Public Key Cryptography

FALGUNI ROY
INSTITUTE OF INFORMATION TECHNOLOGY
NSTU

Private-Key Cryptography

- Traditional **private/secret/single key** cryptography uses **one** key
- Key is shared by both sender and receiver
- If the key is disclosed communications are compromised
- Also known as **symmetric**, both parties are equal
  - Hence does not protect sender from receiver forging a message & claiming is sent by sender

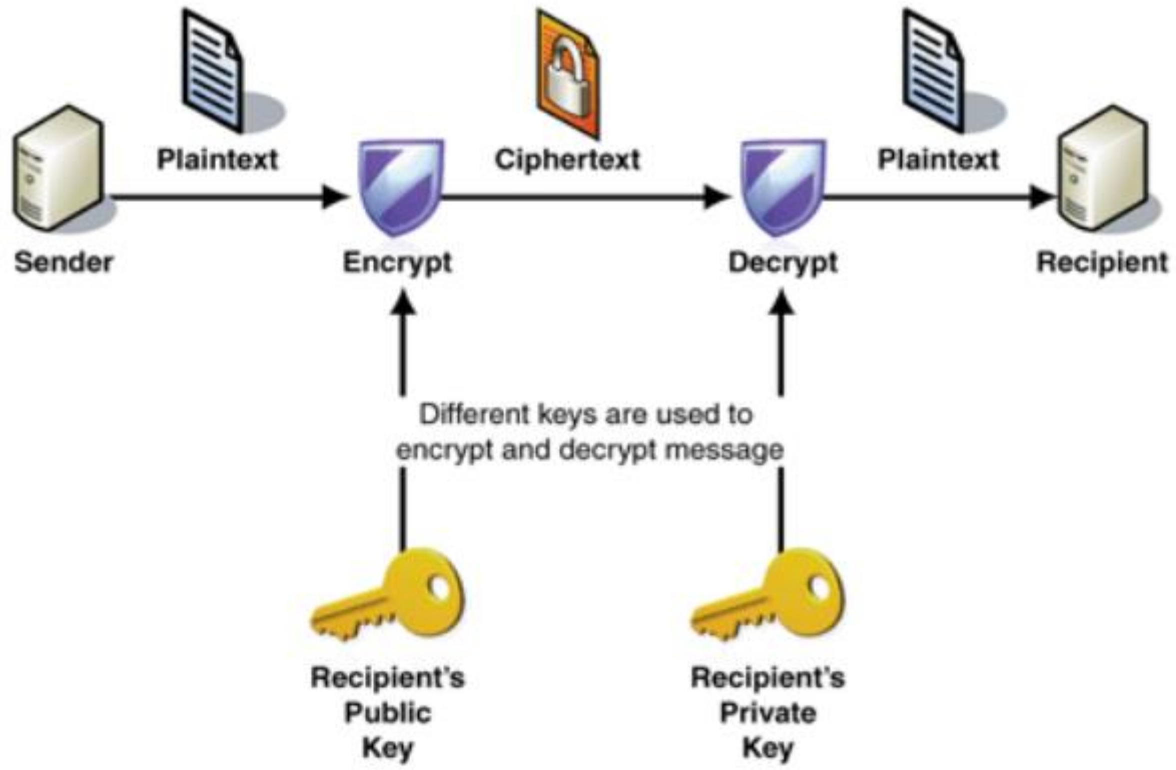
Public-Key Cryptography

- Probably most significant advance in the 3000 year history of cryptography
- Uses **two** keys – a public key and a private key
- **Asymmetric** since parties are **not** equal
- Uses clever application of number theory concepts to function
- Complements **rather than** replaces private key cryptography
- In 1976 Diffie and Hellman Introduced public-key cryptography
- Based on mathematical calculation

Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- **Asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptography



Public-Key Cryptosystems

- A public-key encryption scheme has six ingredients:

Plaintext	Encryption algorithm	Public key	Private key	Ciphertext	Decryption algorithm
The readable message or data that is fed into the algorithm as input	Performs various transform-ations on the plaintext	Used for encryption or decryption	Used for encryption or decryption	The scrambled message produced as output	Accepts the ciphertext and the matching key and produces the original plaintext

