2}$ mod(q)
Y_{E2}
Y_{E1}
Evaluate $K = y_B^{XA}$ mod(q)
Evaluate $K_2 = y_{E2}^{XA}$ mod(q)
Evaluate $K = y_A^{XB}$ mod(q)
Evaluate $K_1 = y_{E1}^{XB}$ mod(q)
Eve also has
$K_2 = y_A^{XE2}$ mod(q) & $K_1 = y_B^{XE1}$ mod(q)
43

四. RSApublic key algorithm

(1)Trapdoor One-Way Function

<1>a ***trapdoor one-way function*** is a one-way function f, together with a secret y, such that, given $f(x)$ and y, it is easy to compute x *一个 trapdoor 的单向函数是一个单向函数 f，还有一个秘密的 y，给定 f (x)和 y，计算 x 很容易*

• *The private key here is our trapdoor.这里的私钥是我们的 trapdoor。*

(2)Relatively Prime Numbers 相对素数

<1>Definition

– Relative primes are two numbers that do not have a common divisor. 不具有公约数

<2>Example:

• 10 and 11 are relative primes

(3)Euler's Totient Function

• This function gives the number of positive integers less than n and *relatively prime* to n.

这个函数给出了小于 n 且与 n 为相对素数的正整数的数目。

- By convention $\phi(1) = 1$
- If p is a prime number $\phi(p) = p-1$
- If $n = pq$ where p and q are prime and are not equal, then

  $\phi(n) = \phi(pq) = \phi(p)\times\phi(q) = (p - 1) \times (q - 1)$

If p & q are equal, then $\phi(n) = p\,(q - 1)$

Example: $\phi(21) = \phi(3) \times \phi(7) = \mathbf{12}$ then integers are 1,2,4,5,8,10,11,13,16,17,19,20（小于 21 且与 21 成相对素数的数）

(4)The congruent module of n 同余

• A congruence relation (or simply congruence) is an equivalence relation on an algebraic structure. 同余关系（或简单的同余）是代数结构上的等价关系

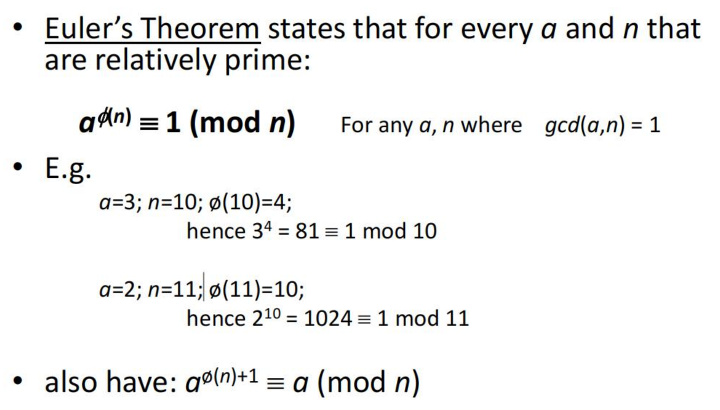
• Basic Example:

If a and b satisfy $a \bmod n = b \bmod n$, then $a \equiv b \bmod n$

e.g. 73 mod 23 = 4 and 4 mod 23 = 4

then $73 \equiv 4 \bmod 23$

(5)Euler's theorem(aka Fermat-Euler theorem or Euler's totient theorem)



- Euler's Theorem states that for every a and n that are relatively prime:

  $a^{\phi(n)} \equiv 1 \pmod{n}$ For any a, n where $\gcd(a,n) = 1$

- E.g.

  $a=3$; $n=10$; $\phi(10)=4$;
  hence $3^4 = 81 \equiv 1 \bmod 10$

  $a=2$; $n=11$; $\phi(11)=10$;
  hence $2^{10} = 1024 \equiv 1 \bmod 11$

- also have: $a^{\phi(n)+1} \equiv a \pmod{n}$

(6)RSA algorithm

<1>

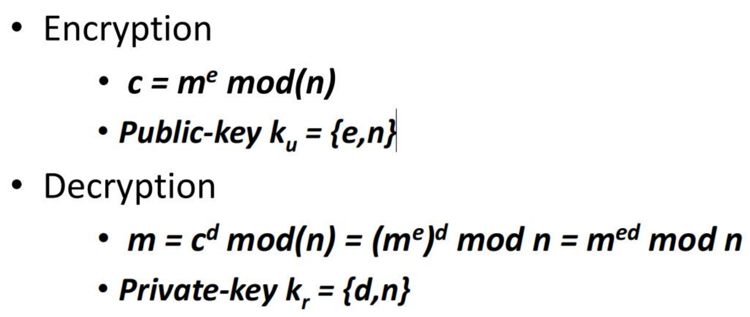
• RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1.

RSA 是一个块密码，其中明文和密文是 0 和 n-1 之间的整数。

• So the first step is to represent the plaintext block in terms of an integer between 0 and $n-1$.

所以第一步是用一个介于 0 和 n-1 之间的整数来表示明文块。

<2>



- Encryption
  - $c = m^e \bmod (n)$
  - Public-key $k_u = \{e,n\}$
- Decryption
  - $m = c^d \bmod (n) = (m^e)^d \bmod n = m^{ed} \bmod n$
  - Private-key $k_r = \{d,n\}$

•Both sender and receiver must known n & e. Only the receiver knows d

<3>The algorithm satisfies:满足

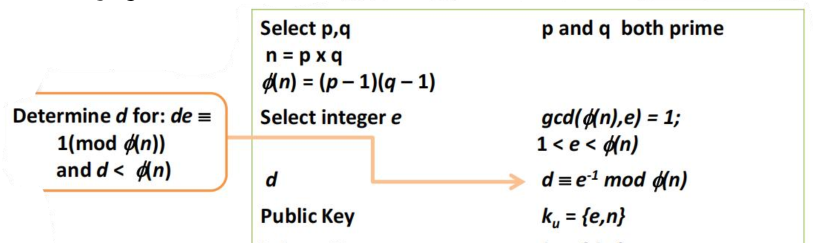
1. It is possible to find e, d and n such that $m^{ed} = m \bmod n$ for all $m<n$.

2. It is easy to calculate m^e and c for all $m < n$

3. For large values of e and n it is not feasible to determine d given e and n.

对于 e 和 n 的大值，给定 e 和 n 来确定 d 是不可行的。

<4>Key generation



Select p,q	p and q both prime
n = p x q	
$\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n),e) = 1$; $1 < e < \phi(n)$
d	$d \equiv e^{-1} \bmod \phi(n)$
Public Key	$k_u = \{e,n\}$

Determine d for: $de \equiv 1 \pmod{\phi(n)}$ and $d < \phi(n)$

Example:

Select p,q	$p = 47$ and $q = 59$
n = p x q	$n = 2773$
$\phi(n) = (p-1)(q-1)$	$\phi(n) = 2668$
Select integer e	e = 17
d	$d \equiv e^{-1} \bmod \phi(n) \rightarrow d = 157$
Public Key	$k_u = \{17,2773\}$
Private Key	$k_r = \{157,2773\}$

$$d = \frac{K \cdot \phi(n)+1}{e}$$

求 d

$d \equiv e^{-1} \bmod \phi(n) \rightarrow ed \bmod \phi(n) = 1 \bmod \phi(n)$

and $1 \bmod \phi(n) = K.\phi(n)+1 \bmod \phi(n)$

where K is an integer {1, 2, 3, ..}

Candidates of 1 mod $\phi(n)$:

2669	5337	8005	10673	13341	16009	18677	21345	24013	26681
29349	32017	34685	37353	40021	42689	45357	48025	50693	53361
56029	58697	61365	64033	66701	69369	72037	74705	77373	80041

- You will need to find two numbers e and d whose product is a number equal to 1 mod $\phi(n)$. i.e. *ed* mod $\phi(n)$= 1
- Here are few answers:
  - 2669 = 17 * 157 → d = 157.
  - 48025 = 17 * 2825 → d = 2825.

$$d = \frac{K \cdot \phi(n)+1}{e}$$

- The above values satisfy the rules:
  - *ed* mod $\phi(n)$= 1
  - Both *e* and *d* are relative primes to $\phi(n)$,

<5>RSA Example（接上求 c，m）

- Encode the literal characters (space), a, b, ..., z with the biagrams 00, 01, ..., 26
- The message is encoded in block of length two

假定 m:

E	R	R	A	R	E
05	18	18	01	18	05
0518		1801		1805	

- Encryption, e.g $c_1 = 518^{17} \bmod 2773 = 1787$

1787	**2003**	**2423**

- Decryption, e.g $m_2 = 2003^{157} \bmod 2773 = 1801$

0518	**1801**	**1805**

<6>Why RSA Works

- Suppose (n,e) and (n,d) are key pair and $m<n$.

