Dr. Ammar Haider
Assistant Professor
School of Computing

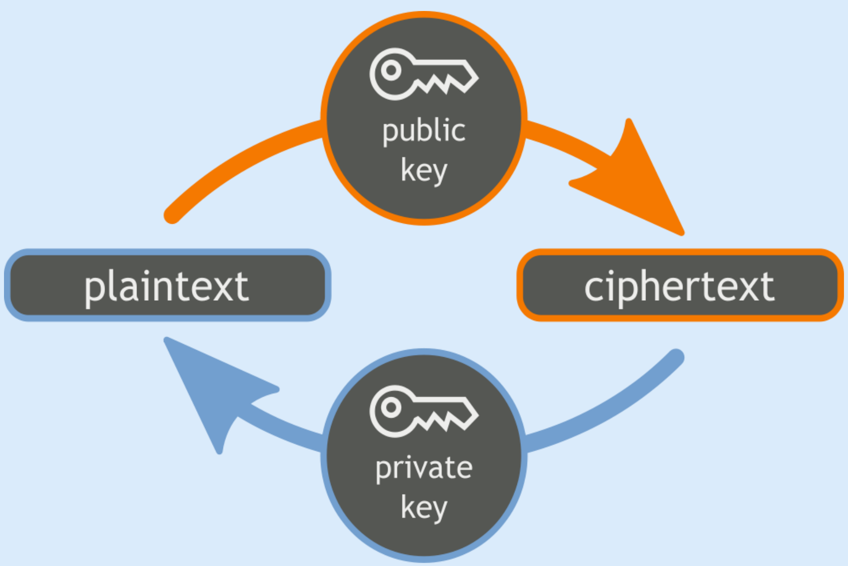
CS3002 Information Security

Asymmetric Cryptography

Reference: Stallings CNS chap 9, 10

Asymmetric Cryptography

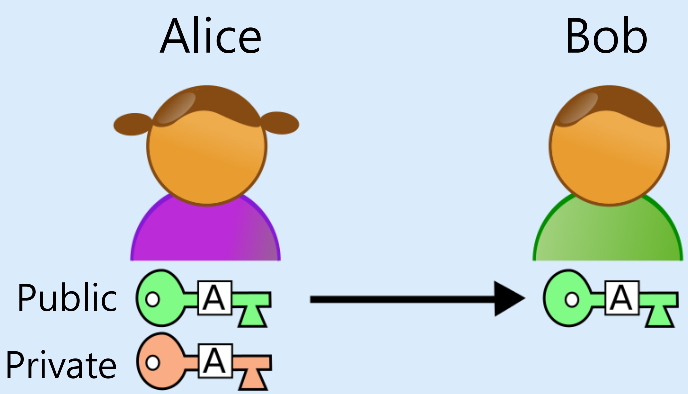
- Newer form of crypto (1970s)
- Uses a pair of keys for each user: one public, one private
- Also known as Public Key Cryptography (PKC)
- The private key can unlock (decrypt) what is locked (encrypted) with the public key and vice versa