Principles of Public-Key Cryptosystems

- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

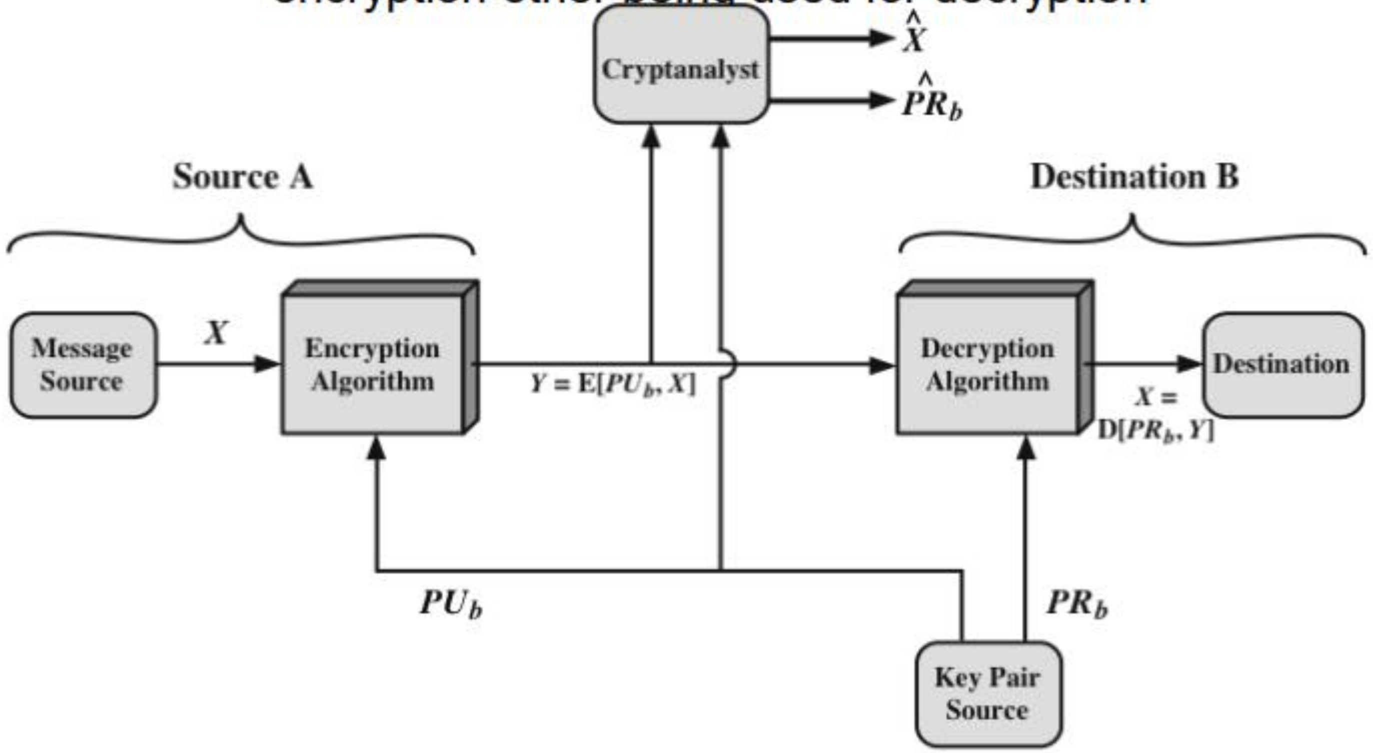
  - **Key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **Digital signatures** – how to verify a message comes intact from the claimed sender

Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key Cryptosystem: encryption using public key -Secrecy

This figure provides confidentiality because two related key used for encryption other being used for decryption



CONFIDENTIALITY

- protecting the information from disclosure to unauthorized parties.

- ensures that only the right people (people who knows the key) can read the information.

