RSA

Public Key CryptoSystem

DIFFIE AND HELLMAN (76)
"NEW DIRECTIONS IN CRYPTOGRAPHY"

- Split the Bob's secret key K to two parts:
- K_E , to be used for encrypting messages to Bob.
- K_D , to be used for decrypting messages by Bob.

- K_E can be made public
  - (public key cryptography, asymmetric cryptography)

Integer Multiplication & Factoring as a One Way Function.

$$\begin{tikzcd} p,q \arrow[rr, bend left, "\text{easy}"] & & N=pq \arrow[ll, bend left, "\text{hard}"] \end{tikzcd}$$

Q.: Can a public key system be based on this observation ?????

Excerpts from RSA paper (CACM, 1978)

- The era of "electronic mail" may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

- At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem," an elegant concept invented by Diffie and Hellman. Their article motivated our research, since they presented the concept but not any practical implementation of such system.

The Multiplicative Group Zpq*

Let p and q be two large primes.
Denote their product $N = pq$

The multiplicative group $Z_N^* = Z_{pq}^*$ contains all integers in the range $[1, pq-1]$ that are relatively prime to *both* p and q. [Prove it]

Since in $[1, pq-1]$ there $(p-1)$ multiples of q and $(q-1)$ multiples of p, the size of the group is $\phi(pq) = pq-1-(p-1)-(q-1)=pq-(p+q)+1=(p-1)(q-1)$, so (by Euler's theorem)
for every $x \in Z_{pq}^*$, $x^{(p-1)(q-1)} = 1$.

Exponentiation in Z_{pq}^*

Motivation: We want to exponentiate for encryption.

Note that not all integers in $\{1,2,\ldots pq-1\}$ belong to Z_{pq}^*. These elements do not necessarily have a unique inverse in Z_{pq}^* (in fact multipl. is not a one-to-one mapping)

Let e be an integer, $1 < e < (p-1)(q-1)$.

Question: When is exponentiation to the e-th power, $x \dashrightarrow x^e$, a one-to-one oper. in Z_{pq}^* ?

EXPONENTIATION IN Z_{pq}^*

Claim: If e is relatively prime to $(p-1)(q-1)$ then $x \dashrightarrow x^e$ is a one-to-one op in Z_{pq}^*

Constructive proof: Since $\gcd(e, (p-1)(q-1))=1$, e has a multiplicative inverse mod $(p-1)(q-1)$. Denote it by d, then $ed=1 + C(p-1)(q-1)$, C constant.

Let $y=x^e$, then $y^d=(x^e)^d=x^{1+C(p-1)(q-1)}=$
$x^1 x^{C(p-1)(q-1)}=x(x^{(p-1)(q-1)})^C = x\cdot 1 = x$

meaning $y \dashrightarrow y^d$ is the inverse of $x \dashrightarrow x^e$ QED

RSA Public Key Cryptosystem

- Let $N = pq$ be the product of two primes
- Choose $1 < e < \phi(N)$ such that $\gcd(e, \phi(N)) = 1$
- Let d be such that $de \equiv 1 \bmod \phi(N)$
- The public key is (e, N)
- The private key is (d, N)

- Encryption of $M \in Z_N$ by $C = E(M) = M^e \bmod N$
- Decryption of $C \in Z_N$ by $M = D(C) = C^d \bmod N$

The above mentioned method should not be confused with the exponentiation technique presented by Diffie and Hellman to solve the key distribution problem".

