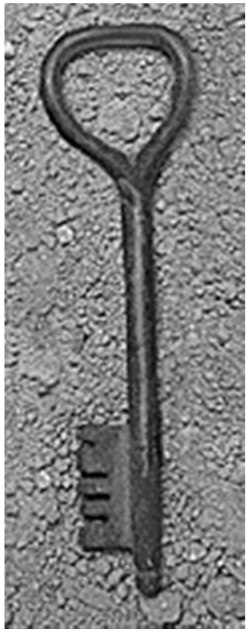
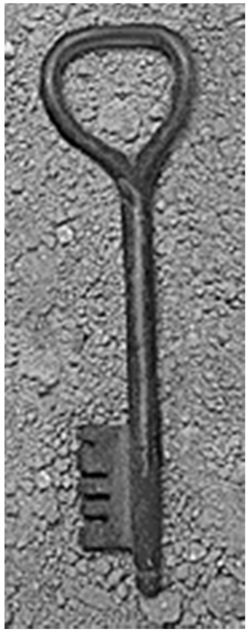
A Brief Introduction To Cryptography

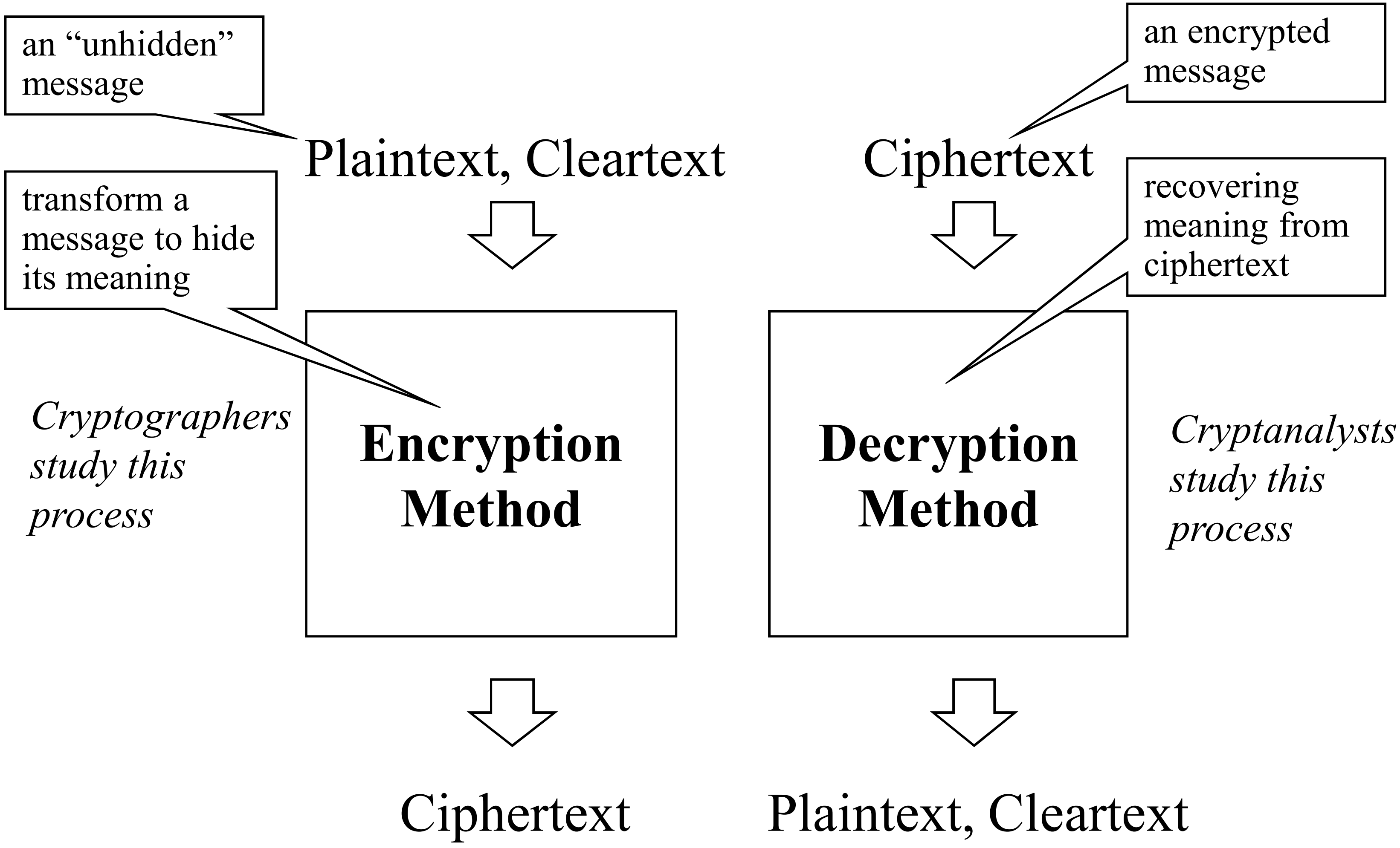


Before We Get Started …

- See for more information:
  - Kevin Mitnick’s *The Art of Deception*
  - Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C*
  - (Many of the examples I use later are loosely based on that book)
  - Both can be downloaded from Internet



Terminology



an "unhidden" message

Plaintext, Cleartext

transform a message to hide its meaning

Cryptographers study this process

Encryption Method

Ciphertext

an encrypted message

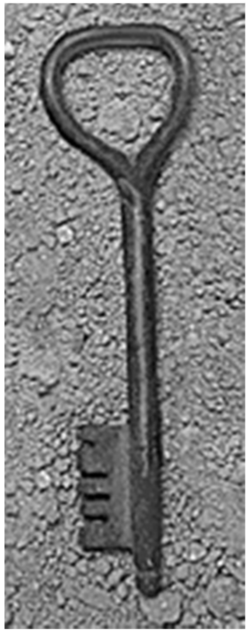
Ciphertext

recovering meaning from ciphertext

Decryption Method

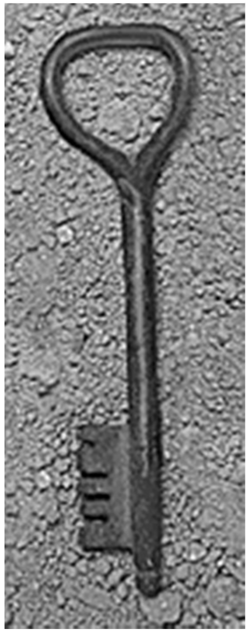
Cryptanalysts study this process

Plaintext, Cleartext



Usual Mathematical Symbols

P	Plaintext
C	Ciphertext
E	Encryption function
D	Decryption function
E(P) = C	encrypting plaintext yields ciphertext
D(C) = P	decrypting ciphertext yields plaintext
D(E(P)) = P	decrypting encrypted plaintext yields plaintext



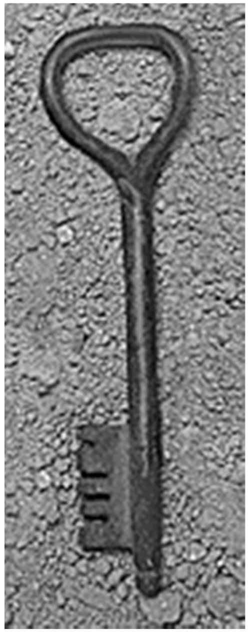
Restricted Algorithm

The security of a restricted algorithm requires keeping the algorithm secret.

Encryption algorithm:
Multiply the plaintext number by 2

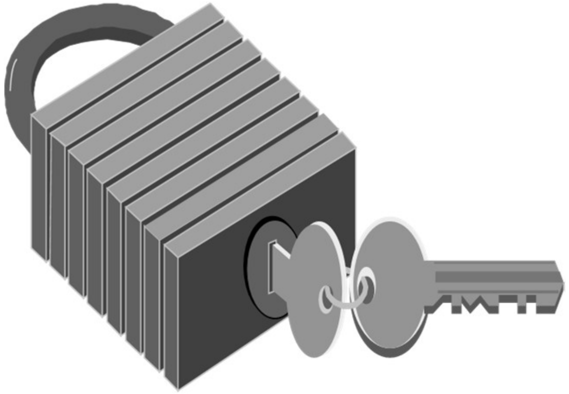
Decryption algorithm:
Divide the ciphertext number by 2

plaintext = SECRET = 19 5 3 18 5 20
ciphertext = 38 10 6 36 10 40



Key-Based Algorithm

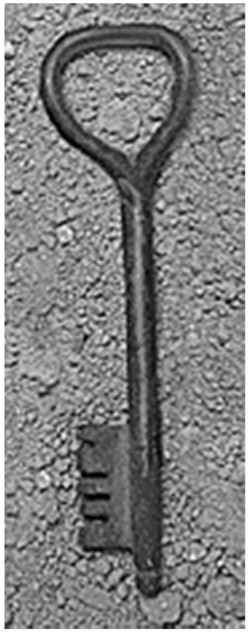
The security of key-based algorithms is based on the secrecy of the the key(s)



Encryption algorithm:
Multiply the plaintext number by 2 and add key.

Decryption algorithm:
Subtract key and divide the ciphertext number by 2.

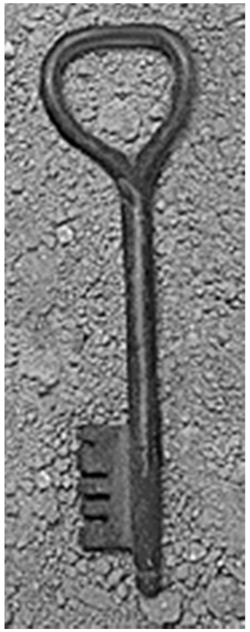
plaintext = SECRET = 19 5 3 18 5 20
key = 3
ciphertext = 41 13 9 39 13 43



Attacks

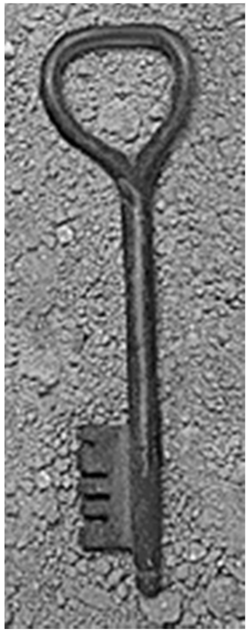
- Types of attacks
  - ciphertext only
    - attackers only have some cipertexts in his hand
  - known plaintext
    - attackers can find some plaintext/cipertext pairs
  - chosen plaintext
    - attackers can generate ciphertext for any plaintext he selected
- Real attacks generally don’t break cryptography!
  - *Don’t pick the lock, tunnel into the vault*
  - *Ex. Brute Dictionary-based password guessing Attack*

Cryptography Algorithms

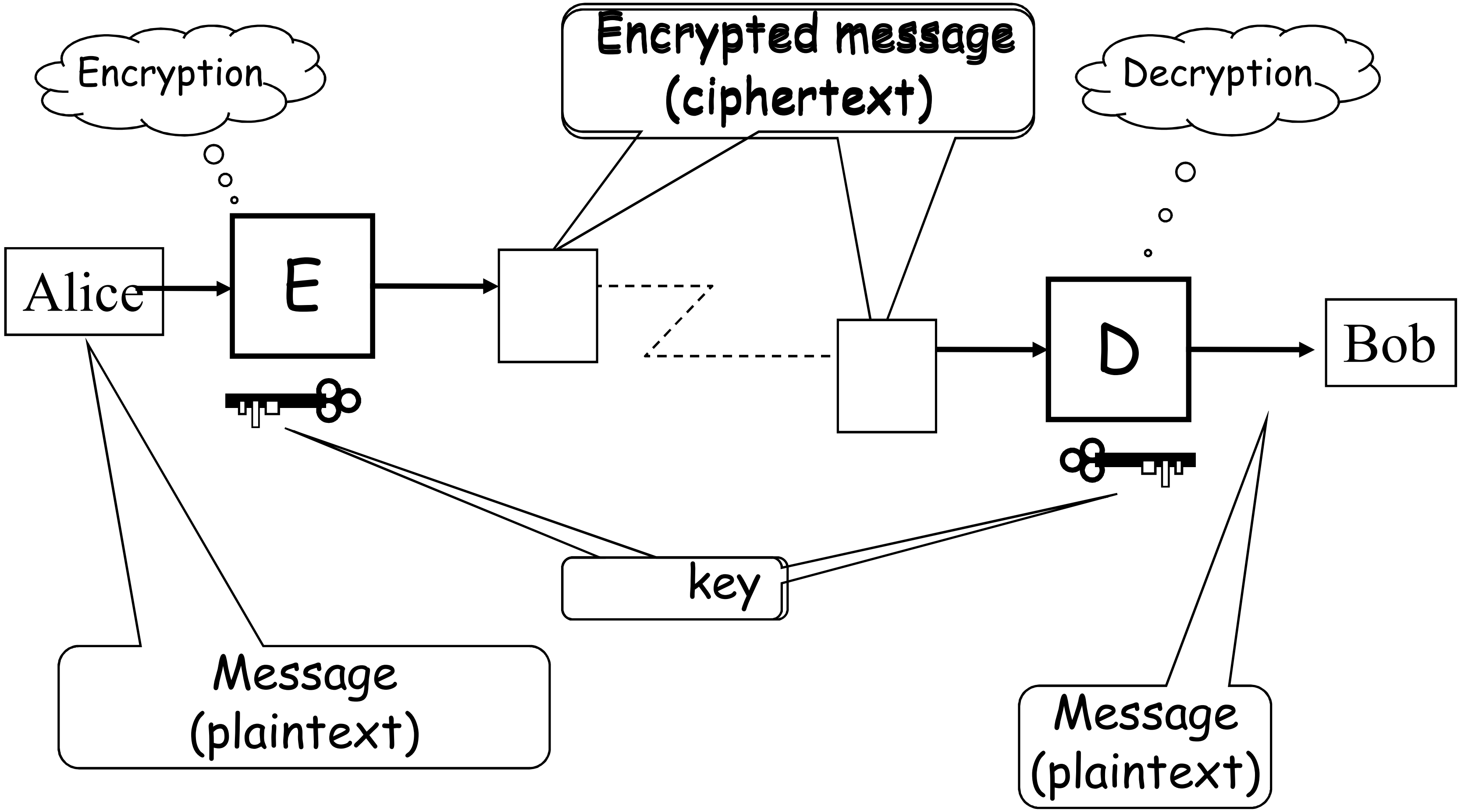


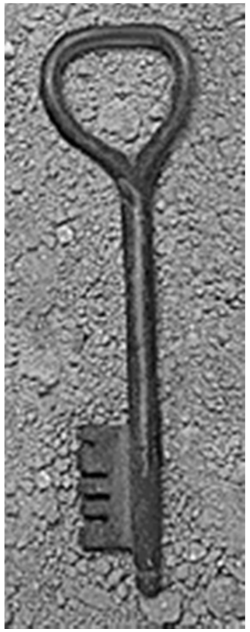
Major Types of Algorithms

- ♦ Secret-Key (Symmetric) Cryptography
- ♦ Public Key (Asymmetric) Cryptography
- ♦ Digital Signatures & Hash Algorithms



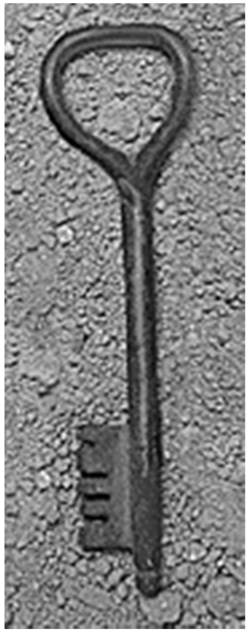
Secret-Key (Symmetric) Cryptography





Concepts

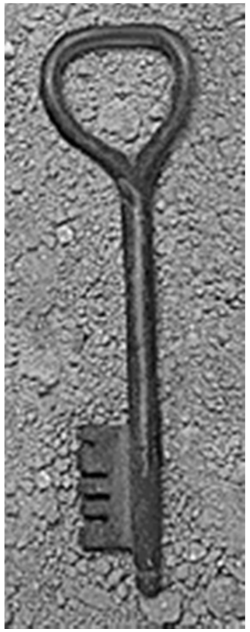
- a private key cipher is composed of two algorithms
  - encryption algorithm E
  - decryption algorithm D
- the same key K is used for encryption & decryption
- K has to be distributed beforehand



Secret-Key (Symmetric) Cryptography: Uses

对称秘钥，替换算法

- Prevent eavesdropping
  - Must be secure channel for key exchange
- Secure storage
  - I have to remember my key
- Authentication
  - Challenge/response
  - *Be careful*
- Integrity Check
  - Checksum on the message; Encrypt the checksum