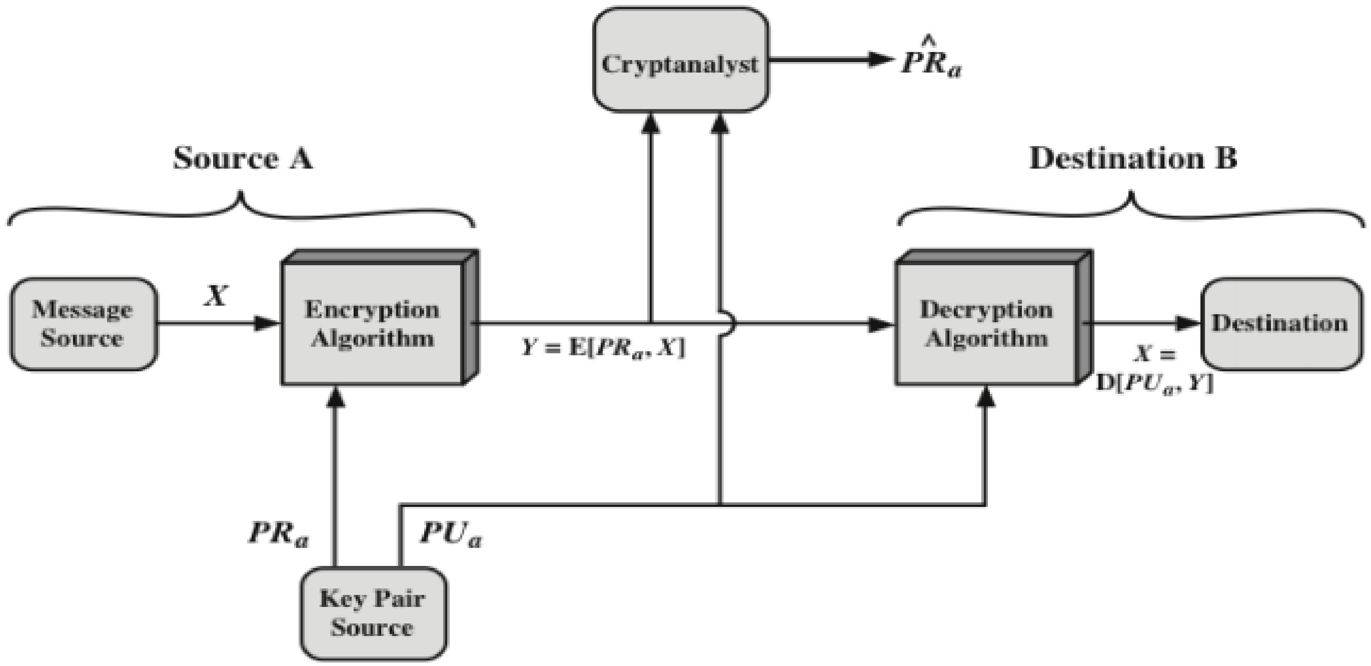
How PKC works when Confidentiality is needed

1. Each system generates a pair of keys.
2. Each system publishes its public key keeping its companion key private.
3. If source A wishes to send a message to source B then it encrypts the message using source B’s public key.
4. When source B receives the message, it decrypts the message using its private key. No one else can decrypt the message because only source B knows its private key.

Public-Key Cryptosystem:

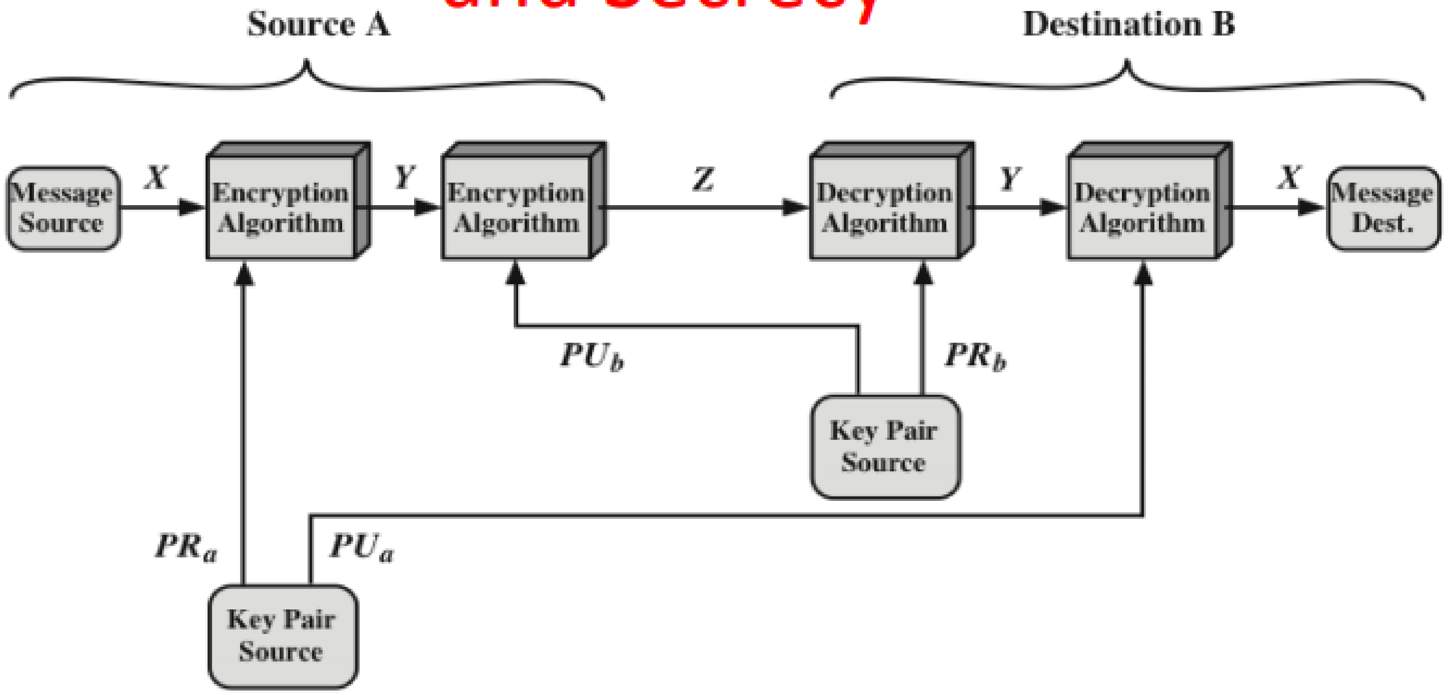
Encryption using private key -Authentication

