RSA and Public Key Cryptography

Chester Rebeiro

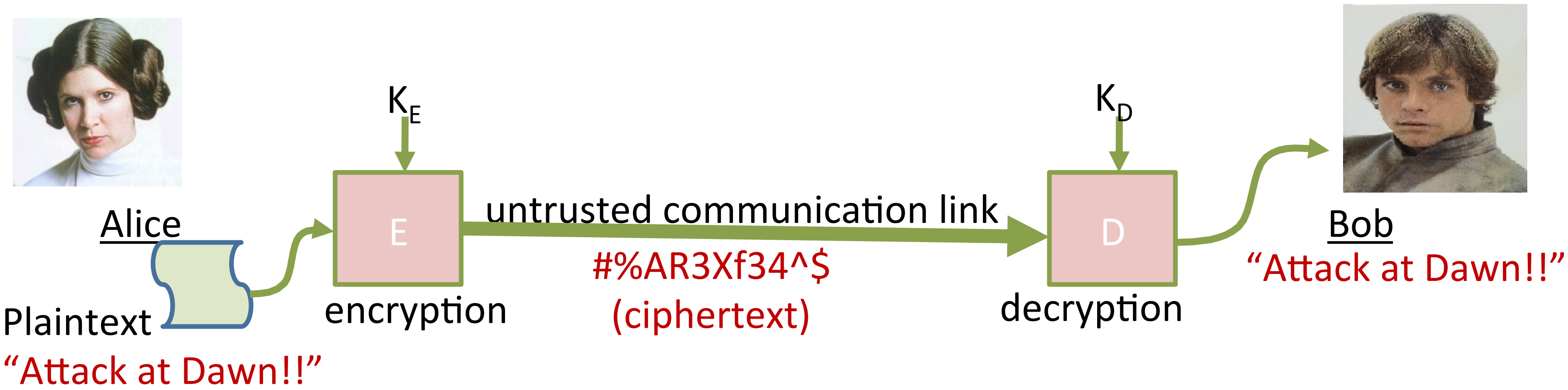
IIT Madras

STINSON : chapter 5, 6

Ciphers

- Symmetric Algorithms
  - Encryption and Decryption use the same key
  - i.e. $K_E = K_D$
  - Examples:
    - Block Ciphers : DES, AES, PRESENT, etc.
    - Stream Ciphers : A5, Grain, etc.
- Asymmetric Algorithms
  - Encryption and Decryption keys are different
  - $K_E \neq K_D$
  - Examples:
    - RSA
    - ECC

Asymmetric Key Algorithms



The Key K is a secret

Encryption Key K_E not same as decryption key K_D

K_E known as Bob's public key;
K_D is Bob's private key

Advantage : No need of secure key exchange between Alice and Bob

Asymmetric key algorithms based on trapdoor one-way functions

One Way Functions

- Easy to compute in one direction
- Once done, it is difficult to inverse

Press to lock
(can be easily done)

Once locked it is
difficult to unlock
without a key

Trapdoor One Way Function

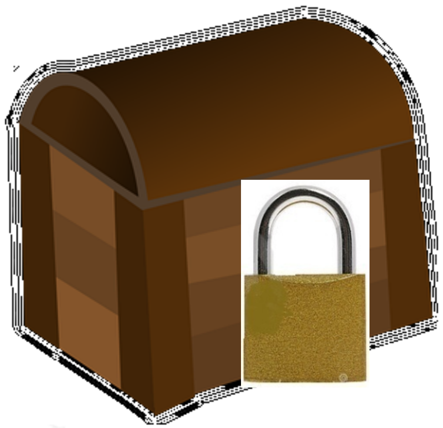
- One way function with a trapdoor
- Trapdoor is a special function that if possessed can be used to easily invert the one way

Locked
(difficult to unlock)

Easily Unlocked

Public Key Cryptography (An Anology)

- Alice puts message into box and locks it
- Only Bob, who has the key to the lock can open it and read the message



Mathematical Trapdoor One way functions

- Examples
  - Integer Factorization (in NP, maybe NP-complete)
    - Given P, Q are two primes
    - and N = P * Q
      - It is easy to compute N
      - However given N it is difficult to factorize into P and Q
    - Used in cryptosystems like RSA
  - Discrete Log Problem (in NP)
    - Consider b and g are elements in a finite group and $b^k = g$, for some k
    - Given b and k it is easy to compute g
    - Given b and g it is difficult to determine k
    - Used in cryptosystems like Diffie-Hellman
    - A variant used in ECC based crypto-systems

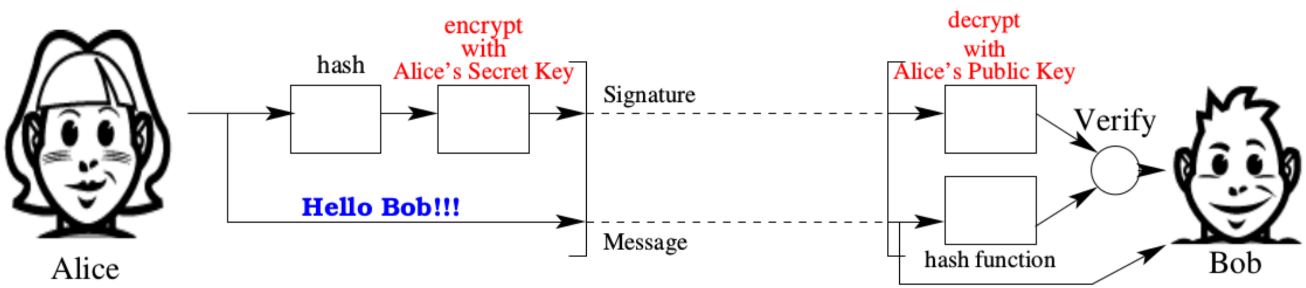
CR

Applications of Public key Cryptography

- **Encryption**
- **Digital Signature :**

  "Is this message really from Alice?"

  - Alice signs by 'encrypting' with private key
  - Anyone can verify signature by 'decrypting' with Alice's public key
  - Why it works?
    - Only Alice, who owns the private key could have signed



CR

Applications of Public key Cryptography

Diffie-Hellman Key Exchange

- **Key Establishment :**

  "Alice and Bob want to use a block cipher for encryption. How do they agree upon the secret key"

Alice and Bob agree upon a prime **p** and a generator **g**.
This is public information

choose a secret **a**
compute **A = g^a mod p**

choose a secret **b**
compute **B = g^b mod p**

B

A

Compute K **= B^a mod p**

Compute K **= A^b mod p**

$$A^b \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p = B^a \bmod p$$

RSA

Shamir, Rivest, Adleman (1977)

More Number Theory

Mathematical Background

RSA : Key Generation

Bob first creates a pair of keys (one public the other private)

1. Generate two large primes $p, q \ (p \neq q)$
2. Compute $n = p \times q$ and $\phi(n) = (p-1)(q-1)$
3. Choose a random $b \ (1 < b < \phi(n))$ and $\gcd(b, \phi(n)) = 1$
4. Compute $a = b^{-1} \bmod(\phi(n))$

$$Bob's\ public\ key\ is\ (n, b)$$

$$Bob's\ private\ key\ is\ (p, q, a)$$

Given the private key it is easy to compute the public key

Given the public key it is difficult to derive the private key

CR

RSA Encryption & Decryption

Encryption

Decryption

$$e_K(x) = y = x^b \bmod n$$

$$where\; x \in Z_n$$

$$d_K(x) = y^a \bmod n$$

RSA Example

1. Take two primes $p = 653$ and $q = 877$
2. $n = 653 \times 877 = 572681$; $\phi(\text{n}) = 652 \times 876 = 571152$
3. Choose public key $\text{b} = 13$; note that $\gcd(13, 571152) = 1$
4. Private key $a = 395413 = 13^{-1} \bmod 571152$

$Message\ x = 12345$

$encryption: y = 12345^{13} \bmod 572681 \equiv 536754$

$\text{decryption}: \text{x} = 536754^{395413} \bmod 572681 \equiv 12345$

Correctness

$$\boxed{when\ \ x \in Z_n\ \ and\ \gcd(x,n) = 1}$$

Encryption

$e_K(x) = y = x^b \bmod n$

$where\ x \in Z_n$

Decryption

$d_K(x) = y^a \bmod n$

$$\begin{aligned} y^a &\equiv (x^b)^a \bmod n \\ &\equiv (x^{ab}) \bmod n \\ &\equiv (x^{t\phi(n)+1}) \bmod n \\ &\equiv (x^{t\phi(n)}x) \bmod n \\ &\equiv x \end{aligned}$$

$$\begin{aligned} ab &\equiv 1 \bmod \varphi(n) \\ ab - 1 &= t\varphi(n) \\ ab &= t\varphi(n) + 1 \end{aligned}$$

From Fermat's theorem

Correctness

$$\boxed{when\ \ x \in Z_n \ \ and\ \gcd(x,n) \neq 1}$$

$$Since\ n = pq, \gcd(x,n) = p\ or\ \gcd(x,n) = q$$

$$If$$

$$x \equiv x^{ab} \bmod p$$

$$x \equiv x^{ab} \bmod q$$

$$\Rightarrow x \equiv x^{ab} \bmod n$$

$$(by\ CRT)$$

$$Assume\ \gcd(n,x) = p$$

$$\Rightarrow p \mid x \Rightarrow pk = x$$

$$LHS : x \bmod p \equiv pk \bmod p \equiv 0$$

$$RHS : x^{ab} \bmod p \equiv 0$$

$$\because \gcd(p,x) = p\ it\ implies\ \gcd(q,x) = 1$$

$$\begin{aligned} x^{ab} \bmod q &\equiv x^{t\phi(n)+1} \bmod q \\ &\equiv x^{t\phi(p)\phi(q)+1} \bmod q \\ &\equiv (x^{\phi(q)})^{t\varphi(p)} \cdot x \bmod q \\ &\equiv (1)^{t\varphi(p)} \cdot x \bmod q \equiv x \end{aligned}$$

CR

RSA Implementation

$$y = x^c \bmod n$$

Algorithm : SQUARE-AND-MULTIPLY(x, c, n)

$z \leftarrow 1$
for $i \leftarrow \ell - 1$ **downto** 0
  do $\begin{cases} z \leftarrow z^2 \bmod n \\ \textbf{if } c_i = 1 \\ \quad \textbf{then } z \leftarrow (z \times x) \bmod n \end{cases}$
return (z)

