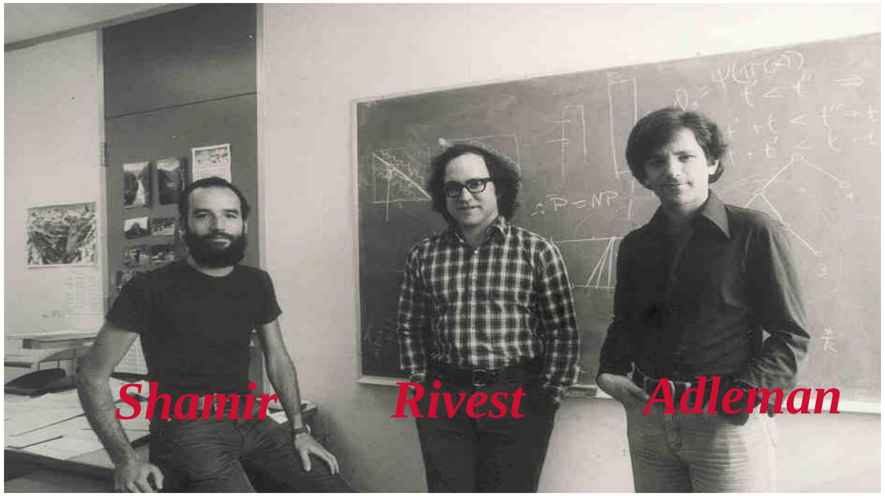
Week 10 -11 : Public Key Cryptosystem and Digital Signatures

1. Public Key Encryptions

RSA, ElGamal,

RSA- PKC(1/3)

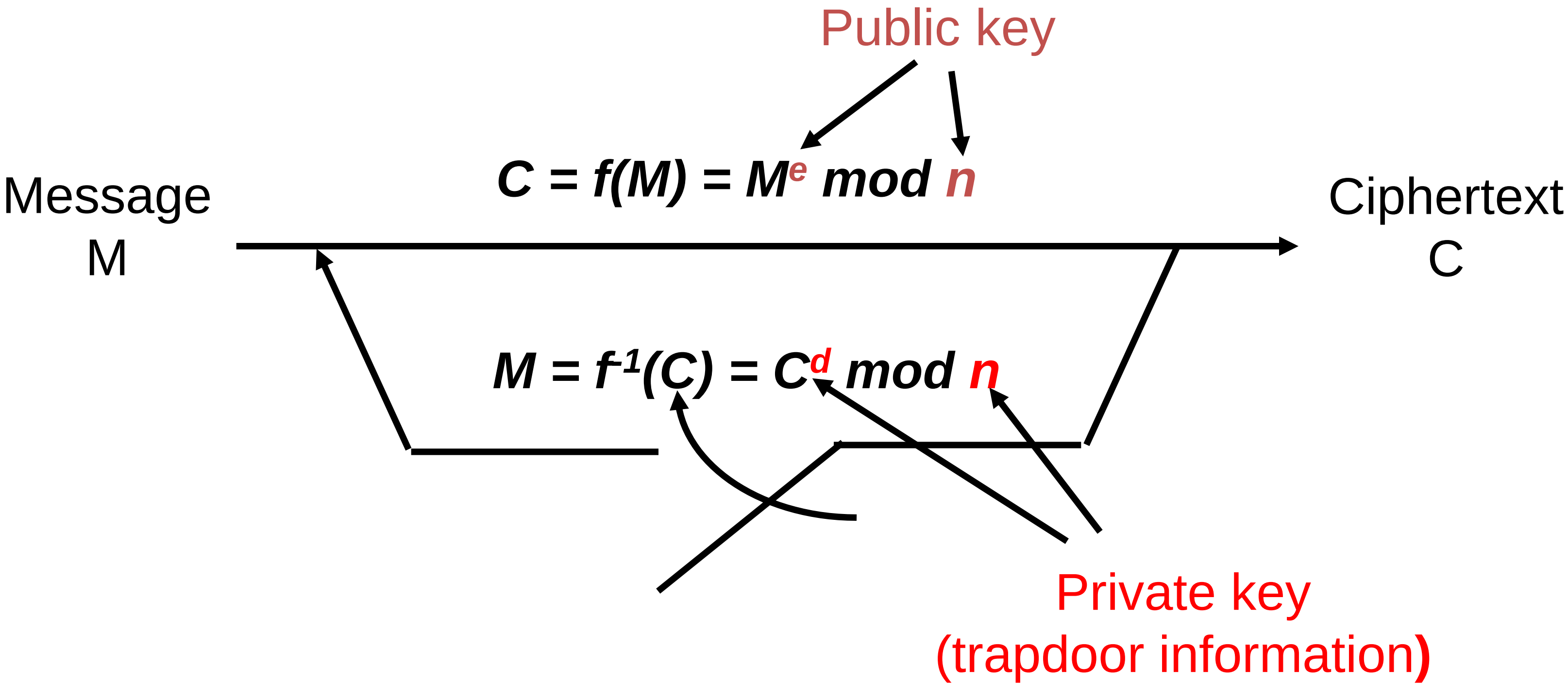
- 1st public key cryptosystem
- R.L.Rivest, A.Shamir, L.Adleman, *"A Method for Obtaining Digital Signatures and Public Key Cryptosystems",* CACM, Vol.21, No.2, pp.120-126,Feb,1978
- Believed to be secure if IFP is hard and worldwide standard for last 30 years



RSA- PKC(2/3)

- Key generation (KeyGen)
  - Select two large (1,024 bits or larger) primes p, q
  - Compute modulus $n = pq$, and $\phi(n) = (p-1)(q-1)$
  - Pick an integer e relatively prime to $\phi(n)$, $gcd(e, \phi(n))=1$
  - Compute d such that $ed = 1 \text{ mod } \phi(n)$ How??
  - Public key (n, e) : public
  - Private key d : keep secret (may hold p and q securely.)
- Encryption(Enc) / Decryption (Dec)
  - E: $C = M^e \text{ mod } n$ for $0 < M < n$
  - D: $M = C^d \text{ mod } n$
  - Proof) $C^d = (M^e)^d = M^{ed} = M^{k\phi(n) +1} = M \{M^{\phi(n)}\}^k = M$
- Special Property
  - $(M^e \text{ mod } n)^d \text{ mod } n = (M^d \text{ mod } n)^e \text{ mod } n$ for $0 < M < n$

RSA as Trapdoor One-way Function



Public key

$C = f(M) = M^e \bmod n$

Message
M

Ciphertext
C

$M = f^{-1}(C) = C^d \bmod n$

Private key
(trapdoor information)

$n = pq$ (p & q: primes)

$ed = 1 \bmod (p-1)(q-1)$

RSA- PKC(3/3)

- Key Generation
  - p=3, q=11
  - $n = pq$ = 33, $\phi(n) =(p-1)(q-1)$ = 2 x10 = 20
  - e = 3 s.t. $gcd(e, \phi(n))$=(3,20)=1
  - Choose d s.t. $ed =1 \bmod \phi(n)$, $3d = 1 \bmod 20$, $d=7$
  - Public key $=\{e,n\}=\{3,33\}$, private key $=\{d\}=\{7\}$
- Encryption
  - $M =5$
  - $C = M^e \bmod n = 5^3 \bmod 33 =26$
- Decryption
  - $M =C^d \bmod n = 26^7 \bmod 33= 5$

Exercise

Let’s practice RSA key generation, encryption, and decryption

1) p=5, q= 7(by hand calculation, Quiz!!) if M= 3
2) p=2,357, q=2,551 (using big number calculator) if M= 5,000
3) p=885,320,963, q=238,855,417 (using big number calculator) if M=10,000

1. Key generation

2. Encryption

3. Decryption

Selecting Primes *p* and *q*

- Idea: Prevent from feasible factorization

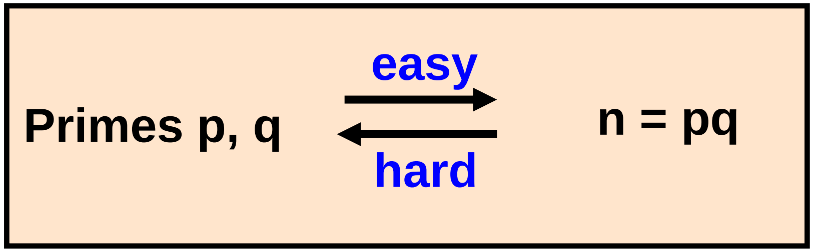
1. $|p| \approx |q|$ to avoid ECM (Elliptic Curve Method for factoring)

