Computer Security and Cryptography

CS381

来学嘉

计算机科学与工程系 电院3-423室

34205440 1356 4100825 laix@sjtu.edu.cn

2015-05

Organization

- Week 1 to week 16 (2015-03 to 2014-06)
- 东中院-3-102
- Monday 3-4节; week 9-16
- Wednesday 3-4节; week 1-16
- lecture 10 + exercise 40 + random tests 40 + other 10
- Ask questions in class – counted as points
- Turn ON your mobile phone (after lecture)
- Slides and papers:
  - http://202.120.38.185/CS381
    - computer-security
  - http://202.120.38.185/references
- TA: Geshi Huang gracehgs@mail.sjtu.edu.cn
- Send homework to the TA

Rule: do the homework on your own!

Contents

- Introduction -- What is security?
- Cryptography
  - Classical ciphers
  - Today's ciphers
  - Public-key cryptography
  - Hash functions and MAC
  - Authentication protocols
- Applications
  - Digital certificates
  - Secure email
  - Internet security, e-banking
- Computer and network security
  - Access control
  - Malware
  - Firewall
- Examples: Flame, Router, BitCoin ??

References

- W. Stallings, *Cryptography and network security - principles and practice*， Prentice Hall.
- W. Stallings, 密码学与网络安全：原理与实践（第4版），刘玉珍等译，电子工业出版社，2006
- Lidong Chen, Guang Gong, *Communication and System Security*, CRC Press, 2012.
- A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997, ISBN: 0-8493-8523-7, http://www.cacr.math.uwaterloo.ca/hac/index.html
- B. Schneier, *Applied cryptography*. John Wiley & Sons, 1995, 2nd edition.
- 裴定一,徐祥, 信息安全数学基础, ISBN 978-7-115-15662-4, 人民邮电出版社,2007.

contents

- Public-key cryptosystems:
  - RSA   - factorization
  - DH , ElGamal -discrete logarithm
  - ECC
- Math
  - Fermat's and Euler's Theorems & ø(n)
  - Group, Fields
  - Primality Testing
  - Chinese Remainder Theorem
  - Discrete Logarithms

- **Issues in Information security**
  - **Scientific like**
    - **Confidentiality**
    - **Authentication**
    - **Access control**
    - **Integrity**
    - **Non-repudiation**
  - **More engineering**
    - **Virus protection**
    - **Intrusion prevention**
    - **Copyright protection**
    - **Content filtering**

Cryptography

Cryptology
(from the Greek for 'hidden word')

Cryptography –密码编码学
Code making

Cryptanalysis-密码分析
Code breaking 破译

Confidentiality
Secrecy

Authenticity
Data
entity

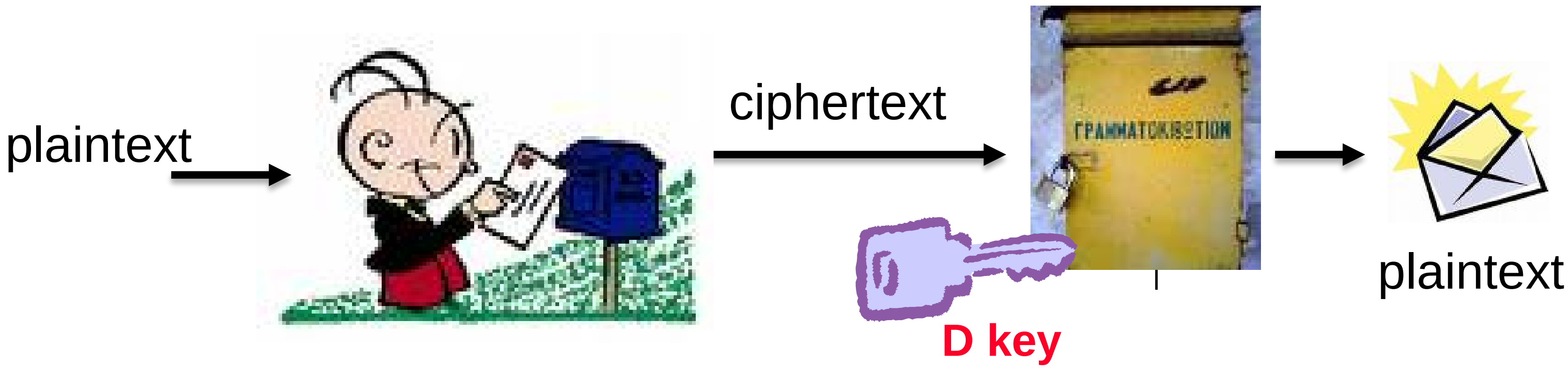
Integrity
Random number

Confidentiality and authenticity are independent attributes

Confidentiality

– Confidentiality : information is not disclosed to unauthorized individuals, entities, or processes. [ISO]

