RSA Cryptosystem

密碼學與應用

海洋大學資訊工程系
丁培毅

Naïve Public Key System

- Encryption and decryption algorithm are not the same
- Public/private key pair: private key is related to public key but can not be easily derived from public key
- Illustrating example:

$$m \in Z_{11}^{*}$$

$$m * 1 = m \pmod{11}$$

$$\underbrace{\underbrace{m * 8}_{\text{encryption}} * 8^{-1}}_{\text{decryption}} = m \pmod{11}$$

(where $8 * 8^{-1} = 1$)

8 is the public key
m * 8 is the ciphertext
$\mathbf{8^{-1}}$ is the private key (if nobody can derive this from the public key, then this system is secure)

Knapsack (Subset Sum) PKC

- Merkel and Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," IT-24, 1978
  - a good application of an NP problem on designing public key cryptosystem; no longer secure
- **Super-increasing sequence:**

  $\{a_1, a_2, \ldots a_n\}$ such that $a_i > \sum_{j=0}^{i-1} a_j$ ex. 1, 3, 5, 10, 20, 40

- **Note:** 1. Given a number c, finding a subset $\{a_j\}$ s.t. $c = \sum_j a_j$ is an easy problem, ex. $48 = 40 + 5 + 3$

  2. Every subset sum $\sum_{j \in S} a_j < 2 \cdot a_M$ where $a_M = \max_{j \in S}\{a_j\}$

  3. Every possible subset sum is unique

  pf: given x, assume $x = \sum_{j \in S} a_j = \sum_{j \in T} a_j$, where $S \neq T$, assume $\max_{j \in S}\{a_j\} \neq \max_{j \in T}\{a_j\}$ ….

Knapsack (Subset Sum) PKC

- choose a number b in Z_p^*, ex. $p = 101$, $b = 23$, and convert the super-increasing sequence to a **normal knapsack sequence** B=$\{b_1, b_2, \ldots, b_n\}$ where

  $$b_i \equiv a_i \cdot b \pmod{p}$$

  ex. 23, 69, 14, 28, 56, 11
- Since gcd(b, p)=1, this conversion is invertible, i.e.

  $$a_i \equiv b_i \cdot b^{-1} \pmod{p}$$

  ex. $b^{-1} \equiv 22 \pmod{101}$ $(b \cdot b^{-1} \equiv 1 \pmod{p})$
- **G**iven a number d, finding a subset $\{b_j\} \subseteq$B s.t.

  $$d = \sum_j b_j \pmod{p}$$

  is an NP-complete problem, ex. 94 = 11 + 14 + 69

Knapsack (Subset Sum) PKC

- Encryption:
  - **public key**: normal knapsack seq. {23, 69, 14, 28, 56, 11}
  - message m, $0 \le m < 2^6$, ex. $(60)_{10} = (111100)_2$
  - sum up the corresponding elements of ‘1’ bits, i.e.

    $23 + 69 + 14 + 28 = 134$ is the ciphertext
- Decryption:
  - **private key**: $b^{-1}=22$, $p=101$, {1, 3, 5, 10, 20, 40}
  - calculate $134 * 22 \bmod 101 = 19$
  - use the corresponding super-increasing knapsack seq. {1, 3, 5, 10, 20, 40} to decrypt as follows:
    - $19 < 40$, mark a ‘0’
    - $19 < 20$, mark a ‘0’
    - $19 \ge 10$, mark a ‘1’ and subtract 10 from 19
    - $9 \ge 5$, mark a ‘1’ and subtract 5 from 9
    - $4 \ge 3$, mark a ‘1’ and subtract 3 from 4
    - $1 \ge 1$, mark a ‘1’ and subtract 1 from 1
  - recovered message is $(111100)_2 = (60)_{10}$

Knapsack (Subset Sum) PKC

- Why does it work?

let the plaintext be $(111100)_2$

ciphertext $c = b_1 + b_2 + b_3 + b_4$

$$\equiv a_1 b + a_2 b + a_3 b + a_4 b \pmod{p}$$

decryption: $c\, b^{-1} \pmod{p} \equiv a_1 + a_2 + a_3 + a_4 \pmod{p}$

is a subset sum problem of a super-increasing sequence

RSA and Rabin

- In the following, we discuss two important cryptosystems based on the difficulty of **integer factoring** (an NP problem)

$n = p \cdot q$

- RSA's underlying problem

  Solving e-th root modulo n is difficult

RSA function

$$y \equiv x^e \pmod{n}$$

- Rabin's underlying problem

  Solving square root modulo n is difficult

$$y \equiv x^2 \pmod{n}$$

Rabin function

both functions are candidates for trapdoor one way function

RSA and Rabin Function

- ✧ Solving e-th root of y modulo n is difficult!!!

  $y \equiv x^e \pmod{n}$, where $\gcd(e, \phi(n)) = 1$

  Why don't we take (e^{-1})-th power of y?

  where $e^{-1} \cdot e \equiv 1 \pmod{\phi(n)}$

  ex. $n = 11 \cdot 13 = 143$, $e = 7$

  $\phi(n) = 10 \cdot 12 = 120$, $e^{-1} = 103$

  Trouble: How do we know $\phi(n)$?

- ✧ Solving square root of y modulo n is difficult!!!

  $y \equiv x^2 \pmod{n}$

  Why don't we take (2^{-1})-th power of y?

  where $2^{-1} \cdot 2 \equiv 1 \pmod{\phi(n)}$

  ex. $n = 11 \cdot 13 = 143$

  $\phi(n) = 10 \cdot 12 = 120$, $\gcd(2, \phi(n)) = 2$

  Remember solving square root of y modulo a prime number p is very easy

  Trouble: $d \cdot 2 \equiv 1 \pmod{\phi(n)}$ has no solution for d

RSA Public Key Cryptosystem

- R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM, pp.120-126, 1978
- Based on the *Integer Factorization* problem
- Choose two large prime numbers: p, q (keep them secret!!)
- Calculate the modulus $n = p \cdot q$ (make it public)
- Calculate $\Phi(n) = (p-1) \cdot (q-1)$ (keep it secret)
- Select a random integer such that $e < \Phi$ and $\gcd(e, \Phi) = 1$
- Calculate the unique integer d such that $e \cdot d \equiv 1 \pmod{\Phi}$
- **Public key: (n, e)** **Private key: d**

RSA Encryption & Decryption

- ✧ Alice wants to encrypt a message ***m*** for Bob
- ✧ Alice obtains Bob's authentic public key (n, e)
- ✧ Alice represents the message as an integer m in the interval $[0, n-1]$
- ✧ Alice computes the modular exponentiation

$$c \equiv m^e \pmod{n}$$

- ✧ Alice sends the ciphertext c to Bob
- ✧ Bob decrypts c with his private key (n, d) by computing the modular exponentiation

$$\hat{m} \equiv c^d \pmod{n}$$

RSA Encryption & Decryption

✧ Why does RSA work? (simpler but incomplete proof)

- **Fact 1:** $e \cdot d \equiv 1 \pmod{\Phi} \Rightarrow e \cdot d = 1 + k\,\Phi$
- **Fact 2:** $\forall m$, $\gcd(m,n)=1$, $m^{\Phi} \equiv 1 \pmod{n}$
  (by Euler's theorem)
- **From Fact 2**: $\forall m$, $\gcd(m,n)=1$,

$$c^{d} \equiv m^{ed} \equiv m^{1+k\,\Phi} \equiv m^{1+k\,(p\text{-}1)(q\text{-}1)} \equiv m \pmod{\mathrm{n}}$$

note: 1. This only proves that for all m that are not multiples of p or q can be recovered after RSA encryption and decryption.

2. For those m that are multiples of p or q, the Euler's theorem simply does not hold because $p^{\Phi} \equiv 0 \pmod{p}$ and
$p^{\Phi} \equiv 1 \pmod{q}$
which means that p^{Φ} ❄ $1 \pmod{n}$ from CRT.