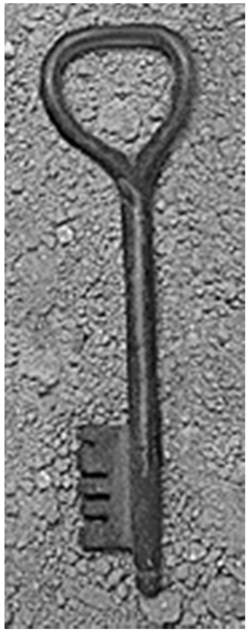


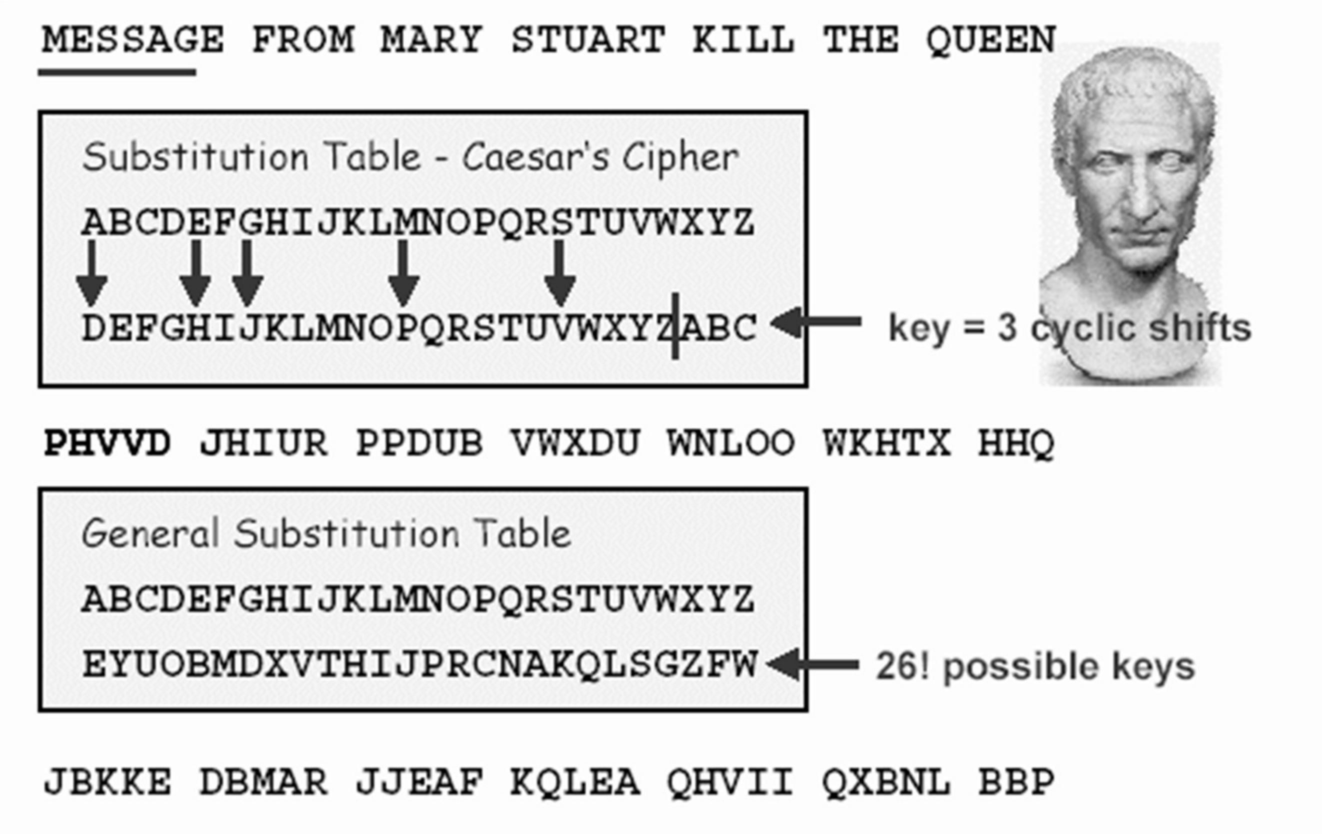
Classic ciphers

- ♦ substitution ciphers
- ♦ transposition (permutation) ciphers
- ♦ product ciphers
  - – using both
    - substitution, and
    - transposition

加密算法是打鸡蛋。可能要打很多次。

Substitution Cipher





MESSAGE FROM MARY STUART KILL THE QUEEN

Substitution Table - Caesar's Cipher

ABCDEFGHIJKLMNOPQRSTUVWXYZ

DEFGHIJKLMNOPQRSTUVWXYZABC

PHVVD JHIUR PPDUB VWXDU WNLOO WKHTX HHQ

General Substitution Table

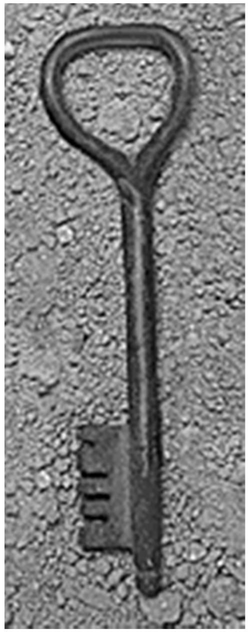
ABCDEFGHIJKLMNOPQRSTUVWXYZ

EYUOBMDXVTHIJPRCNAKQLSGZFW

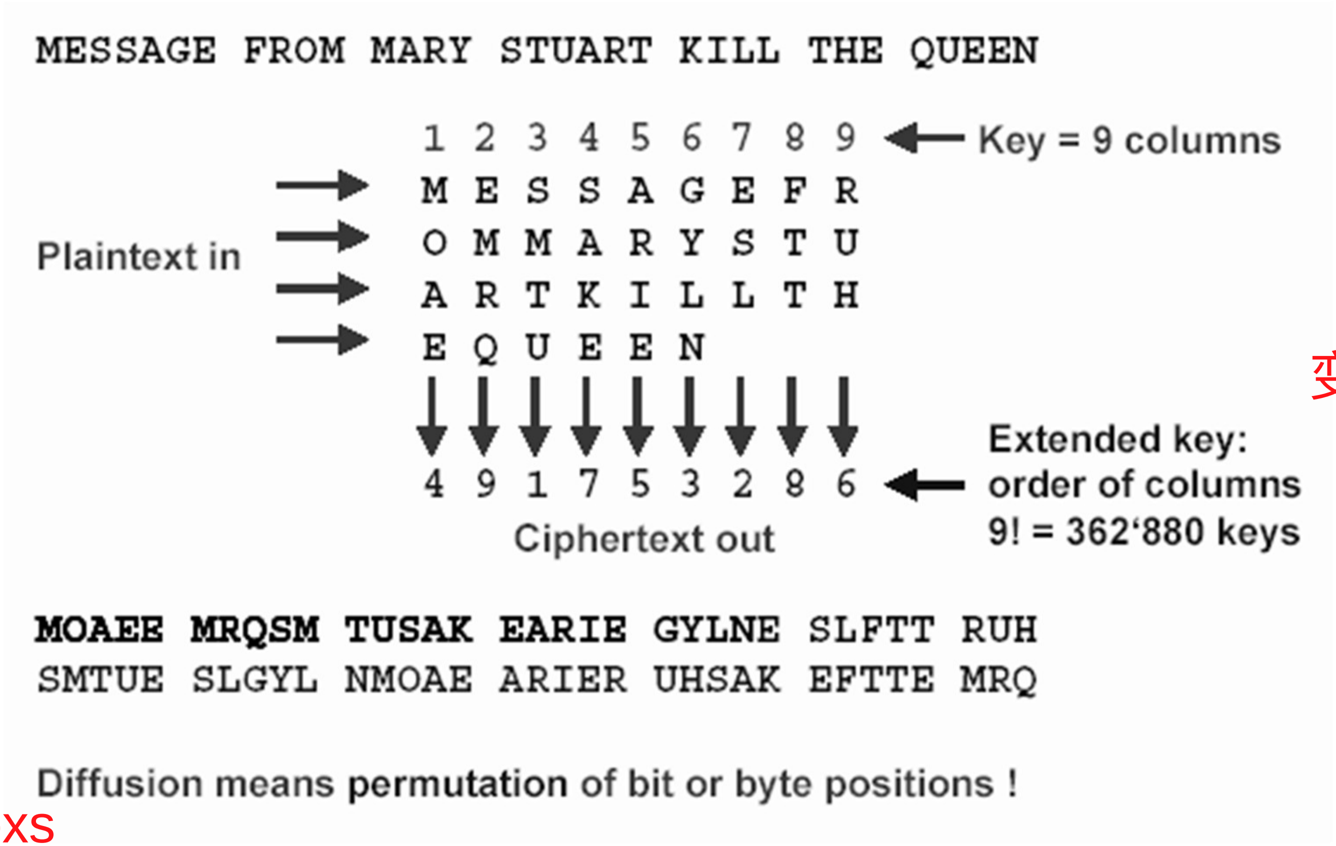
JBKKE DBMAR JJEAF KQLEA QHVII QXBNL BBP

Courtesy:
Andreas
Steffen

- Modern substitution ciphers take in N bits and substitute N bits using lookup table: called S-Boxes



Transposition cipher



MESSAGE FROM MARY STUART KILL THE QUEEN

Key = 9 columns

Plaintext in

1	2	3	4	5	6	7	8	9
M	E	S	S	A	G	E	F	R
O	M	M	A	R	Y	S	T	U
A	R	T	K	I	L	L	T	H
E	Q	U	E	E	N			
4	9	1	7	5	3	2	8	6

Ciphertext out

Extended key:
order of columns
9! = 362‘880 keys

MOAEE MRQSM TUSAK EARIE GYLNE SLFTT RUH

SMTUE SLGYL NMOAE ARIER UHSAK EFTTE MRQ

Diffusion means permutation of bit or byte positions !

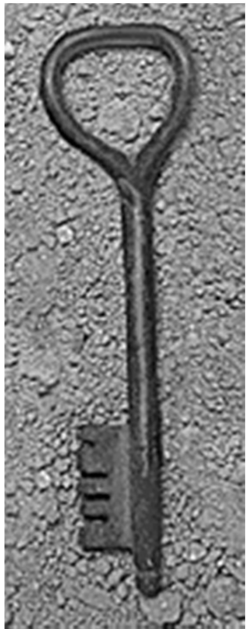
变换算法

扰动算法，只是
打乱了次序。

Courtesy:
Andreas
Steffen

替换算法 X-Boxs
**？？ P-Boxs

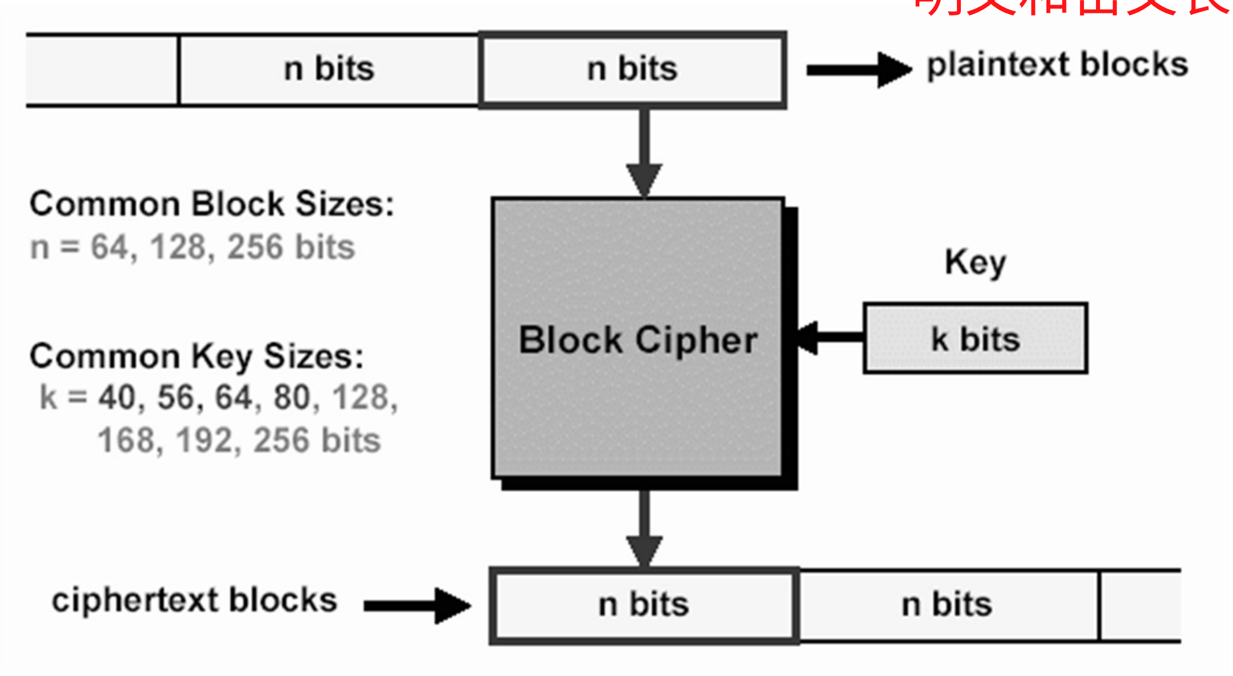
- modern Transposition ciphers take in N bits and permute using lookup table : called P-Boxes



Block Cipher 块密码，输入长度是固定的。方块大小固定。

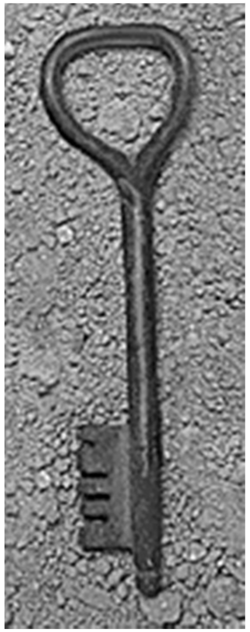
- Divide input bit stream into n-bit sections, encrypt only that section, no dependency/history between sections

明文和密文长度相同。



Courtesy: Andreas Steffen

- In a good block cipher, each output bit is a function of all n input bits and all k key bits



Example: Data Encryption Standard (DES)

- Background
  - **1972, NIST (National Institute of Standards & Technology) initiates the process (Open Policy, compared with Close Policy of Chinese government)**
  - **1974, Tuchman and Meyers (IBM) invented Luciffer Cipher**
  - **1976, NIST announce DES, estimated 2283 years to crack DES**
- Encodes plaintext in 64-bit chunks using a 64-bit key (56 bits + 8 bits parity)
- Decryption in DES – it's symmetric! Use KA again as input and then the same keys except in reverse order

Example: DES (2)