$c = 23 = (10111)_2$

i	e_i	z
4	1	$1^2 * x = x$
3	0	x^2
2	1	$x^4 * x = x^5$
1	1	$X^{10} * x = x^{11}$
0	1	$x^{22} * x = x^{23}$

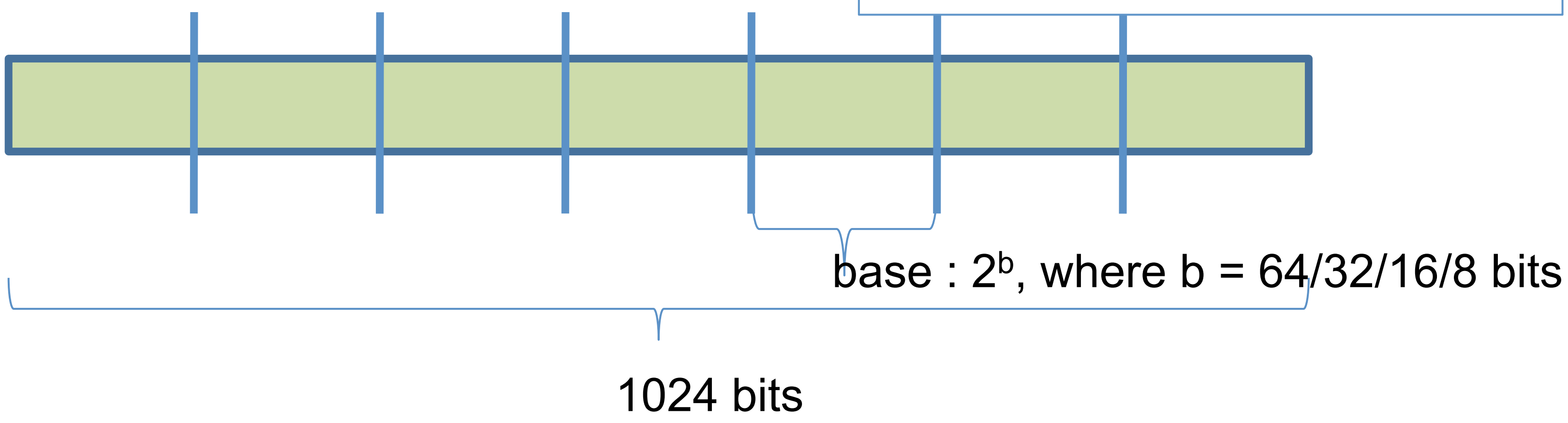
CR

RSA Implementation in Software (Multi-precision Arithmetic)

- RSA requires arithmetic in 1024 or 2048 bit numbers
- Modern processors have ALUs that are 8, 16, 32, 64 bit
  - Typically can perform arithmetic on 8/16/32/64 bit numbers
- solution: multi-precision arithmetic (gmp library)

```
#define NBITS     xxx
#define WORDSIZE  xxx
#define MAXDIGITS (NBITS/WORDSIZE)

typedef unsigned long word;
typedef struct{
  word digits[MAXDIGITS];
  int  sign;
}bignum_t;
```



Multi-precision Addition

- ADD : $a = 9876543210 = (2, 76, 176, 22, 234)_{256}$
  $b = 1357902468 = (80, 239, 242, 132)_{256}$

  base = 8 bit (256)

i	a_i	b_i	c_{in}	$a_i+b_i+c_{in} \pmod{256}$	Carry?	c_{out}
0	234	132	0	110	$(110 < 234)$?	1
1	22	242	1	9	$(9 < 22)$?	1
2	176	239	1	160	$(160 \le 176)$?	1
3	76	80	1	157	$(157 \le 76)$?	0
4	2	0	0	2	$(2 \le 2)$?	0

$a + b = (2, 157, 160, 9, 110)_{256}$
$= 11234445678$

CR

“Computational Number Theory”, Abhijit Das, CRC Press

Multi-Precision Addition Algorithm

Algorithm 2: Add : *Multi-Precision Addition.* The function performs $r = a + b$. Each input is of size n words.

Input: word $*a$, word $*b$, int n
Output: word $*r$

```
begin
    carry ← 0
    for i ∈ (0, 1, 2, ··· n − 1) do
        t ← a[i]
        t ← t + carry
        carry ← (t < carry)
        l ← t + b[i]
        carry ← carry + (l < t)
        r[i] ← l
    end
    return r
end
```

- The asymptotic complexity of multi-precision addition is $\mathcal{O}(\texttt{MAXDIGITS})$.
- The algorithm requires `MAXDIGITS` single precision additions to be performed, where each addition is of `WORDSIZE`.
- This also requires 2×`MAXDIGITS` comparisons as carry is compared with both the operands in each iteration of the loop.

Multi-precision Subtraction

- SUB : a = 9876543210 = (2, 76, 176, 22, 234)$_{256}$
  b = 1357902468 = (80, 239, 242, 132)$_{256}$

  base = 256 (8 bit)

i	a_i	b_i	Cin	Borrow?	C_{out}	a_i-b_i-c_{in}(mod 256)
0	234	132	0	(234 < 132)?	0	102
1	22	242	0	(22 < 242)?	1	-220 = 36
2	176	239	1	(176 < 239)?	1	-64=192
3	76	80	1	(76 < 80)?	1	-5=251
4	2	0	1	(2 < 0)?	0	1

a - b = (1, 251, 192, 36, 102)$_{256}$
= 8658640742

CR

Multi-Precision Subtraction Algorithm

Algorithm 3: Sub : *Multi-Precision Subtraction.* The function performs $r = a - b$. Each input is of size n words.

Input: word $*a$,word $*b$, int n
Output: word $*r$

begin
    $borrow \leftarrow 0$
    for $i \in (0, 1, 2, \cdots n-1)$ **do**
        $r[i] \leftarrow (a[i] - b[i] - borrow)$
        if $(a[i] \neq b[i])$ **then**
            $borrow = (a[i] < b[i])$
        end
    end
    return r
end

Analysis of Multi-Precision Subtraction

- The asymptotic complexity of multi-precision subtraction is $\mathcal{O}$(`MAXDIGITS`).
- The algorithm requires `MAXDIGITS` subtractions to be performed. Each subtraction is of `WORDSIZE`.
- This also requires `MAXDIGITS` comparisons as operands are compared to know the borrow in each iteration of the loop.

Multi-Precision Multiplication

C = A x B mod N

(without Modular operation)

- Classical (School book) algorithm
- Karatusba algorithm
- Toom-3 algorithm
- FFT

CR

Multi-precision Multiplication (Classical Multiplication)

- MUL : a = 1234567 = $(18, 214, 135)_{256}$
  b = 76543210 = $(4, 143, 244, 234)_{256}$
  base = 8 bit (256)

a * b =

$(0\ 85\ 241\ 247\ 25\ 195\ 102)_{256}$

= 99447721140070

i	a_i	j	b_j	$a_i b_j = (h, l)_B$	Operation	c
					Initialization	$(0, 0, 0, 0, 0, 0, 0)_B$
0	135	0	234	$(123, 102)_B$	Add 102 at pos 0	$(0, 0, 0, 0, 0, 0, 102)_B$
					Add 123 at pos 1	$(0, 0, 0, 0, 0, 123, 102)_B$
		1	244	$(128, 172)_B$	Add 172 at pos 1	$(0, 0, 0, 0, 1, 39, 102)_B$
					Add 128 at pos 2	$(0, 0, 0, 0, 129, 39, 102)_B$
		2	143	$(75, 105)_B$	Add 105 at pos 2	$(0, 0, 0, 0, 234, 39, 102)_B$
					Add 75 at pos 3	$(0, 0, 0, 75, 234, 39, 102)_B$
		3	4	$(2, 28)_B$	Add 28 at pos 3	$(0, 0, 0, 103, 234, 39, 102)_B$
					Add 2 at pos 4	$(0, 0, 2, 103, 234, 39, 102)_B$
1	214	0	234	$(195, 156)_B$	Add 156 at pos 1	$(0, 0, 2, 103, 234, 195, 102)_B$
					Add 195 at pos 2	$(0, 0, 2, 104, 173, 195, 102)_B$
		1	244	$(203, 248)_B$	Add 248 at pos 2	$(0, 0, 2, 105, 165, 195, 102)_B$
					Add 203 at pos 3	$(0, 0, 3, 52, 165, 195, 102)_B$
		2	143	$(119, 138)_B$	Add 138 at pos 3	$(0, 0, 3, 190, 165, 195, 102)_B$
					Add 119 at pos 4	$(0, 0, 122, 190, 165, 195, 102)_B$
		3	4	$(3, 88)_B$	Add 88 at pos 4	$(0, 0, 210, 190, 165, 195, 102)_B$
					Add 3 at pos 5	$(0, 3, 210, 190, 165, 195, 102)_B$
2	18	0	234	$(16, 116)_B$	Add 116 at pos 2	$(0, 3, 210, 191, 25, 195, 102)_B$
					Add 16 at pos 3	$(0, 3, 210, 207, 25, 195, 102)_B$
		1	244	$(17, 40)_B$	Add 40 at pos 3	$(0, 3, 210, 247, 25, 195, 102)_B$
					Add 17 at pos 4	$(0, 3, 227, 247, 25, 195, 102)_B$
		2	143	$(10, 14)_B$	Add 14 at pos 4	$(0, 3, 241, 247, 25, 195, 102)_B$
					Add 10 at pos 5	$(0, 13, 241, 247, 25, 195, 102)_B$
		3	4	$(0, 72)_B$	Add 72 at pos 5	$(0, 85, 241, 247, 25, 195, 102)_B$
					Add 0 at pos 6	$(0, 85, 241, 247, 25, 195, 102)_B$

Multi-precision Multiplication (Karatsuba Multiplication)

Let a, b be two multiprecision integers with n B – ary words.