Then **$m = c^d \bmod(n) = (m^e)^d \bmod n = m^{ed} \bmod n$**

- To prove $(m^e)^d \equiv m \bmod n$

$$
\begin{aligned}
m^{ed} &= m^{(1+k(p-1)(q-1))} \\
&= m(m^{(p-1)(q-1)})^k \\
&\equiv m(1^k) \bmod n \\
&\equiv m \bmod n
\end{aligned}
$$

(7)RSA Security

• *Possible approaches to attacking RSA are:* 攻击 RSA 的可能方法有：

– Brute force key search - infeasible given size of numbers
  暴力强制密钥搜索-不可行的给定大小的数字

– Mathematical attacks - based on difficulty of computing ø(n), by factoring modulus n
  数学攻击-基于计算ø(n)的难度，通过分解模量 n

– Timing attacks - on running of decryption
  在解密运行时进行的定时攻击

– Chosen ciphertext attacks - given properties of RSA
  选择的密文攻击-RSA 的给定属性

(8)Problem with Public-key Encryption

<1>Need relatively *long keys*, 1024 or more bits 需要相对较长的密钥，1024 位或更多位

<2>*Computational costs*: slower than DES by a factor of a thousand!

计算成本：比 DES 慢 1000 倍！

<3>*Long-plaintext security issues* 长明文安全问题

– For longer plaintext → split into “blocks” and then encrypt these separately.
  对于较长的明文，→ 被分割成“块”，然后分别加密它们。

– Essentially encrypting these blocks using the public key equivalent of ECB mode for a block cipher, which is not desirable. 本质上是使用类似于 ECB 模式的公钥对这些块进行加密，这是不可取的。

– However, there are no alternative modes of operation proposed for public-key encryption.
  然而，对于公钥加密，并没有其他的操作模式。

<4>Symmetric cryptography and Public-key cryptography have both strength and weakness.对称密码学和公钥密码学既有优点也有缺点。

<5>In particular, we often need the good properties of both: 特别是，我们经常需要这两者的良好特性：

– Ease of key sharing from Public-key cryptography.容易从公钥密码学进行密钥共享。

– Efficiency and simplicity from Symmetric cryptography 对称密码学中获得的效率和简单性

四. *Hybrid Encryption* 混合加密

For secure communications with potentially high traffic, the ***hybrid cryptography*** gives an ideal answer. 对于具有潜在高流量的安全通信，混合加密技术给出了一个理想的答案。

• Alice wishes to exchange several messages with Bob securely. We assume Alice and Bob share their public keys.

爱丽丝希望与 Bob 安全地交换几条信息。我们假设爱丽丝和鲍勃分享了他们的公钥。

• First, Alice creates a fresh session key (symmetric encryption key), KAB.

首先，Alice 创建一个新的会话密钥（对称加密密钥），KAB。

1. A -> B : $\{K_{AB}\}K_{B;\ pub}$
2. B -> A : $\{M1\}K_{AB}$
3. A -> B : $\{M2\}K_{AB}$
4. B -> A : $\{M3\}K_{AB}$

• (In Step 2, we assume Bob decrypts the message and obtains KAB.)

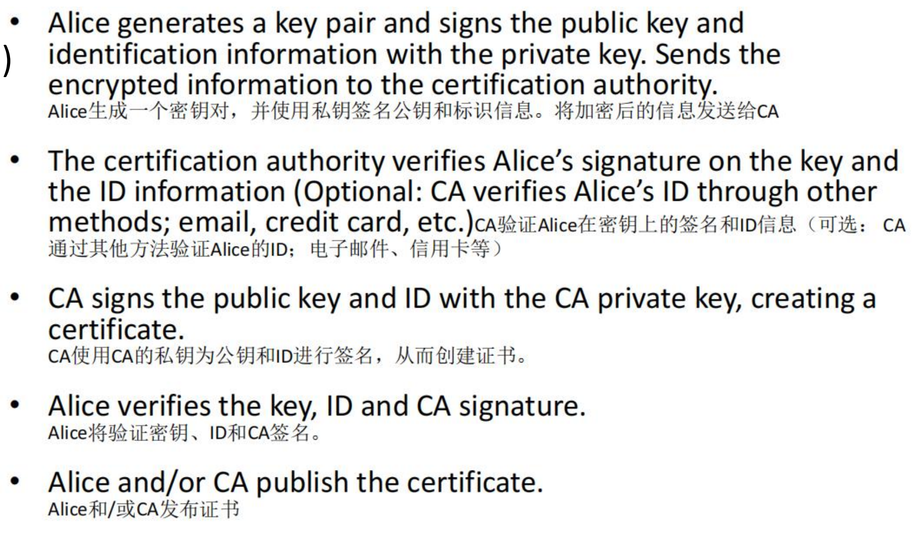
（在步骤 2 中，我们假设 Bob 解密消息并获得 KAB。）

We can see that Public Keys are useful, but the authenticity of public keys is still vital …

我们可以看到公钥是有用的，但公钥的真实性仍然至关重要。

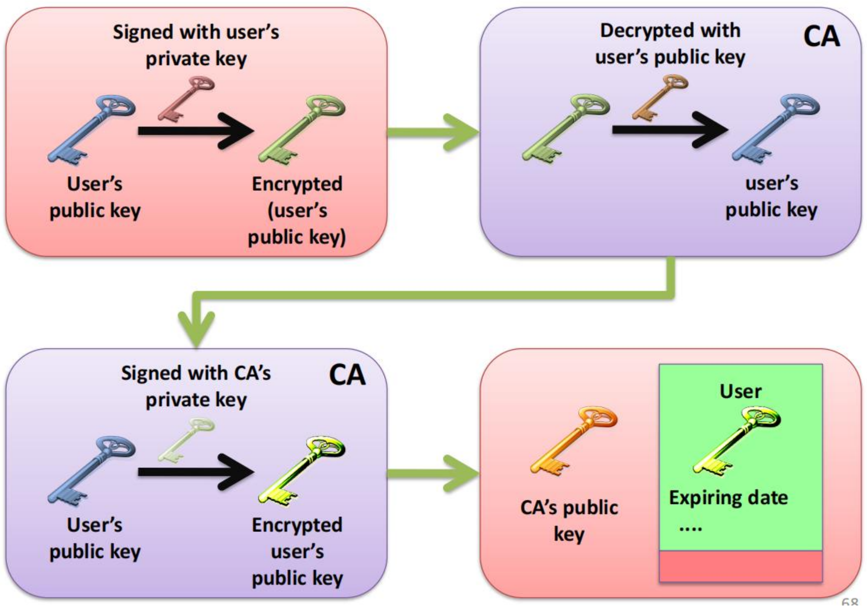
五. *Certificates*

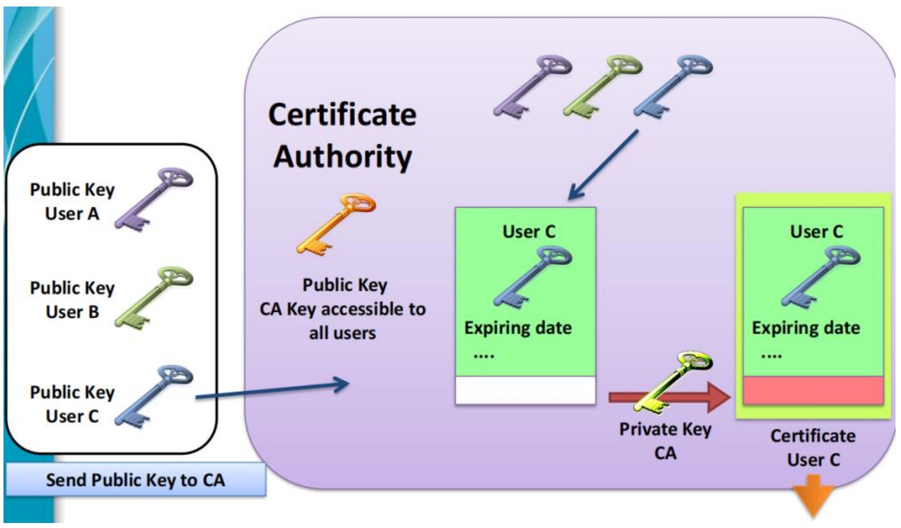
(1)