16轮编码

为了对称，一定是2的倍数。
64个bit中每个bit都有相同的概率会变换。
秘钥的任意一个bit翻转也将影响密文的任意一个bit。

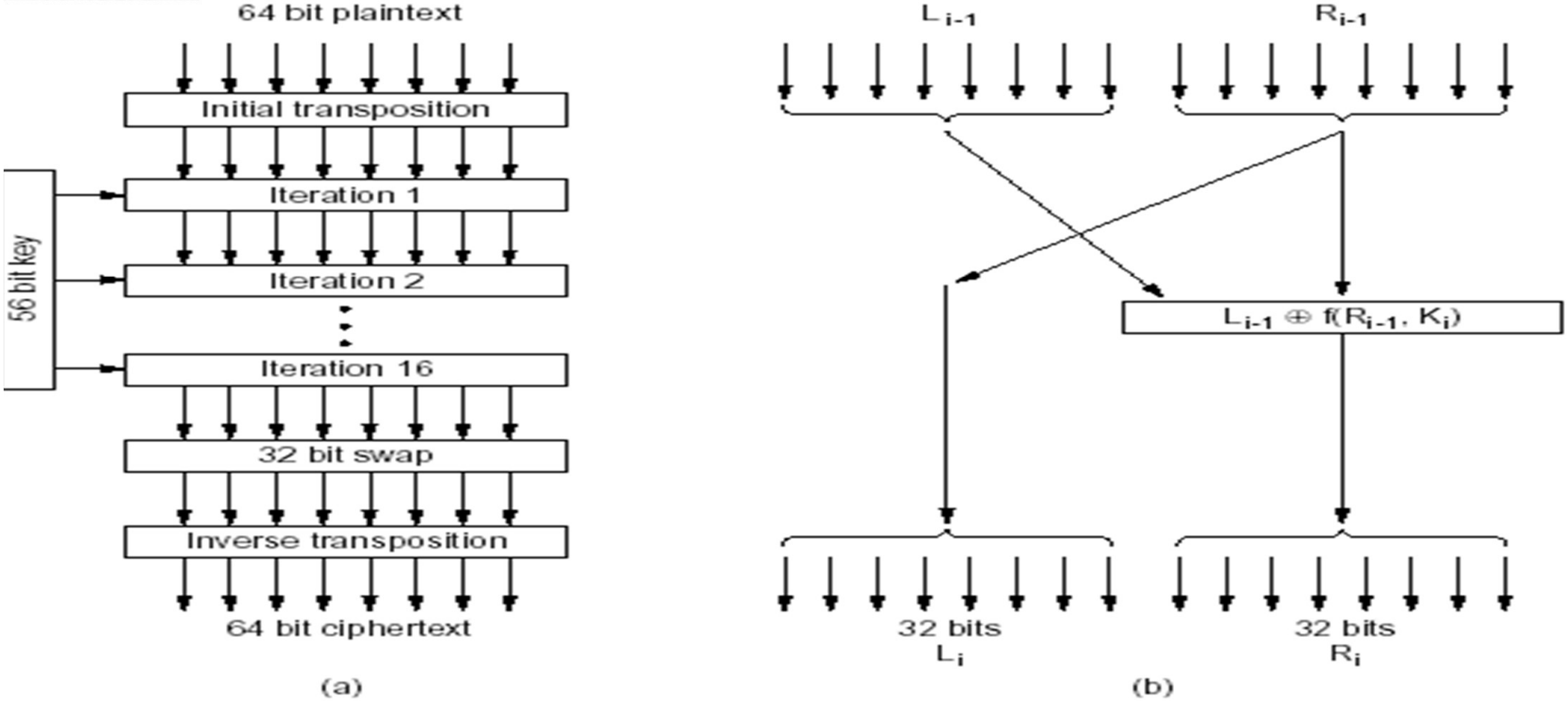
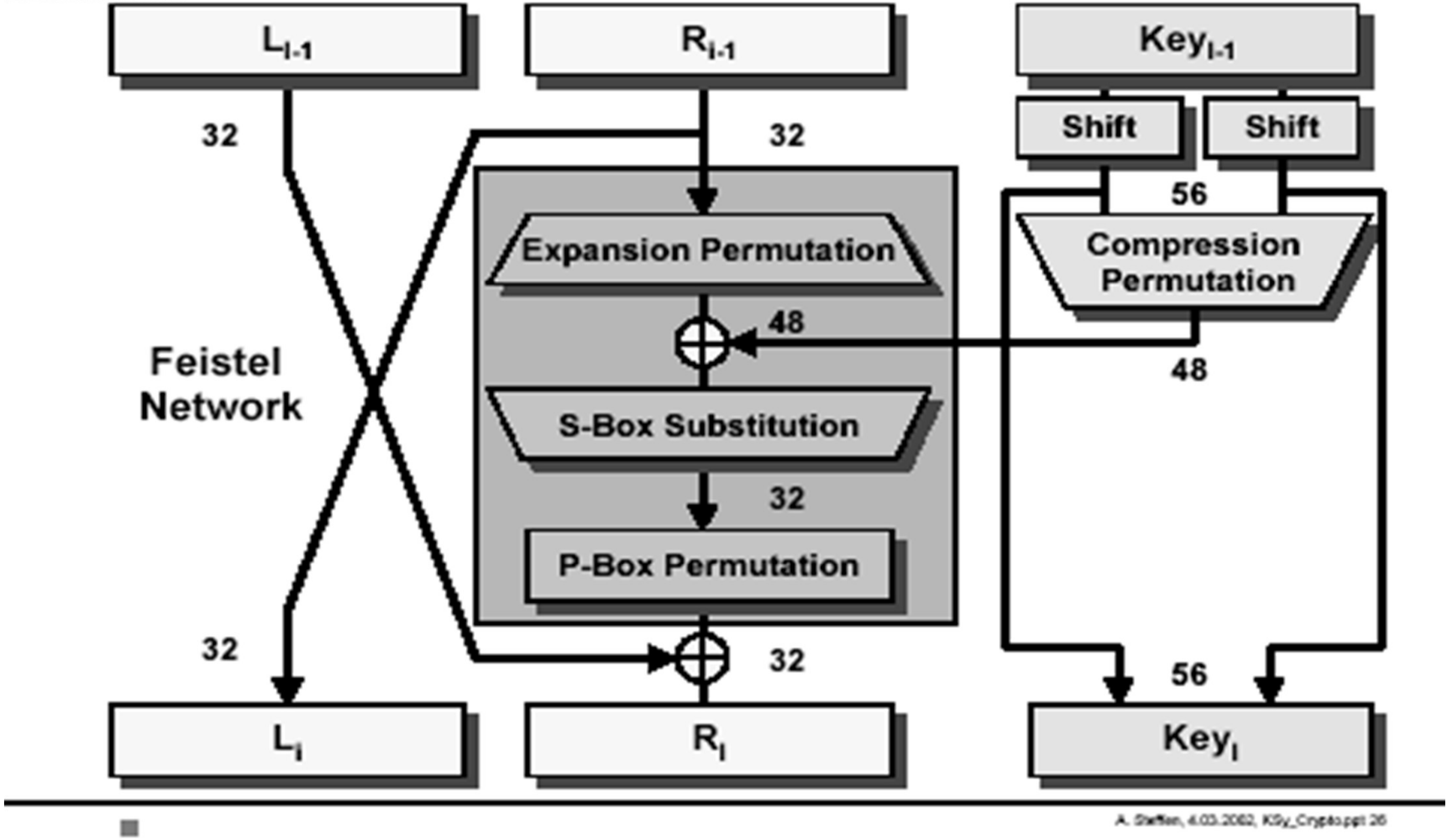


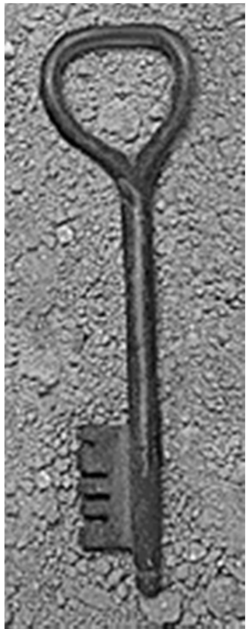
Fig. 7-5. The data encryption standard. (a) General outline. (b) Detail of one iteration.

Example: DES (3)



Cracking DES

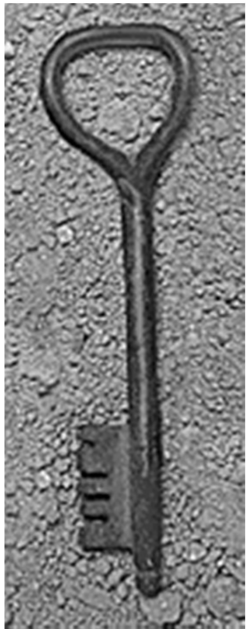
- DES has excellent anti-crack performance. Secure for about 25 years.
  - estimated 2283 years
- Cracked in 1997
  - 秘钥不够长 Key is only 56bits, 2 exp 56 **72,057,584,037,927,936**
  - 计算机能力变强 Computing capability is increasing exponentially
  - 并行破解技术 Parallel attack – exhaustively search key space
  - 1997: Team leaded by Roche Verse using 70000 PCs connected with Internet, 96 days
  - 1998: EFF (Electronic Frontier Foundation) using a specially designed machine ($250,000), 3 days
  - 1999: Using supercomputer, only 22 hours.



Beyond DES, other Block Ciphers

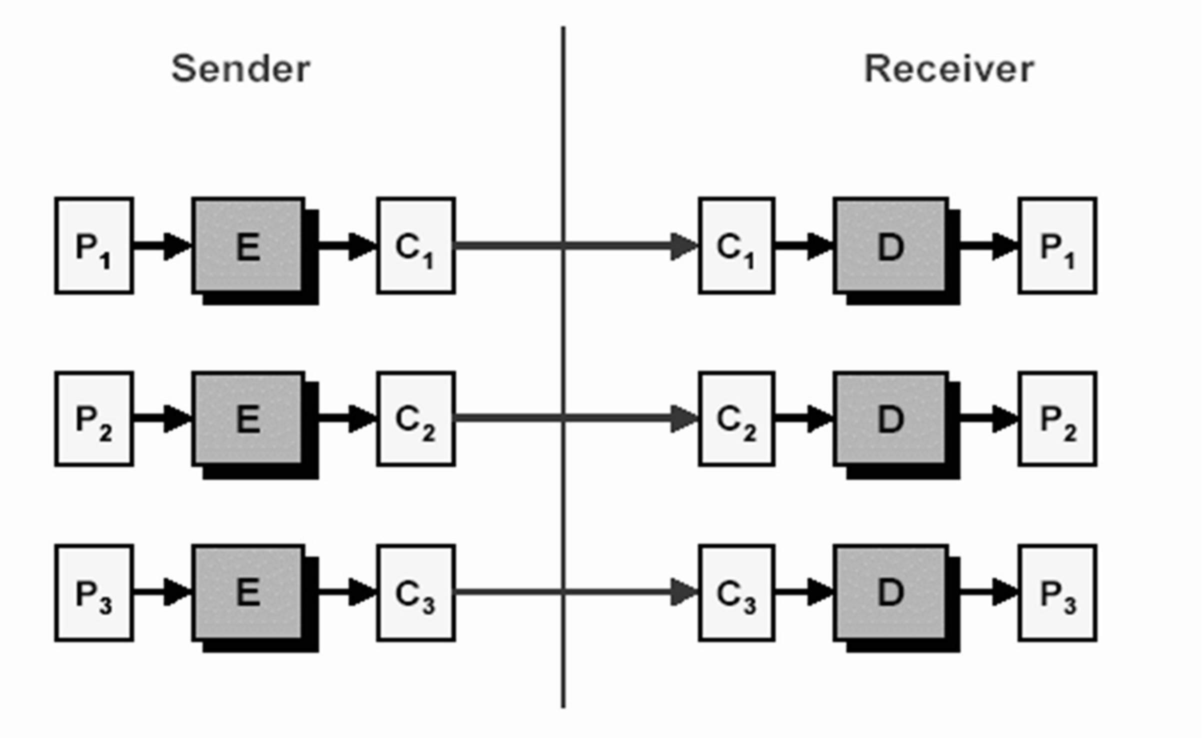
用三次DES算法。 同一算法嵌套使用时，破解的难度随秘钥的增加逐渐趋缓。

- Triple-DES: put the output of DES back as input into DES again with a different key, loop again: 3*56 = 168 bit key
- Advanced Encryption Standard (AES)
  - Initiated by NIST in 1997
  - Requirements:
    - shall be designed so that the key length may be increased as needed.
    - block size n = 128 bits, key size k = 128, 192, 256 bits
  - Candidates: MARS, twofish, RC6, Serpent, Rijndael
  - Winner! (Rijndael)

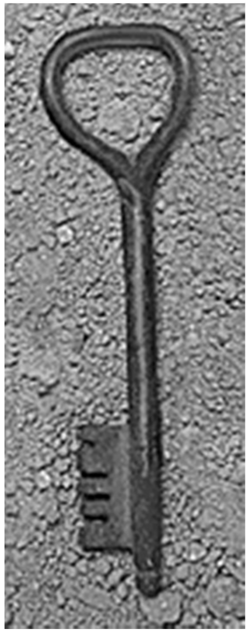


Beyond Block Ciphers, ECB

for block ciphers of a long digital sequence

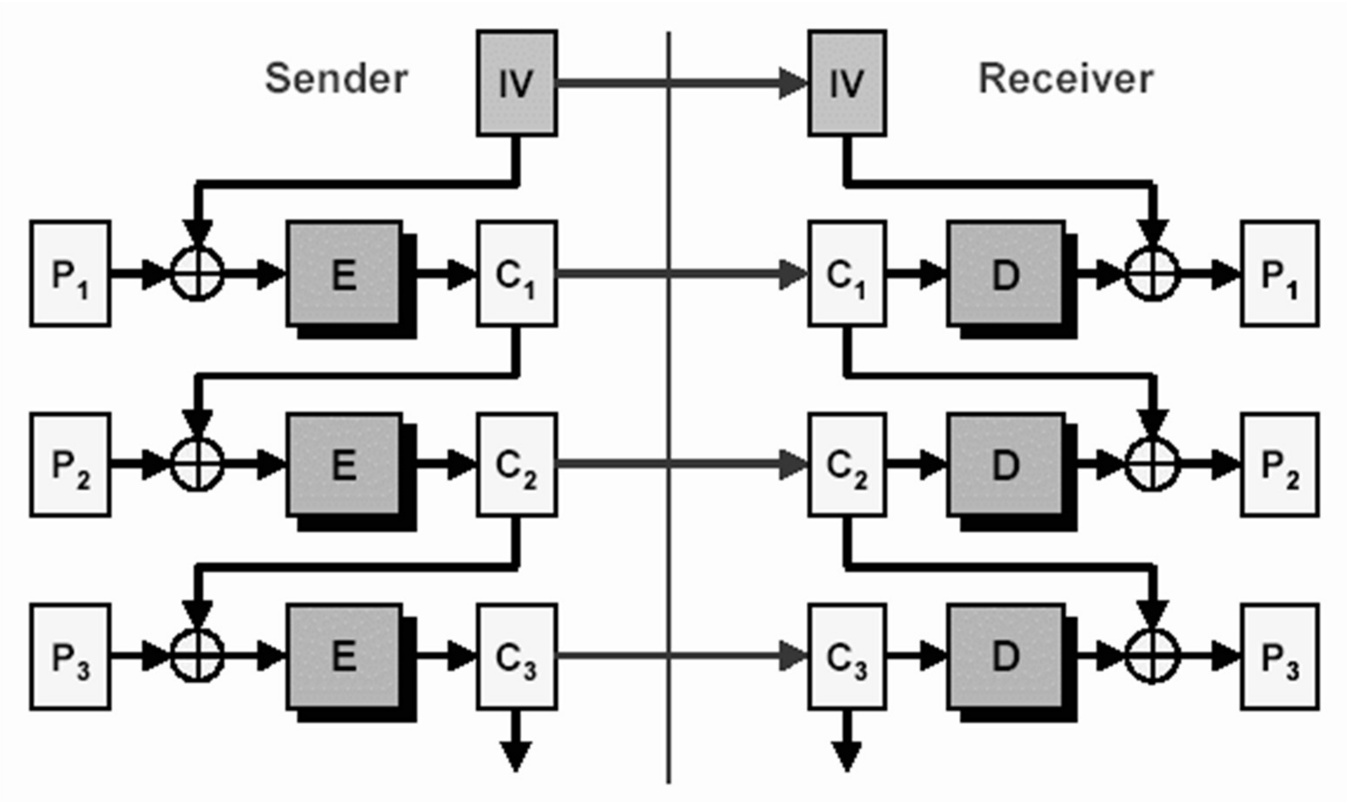


- If an attacker thinks block C_2 corresponds to \$ amount, then substitute another C_k (ciphertext only attacks)
- Attacker can also build a codebook of <C_k, guessed P_k> pairs (chosen plaintext attacks). Replay Attacks?

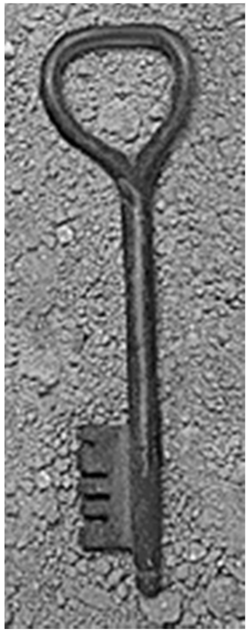


链式块密码

Beyond Block Ciphers, CBC

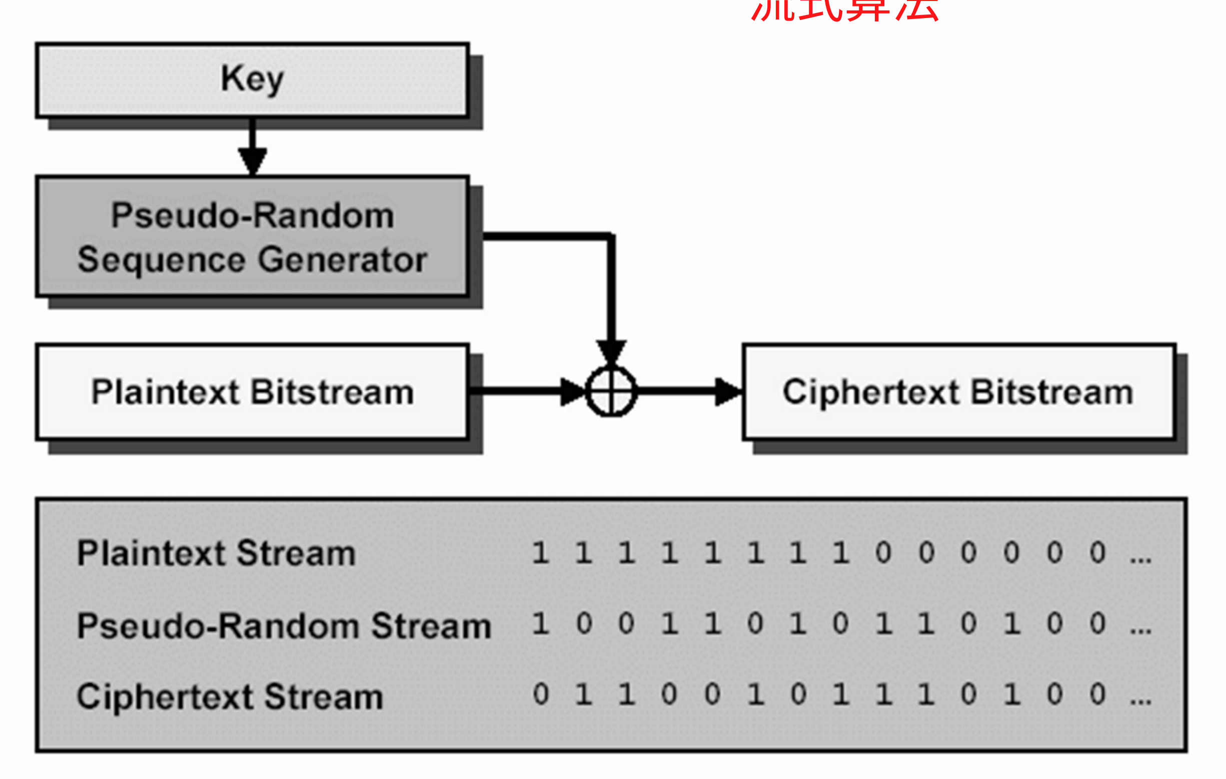


- Inhibits replay attacks and codebook building: identical input plaintext P_i =P_k won't result in same output code due to memory-based chaining
- IV = Initialization Vector – use only once