– Mechanism to achieve confidentiality--Encryption:



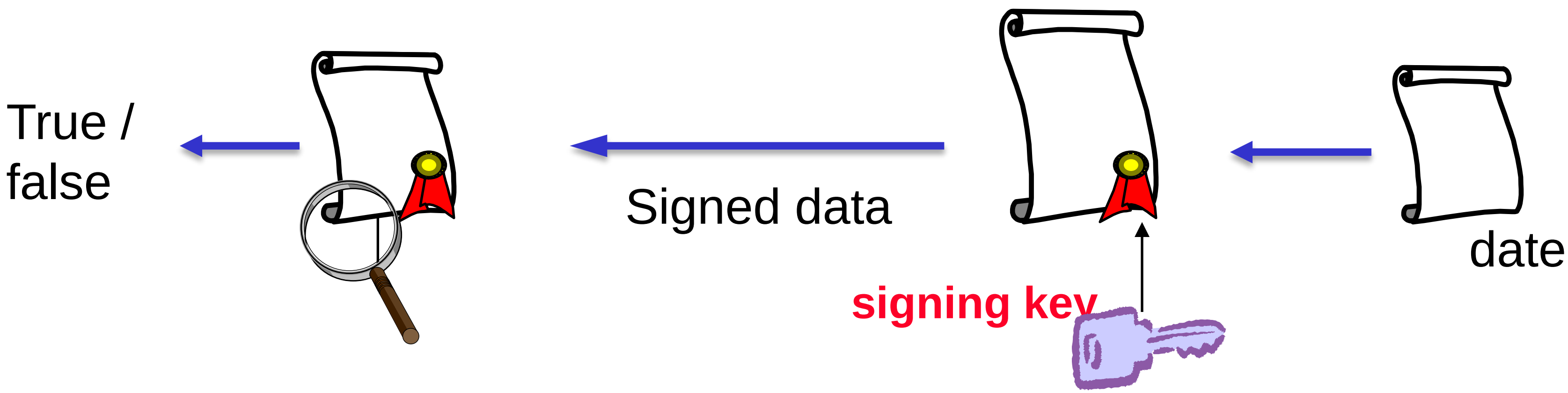
Only the user knowing the decryption key can recover plaintext

–"who can *read* the data"

Authenticity

– Authenticity: assurance of the claimed identity of an entity. [ISO]

– Example: ID-card, password, digital signature



Only the user knowing the secret-key can generate valid signature

"who *wrote* the data"

remark

- Understanding cryptography from the point of view of “read/write” is essential and useful.
- When an application or a functionality involves secret-key, it is helpful to decide whether it is a read or write problem, then pick up the correct approach: encryption or authentication.
- Example: copy-right protection, e-banking access, on-line transaction, e-voting, etc.

ciphersystems

- cipher
  - Asymmetric (public-key)
  - Symmetric (secret-key)
    - Block cipher
    - Stream cipher

cryptosystems

- symmetric cipher, secret-key cryptosystem： encryption key and decryption key are essentially the same, it is easy to derive one from the other.
  - Example: DES, RC2, IDEA, AES
- asymmetric cipher, public-key cryptosystem: encryption key and decryption key are different, it is difficult to derive one (private decryption key) from the other (public encryption key).
  - Example: RSA, ElGamal, ECC

- Symmetric --- sharing some secret
- Asymmetric --- sharing some trusted information

Two cryptosystems

Symmetric-key

- Advantages
  - high data throughput
  - Short size
  - primitives to construct various cryptographic mechanisms
- Disadvantages
  - the key must remain secret at both ends.
  - $O(n^2)$ keys to be managed for n users.

Public-key

- Advantages
  - Only the private key must be kept secret
  - Achieve non-repudiation (digital signature)
  - O(n) keys to be managed
- Disadvantages
  - low data throughput
  - much larger key sizes

The usage

- Public-key cryptography
  - signatures (particularly, non-repudiation) and key management
- Symmetric-key cryptography
  - encryption and some data integrity applications
- Private keys must be larger (e.g., 1024 or 2048 bits for RSA) than secret keys (e.g., 64 or 128 bits)
    - most attack on symmetric-key systems is an exhaustive key search
    - public-key systems are subject to “short-cut” attacks (e.g., factoring)
- **Hybrid system**: Use public-key to encrypt a session-key, then use the symmetric session key to encrypt document.