WHY DECRYPTION WORKS

- since $ed \equiv 1 \pmod{\phi}$, there is integer k s.t. $ed = 1 + k\,\phi$, $\phi = (p - 1)(q - 1)$
- if $\gcd(m, p) = 1$ then (by Fermat) $m^{p-1} \equiv 1 \pmod{p}$.
  - raise both sides to the power $k(q-1)$ and then multiply both sides by m, and get $m^{1+k(p-1)(q-1)} \equiv m \pmod{p}$
- if $\gcd(m, p) = p$ (no other possibilities!) then the congruence still holds because both sides congruent to 0 mod p, because $m = jp$, for some $j \geq 1$ (hence m is multiple of p)
- hence, in both cases, $m^{ed} \equiv m \pmod{p}$
- similarly, $m^{ed} \equiv m \pmod{q}$
- since $\gcd(p, q) = 1$, it follows $m^{ed} \equiv m \pmod{N}$

RSA

- key length is variable
  - the longer, the higher security
  - 512 bits is common
- block size also variable
  - but $|plaintext| \le |N|$ (in practice, for avoiding weak cases, $|plaintext| < |N|$)
  - $|ciphertext| = |N|$
- slower than DES
  - not used in practice for encrypting long messages
  - mostly used to encrypt a secret key, then used for encrypting the message

A Small Example

- Let $p = 47$, $q = 59$, $N = pq = 2773$. $\phi(N) = 46 \times 58 = 2668$.
- Pick $d = 157$ (gcd(2668, 157) = gcd(157, 156) = gcd(156, 1) = gcd(1, 0) = 1), then $157 \times 17 - 2668 = 1$, so $e = 17$ is the inverse of 157 mod 2668 [see next slide for details]
- For $N = 2773$ we can encode two letters per block, using a two digit number per letter:

  blank = 00, A = 01, B = 02, …, Z = 26
- Message: ITS ALL GREEK TO ME is encoded

0920 1900 0112 1200 0718 0505 1100 2015 0013 0500

COMPUTING THE MULTIPLICATIVE INVERSE

- $p = 47$, $q = 59$, $N = pq = 2773$, $\phi(N) = 46 \times 58 = 2668$; choose $d = 157$
- Bézout's identity: $au + bv = d$ ($a \geq b$; u, v signed integers; d greatest common divisor of a and b) + extended Euclid's alg.
  - if a and b coprime then $d = 1$, u is the multiplicative inverse of a (mod b) and v is the multiplicative inverse of b (mod a)
- $-1 \times 2668 + 17 \times 157 = 1$ (Bézout)

THE EXTENDED EUCLID'S ALGORITHM

- given integer x, y s.t. gcd(x,y)=1, find the multiplicative inverse of x (mod y)
  - assume wlog $x \geq y$
- traditional Euclid's alg. for computing gcd(x,y) calculates $r_n = r_{n\text{-}2} \,\%\, r_{n\text{-}1}$, with $r_{\text{-}2} = x$, $r_{\text{-}1} = y$
- when $r_n = 0$ gcd has been found, equal to $r_{n\text{-}1}$
- Bézout identity: there exist u, v s.t. $ux + vy = 1$
- the multiplicative inverse of x (mod y) is a number x^{-1} such that $x^{-1}x \equiv 1$ (mod y), hence, $x^{-1}x - 1 = ky$, for some integer k, that is $x^{-1}x - ky = 1$, or $x^{-1}x + vy = 1$, for some integer v (with $v = -k$)

The extended Euclid's algorithm (2)

- we can extend Euclid's algorithm to compute signed integers u_n and v_n s.t., for each n:
  $u_n x + v_n y = r_n$

(constructive) proof by induction

- let $u_{-2} = 1$, $v_{-2} = 0$, $u_{-1} = 0$, $v_{-1} = 1$
- let $r_n = r_{n-2} - r_{n-1}q_n$, $q_n = \lfloor r_{n-2} / r_{n-1} \rfloor$
- if we set $u_n = u_{n-2} - q_n u_{n-1}$ and $v_n = v_{n-2} - q_n v_{n-1}$, since (by inductive HP) $r_{n-1} = u_{n-1}x + v_{n-1}y$ and $r_{n-2} = u_{n-2}x + v_{n-2}y$, if we compute $r_n = r_{n-2} - r_{n-1}q_n =$
  $= u_{n-2}x + v_{n-2}y - q_n(u_{n-1}x + v_{n-1}y)$ we get
  $r_n = (u_{n-2} - q_n u_{n-1})x + (v_{n-2} - q_n v_{n-1})y$, namely
  $r_n = u_n x + v_n y$ QED

