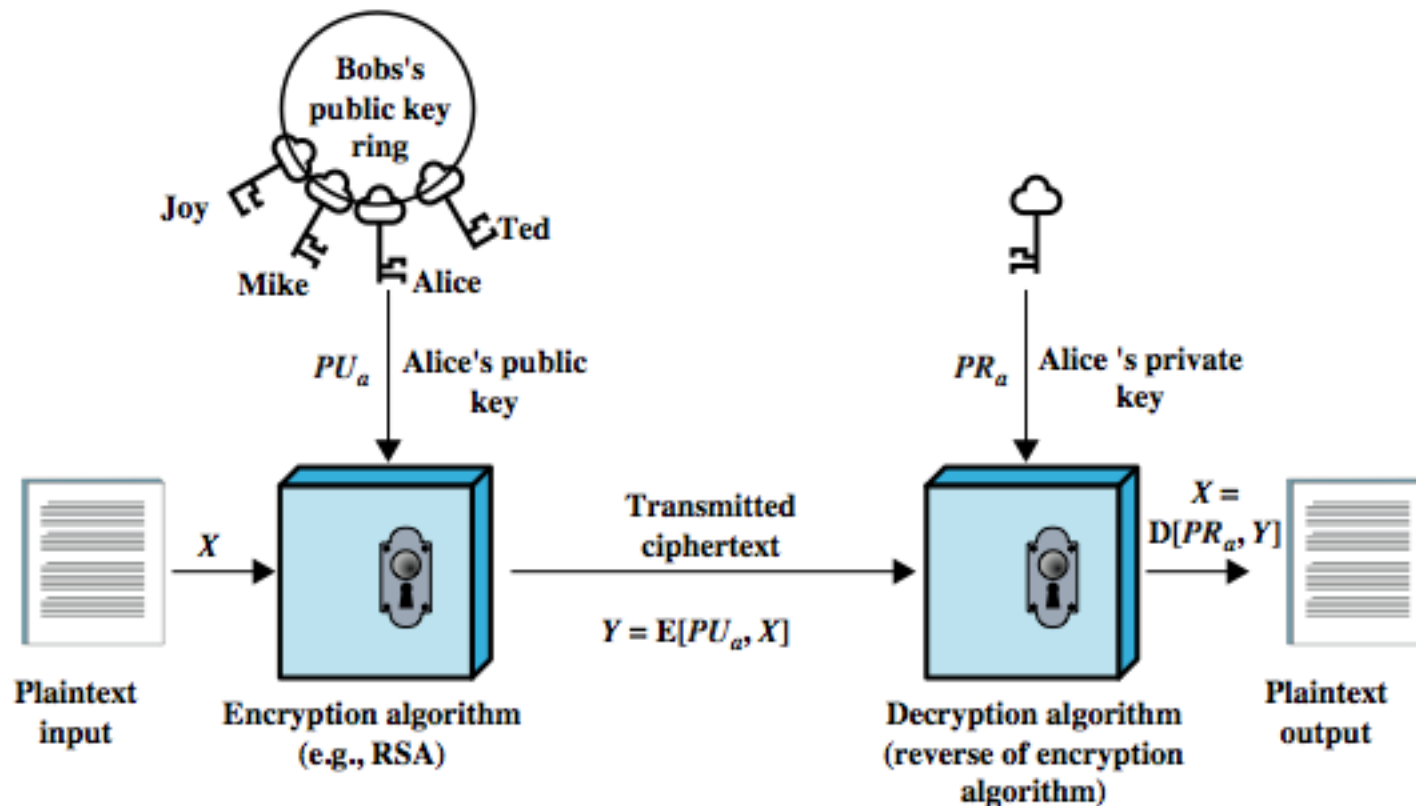


Information Security

CS 3002

Dr. Haroon Mahmood
Assistant Professor
NUCES Lahore

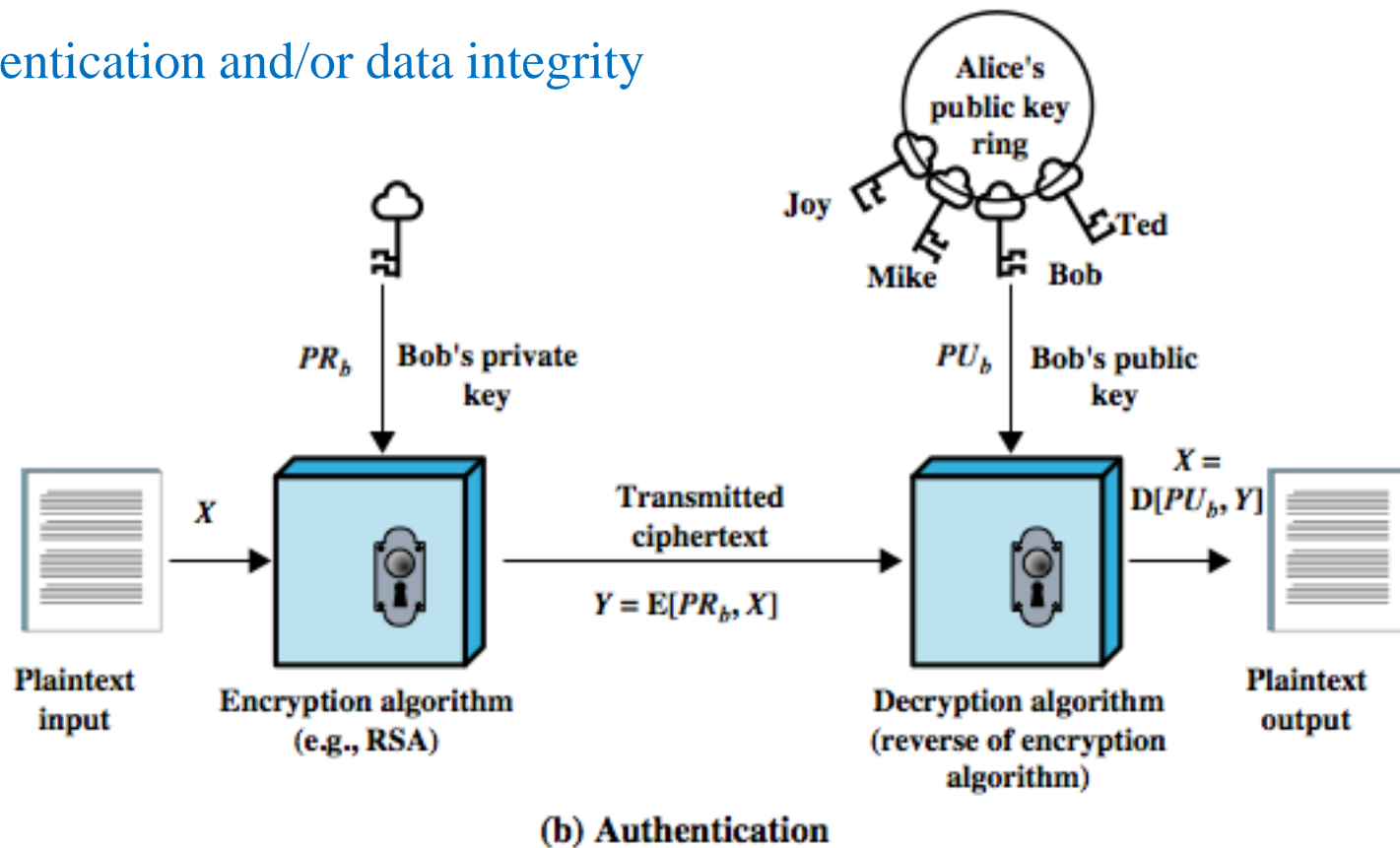
Public Key Encryption



(a) Confidentiality

Public Key Authentication

Authentication and/or data integrity



Public Key Requirements

- 1. computationally easy to create key pairs**
- 2. computationally easy for sender knowing public key to encrypt messages**
- 3. computationally easy for receiver knowing private key to decrypt ciphertext**
- 4. computationally infeasible for opponent to determine private key from public key**
- 5. computationally infeasible for opponent to otherwise recover original message**
- 6. useful if either key can be used for each role**