- Alice generates a key pair and signs the public key and identification information with the private key. Sends the encrypted information to the certification authority.
  Alice生成一个密钥对，并使用私钥签名公钥和标识信息。将加密后的信息发送给CA
- The certification authority verifies Alice's signature on the key and the ID information (Optional: CA verifies Alice's ID through other methods; email, credit card, etc.)CA验证Alice在密钥上的签名和ID信息（可选： CA通过其他方法验证Alice的ID：电子邮件、信用卡等）
- CA signs the public key and ID with the CA private key, creating a certificate.
  CA使用CA的私钥为公钥和ID进行签名，从而创建证书。
- Alice verifies the key, ID and CA signature.
  Alice将验证密钥、ID和CA签名。
- Alice and/or CA publish the certificate.
  Alice和/或CA发布证书

(2)Obtaining a Certificate


Signed with user's private key
User's public key
Encrypted (user's public key)
CA
Decrypted with user's public key
user's public key
CA
Signed with CA's private key
User's public key
Encrypted user's public key
CA's public key
User
Expiring date
....
68



• A certification authority guarantees the connection between key and any user with access to CA can get any certificate from it: 证书颁发机构保证密钥和任何能够访问 CA 的用户之间的连接，可以从它获得任何证书：

– A person

– A role (“Director of Personnel”)

– An organisation

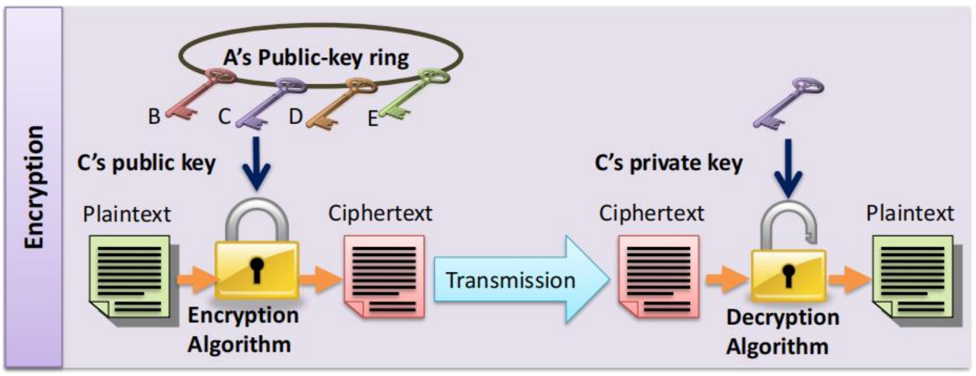
– A piece of hardware/software

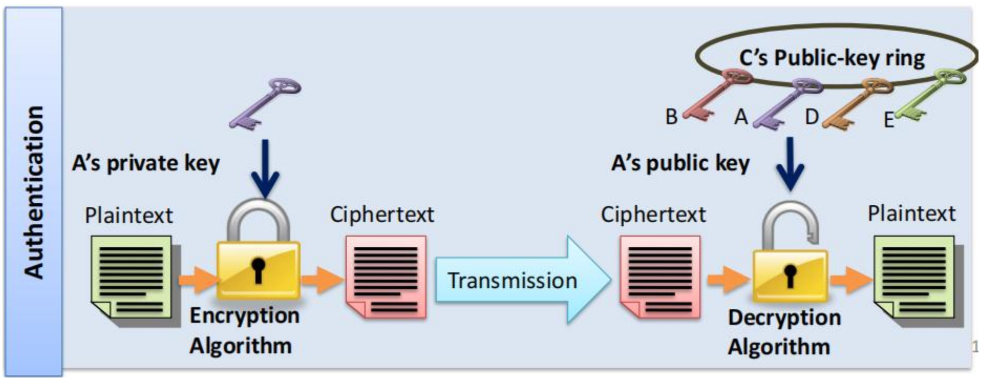
– An account

• Only the CA can modify a certificate. 只有 CA 可以修改证书。

• Because cannot be forged, certificates can be placed in a public directory.
因为不能伪造，所以证书可以放置在公共目录中。

Public Key Cryptography Summary





六. *Message Authentication (Data integrity)*

Prevent:

- Content modification (message integrity) 内容修改（消息完整性）
- Masquerade (validation of the originator's identity) 化妆舞会（对发起者身份的验证）
- Sequence modification 序列修改
- Time modification 时间修改
- Repudiation 拒绝

- ***Prove or show to be true or genuine***

  – Contents of message 消息内容
  – A person, a computer or an address 个人、电脑或地址
  – A timestamp 时间戳
  – A signature 签名

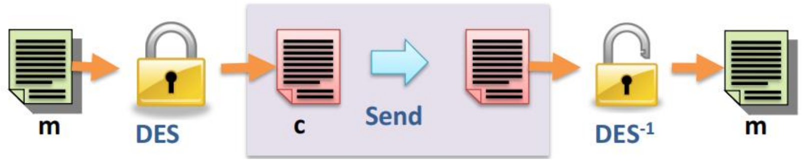
(1)Data Integrity

• Data Integrity: assurance that data has not been altered in an *unauthorised* (which includes accidental) manner. 确保数据未以未经授权（包括意外）的方式进行更改。

• Data integrity is not concerned with the *prevention* of alteration of data, but provides a means for *detecting* whether data has been manipulated in an unauthorised way. 数据的完整性与防止数据的更改无关，而是提供了一种检测数据是否以未经授权的方式被操纵的方法。

(2)Message Authentication

<1>Data integrity can be done by using conventional encryption.



– If only the sender and receiver share the key.

– If the message includes error-detection code and a sequence number the receiver is assured that the message has not been altered and is in the correct sequence.如果该消息包含错误检测代码和序列号，则接收方确信该消息没有被更改，并且处于正确的顺序中。

– If the message includes a time-stamp the receiver is assured that the message has not been delayed. 如果该消息包含一个时间戳，则接收方将确保该消息没有被延迟。

– But it does not provide a signature. i.e. The receiver can fabricate a message, encrypted using the share key and attribute it to the sender.但它并没有提供一个签名。即，接收方可以制造一个消息，使用共享密钥加密，并将其属性给发送者。

<2>It is not always desirable to encrypt a message to provide authentication 以下是没必要的情况

1. *Broadcasting a message* to many destinations with an authentication tag;

使用身份验证标签向许多目的地广播消息；

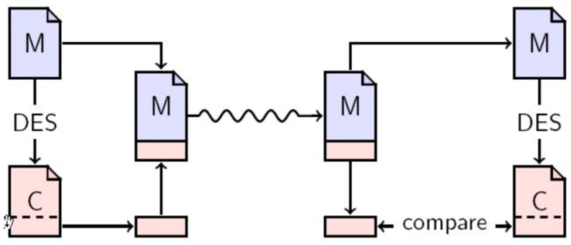
- It is cheaper and more reliable to have one destination responsible for monitoring of authenticity. 有一个目的地负责监测真实性会更便宜、更可靠。
- If the message is not authentic then raise an alarm. 如果该消息不真实，则会发出警报。

2. Due to heavy load in the system, ***decryption cannot be afforded***, so it is better to check the authentication of the message in a random basis. 由于系统中的负载较大，无法进行解密，所以最好是随机地对消息进行身份验证

3. ***Authentication of a computer program***. The program can be checked when assurance about its *integrity* is needed.

对计算机程序的身份验证。当需要保证该程序的完整性时，可以检查该程序。

<3>



• Suppose the ECB mode is used for the DES encryption, is it a good method for message authentication? 假设 ECB 模式用于 DES 加密，它是消息身份验证的好方法吗？

(3)A simple example for Data integrity

Longitudinal redundancy check (LRC)纵向冗余检查（LRC）

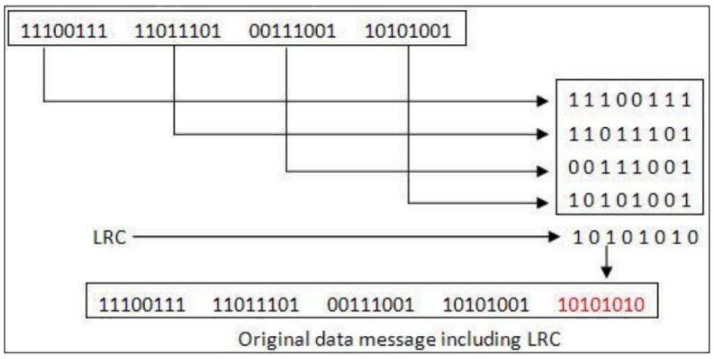
– The input is a sequence of *n*-bit blocks

– The input is processed one block at a time, to produce an *n*-bit hash function

$$c_i = b_{i1} \oplus b_{i2} \oplus b \ldots \oplus b_{im}$$

– $b_{i,j}$ is the *i*th bit of the *j*th block

	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



一列一列算。第一排先和第二排算，结果在和第三排，总结为，1 个数为奇数，则为 1，1 个数为偶数，则为 0

• Good for random data, not so good for predictable formatted data
适合于随机数据，而不利于可预测的格式化数据