The extended Euclid's algorithm (3)

- since we halt the algorithm when $r_n = 0$ and $\gcd(x, y) = r_{n\text{-}1} = 1$ then:

$$r_{n\text{-}1} = u_{n\text{-}1}\, x + v_{n\text{-}1}\, y = 1$$

- hence $x^{-1} = u_{n\text{-}1}$ (unit is unique), and $y^{-1} = v_{n\text{-}1}$

n	r_n	q_n	u_n	v_n
-2	2668		1	0
-1	157		0	1
0	156	16	1	-16
1	1	1	-1	17
2	0	156	157	-2668

A Small Example

- $N = 2773$, $e = 17$ (10001 in binary)
- ITS ALL GREEK TO ME is encoded as

0920 1900 0112 1200 0718 0505 1100 2015
0013 0500

- First block $M = 0920$ encrypts to

$M^e = M^{17} = (((M^2)^2)^2)^2 \times M = 948 \pmod{2773}$

- The whole message (10 blocks) is encrypted as

0948 2342 1084 1444 2663 2390 0778 0774 0219
1655

- Indeed $0948^d = 0948^{157} = 948^{1+4+8+16+128} = 920$ (mod 2773), etc.

QUICK EXPONENTIATION (C CODE)

```
static long fastExp(int base, int exp) {
    long f = 1;
    long b = base;
    while(exp > 0) {
        int lsb = 0x1 & exp;
        exp >>= 1;
        if(lsb) f *= b;
        b *= b;
    }
    return f;
}
```

Add mod p as needed

CONSTRUCTING AN INSTANCE OF RSA

- Alice first picks at random two large primes, p and q
- Let $N = p \cdot q$ be the product of p and q
- Alice then picks at random a large d that is relatively prime to $\phi(N) = (p\text{-}1)(q\text{-}1)$ ($\gcd(d, \phi(N)) = 1$)
- Alice computes e such that $d \cdot e \equiv 1 \bmod \phi(\mathrm{N})$

- Alice publishes the public key (N, e)
- Alice keeps the private key (N, d), as well as the primes p, q and the number $\phi(N)$, in a safe place

RSA: IMPLEMENTATION

1. Find p and q, two large primes
   - Random (an adversary should not be able to guess these numbers; they should be different each time a new key is defined)
2. Choose suitable e to have a fast encoding
   - Use exponentiation algorithm based on repeated squaring:
     - Compute power of 2, 4, 8, 16, …
     - Compute power e by using the binary encoding of e and powers computed in so far (ex.: if $e = 3$ then 2 multiplications needed; if $e = 2^{16} + 1$ then 5 multiplications needed)
   - In this way decoding is generally slower (d is large)
3. Compute d:
   - *Euclid's extended algorithm*

RSA: IMPLEMENTATION (STEP 1)

1. Find two random primes

Algorithm:

- randomly choose a random large odd integer
- test if it is a prime

Note:

- Prime number are relatively frequent (in $[N, 2N]$ there are $\approx N / \ln N$ primes)
  - prime number theorem (http://en.wikipedia.org/wiki/Prime_number_theorem) states that the average gap between prime numbers near N is roughly $\ln(N)$
- Hence randomly choosing we expect to find a prime of N bits every $\ln N$ attempts

RSA: IMPLEMENTATION (STEP 2)

2. Coding algorithm (compute exponentiation)

Compute power by repeated squaring (so computing power of 2, 4, 8, ..) and then executing multiplication (based on binary encoding of the exponent e)