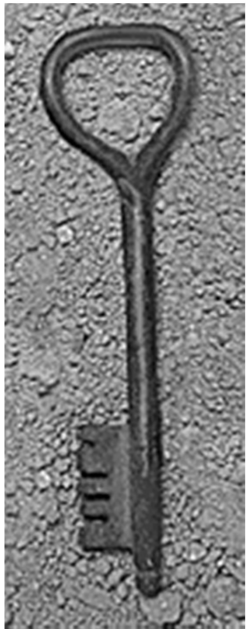
Beyond Block Ciphers, Stream Cipher

流式算法

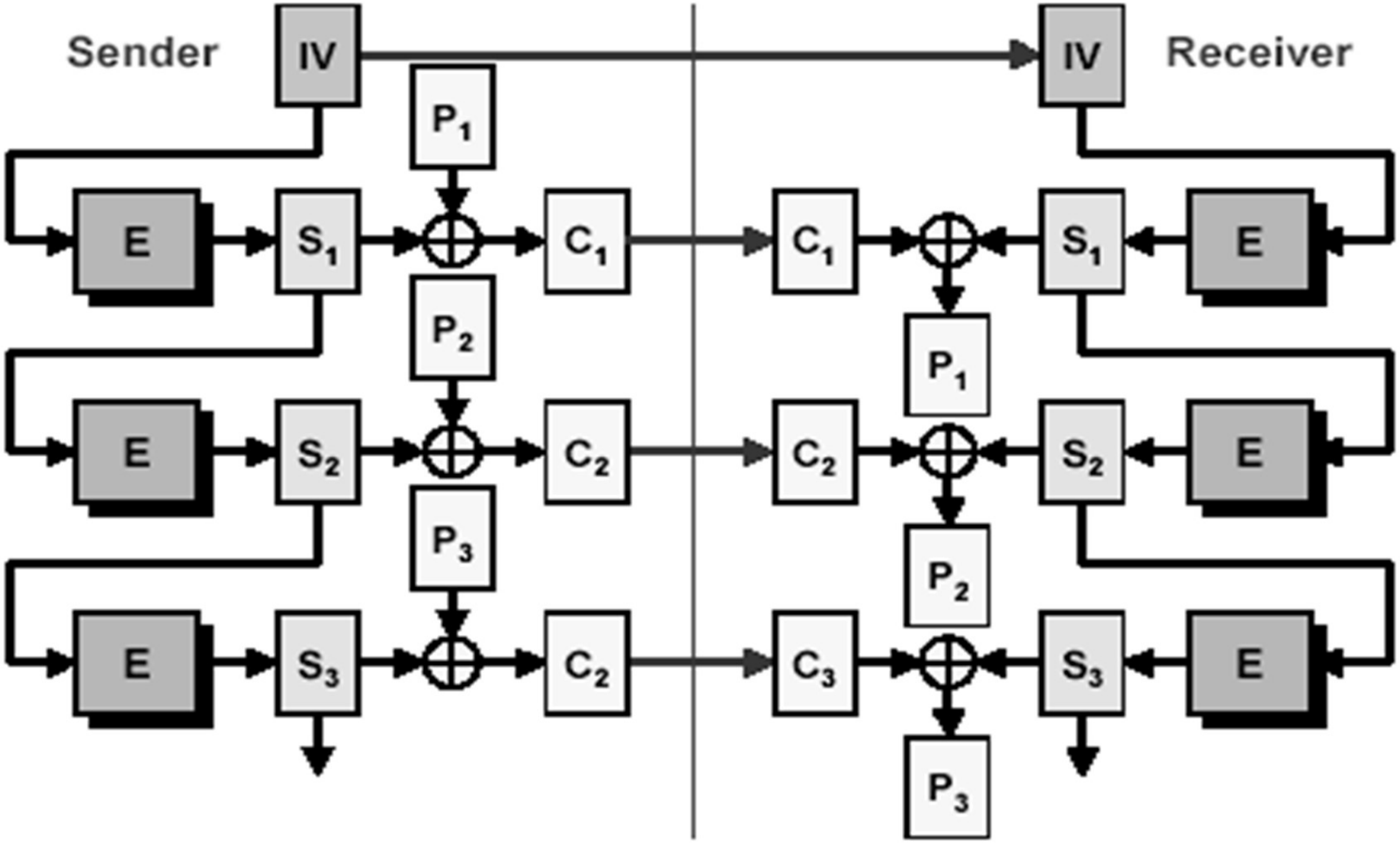


Plaintext Stream	1	1	1	1	1	1	1	1	0	0	0	0	0	0	...
Pseudo-Random Stream	1	0	0	1	1	0	1	0	1	1	0	1	0	0	...
Ciphertext Stream	0	1	1	0	0	1	0	1	1	1	0	1	0	0	...

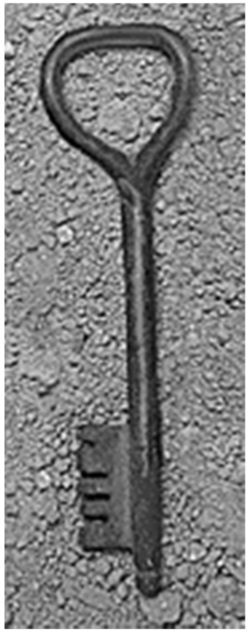
- Rather than divide bit stream into discrete blocks, as block ciphers do, XOR each bit of your plaintext continuous stream with a bit from a pseudo-random sequence
- At receiver, use same symmetric key, XOR again to extract plaintext



Beyond Block Ciphers, Stream Cipher



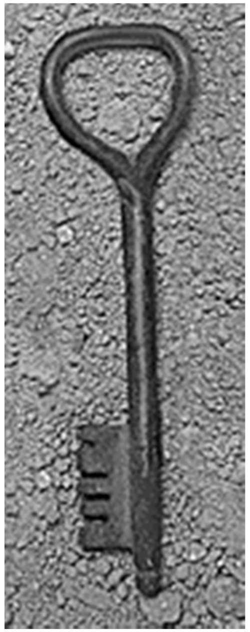
A. Steffen, 4.03.2002, KSy_Crypto.ppt 34



Major Types of Algorithms

- ♦ Secret-Key (Symmetric) Cryptography
- ♦ Public Key (Asymmetric) Cryptography
- ♦ Digital Signatures & Hash Algorithms

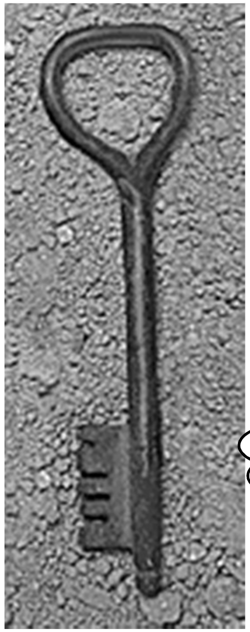
公开秘钥的密码体制，非对称的。

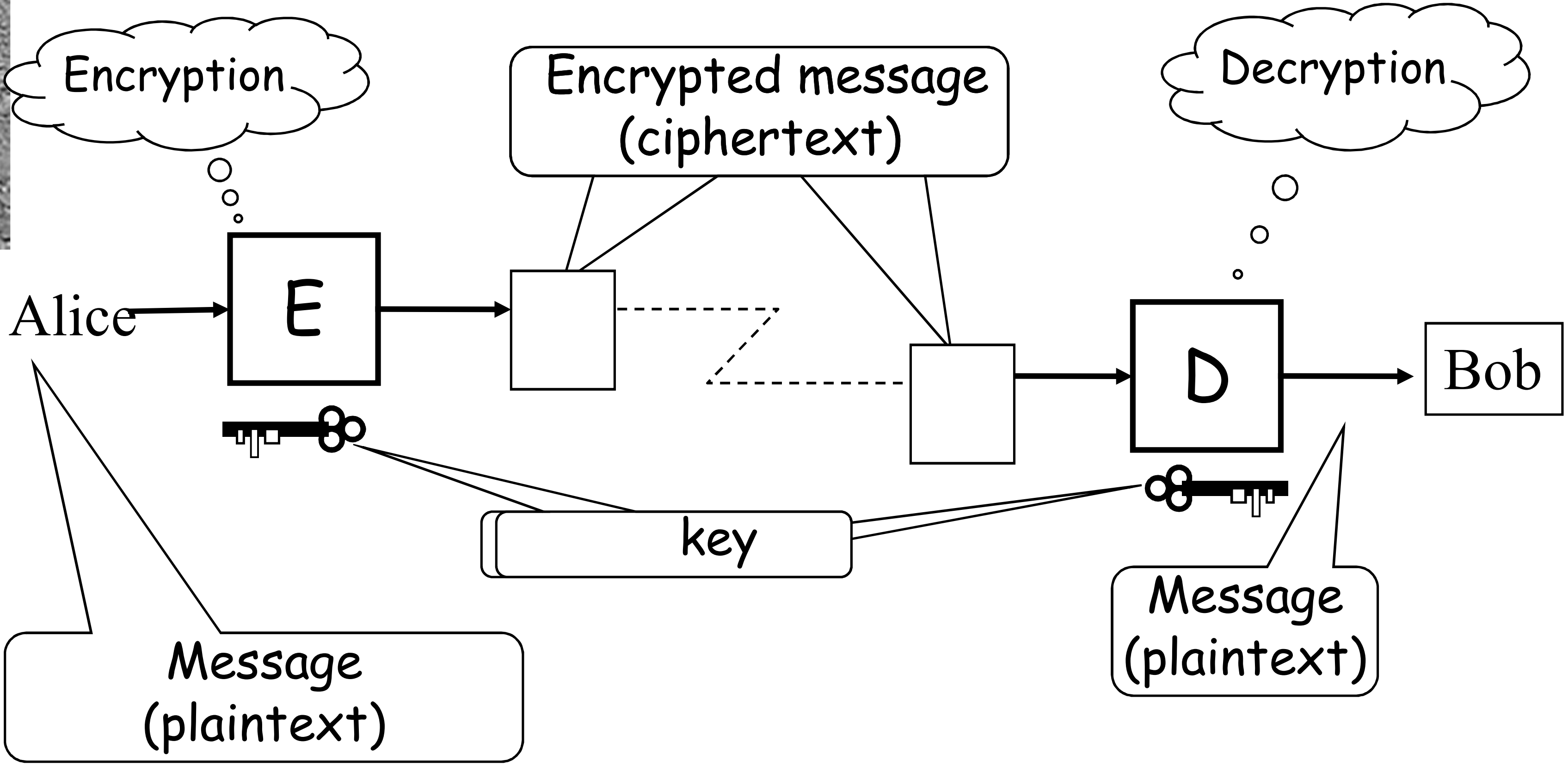


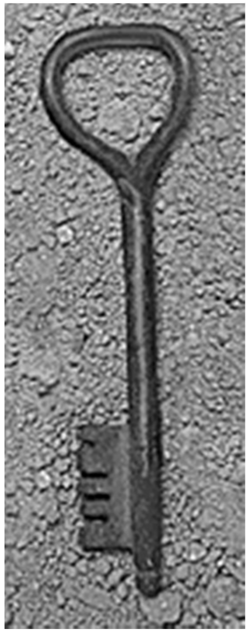
Public Key (Asymmetric) Cryptography

- Why public key cryptography ?
- General principles of public key cryptography
- The RSA public key cryptosystem
- Why RSA is secure?

Review: Private key cipher

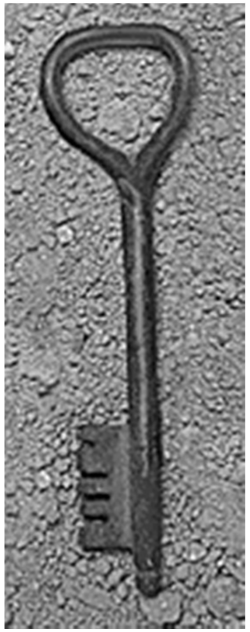






Problems with private key ciphers

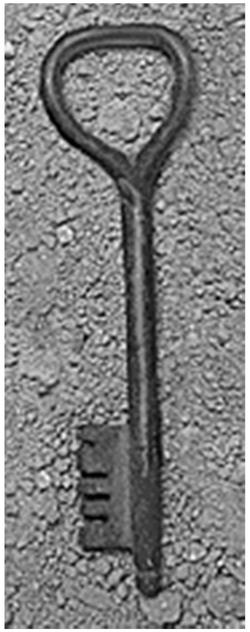
- In order for Alice & Bob to be able to communicate securely using a private key cipher, such as DES, they have to have a shared key in the first place.
  - Question:
    What if they have never met before ?
- Alice needs to keep *100* different keys if she wishes to communicate with *100* different people



Question?

♦ Consider a group of *n* people, each wishing to communicate securely with all other members in the group, by using a private key cipher, say DES.

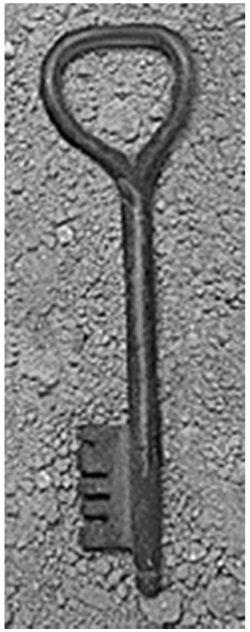
  – How many different secret keys does each member of the group have to keep ?
  – What's the total number of different secret keys that have to be kept by all members of the group ?



Motivation of Diffie & Hellman

- ♦ Is it possible for Alice & Bob, who have no shared secret key, to communicate securely ?
- ♦ This led to the SINGLE MOST IMPORTANT discovery in the history of secure communications:

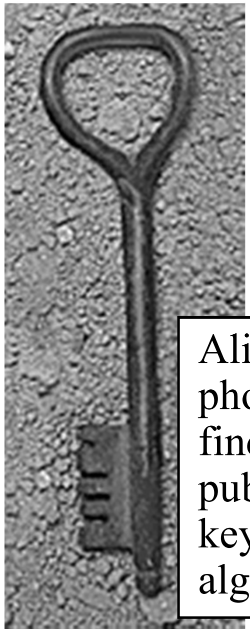
W. Diffie & M. Hellman: *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. IT-22, No.6, Nov. 1976, pp.644-654.

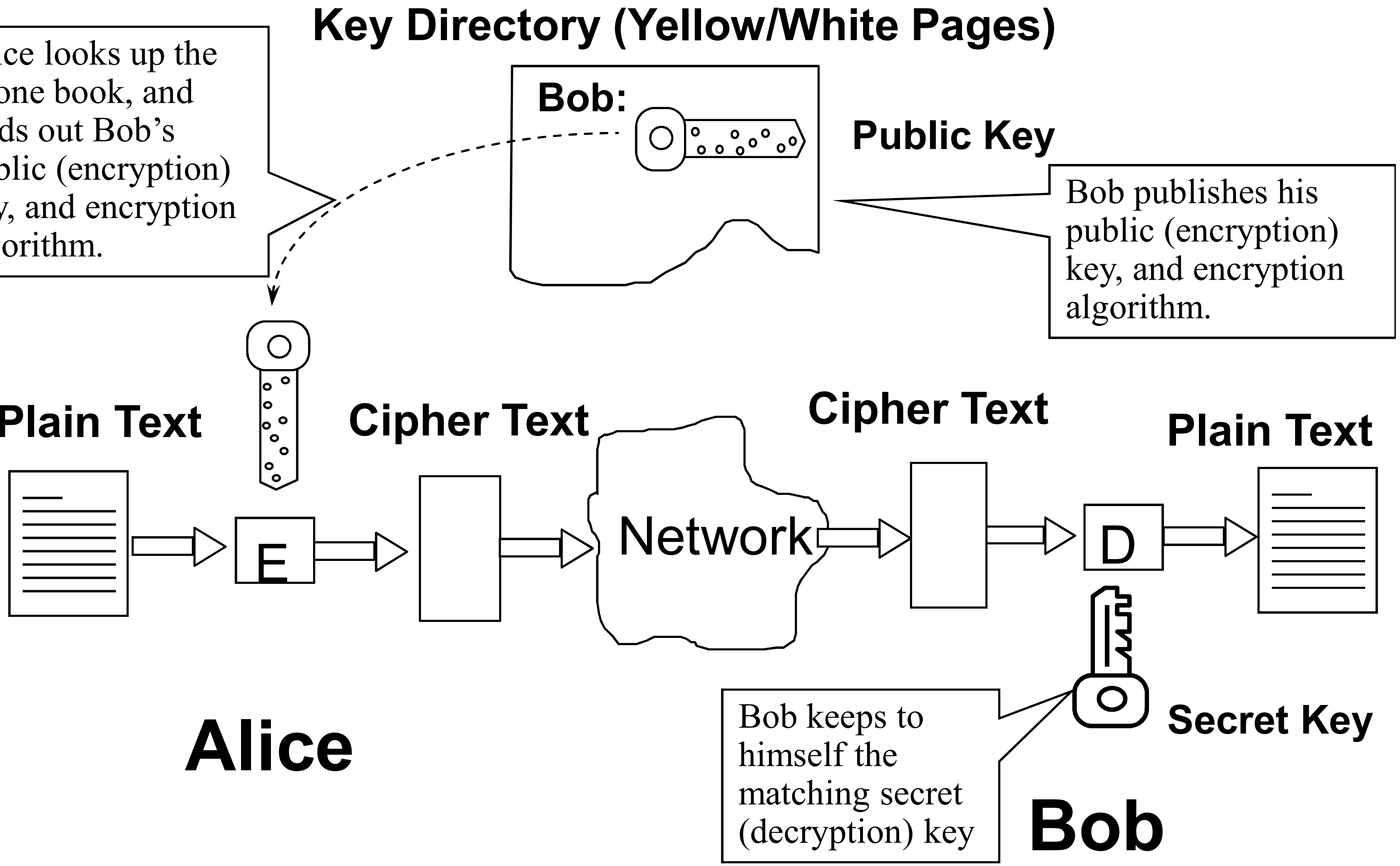


Public Key (Asymmetric) Cryptography

- ♦ Why public key cryptography ?
- ♦ General principles of public key cryptography
- ♦ The RSA public key cryptosystem
- ♦ Why RSA is secure?