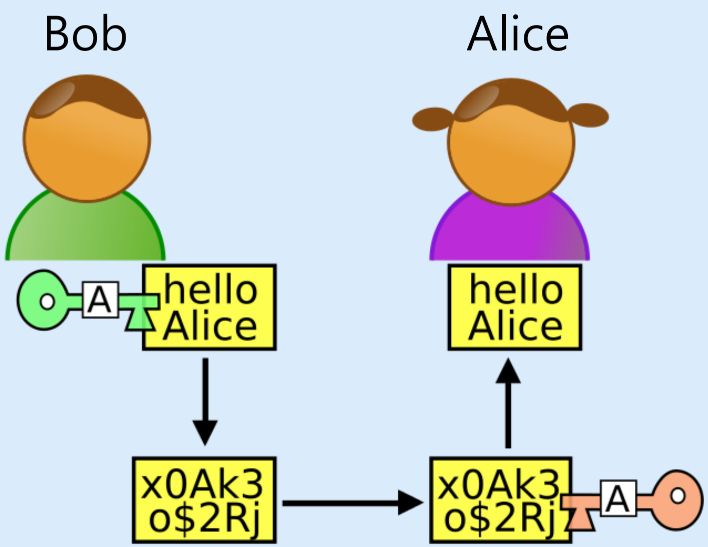
Asymmetric Cryptography

Use Case 1: Data encryption

Aiming for confidentiality



Step1: Announce the public key

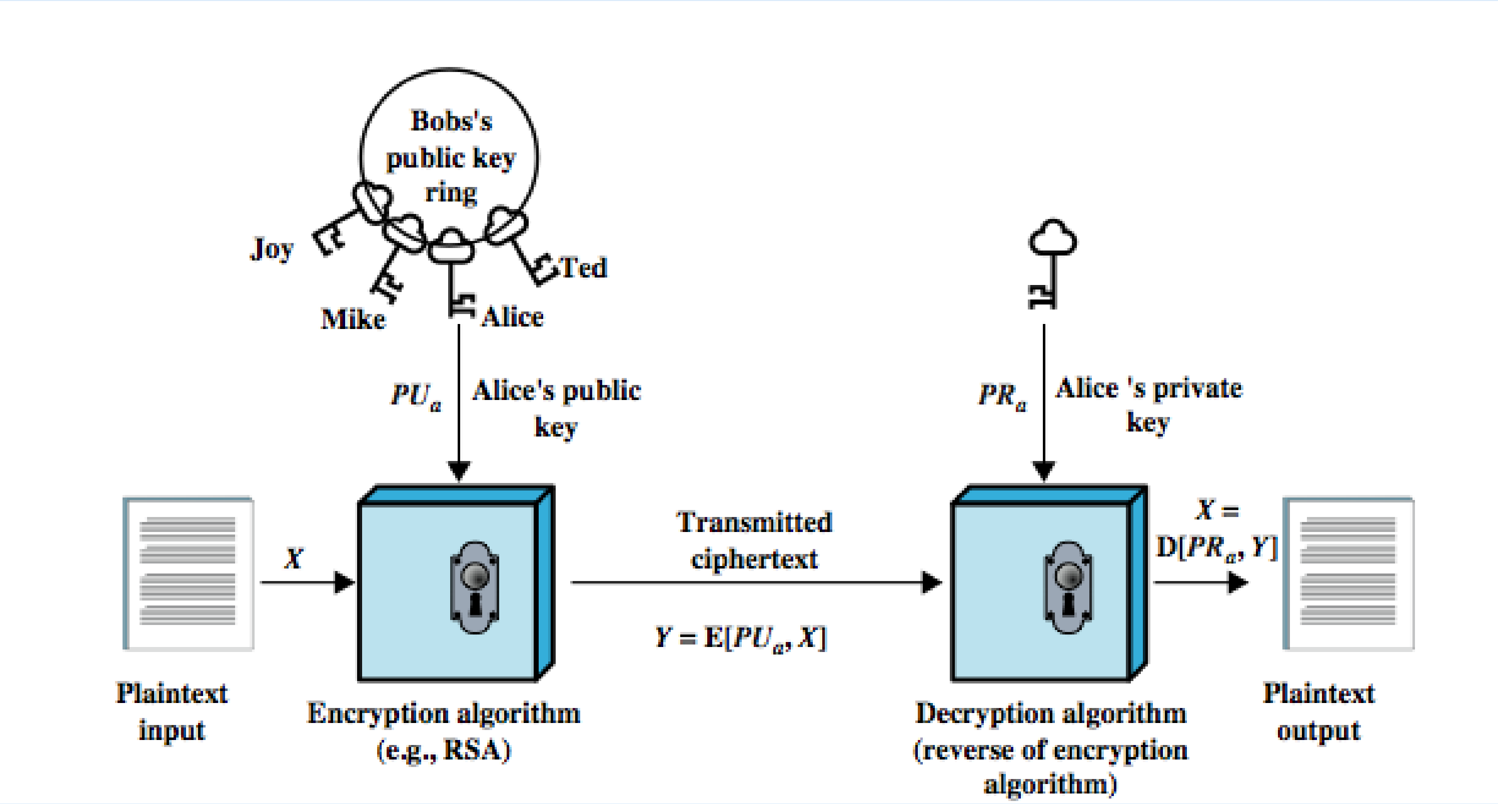


Step2: Encrypt with recipient's public key

Step3: Decrypt with private key

Asymmetric Cryptography

- Confidentiality

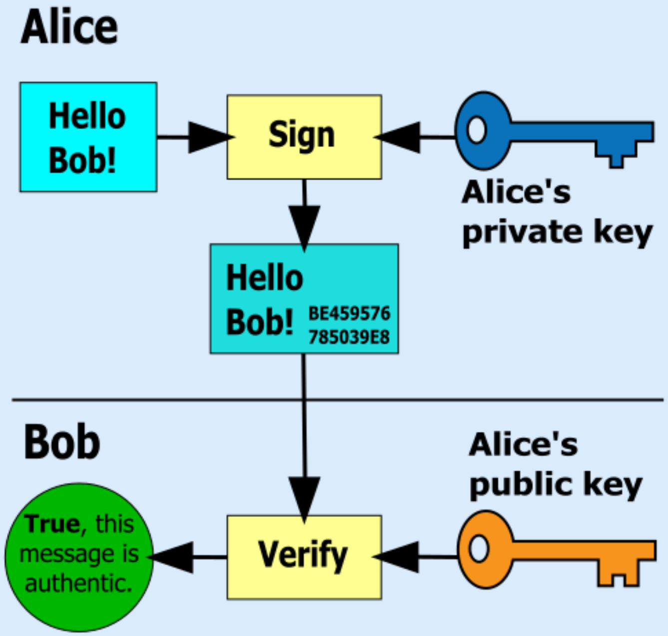


