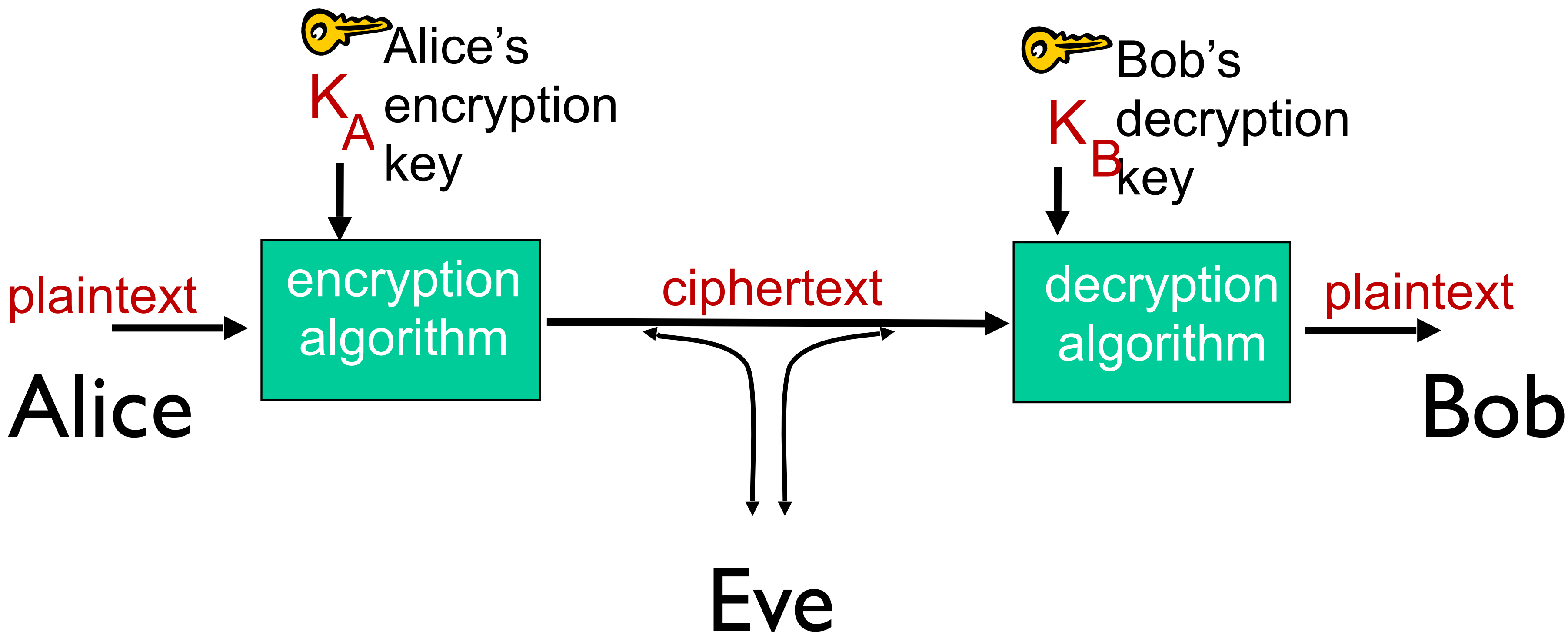
Cryptography

the language of cryptography



m plaintext message

$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

e.g.:

```
Plaintext:  bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

encryption key: mapping from set of 26 letters to set of 26 letters

a more sophisticated encryption approach