One-way functions

- Oneway function f: X ->Y, given x, easy to compute f(x); but for given y in f(X), it is hard to find x, s.t., f(x)=y.
    - Prob[f(A(f(x))=f(x))] < 1/p(n)   (TM definition, existence unknown)
    - Example: hash function, discrete logarithm;
- Keyed function f(X,Z)=Y, for known key z, it is easy to compute f(.,z)
    - Block cipher (fix c, f(c,.) is a oneway function)
- Keyed oneway function: f(X,Z)=Y, for known key z, it is easy to compute f(.,z) but for given y, it is hard to x,z, s.t., f(x,z)=y.
    - MAC function: keyed hash h(z,X), block cipher CBC
- **Trapdoor oneway function** $f_T(x)$: easy to compute and hard to invert, but with additional knowledge T, it is easy to invert.
    - Public-key cipher; RSA: $y=x^e$ mod N, T: N=p*q

Number Theory - Divisibility

- Divisibility

  For any two integers a,b, $a+b$, $a-b$, $a*b$ are all integers, but a/b may not be an integer.

  $a=b*q+r$, where $b>r\geq 0$.

  q is the quotient, and r is the remainder.

- If $r=0$, we call *b* divides *a*, denoted by $b|a$; otherwise we call *b* does not divide *a*, denoted by $b\nmid a$。

For $a,b,c\in Z$,

- If $a|b$, then $a|(bc)$;
- If $a|b$ and $a|c$, then $a|(b+c)$ and $a|(b-c)$;
- for $i,a,b\in Z$, if $a=bq+r$, $i|a$ and $i|b$, then $i|r$.

Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

Prime Factorisation

- to **factor** a number n is to write it as a product of other numbers: $n = a \times b \times c$
- factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes
  - eg. $91 = 7 \times 13$; $3600 = 2^4 \times 3^2 \times 5^2$
- Any number can be written as a product of prime powers

$$a = \prod_{p \in P} p^{a_p}$$

Relatively Prime Numbers

- two numbers a, b are **relatively prime** if they have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely one can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300=2^1\times 3^1\times 5^2$ $18=2^1\times 3^2$ hence $GCD(18,300)=2^1\times 3^1\times 5^0=6$

GCD and LCM

- *d* is the greatest common divisor of *a* and *b* if
  - $d|a$ and $d|b$;
  - If $f|a$ and $f|b$, then $f|d$;

  denoted by $d=\gcd(a,b)$, or (a,b).
- If $d|ab$, and $gcd(d,a)=1$, then $d|b$.

- *m* is the least common multiple of *a* and *b* if
  - $a|m$ and $b|m$;
  - If $a|n$ and $b|n$, then $m|n$;

  Denoted by $m=lcm(a,b)$, or $[a,b]$.

A useful equilvalent definition of GCD

- *Lemma: If d divides both a and b, and d = ax + by for some integers x and y, then d = gcd(a,b).*

Proof.

First, d is a common divisor of a and b, hence

$d \leq gcd(a,b)$.

Second, since gcd(a,b) is a common divisor of a and b, it must also divide ax + by = d, which implies $gcd(a,b) \leq d$.

The Euclid Algorithm

- gcd(a,b)=d
  - Fact 1： gcd(a,b)= gcd(b, a-b)；
  - Fact 2: if a=qb+r, then gcd(a,b)= gcd(b,r)；
  - Fact 3: there exist integers x,y : gcd(a,b)=ax+by
- With the Euclid algorithm to determine d= gcd(a,b);
- With the extended Euclid algorithm to determine x and y s.t. d=ax+by;

The Euclid Algorithm

EUCLID(a, b)
// Input: two integers a and b with $a \geq b \geq 0$
// Output: gcd(a, b)