Public Key Cryptosystem





Key Directory (Yellow/White Pages)

Bob:

Public Key

Alice looks up the phone book, and finds out Bob's public (encryption) key, and encryption algorithm.

Bob publishes his public (encryption) key, and encryption algorithm.

Plain Text

E

Cipher Text

Network

Cipher Text

D

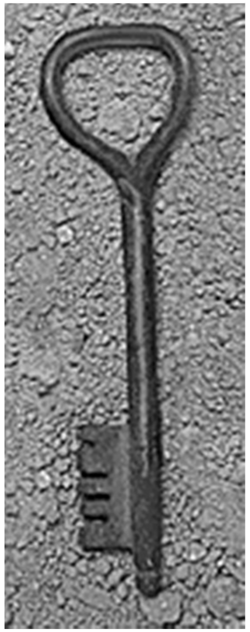
Plain Text

Secret Key

Bob keeps to himself the matching secret (decryption) key

Alice

Bob



Major Differences with Private Key Ciphers

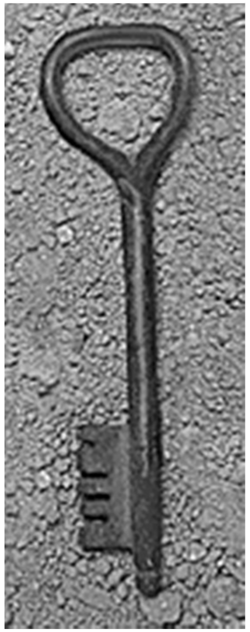
- The public encryption key is different from the secret decryption key.
- Infeasible for an attacker to find out the secret decryption key from the public encryption key.

不可行

理论上可以解决，
从计算上很难的问题

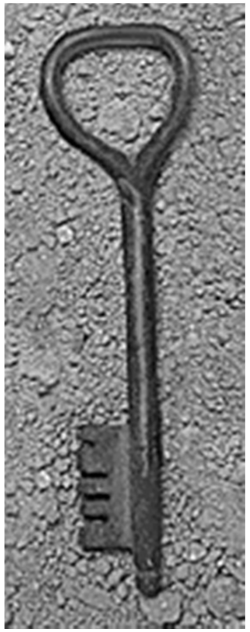
- no need for Alice & Bob to distribute a shared secret key beforehand ! 事先不需要分发密钥
- only one pair of public and secret keys is required for each user ! No matter how many communication counterparties

解决了密钥
爆炸问题



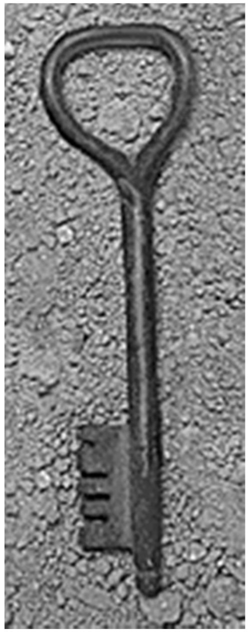
Public Key (Asymmetric) Cryptography

- ♦ Why public key cryptography ?
- ♦ General principles of public key cryptography
- ♦ The RSA public key cryptosystem
- ♦ Why RSA is secure?



Realising Public Key Ciphers

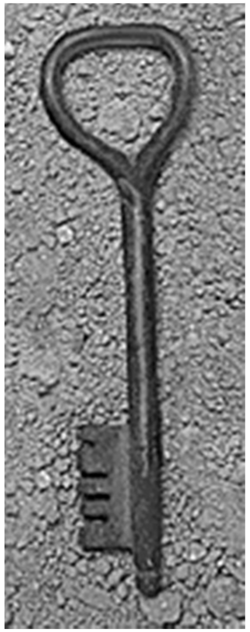
- The most famous system that implements Diffie & Hellman's ideas on public key ciphers is due to
  - Ronald Rivest
  - Adi Shamir
  - Leonard Adleman
- This concrete public key cryptosystem is called RSA.



质数和合数

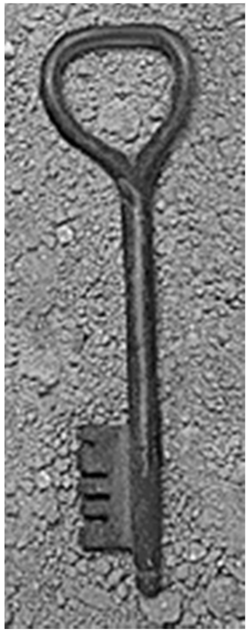
Prime & Composite

- Prime and composite numbers
  - a prime number is an integer that can divided only by 1 and itself
    - E.g. 2, 3, 5, 7, 11, 13, 101, 103, ...
  - all other integers are composite
    - E.g. 4, 6, 8, 9, 10, 12, 523743960876432, 800164386535

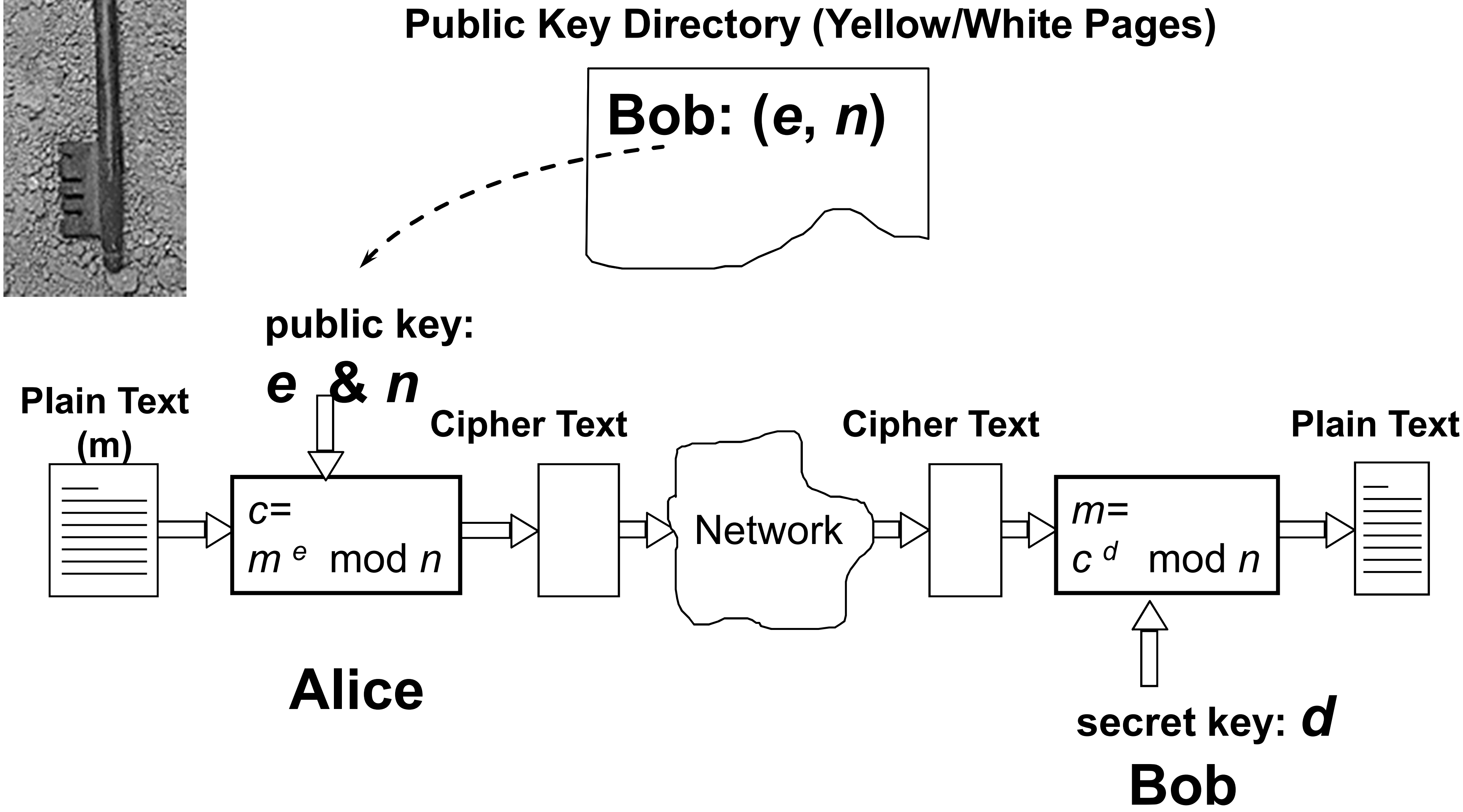


Modular operations

- “remainder”
  - 13 = 3 (mod 5),   1 = 1 (mod 7)
  - 20 = 0 (mod 5),   32 = 4 (mod 7)
- modular exponentiation
  - 2^2 = 1 (mod 3), 3^2 = 0 (mod 3)
  - 2^2 = 4 (mod 5), 10^2 = 8 (mod 92)
  - 4^6 = 6 (mod 10), 3^{11} = 7 (mod 10)



RSA Public Key Cryptosystem



Public Key Directory (Yellow/White Pages)

Bob: (e, n)

public key:

e & n

Plain Text (m)

$c = m^e \bmod n$

Cipher Text

Network

Cipher Text

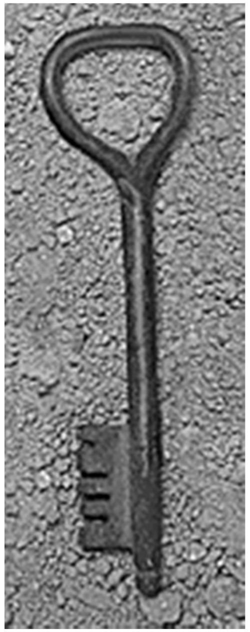
$m = c^d \bmod n$

Plain Text

Alice

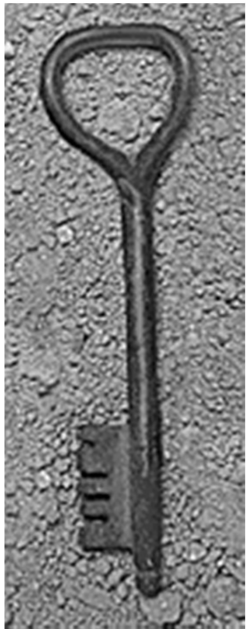
secret key: d

Bob



RSA (1)

- ♦ Bob:
  - – chooses 2 large primes (each at least 100 digits): *p, q*
    multiplies p and q: $n = p*q$
  - – finds out two numbers *e* & *d* such that
    $$e * d = 1 \ (mod\ (p-1)(q-1))$$
  - – public key (published in the phone book)
    - • 2 numbers:(*e, n*)
    - • encryption algorithm: modular exponentiation
  - – secret key: *d*

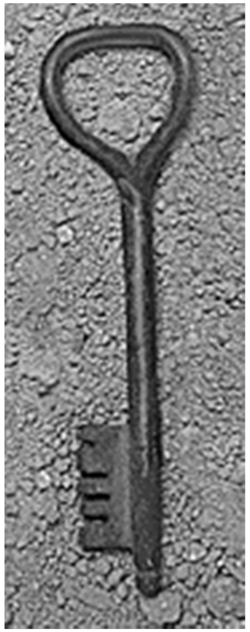


RSA (2)

- Alice has a message *m* to be sent to Bob:
  - finds out Bob’s public encryption key (*e*, *n*)
  - calculates

    $$c = m^e\ (mod\ n)$$

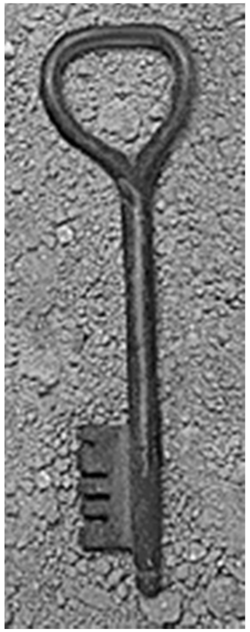
  - sends the ciphertext *c* to Bob



RSA (3)

- ♦ Bob:
  - – receives the ciphertext *c* from Alice
  - – uses his matching secret decryption key *d* to calculate

$$m = c^d \ (mod\ n)$$



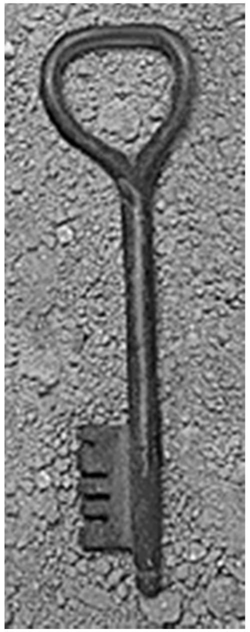
RSA --- 1st small example (1)

♦ Bob:

- chooses 2 primes: *p=5, q=11*
  multiplies p and q: $n = p*q = 55$
- finds out two numbers *e=3* & *d=27* which satisfy

$$3 * 27 = 1 \ (mod\ 40)$$

- Bob's public key
  - 2 numbers:(*3, 55*)
  - encryption algorithm: modular exponentiation
- secret key: *27*

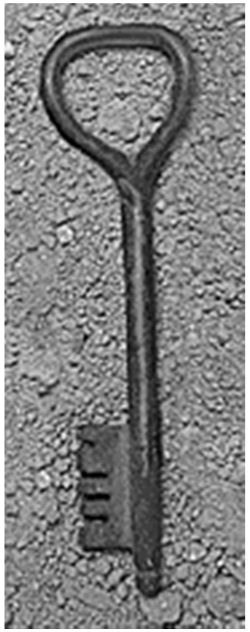


RSA --- 1st small example (2)

- Alice has a message *m=13* to be sent to Bob:
  - finds out Bob's public encryption key (*3, 55*)
  - calculates

$$\begin{aligned} c &= m^e \ (mod\ n) \\ &= 13^3 \ (mod\ 55) \\ &= 2197 \ (mod\ 55) \\ &= 52 \end{aligned}$$

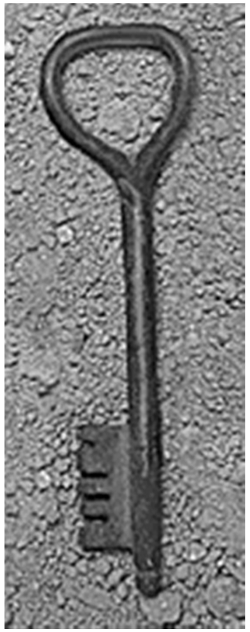
  - sends the ciphertext *c=52* to Bob



RSA --- 1st small example (3)

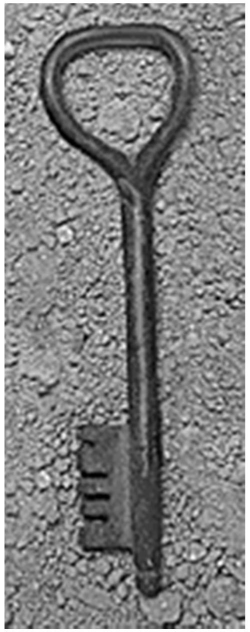
- Bob:
  - receives the ciphertext *c=52* from Alice
  - uses his matching secret decryption key *27* to calculate

$$m = 52^{27} \ (mod\ 55)$$
$$= 13 \ (Alice's\ message)$$



RSA --- 2nd small example (1)

- Bob:
  - chooses 2 primes: *p=101, q=113*
    multiplies p and q: *n = p*q = 11413*
  - finds out two numbers *e=3533* & *d=6597* which satisfy
    $$3533 * 6597 = 1 \ (mod\ 11200)$$
  - Bob's public key
    - 2 numbers:(*3533, 11413*)
    - encryption alg: modular exponentiation
  - secret key: *6597*

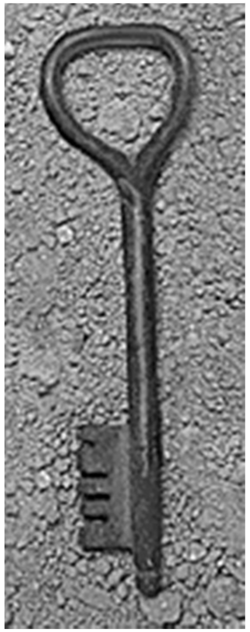


RSA --- 2nd small example (2)

- Alice has a message *m=9726* to be sent to Bob:
  - finds out Bob's public encryption key (*3533*, *11413*)
  - calculates

$$\begin{aligned} c &= m^e \ (mod\ n) \\ &= 9726^{3533} \ (mod\ 11413) \\ &= 5761 \end{aligned}$$

  - sends the ciphertext *c=5761* to Bob



RSA --- 2nd small example (3)

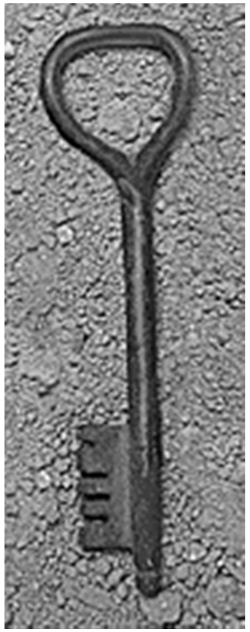
- Bob:
  - receives the ciphertext *c=5761* from Alice
  - uses his matching secret decryption key *6597* to calculate

$$m = c^d \ (mod\ n)$$

$$= 5761^{6597}\ (mod\ 11413)$$

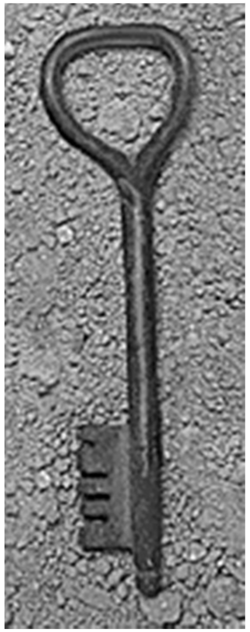
$$= 9726\ (Alice's\ message)$$

指数取余可以在计算机中加速
不需要把指数算出来



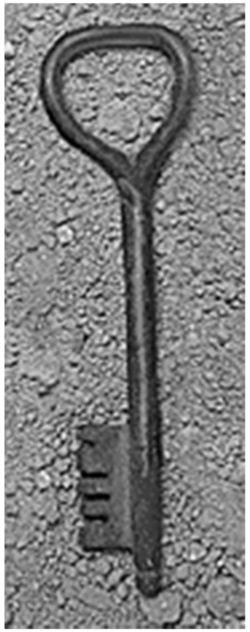
Remarks on RSA

- ♦ The message m has to be an integer between in the range [1, n].
- ♦ To encrypt long messages we can use a hybrid cryptosystem (see later).



