05506037 Cryptography Algorithms

Lecture 8: Public Key Cryptography II: RSA, Rabin and Elgamal Cryptosystem

ดร. รุ่งรัตน์ เวียงศรีพนาวัลย์

This lecture note is the combination from the material from Cryptotraphy and Network Security from Forouzan and from Prof. Willy Susilo's lecture notes in CSCI361: Cryptography and Secure Applications, University of Wollongong, Australia.

Outline

- Public Key Crytography Families
- RSA.
  - Encryption and decryption.
- Using RSA.
- Choosing p and q.
- Implementation considerations.
- Some comments on factoring.
- Finding and testing primes.
- Fast exponentiation.
- Assessing the security of RSA.

Public-Key Algorithm Families of Practical Relevance [3]

- Integer-Factorization Schemes
  - public-key scheme หลายวิธีสร้างขึ้นด้วยหลักการที่ว่า it is difficult to factor large integers. (การแยกตัวประกอบของจำนวนเต็มที่ใหญ่มาก **ยาก**) => **(Factorization Problem)**
  - อัลกอริทึมที่มีชื่อเสียงที่สุดในกลุ่มนี้ คือ RSA.
- Discrete Logarithm Schemes
  - หลาย algorithm สร้างขึ้นจาก ปัญหาทางคณิตศาสตร์ที่รู้จักในชื่อ discrete logarithm problem ใน finite fields.
  - ตัวอย่างที่เป็นที่รู้จัก เช่น Diffie–Hellman key exchange, Elgamal encryption or the Digital Signature Algorithm (DSA).
- Elliptic Curve (EC) Schemes
  - Elliptic curve public-key schemes เป็น Generalization ของ discrete logarithm algorithm.

    แต่ก่อนถูกจำกัดด้วย Hardware performant จึงไม่ใช้ ปัจจุบันใช้เยอะ
  - ตัวอย่างที่เป็นที่รู้จัก เช่น Elliptic Curve Diffie–Hellman key exchange (ECDH) และ Elliptic Curve Digital Signature Algorithm (ECDSA).

RSA ไม่ได้ถูกเลือกเป็น base Alogorithm
แต่ถูกใช้เยอะจนเหมือนจะเป็น base alogrithm

Introduction to RSA

- The RSA Public--Key Cryptosystem (Rivest, Shamir and Adleman (1978)) is the most popular and versatile PKC.
- It is the *de facto* standard for PKC.
- It supports *secrecy and authentication* and can be used to produce *digital signatures.*
- RSA uses the knowledge that
  - *it is easy to find primes and multiply them* together to construct composite numbers,
  - but it is *difficult* to factor a composite number.
- RSA applies the factorization problems as the underlining intractable problem.

RSA Inventors [3]

เช่น $p = 2, q = 5$

$n = p \cdot q$
$= 2 \cdot 5$
$= 10$

$Z_{10} = \{0,1,2,3,4,5,6,7,8,9\}$

• มี 4 ตัวเหมือนกัน

$f(n) = (p-1)(q-1)$
$= (2-1)(5-1)$
$= 1 \cdot 4$
$= 4$ ตัว

∴ 2 กับ 5 มี Relatively prime 4 ตัว

RSA algorithm

Choose two primes p and q.

- Compute n = pq and m=f(n)= (p-1)(q-1).
- f(n) is Euler's totient function:
  It is *the number of positive integers less than n* that are *relatively prime to n*.

Choose e, $1 \leq e \leq m - 1$, such that gcd(e,m)=1.

The Public key is (e, n).

Finds d such that ed=1 mod m.

- This is possible because of the choice of e. ทำไม ???)
- d is the multiplicative inverse of e modulo m (can be found using the extended Euclidean (gcd) algorithm.)

The Private key is (d, p, q).

- Let P denote a plaintext block, and C denote the corresponding ciphertext block.
- Let (z_A, Z_A) denote the private and public components of Alice's key.

- If Bob wants to encrypt a message X for Alice. He uses Alice's public key and forms the cryptogram:

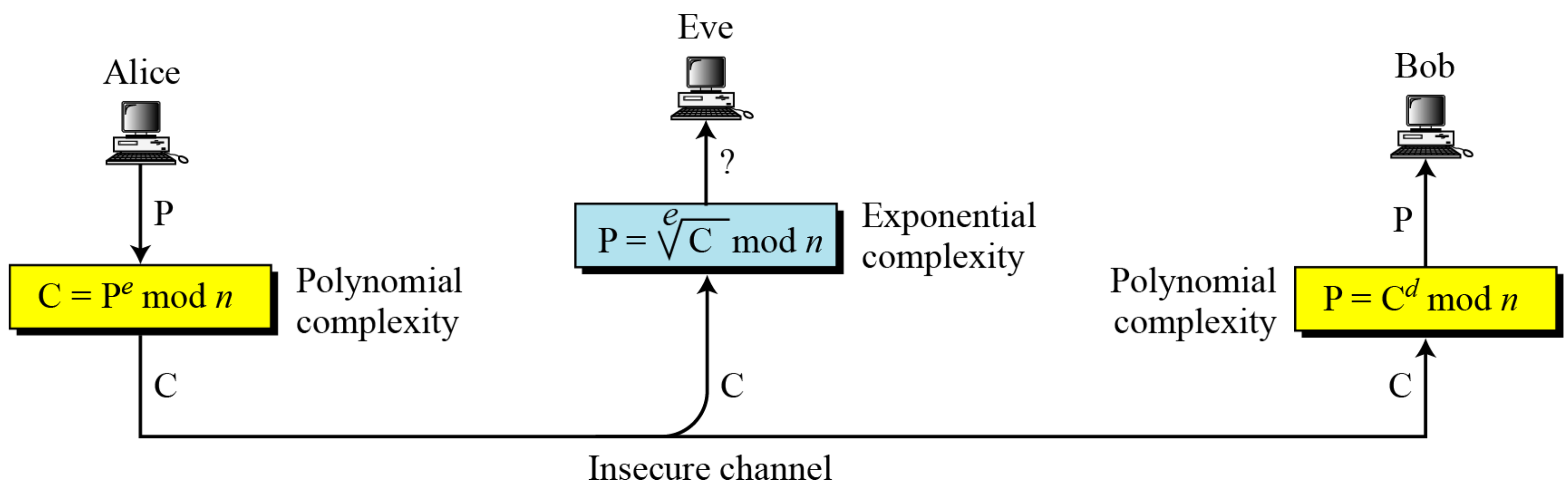
$$C = E_{Z_A}(P) = P^e \bmod n$$

- When Alice wants to decrypt C, she uses the private component of her key $z_A=(d,n)$ and calculates

$$P=D_{z_A}(C) = C^d \bmod n$$

P and C are both integers in {0, 1, 2, ..., n-1}

Diagram from [2]



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

Figure 10.5 *Complexity of operations in RSA*

Example

- Alice chooses p=11 and q=13.