1. **if** $b = 0$ **then** return a
2. return EUCLID$(b, a \mod b)$

- The Euclid Algorithm to determine $\gcd(a,b)$
  - $a=k_1 b+r_1$ $\quad 0<r_1<b$
  - $b=k_2 r_1 +r_2$ $\quad 0<r_2< r_1$
  - $r_1 = k_3 r_2 +r_3$ $\quad 0<r_3< r_2$
  - ……
  - $r_{n-2} = k_n r_{n-1} +r_n$ $\quad 0<r_n< r_{n-1}$
  - $r_{n-1} = k_{n+1} r_n +r_{n+1}$ $\quad r_{n+1}=0$
- $\gcd(a,b)=\gcd(b, r_1)=\gcd(r_1,r_2)=\ldots= r_n$

The extended Euclid algorithm

EXTENDED-EUCLID(a, b)
// Input: two integers a and b with $a \geq b \geq 0$
// Output: integers x, y, d such that $d = \gcd(a, b)$ and $ax+by = d$

1. **if** $b = 0$ **then** return $(1, 0, a)$
2. $(x', y', d) =$ EXTENDED-EUCLID$(b, a \mod b)$
3. return $(y', x' - \lfloor a/b \rfloor y', d)$

Proof of the correctness $\quad d = \gcd(a, b)$ is by the original Euclid's algorithm.

The rest is by induction on b. The case for $b = 0$ is trivial.
Assume $b > 0$, then the algorithm finds $\gcd(a, b)$ by calling $\gcd(b, a \mod b)$.
Since $a \mod b < b$, we can apply the induction hypothesis on this call and conclude

$$\gcd(b, a \mod b) = bx' + (a \mod b)y'.$$

Writing $(a \mod b)$ as $(a - \lfloor a/b \rfloor b)$, we find

$$\begin{aligned} d = \gcd(a, b) &= \gcd(b, a \mod b) = bx' + (a \mod b)y' \\ &= bx' + (a - \lfloor a/b \rfloor b)y' = ay' + b(x' - \lfloor a/b \rfloor y'). \end{aligned}$$

The (extend) Euclid Algorithm is efficient

Lemma

If $a \geq b \geq 0$, *then* $a \mod b < a/2$.

Proof.

If $b \leq a/2$, then we have $a \mod b < b \leq a/2$; and if $b > a/2$, then $a \mod b = a - b < a/2$.

This means that after any *two consecutive rounds*, both arguments, a and b, are at the very least halved in value, i.e., the length of each decreases by at least one bit.

If they are initially n-bit integers, then the base case will be reached within $2n$ recursive calls. And since each call involves a quadratic-time division, the total time is $O(n^3)$.

Congruence

- If *a* and *b* are integers, we say that *a* is congruent to *b* modulo *m* if $m|(a\text{-}b)$.

  We write $a \equiv b \bmod n$

- $a \equiv a'\ \ (\mathrm{mod}\ m) \Leftrightarrow m \mid (a\text{-}a')$
- $ka \equiv kb\ (\mathrm{mod}\ m)$ not$\Rightarrow$ $a{\equiv}b\ (\mathrm{mod}\ m)$
- If $ka \equiv kb\ (\mathrm{mod}\ m)$ and $\gcd(k,m)=d$， then

$$a \equiv b\ (\mathrm{mod}\ m/d)$$

Modular Inverse

Definition: We say x is the multiplicative inverse of a modulo N if $ax \equiv 1 \bmod N$.

Lemma

There can be at most one such x modulo N with $ax \equiv 1 \bmod N$, denoted by a^{-1} .

Note: inverse does not always exist! For instance, 2 is not invertible modulo 6.

Modular Division

Modular division theorem For any $a \mod N$, a has a multiplicative inverse modulo N if and only if it is relatively prime to N (i.e., $\gcd(a, N) = 1$). When this inverse exists, it can be found in time $O(n^3)$ by running the extended Euclid algorithm.

Example

We wish to compute

$$11^{-1} \mod 25.$$

Using the extended Euclid algorithm, we find $15 \cdot 25 - 34 \cdot 11 = 1$, thus $-34 \cdot 11 \equiv 1 \mod 25$ and $-34 \equiv 16 \mod 25$.

This resolves the issue of modular division: when working modulo N, we can divide by numbers relatively prime to N. And to actually carry out the division, we multiply by the inverse.