Public Key Algorithms

- **RSA (Rivest, Shamir, Adleman)**
 - developed in 1977
 - only widely accepted public-key encryption alg
 - given tech advances need 1024+ bit keys
- **Diffie-Hellman key exchange algorithm**
 - only allows exchange of a secret key
- **Digital Signature Standard (DSS)**
 - provides only a digital signature function with SHA-1
- **Elliptic curve cryptography (ECC)**
 - new, security like RSA, but with much smaller keys

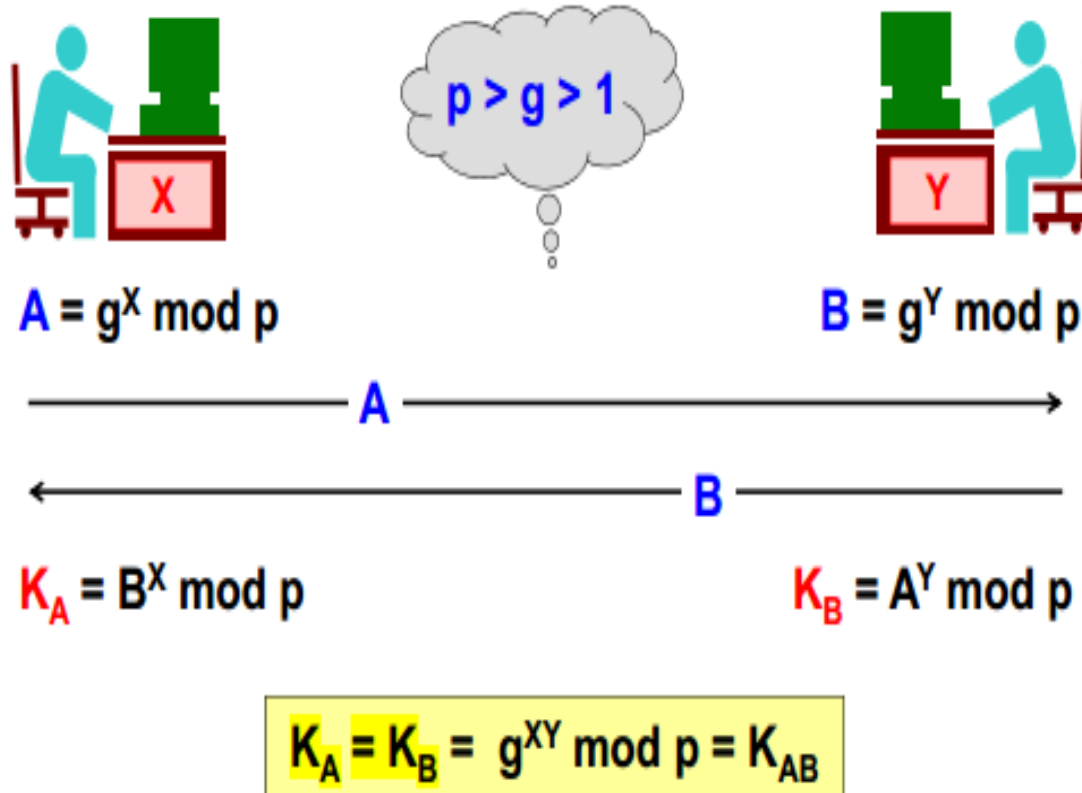
Diffie-Hellman

- First public key algorithm invented
- Published in 1976
- Specific method for securely exchanging cryptographic keys over a public channel
- Concept given by Ralph Merkle
- Named after Whitfield Diffie and Martin Hellman
- **Public key exchange algorithm**, **neither** encryption nor signature