七. Hash functions

(1)介绍

• *Cryptographic* hash functions 加密哈希函数

• A hash function accepts a *variable-size* message ***m*** and produces a *fixed-size* **message digest** ***h*(*m*)** as output. 一个哈希函数接受一个可变大小的消息 m，并生成一个固定大小的消息摘要 h (m)作为输出。

• A hash function does ***not*** use a secret key.哈希函数不使用密钥。

• The sender sends the message and message digest so the receiver can authenticate the message. 发送方发送消息和消息摘要，以便接收方能够对消息进行身份验证。

• Hash functions are *publicly computable*. We always assume an attacker knows the details of a hash function. 哈希函数是公开可计算的。我们总是假设攻击者知道哈希函数的细节。

(2)Practical properties of a hash function

<1>h can be applied to a block of data of any size

h 可以应用于任何大小的数据块

<2>h produces a fixed-length output. h 产生一个固定长度的输出。

<3>h(*m*) is easy to compute for any *m*. h (m)对于任何 m 都很容易计算。

(3)Security Properties of a hash function

<1> **Preimage Resistance**

Given an output (hash) value *c*, it should be a difficult operation to find any input value *m* such that *h(m)=c*. Output *h*(*m*) of a function *h* is often called the ***image of m***, and hence *m* is referred to as a ***preimage of h(m)***. 给定一个输出（哈希）值 c，应该很难找到任何输入值 m，从而使 h (m)=c。函数 h 的输出 h (m)通常被称为 m 的图像，因此 m 被称为 h (m)的**预像**。

<2> **Second Preimage Resistance**

given an input *m* and its output (hash) value *h(m)*, it should be a difficult operation to find any other input value *n* such that *h(m)=h(n)*. 给定一个输入 m 及其输出（哈希）值 h (m)，要找到任何其他输入值 n，使得 h (m)=h (n)应该是一个困难的操作。

<3>**Collision Resistance**

for a hash function *h*, it is hard to find two different inputs *m* and *n* such that *h(m)=h(n)*
对于一个哈希函数 h，很难找到两个不同的输入 m 和 n，从而使 h (m)=h (n)

(4)Application of Hash functions

<1>Application Requiring Preimage Resistance

password storage protection 密码存储保护

<2>Application Requiring Second Preimage Resistance

Software downloading with a hash of the executable code 软件下载与可执行代码的散列

<3>Application Requiring Collision Resistance

Bob writes an IOU message, sends it to Alice, and she signs it. Bob finds two messages with the same hash, one of which requires Alice to pay a small amount and one that requires a large payment. Alice signs the first message, and Bob is then able to claim that the second message is authentic. 鲍勃写了一条借据信息，寄给爱丽丝，爱丽丝签了名。Bob 发现了两个具有相同散列的消息，其中一条需要 Alice 支付少量费用，另一条需要支付一大笔费用。Alice 为第一个消息签名，然后 Bob 就可以声称第二条消息是真实的。

(5)Hash functions

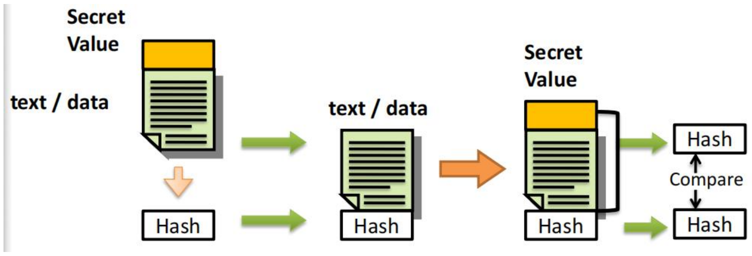
• It is possible to use a hash without using encryption. 可以使用散列而不使用加密。

• The sender and receiver share a common secret value *SAB* value and the message.

发送方和接收方共享一个共同的秘密值 SAB 值和消息。

• The hash is obtained from the concatenation of the secret value and the message.
hash 是从秘密值和消息的连接中获得的。

• The sender sends the message and the hash (not the secret value).
发件人发送消息和 hash（而不是秘密值）。



• The receiver concatenates the secret value & received message.
接收方连接秘密值和接收到的消息。

• The receiver evaluates the hash function and compares it with the senders hash function. If equal, the message is authentic 接收方计算哈希函数，并将其与发送者哈希函数进行比较。如果条件相同，则该信息是真实的

八. Secure Hash Algorithm (SHA)

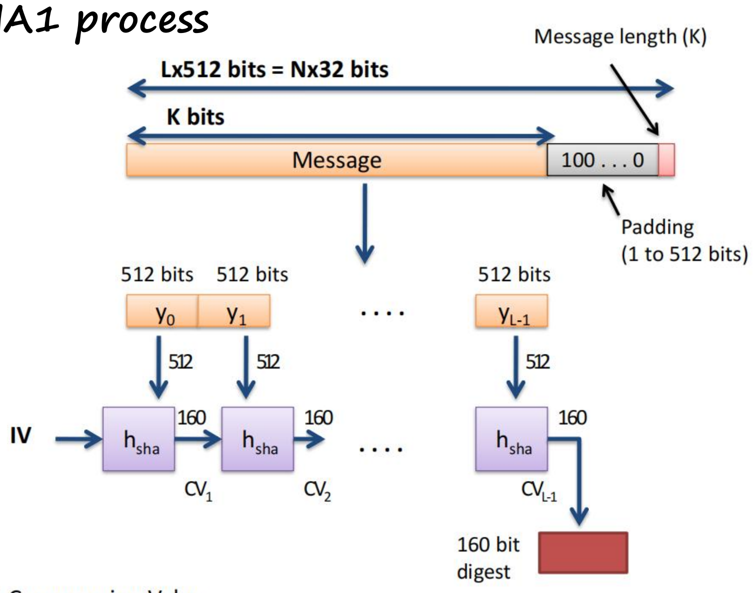
(1)SHA1 (Secure Hash Algorithm 1):

• The processing consists of:
数据填充
1. Append padding bits添加填充位
2. Append length附加长度
3. Initialise MD buffer
4. Process message in 512-bit blocks
5. Output: 160 bits

(2)Hash output

– Every bit of hash code is a function of every bit of the input.
  每一位哈希代码都是每一位输入的函数。
– It is unlikely that two messages chosen at random will have the same hash code.
  随机选择的两个消息不太可能具有相同的哈希代码。
– It is infeasible to produce two different messages having the same message digest.
  生成具有相同消息摘要的两个不同消息是不可行的。

(3)SHA1 process



CV: Compression Value.

在SHA1算法中，我们必须把原始消息（字符串，文件等）转换成位字符串。SHA1算法只接受位作为输入。
假设我们对字符串“abc”产生消息摘要。首先，我们将它转换成位字符串如下：

01100001 01100010 01100011

——————————————

'a'=97 'b'=98 'c'=99

这个位字符串的长度为24。下面我们需要5个步骤来计算MD5。

<1>步骤一：数据填充(Append Padding Bits)

SHA-1 是按照分块进行处理的，分块长度为 512bit，大多数情况下，数据的长度不会恰好满足是 512 的整数倍，因此需要进行「**padding**」到给定的长度。

「**填充规则**」：原始明文消息的 b 位之后补 100…，直到满足 $b + paddingLength \% 512 = 448$，那如果 $b \% 512$ 在[448，512(0)]之间呢，则在增加一个分块，按照前面的规则填充即可。（至少补一位，最多 512 位）下面是例子：

原始信息：01100001 01100010 01100011

补位第一步：01100001 01100010 01100011 1

首先补一个“1”

补位第二步：01100001 01100010 01100011 10.....0

然后补423个“0”