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- cycling pattern:
  - e.g., n=4: M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

Encryption key: n substitution ciphers, and cyclic pattern

- key need not be just n-bit pattern

symmetric key crypto: des

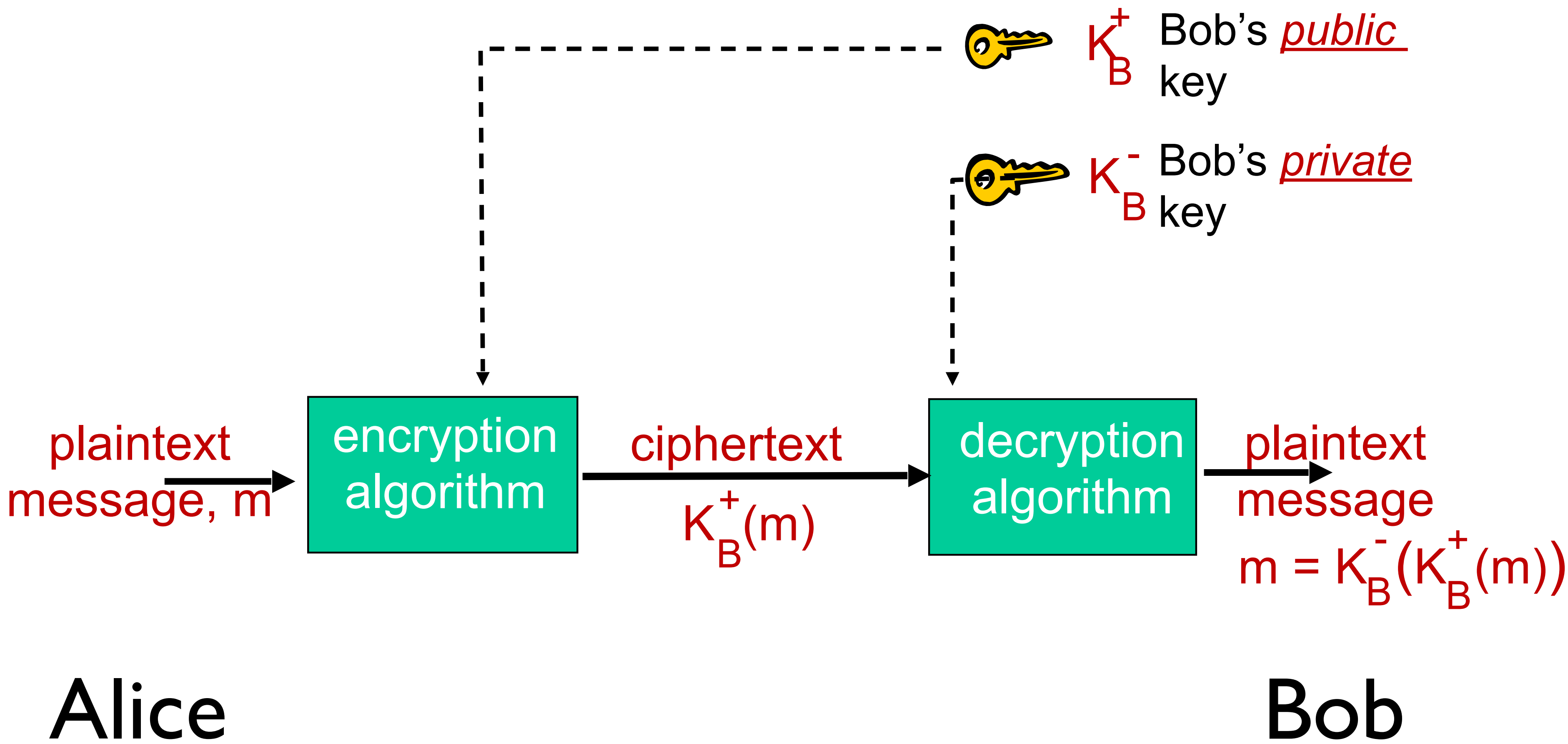
des: data encryption standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is **des**?