$Let\, m = n/2$

$a = a_h B^m + a_l$

$b = b_h B^m + b_l$

$$a \times b = (a_h b_h) B^{2m} + (a_h b_l + a_l b_h) B^m + a_l b_l$$
$$= (a_h b_h) B^{2m} + \left(a_h b_h + a_l b_l - (a_h - a_l)(b_h - b_l)\right) B^m + a_l b_l$$

using $(a_h - a_l)(b_h - b_l) = a_h b_h - a_h b_l - a_l b_h + a_l b_l$

Karatsuba multiplication converts n bit multiplications into 3 multiplications of n/2 bits
The penalty is an increased number of additions

CR

Multi-precision Multiplication (Karatsuba Multiplication)

B = 256;
a = 123456789 = $(7, 91, 205, 21)_{256}$
b = 987654321 = $(58, 222, 104, 177)_{256}$

n=4; m=2
$a_h = (7, 91)$; $a_l = (205, 21)$
$a = (7, 91)256^2 + (205, 21)$

$b_h = (58, 222)$; $b_l = (104, 177)$
$b = (58, 222)256^2 + (104, 177)$

$a_h b_h = (1, 176, 254, 234)_{256}$
$a_l b_l = (83, 222, 83, 133)_{256}$
$a_h - b_h = -(197, 186)_{256}$
$a_l - b_l = -(45, 211)_{256}$
$(a_h - b_h)(a_l - b_l) = (35, 100, 170, 78)_{256}$
$a_h b_l + a_l b_h$
$= a_h b_h + a_l b_l - (a_h - b_h)(a_l - b_l)$
$= (50, 42, 168, 33)_{256}$

1	176	254	234				
		50	42	168	33		
				83	222	83	133
1	177	49	20	251	255	83	133

CR

Performing Modular Reduction

- Divide and get remainder
  (repeated subtraction)

Alternatively, we could use Montgomery multiplication that will not require modular reduction.

Montgomery Multiplication

c = a x b mod m

Select $R = 2^x$, $\gcd(R, m) = 1, R$ slightly greater than m

Use Extended Euclidean Algorithm to find R^{-1} and m'

$s.t \quad R \cdot R^{-1} - m \cdot m' = 1$

Convert multiplicands to Montgomery domain

$\overline{a} = aR \bmod m$

$\overline{b} = bR \bmod m$

$Note\ that \quad c = \overline{a} \cdot \overline{b} \cdot R^{-2} \bmod m$

The Montgomery multiplier computes

$\overline{c} = \overline{a} \cdot \overline{b} \cdot R^{-1} \bmod m$

No specific benefits this way

CR

Montgomery's Trick

$$Montgomery's\ \ trick$$

$$1)\ \ t=\bar{a}\cdot\bar{b}$$

$$2)\ u=(t+((t \bmod r)\cdot m' \bmod r)\cdot m)/r$$

$$3)\ \ if\ \ (u\geq m)\ \text{return}\ \ u-m;\ else\ \text{return}\ u.$$

Montgomery's Trick (why it works)

$Montgomery's\ \ trick$

1) $t = \bar{a} \cdot \bar{b}$

2) $u = (t + ((t \bmod R) \cdot m' \bmod R) \cdot m) / R$

3) $if\ (u \geq m)$ return $u - m$; $else$ return u.

- First note that $R \mid t$
- Then $R \mid (t \cdot m' \cdot m \bmod R)$

  $....this\ \ follows\ because\ RR^{-1} - m'm = 1$; then take $\bmod R$
- Therefore $R \mid (t + t \cdot m' \cdot m \bmod R)$

  $....the\ division\ in\ step\ 2\ is\ valid$
- $u \cdot R = t + t \cdot m' \cdot m \bmod R$

  $= t + t \cdot m' \cdot m$

  $= t + k \cdot m$

  $= t \bmod m$

See google groups for more details

Speeding RSA decryption with CRT

- Decryption is done as follows :

$$x = y^a \bmod n$$

- Bob can also decrypt by using CRT

$$x = y^a \bmod p$$

$$x = y^a \bmod q$$

(since he knows the factors of n, i.e. p,q)

- CRT turns out to be much faster since the size (in bits) of p and q is about ½ that of n

CR

Multi-precision libraries

- GMP : GNU Multi-precision library
- Make use of Intel's SSE/AVX instructions
  – These are SIMD instructions that have large registers (128, 256, 512 bit)
- Crypto libraries
  – OpenSSL, PolarSSL, NaCL, etc.

RSA Speeds

Table 1: Evaluation of RSA on Intel 64-bit System.

Input Size	Without CRT (Seconds)	With CRT (Seconds)
128	0.000074	0.000022
256	0.000523	0.000299
512	0.001707	0.001155
1024	0.012381	0.010940
2048	0.091174	0.077656

Table 2: Evaluation of RSA on Intel 32-bit System.

Input Size	Without CRT (Seconds)	With CRT (Seconds)
128	0.000730	0.000229
256	0.004576	0.002664
512	0.034216	0.026493
1024	0.278812	0.213975
2048	2.280441	1.908730

RSA Speeds

32 Bit ARM Cortex

Table 3: Evaluation of RSA on LPCXpresso 1347.

Input Size	Without CRT (Seconds)	With CRT (Seconds)
128	5.799000	2.344000
256	37.806000	24.069000
512	326.877000	231.231000

16 Bit TI Micro-controller

Table 4: Execution Time on Varying Input Size on MSP-430

Input Size	Without CRT (Seconds)	With CRT (Seconds)
128	5.06	5.029
256	36.025	33.044
512	260.007	254.13
1024	2011	2028

Finding Primes

Test for Primes

- How to generate large primes?
  - Select a random large number
  - Test whether or not the number is prime
- What is the probability that the chosen number is a prime?
  - Let $\pi(N)$ be the number of primes $< N$
  - From number theory, $\pi(N) \approx N/\ln N$
  - Therefore probability of a random number ($< N$) being a prime is $1/\ln N$
    - As N increases, it becomes increasingly difficult to find large primes

GIMPS

- There are infinite prime numbers (proved by Euclid)
- Finding them becomes increasingly difficult as N increases
- GIMPS : Great Internet Mersenne Prime Search
  - Mersenne Prime has the form $2^n - 1$
  - Largest known prime (found in 2016) has 22 million digits $2^{274,207,281} - 1$
    - $3000 to beat this ☺

https://en.wikipedia.org/wiki/Largest_known_prime_number

Primality Tests with Trial Division

- School book methods (trial division)
  - Find if N divides any number from 2 to N-1
  - find if N divides any number from 2 to $N^{1/2}$
  - Find if N divides any prime number from 2 to $N^{1/2}$
  - Too slow!!!
    - Need to divide by N-1 numbers
    - Need to divide by $N^{1/2}$ numbers
    - Need to divide by $(N/\ln N)^{1/2}$ primes
      - For example, if n is approx 2^{1024}, then need to check around 2^{507} numbers
    - Need something better for large primes
      - Randomized algorithms

CR

Randomized Algorithms for Primality Testing

- Monte-carlo Randomized Algorithms
  - Always runs in polynomial time
  - May produce incorrect results with bounded probability
  - Yes-based Monte-carlo method
    - Answer YES is always correct, but answer NO may be wrong
  - No-based Monte-carlo method
    - Answer NO is always correct, but answer YES may be wrong

CR

Finding Large Primes (using Fermat's Theorem)

$$
\begin{array}{l}
is\_prime(n)\{ \\
\quad pick\ a \leftarrow Z_n \\
\quad if\ (a^{n-1} \equiv 1 \bmod n) \\
\quad\quad return\ TRUE \\
\quad else \\
\quad\quad return\ FALSE \\
\}
\end{array}
$$

If n is prime, then $a^{n-1} \equiv 1 \bmod n$ is **true** for any 'a'

If n is composite $a^{n-1} \equiv 1 \bmod n$ is **false** but **may be true** for some values of a.

For example: n = 221 (13*17)
and a = 38 then
$38^{220} \bmod 221 \equiv 1$.

We need to increase our confidence with more values of a

CR

Fermat's Primality Test

- Increasing confidence with multiple bases

$$
\begin{array}{l}
primality\_test(n)\{ \\
\quad c = 0 \\
\quad for(i = 0; i < 1000; ++i)\{ \\
\quad\quad if\,(is\_prime(n) == FALSE) \\
\quad\quad\quad return\; COMPOSITE \\
\quad \} \\
\quad return\; probably\; PRIME \\
\}
\end{array}
$$

CR

Carmichael Number

Some composites act as primes.

Irrespective of the 'a' chosen, the test $a^{n-1} \equiv 1 \bmod n$ passes.

for example Carmichael numbers are composite numbers which satisfy Fermat's little theorem irrespective of the value of a.

Eg. 561 = 3 x 11 x 17

CR

Strong probable-primality test

- If n is prime, the square root of a^{n-1} is either +1 or -1

$$let\ a^{\frac{n-1}{2}} = b$$

$$b^2 \equiv 1 \bmod n$$

$$b^2 - 1 \equiv \ \bmod n$$

$$(b+1)(b-1) \equiv 0 \bmod n$$

$$either\,(b+1) \equiv 0 \bmod n\ or\,(b-1) \equiv 0 \bmod n$$

Miller-Rabin Primality Test

- Yes-base primality test for composites
- Does not suffer due to Carmichael numbers
- Write n-1 = 2^sd
  - where d is odd and s is non-negative
  - n is a composite if

$$a^d \neq 1 \bmod n \quad and \; (a^d)^{2^r} \neq -1 \bmod n$$
$$for\, all\, numbers\, r\, less\, than\, s$$

CR

Proof of Miller-Rabin test

- Write n-1 = $2^s d$

$$a^d \neq 1 \bmod n \quad and \quad (a^d)^{2^r} \neq -1 \bmod n$$
$$for\ all\ number\ r\ less\ than\ s$$

- Proof: We prove the contra-positive. We will assume n to be prime. Thus,

$$a^d \equiv 1 \bmod n \quad or \quad (a^d)^{2^r} \equiv -1 \bmod n$$
$$for\ some\ number\ r\ less\ than\ s$$

CR

Proof of Miller-Rabin test

Proof: We prove the contra-positive. We will assume n to be prime. Thus we prove,

$$a^d \equiv 1 \bmod n \quad or \ (a^d)^{2^r} \equiv -1 \bmod n$$

$$for\ some\ number\ r\ less\ than\ s$$

- Consider the sequence :

1 (Fermat 's)

$$a^d, a^{2^1 d}, a^{2^2 d}, a^{2^3 d}, \cdots\cdots, a^{2^{s-1} d}, a^{2^s d}$$

  - The roots of $x^2 = 1 \bmod n$ is either +1 or -1
  - In the sequence, if a^d is 1, then all elements in the sequence will be 1
  - If a^d is not 1, then there should be some element in the sequence which is -1, in order to have the final element as 1

Miller-Rabin Algorithm (test for composites)

Input n

$T1$. Find an odd integer d such that $n-1=2^s d$

$T2$. Select at random a nonzero $a \in Z_n$

$T3$. Compute $b = a^d \bmod n$

$\quad$ *If* $b = \pm 1$, *return* 'n is prime'

$T4$. For $i = 1, \cdots, r-1$, calculate $c \equiv b^{2^i} \bmod n$

$\quad$ *If* $c = -1$, *return* 'n is prime'

$T5$. Otherwise *return* 'n is composite'

- Pr(input=composite | ans=composite)= 1
- Pr(ans=prime | input=composite)<1/2
- Pr(input=composite | ans=prime) $\leq$ 1/4

Quadratic Residues

Definition. *Let $a, m \in \mathbb{N}$. Then a is a* **quadratic residue of** *m iff* $(a, m) = 1$ *and there is an* $x \in \mathbb{Z}$ *so that* $x^2 \equiv a \pmod{m}$.

a cannot be 0

- Example : m=13, square elements in Z_{13}.