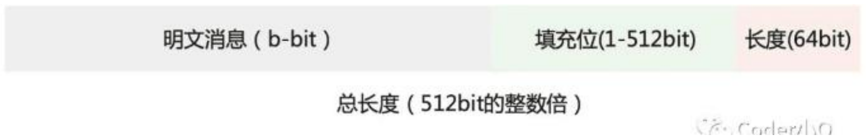
我们可以把最后补位完成后的数据用16进制写成下面的样子

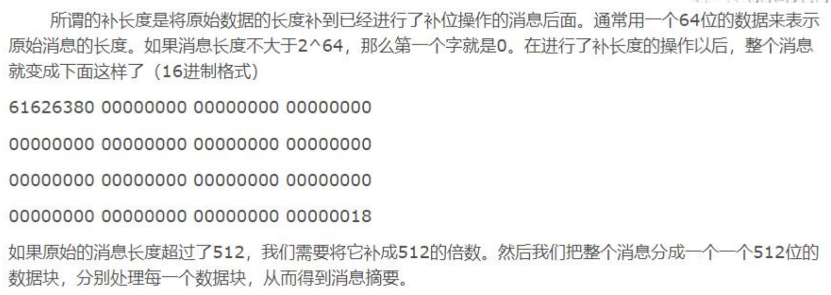
```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

现在数据长度为 448 了

<2>步骤二：长度填充(Append length)

之前说了，需要满足 $b + paddingLength \% 512 = 448$， 那么对于最后一个分块，就还剩 $512 - 448 = 64$ *bit* 这剩下的 *64bit* 存放的是原始消息的长度，也就是 b。「**SHA-1**」最多可以处理明文长度小于等于 *2^64 bit* 的数据。
经过上面两个步骤的处理，最终得到的处理后的数据如下图所示：




所谓的补长度是将原始数据的长度补到已经进行了补位操作的消息后面。通常用一个64位的数据来表示原始消息的长度。如果消息长度不大于2^64，那么第一个字就是0。在进行了补长度的操作以后，整个消息就变成下面这样了（16进制格式）

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

如果原始的消息长度超过了512，我们需要将它补成512的倍数。然后我们把整个消息分成一个一个512位的数据块，分别处理每一个数据块，从而得到消息摘要。

<3>计算摘要

对于每个明文分组的摘要生成过程如下：

（1）将512位的明文分组划分为16个子明文分组，每个子明文分组为32位。

（2）申请5个32位的链接变量，记为A、B、C、D、E。

（3）16份子明文分组扩展为80份。

（4）80份子明文分组进行4轮运算。

（5）链接变量与初始链接变量进行求和运算。

（6）链接变量作为下一个明文分组的输入重复进行以上操作。

（7）最后，5个链接变量里面的数据就是SHA1摘要。

经过添加位数处理的明文，其长度正好为 512 位的整数倍，然后按 512 位的长度进行分组（block），可以划分成 L 份明文分组，我们用 Y_0，Y_1，……Y_{L-1} 表示这些明文分组。对于每一个明文分组，都要重复反复的处理，这些与 MD5 是相同的。

对于 512 位的明文分组，SHA1 将其再分成 16 份子明文分组（sub-block），每份子明文分组为 32 位，我们使用 M[k]（k= 0, 1,……15）来表示这 16 份子明文分组。之后还要将这 16 份子明文分组扩充到 80 份子明文分组，我们记为 W[k]（k= 0, 1,……79），扩充的方法如下。

$$W_t = M_t,\ 当 0 \le t \le 15$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1,\ 当 16 \le t \le 79$$

SHA1 有 4 轮运算，每一轮包括 20 个步骤（一共 80 步），最后产生 160 位摘要，这 160 位摘要存放在 5 个 32 位的链接变量中，分别标记为 A、B、C、D、E。这 5 个链接变量的初始值以 16 进制位表示如下。

A=0x67452301

B=0xEFCDAB89

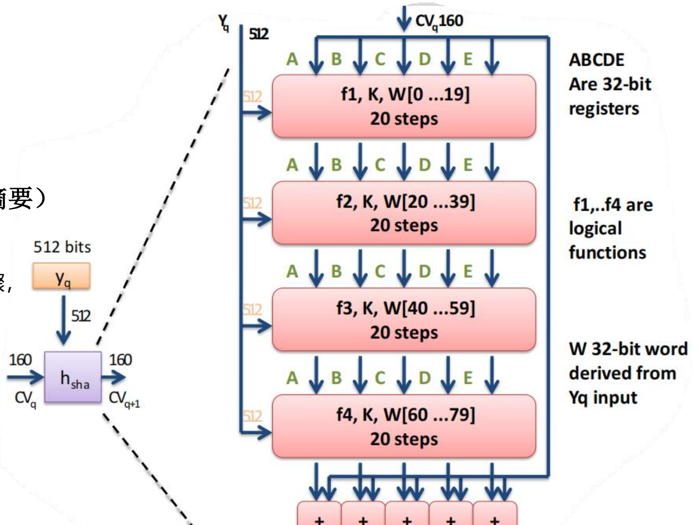
C=0x98BADCFE

D=0x10325476

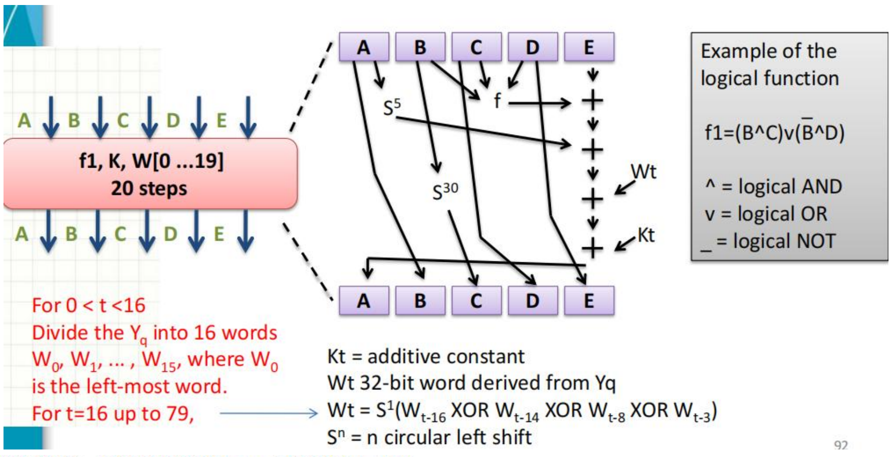
E=0xC3D2E1F0

<4>SHA1 的 4 轮运算（也属于计算摘要）

SHA1 有 4 轮运算，每一轮包括 20 个步骤，



一共 80 步，当第 1 轮运算中的第 1 步骤
开始处理时，A、B、C、D、E 五个链接
变量中的值先赋值到另外 5 个记录单元
A′，B′，C′，D′，E′中。这 5 个值将保留，
用于在第 4 轮的最后一个步骤完成之后与
链接变量 A，B，C，D，E 进行求和操作。



For 0 < t <16
Divide the Y_q into 16 words W_0, W_1, ... , W_{15}, where W_0 is the left-most word.
For t=16 up to 79,

Kt = additive constant
Wt 32-bit word derived from Yq
$Wt = S^1(W_{t-16} \text{ XOR } W_{t-14} \text{ XOR } W_{t-8} \text{ XOR } W_{t-3})$
S^n = n circular left shift

SHA1的4轮运算，共80个步骤使用同一个操作程序，如下:

$A,B,C,D,E \leftarrow [(A<<<5)+ f_t(B,C,D)+E+W_t+K_t],A,(B<<<30),C,D$

其中 $f_t(B,C,D)$为逻辑函数，W_t为子明文分组W[t]，K_t为固定常数。这个操作程序的意义为:

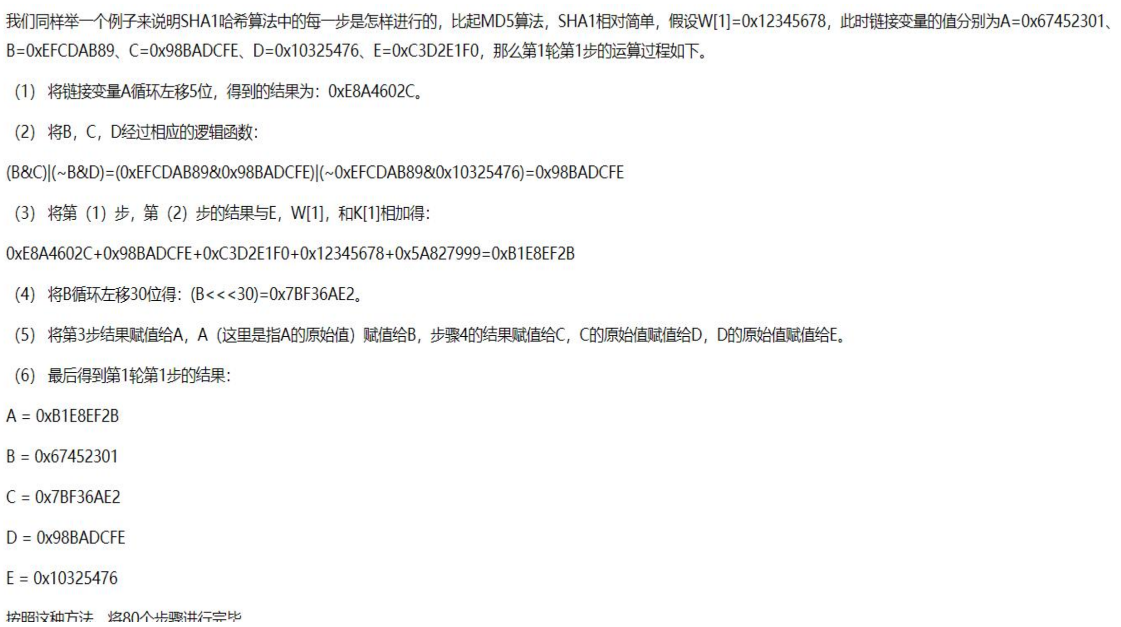
- 将$[(A<<<5)+ f_t(B,C,D)+E+W_t+K_t]$的结果赋值给链接变量A；*A<<<5* 就是二进制位运算循环左移 *5* 位（*S^5*）
- 将链接变量A初始值赋值给链接变量B；
- 将链接变量B初始值循环左移30位赋值给链接变量C；
- 将链接变量C初始值赋值给链接变量D；
- 将链接变量D初始值赋值给链接变量E。

$f_t(B,C,D)$如图所示

1	0≤t≤19	$f_t(B,C,D)=(B·C)V(\sim B·D)$	3	40≤t≤59	$f_t(B,C,D)=(B·C)V(B·D)V(C·D)$
2	20≤t≤39	$f_t(B,C,D)=B \oplus C \oplus D$	4	60≤t≤79	$f_t(B,C,D)=B \oplus C \oplus D$

在操作程序中需要使用固定常数 Ki（i= 0，1，2，……79），Ki 的取值如表所示

1	0≤t≤19	K_t=5A827999	3	40≤t≤59	K_t=8F188CDC
2	20≤t≤39	K_t=6ED9EBA1	4	60≤t≤79	K_t=CA62C1D6



我们同样举一个例子来说明SHA1哈希算法中的每一步是怎样进行的，比起MD5算法，SHA1相对简单，假设W[1]=0x12345678，此时链接变量的值分别为A=0x67452301、B=0xEFCDAB89、C=0x98BADCFE、D=0x10325476、E=0xC3D2E1F0，那么第1轮第1步的运算过程如下。

（1）将链接变量A循环左移5位，得到的结果为：0xE8A4602C。

（2）将B，C，D经过相应的逻辑函数：

(B&C)|(~B&D)=(0xEFCDAB89&0x98BADCFE)|(~0xEFCDAB89&0x10325476)=0x98BADCFE

（3）将第（1）步，第（2）步的结果与E，W[1]，和K[1]相加得：

0xE8A4602C+0x98BADCFE+0xC3D2E1F0+0x12345678+0x5A827999=0xB1E8EF2B

（4）将B循环左移30位得：(B<<<30)=0x7BF36AE2。

（5）将第3步结果赋值给A，A（这里是指A的原始值）赋值给B，步骤4的结果赋值给C，C的原始值赋值给D，D的原始值赋值给E。

（6）最后得到第1轮第1步的结果：

A = 0xB1E8EF2B

B = 0x67452301

C = 0x7BF36AE2