2. *p-q* must be large to avoid trial division

3. *p* and *q* are strong prime
    - *p-1* has large prime factor r (Pollard's p-1)
    - *p+1* has large prime factor (William's p+1)
    - *r-1* has large prime factor (Cyclic attack)

Integer Factorization Problem (IFP)

- Problem: Given a composite number *n*, find its prime factors



Primes p, q

easy

hard

n = pq

- Application: Used to construct RSA-type public key cryptosystems

- (Probabilistic sub-exponential) Algorithms to solve IFP
  - Quadratic sieve
  - General Number Field Sieve
  - *etc.*

Quadratic Sieve (1/3)

- Factor n (=pq) using the quadratic sieve algorithm

- Basic principle:
  Let n be an integer and suppose there exist integers x and y with $x^2 = y^2 \pmod{n}$, but $x \neq \pm y \pmod{n}$. Then gcd(x-y, n) gives a nontrivial factor of n.

- Example
  Consider n=77
  72=-5 mod 77, 45=-32 mod 77
  72*45 = (-5)*(-32) mod 77
  $2^3 * 3^4 * 5 = 2^5 * 5 \bmod 77$
  $9^2 = 2^2 \bmod 77$
  gcd(9-2,77)=7, gcd(9+2,77)=11
  77=11*7 Factorization

Quadratic Sieve (2/3)

- Example: factor n=3837523.

Observe

$9398^2 = 5^5 \times 19 \pmod{3837523}$

$19095^2 = 2^2 \times 5 \times 11 \times 13 \times 19 \pmod{3837523}$

$1964^2 = 3^2 \times 13^3 \pmod{3837523}$

$17078^2 = 2^6 \times 3^2 \times 11 \pmod{3837523}$

Then, we have

$(9398 \times 19095 \times 1964 \times 17078)^2 = (2^4 x 3^2 x 5^3 x 11 x 13^2 x 19)^2 \pmod{3837523}$

$2230387^2 = 2586705^2 \pmod{3837523}$

Compute gcd(2230387-2586705, 3837523) => 1093 (mod 3837523)

3837523 / 1093 = 3511 (mod 3837523)

3837523 = 1093 x 3511 ← Note that Verification is easy !!

Quadratic Sieve (3/3)

1. Initialization: a sequence of quadratic residues $Q(x)=(m+x)^2-n$ is generated for small values of x where $m=\lfloor \mathrm{sqrt}(n) \rfloor$.
2. Forming the factor base: the base consists of small primes. $FB=\{-1,2,p_1,p_2,\ldots,p_{t-1}\}$
3. Sieving: the quadratic residues Q(x) are factored using the factor base till t full factorizations of Q(x) have been found.
4. Forming and solving the matrix: Find a linear combination of Q(x)'s which gives the quadratic congruence. The congruence gives a nontrivial factor of n with the probability ½.

http://www.answers.com/topic/quadratic-sieve?cat=technology

General Number Field Sieve (GNFS)

- Most efficient algorithm known for factoring integers larger than 100 digits.
- Asymptotic running time: sub-exponential

$$L_n[\frac{1}{3}, 1.526] = O\left(e^{(1.526+o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$$

Complexity of algorithm

$$L_n[\alpha, c] = O(e^{c(\ln n)^{\alpha}(\ln \ln n)^{1-\alpha}})$$

- If α=0, polynomial time algorithm
- If α>=1, exponential time algorithm
- If 0<α<1, sub-exponential time algorithm

RSA Challenge

Digits	Year		Algorithm
RSA-100	‘91.4.	7	Q.S.
RSA-110	‘92.4.	75	Q.S
RSA-120	‘93.6.	830	Q.S.
RSA-129	‘94.4.(AC94)	5,000	Q.S.
RSA-130	‘96.4.(AC96)	?	NFS
RSA-140	‘99.2 (AC99)	?	NFS
RSA-155	’99.8	8,000	GNFS
RSA-160	’03.1		Lattice Sieving+HW
RSA-174[*2]	’03.12		Lattice Sieving +HW
RSA-200	‘05.5		GNFS+HW

- MIPS : 1 Million Instruction Per Second for 1 yr = 3.1 x 10^{13} instruction.
- *2: 576bit http://www.rsasecurity.com./rsalabs , 768-bit by 2010 (published),
- Expectation: 1,024-bit by 2018 !!!!

RSA-200

- Date: Mon, 9 May 2005 18:05:10 +0200 (CEST)
- From: Thorsten Kleinjung
- Subject: rsa200
- We have factored RSA-200 by GNFS.

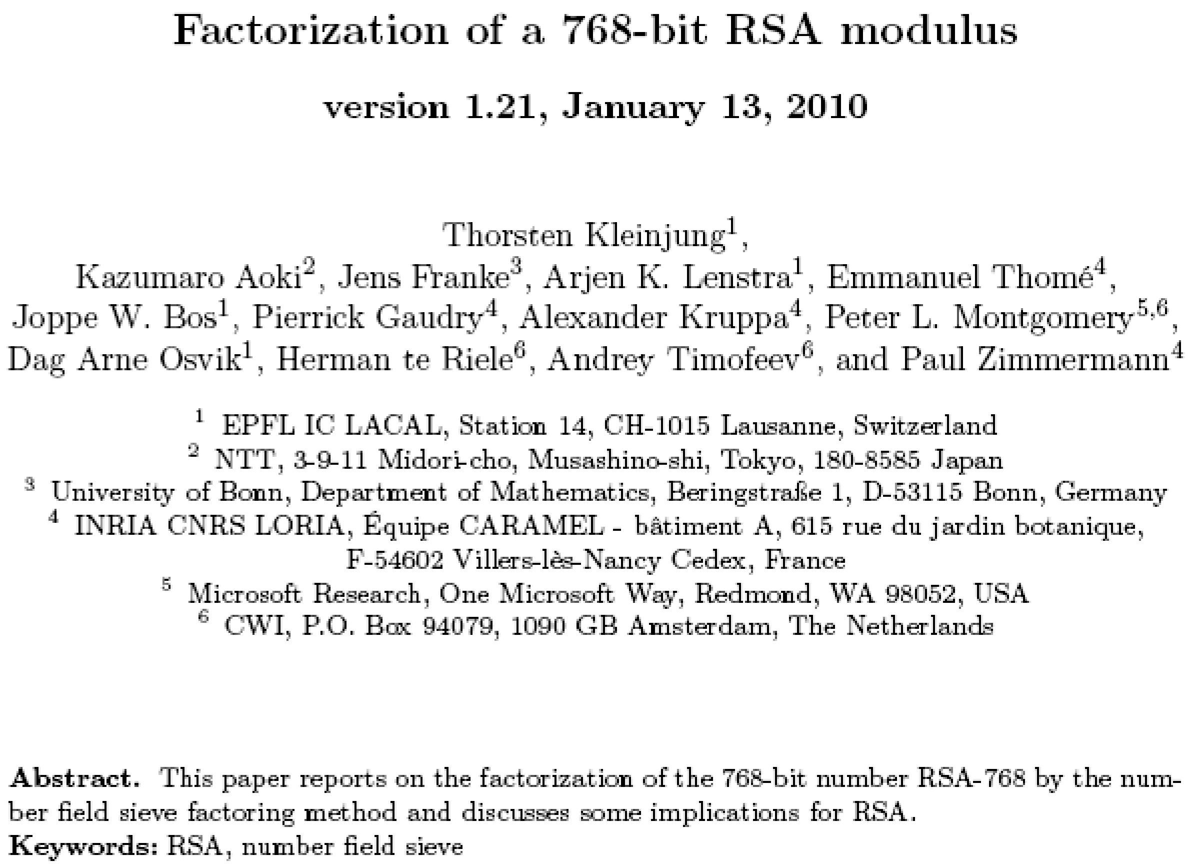
The factors are

p=35324619344027701212726049781984643686711974001976₩
25023649303468776121253679423200058547956528088349 and

q=79258699544783330333470858414800596877379758573642₩
19960734330341455767872818152135381409304740185467

http://www.loria.fr/~zimmerma/records/rsa200

RSA-232 (768 bit)