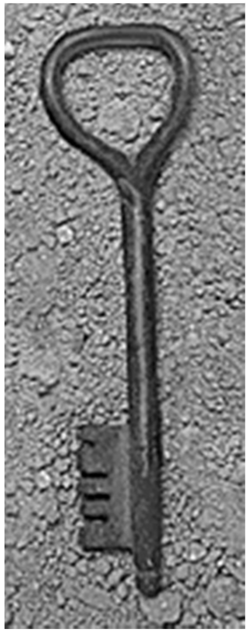
Public Key (Asymmetric) Cryptography

- ♦ Why public key cryptography ?
- ♦ General principles of public key cryptography
- ♦ The RSA public key cryptosystem
- ♦ Why RSA is secure?



Why RSA is Secure

- Attack Scenario:
  - Marvin wants to read Alice's private message (m) intended to be read only by Bob.
  - However, Alice used RSA to encrypt m using Bob's public key (*e, n*), into the ciphertext ***c*** = m^e (mod *n*).
  - Marvin is a determined attacker and managed to intercept the ciphertext c on its way from Alice's to Bob's computer.
  - Marvin also looked up Bob's public key (*e,n*) to help him in his attack.



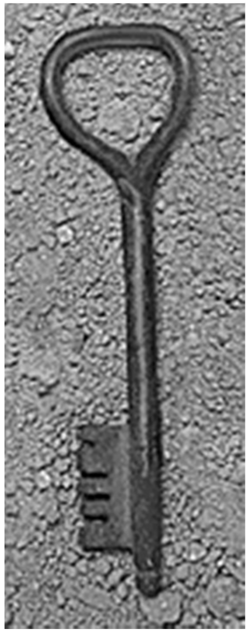
Why RSA is Secure

- Marvin now has (c,e,n) and wants to find out m.
- How can Marvin proceed to find m?
  - Approach 1: If Marvin could also find out Bob's secret key d, he could decrypt c into m in the same way as Bob does.
    - Suppose Bob guards his secret key d very well, what can Marvin do then?
  - Approach 2: Marvin knows that $\boldsymbol{c} = m^{e}$ (mod n). He knows that m is a number between 0 and n-1. So he could use exhaustive 穷举 search through all n possible messages m.
    - But if n is large this takes a long time!

Why RSA is Secure

- Marvin's Attack options (cont):
  - Approach 3: Marvin can try to ***compute*** **Bob's secret key d from (e,n) and then use Approach 1.**
    - Remember that e * d = 1 (mod (p-1)(q-1))
    - Marvin found in a 'Number Theory' book a very fast algorithm called *EUCLID* to solve the following problem: Given two numbers (r,s), the algorithm outputs a number x such that

      r * x = 1 (mod s).
    - Exercise: Explain how Marvin can use algorithm *EUCLID* to find Bob's secret key d very quickly from (e,n) once he manages to 'factorize' n = p*q into the prime factors p and q.



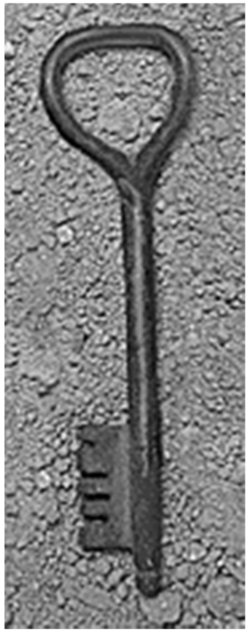
Why RSA is Secure

- Approach 3 is the most efficient known method Marvin can use to attack RSA!

- The time taken for Marvin to execute the attack in Approach 3 is essentially the time to factorize n=p*q into the prime factors p and q.

- Therefore, we say that RSA is *based on* the *factorization problem*:

  While it is easy to multiply large primes together, it is computationally infeasible to factorize or split a large composite into its prime factors !

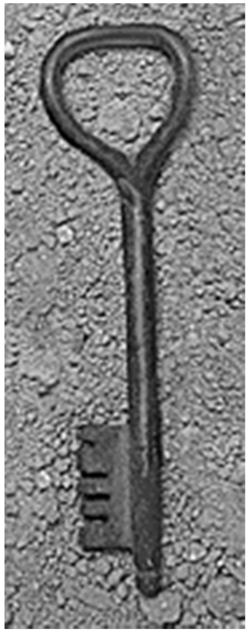


Why RSA is Secure (Optional)

- ♦ The current state of the art in factorization:
  - – Largest RSA number factored so far: 155 decimal digits, as at August 1999
    - It took several months of computing time on many computers around the world
    - Exercise: How long was the binary representation of the above number (bit length)?

      (hint: $\log_2(10) = 3.32$ approximately)
  - – The length of n in an RSA key should therefore be sufficiently longer than 155 decimal digits to be secure against attackers with access to many fast computers.

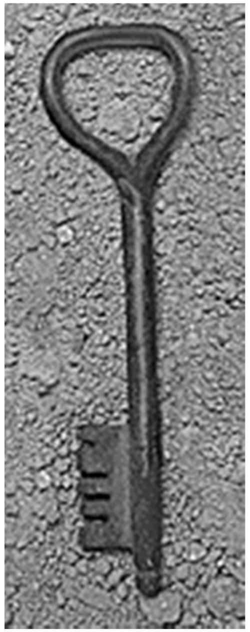
Why RSA is Secure (Optional)



- How many digits should n have to be secure?
- Approximate Factoring Time: For the fastest known factoring algorithm ('Number Field Sieve'):
  - If it takes time T to factorize number of length |n| digits (or bits),
  - Then it takes time $M(k) \times T$ to factorize a number of length k * |n| digits where (with |n| in bits):

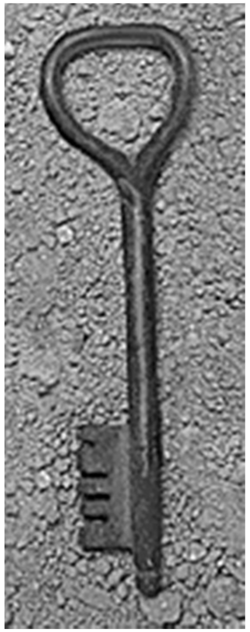
$$M(k) \approx 2^{1.923|n|^{1/3}\left[k^{1/3}(\log_2(k|n|/1.44))^{2/3} - (\log_2(|n|/1.44))^{2/3}\right]}$$

- Assu of length 155 decimal digits, it would take:
  - M(2)*T = 2^{22} days = 20,000 years to factor n of length |n| = 2*155 = 310 digits
  - M(3)*T = 2^{39} days = 2 billion (!!) years to factor n of length |n| = 3*155 = 465 digits…



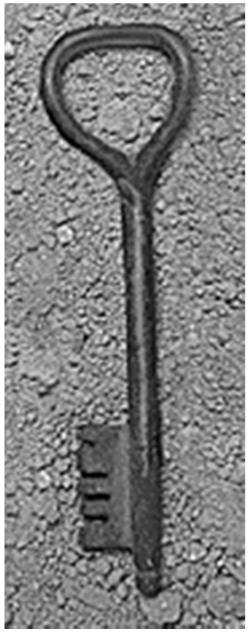
Why RSA is Secure (Optional)

- ♦ Therefore, when both p and q in RSA are of at least 155 digits, the product n=p*q is 310 digits.
- ♦ Then no one can factorize n in less time than a few thousand years, not even Marvin!!
- ♦ Thus the *only* person who can extract the plaintext m from the ciphertext c is Bob, as only he knows the secret decryption key d !



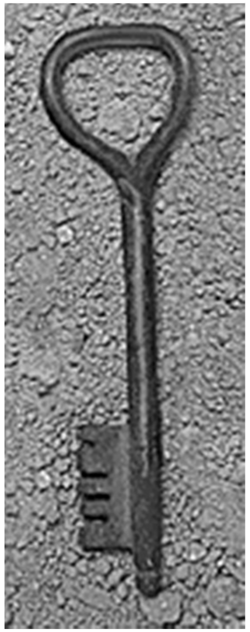
Marvin’s New Attack Idea (Optional)

- Instead of just eavesdropping, Marvin can try a more *active* attack!
- Outline of the New Attack:
  - Marvin generates an RSA key pair
    - Public key = Kpub_* = (N_*, e_*)
    - Secret key = Ksec_* = d_*
  - Marvin sends the following email to Alice, pretending to be Bob:
    - Hi Alice,
      - Please use my new public key from now on to encrypt messages to me. My new public key is Kpub_*.
      - Yours sincerely, Bob.
  - Marvin decrypts any messages Alice sends to Bob (encrypted with Kpub_*), using Ksec_*.



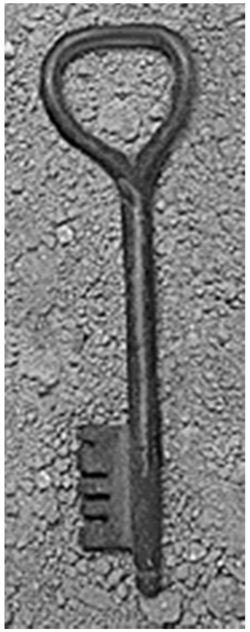
Preventing Marvin’s Active Attack (Optional)

- The active attack works because:
  - Alice was tricked by Marvin into encrypting a message intended for Bob using a “fake” public key which is NOT Bob’s public key (in fact it was Marvin’s).
- To prevent the attack:
  - Before Alice encrypts a message for Bob, she must make sure she has Bob’s CORRECT public key (and not a fake one).
  - Alice needs a way of testing the truth of any “Bob’s key message” informing Alice of Bob’s Public Key.
  - No one besides Bob should be able to produce such a message so that it will pass Alice’s Test.



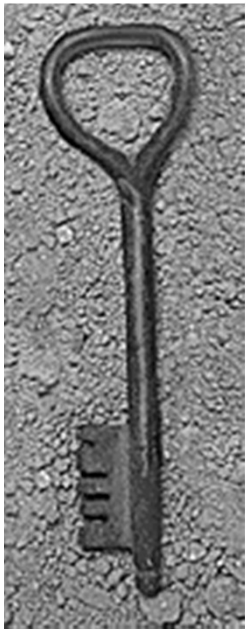
Preventing Marvin's Active Attack (Optional)

- This is a setting where Alice and Bob have a <u>message integrity</u> security requirement!
  - Ie. Alice and Bob want to prevent fabrication and/or modification of a "Bob's key message" (a message informing Alice of Bob's public key) by unautorised parties (like Marvin).
- The main cryptographic tool used to achieve message integrity is "Digital Signatures".
- In a later lecture (after we have covered "Digital Signatures"), we will come back to this topic and see how Digital Signatures can be used to prevent Marvin's Attack!



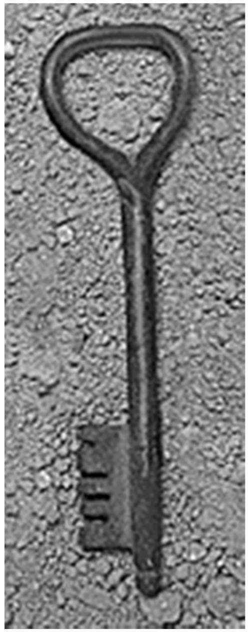
Private key ciphers

- ♦ Good points
  - – in-expensive to use
  - – fast
  - – low cost VLSI chips available
- ♦ bad points
  - – key distribution is a problem



Public key ciphers

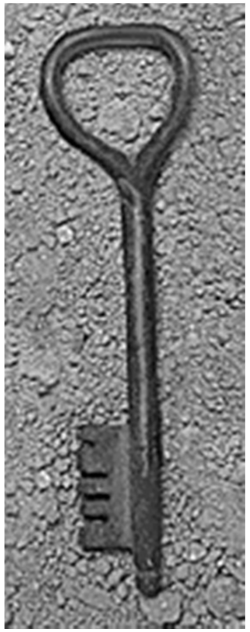
- ♦ good points
  - – key distribution is NOT a problem
- ♦ bad points
  - – relatively expensive to use
  - – relatively slow
  - – VLSI chips not available or relatively high cost



Combining 2 type of ciphers

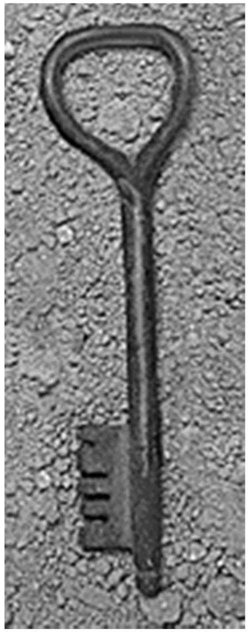
- In practice, we
  - use a public key cipher (such as RSA) to distribute keys
  - use a private key cipher (such as DES) to encrypt and decrypt messages

甲用RSA算法加密对称密钥，对称密钥加密文件，发给乙。
乙用RSA解密对称密钥，再用对称密钥解密文件。



Major Types of Algorithms

- Secret-Key (Symmetric) Cryptography
- Public Key (Asymmetric) Cryptography
- Digital Signatures & Hash Algorithms