  1,4,9, 3, 12, 10, 10, 12, 3, 9, 4, 1

  The quadratic residues Z_{13} are therefore

  {1, 4, 3, 9, 10, 12}

If an element is not a quadratic residue, then it is a **quadratic non-residue**

quadratic non-residues in Z_{13} are {2, 5, 6, 7, 8, 11}

CR

Legendre Symbol

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p \mid a \\ 1 & \text{if } a \text{ is a } QR \bmod p \\ -1 & \text{if } a \text{ is a } QNR \bmod p \end{cases}$$

Given p is an odd prime

Euler's Criteria

A result from Euler

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p$$

$$\text{when } p \mid a$$
$$a^{\frac{p-1}{2}} \equiv 0 \bmod p$$

$$\text{when } a \text{ is a } QR, \exists x \in Z_p \ s.t.\ a \equiv x^2 \bmod p$$
$$\Rightarrow a^{\frac{p-1}{2}} \equiv x^{\frac{2(p-1)}{2}} \bmod p$$
$$\equiv x^{p-1} \bmod p$$
$$\equiv 1$$

CR

when Quadratic Non Residue

when a *is a QNR, no such* $x \in Z_p$ *exists s.t.* $a \equiv x^2 \bmod p$

$consider: \ a^{\frac{p-1}{2}} \bmod p \qquad (note\ p-1\ is\ even, if\ p\ is\ an\ odd\ prime)$

$squaring: \ a^{p-1} \bmod p \equiv 1$

$$so, \left(a^{\frac{p-1}{2}}\right)^2 \equiv 1 \bmod p$$

$$Thus, \ a^{\frac{p-1}{2}} \equiv \pm 1 \bmod p$$

$a^{\frac{p-1}{2}} \not\equiv 1 \bmod p$, since a is not a QR

$$Thus \ a^{\frac{p-1}{2}} \equiv -1 \bmod p$$

CR

Examples

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p$$

$$4 \; is\ a\ QR \bmod 13$$

$$4^{\frac{13-1}{2}} \bmod 13 \equiv 4^6 \bmod 13 \equiv 1$$

$$5 \; is\ a\ QNR \bmod 13$$

$$5^6 \bmod 13 \equiv 12 \bmod 13 \equiv -1$$

Congruence always holds when n is an odd prime

Euler's Witness

$$7^{\frac{15-1}{2}} \bmod 15 \equiv 7^7 \bmod 15 \equiv -2$$

Euler's Liar

$$14^{\frac{15-1}{2}} \bmod 15 \equiv 14^7 \bmod 15 \equiv -1$$

Congruence **may or may not** hold when n is not prime

Solovay Strassen Primality Test

$SOLOVAYSTRASSEN(n)\{$

choose a random integer a such that $1 \leq a \leq n\text{-}1$

compute $x = \left(\dfrac{a}{n}\right)$

$if\ (x = 0)\ return\ COMPOSITE$

$compute\ y = a^{\frac{n-1}{2}} \bmod n$

$if\ (x \equiv y \bmod n)\ return\ possibly PRIME$

$else\ return\ COMPOSITE$

$\}$

How to compute

Legendre's symbol

error probability is at most ½

CR

Jacobi Symbol

- Jacobi Symbol is a generalization of the Legendre symbol
- Let n be any positive odd integer and a>=0 any integer. The Jacobi symbol is defined as:

Suppose n is an odd positive integer with prime factorization

T

$$n = p_1^{e_1} \times p_2^{e_2} \times p_3^{e_3} \times p_4^{e_4} \ldots$$

Then,

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \times \left(\frac{a}{p_2}\right)^{e_2} \times \left(\frac{a}{p_3}\right)^{e_3} \times \left(\frac{a}{p_4}\right)^{e_4} \times \cdots$$

CR

Jacobi Properties

P1. $If\ a \equiv b \bmod n\ then \left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$

P2. $\left(\frac{2}{n}\right) = \begin{cases} 1 & if\ n \equiv \pm 1 \bmod 8 \\ -1 & if\ n \equiv \pm 3 \bmod 8 \end{cases}$

P3. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$

P4. $if\ a\ is\ even,\ a = 2^k t,\ \left(\frac{a}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$

P5. $if\ a\ is\ odd,$

$$\left(\frac{a}{n}\right) = \begin{cases} -\left(\frac{n}{a}\right) & if\ n \equiv a \equiv 3 \bmod 4 \\ \left(\frac{n}{a}\right) & otherwise \end{cases}$$

CR

Computing Jacobi

$$\left(\frac{1001}{9907}\right) = \left(\frac{7}{9907}\right)\left(\frac{11}{9907}\right)\left(\frac{13}{9907}\right).$$

$$\left(\frac{7}{9907}\right) = -\left(\frac{9907}{7}\right) = -\left(\frac{2}{7}\right) = -1.$$

$$\left(\frac{11}{9907}\right) = -\left(\frac{9907}{11}\right) = -\left(\frac{7}{11}\right) = \left(\frac{11}{7}\right) = \left(\frac{4}{7}\right) = 1.$$

$$\left(\frac{13}{9907}\right) = \left(\frac{9907}{13}\right) = \left(\frac{1}{13}\right) = 1.$$

$$\left(\frac{1001}{9907}\right) = -1.$$

From the theorem

P5, P1, then P2

P5, P1, P5, P1, P3, P2

P5, P1
and 1 is a QR mod 13

CR

Factoring Algorithms

Factorization to get the private key

- Public information (n, b)
- If Mallory can factorize n into p and q then,
  - She can compute $\phi(n) = (p-1)(q-1)$
  - She can then computethe private key by finding $a \equiv b^{-1} \bmod \phi(n)$

How to factorize n?

Trial Division

Fundamental theorem of arithmetic

Any integer number (greater than 1) is either prime or a product of prime powers

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k}$$

```
def trial_division(n):
    """Return a list of the prime factors for a natural number."""
    if n < 2:
        return []
    prime_factors = []
    for p in prime_sieve(int(n**0.5) + 1):
        if p*p > n: break
        while n % p == 0:
            prime_factors.append(p)
            n //= p
    if n > 1:
        prime_factors.append(n)
    return prime_factors
```

prime generation algorithm

Prime factors of n cannot be greater than $\lfloor \sqrt{n} \rfloor$

n = n / p : remove this factor from n

Running Time of algorithm order of $\pi(2^{n/2})$

CR

Pollard p-1 Factorization

$n = p \times q$

1

choose a random integer $a\,(1 < a < n)$.
If $\gcd(a,n) \neq 1$, then a is a prime factor.
However, this is most likely not the case.

2

Suppose we select some L and compute d=gcd(a^L-1,n)
if $1 < d < n$ then we have factored n
$d \mid n$ *and* $d \mid (a^L - 1)$
d has to be the prime p or the prime q

why a^L-1?
since d is prime and $d \mid (a^L - 1)$
$a^L \equiv 1 \bmod d$
$\varphi(d) \mid L \Rightarrow (d-1)k = L$
Thus we need to find L which is some factor of $(d-1)$.

3

If $\gcd(a^L - 1, n) = n$
This is possible only when $p \mid n$ and $p \mid a^L - 1$ (*or* $q \mid n$ and $q \mid a^L - 1$)
and $a^L - 1 > n$

How to choose L?
No easy way, trial and error!!
Factorials have a lot of divisors. So that is a nice way.
So, take L as a factorial of some number r.

CR

Pollard p-1 Factorization

Pollard p-1 factorization for n.

$S1.\ a \leftarrow 2$

$S2.$ if $\gcd(a, n) > 1$, then this gcd is a prime factor of n, we are done.

$S3.$ compute $d \leftarrow \gcd(a^{r!}\text{-}1, n)$

  $if\ d = n,$ start again from $S1$ with next value of a

  $else\ if\ d = 1$, increment r and repeat $S3$

  $else\ d$ is the prime factor of n; we are done!

r = 2,3, 4, …..

1. Will the algorithm terminate?
2. When will we choose the next value of a? (will we get an infinite loop?)

When r = d-1 then L = r! = (d-1)! = d-1(d-2)! = (d-1)k
(d-1) | L → we will get the gcd($a^{k(d-1)}$, n) = n or its prime factor.

CR

Pollard Rho Algorithm

- Form a sequence S1 by selecting randomly (all different) from the set Z_n

$$S1 = x_0, x_1, x_2, x_3, x_4, \cdots$$

- Also assume we magically find a new sequence S2 comprising of

$$S2 = \bar{x}_0, \bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \cdots \text{ where } \left\{ \begin{array}{l} \bar{x}_0 \equiv x_0 \bmod p \\ \bar{x}_1 \equiv x_1 \bmod p \\ \bar{x}_2 \equiv x_2 \bmod p \\ \bar{x}_3 \equiv x_3 \bmod p \\ \bar{x}_4 \equiv x_4 \bmod p \end{array} \right.$$

- If we keep adding elements to S1, we will eventually find an x_i and x_j ($i \neq j$) such that $\bar{x}_i = \bar{x}_j$ When this happens,

$$p \mid (x_i - x_j)$$

$$\because p \mid n \; also, \; \gcd((x_i - x_j), n) \; is \; p. We \; found \; a \; factor \; of \; n!!$$

Doing without magic

- Form a sequence S1 by selecting randomly (with replacement) from the set Z_n

$$S1 = x_0, x_1, x_2, x_3, x_4, \cdots$$

- For every pair i,j in the sequence compute

$$d \leftarrow \gcd((x_i - x_j, n)$$

- If d > 1 then it is a factor of n

CR

Selecting elements of S1

To choose the next element of S1, Pollard suggests using a function $f : Z_n \to Z_n$ with requirement that the output looks random.