  $n = p \cdot q = 11 \cdot 13 = 143$

  $m = (p-1)(q-1) = (11-1)(13-1) = 10 \cdot 12 = 120$

  e=37 ➔ gcd(37, 120)=1 Proof by yourself

- Using the gcd algorithm
  - we find d such that ed=1 mod 120; using extended euclidean algorithm.
  - d= 13
- Alice publishes (e,n) (37,143) as her public key and keeps (d,p,q) (13, 11 , 13)

Bob sends his message to Alice – Binary String Input

- 1. Gets Alice's public key ex. from public key directory.
- 2. To encrypt: Break the input binary string into blocks of u bits, where $2^u \leq 142$ (n-1) , so we choose u=7. (ทำไมถึงต้องเป็น n-1)
  - In general there is some agreed padding scheme from the message space into the space on which a "single run" of the cipher can act. Optimal Asymmetric Encryption Padding (OAEP) is often used. (we will talk about it later)
- For each 7-bit block X, a number between 0 and 127 inclusive, we calculate the ciphertext as $C=P^e$ mod n
- For X=2=(0000010) Bob sends

$E_Z(P)=P^{37}=$ 106 (mod) ➔ Y=01101010 ต้องเปลี่ยนเป็นฐาน10

To decrypt:

Alice calculates $P=D_Z(C)=12$ =2 (mod 143)

An Example of RSA public directory

User	(n,e)
Alice	(143,37)
Bob	(117,5)
Fred	(4757,11)

Exercise 1:

- Fred, wants to send a message 1101010 to Alice
- 1. Show the encryption process and and find the ciphertext.

$1101010_2 = 106_{10}$

Alice pk = (37, 143)

$c = M^e \bmod n$

$C = 106^{37} \bmod 143$

$=$

- 2. Show the decryption process that $P=C^d \bmod n$

- An important property of the RSA algorithm is that encryption and decryption are the same function: both exponentiation modulo n.

$$E_{Z_A}(D_{Z_A}(X)) = X$$

- Example:
  - First decrypt: $2^{13} = 41 \bmod 143$
  - Then encrypt: $41^{37} = 2 \bmod 143$

- This is the basis of using RSA for authentication

Using RSA :The RSA system can be used for:

Confidentiality:	• To hide the content of a message P, A sends $E_{Z_B}(P)$ to B.
Authentication (Digital Signature)	• To ensure integrity of a message X,
Secrecy and authentication:	• Both

Authentication:

To ensure integrity of a message X,

- Alice *signs* the message by using her decryption key to form $D_{z_A}(X)$ and sends $(X, D_{z_A}(X)) = (X, S)$ to Bob.
- When Bob wants to *verify the authenticity* of the message:
  - He computes $X' = E_{z_A}(S)$.
  - If $X'=X$ the message is accepted as authentic and from Alice.
- Both message integrity and sender authenticity are verified.
  - This is true because even one bit change to the message can be detected, and because z_A is known only to Alice.
- This method is inefficient. We will see later that Alice may compute a hash value of X and then apply D_{z_A} to the result.
- We will talk about Digital Signatures later anyway.

Secrecy and authentication:

- If we need secrecy and authentication the following can be used:

1. A sends $Y = E_{Z_B}(X, D_{z_A}(X))$ to B. ดังนั้นเฉพาะ B เท่านั้นจึงจะ ถอดรหัสข้อความนี้ได้
2. B ทำการถอดรหัส Y ด้วย private key ของ $B(z_B)$

$$D_{z_B}(Y) = X, D_{z_A}(X)$$

3. B จะได้ค่า สอง ค่า คือ X และ $D_{z_A}(X)$
4. B ใช้ public key ของ A (Z_A) ในการ ตรวจสอบว่า A เป็นคนส่งข้อมูลนี้จริงๆ

   โดยทำการเช็คว่า $E_{Z_A}(D_{z_A}(X)) = X$ หรือไม่

This scheme provides *non-repudiation* if Bob holds on to $D_{z_A}(X)$.

That is, Bob is protected against Alice trying to deny sending the message.

Let’s see how RSA works

Choosing p and q

- The main conditions on p and q are:
  - They must be at least 100 decimal digits long (about 330 bits) (512 now).
  - They must be of similar size , say both 100 digits.

To choose each number the user does the following:

Randomly choose a random number b which is 100 digits long, or whatever length is appropriate.

↓

Checks to see *if b is prime*. Usually using a probabilistic primality testing algorithm.

↓

If b is not prime, choose another value for b.

ปัจจุบัน algorithm ที่ใช้ในการหา primes Number ที่มีประสิทธิภาพสุดคือ Probabilistic (แม้ว่าแต่ไม่ถูก 100 %) อาจจะถูกแค่ 80 - 90 %

Finding primes

- An algorithm to generate all primes does not exist.
- However, given a number n, there exist efficient algorithms
  - to *check whether it is prime or not.*
- Such algorithms are called primality testing algorithms.
- As mentioned earlier, prime generation for RSA is basically just a matter of guessing and testing.
- Primality Testing: 2 Types of algorithms

Deterministic Algorithm (ใส่ค่าแล้วได้คำตอบเลย)
Output จะได้ตัวเดิมตลอด

Deterministic algorithms	• for proving primality are non-trivial and are only advisable on high performance computers.
Probabilistic	• tests allow an *educated guess* as to whether a candidate number is prime or not. • This means that the probability of the guess being wrong can be made arbitrarily small.