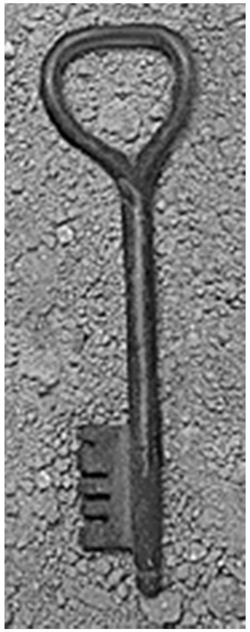
The need of digital signature

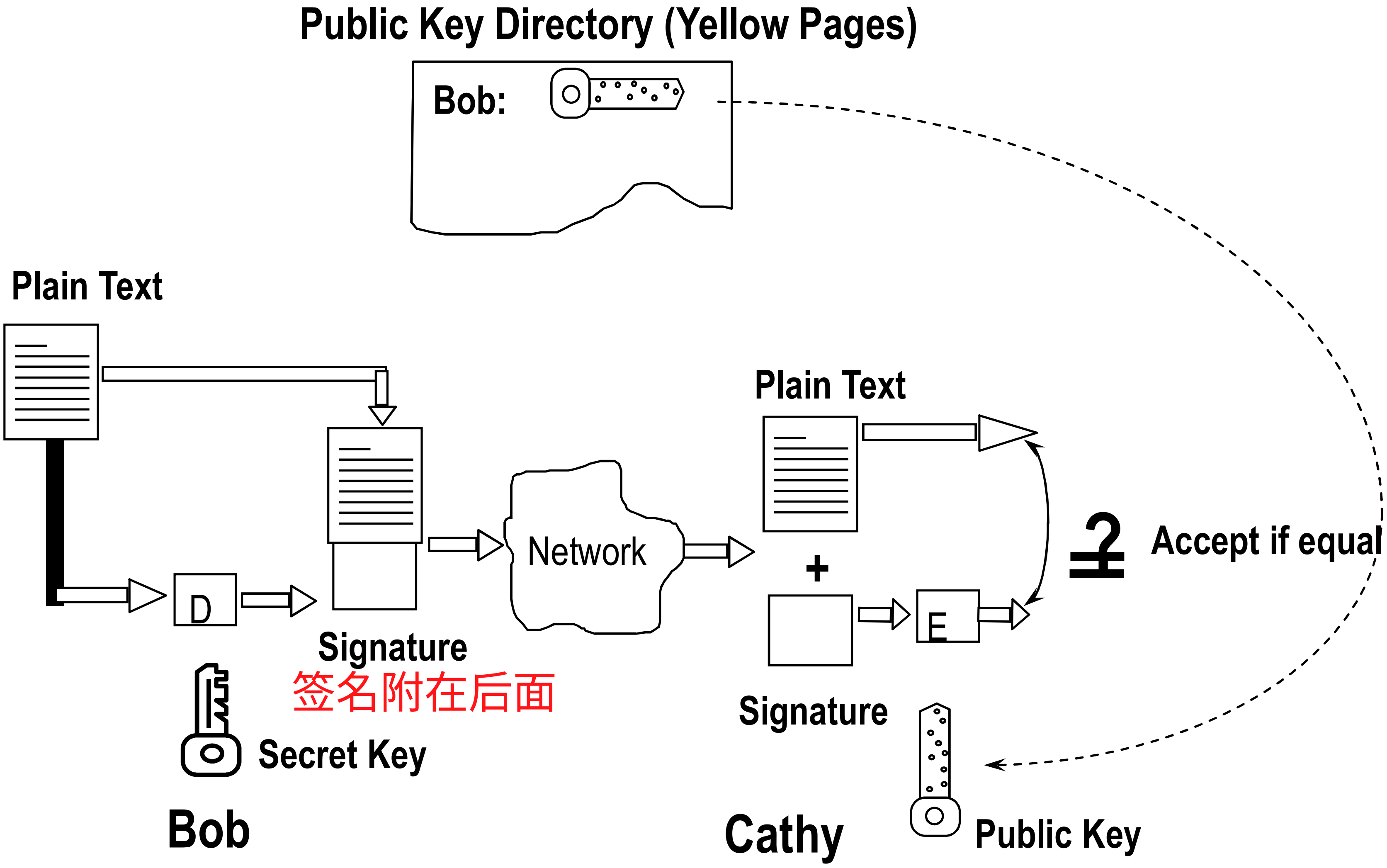
数字签名

- social & business activities and their associated documents are becoming digital
  - digital conferences
  - digital contract signing
  - digital cash payments, ……
- hand-written signatures are not applicable to digital data

伪造比较难，也不能抵赖。

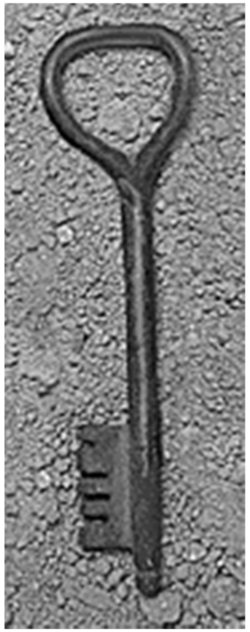
Digital Signature (for short doc)

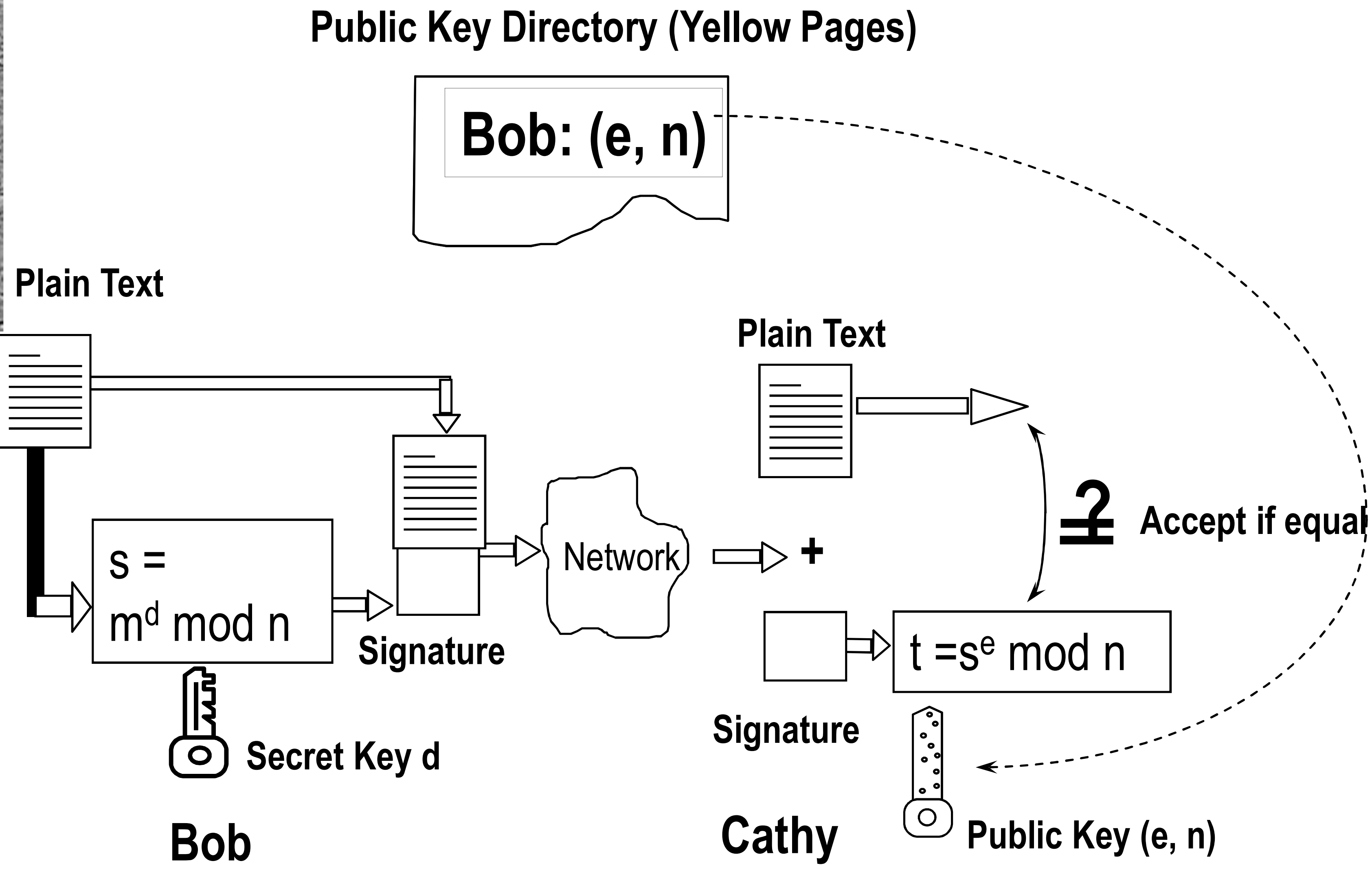


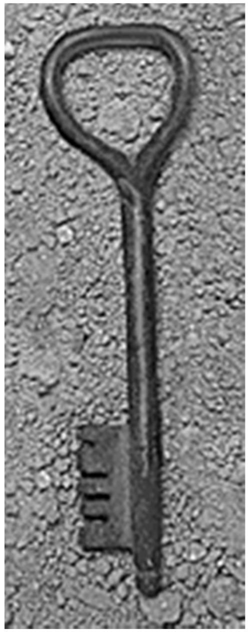


签名附在后面

Digital Signature (based on RSA)

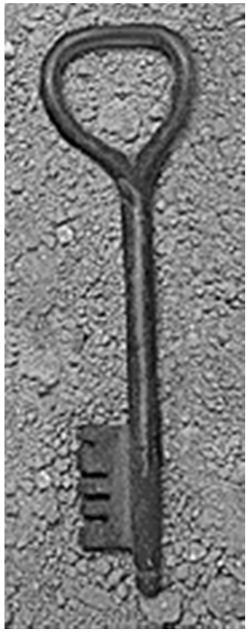






RSA signature --- an eg (1)

- Bob:
  - chooses 2 primes: *p=5, q=11*
    multiplies p and q: $n = p*q = 55$
  - finds out two numbers *e=3* & *d=27* which satisfy
    $$3 * 27 = 1 \ (mod\ 40)$$
  - Bob's public key
    - 2 numbers:(*3, 55*)
    - encryption alg: modular exponentiation
  - secret key: *27*

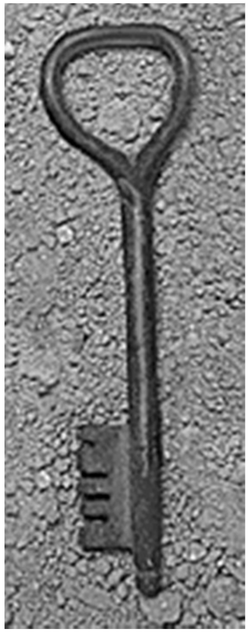


RSA signature --- an eg (2)

- Bob has a document *m=19* to sign:
  - uses his secret key *d=27* to calculate the digital signature of *m=19*:

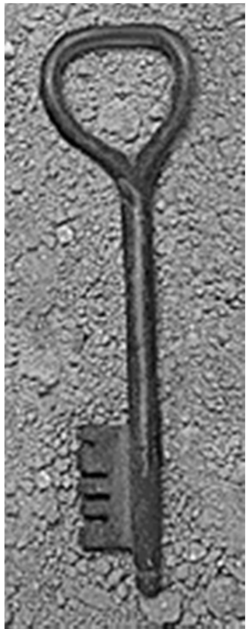
$$
\begin{aligned}
s &= m^d \ (mod\ n) \\
  &= 19^{27} \ (mod\ 55) \\
  &= 24
\end{aligned}
$$

  - appends 24 to 19. Now *(m, s) = (19, 24)* indicates that the doc is 19, and Bob's signature on the doc is 24.



RSA signature --- an eg (3)

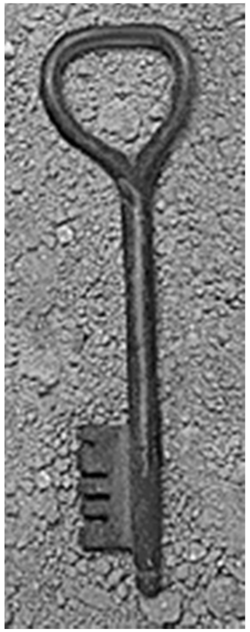
- Cathy, a verifier:
  - receives a pair $(m,s)=(19, 24)$
  - looks up the phone book and finds out Bob's public key $(e, n)=(3, 55)$
  - calculates $t = s^e \pmod{n}$
    $= 24^3 \pmod{55}$
    $= 19$
  - checks whether $t=m$
  - confirms that ($19,24$) is a genuinely signed document of Bob if $t=m$.



How about long documents ?

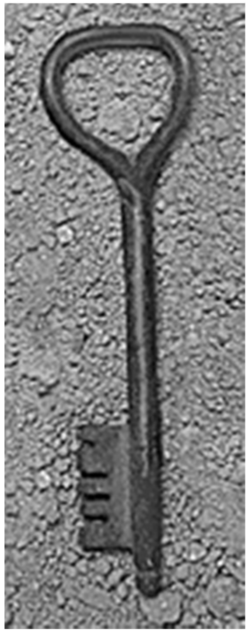
- In the previous example, a document has to be an integer in [0,...,n]
- to sign a very long document, we need a so called one-way hash algorithm
- instead of signing directly on a doc, we hash the doc first, and sign the hashed data which is normally short.

引出单向哈希函数
在数字签名时，不直接在电子合同明文上签名。
先通过单向哈希函数将明文变小，小到n。

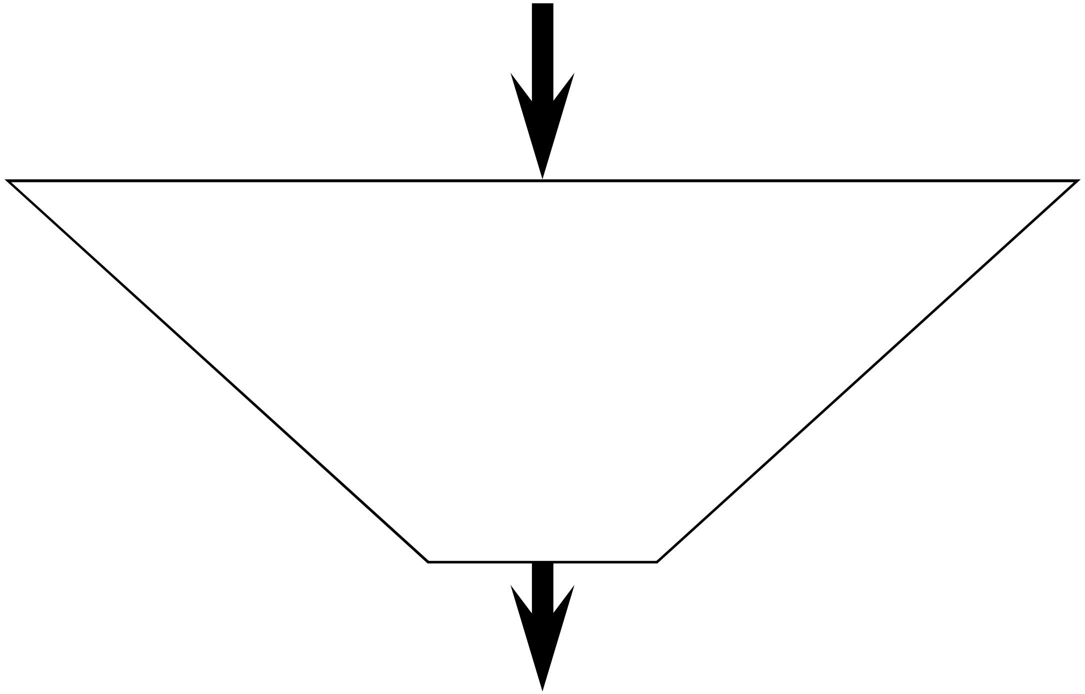


One-Way Hash Algorithm

- ♦ A one-way hash algorithm hashes an input document into a condensed short output (say of 100 bits)
  - – Denoting a one-way hash algorithm by H(.), we have:
    - Input: m - a binary string of any length
    - Output: H(m) - a binary string of L bits, called the "hash of m under H".
    - The output length parameter L is fixed for a given one-way hash function H,
    - eg
      - – The one-way hash function "MD5" has L = 128 bits
      - – The one-way hash function "SHA-1" hash L = 160 bits

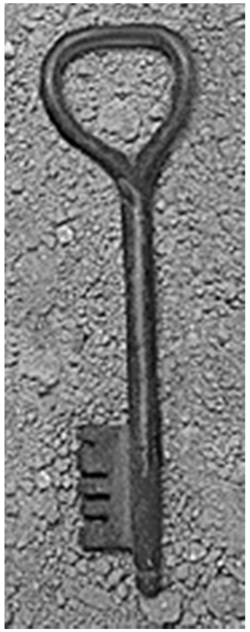


One-Way Hash Algorithm



A document (of any length)

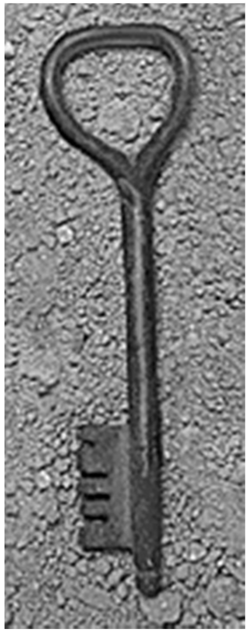
A condensed short output, say of 100 bits



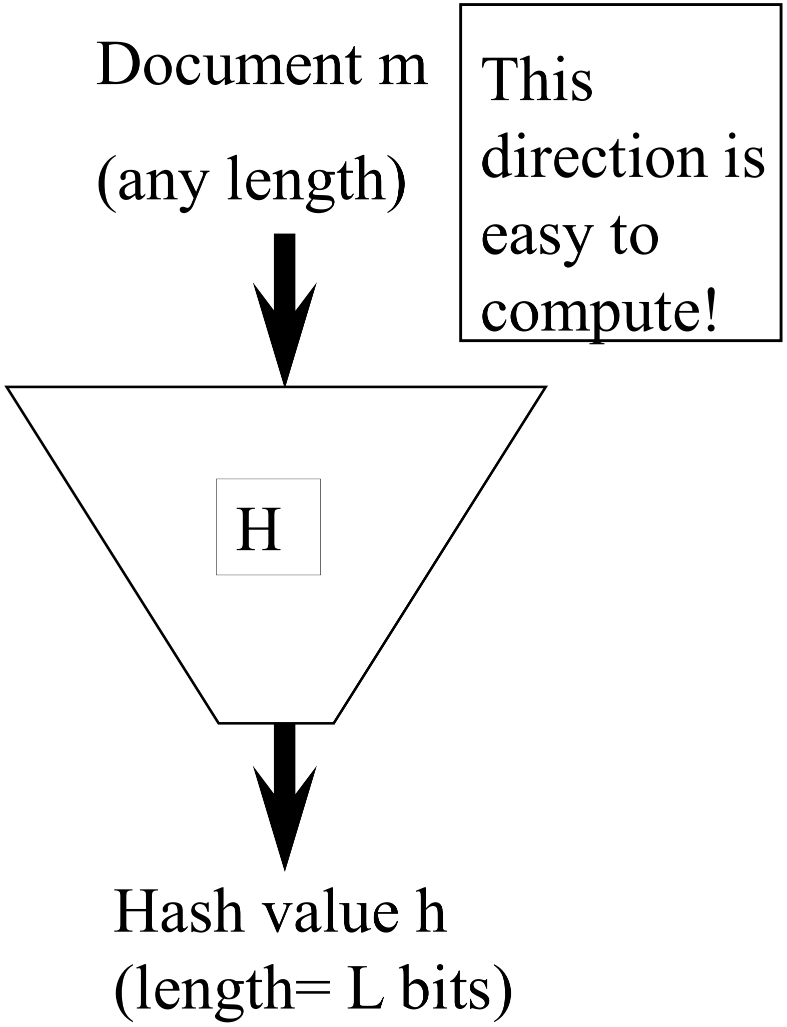
A good one-way hash algorithm H needs to have these properties: 满足三个特性。

- <u>1. Easy to Evaluate:</u> 正向计算速度快。
  - given any document m, the hashed value h = H(m) can be computed quickly.
- <u>2. Hard to Reverse:</u> 反向计算难，几乎不可能。
  - There is no feasible algorithm to “reverse” a hashed value,
  - I.e. given any hashed value h, it is computationally infeasible to find any document m such that H(m) = h.
- NOTE: An algorithm is called <u>‘One-Way’</u> if it has BOTH properties 1 and 2.
- <u>3. Hard to find Collisions:</u> 很难重复，撞库。
  - There is no feasible algorithm to find two or more input documents which are hashed into the same condensed output,
  - I.e it is computationally infeasible to find any two documents m_1, m_2 such that $H(m_1)= H(m_2)$.

