RSA

Public Key Cryptosystem

Introduction

- After Whitfield Diffie and Martin Hellman introduced public-key cryptography (Diffie Hellman Algorithm) in 1976, a new branch of cryptography suddenly opened up.

- In 1977, Ronald Rivest, Adi Shamir and Leonard Adleman proposed a scheme at MIT known as RSA.

- General Information
  - best known & widely used asymmetric cryptographic scheme
  - can be used to provide both secrecy & digital signatures
  - based on exponentiation in a finite field over integers
  - modulo a prime, using large integers (e.g.,2048 bits)

RSA Key Setup

- Each user generates a public/private key pair by the following process:
  1. Select two large distinct primes at random – p and q
  2. Compute modulus $n = p \times q$
  3. Compute $ø(n) = (p-1) \times (q-1)$
  4. Choose an integer e such that 1<e<ø(n), gcd(e,ø(n))=1
     - (e and $\phi(n)$ share no factors other than 1) (e is relatively prime with $\phi(n)$ or co-primes)
     - Consider e as a public key
  5. Calculate decryption key d
     - $e.d = 1\, mod\, ø(n)$ and $0 \leq d \leq n$.
     - $d = e^{-1} mod\ \phi(n)$

RSA Key Setup

- Publish the Public encryption key: $PU = \{e, n\}$
- Keep secret Private Decryption key: $PR = \{d, n\}$
- It is critically important that the factors $p \& q$ of the modulus n are kept secret
- The primes p and q must be of sufficient size that factorization of their product is beyond computational reach

RSA Encryption

- For encryption, sender performs following:
  - Obtains the recipients public key $PU: (e, n)$
  - Represents the plaintext (message to be encrypted) as a positive integer value P
    - Size of plaintext must be less than n, $(0 \leq P \leq n)$
    - If the size of the plaintext is larger than n, it should be divided into blocks
  - Compute the Ciphertext : $C = P^e mod\ n$
  - Send the Ciphertext C to the recipient

RSA Decryption

- Receiver do the following:

- Uses private key PR: (d, n) to decrypt the cipher-text
  - $P = C^d mod\ n$

- Extract the plaintext from the integer representative P

RSA Example - Key Setup

1. Select primes → $p = 17$ and $q = 11$
2. Compute n → $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n)$
   - $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select encryption key e
   - $\gcd(e, \phi(n)) = 1$; choose $e = 7$
5. Decryption key d
   - $d = e^{-1} mod\ \phi(n)$ → $\mathrm{d} = 7^{-1} \mathrm{mod}\ 160 = 23$ (using Extended Euclidean Algorithm)
6. Publish Public Key $PU = \{e, n\}$ → $PU = \{7{,}187\}$
7. Keep secret Private Key $PR = \{d, n\}$ → $\{23{,}187\}$

RSA Example - En/Decryption

- The sample RSA private/public operations are:
  - Given message **P = 88 (note that 88<187)**

  - Encryption is:
    - **$C = 88^7 \bmod 187$**

      $= 88^{(3+3+1)} \bmod 187$

      $=(88^3 \bmod 187)(88^3 \bmod 187)(88 \bmod 187) \bmod 187$

      $= (44 * 44 * 88) \bmod 187$

      $= 11$

  - Decryption is:
    - **$P = 11^{23} \bmod 187 = 88$**

Greatest Common Divisor (GCD)

- a common problem in number theory
- GCD (a,b) of a and b is
  - the largest number that divides evenly into both a and b
  - e.g., GCD(60,24) = 12
- often want no common factors (except 1)
  - e.g., GCD(8,15) = 1
- these numbers are then relatively prime
  - hence 8 & 15 are relatively prime

Euclidean Algorithm

- an efficient way to find the GCD(a,b)
- uses theorem that:
  - GCD(a,b) = GCD(b, a mod b)
- Euclidean Algorithm to compute GCD(a,b) is:

- EUCLID(a,b)
  - 1. A = a; B = b
  - 2. if B = 0 return else A = gcd(a, b)
  - 3. R = A mod B
  - 4. A = B
  - 5. B = R
  - 6. goto 2

Example GCD(1970,1066)

1970 = 1 x 1066 + 904	gcd(1066, 904)
1066 = 1 x 904 + 162	gcd(904, 162)
904 = 5 x 162 + 94	gcd(162, 94)
162 = 1 x 94 + 68	gcd(94, 68)
94 = 1 x 68 + 26	gcd(68, 26)
68 = 2 x 26 + 16	gcd(26, 16)
26 = 1 x 16 + 10	gcd(16, 10)
16 = 1 x 10 + 6	gcd(10, 6)
10 = 1 x 6 + 4	gcd(6, 4)
6 = 1 x 4 + 2	gcd(4, 2)
4 = 2 x 2 + 0	gcd(**2**, 0)