One of Probability Primal Testing: Lehman's test

- Let n be an odd number. For any number define

$$e(a,n) = a^{\frac{n-1}{2}} \bmod n$$

$$G = \{e(a,n) : a \in Z_n^*\}$$

where $Z_n^*=\{1,2,\dots,n-1\}$.

- Example: n=7

$2^3=1,\ 3^3=6,\ 4^3=1,\ 5^3=6,\ 6^3=6 \rightarrow G=\{1,6\}$

Lehman's theorem:

$$G = \{e(a,n) = a^{\frac{n-1}{2}} \bmod n : a \in Z_n^*\}$$

If n is odd, G={1,n-1}* if and only if n is prime.

*G={1,n-1} = {1,2,3,….,n-1)

Example: n=15 isn't prime: (n-1)/2=7

2^7=8 mod 15, 3^7=12 mod 15,

Example: n=13 (prime): (n-1)/2=6

2^6=-1 mod 13, 3^6=1 mod 13, 4^6=1 mod 13,

5^6=-1 mod 13, 6^6=-1 mod 13, 7^6=-1 mod 13,

8^6=-1 mod 13, 9^6=1 mod 13, 10^6=1 mod 13,

11^6=1 mod 13, 12^6=1 mod 13.

สไลด์นี้แสดงวิธีว่าทำไมถึงใช้ Lehman test ในการ ตรวจสอบว่า n เป็นจำนวนเฉพาะหรือไม่

- Thus, we have the following test:

if (gcd(a,n) >1) *return*('composite')
else *if* ($a^{(n-1)/2}=1$) *or* ($a^{(n-1)/2}=-1$)
return('prime_witness')
else
return('composite')

- If for a given n
  - the test returns prime_witness for 100 randomly chosen a,
- then
  - the probability of n being not prime (i.e. being a composite disguised as a prime) is less than 2^{-100}

อย่างไรก็ดีจริงๆ ผลจาก Lehmah test ไม่ใช่จำนวนเฉพาะเสมอไป

- What actually holds is the following:

If n is odd & prime then G={1,n-1}.

- เลขบางตัวจะผ่าน Lehmah test ไม่ว่า a จะเป็นค่าอะไร
- These are the Carmichael numbers…

  561=11*51, 1105=13*85, 1729=19*91, 2465=29*85, 2821=31*91, 6601=7*943, 8911=7*1273, 10585=5*2117, 15841=7*2263, 29341=13*2257 …

There are an infinite number of these, but they become very rare as we look at larger numbers!

Fast exponentiation

- Exponentiation can be performed by repeated multiplication. In general, we can use the *square and multiply* technique.
- To calculate X^{α}:

1. Write α in base two: นำตัวยกกำลังมาเขียนเป็นเลขฐานสอง

$$\alpha = \alpha_0 2^0 + \alpha_1 2^1 + \alpha_2 2^2 + \ldots \alpha_{n-1} 2^{n-1};\ \alpha_I = \{0,1\}$$

2. Calculate X^{2^i}, $1 \leq i \leq n-1$.
3. Use $X^{\alpha} = (X^{2^0})^{\alpha_0} * (X^{2^1})^{\alpha_1} \ldots (X^{2^{n-1}})^{\alpha_{n-1}}$ and

Multiply the X^{2^i} for which α_i is not zero.

- A partial example: n=179,e=73.

$X=2 \rightarrow Y=2^{73}$ mod 179.

$73=64+8+1=2^6+2^3+2^0$

$Y=2^{64+8+1}=2^{64} . 2^8 . 2^1$

This is only a partial example because we haven't looked at calculating the elements of the last line.

Precomputation:

$X^2=X.X$ *mod n*

$X^4= {X^2}^2 =X^2 .X^2$ *mod n*

$\vdots$

$X^{2^{n-1}} = X^{2^{n-2}} .X^{2^{n-2}}$ *mod n*

This is a total of n-1 multiplications, all *mod n*

Example: N=1823, n=$\log_2 1822$=11.

- Calculate Y=5^{375} mod N
- Precomputation:

X^1	5	X^2	25	X^4	625
X^8	503	X^{16}	1435	X^{32}	1058
X^{64}	42	X^{128}	1764	X^{256}	1658
X^{512}	1703	X^{1024}	1639		

There are various other "tricks" for calculating powers too, but we aren't going to look at them here!

Rule:

- Never store a number larger than N.
- Never multiply two numbers large than N.

375=256+64+32+16+4+2+1.

5^{375}=5*25*625*1435*1058*42*1658

= 591 mod 1823

ตัวยอย่างของ พารามิเตอร์ของ RSA สำหรับ n =1024 (p และ q ประมาณ 512

$$
\begin{aligned}
p = {} & E0DFD2C2A288ACEBC705EFAB30E4447541A8C5A47A37185C5A9 \\
& CB98389CE4DE19199AA3069B404FD98C801568CB9170EB712BF \\
& 10B4955CE9C9DC8CE6855C6123_h \\
q = {} & EBE0FCF21866FD9A9F0D72F7994875A8D92E67AEE4B515136B2 \\
& A778A8048B149828AEA30BD0BA34B977982A3D42168F594CA99 \\
& F3981DDABFAB2369F229640115_h \\
n = {} & CF33188211FDF6052BDBB1A37235E0ABB5978A45C71FD381A91 \\
& AD12FC76DA0544C47568AC83D855D47CA8D8A779579AB72E635 \\
& D0B0AAAC22D28341E998E90F82122A2C06090F43A37E0203C2B \\
& 72E401FD06890EC8EAD4F07E686E906F01B2468AE7B30CBD670 \\
& 255C1FEDE1A2762CF4392C0759499CC0ABECFF008728D9A11ADF_h \\
e = {} & 40B028E1E4CCF07537643101FF72444A0BE1D7682F1EDB553E3 \\
& AB4F6DD8293CA1945DB12D796AE9244D60565C2EB692A89B888 \\
& 1D58D278562ED60066DD8211E67315CF89857167206120405B0 \\
& 8B54D10D4EC4ED4253C75FA74098FE3F7FB751FF5121353C554 \\
& 391E114C85B56A9725E9BD5685D6C9C7EED8EE442366353DC39_h \\
d = {} & C21A93EE751A8D4FBFD77285D79D6768C58EBF283743D2889A3 \\
& 95F266C78F4A28E86F545960C2CE01EB8AD5246905163B28D0B \\
& 8BAABB959CC03F4EC499186168AE9ED6D88058898907E61C7CC \\
& CC584D65D801CFE32DFC983707F87F5AA6AE4B9E77B9CE630E2 \\
& C0DF05841B5E4984D059A35D7270D500514891F7B77B804BED81_h
\end{aligned}
$$

RSA implementation considerations

- Relative to DES or AES;
  - RSA is a much slower algorithm.
  - RSA has a much larger secret key.
- Storage of p and q requires from about 330 bits each, n is about 660 bits or larger.
- Exponentiation is relatively slow for large n, especially in software.
- It can be shown that multiplication needs m + 7 clock pulses if n is m-bits in length.