Factorization of a 768-bit RSA modulus

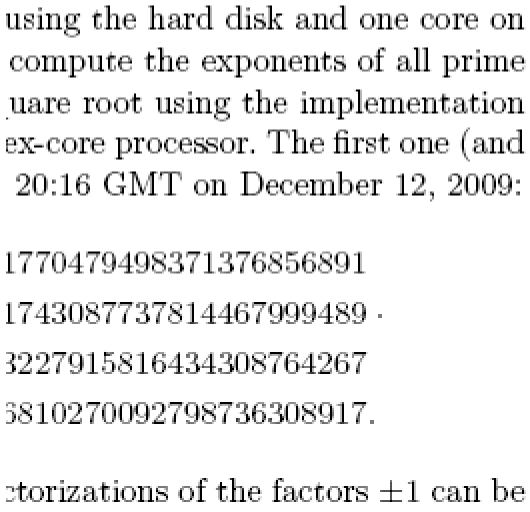
version 1.21, January 13, 2010

Thorsten Kleinjung[1],
Kazumaro Aoki[2], Jens Franke[3], Arjen K. Lenstra[1], Emmanuel Thomé[4],
Joppe W. Bos[1], Pierrick Gaudry[4], Alexander Kruppa[4], Peter L. Montgomery[5,6],
Dag Arne Osvik[1], Herman te Riele[6], Andrey Timofeev[6], and Paul Zimmermann[4]

[1] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
[2] NTT, 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
[3] University of Bonn, Department of Mathematics, Beringstraße 1, D-53115 Bonn, Germany
[4] INRIA CNRS LORIA, Équipe CARAMEL - bâtiment A, 615 rue du jardin botanique,
F-54602 Villers-lès-Nancy Cedex, France
[5] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
[6] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract. This paper reports on the factorization of the 768-bit number RSA-768 by the number field sieve factoring method and discusses some implications for RSA.

Keywords: RSA, number field sieve



using the hard disk and one core on
compute the exponents of all prime
uare root using the implementation
ex-core processor. The first one (and
20:16 GMT on December 12, 2009:

1770479498371376856891

1743087737814467999489 ·

3227915816434308764267

810270092798736308917.

ctorizations of the factors ±1 can be

Security of RSA(1/2)

- Common Modulus attack:
  - If multiple entities share the same modulus $n=pq$ with different pairs of (e_i, d_i), this is not secure.

    Do not share the same modulus!

  - Cryptanalysis: If the same message M was encrypted to different users

    User u_1 : $C_1 = M^{e1} \bmod n$

    User u_2 : $C_2 = M^{e2} \bmod n$

    If $\gcd(e_1,e_2)=1$, there are a and b s.t. $ae_1 + be_2 = 1$ mod n then,

    $(C_1)^a(C_2)^b \bmod n = (M^{e1})^a(M^{e2})^b \bmod n = M^{ae1+be2} \bmod n$

    $= M \bmod n$

Security of RSA(2/2)

- Cycling attack

  If $f(f(\dots f(M)))=f(M)$ where $f(M) = M^e \bmod n$?

  If a given ciphertext appears after some iterations, we can recover the plaintext at collusion point.

  e.g., Let $C=M^e \bmod n$

  If $(((C^e)^e)\dots)^e \bmod n = C^{e\wedge k} \bmod n = C$,

  then $C^{e\wedge(k-1)} \bmod n = M$ for some *k.*

- Multiplicative attack (homomorphic property of RSA)

  $(M_1^e) \times (M_2^e) \bmod n = (M_1 \times M_2)^e \bmod n$

Security of PKC

- Security goals
  - One-wayness (OW): the adversary who sees a ciphertext is not able to compute the corresponding message.
  - Indistinguishability (IND): observing a ciphertext, the adversary learns nothing about the plaintext. Also known as semantic security.
  - Non-malleability (NM): observing a ciphertext for a message *m*, the adversary cannot derive another ciphertext for a meaningful plaintext *m′* related to *m.*

- Original RSA encryption is not secure since
  - IND: deterministic encryption
  - NM: for example, from $c=m^e$, $c' = 2^e c = (2m)^e$ is easily obtained. It cannot be used in bidding scenario.

Formal Definition of IND

$b \in_R \{0,1\}$

m_0, m_1

Challenge: $C=E(m_b)$

PKE(pk, sk)

Guess b?

The adversary wins if he guesses b correctly with a probability significantly greater than ½.

Security Def. of PKC

- Assume the existence of Decryption Oracle
  - Mimics an attacker's access to the decryption device

- Attacking Method
  - Chosen Plaintext Attack (CPA): the adversary can encrypt any plaintext of his choice. In PKC, this is always possible.
  - Non-adaptive Chosen Ciphertext Attack (CCA1): the attacker has access to the decryption oracle before he sees a ciphertext that he wishes to manipulate (aka. lunchtime attack)
  - Adaptive Chosen Ciphertext Attack (CCA2): the attacker has access to the decryption oracle before and after he sees a ciphertext *c* that he wishes to manipulate (but, he is not allowed to query the oracle about the target ciphertext *c*.)

Making RSA to IND-CCA2

- RSA encryption without padding
  - Deterministic encryption
  - Multiplicative property: $m_1^e \, m_2^e = (m_1 m_2)^e \bmod n$
  - Many attacks possible
  - Redundancy checking is required
- RSA encryption with OAEP
  - RSA encryption after OAEP (Optimal Asymmetric Encryption Padding)
  - Proposed by Bellare and Rogaway
  - Probabilistic encoding of message before encryption
  - RSA becomes a probabilistic encryption
  - Secure against IND-CCA2

RSA with OAEP

- OAEP → RSA encryption

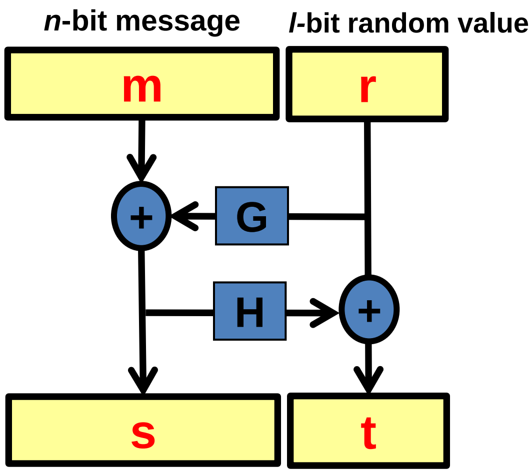
  $s=m\oplus G(r)$
  $t=r\oplus H(s)$ Encryption padding

  $c=E(s,t)$ RSA encryption

- RSA decryption → OAEP

  $(s,t)=D(c)$ RSA decryption

  $r=t\oplus H(s)$
  $m=s\oplus G(r)$ Decryption padding



G
H
Hash function
(Random oracle)

(Note) OAEP looks like a kind of Feistel network
PKCS #1 v2.0, v2.1..