Hence, gcd(1970,1066) = 2

Finding Inverses – Extended Euclidean algorithm

```
EXTENDED_EUCLID(m, b) [Find b^-1 mod m]
    1. (A1, A2, A3)=(1, 0, m);
       (B1, B2, B3)=(0, 1, b)
    2. if B3 = 0
       return A3 = gcd(m, b); no inverse
    3. if B3 = 1
       return B3 = gcd(m, b); B2 = b^-1 mod m
    4. Q = A3 div B3
    5. (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)
    6. (A1, A2, A3)=(B1, B2, B3)
    7. (B1, B2, B3)=(T1, T2, T3)
    8. goto2
```

Inverse of 37 in modulus 160 ($7^{-1} mod\ 160$)

- calling Extended_Euclid(160, 7)

Q	A1	A2	A3	B1	B2	B3
—	1	0	160	0	1	7
22	0	1	7	1	-22	6
1	1	-22	6	-1	23	1

- Hence, $7^{-1} mod\ 160 = 23$

Example: Inverse of 37 in modulus 49 ($37^{-1} mod\ 49$)

- calling Extended_Euclid(49, 37)

Q	A1	A2	A3	B1	B2	B3
—	1	0	49	0	1	37
1	0	1	37	1	-1	12
3	0	1	12	-3	4	1

- Hence 37^{-1} mod 49 = 4 OR $37^{-1} \equiv 4$ mod 49

Inverse of 550 in modulus 1759

- calling Extended_Euclid(1759, 550)

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

- Hence, $550^{-1} mod\ 1759 = 355$

Inverse of 49 in modulus 37

- calling Extended_Euclid(37, 49)

Q	A1	A2	A3	B1	B2	B3
—	1	0	37	0	1	49
0	0	1	49	1	0	37
1	1	0	37	-1	1	12
3	-1	1	12	4	-3	1

- Hence $49^{-1} \bmod 37 = -3$
- But, $-3 \pmod{37} \equiv 34 \pmod{37}$. Hence,
- $34 = 49^{-1} \bmod 37$

Text Example: RSA Key Setup

- Encrypt message “hello” with block size 1
- Given e=3,p=137 and q=131, n = 17947

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- Letters H = 7, E = 4, L = 11,.....
- Encryption of chracter “H”: $C = P^e mod\ n \ \rightarrow 7^3 mod\ 17947 = 343$
- Decryption key, $\phi(n) = 17680 \rightarrow d = 3^{-1} mod\ 17680 = 11787$

Inverse of 3 in modulus 17680

- calling Extended_Euclid(17680, 3)

Q	A1	A2	A3	B1	B2	B3
—	1	0	17680	0	1	3
5893	0	1	3	1	-5893	1

- Hence, $3^{-1} mod\ 17680 = -5893\ \rightarrow -5893\ mod\ 17680 = 11787$

Decryption

- Decryption : $P = C^d mod\ n \ \rightarrow 343^{11787} mod\ 17947$
- $343^{11787} mod\ 17947$
  - 11787= 8192+2048+1024+512+8+2+1
- 343^2 = 9967, 343^8 = 7436, 343^64 = 9880, 343^512=12090, 343^1024 = 7732,
- 343^2048 = 2367, 343^8192 = 9312
- = (9312*2367*7732*12090*7436*9967*343) mod 17947
- = 7 = “H”

Encryption-Decryption

- H = 7, **E = 4**, L = 11
- •e=3,p=137 and q=131, n = 17947, d = 11787
- •Let's encrypt -$\boldsymbol{C = P^e \bmod n}$
- • $4^\wedge 3 \; mod \; 17947 \; = \; 64$
- •Decrypt = $\boldsymbol{P \; = \; C^d \bmod n}$
- •64 ^11787 mod 17947 = ?

Conti…

- H = 7, E = 4, **L = 11, L = 11, O = 14**
- e=3,p=137 and q=131, n = 17947, d = 11787

- Encryption: $L = \ 11^3 \, mod \, 17947 \ = ?$
- $O = \ 14^3 \, mod \, 17947 \ = ?$

Another Text Example – Block Size 3

- given e=3,p=137 and q=131, encrypt the message "HELLO"
- using 00 to25 for letters A to Z for the block size of THREE.
- First block of size three-
  - $HEL = H * 26^2 + E * 26^1 + L * 26^0 = 7 * 26^2 + 4 * 26^1 + 11 * 26^0 =?$ (4847)