Example : $f(x) = x^2 + 1 \bmod n$

$$S1 = \left(\begin{cases} x_0 & where\, x_0\, is\, chosen\, randomly\, from\, Z_n \\ x_i & i > 0\ and\ x_i = f(x_{i-1}) \end{cases} \right)$$

CR

Example

- N= 82123, x_0 = 631, f(x) = x^2 + 1

This column is just for understanding.
In reality we will not know this

i	$x_i \mod N$	$\overline{x_i} = x_i \mod p$
0	631	16
1	69670	11
2	28986	40
3	69907	2
4	13166	5
5	64027	26
6	40816	21
7	80802	32
8	20459	0
9	71874	1

i	$x_i \mod N$	$\overline{x_i} = x_i \mod p$
10	6685	2
11	14314	5
12	75835	26
13	37782	21
14	17539	32
15	65887	0
16	74990	1
17	45553	2
18	73969	5
19	50210	26

Drawback…
Large number of GCD Computations. 55 gcd computations in this case

Can we reduce the number of gcd computations?

Given x_i mod N, we compute gcds of every pair until we find a gcd greater than 1

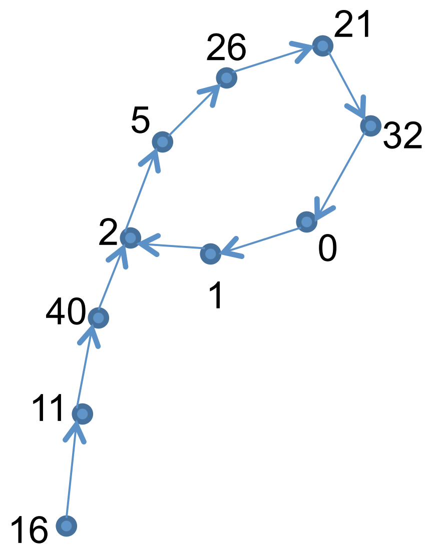
$$\gcd(x_3 - x_{10}, N) = \gcd(63222, 82123) = 41$$

A factor of N

The Rho in Pollard-Rho

- N= 82123, x_0 = 631, f(x) = x^2 + 1

i	$x_i \mod N$	$\overline{x_i} = x_i \mod p$
0	631	16
1	69670	11
2	28986	40
3	69907	2
4	13166	5
5	64027	26
6	40816	21
7	80802	32
8	20459	0
9	71874	1
10	6685	2
11	14314	5
12	75835	26
13	37782	21
14	17539	32
15	65887	0
16	74990	1
17	45553	2
18	73969	5
19	50210	26



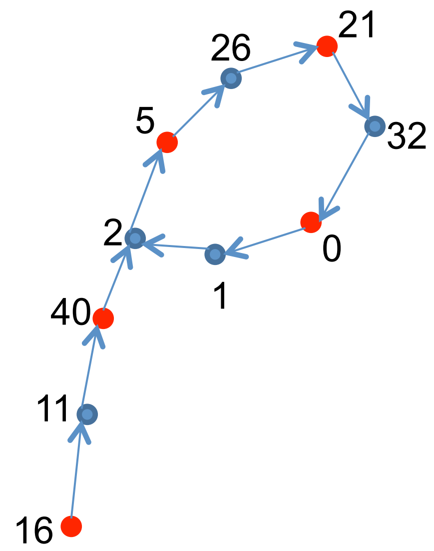
$$\bar{x}_t = \bar{x}_{t+l} \bmod p$$

- The smallest value of t and l, for which the above congruence holds is t=3, l=7
- For l=7, all values of t > 3 satisfy the congruence
- This leads to a cycle as shown in the figure (and a shape like the Greek letter rho)

$$\bar{x}_j = \bar{x}_{j+l} \bmod p \quad t \ge 3$$

Reducing gcd computations

- GCD computations can be expensive.
- Use Floyd's cycle detection algorithm to reduce the number of GCD computations.



$$choose\,a\,random\;x_0 = y_0 \in Z_n$$

loop

$$x_i = f(x_{i-1})$$

$$y_i = x_{2i} = f(f(y_{i-1}))$$

$$If\; d = \gcd(x_i - y_i, N) > 0, return\; d$$

claim : The first time $x_i = y_i$ mod p occurs when $i \leq t + \ell$

This means that we get a collision before x completing an entire circle

The first time $x_i = y_i \bmod p$ occurs is when $i \leq t + l$

- l is the number of points in the cycle
- t is the smallest value of i such that $x_i \equiv y_i \bmod p$

x_i and y_i meet at the same point in the cycle
Therefore, y_i must have traversed (some) cycles more

$$x_i \equiv y_i \bmod N$$

$$x_i \equiv x_{2i} \bmod N$$

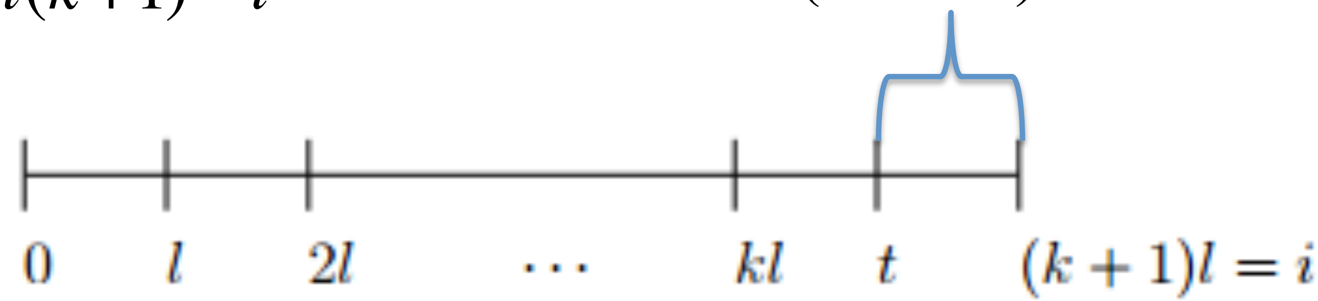
$$l \mid (2i - i)$$

$$l \mid i \Rightarrow l(k+1) = i$$

$$consider\ i = (k+1)l = t + (-t \bmod l)$$

$$\leq t + l$$

$(-t \bmod l)$



Expected number of operations before a collision

- Can be obtained from Birthday paradox to be $\sqrt{p}$

CR

Congruences of Squares

- Given N=p x q, we need to find p and q
- Suppose we find an x and y such that $\boxed{x^2 \equiv y^2 \bmod N}$
- Then,

$$\boxed{N \mid (x^2 - y^2) \Rightarrow N \mid (x-y)(x+y)}$$

- This implies,

$$\boxed{\gcd(N,(x-y)) \text{ or } \gcd(N,(x+y)) \text{ factors } N}$$

CR

Example

- Consider N = 91

$$10^2 \equiv 3^2 \bmod 91$$
$$91 \mid (10-3)(10+3)$$
$$91 \mid (7 \times 13)$$
$$\gcd(91,13) = 13$$
$$\gcd(91,7) = 7$$

$$34^2 \equiv 8^2 \bmod 91$$
$$91 \mid (34+8)(34-8)$$
$$91 \mid 42 \times 26$$
$$\gcd(91,26) = 13$$
$$\gcd(91,42) = 7$$

So… we can use x and y to factorize N.

$$x^2 \equiv y^2 \bmod N$$

But how do we find such pairs?

CR

Another Example

- N = 1649

$$41^2 \equiv 32 \bmod 1649$$
$$43^2 \equiv 200 \bmod 1649$$

32 and 200 are not perfect squares.
However (32x200 = 6400) = 80^2
is a perfect square

$$(41 \times 43)^2 \equiv (32 \times 200) \bmod 1649$$
$$\equiv 80^2 \bmod 1649$$

Thus, it is possible to combine non-squares to form a prefect square

the examples are borrowed from Mark Stamp (http://cs.sjsu.edu/faculty/stamp/)

Forming Perfect Squares

Recall, Fundamental theorem of arithmetic

Any integer number (greater than 1) is either prime or a product of prime powers

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k}$$

Thus, a number is a perfect square if it prime factors have even powers.

$$e_1, e_2, e_3, \dots \ is\, even$$

Thus,

$32 = 2^5 5^0$ not a perfect square

$200 = 2^3 5^2$ not a perfect square

$(32x200) = 2^5 5^0 \times 2^3 5^2 = 2^8 5^2 = (2^4 5^1)^2$ is a prefect square

CR

Dixon's Random Squares Algorithm

1. Choose a set B comprising of 'b' smallest primes. Add -1 to this set.

   (A number is said to be b-smooth, if its factors are in this set)

2. Select an r at random
   - Compute $y = r^2 \bmod N$
   - Test if y factors completely in the set B.
   - If NO, then discard. ELSE save (y, r) (these are called B-smooth numbers)

3. Repeat step 2, until we have b+1 such (y,r) pairs

4. Solve the system of linear congruencies

CR

Example

- N = 1829
- b = 6 B = {-1, 2,3,5,7,11,13}
- Choose random values of r, square and factorize

$$42^2 = 1764 = -65 = -1 \cdot 5 \cdot 13 \pmod{1829}$$
$$43^2 = 20 = 2^2 \cdot 5 \pmod{1829}$$
$$60^2 = 1771 = -58 = -1 \cdot 2 \cdot 29 \pmod{1829}$$
$$61^2 = 63 = 3^2 \cdot 7 \pmod{1829}$$
$$74^2 = 1818 = -11 = -1 \cdot 11 \pmod{1829}$$
$$75^2 = 138 = 2 \cdot 3 \cdot 23 \pmod{1829}$$
$$85^2 = 1738 = -91 = -1 \cdot 7 \cdot 13 \pmod{1829}$$
$$86^2 = 80 = 2^4 \cdot 5 \pmod{1829}$$