Diffie-Hellman / ElGamal-type Systems

- Domain parameter generation
  - Based on the hardness of DLP
  - Generate a large (1,024 bits or larger) prime p
  - Find generator *g* that generates the cyclic group Z_p^*
  - Domain parameter = {p, g}

- Key generation
  - Pick a random integer $x \in [1, p-1]$
  - Compute $y = g^x \bmod p$
  - Public key (p, g, y) : public Private key x : keep secret

- Applications
  - Public key encryption
  - Digital signatures
  - Key agreement

Discrete Logarithm Problem (DLP)

- Problem:

  Given g, y, and prime p, find an integer x, if any, such that $y = g^x \bmod p$ ($x=\log_g y$)

$$\text{Given } g, x, p \xrightarrow{\text{easy}} y = g^x \bmod p$$

$$x = \log_g y \xleftarrow[\text{hard}]{} \text{Given } g, y, p$$

- Application: Used to construct Diffie-Hellman & ElGamal-type public key systems: DH, DSA, KCDSA …

- Algorithms to solve DLP:
  - Shank's Baby Step Giant Step
  - Index calculus

Shank's Baby Step, Giant Step algorithm

- Problem: find an integer x, if any, such that $y = g^x \bmod p$ ($x=\log_g y$)

- Algorithm

  1. Choose an integer $N = \lceil \sqrt{p-1} \rceil$
  2. Computes $g^j \bmod p, \text{ for } 0 \le j < N$ **Baby Step**
  3. Computes $yg^{-Nk} \bmod p, \text{ for } 0 \le k < N$ **Giant Step**
  4. Look for a match between the two lists. If a match is found,

$$g^j = yg^{-Nk} \bmod p$$

Then $y = g^x = g^{j+Nk}$

We solve the DLP. $\boxed{x = j + Nk}$

Index Calculus (1/2)

- Problem: find an integer x, if any, such that $y = g^x$ mod p ($x=\log_g y$)

- Algorithm
  1. Choose a factor base S = {$p_1, p_2, \ldots p_m$} which are primes less than a bound B.
  2. Collect linear relations
     1. Select a random integer k and compute g^k mod p
     2. Try to write g^k as a product of primes in S

$$g^k = \prod_i p_i^{a_i} \bmod p, \quad \text{then } k = \sum_i a_i \log_g p_i \bmod p-1$$

  3. Find the logarithms of elements in S solving the linear relations
  4. Find x

     For a random r, compute yg^r *mod p* and try to write it as a product of primes in S.

$$yg^r = \prod_i p_i^{b_i} \bmod p, \quad \text{then } x = -r + \sum_i b_i \log_g p_i \bmod p-1$$

Index Calculus (2/2)

- Example: Let p=131, g=2, y=37. Find x=$\log_2 37$ mod 131
- Solution

  Let B=10, S={2,3,5,7}

$2^1 = 2$ mod 131
$2^8 = 5^3$ mod 131
$2^{12} = 5 * 7$ mod 131
$2^{14} = 3^2$ mod 131
$2^{34} = 3 * 5^2$ mod 131

⇨

1 = $\log_2 2$ mod 130
8 = 3*$\log_2 5$ mod 130
12= $\log_2 5$ + $\log_2 7$ mod 130
14 = 2*$\log_2 3$ mod 130
34 = $\log_2 3$ + 2*$\log_2 5$ mod 130

⇨

$\log_2 2 = 1$
$\log_2 5 = 46$
$\log_2 7 = 96$
$\log_2 3 = 72$

37 * 2^{43} = 3 * 5 * 7 mod 131
$\text{Log}_2 37$ = -43 + $\log_2 3$ + $\log_2 5$ + $\log_2 7$ mod 130 = 41

Solution : 2^{41} mod 131 = 37

- Complexity of best known algorithm for solving DLP:

$$L_p[\frac{1}{3}, 1.923] = O\left(e^{(1.923+o(1))(\ln p)^{1/3}(\ln \ln p)^{2/3}}\right)$$

ElGamal Encryption Scheme

- Keys & parameters
  - Domain parameter = {p, g}
  - Choose $x \in [1, p-1]$ and compute $y = g^x \bmod p$
  - Public key (p, g, y)
  - Private key x

- Encryption: $m \rightarrow (C_1, C_2)$
  - Pick a random integer $k \in [1, p-1]$
  - Compute $C_1 = g^k \bmod p$
  - Compute $C_2 = m \times y^k \bmod p$

- Decryption
  - $m = C_2 \times C_1^{-x} \bmod p$
  - $C_2 \times C_1^{-x} = (m \times y^k) \times (g^k)^{-x} = m \times (g^x)^k \times (g^k)^{-x} = m \bmod p$

(Ex.) ElGamal Encryption Scheme

- Key Generation
  - Let p=23, g=7
  - Private key x=9
  - Public key $y = g^x \bmod p = 7^9 \bmod 23 = 15$

- Encryption: m → (C_1, C_2)
  - Let m=20
  - Pick a random number k=3
  - Compute $C_1 = g^k \bmod p = 7^3 \bmod 23 = 21$
  - Compute $C_2 = m \times y^k \bmod p = 20 \times 15^3 \bmod 23 = 20 \times 17 \bmod 23 = 18$
  - Send $(C_1, C_2) = (21,18)$ as a ciphertext

- Decryption
  - $m = C_2 / C_1^x \bmod p = 18 / 21^9 \bmod 23 = 18 / 17 \bmod 23 = 20$

2. Digital Signatures

RSA, ElGamal, DSA, KCDSA, Schnorr

Digital Signature

- When do you use Digital Signature?
  - Electronic version of handwritten signature on electronic document
  - Signing using private key (only by the signer)
  - Verification using public key (by everyone)

- Hash then sign: sig(h(m))
  - Efficiency in computation and communication

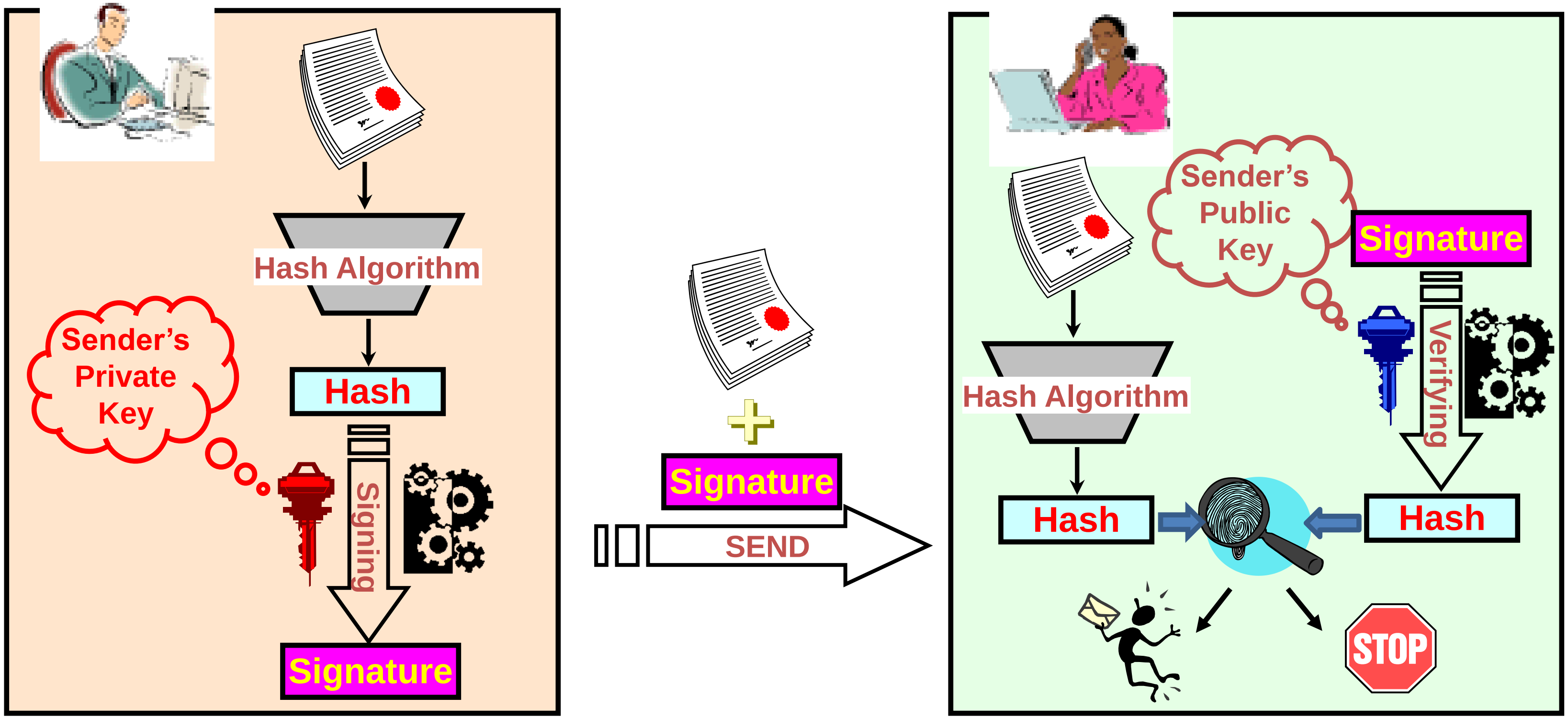
Requirement of DS

- Security requirements for digital signature
  - Unforgeability (위조 방지)
  - User authentication (사용자 인증)
  - Non-repudiation (부인 방지)
  - Unalterability (변조 방지)
  - Non-reusability (재사용 방지)