Second block

- $LOX = L * 26^2 + O * 26^1 + X * 26^0 = 11 * 26^2 + 14 * 26^1 + 23 * 26^0 =?$ (7823)

Encryption $\boldsymbol{C = P^e \, mod \, n}$

- $C1 = (4847)^3 mod \, 17947 = 4978$
- $C2 = (7823)^3 mod \, 17947 = 12882$

Decryption

- $\boldsymbol{P = C^d mod\ n}$
- (4978) ^ 11787 mod 17947 =

//11787 = 8192+2048+1024+512+8+2+1

(4978)^2 mod 17947 = 13624	(4978)^4 mod 17947 = 5502	(4978)^8 mod 17947= 13362
(4978)^16 mod 17947 = 6288	(4978)^64 mod 17947 = 10742	(4978)^512 mod 17947 = 16375
(4978)^1024 mod 17947 = 12445	(4978)^2048 mod 17947 = 13362	(4978)^8192 mod 17947 = 1703

- (4978)^(8192+2048+1024+512+8+2+1) mod 17947
- = 4847 (How to get HEL out of 4847?)

Recover Plaintext

- 4847%26 == 11 (L) Quotient=186
- 186%26 == 4 (E) Quotient=7
- 7%26 == 7 (H) !!!!!

RSA Key Generation

- The users of RSA must
  - determine two primes at random - **p, q**
  - select either **e or d and compute the other**

- primes **p,q must not be easily derived from modulus n=p.q**
  - means must be sufficiently large
  - typically guess and use probabilistic test

- exponents **e, d are inverses, so use inverse algorithm to** compute the other
  - So, the basic operation involved, in either case, is
    - **modular exponentiation**
  - Use binary left-to-right or square-and-multiply method

Computational complexity: Exponentiation

- The Square and Multiply Algorithm
  - a fast, efficient algorithm for exponentiation
  - also known as
    - exponentiation by squaring OR
    - Binary exponentiation OR
- concept is based on repeatedly squaring the base
  - and multiplying in the ones that are needed to compute the result
- can be formally defined as
  - Power (x,n) $= \begin{cases} x & \text{if } n=1 \\ \text{Power}(x^2, n/2) & \text{if } n = \text{even} \\ x \cdot \text{Power}(x^2, (n-1)/2) & \text{if } n = \text{odd} \end{cases}$

Computational complexity: Exponentiation

- The Square and Multiply Algorithm
- Algorithm for Computing: a^b mod n
- The integer b is expressed as a binary number $b_k b_k 1 \ldots b_0$

Algorithm Square_Multiply(,)

- c = 0; f = 1
- for i = k down to 0
  - do c = 2 * c
    - f = (f * f) mod n
  - if bi == 1 then
    - c = c + 1
    - f = (f x a) mod n
- return f

Result of Square and Multiply Algorithm

- **where a = 7, b = 560 = 1000110000, n = 561**
- bi = **1000110000**

I	9	8	7	6	5	4	3	2	1	0
bi	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

- **Result** a^b mod n = 7^{560} mod 561 = 1 (final return value of f)

Result of Square and Multiply Algorithm

- **where a = 4978, b = 11787 = 10111000001011, n = 17947**
- **bi= 10111000001011**

- **Result** a^b mod n = 4978^{11787} mod 17947

Result of Square and Multiply Algorithm

- **where a = 4978, b = 11787 = 10111000001011, n = 17947**
- **bi= 10111000001011**

I	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bi	1	0	1	1	1	0	0	0	0	0	1	0	1	1
c	1	2	5	11	23	46	92	184	368	736	1473	2946	5893	11787
f	4978	13624	1834	2489	13100	786	7598	12052	5633	393	15589	14541	2358	4847

- **Result** a^b mod n = 4978^{11787} mod 17947 = 4847 (final return value of f)

Text Example: RSA Key Setup

- Consider the text grouping in the groups of three i.e.
  - ATTACKXATXSEVEN = ATT ACK XAT XSE VEN
- Represent the blocks in base 26 using A=0, B=1, C=2…..
  - ATT = $0 * 26^2 + 19 * 26^1 + 19 = 513$
  - ACK = $0 * 26^2 + 2 * 26^1 + 10 = 62$
  - XAT= $23 * 26^2 + 0 * 26^1 + 19 = 15567$
  - XSE= $23 * 26^2 + 18 * 26^1 + 4 = 16020$
  - VEN = $21 * 26^2 + 4 * 26^1 + 13 = 14313$
- Next issue is designing the cryptosystem – selecting the parameters.
- What should be the value of n ?