All numbers are 6-smooth except 60 and 75. Leave these and consider all others

Check Exponents

	-1	2	3	5	7	11	13
-65	1	0	0	1	0	0	1
20	0	2	0	1	0	0	0
63	0	0	2	0	1	0	0
-11	1	0	0	0	0	1	0
-91	1	0	0	0	1	0	1
80	0	4	0	1	0	0	0

Check Exponents

	-1	2	3	5	7	11	13
-65	1	0	0	1	0	0	1
20	0	2	0	1	0	0	0
63	0	0	2	0	1	0	0
-11	1	0	0	0	0	1	0
-91	1	0	0	0	1	0	1
80	0	4	0	1	0	0	0

Find rows where exponents sum is even
-65, 20, 63, -91

sum	2	2	2	2	2	0	2

$$(42 \times 43 \times 61 \times 85)^2 \equiv (-1 \times 2 \times 3 \times 5 \times 7 \times 13)^2 \bmod 1829$$

$$1459^2 \equiv 901^2 \bmod 1829$$

CR

Final Steps

$$(42\times 43\times 61\times 85)^2 \equiv (-1\times 2\times 3\times 5\times 7\times 13)^2 \bmod 1829$$

$$1459^2 \equiv 901^2 \bmod 1829$$

$$1829 \mid (1459+901)(1459-901)$$

$$\Rightarrow 1829 \mid 2360 \qquad \gcd(1829,2360) = 59$$

$$\Rightarrow 1829 \mid 558 \qquad \gcd(1829,558) = 31$$

$$\textit{Thus} \;\; 1829 = 59\times 31$$

CR

State of the Art Factorization Techniques

- Quadratic Sieve
  - Fastest for less than 100 digits
- General Number field Sieve
  - Fastest technique known so far for greater than 100 digits
  - Open source code (google GGNFS)
- RSA factoring challenge
  - Best so far is 768 bit factorization
  - Current challenges 896 bits (reward $75,000), 1024 bit ($100,000)

https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

RSA Attacks

attacks that don't require factorization algorithms

Φ(n) leaks

- If an attacker gets Φ(n) then n can be factored

$$n = pq \qquad\qquad q = n/p$$

$$\phi(n) = (p-1)(q-1)$$

$$= pq - (p+q) + 1$$

$$\phi(n) = n - (p + \frac{n}{p}) + 1$$

$$p^2 - (n - \phi(n) + 1)p + n = 0$$

Solve to get p (a factor of n)

CR

square roots of 1 mod n

There are two trivial and two non-trivial solutions for $y^2 \equiv 1 \bmod n$

The trivial solutions are +1 and -1

By CRT, these congruences are equivalent

$$y^2 \equiv 1 \bmod n \quad \langle = \rangle \begin{cases} y^2 \equiv 1 \bmod p \\ y^2 \equiv 1 \bmod q \end{cases}$$

$$\begin{cases} y \equiv 1 \bmod p \\ y \equiv -1 \bmod p \end{cases}$$

$$\begin{cases} y \equiv 1 \bmod q \\ y \equiv -1 \bmod q \end{cases}$$

To get the non-trivial solutions solve using CRT

$$\begin{aligned} y &\equiv +1 \bmod p \\ y &\equiv -1 \bmod q \end{aligned}$$

$$\begin{aligned} y &\equiv -1 \bmod p \\ y &\equiv +1 \bmod q \end{aligned}$$

CR

Example

- n=403 = 13 x 31
- To get the non-trivial solutions of $y^2 \equiv 1 \bmod n$ solve using CRT

$$y \equiv +1 \bmod p \qquad y \equiv -1 \bmod p$$
$$y \equiv -1 \bmod q \qquad y \equiv +1 \bmod q$$

$$(31 \cdot 31^{-1} \bmod 13 - 13 \cdot 13^{-1} \bmod 31) \bmod 403$$
$$(31 \cdot 8 - 13 \cdot 12) \bmod 403 \equiv 92$$
$$403 - 91 = 311$$

The non-trivial solutions are 92 and 311

$Note: 92^2 \equiv 311^2 \equiv 1 \bmod 403$

What happens when we solve $y \equiv +1 \bmod p$, $y \equiv +1 \bmod q$

Decryption exponent leaks

- If the decryption exponent 'a' leaks, then n can be factored
- The attacker can then compute ab

$$ab \equiv 1 \bmod \phi(n) \qquad\qquad k\phi(n) = (ab-1)$$

- Now, for any message x ≠ 0

$$x^{ab-1} \equiv 1 \bmod n$$

- Attack Plan, take square root : $y \equiv x^{\frac{ab-1}{2}} \bmod n$

  i.e., $y^2 \equiv 1 \bmod n \;\; \Rightarrow \;\; n \mid (y^2 - 1)$

$$\Rightarrow \;\; n \mid (y-1)(y+1)$$

$$\gcd(n, y-1) \; \textit{is a factor of } n$$

However we need

$$y \neq \pm 1$$

to have a non-trivial result

CR

The Attack (basic idea)

$1.\text{given } a \text{ compute } ab-1$

$2.\text{Represent } t = \dfrac{ab-1}{2}$

$3.\text{ choose any message } x$

$4.\text{put } y = x^{t} \bmod n$

$5.\text{compute } d \leftarrow \gcd(y-1, n)$

$6.\textit{if } d \neq 1, \textit{ return} \text{"a factor of n is d"}; \textit{ exit}$

$7.\textit{if } (t \text{ is even}) \; t = t/2; \; \textit{goto} \text{ step } 4$

$\quad \textit{else return}\text{" }\textit{failure}\text{"}$

we assume we know the private key a

$$ab \equiv 1 \bmod \phi(n)$$

$$k\phi(n) = ab - 1$$

$$y_1 = x^{\frac{ab-1}{2}} \equiv 1 \bmod n$$

$$thus, (y_1^2 - 1) \equiv 0 \bmod n$$

$$n \mid (y+1)(y-1)$$

This will only work if y ≠±1 mod n.
If y = ±1 mod n. then goto step 7

Probability of success of the attack is at-least 1/2

CR

Example

- N=403, b=23, a=47

$$t = ab - 1 = 1080 \qquad x = 2$$

$$loop1: t = \frac{1080}{2} = 540 \qquad y \equiv x^t \bmod 403 = 2^{540} \bmod 403 \equiv 1$$

$$loop2: t = \frac{540}{2} = 270 \qquad y \equiv x^t \bmod 403 = 2^{270} \bmod 403 \equiv 311$$

$$\gcd(310, 403) = 31 \ (a \ factor \ of \ n)$$

$$t = ab - 1 = 1080 \qquad x = 9$$

$$loop1: t = \frac{1080}{2} = 540 \qquad y \equiv x^t \bmod 403 = 9^{540} \bmod 403 \equiv 1$$

$$loop2: t = \frac{540}{2} = 270 \qquad y \equiv x^t \bmod 403 = 9^{270} \bmod 403 \equiv 1$$

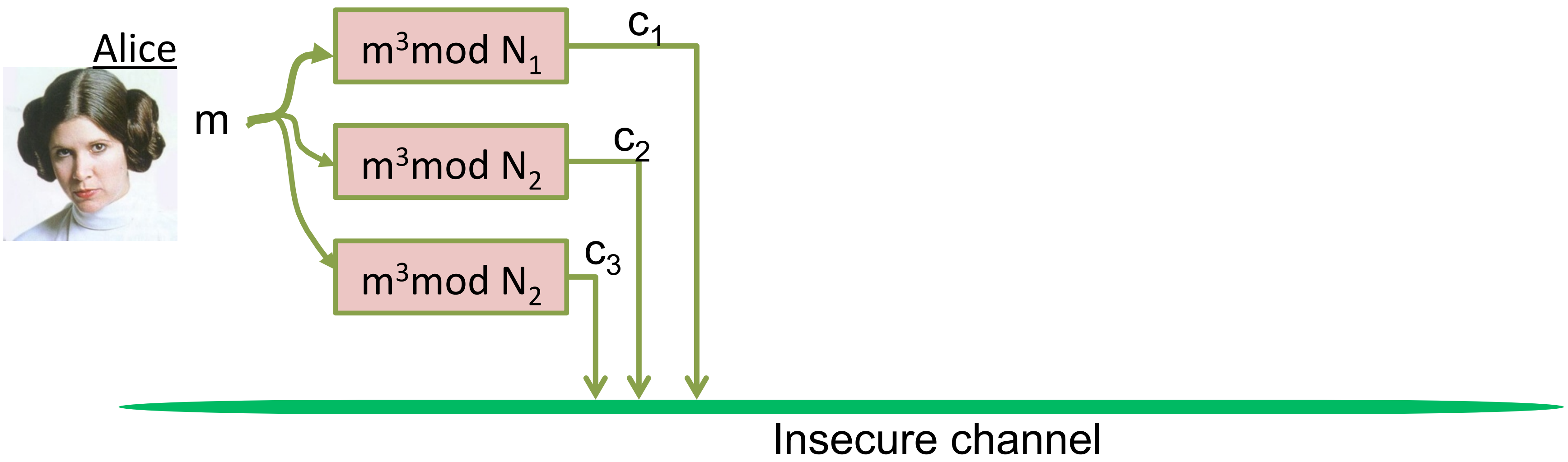
$$loop3: t = \frac{270}{2} = 135 \qquad y \equiv x^t \bmod 403 = 9^{135} \bmod 403 \equiv 1$$