RSA มีปัญหาการทำ Factorization

Some comments on factoring I

- The most powerful known (pretty much general purpose) factoring algorithm is the *general number field sieve*.
- It uses

$$\boxed{O\left(\exp\left(\frac{64}{9}\log n\right)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right)}$$

steps to factor a number n.

Some comments on factoring I

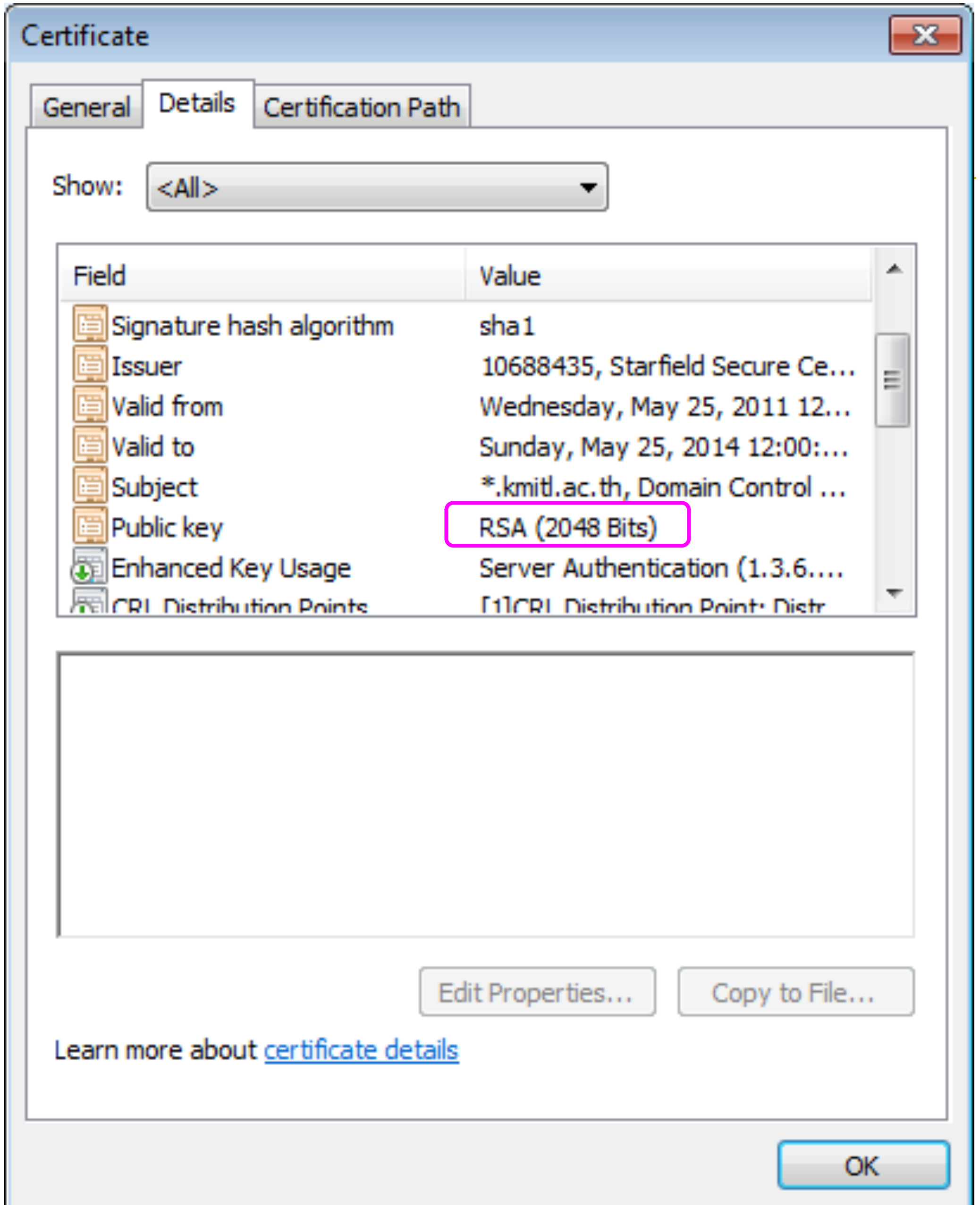
- One of the best factoring results is for a 663-bit number (RSA-200).
  - It was factored using a general number field sieve (announced May 9 2005).
  - It took several months of work undertaken by a system of 80 AMD Opteron processors (~2.6GhZ).
- Today for sensitive applications, a 1024 bit modulus, and in some cases a 2048 bit modulus, are considered necessary.
  - If you look at the Certificates in web browsers you will sometimes find as large as 4096 bits.

allenge Number	Prize ($US)	Status	Submission Date	Submitter(s)
RSA-576	$10,000	Factored	December 3, 2003	J. Franke et al.
RSA-640	$20,000	Factored	November 2, 2005	F. Bahr et al.
RSA-704	$30,000	Not Factored		
RSA-768	$50,000	Not Factored		
RSA-896	$75,000	Not Factored		
RSA-1024	$100,000	Not Factored		
RSA-1536	$150,000	Not Factored		
RSA-2048	$200,000	Not Factored		