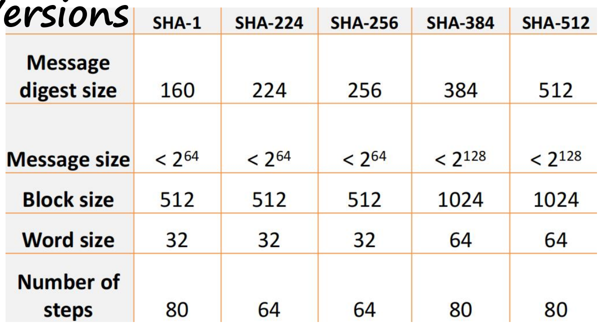
D = 0x98BADCFE

E = 0x10325476

按照这种方法，将80个步骤进行完毕

第四轮最后一个步骤的 A，B，C，D，E 输出，将分别与记录单元 A′，B′，C′，D′，E′中的数值求和运算。其结果将作为输入成为下一个 512 位明文分组的链接变量 A，B，C，D，E，当最后一个明文分组计算完成以后，A，B，C，D，E 中的数据就是最后散列函数值。

(4)SHA Versions



	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

(5)SHA-512

Processing message in 1024-bit blocks (Block size)

Consists of 80 rounds (Number of steps)

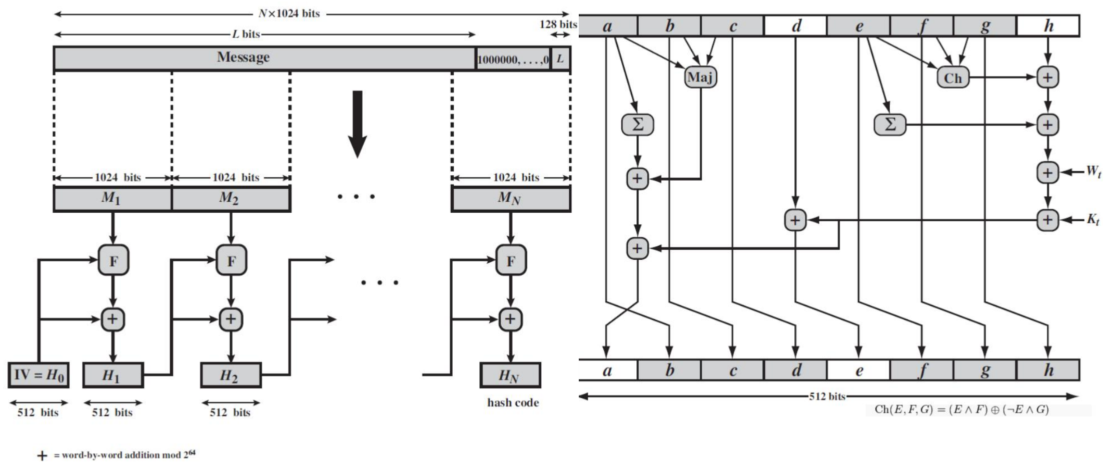
  Updating a 512-bit buffer

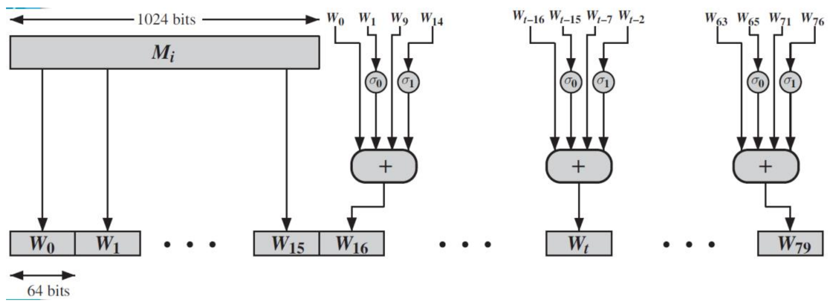
  Using a 64-bit value Wt derived from the current message block

  使用来自当前消息块的 64 位值 Wt

  and a round constant based on cube root of first 80 prime numbers

  以及一个基于前 80 个素数的立方根的圆常数




1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{65} W_{71} W_{76}
σ_0 σ_1
+
W_0 W_1 W_{15} W_{16} W_t W_{79}
64 bits

(6)SHA-3

<1>背景

• SHA-1 not yet "broken”

  – Similar to broken MD5 & SHA-0

  – Hence, considered insecure 不安全

• SHA-2 (esp. SHA-512) seems secure

  – Same structure and mathematical operations as predecessors so have concern.

  与前辈相同的结构和数学运算也值得关注。

• SHA-3 Cryptographic hash function to complement SHA-2.

  SHA-3 加密哈希函数，以补充 SHA-2。

<2>Requirements

Replace SHA-2 with SHA-3 in any use

  Same hash sizes 相同哈希大小 SHA-3 代替 SHA-2

Preserve the online nature of SHA-2 保持 SHA-2 的在线性质

  Must process small blocks (512 / 1024 bits) 必须处理小块

Evaluation criteria 评定标准

  Security close to theoretical max for hash sizes 安全接近哈希大小的最大理论值

  Cost in time & memory 在时间和内存上的成本

  Characteristics: such as flexibility & simplicity 特点：如灵活性和简单性

九. Other hashes

	MD5	SHA-1	SHA-2	SHA-3	RIPEMD-160
Digest length	128 bits	160 bits	512 bits	512 bits	16 bits
Basic Unit of processing	512 bits	512 bits	1024 bits	576 bits	512 bits
Number of steps	64	80	80	24	160
Maximum message length	∞ (4 rounds of 20)	2^{64} -1 (4 rounds of 20)	2^{128} -1 (4 rounds of 20)	Unlimited	∞ (5 rounds of 16)

(1)Other Hash Function Uses 其他哈希函数功能

<1>To create a one-way password file 创建单向密码文件

  – Store hash of password not actual password 存储密码的散列，而不是实际的密码

<2>For intrusion detection and virus detection 用于入侵检测和病毒检测

  – Keep & check hash of files on system 保留和检查系统上文件的散列

<3>Pseudorandom function (PRF) or pseudorandom number generator (PRNG)

伪随机函数（PRF）或伪随机数发生器（PRNG）

(2)Attacks on Hash Functions

• Both brute-force attacks and cryptanalysis 同时进行蛮力攻击和密码分析
• A preimage or second preimage attack 预像或第二次预像攻击

– Find *y* such that *H(y)* equals a given hash value 查找 y，使 H (y)等于一个给定的哈希值

– For an *m*-bit hash value, the adversary would have to try on average 2^m–1 values of y to find

one that generates a given hash value *h*. 对于一个 m 位哈希值，对手必须尝试平均 2^m-1 个

y 值，才能找到一个生成给定哈希值 h 的值。
• Collision resistance attack

– Find two messages x & *y* with same hash so H(x) = H(y)

找到两个具有相同哈希值的消息 x & y，所以 H (x) = H (y)

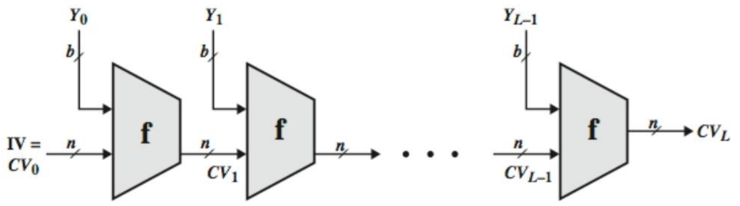
– Birthday Attack: value 2*m/2* determines strength of hash code against brute-force attacks

生日攻击：值 2m/2 决定了针对暴力攻击的哈希代码的强度
– 128-bits inadequate, 160-bits suspect

(3)Hash Function Cryptanalysis

<1>Cryptanalytic attacks exploit some property of the algorithm, so it is faster than exhaustive search 密码分析攻击利用了该算法的某些特性，因此比穷举搜索速度快
<2>Hash functions use iterative structure 哈希函数使用迭代结构
Process message in blocks (incl length) 以块处理消息（包括长度）