RSA Encryption & Decryption

- Why does RSA work?
  - **Fact 1:** $e \cdot d \equiv 1 \pmod{\Phi} \Rightarrow e \cdot d = 1 + k\,\Phi$
  - **Fact 2:** $\forall m$, $\gcd(m,p)=1$, $m^{p-1} \equiv 1 \pmod{p}$ (by Fermat's Little theorem)
  - **From Fact 2:** $\forall m$, $\gcd(m,p)=1$

    $m^{1+k(p-1)(q-1)} \equiv m \pmod{p}$

    note: this equation is trivially true when m = kp
  - **From Fact 2:** $\forall m$, $\gcd(m,q)=1$

    $m^{1+k(p-1)(q-1)} \equiv m \pmod{q}$

    note: this equation is trivially true when m = kq
  - **From CRT**: $\forall m$,

    $c^d \equiv m^{ed} \equiv m^{1+k\,\Phi} \equiv m^{1+k(p-1)(q-1)} \equiv m \pmod{n}$

RSA Function is a Permutation

- RSA function is a permutation: (1-1 and onto, bijective)
- Goal: "$\forall x_1, x_2 \in Z_n$ if $x_1^e \equiv x_2^e \pmod n$ then $x_1 = x_2$"
  - $\forall x \neq r \cdot p,\ x^{p-1} \equiv 1 \pmod p$, $\forall x \neq s \cdot q,\ x^{q-1} \equiv 1 \pmod q$
    $\Rightarrow \forall k, \forall x \neq r \cdot p,\ x^{k\phi(n)} \equiv 1 \pmod p$, $\forall k, \forall x \neq s \cdot q,\ x^{k\phi(n)} \equiv 1 \pmod q$
    $\Rightarrow \forall k, \forall x,\ x^{k\phi(n)+1} \equiv x \pmod p$, $\forall k, \forall x,\ x^{k\phi(n)+1} \equiv x \pmod q$
    CRT $\Rightarrow \forall k, \forall x,\ x^{k\phi(n)+1} \equiv x \pmod n$
  - $\gcd(e,\phi(n))=1$ $\Rightarrow$ inverse of e (mod $\phi(n)$) exists
    $\Rightarrow$ d is the inverse s.t. $e \cdot d \equiv 1 \pmod{\phi(n)}$
  - $\forall x_1, x_2 \in Z_n$ if $x_1^e \equiv x_2^e \pmod n$

$\Rightarrow (x_1^e)^d \equiv (x_2^e)^d \pmod n$

$\Rightarrow (x_1)^{1+k\,\phi(n)} \equiv (x_2)^{1+k\,\phi(n)} \pmod n$

$\Rightarrow x_1 \equiv x_2 \pmod n$

Note: Euler Thm is valid only when $x \in Z_n^*$

RSA Cryptosystem

- Most popular PKC in practice
- Tens of dedicated crypto-processors are specifically designed to perform modular multiplication in a very efficient way.
- **Disadvantage:** long key length,
  complex key generation scheme,
  deterministic encryption
- For acceptable level of security in commercial applications, 1024-bit (300 digits) keys are used. For a symmetric key system with comparable security, about 100 bits keys are used.
- In constrained devices such as smart cards, cellular phones and PDAs, it is hard to store, communicate keys or handle operations involving large integers

Matlab examples

- rsatest.m
  - maple('p := nextprime(1897345789)')
  - maple('q := nextprime(278478934897)')
  - maple('n := p*q');
  - maple('x := 101');
  - maple('e := nextprime(12345678)')
  - maple('d := e&^(-1) mod ((p-1)*(q-1))')
  - maple('y := x&^(e) mod n')
  - maple('xp := y&^(d) mod n')

Very likely to be relatively prime with (p-1)(q-1)

extended Euclidean algo.

Rabin Cryptosystem (1/3)

- M.O. Rabin, "Digitalized Signatures and Public-key Functions As Intractable As Factorization", Tech. Rep. LCS/TR212, MIT, 1979

- Choose two large prime numbers: p, q (keep them secret!!)
- Calculate the modulus $n = p \cdot q$ (make it public)
- **Public Key** n
- **Private Key** p, q

Rabin Cryptosystem (2/3)

- Alice want to encrypt a message ***m*** (with some fixed format) for Bob
- Alice obtains Bob's authentic public key n
- Alice represents the message as an integer m in the interval $[0, n-1]$
- Alice computes the modular square

$$c \equiv m^2 \pmod{n}$$

- Alice sends the ciphertext c to Bob
- Bob decrypts c using his private key p and q
- Bob computes the four square roots $\pm m_1$, $\pm m_2$ using CRT, one of them satisfying the fixed message format is the recovered message

Rabin Cryptosystem (3/3)

- The range of the Rabin function is not the whole set of Z_n^* (compare with RSA).
  - The range covers all the quadratic residues. (for a prime modulus, the number of quadratic residues in Z_p^* is $(p-1)/2$; for a composite integer $n=p \cdot q$, the number of quadratic residues in Z_n^* is $(p-1)(q-1)/4$)
  - In order to let the Rabin function have inverse, it is necessary to make the Rabin function a permutation, ie. 1-1 and onto. Therefore, the number of elements in the domain of the Rabin function should also be $(p-1)(q-1)/4$ for $n=p \cdot q$. There are 4 possible numbers with their square equal to y, and we have to make 3 of them illegal.

Number of Quadratic Residues

- For a prime modulus p: number of QR_p's in Z_p^* is (p-1)/2

  pf: find a primitive g, at least $\{g^2, g^4, \ldots g^{p-1}\}$ are QR_p's
  assume there are (p+1)/2 QRs,
  since <u>there are exactly two square roots of a QR modulo p</u>
  there are p+1 square roots for these (p+1)/2 QRs, i.e. there must be at least two pairs of square roots are the same (pigeon-hole), i.e. two out of these (p+1)/2 QRs are the same, contradiction

- For a composite modulus p·q: number of QR_n's in $Z_{p\cdot q}^*$ is (p-1)(q-1)/4

  pf: find a common primitive in Z_p^* and Z_q^* g, at least $\{g^2, g^4, \ldots, g^{p-1} \ldots, g^{q-1} \ldots, g^{\lambda(n)}\}$ are QR_n's, where $\lambda(n) = \text{lcm}(p-1,q-1)$ can be as large as (p-1)(q-1)/2, this set has (p-1)(q-1)/4 distinct elements
  assume there are (p-1)(q-1)/4+1 QR_n's in Z_n^*, since there are exactly four square roots of a QR modulo p·q, these QR_n's have (p-1)(q-1)+4 square roots in total, which include repeated elements, therefore, there are at most (p-1)(q-1)/4 QR_n's in Z_n^*

Matlab examples

```
maple('p:= nextprime(189734535789)')       % 189734535811 = 4 k + 3
maple('p mod 4')
maple('q:= nextprime(27847815934897)')  % 27847815934931 = 4 k + 3
maple('q mod 4')
maple('n:=p*q');
maple('x:=07041111142214171103000O')   % text2int('helloworld')
maple('c:= x&^2 mod n')
```

```
maple('c1:= c mod p')
maple('r1:= c1&^((p+1)/4) mod p')            % maple('r1&^2 mod p')

maple('c2:= c mod q')
maple('r2:= c2&^((q+1)/4) mod q')            % maple('r2&^2 mod q')


maple('m1:= chrem([r1, r2], [p, q])')    % 3704440302544264662351219
maple('m2:= chrem([-r1, r2], [p, q])')   % 70411111422141711030000
maple('m3:= chrem([r1, -r2], [p, q])')   % 5213281318342160554284041
maple('m4:= chrem([-r1, -r2], [p, q])')  % 1579252127220037602962822
```

Security of the RSA Function

- **Break RSA** means 'inverting RSA function without knowing the trapdoor'

  $y \equiv x^e \pmod{n}$

- Factor the modulus $\Rightarrow$ Break RSA
  - If we can factor the modulus, we can break RSA
  - If we can break RSA, we don't know whether we can factor the modulus…**open problem** (with negative evidences)
- Factor the modulus $\Leftrightarrow$ Calculate private key d
  - If we can factor the modulus, we can calculate the private exponent d (the trapdoor information).
  - If we have the private exponent d, we can factor the modulus.

  will be illustrated later after factorization

Security of Rabin Function

- ✧ Security of Rabin function *is equivalent to* integer factoring
- ✧ inverting '$y \equiv f(x) \equiv x^2 \pmod{n}$' without knowing p and q $\Leftrightarrow$ factoring n
  - ✶ $\Leftarrow$
    - if you can factor $n = p \cdot q$ in polynomial time
    - you can solve $y \equiv x_1^2 \pmod{p}$ and $y \equiv x_2^2 \pmod{q}$ easily
    - using CRT you can find x which is $f^{-1}(y)$
  - ✶ $\Rightarrow$
    - given a quadratic residue y if you can find the four square roots $\pm x_1$ and $\pm x_2$ for y in polynomial time
    - you can factor n by trying $\gcd(x_1 - x_2, n)$ and $\gcd(x_1 + x_2, n)$

Basic Factoring Principle (1/4)

- Let n be an integer and suppose there exist integers x and y with $x^2 \equiv y^2 \pmod{n}$, but $x \not\equiv \pm y \pmod{n}$. Then ❶ n is composite, ❷ both gcd(x-y, n) and gcd(x+y, n) are nontrivial factors of n.

Proof:

let d = gcd(x-y, n).