There is no protection of confidentiality because any observer can decrypt the message by using the sender's public key



- The sender and receiver can confirm each other's identity and origin/destination of the information

Public-Key Cryptosystem: Authentication and Secrecy



we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided

DIFFERENCE BETWEEN SYMMETRIC AND ASYMMETRIC KEY CRYPTOGRAPHY

Characteristic	Symmetric key cryptography	Asymmetric key cryptography
Key used for encryption/decryption	Same key is used	One key is used for encryption and another ;different key is used for decryption
Speed of encryption/decryption	Very fast	Slower
Size of resulting encrypted text	Usually same as or less than the original plain text size.	More than the original plain text size
Known keys	Both parties should know the key in symmetric key encryption	Only, either one of the keys is known by the two parties in public key encryption.
Usage	Confidentiality	Confidentiality, digital signature etc.

Public-Key Applications

- Can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- But keys used are too large (>512bits)
- Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, its just made too hard to do in practise
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

Public-Key Requirements

1. It is computationally easy for a party B to generate a pair (public key PUb, private key PRb)
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message
4. It is computationally infeasible for an adversary, knowing the public key, to determine the private key.
5. It is computationally infeasible for an adversary, knowing the public key and a ciphertext, to recover the original message.
6. The two keys can be applied in either order.

Public-Key Requirements

trap-door one-way function

- A **trapdoor function** is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction (finding its inverse) without special information, called the "trapdoor". Trapdoor functions are widely used in cryptography.
  - $Y = f(X)$ easy
  - $X = f^{-1}(Y)$ infeasible

Public-Key Requirements

A trap-door one-way function is a family of invertible functions f_k, such that

$Y = f_k(X)$ easy, if k and X are known
$X = f_k^{-1}(Y)$ easy, if k and Y are known
$X = f_k^{-1}(Y)$ infeasible, if Y known but k not known

A practical public-key scheme depends on a suitable trap-door one-way function

Knapsack cryptosystems

- Knapsack Cryptosystems are cryptosystems which security is based on the hardness of solving the knapsack problem
  - a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
- **Knapsack Encryption Algorithm** is the first general public key cryptography algorithm.
- It is developed by **Ralph Merkle** and **Mertin Hellman** in 1978.
- As it is a Public key cryptography, it needs two different keys.
- One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process.
- In this algorithm, two different knapsack problems is used in which one is easy and other one is hard.
- The easy knapsack is used as the private key and the hard knapsack is used as the public key.
- The easy knapsack is used to derived the hard knapsack.

Knapsack cryptosystems

- For the easy knapsack, choose a **Super Increasing knapsack problem**.
- Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.
- {1, 2, 4, 10, 20, 40} is a super increasing as
- $1<2$, $1+2<4$, $1+2+4<10$, $1+2+4+10<20$ and $1+2+4+10+20<40$.
- Step-1: Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.
- Step-2:
  - Choose two numbers n and m.
  - The value of m must be greater then the sum of all values in private key, for example 110.
  - And the number n should have no common factor with m, for example 31.
  - Multiply all the values of private key by the number n and then find modulo m.
- Step-3: Calculate the values of Public key using m and n.