Cost: O(log N) operations

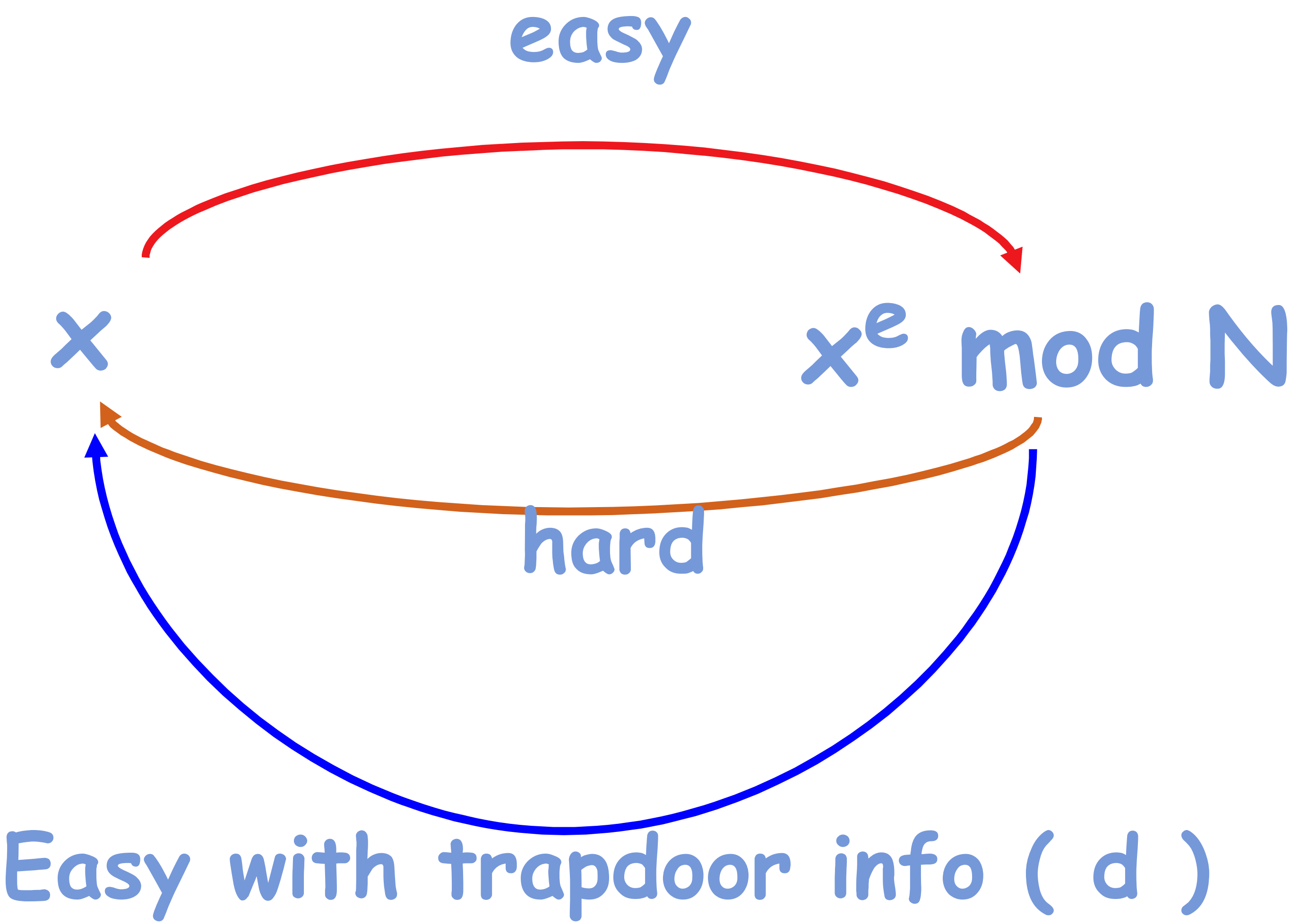
Constant no. of operations if *e* is small and its binary representation has few ones

Examples: e= 3, 2 multiplications

e= 65537= 2^{16} +1, compute powers M^2, M^4, M^8, M^{16},.. M^{65536} and then M^{65537} =M^{65536}* * M

(total 16 multiplication)

RSA as a One Way Trapdoor Function.