Case 1: assume $d = n \Rightarrow x \equiv y \pmod{n}$ contradiction

Case 2: assume d is 1 (the trivial factor)

$x^2 \equiv y^2 \pmod{n} \Rightarrow x^2 - y^2 = (x-y)(x+y) = k \cdot n$

d=1 means gcd(x-y, n)=1 $\Rightarrow$

$n \mid x+y \Rightarrow x \equiv -y \pmod{n}$ contradiction

Case 1 and 2 implies that $1 < d < n$

i.e. d must be a nontrivial factor of n

Basic Factoring Principle (2/4)

- $x^2 \equiv y^2 \pmod{p}$ implies $x \equiv \pm y \pmod{p}$ since $p \mid (x+y)(x-y)$ implies $p \mid (x+y)$ or $p \mid (x-y)$,
  i.e. $x \equiv -y \pmod{p}$ or $x \equiv y \pmod{p}$
- $x^2 \equiv y^2 \pmod{n}$
  $pq \mid (x+y)(x-y)$ implies the following 4 possibilities
  1. $pq \mid (x+y)$ i.e. $x \equiv -y \pmod{n}$
  2. $pq \mid (x-y)$ i.e. $x \equiv y \pmod{n}$
  3. $p \mid (x+y)$ and $q \mid (x-y)$ i.e. $x \equiv -y \pmod{p}$ and $x \equiv y \pmod{q}$
  4. $q \mid (x+y)$ and $p \mid (x-y)$ i.e. $x \equiv -y \pmod{q}$ and $x \equiv y \pmod{p}$
  - Case 1 and case 2 are useless for factorization
  - Case 3 leads to the factorization of n, i.e. $\gcd(x+y, n) = p$ and $\gcd(x-y, n) = q$
  - Case 4 leads to the factorization of n, i.e. $\gcd(x+y, n) = q$ and $\gcd(x-y, n) = p$

Basic Factoring Principle (3/4)

- This principle is used in *almost all factoring algorithms*.
- Why is it working?
  - take $n = p \cdot q$ (p and q are prime) for example
  - $x^2 \equiv y^2 \pmod{n}$ implies $x^2 \equiv y^2 \pmod{p}$ and $x^2 \equiv y^2 \pmod{q}$
  - we know '$x \equiv \pm y \pmod{p}$ are the only solution to $x^2 \equiv y^2 \pmod{p}$' and '$x \equiv \pm y \pmod{q}$ are the only solution to $x^2 \equiv y^2 \pmod{q}$'
  - therefore, from CRT we know $x^2 \equiv y^2 \pmod{n}$ has four solutions,
    - $x \equiv y \pmod{p}$ and $x \equiv y \pmod{q}$ $\Rightarrow$ $x \equiv y \pmod{n}$
    - $x \equiv -y \pmod{p}$ and $x \equiv -y \pmod{q}$ $\Rightarrow$ $x \equiv -y \pmod{n}$
    - $x \equiv y \pmod{p}$ and $x \equiv -y \pmod{q}$ $\Rightarrow$ $x \equiv z \pmod{n}$
    - $x \equiv -y \pmod{p}$ and $x \equiv y \pmod{q}$ $\Rightarrow$ $x \equiv -z \pmod{n}$
  - as long as we have z (where $z \not\equiv \pm y$), we can factor n into $\gcd(y-z, n)$ and $\gcd(y+z, n)$

Basic Factoring Principle (4/4)

- Ex: Consider the roots of 4 (mod 35), i.e. solving x from $x^2 \equiv 4 \pmod{35}$
  - try to take square root of both sides, we find $x = \pm 2$ or ± 12
  - i.e. $12^2 \equiv 2^2 \pmod{35}$, but $12 \neq \pm 2 \pmod{35}$
  - therefore 35 is composite
  - gcd(12-2, 35) = 5 is a nontrivial factor of 35
  - gcd(12+2, 35) = 7 is a nontrivial factor of 35

Miller-Rabin Test

Is ***n*** a composite number?

- Let $n > 1$ be odd, write $n\text{-}1 = 2^k \cdot m$ with m being odd
- Choose a random integer $\boldsymbol{a}$ with $1 < \boldsymbol{a} < n\text{-}1$
- Compute $b_0 \equiv \boldsymbol{a}^m \pmod{n}$
  if $b_0 \equiv \pm 1 \pmod{n}$, stop, n is probably prime
- Compute $b_1 \equiv b_0^{\,2} \pmod{n}$
  if $b_1 \equiv 1 \pmod{n}$, stop, $\gcd(b_0\text{-}1, n)$ is a factor of n
  if $b_1 \equiv \text{-}1 \pmod{n}$, stop, n is probably prime
- Compute $b_2 \equiv b_1^{\,2} \pmod{n}$
  ……..
- Compute $b_{k-1} \equiv b_{k-2}^{\,2} \pmod{n}$
  if $b_{k-1} \equiv 1 \pmod{n}$, stop, $\gcd(b_{k-2}\text{-}1, n)$ is a factor of n
  if $b_{k-1} \equiv \text{-}1 \pmod{n}$, stop, n is probably prime
- Compute $b_k \equiv b_{k-1}^{\,2} \pmod{n}$
  if $b_k \equiv 1 \pmod{n}$, stop, $\gcd(b_{k-1}\text{-}1, n)$ is a factor of n
  otherwise n is composite (Fermat Little Thm, $b_k \equiv a^{n-1} \pmod{n}$)

n will pass Fermat test
n is called pseudo prime with respect to base a

…

Miller-Rabin Test Illustrated

$b_0 \equiv a^m \pmod{n}$

$b_1 \equiv a^{2 \cdot m} \pmod{n}$

…

$b_k \equiv a^{2^k \cdot m} \equiv a^{n-1} \pmod{n}$

$n-1 = 2^k \cdot m$

Consider 4 possible cases:

① $b_0 \equiv \pm 1 \pmod{n}$
all $b_i \equiv 1 \pmod{n}$, $i=1,2,\ldots k$
there is no chance to use
Basic Factoring Principle, **abort**

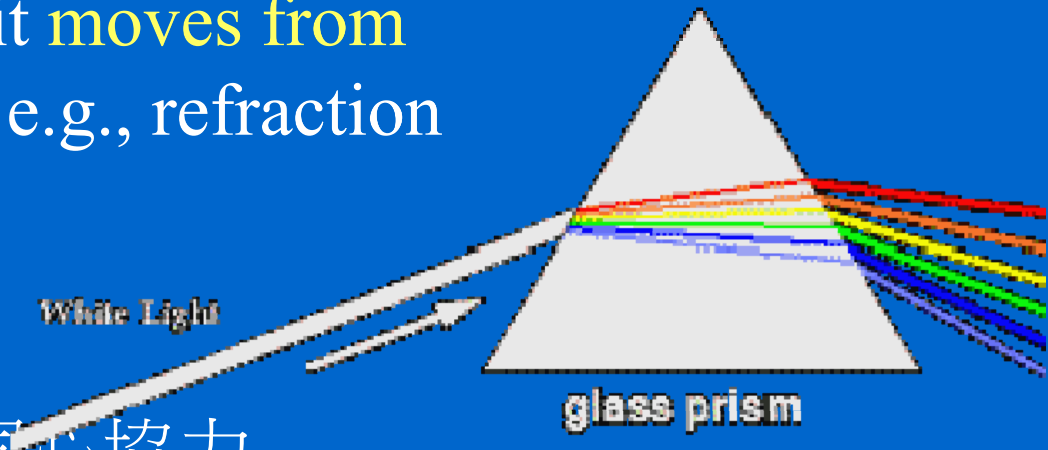
② ① is not true,
$b_{i-1} \not\equiv \pm 1 \pmod{n}$ and
$b_i \equiv 1 \pmod{n}$, $i=1,2,\ldots k$
Basic Factoring Principle applied, **composite**

③ ① and ② are not true,
$b_i \equiv -1 \pmod{n}$, $i=1,2,\ldots k$
all subsequent $b_j \equiv 1 \pmod{n}$,
there is no chance to use
Basic Factoring Principle, **abort**

④ ①, ②, and ③ are not true,
$b_k \equiv a^{n-1} \pmod{n}$
if n is prime, $b_k \equiv 1 \pmod{n}$
i.e. if $b_k \not\equiv 1 \pmod{n}$ n is **composite**
($b_k \equiv 1 \pmod{n}$ is covered by ②)

Uncoordinated Behaviors

- ✧ Light changes speed as it moves from one medium to another, e.g., refraction caused by a prism



- ✧ 趣味競賽: 兩人三腳, 同心協力, …
- ✧ Squaring a number modulo different prime numbers

	2^2	2^3	2^4	2^5	2^6	2^7	2^8
mod 11	4	8	**5**	**10**	**9**	**7**	**3**
mod 13	4	8	**3**	**6**	**12**	**11**	**9**

When/How does Basic Factoring Principle work in M-R test?