Euler Totient Function

Euler Totient Function

$$\phi(m) = \#\{j,\ \gcd(j, m)=1,\ \ 0 \le j \le m\text{-}1\}$$

Exa. $\phi(15)=\#\{1,2,4,7,8,11,13,14\}=8$

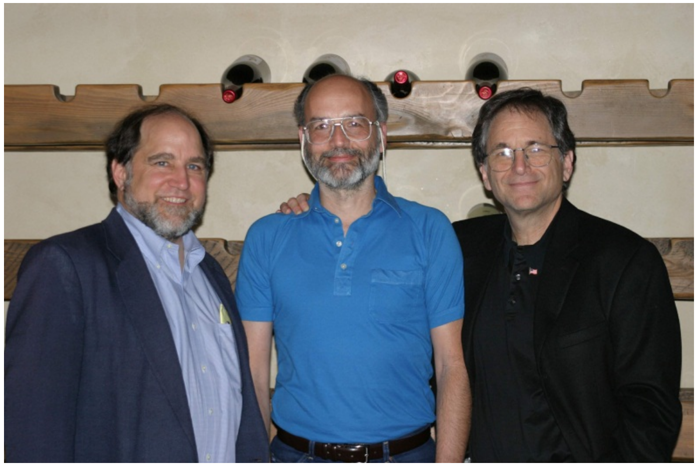
- for p prime, $\varphi(p)= p\text{-}1$, $\varphi(p^k) = p^k - p^{k-1}$
- gcd(a,b)=1, $\varphi(ab)=\varphi(a)\varphi(b)$

- **Euler's Theorem**: if $\gcd(a,m)=1$

  then $a^{\phi(m)} \equiv 1 (\mathrm{mod}\ m)$

- **Fermat's (little) Theorem** : for a prime *p*,
  - if $\gcd(p,a)=1$, then $a^{p\text{-}1} \equiv 1(\mathrm{mod}\ p)$
  - $a^p \equiv a\ (\mathrm{mod}\ p)$

RSA Public Key Cryptosystem



- The Inventors
  - R - Ron Rivest
  - S - Adi Shamir
  - A - Leonard Adleman
- The Trap-Door One-Way Function
  - The exponentiation function $y = f(x) = x^e \bmod n$ can be computed with reasonable effort.
  - Its inverse $x = f^{-1}(y)$ is difficult to compute.
- The Hard Problem Securing the Trap Door
  - based on the hard problem of factoring a large number into its prime factors.

RSA Key Setup

- each user generates a public/private key pair:
  - selecting two large primes at random p, q
  - computing their system modulus $n=p.q$
    - note $\phi(n)=(p-1)(q-1)$
  - selecting at random the encryption key e
    - where $1<e<\phi(n)$, $\gcd(e,\phi(n))=1$
  - solve following equation to find decryption key d
    - $e.d \equiv 1 \bmod \phi(N)$ and $0 \le d \le n$
- publish their public encryption key: PK={e,n}
- keep secret private decryption key: SK={d,p,q}

RSA public-key encryption

- Encrypt with (e, n)
  - ciphertext： $0<M<n$, ciphertext $C \equiv M^{e} \pmod{n}$.
- Decrpt with (d, n)
  - ciphertext： C ciphertext： $M \equiv C^{d} \pmod{n}$