(3)Attacks focus on collisions in function *f* **攻击集中于函数 *f* 中的** collision



(4)The Birthday Attack

<1>
两个人相同生日的概率
5 个人里$\frac{(365\times364\times363\times362\times361)}{365^5}=0.97$，3%的概率
140 人就是 99.999999999%
所以通过计算得出，至少有 23 个人才能使两个人有相同生日的概率超过 50%
<2>The birthday attack is based on the birthday paradox 生日攻击是基于生日悖论
– Number of days per year ⇔ number of possible hashes 每年的天数 → 可能的散列数
– How many pairs of modified original/fraudulent messages we need to create to have a 50% chance that they have the same hash. 我们需要创建多少对修改过的原始/欺诈性消息，才能有 50%的机会拥有相同的散列。

– For an *m*-bit hash we need $2^{m/2}$modified messages.

对于 m 位散列，我们需要 2^m/2 个修改的消息

<3>An attacker needs to find a message M’ such that the hash of this message is the same as the hash of the original message M, that is H(M’) = H(M).攻击者需要找到一个消息 M‘，以便此消息的哈希与原始消息 M 的散列相同，即 H（M’）= H (M)。

<4>It is based on the birthday paradox and here how it works:

它是基于生日悖论和这里它是如何工作的：

– Alice is ready to ‘sign’ a message by encrypting it, using her private-key, to generate the m-bit hash code of the message.

Alice 准备通过使用她的私钥加密一个消息来“签名”它，以生成消息的 m 位哈希码。

– Eve generatces 2^m/2 variations of the message, all of which convey the same meaning.

Eve 概括了 2^m/2 的变体，所有这些都传达了相同的意思。

– Eve also generates an equal number of messages all of which are variations of the fraudulent message that Eve wants to substitute for the original one. Eve 还生成了相同数量的信息，所有这些信息都是 Eve 想要替代原始信息的欺诈性信息的变体。

– Eve searches in the two sets of messages for a pair that produce the same hash code. i.e. [hash(variation of original) = hash(variation of fraudulent)] Eve 在这两组消息中搜索产生相同哈希代码的一对消息。即[散希（原始的变化）=散希（欺诈的变化）]

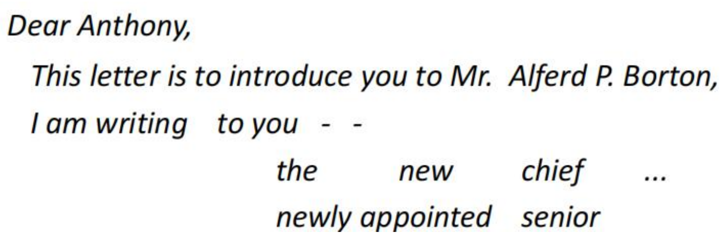
– The probability success is greater than 0.5. 成功的概率大于 0.5。

–For the 64-bit hash code, Eve needs to create 2^32 variations of the message (232 = 4,294,967,296) 对于 64 位的哈希代码，Eve 需要创建消息的 2^32 个变体

–Eve presents the valid variation of the original message to Alice for signature. After Alice signs the hash, the hash is attached to the fraudulent version of the message and that is it.

Eve 将原始信息的有效变化呈现给爱丽丝以进行签名。在 Alice 签署哈希之后，哈希就会被附加到消息的欺诈版本上，仅此而已

–For Eve it is not difficult to create variations of the message. Here is an example (32 versions of simple message). 对于 EVE 来说，创造信息的变化并不难。



Dear Anthony,

This letter is to introduce you to Mr. Alferd P. Borton,

I am writing to you - -

the new chief ...

newly appointed senior

十. Message Authentication Code (MAC)

(1)介绍

• Message Authentication involves generating a small block of data (the MAC) that is appended to the message. 消息身份验证涉及生成附加到消息中的一小块数据（MAC）。

• The MAC depends on the contents of the message and a secret key MAC(m) = $f(kAB,m)$,

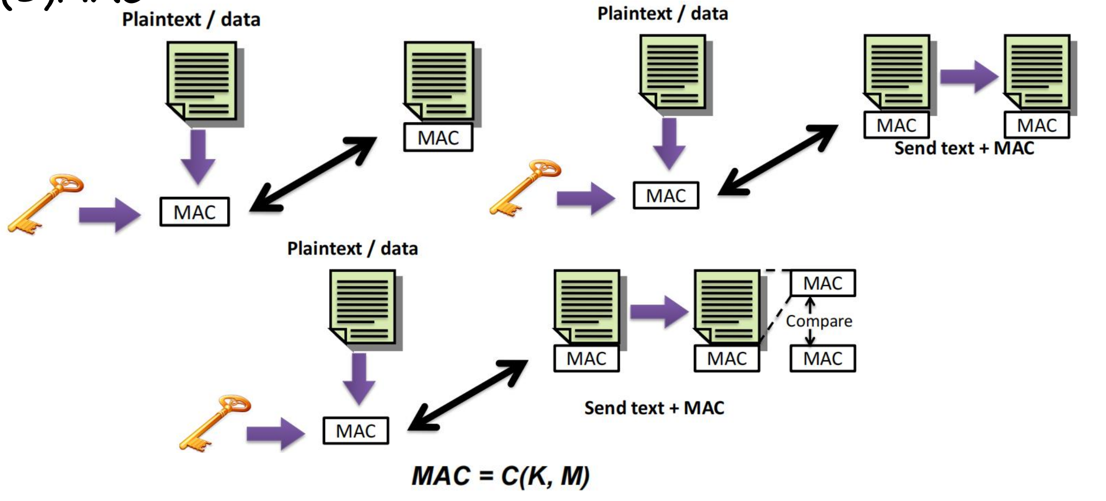
where m is the plaintext, kAB is the shared key between users A and B.

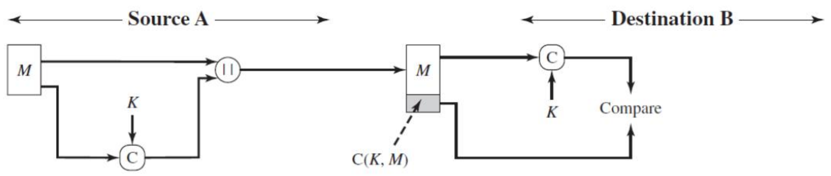
• The message plus MAC are transmitted to the recipient.消息加上 MAC 被传输给收件人
• The recipient, which knows the secret key, calculates the MAC of the message and compares it with the MAC just received. 知道密钥的接收方计算消息的 MAC，并与刚刚收到的 MAC 比较。
• Any changes in the message are noticed if the MAC calculated by the receiver and the MAC from the sender mismatch.如果接收方计算出的 MAC 和发送方不匹配的 MAC，就需要注意到消息中的任何变化。

(2)Authentication