- When:
  - explicitly: $b_{i-1} \neq \pm 1 \pmod{n}$ and $b_i \equiv b_{i-1}^2 \equiv 1 \pmod{n}$

If n is not prime, not often when $b^k \equiv a^{n-1} \pmod{n}$ but often when $b^k \equiv a^{r\phi(n)} \pmod{n}$ in universal exponent factoring

- How:
  - implicitly: let p | n and q | n (p, q be two factors of n)

    $$b_{i-1}^2 \equiv 1 \pmod{p} \text{ and } b_{i-1}^2 \equiv 1 \pmod{q}$$

    but either $b_{i-1} \not\equiv 1 \pmod{p}$ or $b_{i-1} \not\equiv 1 \pmod{q}$
  - catching the moment that $b_0, b_1, \ldots$ behave differently while taking square in (mod p) component and (mod q) components

Miller-Rabin Test Example

- Ex. n = 561 ← A Carmichael number: pass the Fermat test for all bases

  $n-1 = 560 = 16 \cdot 35 = 2^4 \cdot 35$

  let a = 2

  $b_0 \equiv 2^{35} \equiv 263 \pmod{561}$

  $b_1 \equiv b_0^2 \equiv 2^{2\cdot 35} \equiv 166 \pmod{561}$

  $b_2 \equiv b_1^2 \equiv 2^{2^2\cdot 35} \equiv 67 \pmod{561}$

  $b_3 \equiv b_2^2 \equiv 2^{2^3\cdot 35} \equiv 1 \pmod{561}$

  561 is composite $(3\cdot 11\cdot 17)$,

  $\gcd(b_2-1, 561) = 33$ is a factor

  Note: $3-1=2$, $11-1=2\cdot 5$, $17-1=2^4$

  $\phi(561) = 561(1-1/3)(1-1/11)(1-1/17)=2\cdot 10\cdot 16$

  $\phi(561) \mid n-1$ for this special case

mod	3	11	17
	2	10	8
	1	1	13
	1	1	16
	1	1	1

$ord_{17}(2)=2^3$

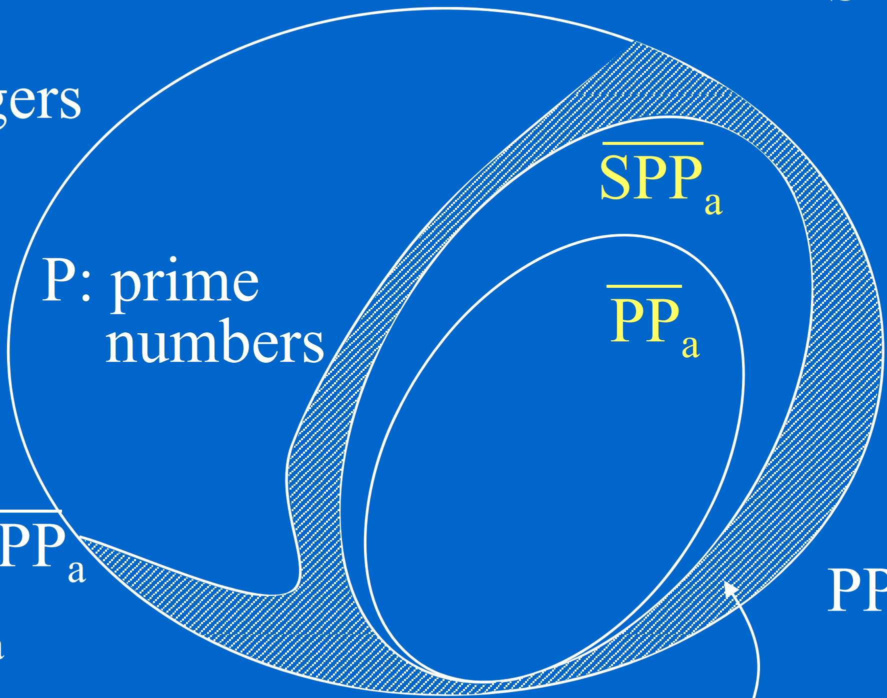
Pseudo Prime and Strong Pseudo Prime

- If n is not a prime but satisfies $a^{n-1} \equiv 1 \pmod{n}$ we say that n is a pseudo prime number for base a.
  - Ex. $2^{560} \equiv 1 \pmod{561}$
- If n is not a prime but passes the Miller-Rabin test with base a (without being identified as a composite), we say that n is a strong pseudo prime number for base a.
- Up to 10^{10}, there are 455052511 primes, there are 14884 pseudo prime numbers for the base 2, and 3291 strong pseudo prime numbers for the base 2

Fermat and Miller-Rabin Test

- Both of these two tests are for identifying subsets of composite numbers

I: integers



SPP_a: strong pseudo prime numbers for base a, the set of composite n where M-T test says 'probably prime'

C: composite numbers

$I = P \cup C$

$C = SPP_a \cup \overline{SPP_a}$
$= PP_a \cup \overline{PP_a}$

PP_a: pseudo prime numbers for base a, the set of composite n where $a^{n-1} \equiv 1 \pmod{n}$

$SPP_a \subset PP_a$

$\overline{PP_a} \subset \overline{SPP_a} \subset C$

(hatched area): mysterious part
not prime, but cannot be identified as composite

Composite Witness

- Note that the M-R test and probably together with the Lucas test leave the strong pseudo prime number *an extremely small set*.
- In other words, these tests are very close to a *real 'primality test'* between prime numbers and composite numbers.
- If you have an RSA modulus n=p·q, you certainly can test it and find out that it is actually a composite number.
- However, these tests do not necessarily give you the factors of n in order to tell you that n is a composite number. The factors of n, i.e. p or q, are certainly a kind of witness about the fact that n is composite.
- However, there are other kind of witness that n is composite, e.g., "2^{n-1} (mod n) does not equal to 1" is also a witness that n is composite.
- A composite number will be factored out by the M-R test only if it is a pseudo prime but it is not a strong pseudo prime number.

Matlab Example

- primetest(n)
  - Miller-Rabin test for 30 randomly chosen base ***a***
  - output 0 if n is composite
  - output 1 if n is prime
  - Matlab program can not be used for large n
  - use Maple isprime(n), one strong pseudo-primality test and one Lucas test
- primetest(2563)
  ans= 0
- factor(2563)
  ans = 11 233

Questions

- What is the probability that Miller-Rabin test fails???
  - If n is a prime number, it will not be recognized as a composite number
  - If $n = p \cdot q$, but
    $b_k \equiv a^{n-1} \equiv 1 \pmod{n}$ meets Fermat test (pseudo prime number)
    $0 < i \leq k$ $b_i \equiv 1 \pmod{n}$ and $b_{i-1} \equiv -1 \pmod{n}$
    meets Miller-Rabin test (strong pseudo prime number)
    $$\left(\begin{array}{l} \text{or } b_i \equiv 1 \pmod{n} \equiv 1 \pmod{p} \equiv 1 \pmod{q} \\ \quad b_{i-1} \equiv -1 \pmod{n} \equiv -1 \pmod{p} \equiv -1 \pmod{q} \end{array} \right)$$
  - Note: $a^{pq-1} \equiv 1 \pmod{n}$
    $a^{(p-1)(q-1)} \equiv 1 \pmod{n}$
    $a^{\mathrm{lcm}(p-1,\, q-1)} \equiv 1 \pmod{n}$

Note on Primality Testing

- Primality testing is *different* from factoring
  - Kind of interesting that we can tell something is composite without being able to actually factor it
- Recent result (2002) from IIT trio (Agrawal, Kayal, and Saxena)
  - Recently it was shown that deterministic primality testing could be done in polynomial time
    - Complexity was like $O(n^{12})$, though it's been slightly reduced since then
  - Does this meant that RSA was broken?
- Randomized algorithms like Rabin-Miller are far more efficient than the IIT algorithm, so we'll keep using those

Finding a Random Prime

- Find a prime of around 100 digits for cryptographic usage
- Prime number theorem ($\pi(x) \approx x/\ln(x)$) asserts that the density of primes around x is approximately $1/\ln(x)$
- $x = 10^{100}$, $1/\ln(10^{100}) = 1/230$
  if we skip even numbers, the density is about 1/115
- pick a random starting point, throw out multiples of 2, 3, 5, 7, and use Miller-Rabin test to eliminate most of the composites.

- maple('a:=nextprime(189734535789)')

Factoring

- General number field sieve (GNFS): fastest

$$e^{(1.923+O(1))(\ln(n))^{1/3}(\ln(\ln(n)))^{2/3}}$$

- Quadratic sieve (QS)
- Elliptic curve method (ECM), Lenstra (1985)
- Pollard's Monte Carlo algorithm
- Continued fraction algorithm
- Trial division, Fermat factorization
- Pollard's p-1 factoring (1974), Williams's p+1 factoring (1982)
- Universal exponent factorization, exponent factorization

Simple Factoring Methods

- Trial division:
  - dividing an integer n by all primes $p \leq \sqrt{n}$ ... too slow
- Fermat factorization:
  - ex. n = 295927 calculate $n+1^2$, $n+2^2$, $n+3^2$… until finding a square, i.e. $x^2 = n + y^2$, therefore, $n = (x+y)(x-y)$ … if $n = p \cdot q$, it takes on average $|p-q|/2$ steps … too slow

    assume p>q, $n+y^2 = p\cdot q+((p-q)/2)^2=(p^2+2pq+q^2)/4=((p+q)/2)^2$
  - in RSA or Rabin, avoid p, q with the same bit length
- By-product of Miller-Rabin primality test:
  - if n is a pseudoprime and not a strong pseudoprime, Miller-Rabin test can factor it. about 10^{-6} chance

Universal Exponent Factorization

- if we have an exponent r, s.t. $a^r \equiv 1 \pmod{n}$ for all a gcd(a,n)=1
- write $r = 2^k \cdot m$ with m odd
- choose a random a, $1<a<n-1$
- if $\gcd(a, n) \neq 1$, we have a factor
- else
  - let $b_0 \equiv a^m \pmod{n}$, if $b_0 \equiv \pm 1$ stop, choose another a
  - compute $b_{u+1} \equiv b_u^2 \pmod{n}$ for $0 \leq u \leq k-1$,
  - if $b_{u+1} \equiv -1$, stop, choose another a
  - if $b_{u+1} \equiv 1$ then $\gcd(b_u - 1, n)$ is a factor (basic factoring principle)

r must be even since we can take a=-1 $(-1)^r \equiv 1 \pmod{n}$ requires r being even