- RSA-576
- Prize: $10,000
- Status: Factored
- Decimal Digits: 174
- 18819881292060796383869723946165043980716356337941
- 73827007633564229888597152346654853190606065047430
- 45317388011303396716199692321205734031879550656996
- 221305168759307650257059
- Digit Sum: 785

- RSA-640
- Prize: $20,000
- Status: Factored
- Decimal Digits: 193
- 31074182404900437213507500358885679300373460228427
- 27545720161948823206440518081504556346829671723286
- 78243791627283803341547107310850191954852900733772
- 4822783525742386454014691736602477652346609
- Digit Sum: 806

- RSA-704
- Prize: $30,000
- Status: Not Factored
- Decimal Digits: 212
- 74037563479561712828046796097429573142593188889231
- 28908493623263897276503402826627689199641962511784
- 39958943305021275853701189680982867331732731089309
- 00552505116877063299072396380786710086096962537934
- 650563796359
- Decimal Digit Sum: 1009
- Download Text
- RSA-768
- Prize: $50,000
- Status: Not Factored
- Decimal Digits: 232
- 12301866845301177551304949583849627207728535695953
- 34792197322452151726400507263657518745202199786469
- 38995647494277406384592519255732630345373154826850
- 79170261221429134616704292143116022212404792747377
- 94080665351419597459856902143413
- Decimal Digit Sum: 1018
- Download Text
- RSA-896
- Prize: $75,000
- Status: Not Factored
- Decimal Digits: 270
- 41202343698665954385553136533257594817981169984432
- 79828454556264338764455652484261980988704231618418
- 79261420247188869492560931776375033421130982397485
- 15094490910691026986103186270411488086697056490290
- 36536588674337317208131041051908642547932826013912
- 57624033946373269391
- Decimal Digit Sum: 1222
- Download Text


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Signature hash algorithm
sha1
Issuer
10688435, Starfield Secure Ce...
Valid from
Wednesday, May 25, 2011 12...
Valid to
Sunday, May 25, 2014 12:00:...
Subject
*.kmitl.ac.th, Domain Control ...
Public key
RSA (2048 Bits)
Enhanced Key Usage
Server Authentication (1.3.6....
Edit Properties...
Copy to File...
Learn more about certificate details
OK

A weakness in RSA I

- In RSA not all the messages are concealed, i.e. the plaintext and ciphertext are the same.
- Example: n=35=5*7, m=4*6.
  - P=8.
  - $C=8^5 \bmod 35=8$
  - P=C !!!!!!
- For any key, at least 9 messages will not be concealed. But for $n \geq 200$ or so, this is not important.
- However if *e* is poorly chosen, less than 50% of the messages are concealed.
- It is less likely that a system will have this problem if the primes are safe.
  - A prime is safe if p=2q+1, where q is a prime itself.

Poorly chosen RSA Key

- Example: n=35, e=17.

  {1,6,7,8,13,14,15,20,21,22,27,28,29,34} are not concealed. ☹

  Prove it by yourself !!!

Assessing the security of RSA I

- The security of the private component of a key depends on the difficulty of factoring large integers.
- Justification:

Let Z=(e,n). If n can be factorised then an attacker can find

$$n=pq$$

$$m=(p-1)(q-1)$$

... and use the gcd algorithm to find the private key,

$$d=e^{-1} \bmod m.$$

- No efficient algorithm for factoring *is known*.
- So knowing n = pq does not give the knowledge of p or q.
- This implies that m=(p-1)(q-1) cannot be found and d cannot be computed.

Assessing the security of RSA II

Finding secret exponent	• If an attacker knows X and $Y = D_z(X))$ to find d they must solve • $X = Y^d \bmod n$ or $d = \log_Y X$
Finding plaintext:	• If the attacker knows Y and e to determine X they must take roots modulo a composite number: i.e. they need to solve $Y = X^e$.

!!!!It is important to note that the security of RSA is not provably equivalent to the difficulty of factoring. That is, it might be possible to break RSA without factoring n.

Important attacks

- It is sufficient for the cryptanalyst to compute $\phi(n)$.
- Knowledge of n and $\phi(n)$ gives two equations:

$$n = pq \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$\phi(n) = (p-1)(q-1) \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

This system can be solved, for p say, by solving a quadratic equation:

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

Weak implementations: Common Modulus Attack

Some implementations allow attacks:

Consider a group of users whose public keys consists of the same modulus(n) and different exponents (e_i)

If an intruder intercept two cryptograms where

- They are encryptions of the same message with different keys.
- The two encryption exponents do not have any common factor.

Then the attacker can find the plaintext.!!!

Common Modular attack illustration

- Let us consider why:
- The enemy knows
  - e_1, e_2, N
  - Y_1 and Y_2,

    $Y_1 = X^{e_1} \bmod N$ and $Y_2 = X^{e_2} \bmod N$
- Since e_1 and e_2 are relatively prime, the Extended Euclidean algorithm can be used to find a and b such that $ae_1 + be_2 = 1$.

$$\boxed{ae_1 + be_2 = 1.}$$

But then

- Using a common modulus among a number of participants is not advisable!