Asymmetric Cryptography

Use Case 2: Message authentication

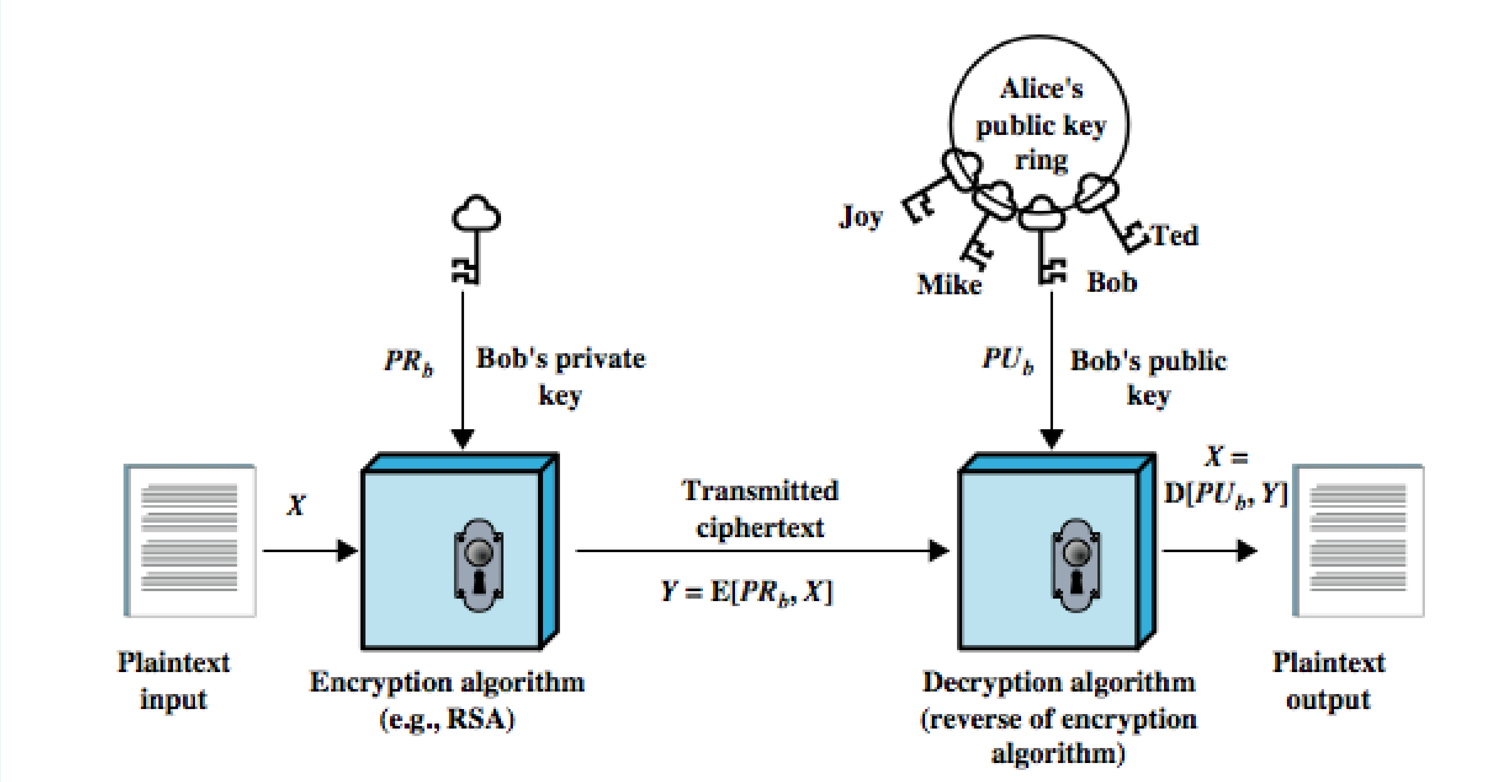
- proves the authenticity and origin of a message.
- Recipient is sure of the origin of the message
- Sender can not deny having sent the message (non-repudiation)

Message Authentication Code (MAC) produced using one's private key is called **Digital Signature**