• If the received MAC matches the calculated MAC, then the receiver is assured that the message 如果接收到的 MAC 与计算的 MAC 匹配，那么接收方需保证消息
  – has not been altered (integrity of the message). 未被更改（信息的完整性）
  – is from the alleged sender (originator identity validation). 来自发送者（发起者身份验证）
  – If the message has a sequence number, the receiver can check it has a proper sequence. 如果消息有一个序列号，接收方可以检查它是否有一个正确的序列。

(3)MAC





(4)Security of MACs

• As with block ciphers, there are:

<1>Brute-force attacks exploiting 蛮力攻击

• Strong collision resistance hash have cost 2^m/2
• MACs with known message-MAC pairs
  – Can either attack keyspace (key search) or MAC 可以攻击密钥空间（密钥搜索）或 MAC
  – At least 128-bit MAC is needed for security 至少需要 128 位 MAC

<2>Cryptanalytic attacks exploit structure 密码分析攻击利用结构

• More variety of MACs so harder to generalise about cryptanalysis
更多种类的 MAC 更难概括密码分析

(5)An authentication standard

• From the National Bureau of Standards (NBS)
– Use DEA to encrypt the message 使用 DEA 对消息进行加密
– The MAC is the last bits of Ciphertext (16-32 bits is typical) MAC 是密文的最后位(16-32 位）

• Is similar to encryption, ***but*** it doesn't need to be reversible (we don't need decryption)

类似于加密，但它不需要是可逆的（我们不需要解密）

十一. HMAC

(1)介绍

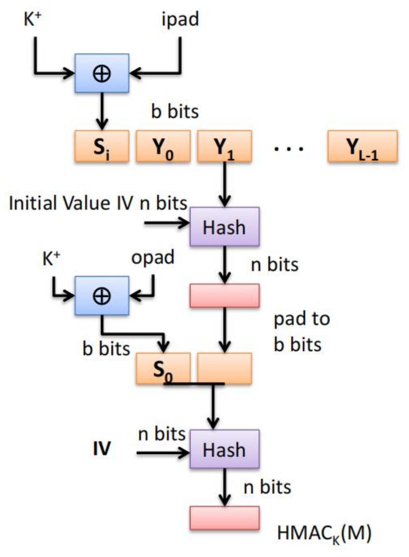
• HMAC is a MAC derived from a cryptographic hash code, that is, it incorporates a secret key into an existing hash algorithm. HMAC 是一个由加密 hash 代码衍生出来的 MAC，也就是说，它将一个密钥合并到现有的哈希算法中。
Why?
1. Cryptographic hash functions execute faster (software) than conventional encryption algorithm (e.g. DES) 加密哈希函数（软件）执行速度超过传统加密算法（如 DES)
2. Code for cryptographic hash functions is widely available.加密哈希函数的代码已被广泛使用。

(2)Design Objectives

– To use available hash functions. 使用可用的哈希函数

– To allow for easy ***replaceability*** of the embedded hash function.

允许嵌入式哈希函数的容易替换。
– To preserve the original performance of the hash function.保持哈希函数的原始性能
– To use and handle keys in a simple way. 以一种简单的方式使用和处理 key。
– To have a well-understood cryptographic analysis of the strength of the authentication mechanism. 对身份验证机制的强度进行一个有良好理解的密码分析。



K+ = secret key, padded with zeros so is b-bits long.

Yi = *i*-th block of the message M

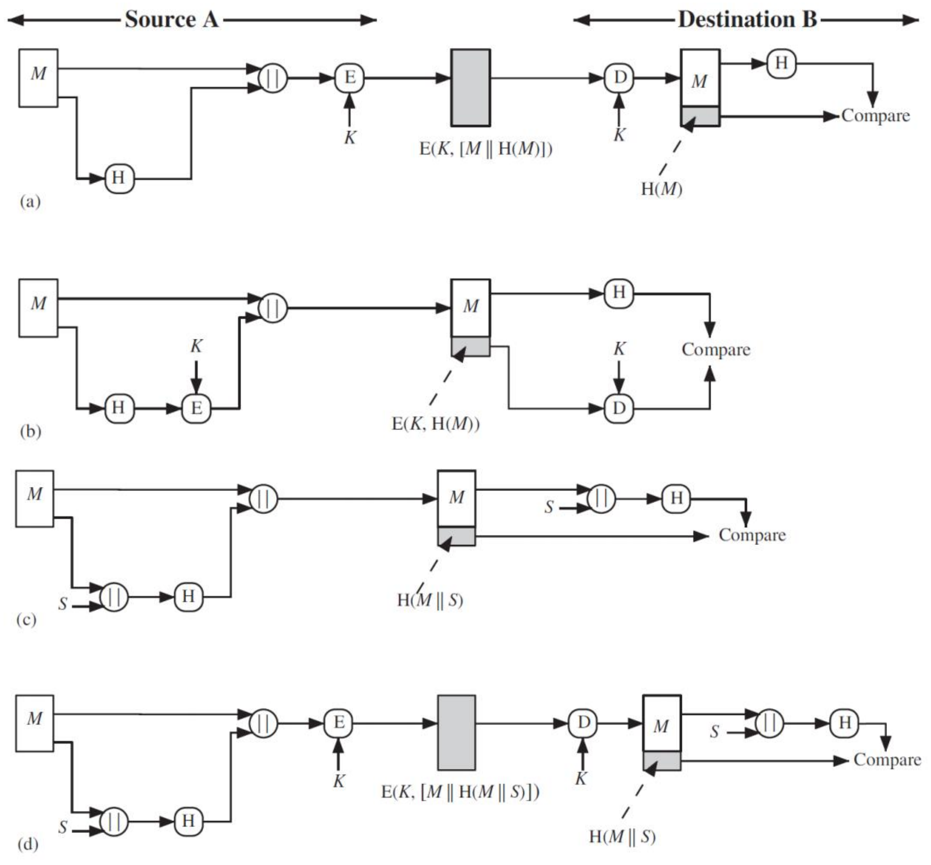
L = number of blocks in M

B = number of bits in a block

n = length of the hash code

ipad = padding
opad = padding

(3)Hash Functions & Message Authentication



(4)Digital Signature

Digital signature scheme provides:

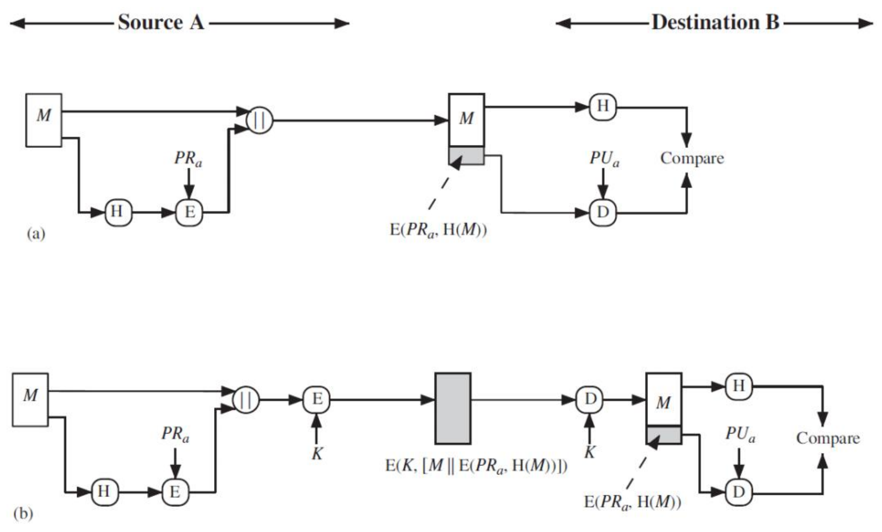
<1>***Data origin authentication of the signer***. 签名者的数据来源认证 A digital signature validates the underlying data in the sense that assurance is provided about the integrity of the data and the identity of the signer. 数字签名验证底层数据的意义是，提供了对数据的完整性和签名者的身份的保证

<2>***Non-repudiation***. A digital signature can be stored by anyone who receives it as evidence. 不可否认。任何收到一个数字签名的人都可以存储该数字签名作为证据。

<3>A digital signature on some data will need to be computed from:
一些数据上的数字签名将需要从

- *The data*.
- *A secret parameter known only by the signer*.一个只有签名者才知道的秘密参数

(5)Hash Functions & Digital Signatures



Summary

- Public-Key Encryption
  - – Introduction
  - – *Diffie-Hellman* Key Exchange
  - – RSA algorithm
  - – Certificate
  - – Data Integrity/Message Authentication
  - – Digital signature

  - – Attacks
    - Man-in-the-middle, Birthday/hash attacks.