- Services provided by digital signature
  - Authentication
  - Data integrity
  - Non-Repudiation

Signing & Verification

- ✓ Combine Hash with Digital Signature and use PKC
- ✓ Provide Authentication and Non-Repudiation
- ✓ (Ex.) RSA, ElGamal DSA, KCDSA, ECDSA, EC-KCDSA



Security of Digital Signature

- Forgery
  - Total break: adversary is able to find the secret for signing, so he can forge then any signature on any message.
  - Selective forgery: adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.
  - Existential forgery: adversary can create a pair (message, signature), s.t. the signature of the message is valid.
- Attacking
  - Key-only attack: Adversary knows only the verification function (which is supposed to be public).
  - Known message attack: Adversary knows a list of messages previously signed by Alice.
  - Chosen message attack: Adversary can choose what messages wants Alice to sign, and he knows both the messages and the corresponding signatures.

RSA-Signing

- Key generation
  - Choose two large (512 bits or more) primes p & q
  - Compute modulus $n = pq$, and $\phi(n) = (p-1)(q-1)$
  - Pick an integer e relatively prime to $\phi(n)$, $\gcd(e, \phi(n))=1$
  - Compute d such that $ed = 1 \bmod \phi(n)$
  - Public key (n, e) : publish
  - Private key d : keep secret (may keep p and q securely.)

- Signing / Verifying
  - S: $s = m^d \bmod n$ for $0 < m < n$
  - V: $m =? s^e \bmod n$
  - S: $s = h(m)^d \bmod n$ --- hashed version
  - V: $h(m) =? s^e \bmod n$

- RSA signature without padding
  - Deterministic signature, no randomness introduced

Forging RSA-signature

- RSA signature forgery: Attack based on the multiplicative property of RSA.

  $y_1 = (m_1)^d \quad y_2 = (m_2)^d,$

  then $(y_1y_2)^e = m_1m_2$

  Thus, y_1y_2 is a valid signature of m_1m_2

- This is an existential forgery using a known message attack.

- (HW) RSA-PSS required like RSA-OAEP

ElGamal Signature

- Keys & parameters
  - Domain parameter = {p, g}
  - Choose $x \in [1, p-1]$ and compute $y = g^x \bmod p$
  - Public key (p, g, y)
  - Private key x

- Signature generation: (r, s)
  - Pick a random integer $k \in [1, p-1]$
  - Compute $r = g^k \bmod p$
  - Compute s such that $m = xr + ks \bmod p-1$

- Signature verification
  - $y^r r^s \bmod p =? g^m \bmod p$
    - If equal, accept the signature (valid)
    - If not equal, reject the signature (invalid)

Digital Signature Algorithm (DSA)

Private : x
Public : p, q, g, y

p : 512 ~ 1024-bit prime
q : 160-bit prime, $q \mid p-1$
g : generator of order q
x : $0 < x < q$
$y = g^x \bmod p$

- **Signing**

Pick a random k s.t. $0 < k < q$

$$r = (g^k \bmod p) \bmod q$$
$$s = k^{-1}(\mathrm{SHA1(m)} + xr) \bmod q$$

$m, (r,s)$ ⇒

- **Verifying**

$m, (r,s)$ ⇒

$$w = s^{-1} \bmod q$$
$$u1 = \mathrm{SHA1(m)} \times w \bmod q$$
$$u2 = r \times w \bmod q$$
$$v = (g^{u1} \times y^{u2} \bmod p) \bmod q$$
$$v =? \ r$$

KCDSA

Private : x
Public : p, q, g, y

z=h(Cert_Data)

p : 768+256k (k=0 ~ 5) bit prime
q : 160+32k (k=0~3) bit prime, $q \mid p-1$
g : generator of order q
x : $0 < x < q$
$y = g^{x'} \bmod p,\ x' = x^{-1} \bmod q$

- **Signing**

Pick a random k s.t. $0 < k < q$

$$r = \text{HAS160}(g^k \bmod p)$$
$$e = r \oplus \text{HAS160}(z \,||\, \text{m})$$
$$s = x(k - e) \bmod q$$

$$\xRightarrow{m,\ (r,s)}$$

- **Verifying**

$$\xRightarrow{m,\ (r,s)}$$

$$e = r \oplus \text{HAS160}(z \,||\, \text{m})$$
$$v = y^s \times g^e \bmod p$$
$$\text{HAS160}(v) =?\ r$$

Schnorr Signature Scheme

- Domain parameters
  - p = a large prime (~ size 1024 bit), q = a prime (~size 160 bit)
  - q = a large prime divisor of p-1 (q | p-1)
  - g = an element of Z_p of order q, i.e., $g \neq 1$ & $g^q = 1 \bmod p$
  - Considered in a subgroup of order q in modulo p
- Keys
  - Private key $x \in_R [1, q-1]$: a random integer
  - Public key $y = g^x \bmod p$
- Signature generation: (r, s)
  - Pick a random integer $k \in_R [1, q-1]$
  - Compute $r = h(g^k \bmod p, m)$
  - Compute $s = k - xr \bmod q$
- Signature verification
  - $r =? h(y^r g^s \bmod p, m)$

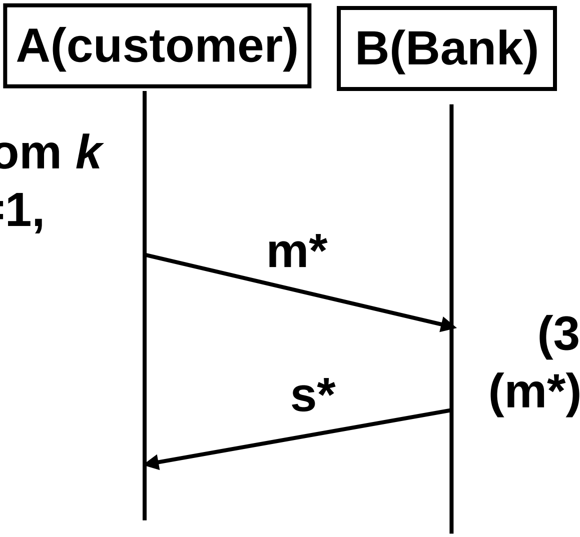
Advanced Digital Signature

- Blind signature
- One-time signature
  - Lamport scheme or Bos-Chaum scheme
- Undeniable signature
  - Chaum-van Antwerpen scheme
- Fail-stop signature
  - van Heyst-Peterson scheme
- Proxy signature
- Group (Ring) signature: group member can generate signature if dispute occurs, identify member. *etc*.

Blind Signature(I)

Without B seeing the content of message M, A can get a signature of M from B.