$a \equiv \pm 1$ do not work

- Question: How do we find a universal exponent ***r*** ??? Hard
- Note: if know ϕ(n), then any r = k ϕ(n) will do, however, knowing factors of n is a prerequisite of know ϕ(n)
- Note: For RSA, if the private exponent d is recovered, then

  $\phi(n) \mid d \cdot e - 1$, $d \cdot e - 1$ is a universal exponent

Universal Exponent Factorization

- Ex.

  n=211463707796206571; e=9007; d=116402471153538991

  r=e*d-1=1048437057679925691936; powermod(2,r,n)=1

  let $r=2^5$*r1; r1=32763658052497677873

  powermod(2,r1,n)=187568564780117371≠±1

  powermod(2,2*r1,n)=113493629663725812≠±1

  powermod(2,4*r1,n)=1 => gcd(2*r1-1,n)=885320963 is a factor

- Note: $n = 211463707796206571 = 238855417 \cdot 885320963$

  $238855417 - 1 = 2^3 \cdot 3 \cdot 73 \cdot 136333 = 2^{k_1} \cdot p_1$

  $885320963 - 1 = 2 \cdot 2069 \cdot 213949 = 2^{k_2} \cdot q_1$

  This method works only when k_1 does not equal k_2.

- Exponent factorization even if *r* is valid for one *a*, you can still try the above procedure

p-1 factoring (1/2)

- If one of the prime factors of n has a special property, it is sometimes easier to factor n.
  - ex. if $p-1$ has only small prime factors
  - Pollard 1974
- Algorithm
  - Choose an integer $a > 1$ (often $a = 2$ is used)
  - Choose a bound B
  - Compute $b \equiv a^{B!}$ as follows:
    - $b_1 \equiv a \pmod{n}$ and $b_j \equiv b_{j-1}^{\,j} \pmod{n}$ then $b \equiv b_B \pmod{n}$
  - Let $d = \gcd(b-1, n)$, if $1 < d < n$, we have found a factor of n

have a chance of being larger than all the prime factors of p-1

If B is larger than all the prime factors of $p-1$ $\overset{\text{(very likely)}}{\Longrightarrow}$ $p-1|B!$
therefore $b \equiv a^{B!} \equiv (a^{p-1})^k \equiv 1 \pmod{p}$, i.e. $p|b-1$

Fermat Little's Thm

If $n=p\cdot q$, $p-1$ and $q-1$ both have small factors that are less than B, then $\gcd(b-1,n)=n$, (useless) however, $b \equiv a^{B!} \equiv 1 \pmod{n}$ and we can use the Universal exponent method

p-1 factoring (2/2)

- How do we choose B?
  - small B will be faster but fails often
  - large B will be very slow
- In RSA, Rabin, Paillier, or other systems based on integer factoring, usually $n=p\cdot q$, we should ensure that p-1 has at least one large prime factor.
  - How do we do this?

    ex. we want to choose p around 100 digits
    - choose a prime number p_0 around 40 digits
    - look at integer $k\cdot p_0+1$ with k around 60 digits and do primality test
- Generalization:

  Elliptic curve factorization method, Lenstra, 1985
- Best records: p-1: 34 digits (113 bits), ECM: 47 digits (143 bits)

Quadratic Sieve (1/4)

- Example: factor $n = 3837523$
  - form the following relations

    individual factors are small

    $$9398^2 \equiv 5^5 \cdot 19 \pmod{3837523}$$
    $$19095^2 \equiv 2^2 \cdot 5 \cdot 11 \cdot 13 \cdot 19 \pmod{3837523}$$
    $$1964^2 \equiv 3^2 \cdot 13^3 \pmod{3837523}$$
    $$17078^2 \equiv 2^6 \cdot 3^2 \cdot 11 \pmod{3837523}$$

    make the number of each factors even

  - multiply the above relations

    $$(9398 \cdot 19095 \cdot 1964 \cdot 17078)^2 \equiv (2^4 \cdot 3^2 \cdot 5^3 \cdot 11 \cdot 13^2 \cdot 19)^2$$
    $$2230387^2 \equiv 2586705^2$$

    hoping they are not equal

  - since $2230387 \neq \pm 2586705 \pmod{3837523}$
  - $\gcd(2230387-2586705, 3837523) = 1093$ is one factor of n
  - the other factor is $3837523/1093 = 3511$

Quadratic Sieve (2/4)

- Quadratic? $x^2 \equiv$ product of small primes
- How do we construct these useful relations systematically?
- Properties of these relations:
  - product of small primes called factor base
  - make all prime factors appear even times
- Put these relations in a matrix

	2	3	5	7	11	13	17	19
9398	0	0	5	0	0	0	0	1
19095	2	0	1	0	1	1	0	1
1964	0	2	0	0	0	3	0	0
17078	6	2	0	0	1	0	0	0
8077	1	0	0	0	0	0	0	1
3397	5	0	1	0	0	2	0	0
14262	0	0	2	2	0	1	0	0

add

Pick rows where sums of each column are even

Quadratic Sieve (3/4)

- Look for linear dependencies mod 2 among the rows
  - 1st + 5th + 6th = (6, 0, 6, 0, 0, 2, 0, 2) ≡ **0** (mod 2)
  - 1st + 2nd + 3rd + 4th = (8, 4, 6, 0, 2, 4, 0, 2) ≡ **0** (mod 2)
  - 3rd + 7th = (0, 2, 2, 2, 0, 4, 0, 0) ≡ **0** (mod 2)
- When we have such a dependency, the product of the numbers yields a square.
  - $(9398 \cdot 8077 \cdot 3397)^2 \equiv 2^6 \cdot 5^6 \cdot 13^2 \cdot 19^2 \equiv (2^3 \cdot 5^3 \cdot 13 \cdot 19)^2$
  - $(9398 \cdot 19095 \cdot 1964 \cdot 17078)^2 \equiv (2^3 \cdot 3^2 \cdot 5^3 \cdot 11 \cdot 13^2 \cdot 19)^2$
  - $(1964 \cdot 14262)^2 \equiv (3 \cdot 5 \cdot 7 \cdot 13^2)^2$
- Looking for those $x^2 \equiv y^2$ but x ❄ y

Quadratic Sieve (4/4)

- How do we find numbers x s.t.

  $x^2 \equiv$ product of small primes?

  - produce squares that are slightly larger than a multiple of n

    ex. $\left\lfloor \sqrt{i \cdot n} + j \right\rfloor$ for small j

    the square is approximately $i \cdot n + 2j\sqrt{i \cdot n} + j^2$

    which is approximately $2j\sqrt{i \cdot n} + j^2 \pmod{n}$

$$8077 = \left\lfloor \sqrt{17n} + 1 \right\rfloor$$

$$9398 = \left\lfloor \sqrt{23n} + 4 \right\rfloor$$

Probably because this number is small, the factors of it should not be too large. However, there are a lot of exceptions. So it takes time. Also, there are a lot of other methods to generate qualified x values.

The RSA Challenge

- 1977 Rivest, Shamir, Adleman   US$100
  - given RSA modulus n, public exponent e, ciphertext c

    n = 114381625757888867669235779976146612010218296721242362
    562561842935706935245733897830597123563958705058989075
    147599290026879543541

    e = 9007

    c = 968696137546220614771409222543558829057599911245743198
    746951209308162982251457083569314766228839896280133919
    90551829945157815154
  - Find the plaintext message
- 1994 Atkins, Lenstra, and Leyland
  - use 524339 small primes (less than 16333610)
  - plus up to two large primes (16333610 ~ 2^{30})
  - 1600 computers, 600 people, 7 months
  - found 569466 '$x^2 \equiv$small products' equations, out of which only 205 linear dependencies were found

Factorization Records

Year	Number of digits
1964	20
1974	45
1984	71
1994	129 (429 bits)
1999	155 (515 bits)
2003	174 (576 bits)

Next challenge
RSA-640

3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286
782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609

Security of the RSA Function

- **Break RSA** means 'inverting RSA function without knowing the trapdoor'
  - RSA function: $y \equiv x^e \pmod{n}$
- Factor the modulus $\Rightarrow$ Break RSA
  - If we can factor the modulus, we can break RSA
  - If we can break RSA, we don't know whether we can factor the modulus…**open problem** (with negative evidences)
- Factor the modulus $\Leftrightarrow$ Calculate private key d
  - If we can factor the modulus, we can calculate the private exponent d (the trapdoor information).
  - If we have the private exponent d, we can factor the modulus.

Factoring reduces to RSA key recovery

- DeLaurentis, "A Further Weakness in the Common Modulus Protocol for the RSA Cryptosystem," Cryptologia, Vol. 8, pp. 253-259, 1984
  - If you have a pair of RSA public-key/private-key, you can factoring $n=p\cdot q$ with a probabilistic algorithm.
  - An example of the Universal Exponent Factorization method
- Basic idea: find a number b, $0<b<n$ s.t.