Asymmetric Cryptography

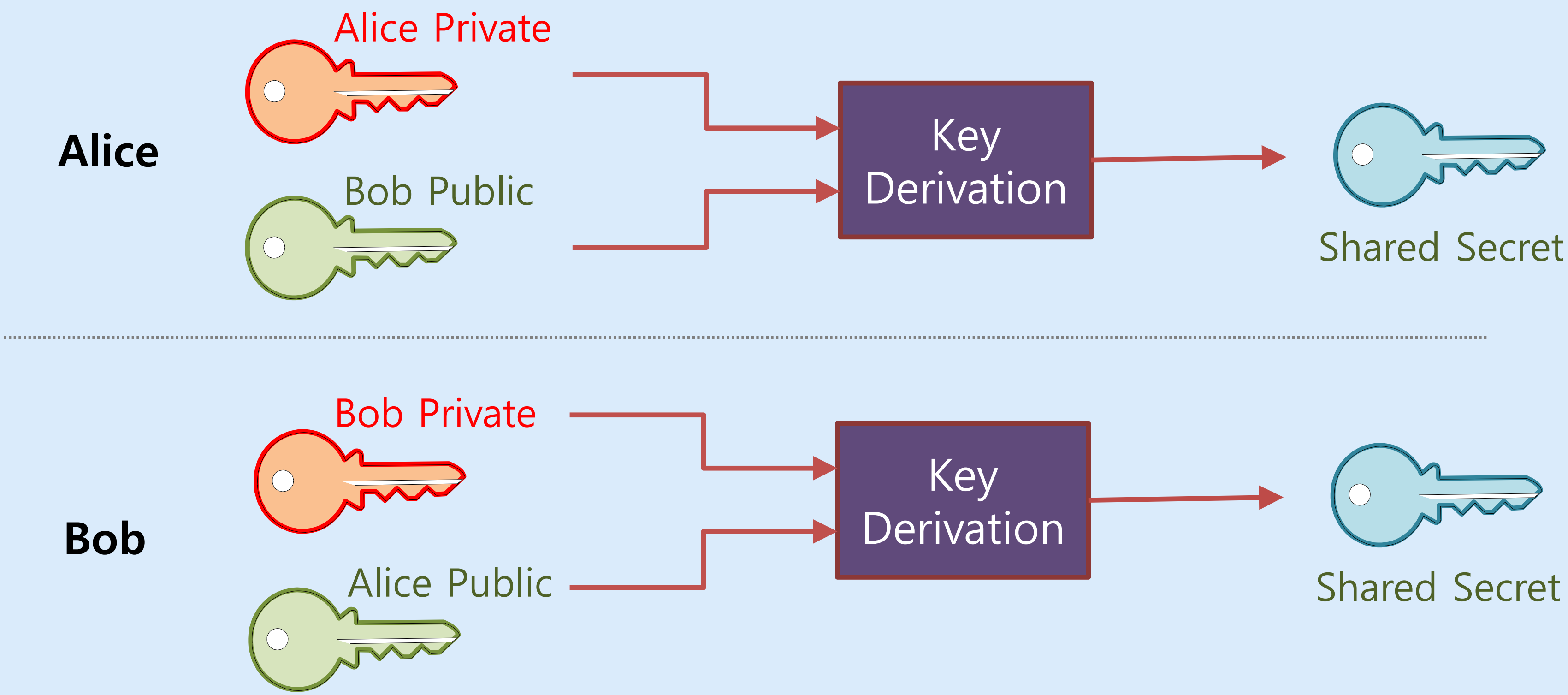
- Authenticity and Data Integrity



Asymmetric Cryptography

Use Case 3: Symmetric key exchange

- PKC can provide a secret channel for sharing a secret key K (to be used for symmetric crypto AES, DES etc.).



PKC 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

PKC Algorithms

- RSA (Rivest, Shamir, Adleman)
  - developed in 1977
  - most widely accepted public-key encryption algo
  - need 2048+ bit keys
- Diffie-Hellman key exchange algorithm
  - only allows exchange of a secret key
- Digital Signature Standard (DSS)
  - provides only a digital signature function using SHA-1 hashes
- Elliptic curve cryptography (ECC)
  - new family of algorithms
  - security like RSA, but with much smaller keys

RSA

- Invented by Ron Rivest, Adi Shamir, and Len Adleman at MIT 1977
- 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
- Uses product of prime numbers, factoring of result
- Applications: secrecy and digital signatures

RSA

Co-prime numbers

- Two numbers are co-prime (also called relatively 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 are coprime because their gcd is 1 only

- It isn't necessary that the two numbers are prime! They just have to be prime to each other!

RSA

Euler’s Totient Function

- If n is a positive integer, φ (phi) function counts all the positive integers less than n that are co-prime to n.
- For example, $\varphi(10) = 4$
  - because 1, 3, 7, 9 are all coprime to 10

- For a **prime number** p, totient function is very straightforward: $\varphi(p) = (p-1)$
- If n is a product of two prime numbers p and q, $\varphi(n) = \varphi(pq) = \varphi(p)\,\varphi(q) = (p-1)(q-1)$
  - e.g. $\varphi(77) = \varphi(7\times11) = 6\times10 = 66$

RSA Algorithm

Key Construction

- Select two large primes: p, q, $p \neq q$.
- Define $n = p \times q$
- Calculate totient function of n, that is $\varphi(n) = (p-1)(q-1)$
- Select e relatively prime to φ, that is, $\gcd(\varphi,e) = 1$; and $1 < e < \varphi$
- Calculate d as the multiplicative inverse of e with mod φ
  $d * e \bmod \varphi = 1$

- public key = (e, n), private key = (d, n)

- The roles of e & d are interchangeable. i.e.

$$(x^d)^e \bmod n = (x^e)^d \bmod n$$

RSA Algorithm

Key Construction Example

- Select two large primes: p, q, p ≠ q. Let p = 17, q = 11
- Define n = p × q = 17 × 11 = 187
- Calculate φ = (p-1)(q-1) = 16 × 10 = 160
- Select e (1<e<φ), such that gcd(φ, e) = 1; say e = 7
- Calculate d such that → d × e mod φ = 1
  - Use Euclid's algorithm to find d = e^{-1} mod φ
  - 160k+1 = 161, 321, 481, 641, ...
  - check which of these is divisible by e
  - 161 is divisible by 7 giving d = 161/7 = 23
- Public key: Key_1 = {7, 187}
- Private key: Key_2 = {23, 187}

RSA Algorithm

Encryption and Decryption

- Plain text M and ciphertext C are integers between 0 and n-1 (n is product of two prime numbers).
- $Key_1 = \{e, n\}$, $Key_2 = \{d, n\}$

- To encrypt and decrypt

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

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

Decryption is the inverse of encryption thanks to Euler's theorem.
https://www.youtube.com/watch?v=UjIPMJd6Xks

RSA Algorithm

Encryption and Decryption Example

- Messages to encrypt: $M_1 = 2$ and $M_2 = 5$
- Public key = {7, 187}
- Private key = {23, 187}

- Encryption
  $C_1 = 2^7 \bmod 187 = 128 \bmod 187 = 128$
  $C_2 = 5^7 \bmod 187 = 78125 \bmod 187 = 146$