Knapsack cryptosystems

- {1, 2, 4, 10, 20, 40} is the private key.
- 1x31 mod(110) = 31
- 2x31 mod(110) = 62
- 4x31 mod(110) = 14
- 10x31 mod(110) = 90
- 20x31 mod(110) = 70
- 40x31 mod(110) = 30
- {31, 62, 14, 90, 70, 30} is the public key

Example

- plain text is 100100111100101110.
- **Encryption**
  - knapsacks contain six values, so split the plain text in a groups of six
  - 100100 111100 101110
  - Multiply each values of public key with the corresponding values of each group and take their sum.

  - 100100 {31, 62, 14, 90, 70, 30}
  - 1x31+0x62+0x14+1x90+0x70+0x30 = 121

  - 111100 {31, 62, 14, 90, 70, 30}
  - 1x31+1x62+1x14+1x90+0x70+0x30 = 197

  - 101110 {31, 62, 14, 90, 70, 30}
  - 1x31+0x62+1x14+1x90+1x70+0x30 = 205
  - cipher text is 121 197 205.

Example

- **Decryption**
  - The receiver receive the cipher text which has to be decrypt.
  - The receiver also knows the values of m and n.
  - So, first need to find the n^-1, which is multiplicative inverse of n mod m

$$n \times n^{-1} \bmod(m) = 1$$
$$31 \times n^{-1} \bmod(110) = 1$$
$$n^{-1} = 71$$

  - multiply 71 with each block of cipher text take modulo m.
  - 121 x 71 mod(110) = 11
  - make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40}
  - 1+10=11 so make that corresponding bits 1 and others 0 which is 100100.
  - 197 x 71 mod(110) = 17
  - 1+2+4+10=17 = 111100

  - And, 205 x 71 mod(110) = 35
  - 1+4+10+20=35 = 101110
  - 100100111100101110 which is the plain text.

RSA Algorithm

- RSA is the algorithm used by modern computers to encrypt and decrypt messages.
- It is an asymmetric cryptographic algorithm.
- One of the first successful responses to the challenge was Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- best known & widely used public-key scheme
- RSA is the first practical public-key cryptosystems and is widely used for secure data transmission

RSA Algorithm

- Every number has a prime factorization and it is hard to find if the number is very large. The time complexity is also very high. But on the other hand the multiplication is easy and time complexity is also not very high.
  - For example the multiplication of 15 and 30 is 450 but the prime factorization of 450 is not easy and also it takes more time as compare to multiplication
- The RSA algorithm involves three steps:
  - key generation,
  - encryption and
  - decryption.

Mathematics behind RSA

Theorem 1 : Let a, p be two integers that are co-prime to each other. Then, there is only a unique integer $x \in Zp$ satisfying

$$ax \bmod p = b$$

Regardless of the value of *b*.

Corollary(1) : If *a* and *p* are co-prime to each other, then *0, a, 2a, ..., (p - 1)a* are all distinct after modulo *p*.

Mathematics behind RSA

- **Theorem 2(Fermat's Little Theorem) :** if n is a prime, and a is any integer co-prime to n, then a^n
  - – a is divisible by n without remainder

$$a^{n-1} = 1 \pmod{n}$$

Mathematics behind RSA

Eulers Totient Function(ø) : When doing arithmetic modulo **n** then

- The **complete set of residues** is: 0...(n-1)
- **The reduced set of residues** is those numbers (residues) which are relatively prime to n
- Number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**
- If the the number *'n'* is a prime number then the value of **ø(n)=n-1**

- Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n.
- It is written using the Greek letter phi as ϕ(n) or φ(n), and may also be called Euler's phi function.
- In other words, it is the number of integers k in the range 1 ≤ k ≤ n for which the greatest common divisor gcd(n, k) is equal to 1
- For example, a complete residue system modulo 12 is {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}.
- The so-called totatives 1, 5, 7 and 11 are the only integers in this set which are relatively prime to 12,
- the corresponding reduced residue system modulo 12 is {1, 5, 7, 11}.