  - **des** challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - no known good analytic attack
- making **des** more secure:
  - **3des**: encrypt 3 times with 3 different keys

aes: advanced encryption standard

- symmetric-key NIST standard, replaced **des** (nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on **des**, takes 149 trillion years for **aes**

public key crypto



public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+ it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:

  (a+b) mod n = [(a mod n) + (b mod n)] mod n

  (a-b) mod n = [(a mod n) - (b mod n)] mod n

  (a*b) mod n = [(a mod n) * (b mod n)] mod n
- thus

  $a^d \bmod n = (a \bmod n)^d \bmod n$
- example: x=14, n=10, d=2:

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: important property

follows directly from modular arithmetic:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$
$$= m^{de} \bmod n$$
$$= (m^d \bmod n)^e \bmod n$$

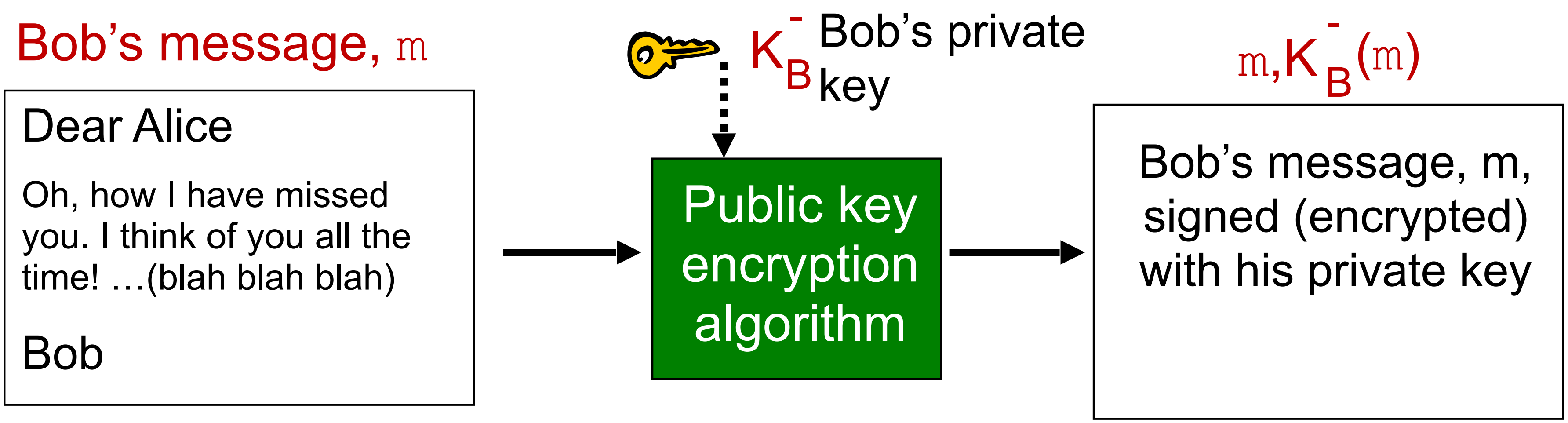