Trap-Door OWF

- Definition: f: $D \rightarrow R$ is a *trapdoor one way function* if there is a trapdoor s such that:
  - Without knowledge of s, the function f is a one way function
  - Given s, inverting f is easy
- Example: $f_{g,p}(x) = g^x \bmod p$ is not a trap-door one way function.
- Example: RSA is a trap-door OWF.

RSA: A COLLECTION OF TRAP-DOOR OWF

- Note: RSA defines a function that depends on the Key
- Definition: Let I be a set of indices and A a finite set. A collection of trap door one-way functions is a set of functions F parameterized by elements in I
- F verifies: For all values i in I there exists f_i such that
  - f_i *belongs to F*
  - $f_i: D \rightarrow R_i$ is a *trap-door one way function*
- Idea: we need an algorithm that given a security parameter (the key) selects a function in F together with a trapdoor information

ATTACKS ON RSA

Recall RSA robustness does not imply it is always robust

1. Factor $N = pq$. This is believed hard unless p, q have some "bad" properties. To avoid such primes, it is recommended to
   - Take p, q large enough (100 digits each).
   - Make sure p, q are not too close together.
   - Make sure both (p-1), (q-1) have large prime factors (to foil Pollard's rho algorithm)
     - special-purpose integer factorization algorithm (John Pollard, 1975); particularly effective at splitting composite numbers with small factors.
2. Some messages might be easy to decode

RSA AND FACTORING

- **Fact 1**: given n, e, p and q it is easy to compute d
- **Fact 2**: given n, e
  - if you factor n then you can compute $\phi(n)$ and d

- **Conclusion**:
  - If you can factor n then you break RSA
  - If you invert RSA then you are able to factor n?
    OPEN PROBLEM

FACTORING RSA CHALLENGES

RSA challenges: factor $N = pq$:

RSA Security has published factoring challenges

- RSA 426 bits, 129 digits: factorized in 1994 (8 months, 1600 computers in the Internet (10000 Mips))
- RSA 576 bits, 173 digits: factorized in dec. 2003, $10000
- RSA 640 bits, nov. 2005, 30 2.2GHz-Opteron-CPU
- Challenge is not active anymore (2007)

June 2008: Kaspersky Labs proposed an international distributed effort to crack a 1024-bit RSA key used by the Gpcode Virus. From their website: *We estimate it would take around 15 million modern computers, running for about a year, to crack such a key. (??)*

RSA - Attacks

- There are messages easy to decode:
  if $m = 0, 1, n\text{-}1$ then $\mathrm{RSA}(m) = m$
  - notice e must be odd ≥ 3, hence $(n\text{-}1)^e \bmod n = n\text{-}1$, because $(n\text{-}1)^2 \bmod n = 1$

  SOLUT: rare, use salt
- If both m and e are small (e.g. $e = 3$) then we might have $m^e < n$; hence

  $$m^e \bmod n = m^e$$

  compute eth root in arithmetic is easy: adversary compute it and finds m

  SOLUT. Add non zero bytes to avoid small messages

RSA - ATTACKS

Small e (e.g. $e = 3$)

- Suppose adversary has two encoding of similar messages such as

  $$c_1 = m^3 \bmod n \text{ and } c_2 = (m+1)^3 \bmod n$$

  Therefore

  $$m = (c_2 + 2\, c_1 - 1) / (c_2 - c_1 + 2)$$

- The case m and $(am + b)$ is similar (see next slide)

SOLUT. Choose large e

Low-Exponent RSA with Related Messages

Don Coppersmith* Matthew Franklin** Jacques Patarin*** Michael Reiter†

Abstract. In this paper we present a new class of attacks against RSA with low encrypting exponent. The attacks enable the recovery of plaintext messages from their ciphertexts and a known polynomial relationship among the messages, provided that the ciphertexts were created using the same RSA public key with low encrypting exponent.

