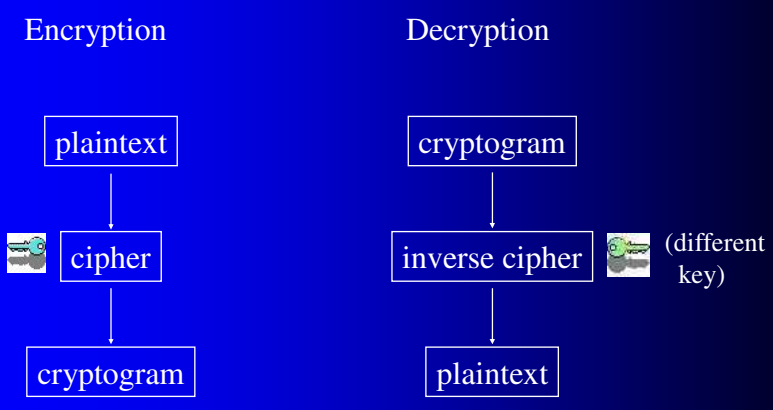
The RSA algorithm

a foundation of public-key substitution ciphers

David Morgan

Public-key crypto



Different algorithms work

- RSA — Rivest, Shamir, Adelman; MIT
- ElGamal — Taher ElGamal, Netscape
- DSA — NSA, NIST

RSA key generation steps

1. choose 2 primes — call them p, q
2. multiply them — call product n
3. multiply their "predecessors" (p-1,q-1) — call product φ
4. pick some integer — call it e
   - between 1 and φ (exclusive)
   - sharing no prime factor with φ
5. find the integer (there's only one) that — call it d
   - times e divided by φ leaves 1

then your keys are:

- public: e together with n (e is for "encryption")
- private: d together with n (d is for "decryption")

Encrypting with public key {e,n}

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

1. choose a cleartext message — call it m
   - in the form of a number less than n
2. raise it to power e
3. divide that by n — call remainder c

then your ciphertext result is c

Decrypting with private key {d,n}

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

1. take ciphertext c
2. raise it to power d
3. divide that by n — call remainder r

then your recovered result is r

- r is identically the original cleartext message m

How will we do keygen step 4?

1. choose 2 primes — easy
2. multiply them — easy
3. multiply their "predecessors" (p-1,q-1) — easy
4. pick some integer — not easy
   - between 1 and φ (exclusive)
   - sharing no prime factor with φ
5. find the integer (there's only one) that — not easy
   - times e divided by φ leaves 1

then your keys are:

- public: e together with n (e is for "encryption")
- private: d together with n (d is for "decryption")

Numbers *sans* common prime factor

- numbers whose gcd* is 1 will do
- find x such that gcd(x, φ)=1
- how do we find gcd of 2 numbers
  - Euclid's algorithm

*greatest common divisor

Euclid's algorithm

- given a and b
- divide: a = qb + r (r a.k.a. "a mod b")
- ***greatest*** common divisor of a-and-b must be ***a*** common divisor of b-and-r*

*because it divides r (since r = a – qb and it divides both terms). By definition it divides b. Therefore it divides both.

Euclid's algorithm

- any common divisor of b-and-r must also divide a*
- and therefore a-and-b
- the sets of common divisors of a-and-b, and of b-and-r, are the same set
- the greatest of them is the shared gcd of both a-and-b and b-and-r

→ gcd(a,b)=gcd(b,r) ←

*Why? a = qb + r , and it divides both terms

Euclid's algorithm

- we now know gcd(a,b) and gcd(b,r) are the same
- but we don't know what "they are"/"it is" (gcd's actual value)
- to find out, apply the equation repeatedly

Euclid's algorithm example

What is gcd(197235, 1050) ?		197235 = 187 x 1050 + 885
it's →	gcd(1050, 885)	1050 = 1 x 885 + 165
→	gcd(885, 165)	885 = 5 x 165 + 60
→	gcd(165, 60)	165 = 2 x 60 + 45
→	gcd(60, 45)	60 = 1 x 45 + 15
→	gcd(45, 15)	45 = 3 x 15 + 0

gcd(197235, 1050) = gcd(45, 15) = 15

How will we do keygen step 5?