which leads to:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

digital signatures

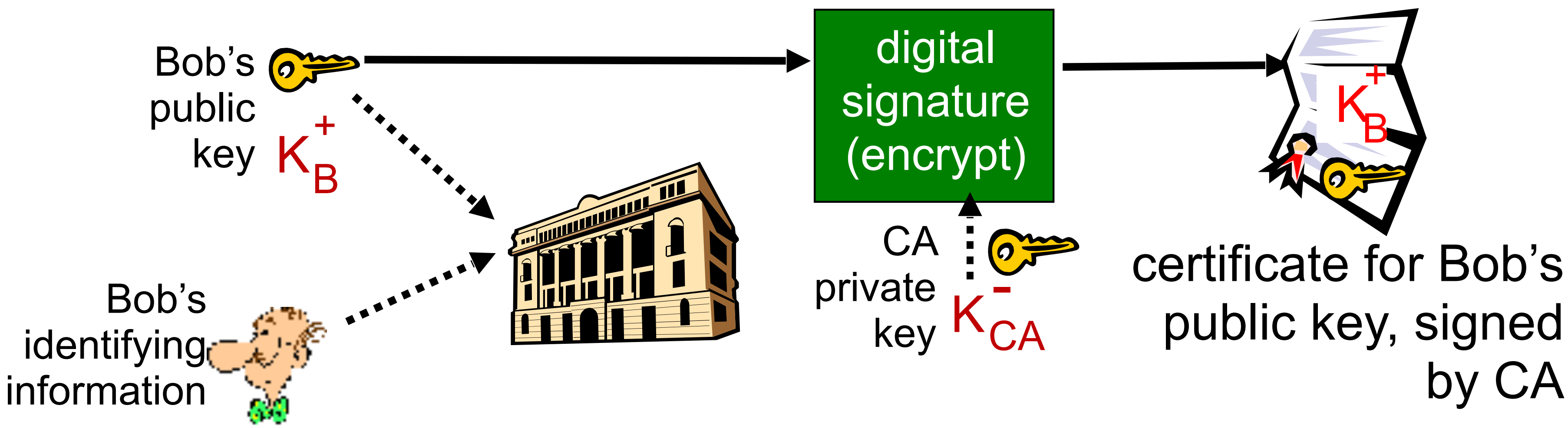
simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating **"signed"** message, $K_B^-(m)$



public-key certification

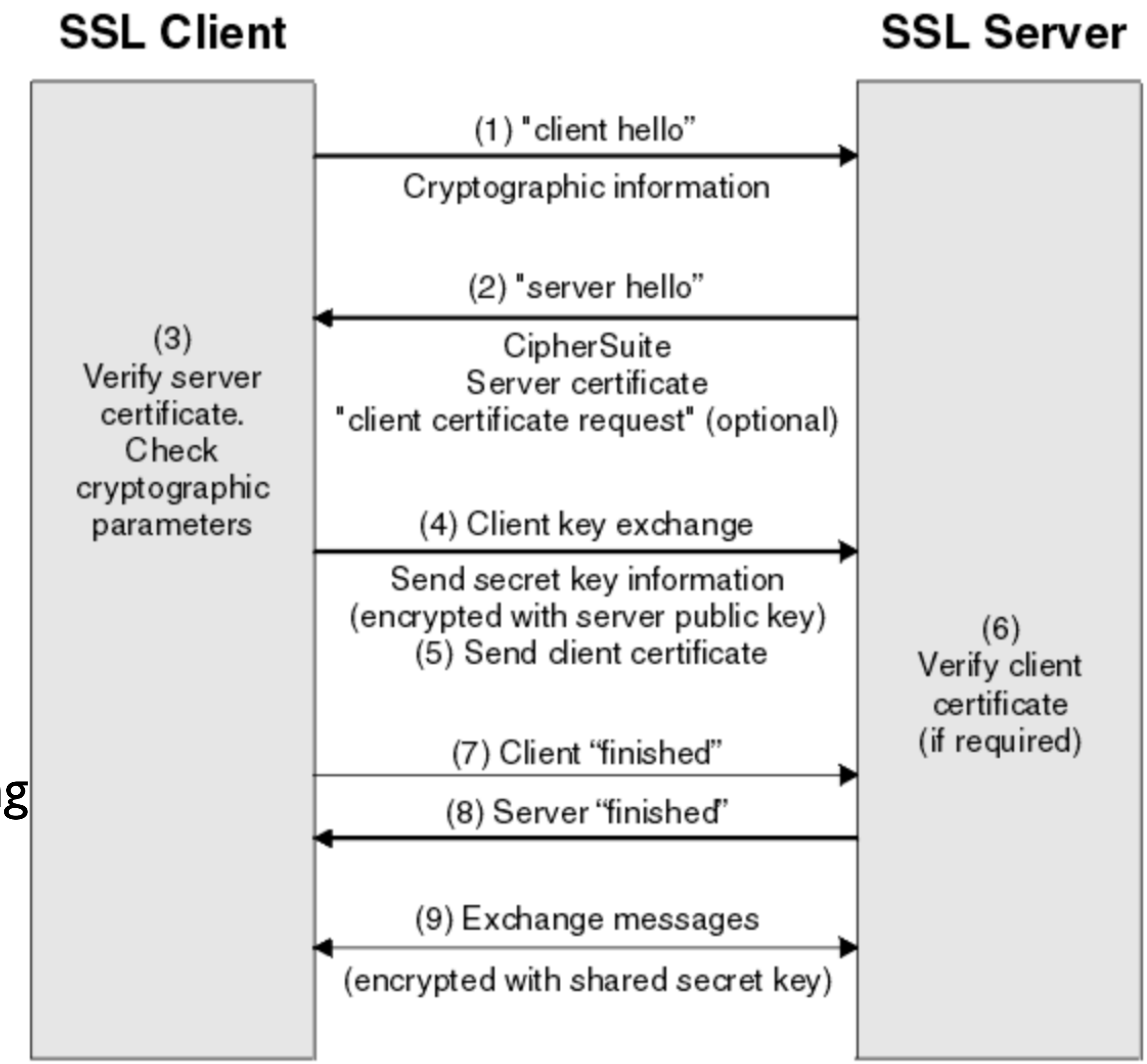
- *certification authority (CA):* binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



pk crypto in practice:

- you get a ssl certificate for your server
  - this includes, a pair of public and private keys
- once the certificate is installed on your server
- clients' browsers can verify it, via the trusted CA's they know
- before sending you a secret key and exchanging messages with your server

Figure 1. Overview of the SSL or TLS handshake



http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660.htm

Forum Participation Vote

- Current weight: 5%
- ~ 10% students == 5
- ~ 20% students == 0

- Vote

* Keep participation AS IS

* Transfer 5% to Quiz #10
  * (Won unanimously)