  $b^2 \equiv 1 \pmod{n}$ and $b \neq \pm 1 \pmod{n}$ i.e. $1<b<n-1$
  - Note: There are four roots to the equation $b^2 \equiv 1 \pmod{n}$, ± 1 are two of them, all satisfy $(b+1)(b-1) = k\cdot n = k\cdot p\cdot q$, since $0<b-1<b+1<n$, we have either ($p \mid b-1$ and $q \mid b+1$) or ($q \mid b-1$ and $p \mid b+1$), therefore, one of the factor can be found by gcd(b-1,n) and the other by n/gcd(b-1,n) or gcd(b+1,n)

Factoring reduces to RSA key recovery

- Algorithm to find b: Pr{success per repetition} = ½
  1. Randomly choose a, 1<a<n-1, such that gcd(a, n) = 1
  2. Find minimal j, $a^{2^j h} \equiv 1 \pmod n$ (where h satisfies $e \cdot d - 1 = 2^t h$)
  3. $b = a^{2^{j-1} h}$, if b ❄ -1 (mod n), then gcd(b-1, n) is the result, else repeat 1-3
- Note: If we randomly choose $b \in Z_n^*$ and find out that $b^2 \equiv 1 \pmod n$, the probability that b=1, b=-1, b=c(≠±1), or b=-c(≠±1) would be equal; $Pr\{success\}=Pr\{a^{2^{j-1}h} \neq \pm 1\}=1/2$
- Ex: p=131, q=199, n=p·q=26069, e=7, d=22063

  $\phi(n)=(p-1)(q-1)=25740=2^2 * 6435$ | $ed-1=154440 = 2^3 * 19305$,

  choose a=3, try j=1 ($3^{2^1 19305}=1$), $b= a^{2^{j-1}h}= 3^{19305} = 5372$ ($\neq \pm 1$)

  p = gcd(b-1,n) = gcd(5371,26069) = 131, q = n/p = 199

Factoring reduces to RSA key recovery

- The above result says that "if you can recover a pair of RSA keys, you can factoring the corresponding $n=p \cdot q$" i.e. "once a private key d is compromised, you need to choose a new pair of (n, e) instead of changing e only"
- The above result suggests that a scheme using (n, e_1), (n, e_2), … (n, e_k) with a common n for each k participants without giving each one the value of p, q is insecure. You should not use the same n as some others even though you are not explicitly told the value of p and q.

Factoring reduces to RSA key recovery

- The above result also suggests that if you can recover arbitrary RSA key pair, you can solve the problem of factoring n. Whenever you get an **n**, you can form an RSA system with some ***e*** (assuming gcd(e, ϕ(n))=1), then use your method to solve the private exponent ***d*** without knowing p and q, after that you can factor n.
- Although factoring is believed to be hard, and factoring breaks RSA, breaking RSA does not simplify factoring. Trivial non-factoring methods of breaking RSA could therefore exist. (What does it mean by breaking RSA? plaintext recovery? key recovery?…)

different things

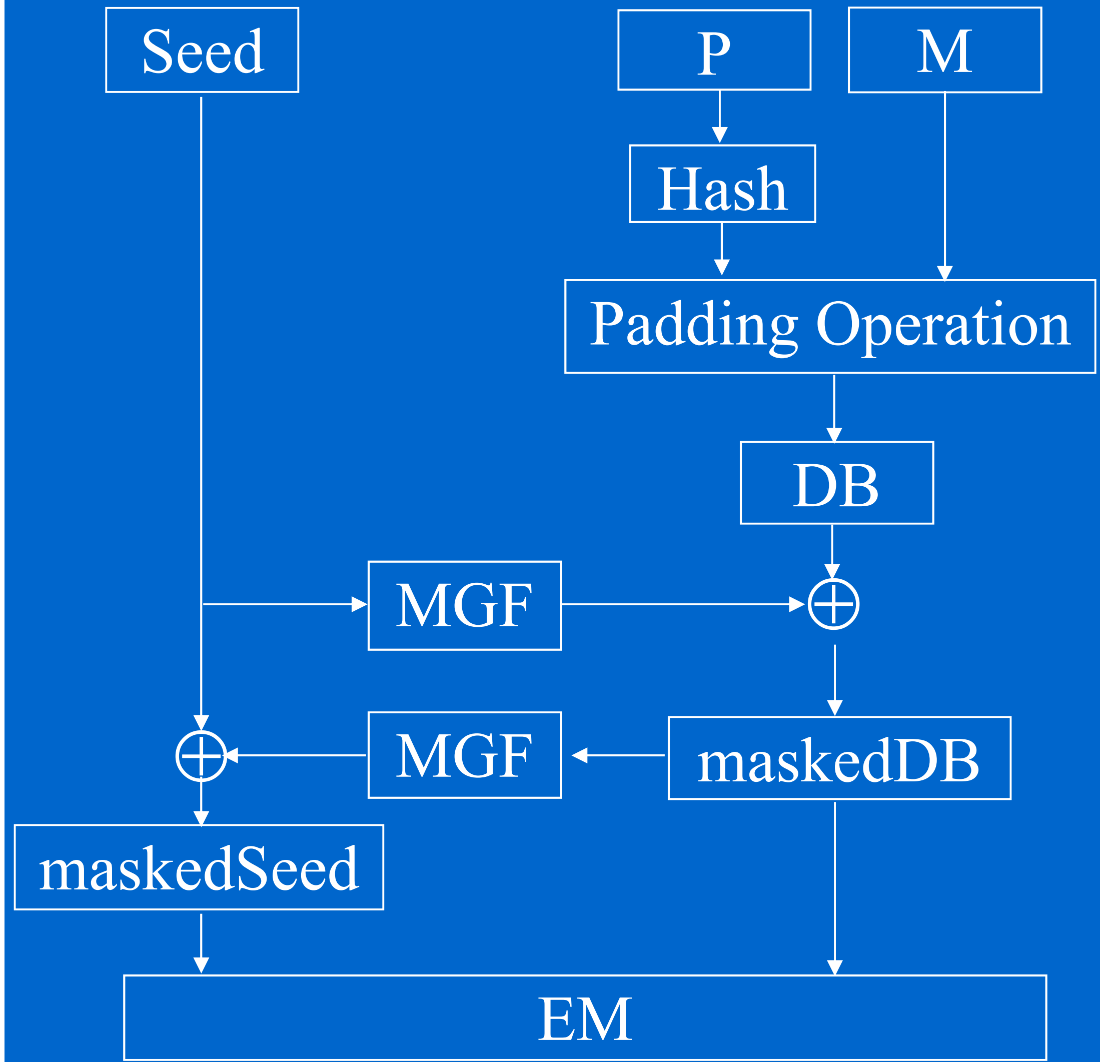
Deterministic Encryption

- RSA Cryptosystem is a deterministic encryption scheme, i.e. a plaintext message is encrypted to a fixed ciphertext message
- Suffers from chosen plaintext attack
  - an attacker compiles a large codebook which contains the ciphertexts corresponding to all possible plaintext messages
  - in a two-message scheme, the attacker can always distinguish which plaintext was transmitted by observing the ciphertext (does not satisfy the Semantic Security Notation)
- Add randomness through padding

RSA PKCS #1 v1.5 padding

- Ex. k=128 bytes (1024 bits) PKCS#1 v1.5 RSA
  - plaintext message M (at most 128-3-8=117 bytes)
  - pseudorandom nonzero string PS (at least 8 bytes)
  - message to be encrypted $m = 00||02||PS||00||M$
  - encryption: $c \equiv m^e \pmod{n}$
  - decryption: $m \equiv c^d \pmod{n}$
- c is now random corresponding to a fixed m, however, this only adds difficulties to the compilation of ciphertexts (a factor of 2^{64} times if PS is 8 bytes)

PKCS #1 v2 padding - OAEP



M: message (emLen-1-2hLen bytes)
P: encoding parameters,
an octet string
MGF: mask generation function
Hash: selected hash function
(hLen is the output bytes)
DB=Hash(P)||PS||01||M
PS is length emLen-
||M||-2hLen-1 null bytes
Seed: hLen random bytes
dbMask: MGF(seed, emLen-hLen)
maskedDB = DB ⊕ dbMask
seedMask:
MFG(maskedDB, hLen)
maskedSeed = seed ⊕ seedMask