RSA Algorithm

- Plaintext is encrypted in blocks with each block having a binary value less than some number *n*
- Encryption and decryption are of the following form, for some plaintext block *M* and cipher text block C

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

- Both sender and receiver must know the value of *n*
- The sender knows the value of *e*, and only the receiver knows the value of *d*
- This is a public-key encryption algorithm with a public key of *PU={e,n}* and a private key of *PR={d,n}*

Algorithm Requirements

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M<n$
2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$
3. It is infeasible to determine d given e and n

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus $N=p.q$
  - note $\varnothing(N)=(p-1)(q-1)$
- selecting at random the encryption key e
  - where $1<e<\varnothing(N)$, $\gcd(e,\varnothing(N))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod \varnothing(N)$ and $0\le d\le N$
- publish their public encryption key: KU={e,N}
- keep secret private decryption key: KR={d,p,q}

RSA Use

- to encrypt a message M the sender:
  - obtains **public key** of recipient KU={e,N}
  - computes: $C=M^e \bmod N$, where $0 \leq M < N$
- to decrypt the ciphertext C the owner:
  - uses their private key KR={d,p,q}
  - computes: $M=C^d \bmod N$
- note that the message M must be smaller than the modulus N (block if needed)

Why RSA Works

- because of Euler's Theorem:
- $a^{\varnothing(n)} \bmod N = 1$
  - where $\gcd(a,N)=1$
- in RSA have:
  - $N=p.q$
  - $\varnothing(N)=(p-1)(q-1)$
  - carefully chosen e & d to be inverses $\bmod \varnothing(N)$
  - hence $e.d=1+k.\varnothing(N)$ for some k
- hence :
  $C^d = (M^e)^d = M^{1+k.\varnothing(N)} = M^1.(M^{\varnothing(N)})^q = M^1.(1)^q$
  $= M^1 = M \bmod N$

RSA Example

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\varnothing(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e : gcd(e,160)=1; choose $e=7$
5. Determine d:
   - $de = 1 \bmod 160$ and $d < 160$
   - Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key KU={7,187}
7. Keep secret private key KR={23,17,11}

RSA Example cont

- sample RSA encryption/decryption is:
- given message `M = 88` (nb. `88<187`)
- encryption:

  $C = 88^{7} \bmod 187 = 11$
- decryption:

  $M = 11^{23} \bmod 187 = 88$

Exponentiation

- can use the Square and Multiply Algorithm
- a fast, efficient algorithm for exponentiation
- concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent
- only takes $O(\log_2 n)$ multiples for number n
  - eg. $7^5 = 7^4.7^1 = 3.7 = 10 \mod 11$
  - eg. $3^{129} = 3^{128}.3^1 = 5.3 = 4 \mod 11$

Exponentiation

```
c ← 0; d ← 1
for i ← k downto 0
    do  c ← 2 × c
        d ← (d × d) mod n
        if  b_i = 1
            then  c ← c + 1
                  d ← (d × a) mod n
return d
```

RSA Key Generation

- users of RSA must:
  - determine two primes at random - `p, q`
  - select either `e` or `d` and compute the other
- primes `p,q` must not be easily derived from modulus `N=p.q`
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents `e`, `d` are inverses, so use Inverse algorithm to compute the other

RSA Security

- three approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N)
  - timing attacks (on running of decryption)

Factoring Problem

- mathematical approach takes 3 forms:
  - factor `N=p.q`, hence find `ø(N)` and then d
  - determine `ø(N)` directly and find d
  - find d directly
- currently believe all equivalent to factoring
  - have seen slow improvements over the years
    - as of Aug-99 best is 130 decimal digits (512) bit with GNFS
  - biggest improvement comes from improved algorithm
    - cf “Quadratic Sieve” to “Generalized Number Field Sieve”
  - barring dramatic breakthrough 1024+ bit RSA secure
    - ensure p, q of similar size and matching other constraints

Timing Attacks

- developed in mid-1990’s
- exploit timing variations in operations
  - eg. multiplying by small vs large number
  - or IF's varying which instructions executed
- infer operand size based on time taken
- RSA exploits time taken in exponentiation
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

Summary

- have considered:
  - principles of public-key cryptography
  - RSA algorithm, implementation, security

Presentation Topics : 16 march 2021

- **Proof of RSA, Approaches to attacking RSA, Public Key Infrastructure**
    - Sunaan Sultan, Dhruva kanti

- **Diffie–Hellman, ElGamal encryption with attacks**
    - Ikra Chowdhury, Ishrat Jahan Rintu

- **Elliptic-curve cryptography, Paillier cryptosystem with attacks**
    - Armanur Rashid, Md Alamin