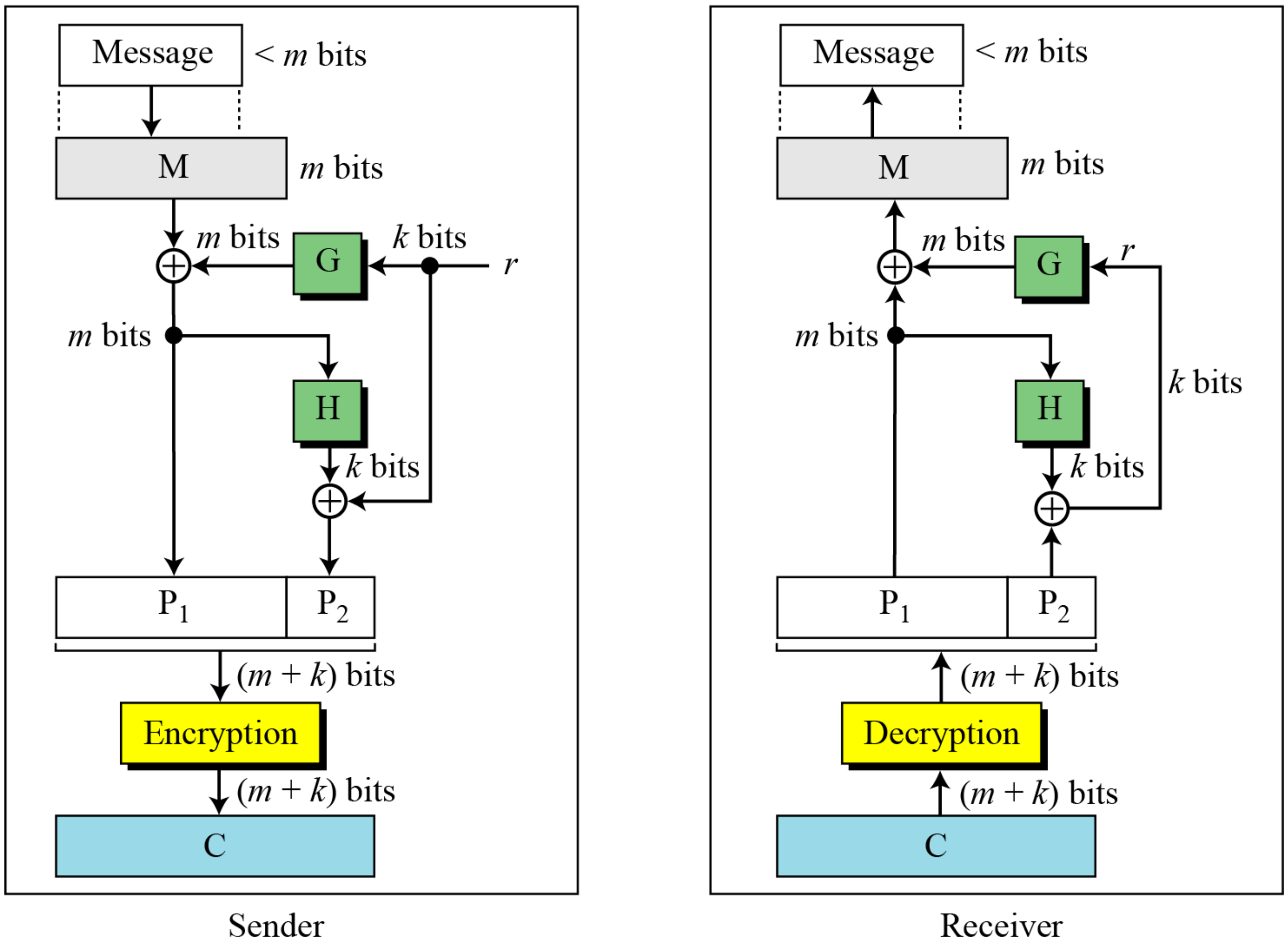
$$\boxed{Y_1^{\,a} Y_2^{\,b} = X^{ae_1} * X^{be_2} = X^{ae_1 + be_2} = X}$$

Short Message Attack

- If Eve knows the set of possible plaintexts,
  - She has one more piece of information apart from the ciphertext.
  - She can encrypt all of the possible messages until the result is the same as the ciphertext intercepted.
  - Example
    - If Alice is sending a four-digit number to Bob.
    - Eve can easily try plaintext numbers from 0000-9999 to find the plaintext.
- Hence:
  - Short message must be padded with random bits at the front and the end to avoid this attack.
- It is strongly recommended that messages be padded with random bits before encryption using a method call OAEP.

OAEP – Optimal asymmetric encryption padding [2]

M: Padded message
r: One-time random number
P: Plaintext ($P_1 \| P_2$)
C: Ciphertext
G: Public function (k-bit to m-bit)
H: Public function (m-bit to k-bit)



สรุป RSA ใช้ *Two Algebraic Structures [2]*

Encryption/Decryption Ring: $R = \langle Z_n, +, \times \rangle$

Key-Generation Group: $G = \langle Z_{\phi(n)}^*, \times \rangle$

RSA uses two algebraic structures:
a public ring $R = \langle Z_n, +, \times \rangle$ and a private group $G = \langle Z_{\phi(n)}^*, \times \rangle$.

In RSA, the tuple (e, n) is the public key; the integer d is the private key.

Rabin Cryptosystem

Rabin Cryptosystem I

- The security of this system is equivalent to the difficulty of factoring. (RSA isn't)
- Key generation:
  - Bob randomly chooses two large primes,
    - p and q, and
    - calculates N=pq.
  - The public key is N and
    - The secret key is (p,q).

Rabin Cryptosystem II

- To Encrypt:
  - the ciphertext Y for a message (plaintext) X:

$$Y=X^2 \bmod N$$

Note this is RSA with e=2

To Decrypt

- Bob must find the square root of Y mod N.
- Knowing (p,q) it is easy to find the square root, otherwise it is provably as difficult as factoring.

Rabin Cryptosytem:Special Case

- Special case: p and q are congruence to 3 mod 4.
  - We construct four intermediate factors.

$$x_1 = Y^{(p+1)/4} \bmod p$$

$$x_2 = p - x_1$$

$$x_3 = Y^{(q+1)/4} \bmod q$$

$$x_4 = q - x_3$$

Rabin Cryptosytem

- We define

  $a=q(q^{-1} \bmod p)$ $b=p(p^{-1} \bmod q)$

- This means, for example, that you calculate q^{-1} mod p and multiply the result by q.
- Then 4 possible plaintexts can be calculated.

$$X_1 = (ax_1+bx_3) \bmod N$$
$$X_2 = (ax_1+bx_4) \bmod N$$
$$X_3 = (ax_2+bx_3) \bmod N$$
$$X_4 = (ax_2+bx_4) \bmod N$$

Rabin Cryptosytem Example I

- We choose p=7, q=11 so N=77.
  - Note that p and q are 3 mod 4.

  ทั้ง p และ q เมื่อมา mod 4 ได้ 3
- Bob's public key is 77,
- and his private key is (7,11).
- To encrypt X=3 Alice calculates

  $Y=3^2 \bmod N = 9 \bmod 77.$
- To decrypt Bob calculates

  $x_1=9^2 \bmod 7 = 4$ $x_2=7-4=3$

  $x_3=9^3 \bmod 11 = 3$ $x_4=11-3=8$

$x1=Y^{(p+1)/4} \bmod p$

$x2=p-x1$

$x3=Y^{(q+1)/4} \bmod q$

$x4=q-x3$

You can calculate $9^3 \bmod 11$ as $(-2)^3 \bmod 11=-8 \bmod 11=3$.