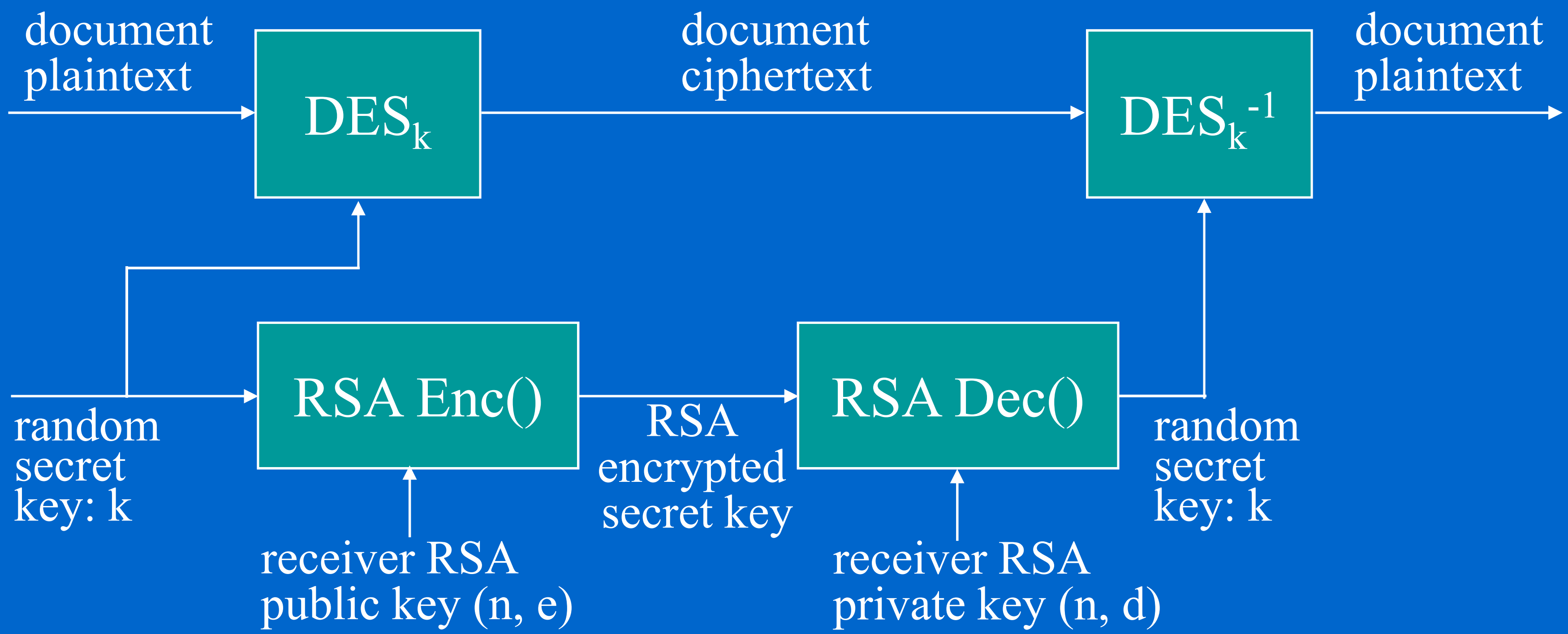
EM: encoded message (emLen bytes)
EM = maskedSeed||makedDB

PKCS #1 v2 padding - OAEP

- Optimal Asymmetric Encryption (OAE)
  - M. Bellare, "Optimal Asymmetric Encryption - How to Encrypt with RSA," Eurocrypt'94
- Optimal Padding in the sense that
  - RSA-OAEP is semantically secure against adaptive chosen ciphertext attackers in the random oracle model
  - the message size in a k-bit RSA block is as large as possible (make the most advantage of the bandwidth)
- Following by more efficient padding schemes:
  - $OAEP^{+}$, $SAEP^{+}$, REACT

Digital Envelop

- Hybrid system (public key and secret key)
  - computation of RSA is about 1000 times slower than DES
  - smaller exponent is faster (but usually dangerous)



RSA Fast Decryption with CRT

- Public key (n, e)

  $n = p \cdot q$, p and q are large prime integers
  $\gcd(e, \phi(n)) = 1$ s.t. $\exists d,\ e \cdot d \equiv 1 \pmod{\phi(n)}$
  $\phi(n) = (p-1)(q-1)$ $3 \le e \le n-1$

- Private Key (n, d) or

  (n, p, q, dp, dq, qInv)

  $e \cdot dp \equiv 1 \pmod{p-1}$
  $e \cdot dq \equiv 1 \pmod{q-1}$
  $q \cdot qInv \equiv 1 \pmod{p}$

- Encryption $c \equiv m^e \pmod{n}$
- Decryption $m \equiv c^d \pmod{n}$ or

  $m_1 \equiv c^{dp} \pmod{p}$
  $m_2 \equiv c^{dq} \pmod{q}$
  $h \equiv qInv \cdot (m_1 - m_2) \pmod{p}$
  $m \equiv m_2 + h \cdot q \pmod{n}$

  CRT

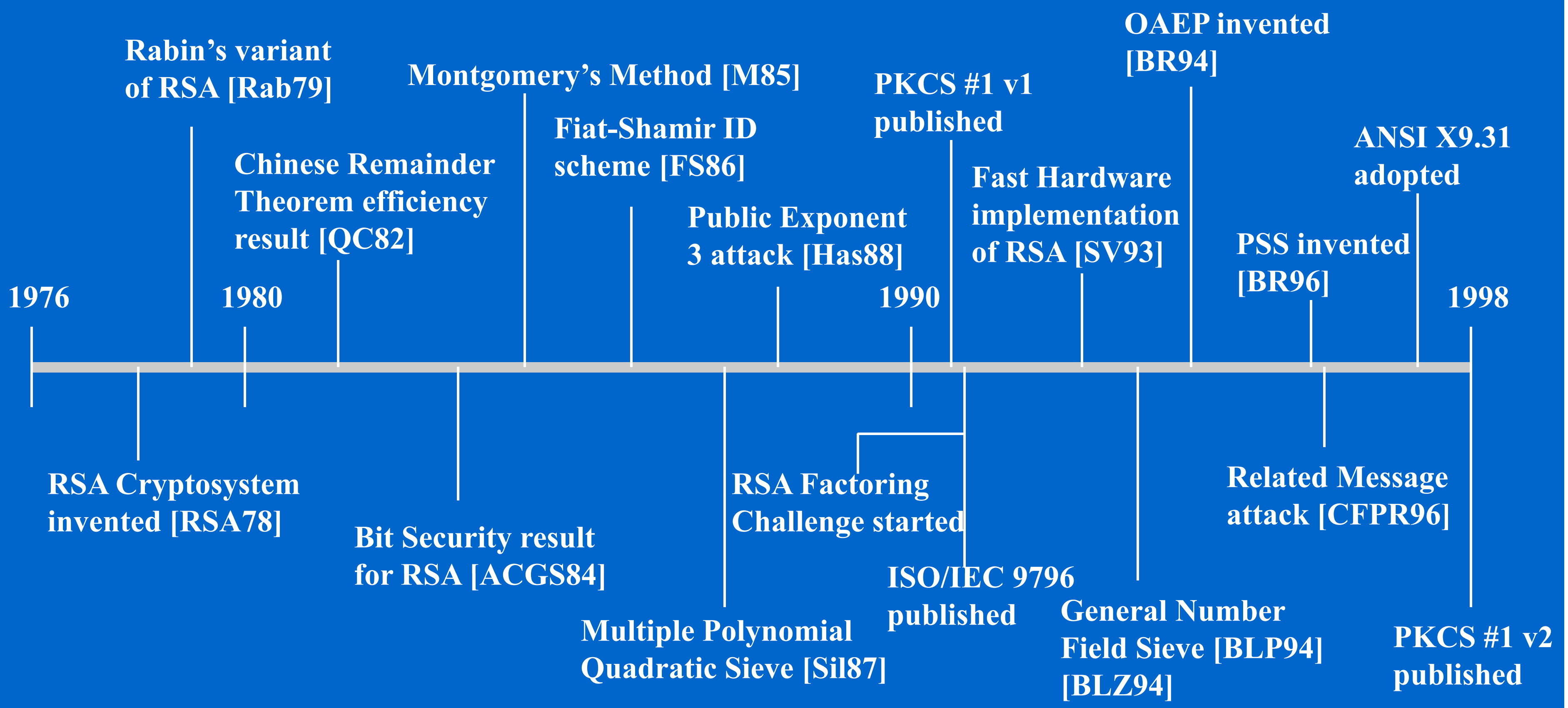
  $m_1 \equiv (m^e)^{dp} \equiv m^{e \cdot dp} \equiv m \pmod{p}$
  $m_2 \equiv (m^e)^{dq} \equiv m^{e \cdot dq} \equiv m \pmod{q}$

  $m \equiv m_2 \pmod{q}$ and
  $m \equiv m_2 + qInv \cdot (m_1 - m_2) \cdot q \equiv m_1 \pmod{p}$