Alice $PK_A = (n_A, e_A)$
$SK_A = (p_A, q_A, d_A)$

Bob $PK_B = (n_B, e_B)$
$SK_B = (p_B, q_B, d_B)$

Get PK_B,
Compute C

$$C = E_{PKB}[M] = (M)^{eB} \bmod n_B$$

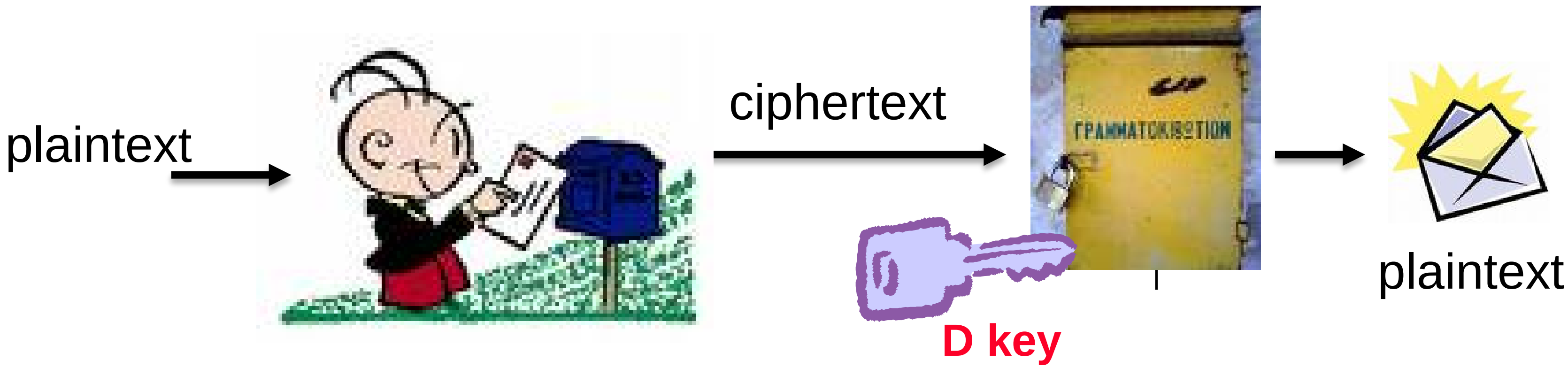
$$\xrightarrow{\qquad\qquad}$$

$$C^d = (M^e)^d = M^{k\phi(n)+1} = M^{k\phi(n)} M = M$$

$$M = E_{SKB}[C] = (C)^{dB} \bmod n_B$$

Confidentiality

- Confidentiality : information is not disclosed to unauthorized individuals, entities, or processes. [ISO]
- Mechanism to achieve confidentiality--Encryption:



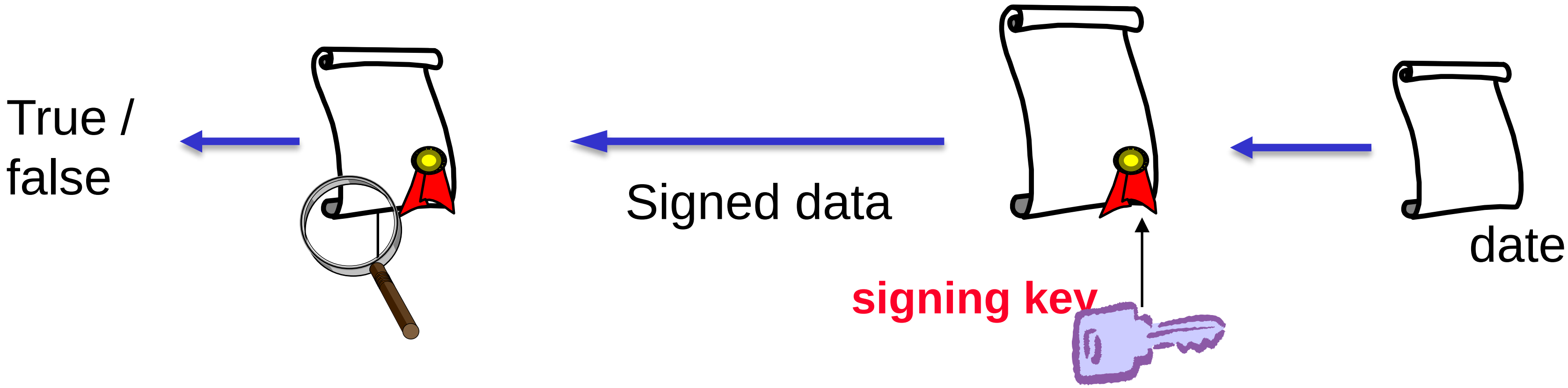
Only the user knowing the decryption key can recover plaintext

–"who can *read* the data"

Authenticity

– Authenticity: assurance of the claimed identity of an entity. [ISO]

– Example: ID-card, password, digital signature



Only the user knowing the secret-key can generate valid signature

"who *wrote* the data"

RSA digital signature

- Parameters PK={e,n} , SK={d,p,q} as before.
- The signature of the message *M* is *S*
  - $S \equiv M^d \pmod{n}$ (signing)
- receiver recover the message
  - $M \equiv S^e \pmod{n}$ (verification)

Alice		Bob
$S \equiv M^{dA} \pmod{n_A}$	$\xrightarrow{S}$	$M \equiv S^{eA} \pmod{n_A}$

Bob verify that only Alice can generate S
--M must be redundant (has clear structure)

RSA digital signature

Alice $PK_A = (n_A, e_A)$
$SK_A = (p_A, q_A, d_A)$

Bob $PK_B = (n_B, e_B)$
$SK_B = (p_B, q_B, d_B)$

Compute $H(M)$
Compute the signature
$S = H(M)^{dA} \bmod nA$

(M, S) →

Get PK_A,

(1) From M, compute $H(M)$
(2) From S, recover
$H(M) = E_{PKA}[S] = (S)^{eA} \bmod n_A$
(3) Check if $H(M) = H(M)$

In real use, a hash function is used to

- **prevent S(xy)=S(x)S(y)**
- **provide redundancy**

RSA digital signature

- M， a public hash function H with domain of $\{0,1,\ldots,n-1\}$。
- Signature

Compute the hash value of M, and get $H(M)\in\{0,1,\ldots,n-1\}$

The input of hash function is of arbitrary length.