Rabin Cryptosytem Example II

- Bob then finds a and b:
  - $a=q(q^{-1} \bmod p)$
    - $7(7^{-1} \bmod 11) = 7 \times 8 = 56$
  - $b=p(p^{-1} \bmod q)$
    - $11(11^{-1} \bmod 7) = 11\times 2 = 22$

$X1 = (ax1+bx3) \bmod N$

$X2 = (ax1+bx4) \bmod N$

$X3 = (ax2+bx3) \bmod N$

$X4 = (ax2+bx4) \bmod N$

- ... and then the four possible plaintexts.

$X_1= 4\times 22 + 3 \times 56 = 11+14=25 \bmod 77$

$X_2= 4 \times 22 + 8 \times 56 = 11+(-14)= -3 =74 \bmod 77$

$X_3= 3 \times 22 + 3 \times 56 = -11+14 = 3 \bmod 77$

$X_4= 3 \times 22 + 8 \times 56 = -11-14 = -25 =52 \bmod 77$

- แล้วจะรู้ได้อย่างไรว่า ค่าไหนคือ ค่า Plaintext จริงๆ ???

Rabin Cryptosystem: Adv and Dis

- Advantage
  - Provable security.
  - Unless the RSA exponent *e* is small, Rabin's encryption is considerably faster than RSA, requiring one modular exponentiation.
    - Decryption requires roughly the same time as RSA.
- Disadvantage
  - Decryption of a message generates 4 possible plaintexts.
    - The receiver needs to be able to decide which one is the right message. One can append messages with known patterns, for example 20 zeros, to allow easy recognition of the correct plaintext.
- Rabin's system is mainly used for authentication (signatures).

Elgamal Cryptosystem: Outline

- Some symbols
- Generator
- Discrete Logarithm Problem (DLP)
- Elgamal
  - Encryption
  - Decryption

Symbols you should know

- Z_n = { 0,1,2,...,n-1}
  - Ex. Z_{10} = { 0,1,2,...,9}
- Z_p = { 0,1,2,...,p-1}, where p is prime
  - Ex. Z_{11} = { 0,1,2,3,4,5,6,7,8,9,10}
- Z_p^* = {1,2,...,p-1}, where p is prime
  - Ex. Z_{11}^* = {1,2,3,4,5,6,7,8,9,10}

Generator of Z_p^*

- An element α is a generator of Z_p^* if

  $\alpha^{i},\ 0< i \leq p-1$ generates all numbers *1,.. p-1*

- is also called a primitive element
- Finding a generator in general is a hard problem with no efficient algorithm known.

Example

- If 2 is a generater
  - $2^1 = 2 \mod 11$
  - $2^2 = 4 \mod 11$
  - $2^3 = 8 \mod 11$
  - $2^4 = 5 \mod 11$
  - $2^5 = 10 \mod 11$
  - $2^6 = 9 \mod 11$
  - $2^7 = 7 \mod 11$
  - $2^8 = 3 \mod 11$
  - $2^9 = 6 \mod 11$
  - $2^{10} = 1 \mod 11$

2 is a generator since $2^i \mod 11$,

$0 < i \leq p-1 = \{1,2,\ldots,10\} = Z_{11}^*$

Example II

- If 3 is a generator
  - $3^1 = 3 \mod 11$
  - $3^2 = 9 \mod 11$
  - $3^3 = 5 \mod 11$
  - $3^4 = 4 \mod 11$
  - $3^5 = 1 \mod 11$
  - $3^6 = 3 \mod 11$
  - $3^7 = 9 \mod 11$
  - $3^8 = 5 \mod 11$
  - $3^9 = 4 \mod 11$
  - $3^{10} = 1 \mod 11$

3 is NOT a generator since $2^i \mod 11$, $0 < i \le p-1 = \{1,3,4,5,9\}$

Algorithm to find a primitive element

- If factorization of *p-1* is known, it is not hard.
- In particular if $p=2p_1+1$ where p_1 is also a prime we have the following:
  - α in Z_p^* and $\alpha \neq \pm 1 \bmod p$.
  - Then α is a primitive element if and only if

$$\alpha^{\frac{p-1}{2}} \neq 1 \bmod p$$

- Suppose α in Z_p^* and α is not a primitive element.
  - Then $-\alpha$ is a primitive element.
- This gives an efficient algorithm to find a primitive element.

- Example: Z^*_{11}
  - p-1/2 = 5
  - 3^5=1 mod 11
  - 3 is not primitive, -3 = 8 is primitive

	1	2	3	4	5	6	7	8	9	10
1	1	1								
2	**2**	**4**	**8**	**5**	**10**	**9**	**7**	**3**	**6**	**1**
3	3	9	5	4	1	3				
4	4	5	9	3	1	4				
5	5	3	4	9	1	5				
6	**6**	**3**	**7**	**9**	**10**	**5**	**8**	**4**	**2**	**1**
7	**7**	**5**	**2**	**3**	**10**	**4**	**6**	**9**	**8**	**1**
8	**8**	**9**	**6**	**4**	**10**	**3**	**2**	**5**	**7**	**1**
9	9	4	3	5	1	9				
10	10	1	10							

Discrete Logarithm Problem (DLP)

INPUT:

- Z_p^*
- g *in* Z_p^* , g a generator of Z_p^*
- h in Z_p^*

Find the unique number $a<p$ such that $h=g^a \mod p$

DL Assumption: There is no efficient algorithm (polynomial time) to solve DL problem.

- It is widely believed that this assumption holds.

Discrete Logarithm Problem (DLP) II

- When p is small discrete log can be found by exhaustive search
- The size of prime for which DL can be computed is approximately the same as the size of integers that can be factored
  - In 2001: 120 digit DL, 156 digit factorization
- DL is an example of one-way function:
- A function f is one-way if
  - Given x, finding f(x) is easy
  - Given y, finding x such that f(x)=y is hard

Elgamal Key Generation

- Alice chooses
  - a prime p
  - two random numbers g and u, both less than p, where g is a generator of Z_p^*.
- Then she finds:

$$y = g^u \bmod p$$

- Alice's
  - Public key is (p, g, y),
  - her Secret key is u.