Multi-Prime RSA

- RSA PKCS#1 v2.0 Amendment 1
- the modulus n may have more than two prime factors
- only private key operations and representations are affected (p, q, dp, dq, qInv) (r_i, d_i, t_i)
  - $n = r_1 \cdot r_2 \cdot \ldots \cdot r_k$, $k \geq 2$, where $r_1 = p$, $r_2 = q$
  - $e \cdot d_i \equiv 1 (\text{mod } r_i - 1)$, $i=3,\ldots k$
  - $r_1 \cdot r_2 \cdot \ldots \cdot r_{i-1} \cdot t_i \equiv 1 \ (\text{mod } r_i)$ $i=3,\ldots k$
- Decryption:
  1. $m_1 \equiv c^{dp} \ (\text{mod } p)$
  2. $m_2 \equiv c^{dq} \ (\text{mod } q)$
  3. if $k>2$ $m_i \equiv c^{d_i} \ (\text{mod } r_i)$, $i=3,\ldots, k$
  4. $h \equiv (m_1 - m_2)\ qInv \ (\text{mod } p)$
  5. $m = m_2 + q \cdot h$
  6. if $k>2$, $R= r_1$, for $k=3$ to k do
     a. $R = R \cdot r_{i-1}$
     b. $h \equiv (m_i - m) \cdot t_i \ (\text{mod } r_i)$
     c. $m = m + R \cdot h$
- advantages: lower computational cost for the decryption (and signature) primitives if CRT is used (also see **6.8.14**)

Factoring & RSA Timeline



Alternative PKC's

- ElGamal Cryptosystem (Discrete-log based)
  - Also suffers from long keys
- NTRU (Lattice based)
  - Utilizes short keys
  - Proprietary (License issues prevent from wide implementation)
  - Recently, a weakness found in the signature scheme
- Elliptic Curve Cryptosystems
  - Emerging public key cryptography standard for constrained devices.
- Paillier Cryptosystem (High order composite residue based)
- Goldwasser-Micali Cryptosystem (QR based)
  - very low efficiency

Miller-Rabin Primality Test

- Why does it work?

bottom line of Miller-Rabin test

  - if n is prime, $a^{n-1} \equiv 1 \pmod{n}$ (Fermat Little theorem)
  - therefore, if $b_k \equiv a^{2^k m} \equiv a^{n-1}$ ❄ 1 (mod n), n must be composite
  - however, there are many composite numbers that satisfy $a^{n-1} \equiv 1 \pmod{n}$, Miller-Rabin test can detect many of them
  - $b_0, b_1, \ldots, b_{k-1}$ ($\equiv a^{(n-1)/2} \pmod{n}$) is a sequence s.t. $b_{i-1}^2 \equiv b_i \pmod{n}$
  - we consider only $b_{k-1}^2 \equiv a^{n-1} \equiv 1 \pmod{n}$ ← n is pseudo prime
  - if $b_i \equiv 1$ and b_{i-1} ❄ ± 1, then n is composite ← basic factoring principle
  - if $b_i \equiv 1$ and $b_{i-1} \equiv 1$, consider b_{i-1} and then b_{i-2}…
    if $b_0 \equiv 1$, could be prime, no guarantee
  - if $b_i \equiv 1$ and $b_{i-1} \equiv -1$ (b_{i-2} ❄ ± 1), could be prime, no guarantee

there is no chance to apply basic factoring principle

Miller-Rabin Primality Test

- In summary:

  $b_0, b_1, b_2, \ldots b_{i-1}, b_i, \ldots b_k$

  there are four cases:
  - Case 1: $b_k \neq 1$ n is a composite number
  - Case 2: $b_k = 1$, let i be the minimal i, $k \geq i > 0$ such that $b_i = 1$ and $b_{i-1} \neq \pm 1$ n is a composite number (with nontrivial factors calculated)
  - Case 3: $b_k = 1$, let i be the minimal i, $k \geq i > 0$ such that $b_i = 1$ and $b_{i-1} = -1$ a pseudo prime number
  - Case 4: $b_k = 1$, $b_0 = 1$ a pseudo prime number

---

4 possible sequences for $b_0, b_1, b_2, \ldots b_{i-1}, b_i, \ldots b_k$:

342, 22, 5, 1, 1, 1, 1, …, 1	composite, factored
45, 5634, 325, 213, -1, 1, …, 1	possibly prime
1, 1, 1, …, 1	possibly prime
214, 987, …, 8931, 321, 134	composite

M-R Test: Prime Modulus

- consider n being a *prime number* p
- p-1 is an even number, therefore, let $p-1=2^k \cdot m$, m is odd
- choose one $a \in_R Z_p^*$, let r be the smallest integer s.t. $a^r \equiv 1 \pmod p$, i.e. r is the order of a modulo p, $ord_p(a)$
- (exercise 3.9) $a^{p-1} \equiv 1 \pmod p \Rightarrow r \mid p-1$
- because $r \mid p-1$ $(= 2^k \cdot m)$, one of $\{m, 2 \cdot m, 2^2 \cdot m, \ldots 2^k \cdot m\}$ might be r (probability reduces if m has many factors)
- Case 1: if "$2^i \cdot m$ (for some i>0) is r", $a^{2^{i-1} \cdot m}$ must be -1
  - r is the smallest integer s.t. $a^r \equiv 1 \Rightarrow$ square root of a^r must be –1
  - $\{a^m, a^{2 \cdot m}, \ldots a^{2^i \cdot m}\}$ is $\{?, ?, -1, 1, \ldots 1\}$
- Case 2: if "none of $2^i \cdot m$ is r" or "m is r", $a^{2^i \cdot m}$ must all be 1,
  - $\{a^m, a^{2 \cdot m}, \ldots a^{2^i \cdot m}\}$ is $\{1, 1, 1, 1, \ldots 1\}$
  - try some other $a \in Z_p^*$

Miller-Rabin Primality Test

Why does it work??? an inside view

- $b_i \equiv 1 \pmod{n}$ and $b_{i-1} \not\equiv \pm 1 \pmod{n}$ happens when $b_i \equiv 1 \pmod{p_i}$ for all prime factors p_i of n and

  $b_{i-1} \equiv 1 \pmod{p_i}$ for some prime factors p_i but
  $b_{i-1} \equiv -1 \pmod{q_i}$ for other prime factors q_i

  Note: for a prime modulus p, $a^{ord_p(a)} \equiv 1 \pmod{p}$
  if $ord_p(a)$ is even then $a^{ord_p(a)/2} \equiv -1 \pmod{p}$

- ex. $n = 561 = 3 \times 11 \times 17$, $560 = 16 \times 35 = 2^4 \times 35$
  let $a = 2$

  $b_0 \equiv 263 \pmod{561} \equiv -1 \pmod{3} \equiv -1 \pmod{11} \equiv 8 \pmod{17}$
  $b_1 \equiv 166 \pmod{561} \equiv 1 \pmod{3} \equiv 1 \pmod{11} \equiv -4 \pmod{17}$
  $b_2 \equiv 67 \pmod{561} \equiv 1 \pmod{3} \equiv 1 \pmod{11} \equiv -1 \pmod{17}$
  $b_3 \equiv 1 \pmod{561} \equiv 1 \pmod{3} \equiv 1 \pmod{11} \equiv 1 \pmod{17}$

  i.e. inconsistent progress w.r.t each prime factor

Subset Sum Problem in NP-Complete

- Subset Sum Problem (SSP)

  Given a set B of positive numbers and a number d

  - Search SSP: find a subset $\{b_j\} \subseteq B$ s.t. $d = \sum b_j$
  - Decision SSP: decide if there exists a subset $\{b_j\} \subseteq B$ s.t. $d = \sum b_j$
  - Decision SSP is equivalent to Search SSP: (by elimination)

- Subset Sum Problem is NP-complete
  - Cook-Levin Thm: Satisfiability Problem (SAT) is NP-Complete
  - SAT $\leq_M$ SSP: there exists a poly-time reduction to convert a formula ϕ to an instance <B,d> of SSP problem
    - If the formula ϕ is satisfiable, <B,d> $\in$ SSP
    - If <B,d> $\in$ SSP, formula ϕ is satisfiable

  Therefore, SSP is also NP-complete

SAT $\leq_M$ D-Subset Sum

- Given a formula ϕ with k clauses $C_1, C_2, \ldots, C_k$ and n variables
  - For each variable x, create 2 integers n_{xt} and n_{xf}
  - For each clause C_j of lengh ℓ_j, create ℓ_j-1 integers m_{j1}, m_{j2}, …
  - Choose t so that T must contain exactly one of each (n_{xt} or n_{xf}) pairs and at least one from each clause
- This construction can be carried out in poly-time
- ϕ is satisfiable iff there exists solution to this SSP

SAT $\leq_M$ D-Subset Sum (cont'd)

Example: $(x \vee y \vee z) \wedge (\neg x \vee \neg a) \wedge (a \vee b \vee \neg y \vee \neg z)$

	x	y	z	a	b	C_1	C_2	C_3
n_{xt}	1	0	0	0	0	1	0	0
n_{xf}	1	0	0	0	0	0	1	0
n_{yt}		1	0	0	0	1	0	0
n_{yf}		1	0	0	0	0	0	1
n_{zt}			1	0	0	1	0	0
n_{zf}			1	0	0	0	0	1
n_{at}				1	0	0	0	1
n_{af}				1	0	0	1	0
n_{bt}					1	0	0	1
n_{bf}					1	0	0	0
m_{11}						1	0	0
m_{12}						1	0	0
m_{21}						0	1	0
m_{31}						0	0	1
m_{32}						0	0	1
m_{33}						0	0	1
t	1	1	1	1	1	3	2	4

Encode all numbers with a base larger than all entries of t e.g. 10