1. choose 2 primes — easy
2. multiply them — easy
3. multiply their "predecessors" (p-1,q-1) — easy
4. pick some integer — not easy
   - between 1 and φ (exclusive)
   - sharing no prime factor with φ
5. find the integer (there's only one) that — not easy
   - times e divided by φ leaves 1

then your keys are:

- public: e together with n (e is for "encryption")
- private: d together with n (d is for "decryption")

Successively test candidates

- multiply each by e
- divide by φ
- check if remainder is 1
- keep going till you find the one that is

How will we do en/de-crypting's $x^y \bmod n$

Use a property of modular arithmetic:

$(a \times b) \bmod n = [\,(a \bmod n) \times (b \bmod n)\,] \bmod n$

modulo of a product = modulo of the product of its multipliers' modulos

Strategy - reduce x^y into $x^a x^b x^c$ where $y=abc$ using powers of 2 for a, b, c (because they are easy). Then apply above property

Example: calculate $13^{27} \bmod 55$

Preliminary – find out $13^t \bmod 55$ for the various t that are powers of 2

$(a \times b) \bmod n = [\,(a \bmod n) \times (b \bmod n)\,] \bmod n$

$13^1 \bmod 55$ = **13**

$13^2 \bmod 55$ = $[(13 \bmod 55) \times (13 \bmod 55)] \bmod 55$
= $[\,13 \times 13\,] \bmod 55 = 169 \bmod 55 =$ **4**

$13^4 \bmod 55$ = $[(13^2 \bmod 55) \times (13^2 \bmod 55)] \bmod 55$
= $[\,4 \times 4\,] \bmod 55 = 16 \bmod 55 =$ **16**

$13^8 \bmod 55$ = $[(13^4 \bmod 55) \times (13^4 \bmod 55)] \bmod 55$
= $[\,16 \times 16\,] \bmod 55 = 256 \bmod 55 =$ **36**

$13^{16} \bmod 55$ = $[(13^8 \bmod 55) \times (13^8 \bmod 55)] \bmod 55$
= $[\,36 \times 36\,] \bmod 55 = 1296 \bmod 55 =$ **31**

$13^{32} \bmod 55$ = $[(13^{16} \bmod 55) \times (13^{16} \bmod 55)] \bmod 55$
= $[\,31 \times 31\,] \bmod 55 = 961 \bmod 55 =$ **26**

Keep these in your pocket

Example: calculate 13^{27} mod 55

13^t mod 55 for the various t that are powers of 2:

13^1 mod 55 = 13
13^2 mod 55 = 4
13^4 mod 55 = 16
13^8 mod 55 = 36
13^{16} mod 55 = 31
13^{32} mod 55 = 26

Keep these in your pocket

Example: calculate 13^{27} mod 55

Reduce x^y into $x^a x^b x^c \ldots$ where y=a+b+c+… using powers of 2 for a, b, c

$13^{27} = 13^{16}13^8 13^2 13^1$ where 27=16+8+2+1
…or for starters instead
$13^{27} = 13^{24}13^3$
so we can apply (a x b) mod n = [(a mod n) x (b mod n)] mod n

13^{27} mod 55 = $13^{24}13^3$ mod 55 = [(13^{24} mod 55) x (13^3 mod 55)] mod 55

Apply it again (twice)

$13^{16}13^8$ mod 55 = [(13^{16} mod 55) x (13^8 mod 55)] mod 55

$13^2 13^1$ mod 55 = [(13^2 mod 55) x (13^1 mod 55)] mod 55

Example: calculate 13^{27} mod 55

$13^{27} \bmod 55 = 13^{24}13^{3} \bmod 55 = [(13^{24} \bmod 55) \times (13^{3} \bmod 55)] \bmod 55$

Apply it again (twice)

$13^{16}13^{8} \bmod 55 = [(13^{16} \bmod 55) \times (13^{8} \bmod 55)] \bmod 55$

$13^{2}13^{1} \bmod 55 = [(13^{2} \bmod 55) \times (13^{1} \bmod 55)] \bmod 55$

$13^{1} \bmod 55 = 13$
$13^{2} \bmod 55 = 4$
$13^{4} \bmod 55 = 16$
$13^{8} \bmod 55 = 36$
$13^{16} \bmod 55 = 31$
$13^{32} \bmod 55 = 26$

take these out of your pocket and plug them in

Example: calculate 13^{27} mod 55

$13^{27} \bmod 55 = 13^{24}13^{3} \bmod 55 = [(13^{24} \bmod 55) \times (13^{3} \bmod 55)] \bmod 55$