RSA scheme, B's public key :{n,e}, private key:{d}



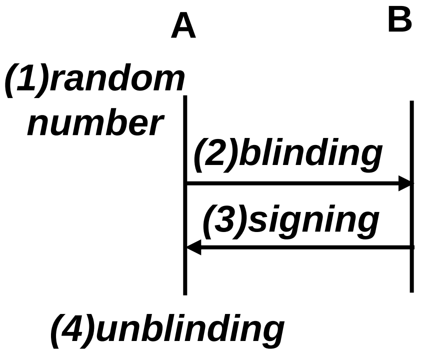
(1) select random *k* s.t. gcd(n,k)=1, 1<k<n-1

(2) $m^*=mk^e$ mod n

(3) $s^* = (m^*)^d \bmod n$

(4) $s=k^{-1} s^* \bmod n$

(signature of M by B : $k^{-1}(mk^e)^d = k^{-1} m^e k^{ed} = m^d$)



$g(S_B f(m))=S_B(m)$

f:blinding ft

g:unblinding ft only A knows

f(m) : blinded message

Blind Signature(II)

(Preparation) p=11, q=3, n=33,$\phi(n)$= 10 x 2=20

gcd(d, $\phi(n)$)=1 => d=3, ed =1 mod $\phi(n)$ => 3 d = 1 mod 20 => e=7

B: public key :{n,e}={33,7}, private key ={d}={3}

(1) A's blinding of m=5

select *k* s.t. gcd(k,n)=1. gcd(*k*,33)=1 => k=2

m* = m k^e mod n= 5 2^7 mod 33 = 640 mod 33 = 13 mod 33

(2) B's signing without knowing the original *m*

s*= $(m^*)^d$ mod n = 13^3 mod 33 =2197 mod 33 =19 mod 33

(3) A's unblinding

s=k^{-1} s* mod n (2 k^{-1}=1 mod 33 => k=17)

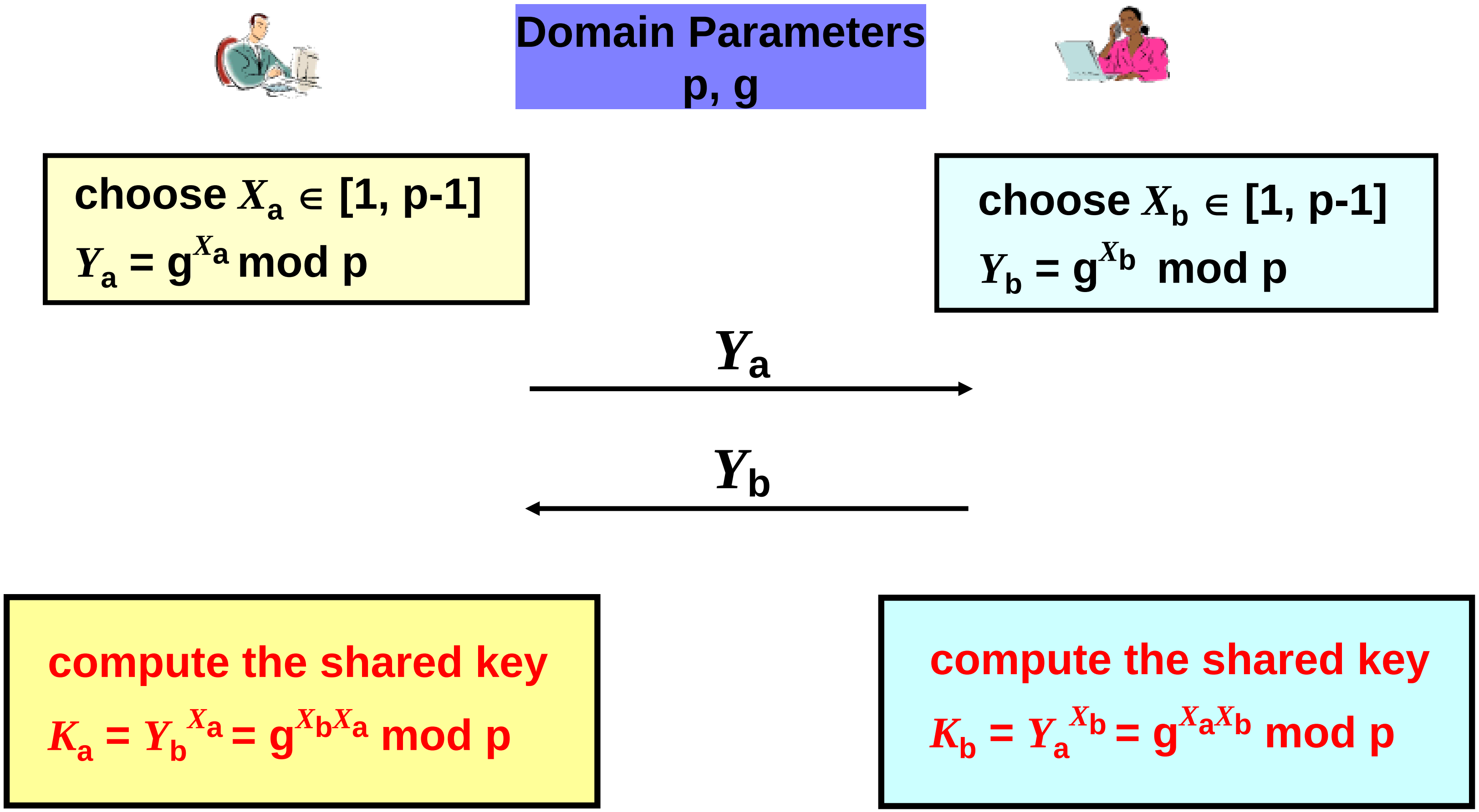
= 17 19 mod 33 =323=26 mod 33

* Original Signature : m^d mod n = 5^3 mod 33 =125 =26 mod 33

3. Key Exchange

Diffie-Hellman

DH Key Agreement



Domain Parameters
p, g

choose $X_a \in [1, p-1]$

$Y_a = g^{X_a} \bmod p$

choose $X_b \in [1, p-1]$

$Y_b = g^{X_b} \bmod p$

Y_a →

← **Y_b**

compute the shared key

$K_a = Y_b^{X_a} = g^{X_b X_a} \bmod p$

compute the shared key

$K_b = Y_a^{X_b} = g^{X_a X_b} \bmod p$

Diffie-Hellman Problem

- **Computational Diffie-Hellman (CDH) Problem**

  Given $Y_a = g^{X_a} \bmod p$ and $Y_b = g^{X_b} \bmod p$,

  compute $K_{ab} = g^{X_a X_b} \bmod p$

- **Decision Diffie-Hellman (DDH) Problem**

  Given $Y_a = g^{X_a} \bmod p$ and $Y_b = g^{X_b} \bmod p$,

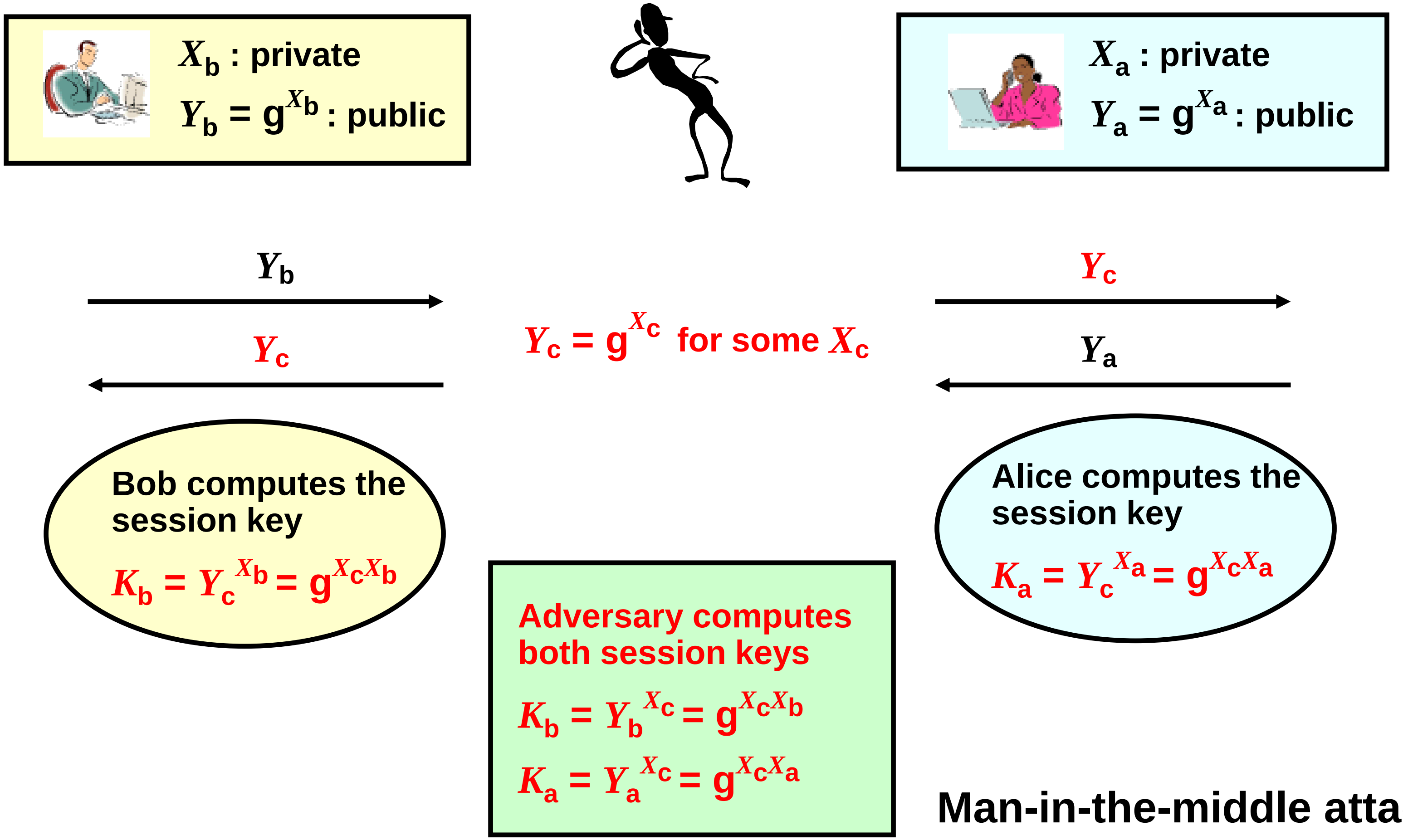
  distinguish between $K_{ab} = g^{X_a X_b} \bmod p$ and a random string