Suppose we have two messages m_1 and m_2 related by a known affine relation

$$m_2 = \alpha m_1 + \beta.$$

Suppose further that the messages are encrypted under RSA with an exponent of 3 using a single public modulus N.

$$c_i = m_i^3 \bmod N, \quad i = 1,2$$

Eurocrypt 1996

Then from

$$c_1, c_2, \alpha, \beta, N$$

we can calculate the secret messages m_i algebraically as follows:

$$\frac{\beta(c_2 + 2\alpha^3 c_1 - \beta^3)}{\alpha(c_2 - \alpha^3 c_1 + 2\beta^3)} = \frac{3\alpha^3\beta m_1^3 + 3\alpha^2\beta^2 m_1^2 + 3\alpha\beta^3 m_1}{3\alpha^3\beta m_1^2 + 3\alpha^2\beta^2 m_1 + 3\alpha\beta^3} = m_1 \bmod N.$$

The algebra is more transparent if we assume (without loss of generality) that $\alpha = \beta = 1$.

$$\frac{(m+1)^3 + 2m^3 - 1}{(m+1)^3 - m^3 + 2} = \frac{3m^3 + 3m^2 + 3m}{3m^2 + 3m + 3} = m \bmod N. \qquad (1)$$

Chinese remainder theorem

- Suppose n_1, n_2, …, n_k are positive integers which are pairwise coprime. Then, for any given integers a_1, a_2, …, a_k, there exists an integer x solving the system of simultaneous congruencies

$$x \equiv a_i \pmod{n_i} \quad \text{for } i = 1, \ldots, k.$$

- Furthermore, all solutions x to this system are congruent modulo the product $N = n_1 n_2 \ldots n_k$.

RSA - ATTACKS

Assume $e = 3$ and then send the same message three times to three different users, with public keys

$$(3, n_1)\ (3, n_2)\ (3, n_3)$$

Attack: adv. knows public keys and

$m^3 \bmod n_1,\ m^3 \bmod n_2,\ m^3 \bmod n_3$

- He can compute (using Chinese remainder theorem) $m^3 \bmod (n_1 \cdot n_2 \cdot n_3)$
- Moreover $m < n_1, n_2, n_3$; hence $m^3 < n_1 \cdot n_2 \cdot n_3$ and

$$m^3 \bmod n_1 \cdot n_2 \cdot n_3 = m^3$$

- Adv. computes cubic root and gets result

SOLUT. Add random bytes to avoid equal messages

RSA - ATTACKS

- If message space is small then adv. can test all possible messages

  ex.: adv. knows encoding of m and knows that m is either m1=10101010 or m2=01010101

  adv encodes m1 and m2 using public key and verifies

- SOLUT. Add random string in the message

RSA - Attacks

- If two user have same n (but different e and d) then bad.
  - One user could compute p and q starting from his e, d, n (somewhat tricky, based on Euler's theorem – see for instance "The Handbook of Applied Cryptography")
  - Then, the user could easily discover the secret key of the other user, given his public key

SOLUT. Each person chooses his own n (the probability two people choose same n is very very low)

RSA - ATTACKS

- Multiplicative property of RSA:
  - If $M = M1 * M2$ then $(M1 * M2)^e \bmod N = ((M1^e \bmod N) * (M2^e \bmod N)) \bmod N$

  Hence an adversary can proceed using small messages (chosen ciphertext, see next slide)

  - Can be generalized if $M = M1 * M2 * \ldots * Mk$
  - Solut.: padding on short messages

RSA - CHOSEN CIPHERTEXT ATTACK

Adversary wants to decrypt $C = M^e \bmod n$

1. adv. computes $X = (C \cdot 2^e) \bmod n$
2. adv. uses X as chosen ciphertext and asks the oracle for $Y = X^d \bmod n$

but….

$X = (C \bmod n) \cdot (2^e \bmod n) = (M^e \bmod n) \cdot (2^e \bmod n) =$
$= (2M)^e \bmod n$

thus adv. got $Y = (2M)$

RSA - ATTACKS

chosen ciphertext attack:

- Adv. T knows $c = M^e \bmod n$
- T randomly chooses X and computes $c' = c X^e \bmod n$ and ASKS ORACLE TO DECODE c'
- T computes $(c')^d = c^d (X^e)^d = M X \bmod n$!!