can't divide 135 further. failure

Small Encryption Exponent

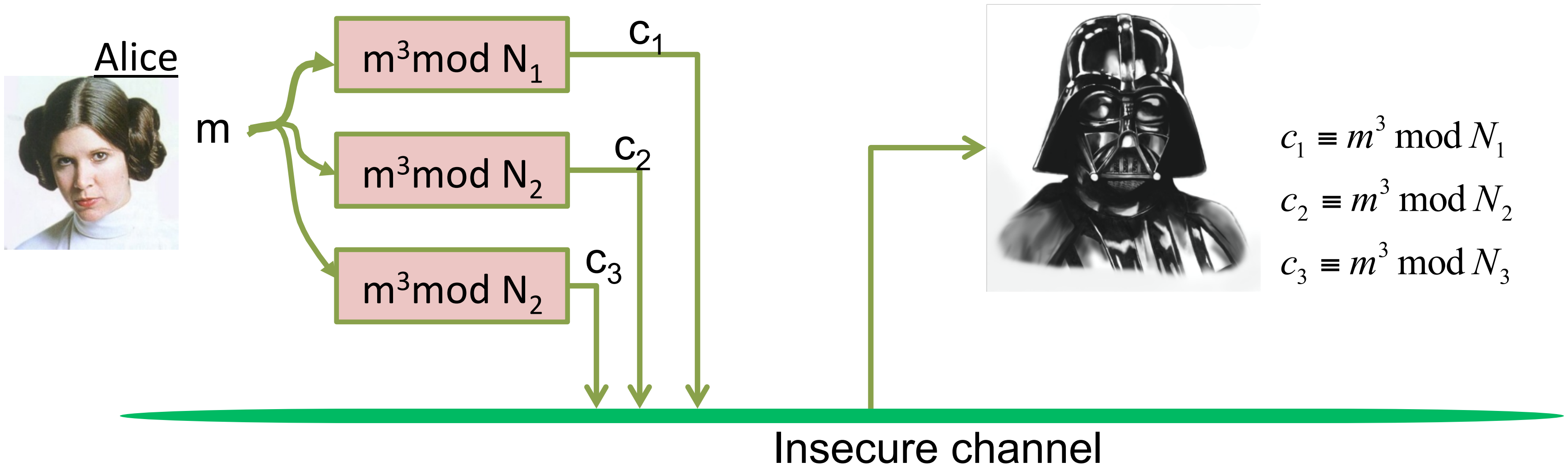
- In order to improve efficiency of encryption, a small encryption exponent is preferred
- However, this can lead to a vulnerability

Small Encryption Exponent



- Consider, Alice sending the same message x to 3 different people.
  - Each having a different N (say N_1, N_2, N_3)
  - But same public key b (say 3)

Small Encryption Exponent



- Consider, Alice sending the same message x to 3 different people.
  - Each having a different N (say N_1, N_2, N_3)
  - But same public key b (say 3)
- This allows Mallory to snoop in and get 3 ciphertexts

Small Encryption Exponent

By CRT

$$\begin{cases} c_1 \equiv m^3 \bmod N_1 \\ c_2 \equiv m^3 \bmod N_2 \\ c_3 \equiv m^3 \bmod N_3 \end{cases} \langle = \rangle \; X \equiv m^3 \bmod (N_1 \cdot N_2 \cdot N_3)$$

- Thus, Mallory can compute X
- Since m < N_1, m<N_2, m<N_3 => n < (N_1 x N_2 x N_3)
- Thus, $X^{1/3}$=m
  - i.e. The message can be decrypted

It is tempting to have small private and public keys, so that encryption or decryption may be carried out efficiently. However you would do this at the cost of security!!

CR

Low Decryption Exponent

- The attack applies when the private key a is small, $a < \frac{\sqrt[4]{n}}{3}$

- In such a case 'a' can be computed efficiently

Partial Information of Plaintexts

Computing Jacobi of the plaintext

$y \equiv x^b \bmod n$ y is the ciphertext; x the message

b is the public key and $\gcd(b, \varphi(n)) = 1$

Thus, $\gcd(b, (p\text{-}1)(q\text{-}1)) = 1$

$(p-1)(q-1)$ is even, therefore b must be odd

consider Jacobi

$$\left(\frac{y}{n}\right) = \pm 1$$

$$\left(\frac{y}{n}\right) = \left(\frac{x}{n}\right)^b = \left(\frac{x}{n}\right)$$

since b is odd

thus, RSA encryption leaks the value of the Jacobi symbol $\left(\frac{x}{n}\right)$

Partial Information of Plaintexts first half or second half?

- given $y = x^b \bmod n$,
  - is it possible to determine if

$$(0 \leq x < n/2) \quad \text{or} \quad (n/2 \leq x < n-1)$$

first half *second half*

- We prove that RSA does not leak this information
  - If there exists an efficient algorithm that can determine if x is in the first or second half then, the entire plaintext can be obtained

CR

Binary Search Trees on x

Consider this function

$$HALF(x) = \begin{cases} 0 & if \;\; 0 \le x < \frac{n}{2} \\ 1 & if \;\; \frac{n}{2} \le x < n-1 \end{cases}$$

example

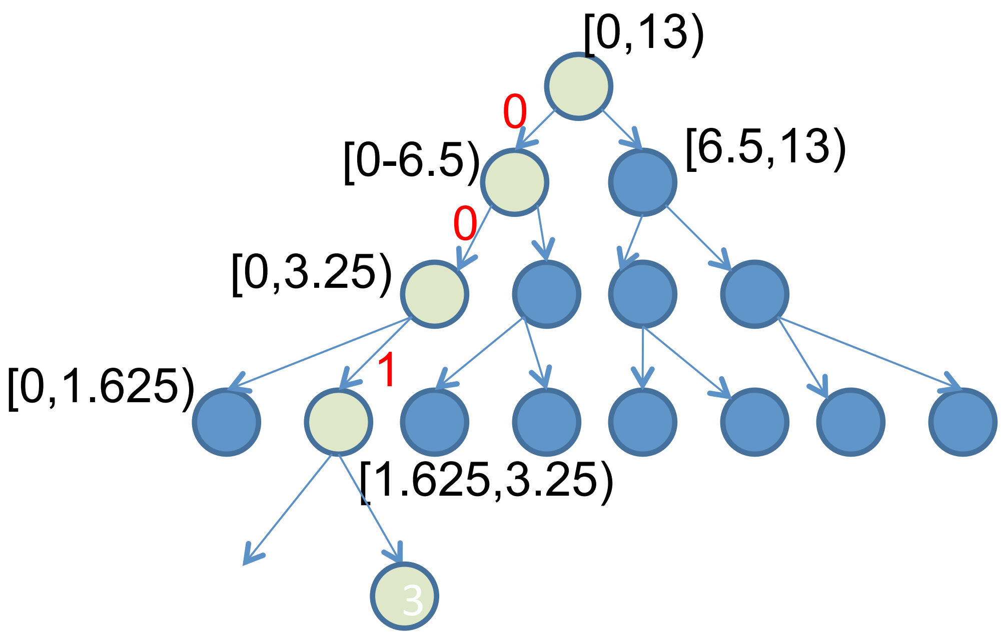
$x = 3 \bmod 13 \qquad HALF(x) = 0$

$2x \equiv 6 \bmod 13 \qquad HALF(2x) = 0$

$4x \equiv 12 \bmod 13 \qquad HALF(4x) = 1$

$8x \equiv 11 \bmod 13 \qquad HALF(8x) = 1$

$16x \equiv 9 \bmod 13 \qquad HALF(16x) = 1$



Partial Information of Plaintexts
(first or second half proof)

- Assume a hypothetical oracle called HALF as follows

$$HALF(n,b,y) = \begin{cases} 0 & if\ \ 0 \le x < \frac{n}{2} \\ 1 & if\ \frac{n}{2} \le x < n-1 \end{cases}$$

$$y \equiv x^b \bmod n$$

$$2^b \cdot y \equiv (2x)^b \bmod n$$

$$4^b \cdot y \equiv (4x)^b \bmod n$$

$$8^b \cdot y \equiv (8x)^b \bmod n$$

$$16^b \cdot y \equiv (16x)^b \bmod n$$

$$\boxed{HALF(y) = 0 \quad \Rightarrow\ x \in \left[0, \frac{n}{2}\right)}$$

$$\boxed{HALF(2^b y) = 0 \quad \Rightarrow\ x \in \left[0, \frac{n}{4}\right)} \qquad \boxed{HALF(2^b y) = 1 \quad \Rightarrow\ x \in \left[\frac{n}{4}, \frac{n}{2}\right)}$$

$$\boxed{HALF(2^{2b} y) = 0 \quad \Rightarrow\ x \in \left[0, \frac{n}{8}\right)} \qquad \boxed{HALF(2^{2b} y) = 0 \quad \Rightarrow\ x \in \left[\frac{n}{8}, \frac{n}{4}\right)}$$

Example

n=1457, b=779, y=722

Algorithm : ORACLE RSA DECRYPTION(n, b, y)

external HALF
$k \leftarrow \lfloor \log_2 n \rfloor$
for $i \leftarrow 0$ **to** k
  do $\begin{cases} h_i \leftarrow \text{HALF}(n, b, y) \\ y \leftarrow (y \times 2^b) \bmod n \end{cases}$
$lo \leftarrow 0$
$hi \leftarrow n$
for $i \leftarrow 0$ **to** k
  do $mid \leftarrow (hi + lo)/2$
    if $h_i = 1$
      then $lo \leftarrow mid$
      else $hi \leftarrow mid$
return ($\lfloor hi \rfloor$)

h_i	i	lo	mid	hi
1	0	0.00	728.50	1457.00
0	1	728.50	1092.75	1457.00
1	2	728.50	910.62	1092.75
0	3	910.62	1001.69	1092.75
1	4	910.62	956.16	1001.69
1	5	956.16	978.92	1001.69
1	6	978.92	990.30	1001.69
1	7	990.30	996.00	1001.69
1	8	996.00	998.84	1001.69
0	9	998.84	1000.26	1001.69
0	10	998.84	999.55	1000.26
		998.84	999.55	999.55

Thus, if we have an efficient function HALF, we can recover the plaintext message.