- **Discrete Logarithm Problem (DLP)**

  Given $Y = g^{X} \bmod p$, compute $X = \log_b Y$.

The Security of the Diffie-Hellman key agreement depends on the difficulty of CDH problem.

MIMT in DH Scheme



X_b : private

$Y_b = g^{X_b}$: public

X_a : private

$Y_a = g^{X_a}$: public

Y_b

Y_c

$Y_c = g^{X_c}$ for some X_c

Y_c

Y_a

Bob computes the session key

$K_b = Y_c^{X_b} = g^{X_c X_b}$

Alice computes the session key

$K_a = Y_c^{X_a} = g^{X_c X_a}$

Adversary computes both session keys

$K_b = Y_b^{X_c} = g^{X_c X_b}$

$K_a = Y_a^{X_c} = g^{X_c X_a}$

Man-in-the-middle attack comes from no authentication

DH Key Agreement with Certified Key

Domain Parameters
p, g

choose $X_a \in [1, p-1]$
$Y_a = g^{X_a} \text{ mod p}$

Certified key
Y_a and Y_b

choose $X_b \in [1, p-1]$
$Y_b = g^{X_b} \text{ mod p}$

compute the shared key
$K_a = Y_b^{X_a} = g^{X_b X_a} \text{ mod p}$

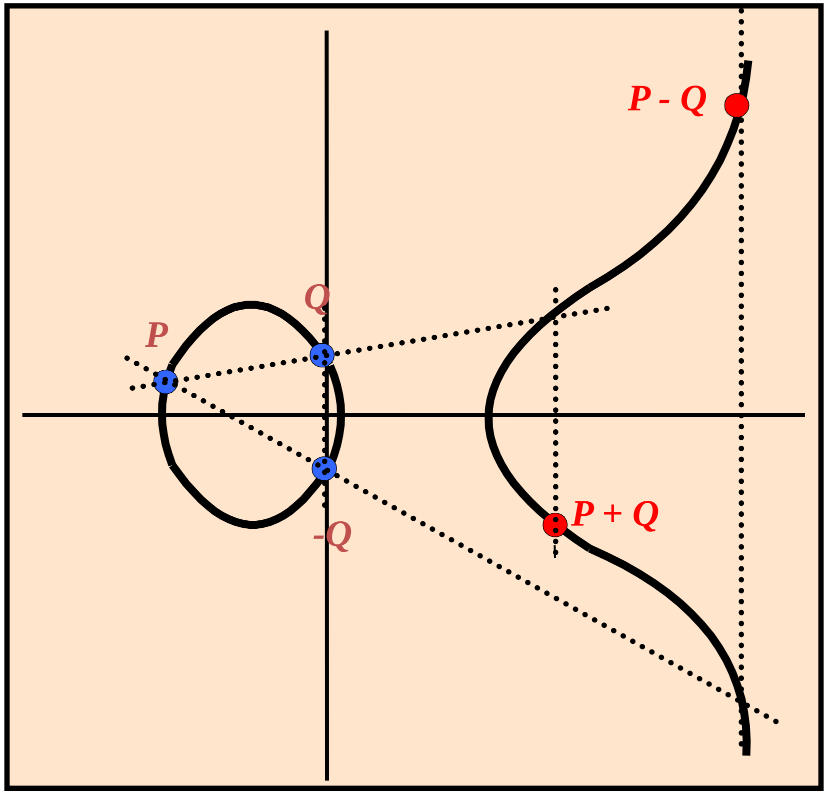
compute the shared key
$K_b = Y_a^{X_b} = g^{X_a X_b} \text{ mod p}$

- **Interaction is not required**
- **Agreed key is fixed, long-term use**

Elliptic Curve (1/2)

- **Weierstrass form of Elliptic Curve**
  - ✓ $y^2 + a_1 xy + a_3 = x^3 + a_2 x^2 + a_4 x + a_6$
- **Example (over rational field)**
  - ✓ $y^2 = x^3 – 4x + 1$
  - ✓ E(Q)

    $= \{(x,y) \in Q^2 \mid y^2 = x^3 – 2x + 2\} \cup O_E$
  - ✓ P = (2, 1),   –P = (2, –1)
  - ✓ [2]P = (12 , -41)
  - ✓ [3]P = (91/25, 736/125)
  - ✓ [4]P = (5452/1681, -324319/68921)



Elliptic Curve (2/2)

- Example (over finite field GF(p) : p = 13)
  - ✓ P = (2,1), –P = (2, 12), [2]P = (12, 11)
  - ✓ [3]P = (0, 1), [4]P = (11, 12), …… , [18]P = O_E
  - ✓ Hasse's Theorem : $p - 2\sqrt{p} \leq \# \text{ of } E(p) \leq p + 2\sqrt{p}$
  - ✓ Scalar multiplication: [d]P
- Elliptic Curve Discrete Logarithm
  - ✓ Base of Elliptic Curve Cryptosystem (ECC)

$$y = g^x \bmod p \quad \Leftrightarrow \quad Q = [d]P$$

Find x for given g, p, Y

Find d for given P, Q

ECC

- Advantages
  - ✓ Breaking PKC over Elliptic Curve is much harder.
  - ✓ We can use much shorter key about 1/6.
  - ✓ Encryption/Decryption is much faster than other PKCs.
  - ✓ Suitable for restricted environments like mobile phone, smart card.

- Disadvantages
  - ✓ It's new technique ➔ There may be new attacks.
  - ✓ Too complicated to understand.
  - ✓ ECC is a minefield of patents.

    : *e.g.*, US patents
    4587627/739220 – Normal Basis, 5272755 – Curve over GF(p)
    5463690/5271051/5159632 – p=2^q-c for small c, *etc*…