根据抽屉原理，0-n个抽屉，每个抽屉一定对应多个明文。不同明文对应同一hash，但是有心去找准确的重复的存在时，是找不到的。



The One-way Property

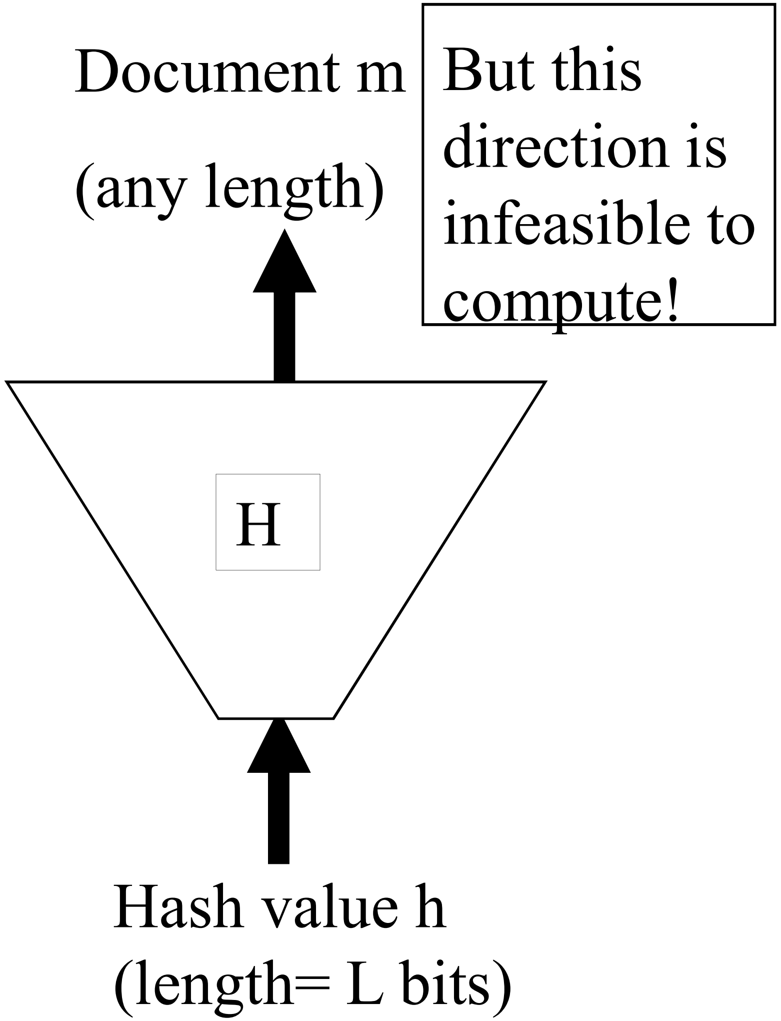


Document m

(any length)

This direction is easy to compute!

Hash value h
(length= L bits)

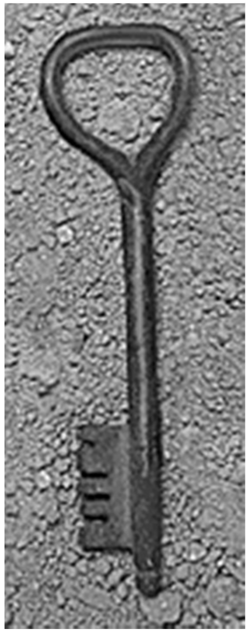


Document m

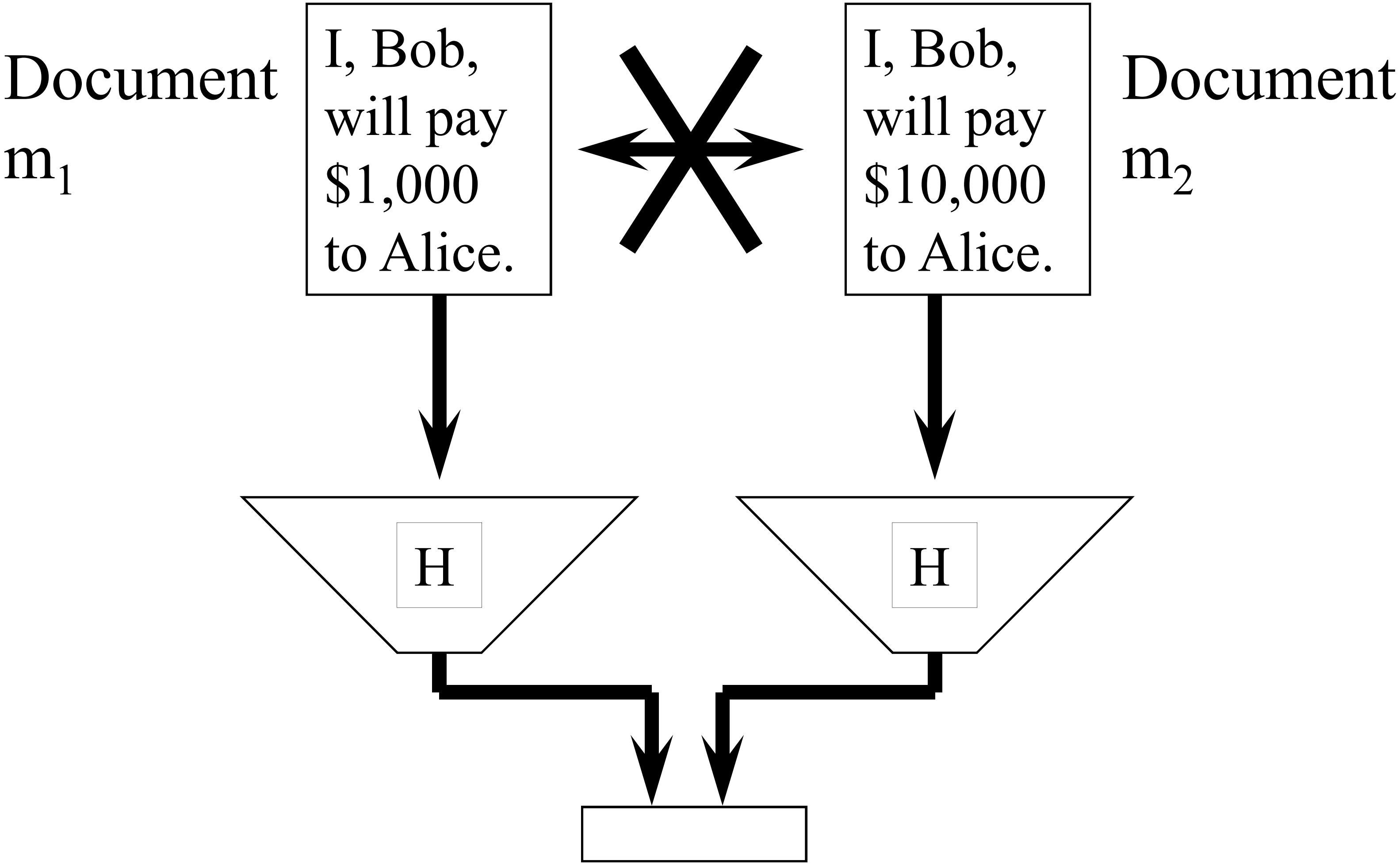
(any length)

But this direction is infeasible to compute!

Hash value h
(length= L bits)



Finding collision is infeasible



Document m_1

I, Bob, will pay $1,000 to Alice.

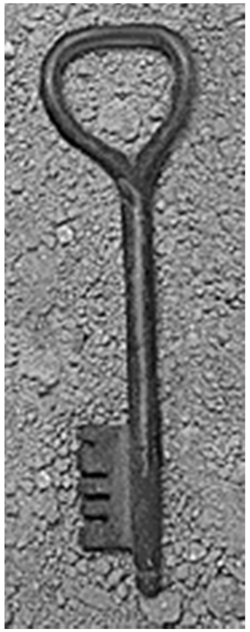
I, Bob, will pay $10,000 to Alice.

Document m_2

H

H

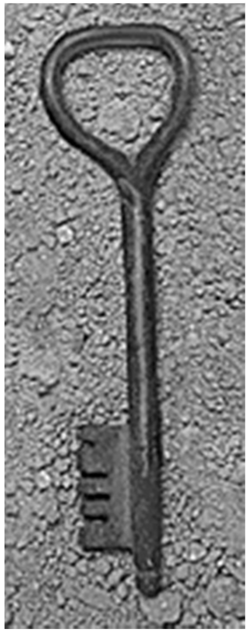
(same condensed output)

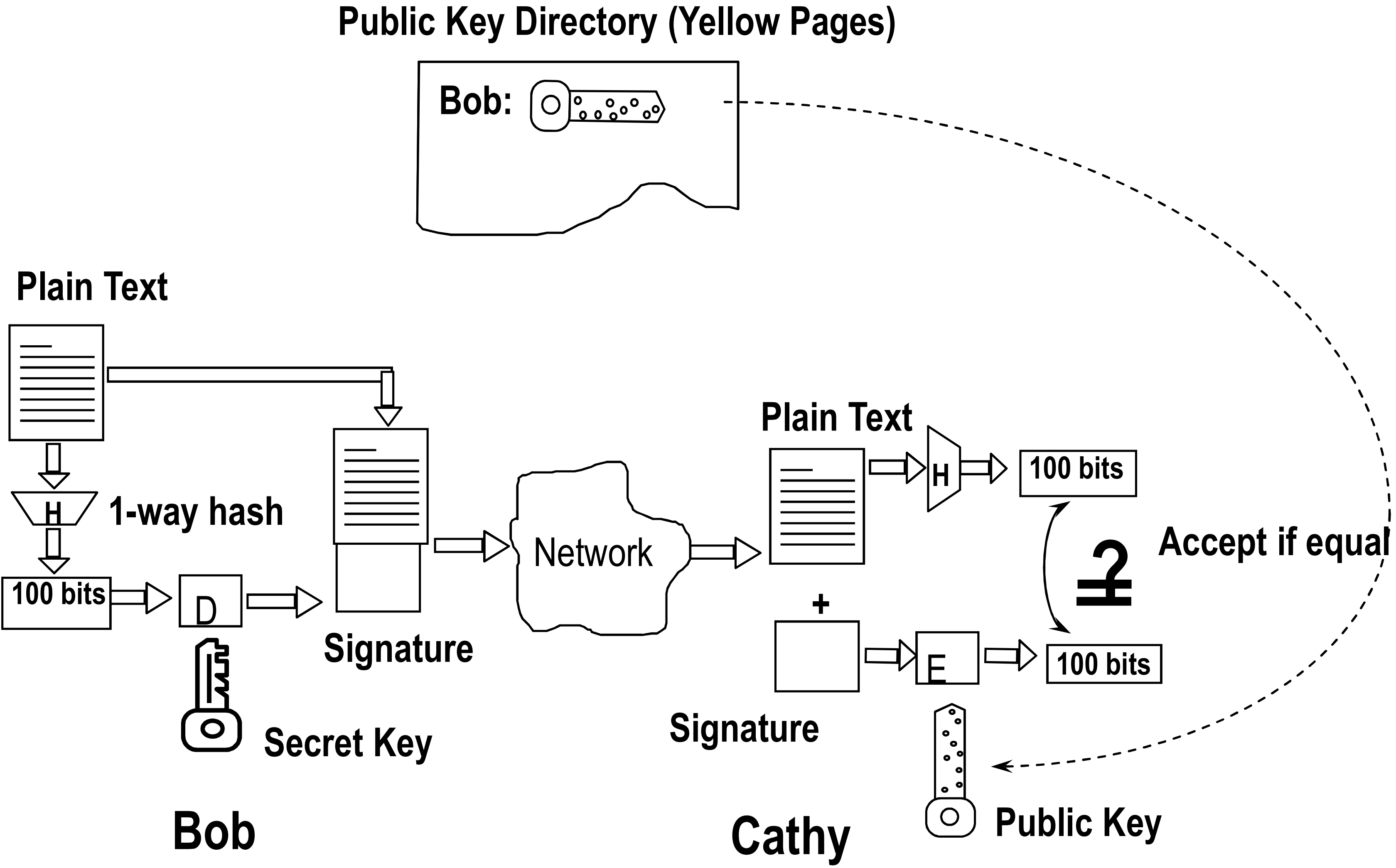


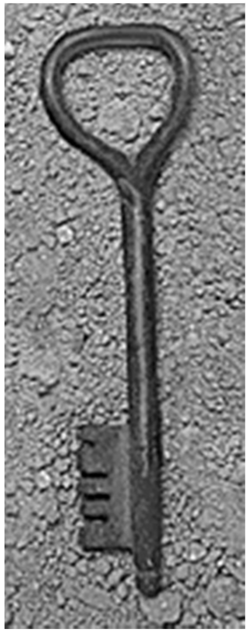
Good one-way hashing algorithms

- MD5 (R. Rivest, 1992)
- SHS (secure hashing standard, USA, 1992, modified in 1995)
- HAVAL (Y. Zheng, 1992)

Digital Signature (for long doc)

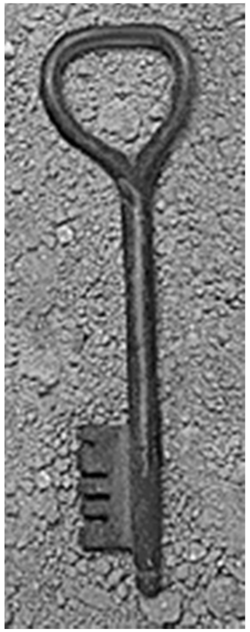






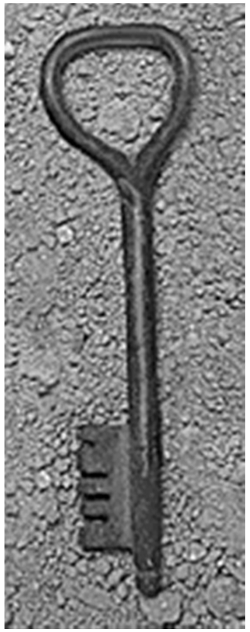
Why Digital Signature ?

- ♦ Unforgeable
  - – takes 1 billion years to forge !
- ♦ Un-deniable by the signatory
- ♦ Universally verifiable
- ♦ Differs from doc to doc
- ♦ Easily implementable by
  - – software or
  - – hardware or
  - – software + hardware



Important digital signatures

- ♦ RSA
  - – strongly supported by industries
  - – a de facto industrial standard
- ♦ Schnorr digital signature
  - – derived from ElGamal digital signature
  - – based on infeasibility of discrete logarithm
- ♦ DSS (digital signature standard, USA)
  - – derived from ElGamal digital signature
  - – based on infeasibility of discrete logarithm
  - – strongly pushed forward by US government
- ♦ Signature schemes using elliptic curves



Cracking MD5

- Year 2005
- Three female researchers, Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, Professor of Shandong University of Technology's mathematics department
- Find Collision of MD5
- Reduced the amount of time needed to find two documents with the same signature by a factor of more than 2000.
  - Based on this, Time to crack SHA-1 from 2^69 to 2^63.
  - Y2013, Marc Stevens, reduced to 2^61
  - Y2016, Marc Stevens, reduced to 2^57.5. 100GPU for a year
  - Y2017.2.23, Google announced the collision of SHA-1