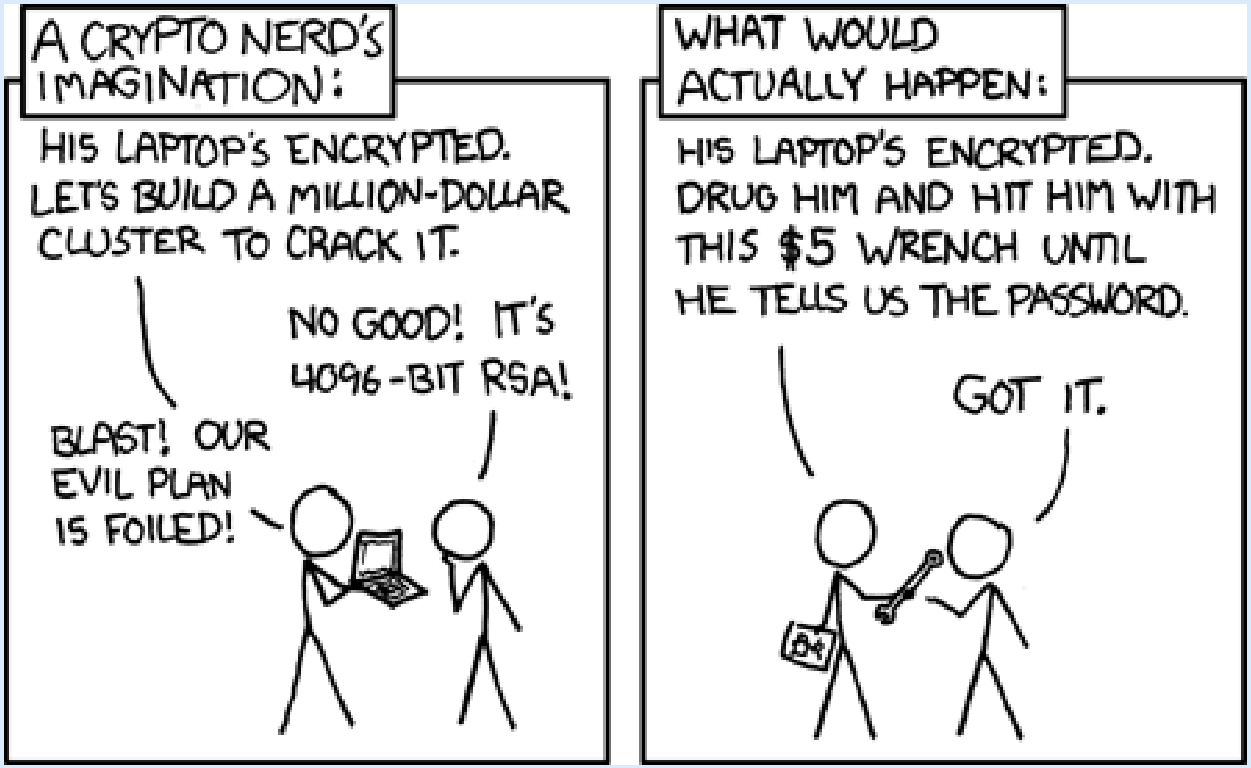
- Decryption
  $M_1 = 128^{23} \bmod 187 = 2$
  $M_2 = 146^{23} \bmod 187 = 5$

RSA Encryption Strength

- Public and private keys are mathematically related, but the relationship is hidden from attackers
- Both keys e and d are derived from p and q. Attacker only knows e and n. To figure out d, they must first find out φ, which requires knowledge of factors of n
  - since $\varphi=(p-1)*(q-1)$ and $n=p*q$
- Factorizing n seems trivial for small numbers, but when n is sufficiently large (i.e. several hundred digits!) decomposing it into factors is computationally infeasible
- That’s why RSA recommended key length (size of n) is at least 2048 bits (which is 617 decimal digits)
  - So individually p and q are 1024 bits each