Sign $H(M)$ with the private key d, and get

$S \equiv H(M)^d \pmod{n}$

Send (M, S) to the receiver

- Verification

After getting (M,S), recover $V \equiv S^e \pmod{n}$, and verify $V=H(M)$

The trap-door

- For an integer *n=pq*， given *M* and *e*, modular exponentiation $C \equiv M^e \pmod{n}$ is a simple operation；
- Given $C \equiv M^e \pmod{n}$， to find $M \equiv C^{1/e} \pmod{n}$ is a difficult problem；
- When the prime factorization of *n* is known (trapdoor), to find $M \equiv C^{1/e} \pmod{n}$ is easy.

Knowing d $\Leftrightarrow$ knowing the factorization

Cost of factorization

- For currently known algorithms, to complexity of factoring large number n is about

  $\exp(\ b^{1/3} \log^{2/3}(b)\ )$ $b=\log(n)$

- Record:
  - RSA: 768-bit modulo (2010) , RSA 640-bit (2005)
  - Special Numbers: 2^{1039}-1 (2007) , 6^{353}-1 (2006)

- Question: Integer factorization ⇔ Breaking RSA (?)

- Size of n: now 1024-bit (5year?); recommended: 2048-bit

RSA module Length (EMV)

Length	Current Expiry Date	
1024 bits	31 Dec 2009	
1152 bits	31 Dec 2021	
1408 bits	31 Dec 2023	
1984 bits	31 Dec 2023	

2013 recommendation

Parameters of RSA

- length of n is at least 1024 bits
- p and q are large.
- $|p-q|$ is large
- p,q should be random/strong prime numbers.
  $p=2p'+1$, $q=2q'+1$, where p' q' are both primes
- $d>n^{1/4}$
- Public-key e: can be small for efficiency
  - ISO9796 allows 3, (problems?)
  - EDI $2^{16}+1=65537$

Summary

- Public-key cryptosystems:
  - RSA - factorization
  - DH , ElGamal -discrete logarithm
  - ECC
- Math
  - Fermat's and Euler's Theorems & ø(n)
  - Group, Fields
  - Primality Testing
  - Chinese Remainder Theorem
  - Discrete Logarithms

Exercise 7

1. Recall the definition of pseudorandom generaor (PRG): G:$\{0,1\}^n \to \{0,1\}^l$ $(l > n)$ is a PRG if it is polynomial-time computable and for every probabilistic polynomial-time (PPT) D:$\{0,1\}^l \to \{0,1\}$ it holds that

$$|\Pr_{x\leftarrow\{0,1\}^n}\left[D\left(G(x)\right) = 1\right] - \Pr_{y\leftarrow\{0,1\}^l}[D(y) = 1]| < \frac{1}{superpoly(n)}$$

where $x \leftarrow \{0,1\}^n$ denotes sampling x uniformly at random from $\{0,1\}^n$.

Notice that the above D is bounded by running time. Show that this restriction is necessary, i.e., there exists (not necessarily efficient) D such that $|\Pr_{x\leftarrow\{0,1\}^n}\left[D\left(G(x)\right) = 1\right] - \Pr_{y\leftarrow\{0,1\}^l}[D(y) = 1]| \geq 1/2$

Deadline: before next Tuesday (May 5th)

Format: Subject: CS381-yourname-EX.#

Send it to gracehgs@mail.sjtu.edu.cn

Exercise 8

1.Determine the complexity (in terms of the number of arithmetic operations) of

– computing gcd(a,b);

– computing RSA encryption $C=M^e \bmod n$

2. Show that in RSA, knowing $\phi(n)$ is equivalent to knowing the factorization of n

3. For RSA, it requires |p-q| should not be small.

Task: design an attack if |p-q| is smaller than 10000.

Deadline: May 12, 2015 (Next Tuesday)

Send it to : gracehgs@mail.sjtu.edu.cn

Format: Subject: CS381--EX.#-your name