Apply it again (twice)

$13^{16}13^{8} \bmod 55 = [(31) \times (36)] \bmod 55$

$13^{2}13^{1} \bmod 55 = [(4) \times (13)] \bmod 55$

$13^{1} \bmod 55 = 13$
$13^{2} \bmod 55 = 4$
$13^{4} \bmod 55 = 16$
$13^{8} \bmod 55 = 36$
$13^{16} \bmod 55 = 31$
$13^{32} \bmod 55 = 26$

plug these in

Example: calculate 13^{27} mod 55

13^{27} mod 55 = $13^{24}13^{3}$ mod 55 = [(13^{24} mod 55) x (13^{3} mod 55)] mod 55

Apply it again (twice)

now plug these in

$13^{16}13^{8}$ mod 55 = [(31) x (36)] mod 55 = 1116 mod 55 = **16**

$13^{2}13^{1}$ mod 55 = [(4) x (13)] mod 55 = 52 mod 55 = **52**

Example: calculate 13^{27} mod 55

13^{27} mod 55 = $13^{24}13^{3}$ mod 55 = [(16) x (52)] mod 55

Apply it again (twice)

$13^{16}13^{8}$ mod 55 = [(31) x (36)] mod 55 = 1116 mod 55 = **16**

$13^{2}13^{1}$ mod 55 = [(4) x (13)] mod 55 = 52 mod 55 = **52**

Example: calculate 13^{27} mod 55

13^{27} mod 55 = $13^{24}13^{3}$ mod 55 = [(16) x (52)] mod 55 = 832 mod 55 = 7

$$13^{27} \bmod 55 = 7$$

13^{27} is 1192533292512492016559195008117

we solved the problem by reducing it from

1192533292512492016559195008117 mod 55 to 832 mod 55

RSA key generation example

1. choose 2 primes — p=5 q=11
2. multiply them — n=55
3. multiply their "predecessors" (p-1,q-1) — φ=40
4. pick some integer — e=3
   - between 1 and φ (exclusive)
   - sharing no prime factor with φ
5. find the integer (there's only one) that — d=27
   - times e divided by φ leaves 1

then your keys are:

- public: e together with n — 3, 55
- private: d together with n — 27, 55

Encrypting with public key {e,n}

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

1. choose a cleartext message — m=7
   - in the form of a number less than n
2. raise it to power e — $7^3=343$
3. divide that by n — $343 = 55x6+13$

then your ciphertext result is c — c=13

Decrypting with private key {d,n}

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

1. take ciphertext c — 13
2. raise it to power d

$$13^{27}=11925332925124920165591950081 17$$

3. divide that by n

$$11925332925124920165591950081 17 = 55 \times 249764639940835239319763167 + 7$$

then your recovered result is r — r=7

- r is identically the original cleartext message m

Blocking data - by ascii

- RED APPLE = 8269683265808076 69
- use ascii's 2 digits as block basis
- separately encrypt:
  82 69 68 32 65 80 80 76 69
- be prepared for maximum ~ 99
- minimum φ 100, eg p=11 q=13

ASCII Alphabet Characters

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Blocking data

- RED APPLE = 8269683265808076 69
- use 3-decimal-digit blocks
- separately encrypt:
  826 968 326 580 807 669
- be prepared for maximum ~ 999
- minimum φ 1000, eg p=31 q=37

ASCII Alphabet Characters

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Blocking data

- ABC = 01000001 01000010 01000011
- use 12-bit blocksize
- separately encrypt:
  010000010100 001001000011
- be prepared for maximum – 4096
- minimum φ 4097, eg p=67 q=71

ASCII Alphabet Characters

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Blocking data

- HELLO =

  01001000 01000101 01001100 01001100 01001111
- use 20-bit blocksize
- separately encrypt:
  01001000010001010100 11000100110001001111
- be prepared for maximum – 1048576 (1M)
- minimum φ 1048577, eg p=1021 q=1031

ASCII Alphabet Characters

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Some considerations

- RSA "key size" – refers to n
- p and q should be about equal length
- but not extremely close (eg avoid successive primes)
- larger key, slower operation
  - double n → pubkey ops 2x slower, privkey 4x
  - e can stay fixed while n rises, but d up proportionately
- practical keylengths, 1024 or 2048 bits
- RSA and DES per-keylength security comparisons apples and oranges

http://www.rsa.com/rsalabs/node.asp?id=2218