Diffie-Hellman

- Uses modular arithmetic also called the clock arithmetic:
 - $g \bmod p$. where g is the generator and p is the prime modulus
 - $1 < g < p$
- g is a primitive root of p
- Consider two numbers g & p shared publically between A & B
- A computes $X = g^x \bmod p$ (x is the secret from Alice)
- B computes $Y = g^y \bmod p$ (y is the secret from Bob)
- A & B exchanged X & Y
- A computes $K_{AB} = Y^x \bmod p$ and B computes $K_{BA} = X^y \bmod p$
- $K_{AB} = K_{BA} = g^{xy} \bmod p$

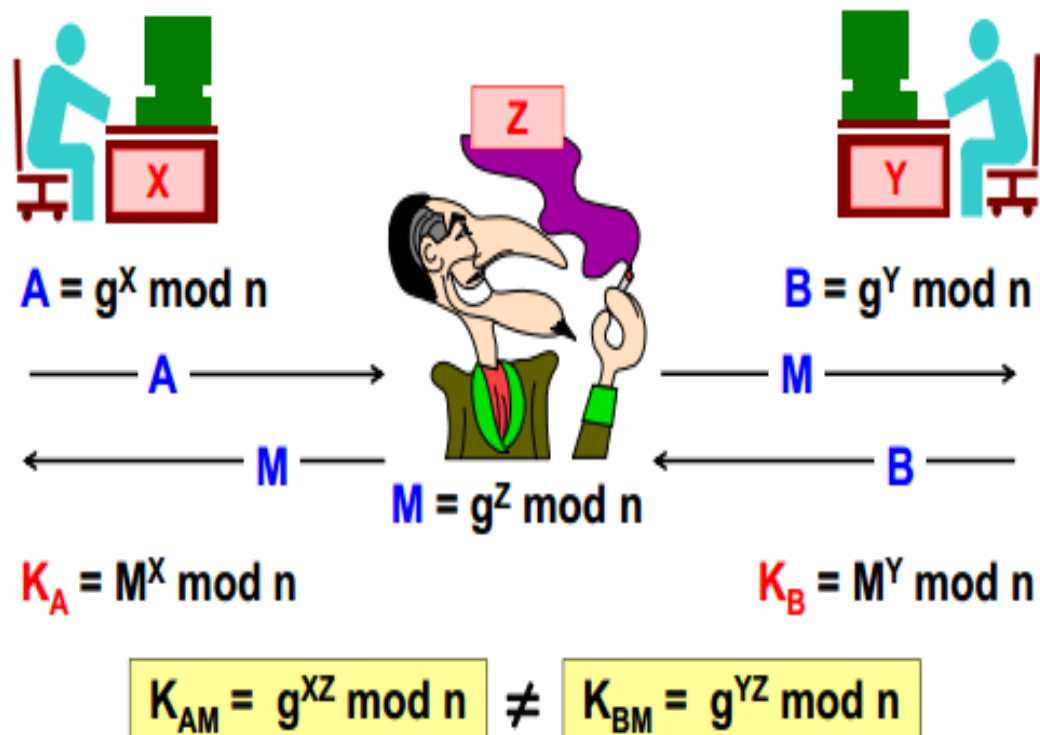
Diffie-Hellman



Diffie-Hellman: example

- E.g: $g = 3, p = 353$
- Alice computes secret $\Rightarrow x = 97$
 $g^x \bmod p = 3^{97} \bmod 353 = 40$
- Bob computes secret $\Rightarrow y = 233$
 $g^y \bmod p = 3^{233} \bmod 353 = 248$
- Alice gets 248 from Bob \Rightarrow
- Alice computes $\Rightarrow 248^{97} \bmod 353 = 160$
- Bob gets 40 from Alice \Rightarrow
- Bob computes $\Rightarrow 40^{233} \bmod 353 = 160$