https://www.mobilefish.com/services/rsa_key_generation/rsa_key_generation.php

RSA Encryption Strength

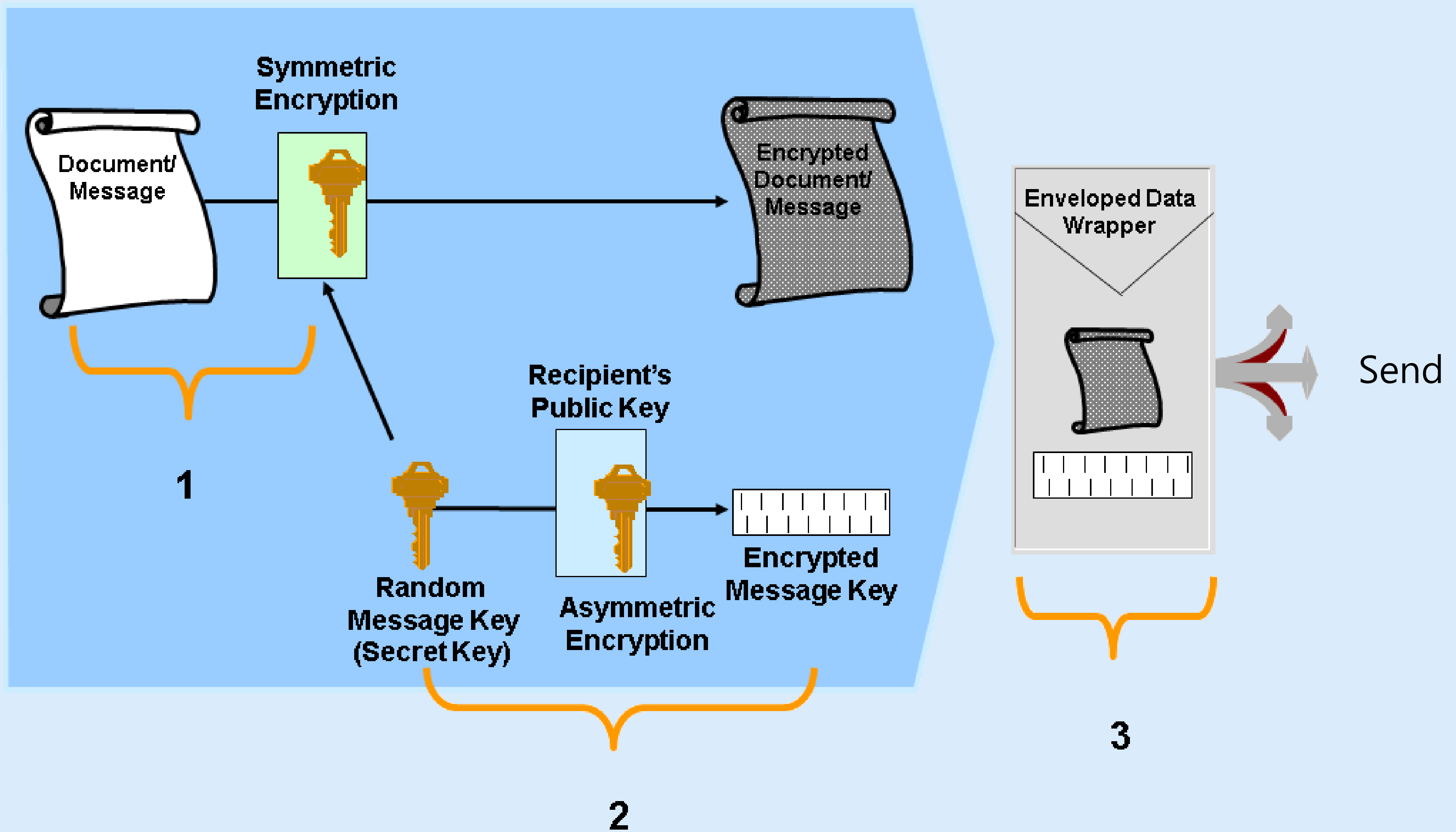


https://xkcd.com/538/

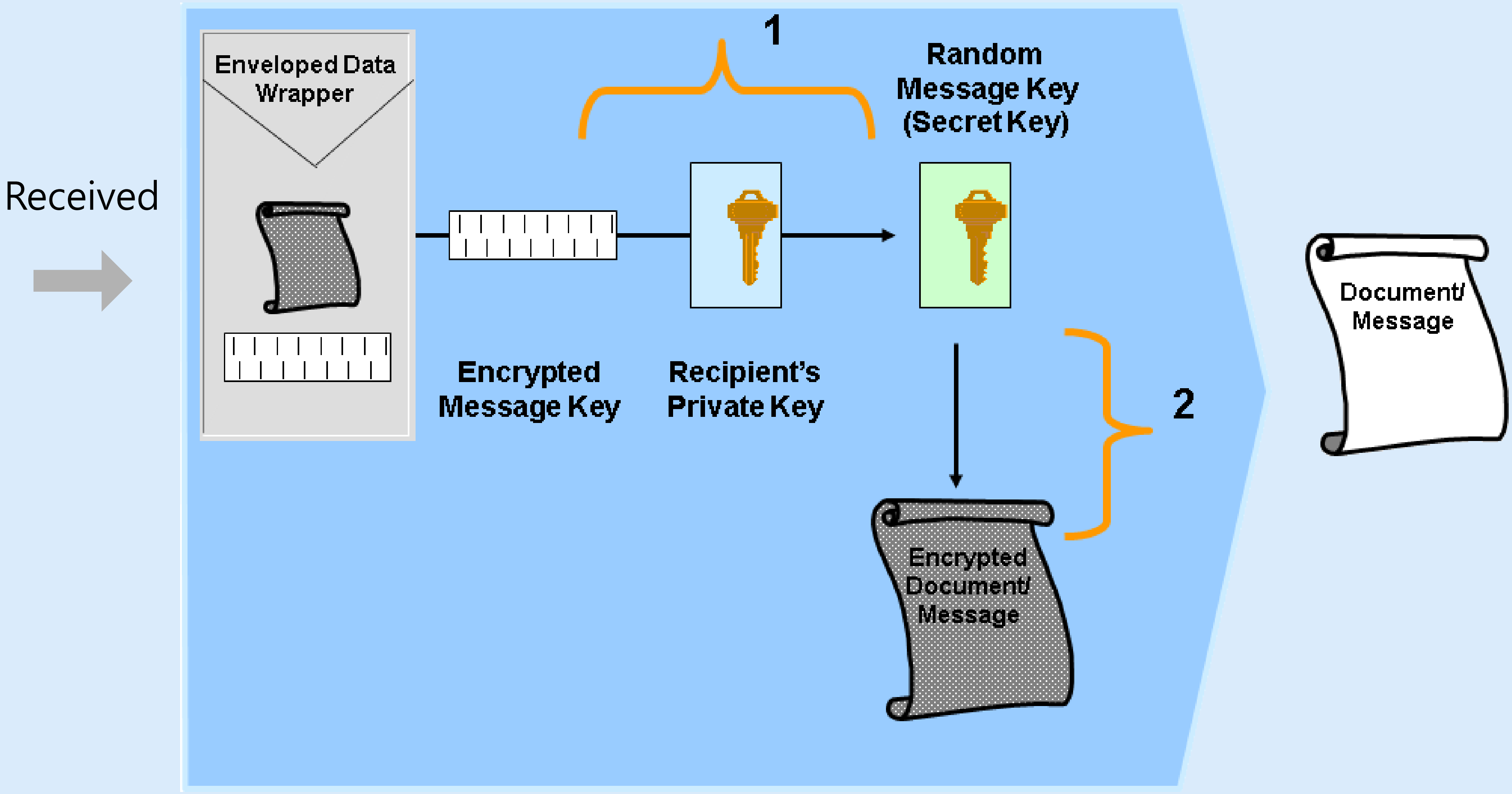
Digital Envelope (Hybrid Encryption)