- SOLUT: require that messages verify a given structure (oracle does not answers if M does not verify requirements)

CHOSEN PLAINTEXT ATTACK (CPA)

- Attack model which presumes that attacker has capability to choose arbitrary plaintexts to be encrypted and obtain corresponding ciphertexts
  - goal is to gain further information which reduces security of encryption scheme
- Modern cryptography is implemented in software or hardware; for many cases, a CPA is feasible
  - CPAs become extremely important in the context of public key cryptography, where the encryption key is public and attackers can encrypt any plaintext they choose.
- Two forms of CPA
  - Batch CPA, where all plaintexts are chosen before any of them are encrypted
  - Adaptive CPA, where subsequent plaintexts are based on information from the previous encryptions.

RSA - Attacks

Implementation attacks

- Timing: based on time required to compute C^d
- Energy: based on energy required to compute (smart card) C^d

- Solut.: add random steps in implementation

RSA - ATTACKS : CONCLUSION

Textbook implementation of RSA is NOT safe

- It does not verify security criteria
- Many attacks

There exists a standard version

- Preprocess M to obtain M’ and apply RSA to M’ (clearly the meaning of M e M’ is the same)

$$M \longrightarrow M' \xrightarrow{\text{RSA}} C$$

Properties of RSA

- The requirement $(e,\varphi(n))=1$ is important for uniqueness
- Finding d, given p and q is easy. Finding d given only n and e is assumed to be hard (the RSA assumption)
- The public exponent e may be small. Typically its value is either 3 (problematic) or $2^{16}+1$
- Each encryption involves several modular multiplications. Decryption is longer.

Public-Key Cryptography Standard (PKCS)

Set of standard devised and published by RSA Security; many versions with different goals (1-15). See Wikipedia for details

PKCS#1: standard to send messages using RSA (byte)

$m = 0 \parallel 2 \parallel$ at least 8 non-zero bytes $\parallel 0 \parallel M$

(M original message)

- first byte 0 implies message is less than N
- second byte (= 2) denotes encrypting of a message (1 denotes dig. signature) and implies m is not small
- random bytes imply
  - same message sent to many people is always different
  - space of message is large

RSA and Data integrity: OAEP

- Padding scheme often used together with RSA encryption
  - introduced by Bellare and Rogaway: "Optimal Asymmetric Encryption - How to Encrypt with RSA" 1994, 1995 (http://cseweb.ucsd.edu/users/mihir/papers/oae.pdf)
- OAEP uses a pair of random oracles G and H to process the plaintext prior to asymmetric encryption
  - a random oracle is a mathematical function mapping every possible query to a random response from its output domain.
  - when combined with RSA it is secure under chosen plaintext/ciphertext attacks
- OAEP satisfies the following two goals
  1. Add an element of randomness which can be used to convert a deterministic encryption scheme (e.g., traditional RSA) into a probabilistic scheme.
  2. Prevent partial decryption of ciphertexts (or other information leakage) by ensuring that an adversary cannot recover any portion of the plaintext without being able to invert the trapdoor one-way permutation f.

Optimal Asymmetric Encryption Padding (OAEP)