Man in the middle attack on DH



RSA

- **Invented by Ron Rivest, Adi Shamir, and Len Adleman at MIT 1978**
- **Block size can be variable**
- **Key length can be variable**
- **Plaintext must be smaller than the key length**
- **Ciphertext block will be the length of the key**
- **product of prime numbers, factoring of result**
- **Applications: secrecy(secret exchange of keys) and digital signature**

Co-prime or relatively prime

- Two numbers are co-prime if the greatest common divisor (GCD) between them is only 1.
- A and B are coprime iff $\gcd(A,B) = 1$.
- E.g: 6 and 11. gcd between 6 & 11 is 1 only
- It isn't necessary that the two numbers are prime! They have to be prime to each other!

Euler Totient function

- If n is a positive integer, ϕ (phi) function counts all the positive integers less than n that are relatively prime to n .
- $\phi(n) = \text{Totient}(n) = \text{numbers of integers } < n \text{ that are coprime to } n$
- $\phi(10) = 4$
 - $(1,3,7,9)$ are all coprime to 10
- Generally, $\phi(p) = (p-1)$ where p is a prime number

Multiplicative Inverse

- **Multiplicative Inverse of a number X is Y iff Y multiplied with X yields 1.**
- **i.e. $X * Y = 1$, Both X and Y are the multiplicative inverses of each other.**

PKC – How it works?

- Plain text M and ciphertext C are integers between 0 and $n-1$ (n is product of two prime numbers).
- $\text{Key}_1 = \{e, n\}$, $\text{Key}_2 = \{d, n\}$
 - Where e and d are two secret number
 - $C = M^e \bmod n$
 - $M = C^d \bmod n$

RSA - Key Construction

- Select two large primes: $p, q, p \neq q$
- $n = p \times q$
- Calculate Euler's Totient Function, $\phi(n) = (p-1)(q-1)$
- Select e relatively prime to $\phi \Rightarrow \gcd(\phi, e) = 1; 0 < e < \phi$
- Calculate $d = \text{inverse of } e \text{ mod } \phi \Rightarrow d * e \text{ mod } \phi = 1$

- public key = (e, n)
- private key = (d, n)
- The roles of e & d are interchangeable. i.e. $(x^d)^e \text{ mod } n = (x^e)^d \text{ mod } n$

RSA Key Construction: Example

- Select two large primes: $p, q, p \neq q$
- $p = 17, q = 11$
- $n = p \times q = 17 \times 11 = 187$
- Calculate $\phi = (p-1)(q-1) = 16 \times 10 = 160$
- Select e , such that $\gcd(\phi, e) = 1; 0 < e < \phi$ say, $e = 7$
- Calculate d such that $\rightarrow de \bmod \phi = 1$
- Use Euclid's algorithm to find $d = e^{-1} \bmod \phi$
 - $160k + 1 = 161, 321, 481, 641$
 - Check which of these is divisible by 7
 - 161 is divisible by 7 giving $d = 161/7 = 23$
- $\text{Key}_1 = \text{Public key} = \{7, 187\}, \text{Key}_2 = \text{Private key} = \{23, 187\}$

RSA – encryption: Example

- Messages to encrypt:
- $m_1 = 2$ & $m_2 = 3$
- $\text{Key}_1 = \text{Public key} = \{7, 187\}$, $\text{Key}_2 = \text{Private key} = \{23, 187\}$
- $c_1 = 2^7 \bmod 187 = 128 \bmod 187 = 128$
- $c_2 = 3^7 \bmod 187 = 2187 \bmod 187 = 130$
- $m_1 = 128^{23} \bmod 187 = 2$
- $m_2 = 130^{23} \bmod 187 = 3$