- PKC is convenient, but much slower to compute compared to symmetric algorithms
  - e.g. in RSA, you have to compute $C^d \, mod \, n$ where both C and d are very large numbers
- Hybrid encryption allows using (v. efficient) symmetric key encryption is without sharing the key in advance
- A random secret key is generated for protecting the message. The key itself is protected using recipient's public key
- Encrypted message and the encrypted secret key are joined together in a wrapper, called digital envelope

Digital Envelope - Creating



Digital Envelope - Opening



Diffie-Hellman Key Exchange

- First public key algorithm invented
  - Published in 1976
- Specific method for securely exchanging cryptographic keys over a public channel
- Named after inventors Whitfield Diffie and Martin Hellman

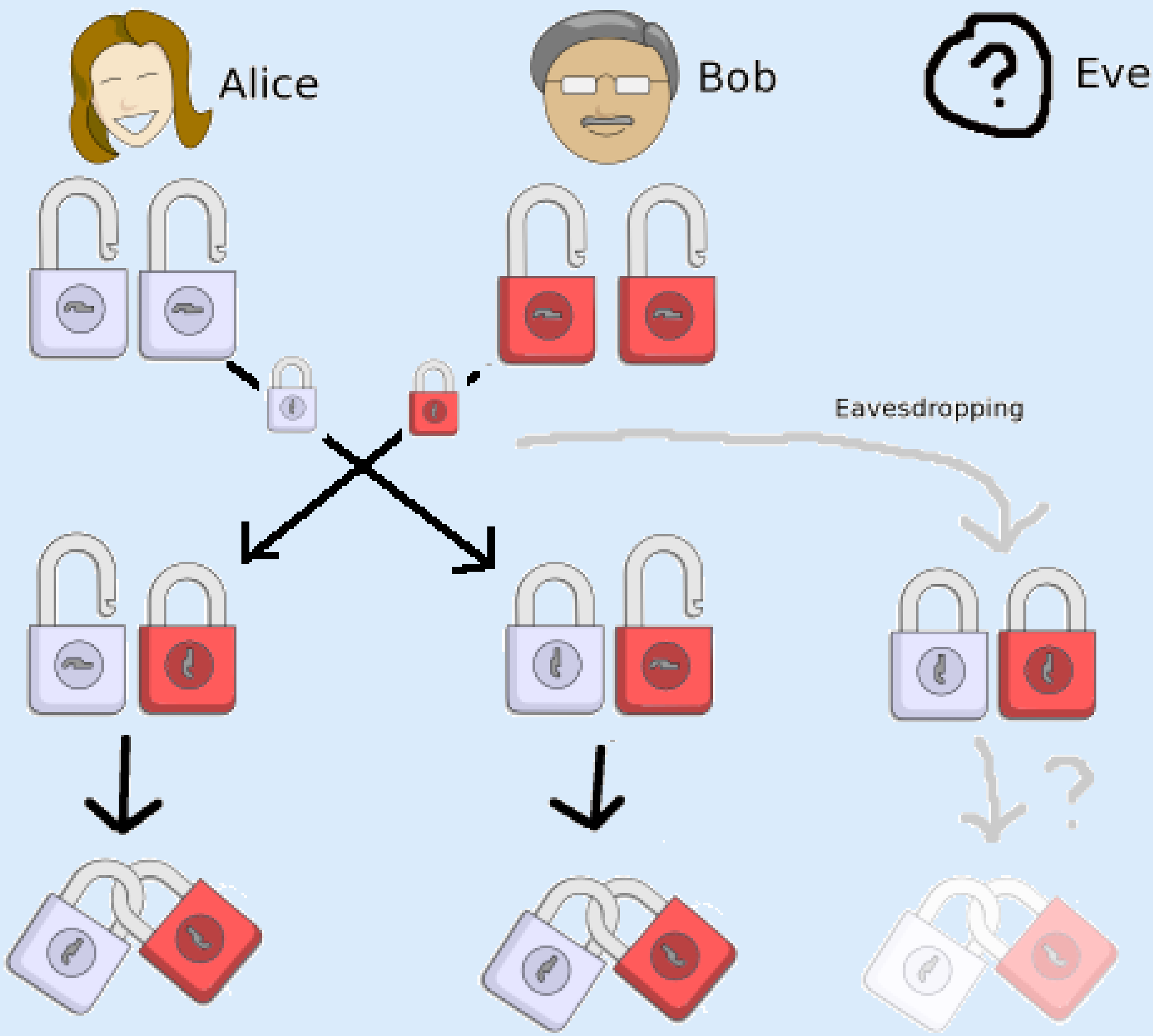
- It is an algorithm for establishing shared secret key, not meant for encryption or signatures

Diffie-Hellman Algorithm

- Consider two numbers g and p shared publically between Alice and Bob.
  - p is a prime number
  - g, called generator, is a primitive root of p
  - $1 < g < p$

- Alice computes $A = g^x \bmod p$ (x is the secret with Alice)
- Bob computes $B = g^y \bmod p$ (y is the secret with Bob)
- Alice and Bob exchange A & B
- Alice computes $K_{Alice} = B^x \bmod p$.
- Bob computes $K_{Bob} = A^y \bmod p$
- $K_{Alice} = K_{Bob} = g^{xy} \bmod p$