n = number of bits in RSA modulus

k0 and k1 = integers fixed by the protocol

m = plaintext message, a (n – k0 – k1)-bit string

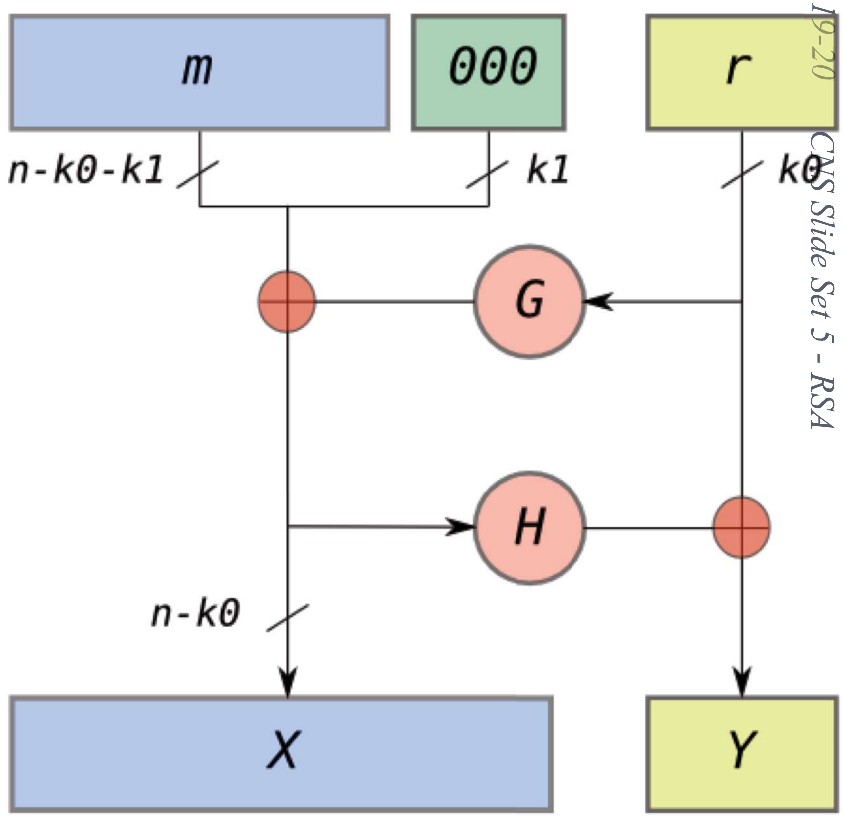
G and H = cryptographic hash functions fixed by the protocol

To encode

- messages are padded with k1 zeros to be n – k0 bits in length.
- r is a random k0-bit string
- G expands the k0 bits of r to n – k0 bits.
- $X = m00..0 \oplus G(r)$
- H reduces the n – k0 bits of X to k0 bits.
- $Y = r \oplus H(X)$
- output is $X \| Y$

To decode

- recover the random string as $r = Y \oplus H(X)$
- recover the message as $m00..0 = X \oplus G(r)$



OAEP - "ALL-OR-NOTHING" SECURITY

"all-or-nothing" security

- to recover m, you must recover the entire X and the entire Y
  - X is required to recover r from Y, and r is required to recover m from X
- since any bit of a cryptographic hash completely changes the result, the entire X, and the entire Y must both be completely recovered

BASIC SCHEME

- A public key encryption scheme includes the following elements:
  - A private key k
  - A public key k'
  - An encryption algorithm, which is a trap door OWF. The trap-door info is the private key
- Public key is published
- Encryption uses the public key (anyone can encrypt)
- Decryption requires the private key

ElGamal Encryption

- Constructed by ElGamal in 1984
- Based on DH and consists of three components: key generator, encryption algorithm, decryption algorithm
- Alice publishes p, $g \in Z_p$ as public parameters
  - g is generator of a cyclic group of order p
- Alice chooses x as a private key and publishes g^x mod p as a public key
  - x chosen at random in {0, 1, …, p-1}
- Encryption (Bob, for Alice) of $m \in Z_p$ by sending (g^y mod p, mg^{xy} mod p)
  - y chosen at random in {0, 1, …, p-1}
- Decryption: Alice computes $(g^y)^x$ mod p = g^{xy} mod p, then computes $(g^{xy})^{-1}$ mod p for obtaining $mg^{xy}(g^{xy})^{-1}$ mod p = m
- Requires two exponentiations per each block transmitted
- The involved math is not trivial

Real World usage

Two words:

Key Exchange

(use RSA (ElGamal) to define a secret key that is used with a faster protocol like AES)