Text Example: RSA Key Setup

- The value of n should be greater than 17575. How & why ?
  - **0≤m<n (Here maximum value of message can ZZZ =** $25 * 26^2 + 25 * 26^1 + 25 = 17575$
- Let p = 137 and q = 131; so that n = pq = 17947

Text Example: RSA Key Setup

- Compute phi value
  - ***ø(n)=(p–1)(q-1)=136 x 130=17680***
- Select encryption parameter
  - **e: gcd(e,17680)=1; choose *e=3***
- Determine decryption parameter
  - **d: *de=1 mod 17680 and d < 17680***

- Value is **d=11787 since 3^{-1} mod 17680 = 11787**
- Publish public key **PU={3,17947}**
- Keep secret private key **PR={11787,17947}**

Text En/Decryption using RSA

- The sample RSA private/public operations are:
  - Given message **M = ATT = 513**

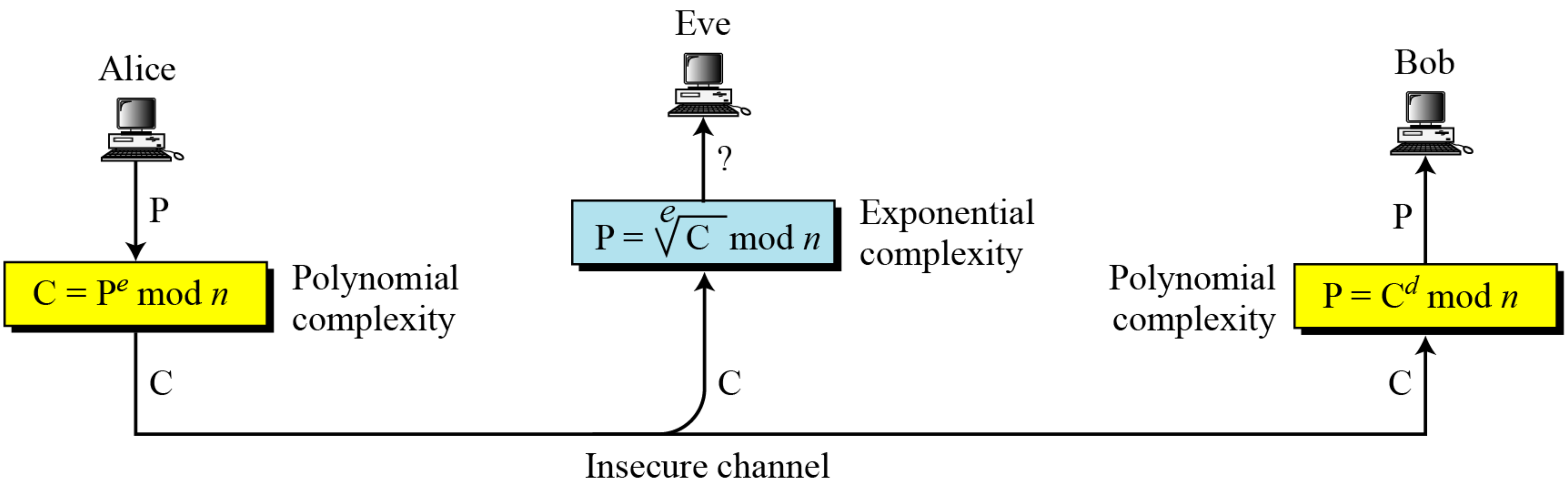
  - Overall the plaintext is represented as the set of integers m
    - {513,62,15567,16020,14313}
  - Encryption is
    - **$C = 513^3 \bmod 17947$ ($m^e \bmod n$)**

      = 8363
  - Overall the Ciphertext is represented as the set of integers m
    - {8363,5017,11884,9546,13366}
  - Decryption is

    $M = 8363^{11787} \bmod 17947$ ($c^d \bmod n$)

    =513

Complexity of operations in RSA



RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.

RSA Challenge

number	digits	prize	factored
RSA-100	100		Apr. 1991
RSA-110	110		Apr. 1992
RSA-120	120		Jun. 1993
RSA-129	129	$100	Apr. 1994
RSA-130	130		Apr. 10, 1996
RSA-140	140		Feb. 2, 1999
RSA-150	150		Apr. 16, 2004
RSA-155	155		Aug. 22, 1999
RSA-160	160		Apr. 1, 2003
RSA-200	200		May 9, 2005
RSA-576	174	$10,000	Dec. 3, 2003
RSA-640	193	$20,000	Nov. 4, 2005
RSA-704	212	$30,000	open
RSA-768	232	$50,000	open
RSA-896	270	$75,000	open
RSA-1024	309	$100,000	open
RSA-1536	463	$150,000	open
RSA-2048	617	$200,000	open