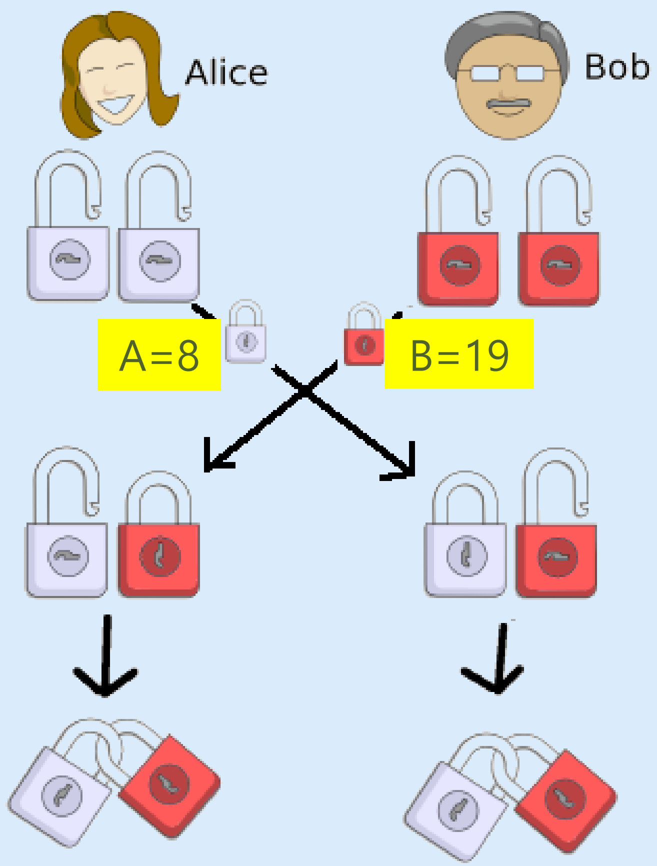
Diffie-Hellman Analogy



Diffie-Hellman Example

Alice and Bob agree on $p = 23 \ and \ g = 5$

Chooses a secret $x = 6$,
and computes:
$A = g^x \ mod \ p$
$\quad = 5^6 \ mod \ 23 = 8$



Chooses a secret $y = 15$,
and computes:
$B = g^y \ mod \ p$
$\quad = 5^{15} \ mod \ 23 = 19$

Computes shared key
$K = B^x \ mod \ p$
$\quad = 19^6 \ mod \ 23 = 2$

Computes shared key
$K = A^y \ mod \ p$
$\quad = 8^{15} \ mod \ 23 = 2$

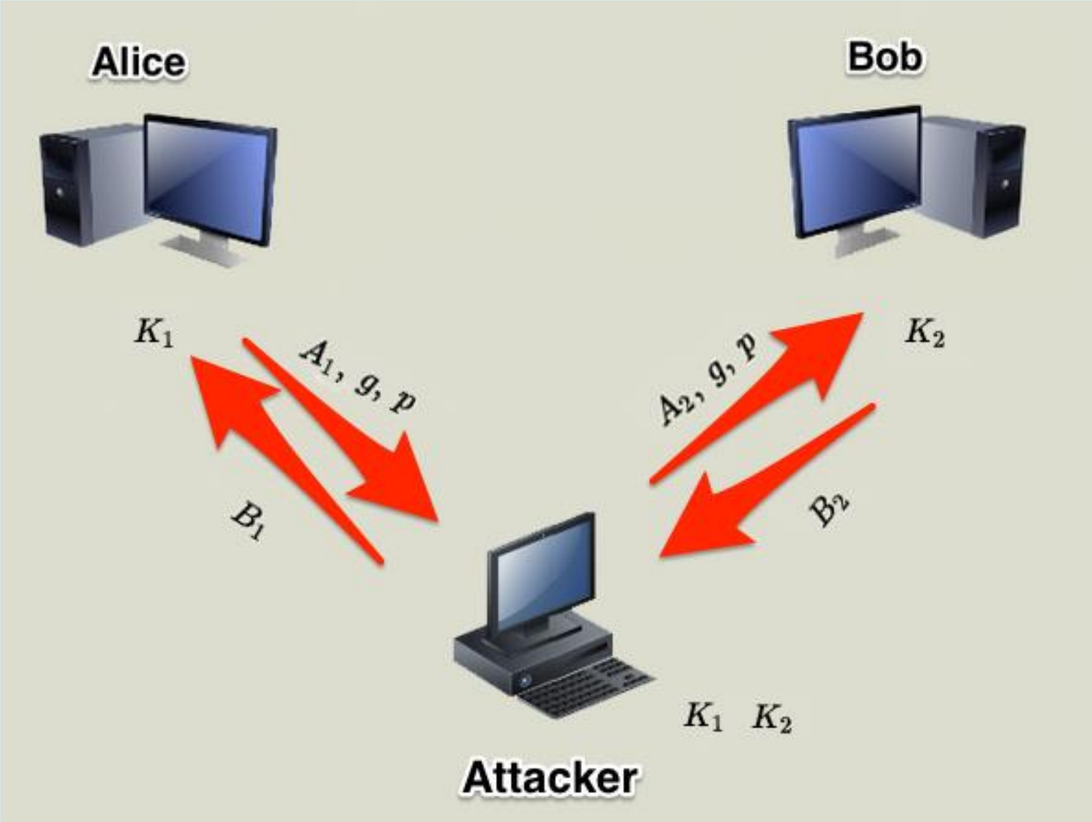
Question

Both Diffie-Hellman algorithm and Digital Envelope allow us to combine the efficiency of symmetric encryption with the convenience of PKC (asymmetric).

- When would you prefer one over the other?

MITM against Diffie-Hellman

- Vulnerable to main in the middle attack



MITM in PKC

- MITM is not unique to Diffie-Hellman key exchange
- All kinds of asymmetric crypto (RSA, digital signatures, digital envelope etc.) is vulnerable to such attacks
- Whenever public keys are exchanged over an insecure channel, we can not blindly trust the received public key.