Man in the Middle Attack

- The process of encryption with a public key cipher

Bob sends his public key

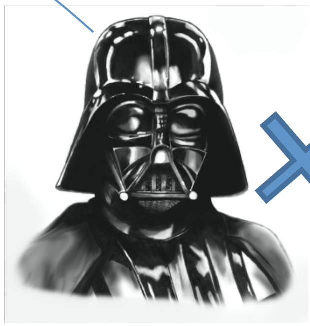
Alice encrypts with Bob's public key

Bob decrypts with his private key

Man in the Middle Attack

- The process of encryption with a public key cipher

Man in the middle
Intercepts messages



Bob
sends his public key

Mallory
sends her public key

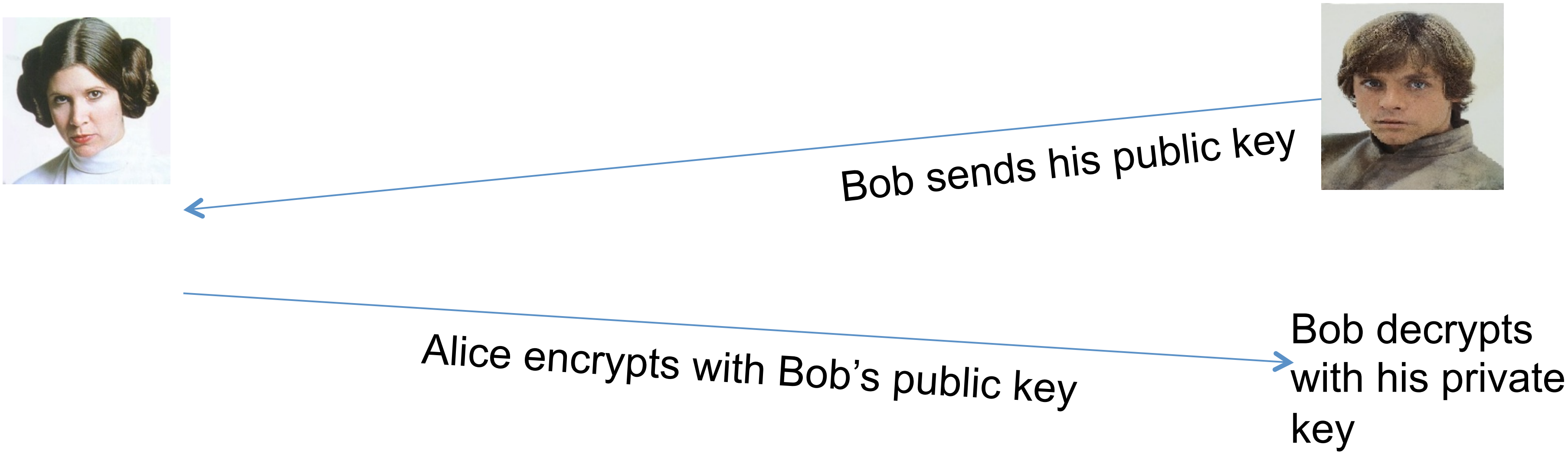
Alice encrypts
with Mallory's public key

Mallory decrypts
with her private
key and re-
encrypts
with Bob's
public key

Bob decrypts
with his private
key

Searching the Message Space

- Suppose message space is small,
  - Mallory can try all possible messages, encrypt them (since she knows Bob's public key) and check if it matches Alice's ciphertext



Bad Prime Generation Algorithms

- Suppose the prime generation was faulty
  - So that, primes generated were always from a small subset
  - Then, RSA can be broken
- Pairwise GCD of over a million RSA modulii collected from the Internet showed that
  - 2 in 1000 have a common prime factor

Ron was Wrong, Whit is right, 2012

Discrete Log Problem, ElGamal, and Diffie Hellman

Primitive Elements of a Group

Let $(G,\cdot)$ be a group of order n.

Let $\alpha \in G$,

The order of α is the smallest integer m such that $\alpha^m = 1$

α is termed as a *primitive element* if it has order n.

If α is a primitive element then

$$\langle \alpha \rangle = \{\alpha^i : 0 \le i \le n-1\} \quad \text{generates all elements in G}$$

Consider $Z^*_{13} = \{1,2,3,\cdots,12\}$

$(Z^*_{13},\cdot)$ forms a group of order 12

Let $7 \in Z^*_{13}$,

$\langle 7 \rangle = \{7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2, 1\}$

<7> has order 12
and generates all elements in Z.
Thus, 7 is a primitive element

CR

Discrete Log Problem

$Let\ (G,\cdot)\ be\ a\ group$

$Let\ \alpha \in G$ be a primitive element in the group with order n

Define the set

$$\langle \alpha \rangle = \{\alpha^i : 0 \le i \le n-1\}$$

For any unique integer $a\ (0 \le a \le n-1)$,

$$\text{let } \alpha^a = \beta$$

Denote $a = \log_\alpha \beta$ as the discrete logarithm of β

Given α and a, it is easy to compute β
Given α and β it is computationally difficult to determine what a was

ElGamal Public Key Cryptosystem

- Fix a prime p (and group Z_p)
- Let $\alpha \in Z_p$ be a primitive element
- Choose a secret 'a' and compute $\beta \equiv \alpha^a \bmod p$

Public keys : α, β, p

Private key : a

Encryption

$choose\ a\, random\ (\text{secret})\ k \leftarrow Z_p$

$e_k(x) = (y_1, y_2)$

$where\ \ y_1 = \alpha^k \bmod p,$

$\qquad\quad y_2 = x \cdot \beta^k \bmod p$

Decryption

$$
\begin{aligned}
d_k(x) &= y_2 (y_1^a)^{-1} \bmod p \\
&= x \cdot \beta^k (\alpha^{ka})^{-1} \bmod p \\
&= x \cdot \alpha^{ka} (\alpha^{ka})^{-1} \bmod p \\
&\equiv x
\end{aligned}
$$

ElGamal Example

- $p = 2579$, $\alpha = 2$ (α is a primitive element mod p)
- Choose a random a = 765
- Compute $\beta \equiv 2^{765} \bmod 2579$

Encryption of message x = 1299

choose a random key k = 853

$$y_1 = 2^{853} \bmod 2579 = 435$$

$$y_2 = 1299 \times 949^{853} = 2396$$

Decryption of cipher (435, 2396)

$$2396 \times (435^{765})^{-1} \bmod p$$

$$= \quad 1299$$

CR

Finding the Log

$$\beta \equiv \alpha^a \bmod p$$

Given α and β it is computationally difficult to determine what a was

- Brute force (compute intensive)

  compute $\alpha, \alpha^2, \alpha^3, \alpha^4 \ldots\ldots$ (until you reach β)

  this would definitely work, but not practical if p is large

  complexity O(p), space complexity O(1)

- Memory Intensive

  precompute $\alpha, \alpha^2, \alpha^3, \alpha^4 \ldots\ldots$ (all values). Sort and store.
  For any given β look up the table of stored values.

  complexity O(1) but space complexity O(n)

CR

Shank's Algorithm
(also known as Baby-step Giant-step)

$$\beta \equiv \alpha^{a} \bmod p$$

Rewrite a as $a = mq + r$

where $\mathrm{m} = \lceil \sqrt{\mathrm{p}} \rceil$

$$\beta \equiv \alpha^{mq}\alpha^{r} \bmod p$$

$$\beta\left(\alpha^{-m}\right)^{q} \equiv \alpha^{r} \bmod p$$

We neither know q nor r, so we need to try out several values for q and r until we find a collision

CR

Shank's Algorithm (example)

- p= 31 and α=3. Suppose β=6.
- What is a?

$$m = \left\lceil \sqrt{31} \right\rceil = 6$$

collision

List 1

$$\alpha \equiv 3$$
$$\alpha^2 \equiv 9$$
$$\alpha^3 \equiv 27$$
$$\alpha^4 = 81 \equiv 19 \bmod 31$$
$$\alpha^5 = 19 \cdot 3 \equiv 26 \bmod 31$$

$$(3^{-1})^6 \bmod 31 = 2$$

List 2

$$\beta(\alpha^{-6})^0 = 6 \cdot 2^0 = 6$$
$$\beta(\alpha^{-6})^1 = 6 \cdot 2^1 = 12$$
$$\beta(\alpha^{-6})^2 = 6 \cdot 2^2 = 24$$
$$\beta(\alpha^{-6})^3 = 6 \cdot 2^3 \equiv 17 \bmod 31$$
$$\beta(\alpha^{-6})^4 = 6 \cdot 2^4 \equiv 3 \bmod 31$$

Thus, m=6, q=4, r=1, a= mq+r = 25

Shank's Algorithm

Algorithm 6.1: SHANKS(G, n, α, β)

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m - 1$
   do compute α^{mj}
3. Sort the m ordered pairs (j, α^{mj}) with respect to their second coordinates, obtaining a list L_1
4. **for** $i \leftarrow 0$ **to** $m - 1$
   do compute $\beta\alpha^{-i}$
5. Sort the m ordered pairs $(i, \beta\alpha^{-i})$ with respect to their second coordinates, obtaining a list L_2
6. Find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
7. $\log_\alpha \beta \leftarrow (mj + i) \bmod n$

Create List 1 (steps 2–3)

Create List 2 (steps 4–5)

Find collision (steps 6–7)

Complexity of Shank's Algorithm

Algorithm 6.1: SHANKS(G, n, α, β)

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m - 1$
   do compute α^{mj} — O(m)
3. Sort the m ordered pairs (j, α^{mj}) with respect to their second coordinates, obtaining a list L_1 — O(mlog m)
4. **for** $i \leftarrow 0$ **to** $m - 1$
   do compute $\beta\alpha^{-i}$ — O(m)
5. Sort the m ordered pairs $(i, \beta\alpha^{-i})$ with respect to their second coordinates, obtaining a list L_2 — O(mlog m)
6. Find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
7. $\log_\alpha \beta \leftarrow (mj + i) \bmod n$

O(log m)

O(mlogm) ~ O(m) = O($p^{1/2}$)

CR

Other Discrete Log Algorithms

$$\beta \equiv \alpha^{a} \bmod n$$

- Pollard-Hellman Algorithm

  used when n is a composite

- Pollard-Rho Algorithm

  about the same runtime as the Shank's algorithm, but has much less memory requirements

Diffie Hellman Problem

$Let\,(G,\cdot)\,be\,a\,group$

$Let\,\alpha \in G$ be a primitive element in the group with order n

Define the set

$$\langle \alpha \rangle = \{\alpha^i : 0 \le i \le n-1\}$$

$given\ \alpha^a\ and\ \alpha^b,\ find\ \alpha^{ab}$ Computational DH (CDH)

$given\ \alpha^a, \alpha^b\ and\ \alpha^c,$ determine if $c \equiv ab \bmod n$ Decision DH (DDH)

CR

Recall…

Diffie Hellman Key Exchange

Alice and Bob agree upon a prime **p** and a generator **g**.
This is public information

choose a secret **a**
compute $\mathbf{A = g^a \bmod p}$

choose a secret **b**
compute $\mathbf{B = g^b \bmod p}$

B

A

Compute K $\mathbf{= B^a \bmod p}$

Compute K $\mathbf{= A^b \bmod p}$

$$\mathbf{A^b \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p = B^a \bmod p}$$