Elgamal: Encryption and Decryption

- To encrypt a message X for Alice,

1. Bob chooses a random number k such that gcd(k,p-1)=1.
2. Then he calculates:

$$a = g^k \bmod p$$
$$b = y^k . X \bmod p$$

- The cryptogram is (a,b)
- The length is twice the length of the plaintext

- To decrypt (a,b) Alice calculates

$$X = \frac{b}{a^u} \bmod p$$

Elgamal Cryptosystem Summary

- Public key: $y=g^u \bmod p$, Secret Key: u
- Encryption
  - $a = g^k \bmod p$
  - $b = y^k \times X \bmod p$,
  - ciphertext (a,b)
- Decryption
  - $X = b/a^u$ division use Euclid Extended algorithm.
  - To avoid division
    - $X = b/a^u = b \times a^{p-1-u}$

$$\boxed{a^{p-1} \equiv 1 \bmod p}$$

Example I

- We choose p=11, g=2, u=6.
  - calculate $y=2^6=9 \bmod 11$

    $2^6 \bmod 11 = 64 \bmod 11 = -2 \bmod 11 = 9 \bmod 11$
- The public key is (11,2,9). (p,g,y)

- To encrypt X=6,
  - Bob chooses k=7 and calculates:

    $a=2^7=7 \bmod 11,$

    $b=9^7 \times 6=2 \bmod 11$

    The cryptogram is (7,2).

- To decrypt Alice finds:

  $X=2/7^6 \bmod 11$

  $=2/4=6 \bmod 11$

  $= 2.(4)^{-1} \bmod 11$

  $= 2.3 \Rightarrow 6 \bmod 11$

Security problems with ElGamal

$$Given\ (a,b) = (g^k, y^k \times m) \quad \bmod p$$

$$Eve: \ -chooses\ a\ random\ number\ r\ in\ Z_p^*,$$

$$-computes\ rb$$

$$-sends\ (a, rb)\ to\ Alice$$

$$Alice\ returns\ the\ decryption: \quad r \times m \quad \bmod p$$

$$Eve\ will\ find: \quad \frac{r \times m}{r} = m \quad \bmod p$$

How to define this type of attack?

Provable security

- การที่จะพิสูจน์ว่าระบบมี security อยู่ในระดับไหน Cryptographer ใช้หลักการของ provable security
- Provable security ต่างจากการวัด security ที่นักศึกษารู้จักในที่ว่า Provable security วัตถุประสงค์ไม่ได้อยู่ที่การหาคีย์แต่อยู่ที่ว่า attacker ไม่สามารถแยกความแตกต่างของ ciphertext สองตัวได้ (มองว่าถ้า attacker สามารถแยกได้แสดงว่าข้อมูลบางอย่างรั่ว)
- Adversary power
  - Eve is polynomially bounded
    - time and memory is bounded by a polynomial in the size of input
    - Cannot exhaustively search
  - Eve has access to ‘oracles’:
    - Can ask queries and receive responses
    - Attacks are modeled as a ‘game’ between an attacker Eve and an innocent user (Oracle)
- Security Goal
  - defined as indistinguishability
  - Adversary cannot distinguish two different ciphertexts

IND-CPA: Indistinguishability under Chosen Plaintext Attack

- A challenger and an Adversary:
- The challenger sets up the system:
  - generates a key pair *PK*, *SK* and publishes *PK*
  - The challenger keeps *SK* secert
- The adversary may perform some encryptions of different messages
- The adversary chooses two plaintexts m_0,m_1 and gives them to the challenger.
- The challenger
  - randomly selects a bit b = {0 , 1}
  - sends the *challenge* ciphertext C = E(PK, m_b) to the adversary.
- The adversary
  - performs more encryption (or other operations).
  - it outputs a guess for the value of b.
  - The encryption system is secure if his success chance in correct guessing is not much better than 1/2

Note...

- Note that the task of Eve is much easier than finding the plaintext: she only needs to find one bit information.

- This means that the ciphertext does not leak any information about the plaintext (Micali and Goldwasser 1984)
  - Also called *Semantic Security*

- This is important because in many applications the attacker has apriori information about the message: one of the two candidate, BUY or SELL, etc

Making ElGamal semantically secure

- Original Elgamal scheme is not semantically secure. Some modifications are required. However, the previous security problem mention earlier is still not solved.

- Z_p^*, $p = 2q+1$, q a prime
- Choose h in Z_p^*, $g = h^2 \bmod p$
- Plaintext is in subgroup of Z_p^*, generated by g
- Stronger notions of security are also defined.
- Encryption algorithms should be proved with respect to the attack model (game).

Hence, Elgamal is only semantically secure.

References

[1] CSCI361 Lecture Notes by A/Prof Willy Susilo, University of Wollongong

[2] Forouzan, Cryptography and Network Security, McGraw Hill, 2008.

[3] Parr and Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer, 2010.

Exercise 1:

- **Somying creates a pair of keys for herself.**
- **She chooses *p* = 397 and *q* = 401.**
- **1. Calculate n = ...**
- **2. Calculate ϕ(n) = ...**
- **3. If she chooses e = 343,**
  - **Show that e is a valid key**
  - **calculate d = ...**
- **4. Show how Somchai can send a message to Somying if**
- **he knows *e* and *n*.**
- **5. Suppose Somchai wants to send the message "NO" to Somying .**
  - **His encoding scheme is as follows:**
    - **He changes each character to a number (from 00 to 25), with each character coded as two digits.**
    - **He then concatenates the two coded characters and gets a four-digit number.**
  - **What is the plaintext? What is the cipher text?**

Exercise 2 :Elgamal Cryptosystem

- Let p=43
- Alice chooses g =3, u = 7
  - What is her public key?
- If Bob wants to send his plaintext X= 14 to Alice.
- Show how he computes the ciphertext (encryption process) (let k =26)
- Show how Alice computes the plaintext