Implementation

- RSA Encryption/Decryption

	Encryption	Decryption
PKCS#1-v1.5	1.49 ms	18.05 ms
PKCS#1-OAEP	1.41 ms	18.09 ms

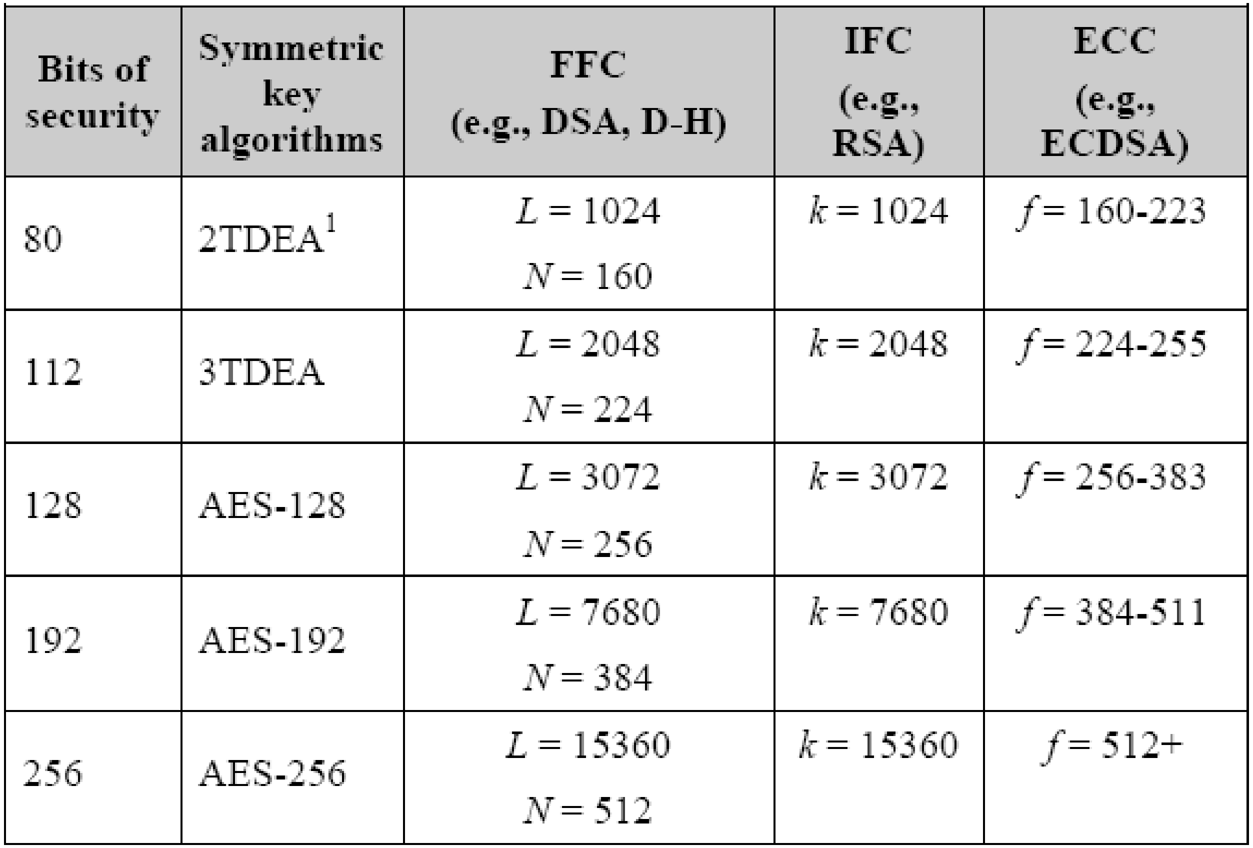
- Signature

	Signing	Verifying
PKCS#1-v1.5	18.07 ms	1.24 ms
PKCS#1-PSS	18.24 ms	1.28 ms
DSA with SHA1	2.75 ms	9.85 ms
KCDSA with HAS160	2.42 ms	9.55 ms

- Modular Exponentiation vs. Scalar Multiplication of EC

M.E. (1024-bit)	S.M. (GF(2^{162}))	S.M. (GF(p))
52.01 ms	2.24 ms	1.17 ms

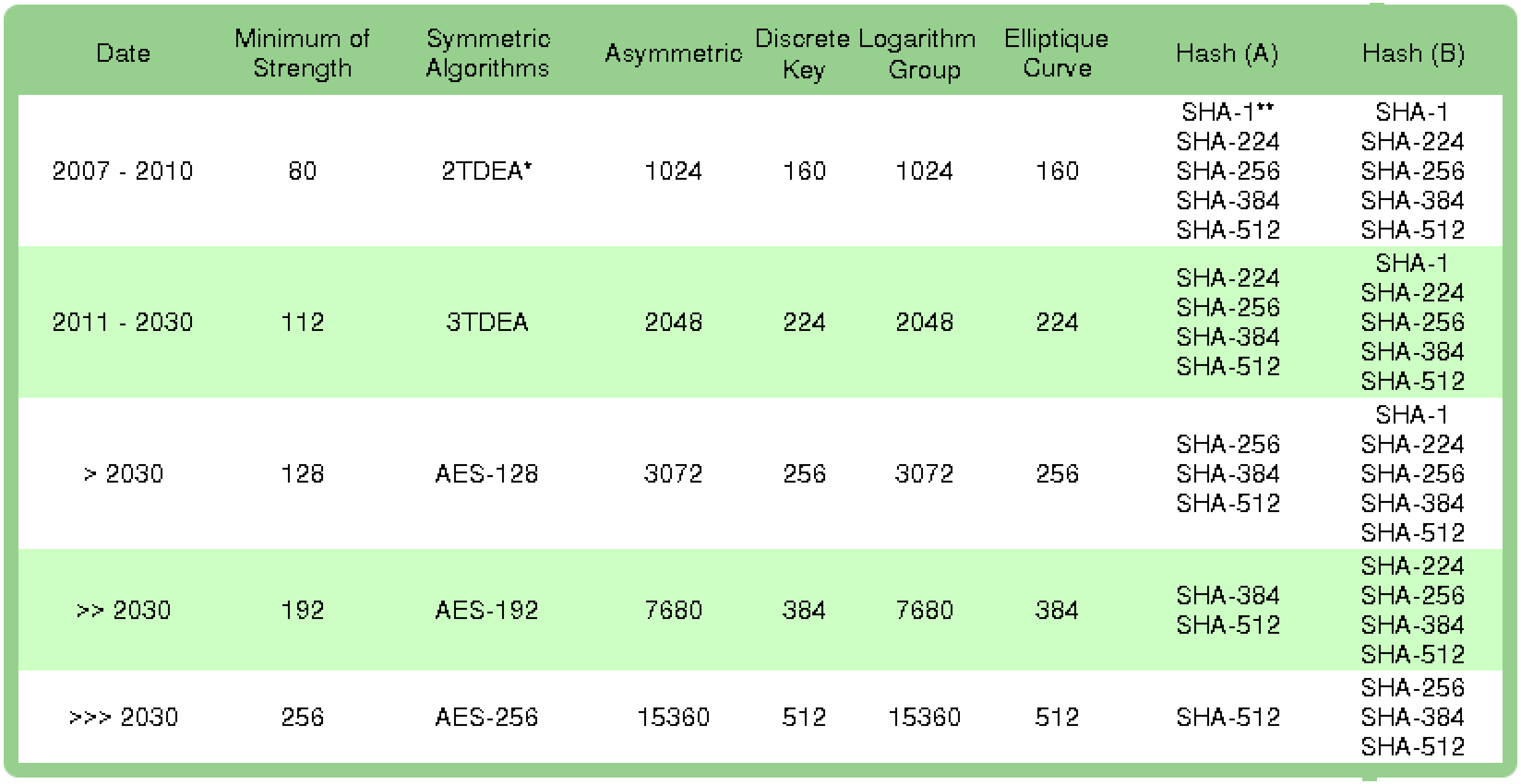
Equivalent Key Size



Bits of security	**Symmetric key algorithms**	**FFC (e.g., DSA, D-H)**	**IFC (e.g., RSA)**	**ECC (e.g., ECDSA)**
80	2TDEA[1]	$L = 1024$ $N = 160$	$k = 1024$	$f = 160\text{-}223$
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224\text{-}255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256\text{-}383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384\text{-}511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

Recommendation for the Transition of Cryptographic Algorithm and Key Sizes, NIST800-121, Jan. 2010.

Key Length by NIST



Date	Minimum of Strength	Symmetric Algorithms	Asymmetric	Discrete Logarithm		Elliptique Curve	Hash (A)	Hash (B)
				Key	Group			
2007 - 2010	80	2TDEA*	1024	160	1024	160	SHA-1** SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
2011 - 2030	112	3TDEA	2048	224	2048	224	SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
> 2030	128	AES-128	3072	256	3072	256	SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
>> 2030	192	AES-192	7680	384	7680	384	SHA-384 SHA-512	SHA-224 SHA-256 SHA-384 SHA-512
>>> 2030	256	AES-256	15360	512	15360	512	SHA-512	SHA-256 SHA-384 SHA-512

Recommendation for Key Management,
Special Publication 800-57 Part 1, NIST, 03/2007. http://www.keylength.com