CSC309 *Programming on the Web*

week 11: cryptography

Amir H. Chinaei, Spring 2017

Office Hours: M 3:45-5:45 BA4222

ahchinaei@cs.toronto.edu

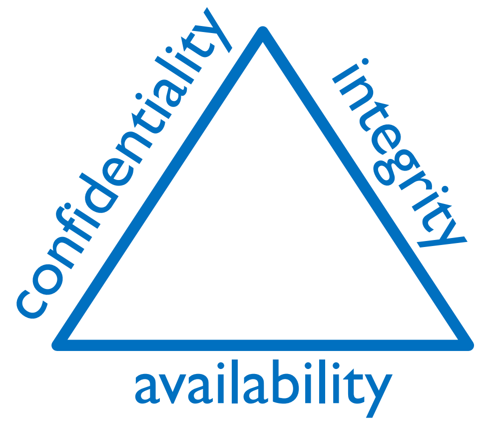
http://www.cs.toronto.edu/~ahchinaei/

some contents are from:

- Security in Computing: Pfleeger et al.
- Computer Security: Principles and Practice, Stallings et al.

review

- security requirements

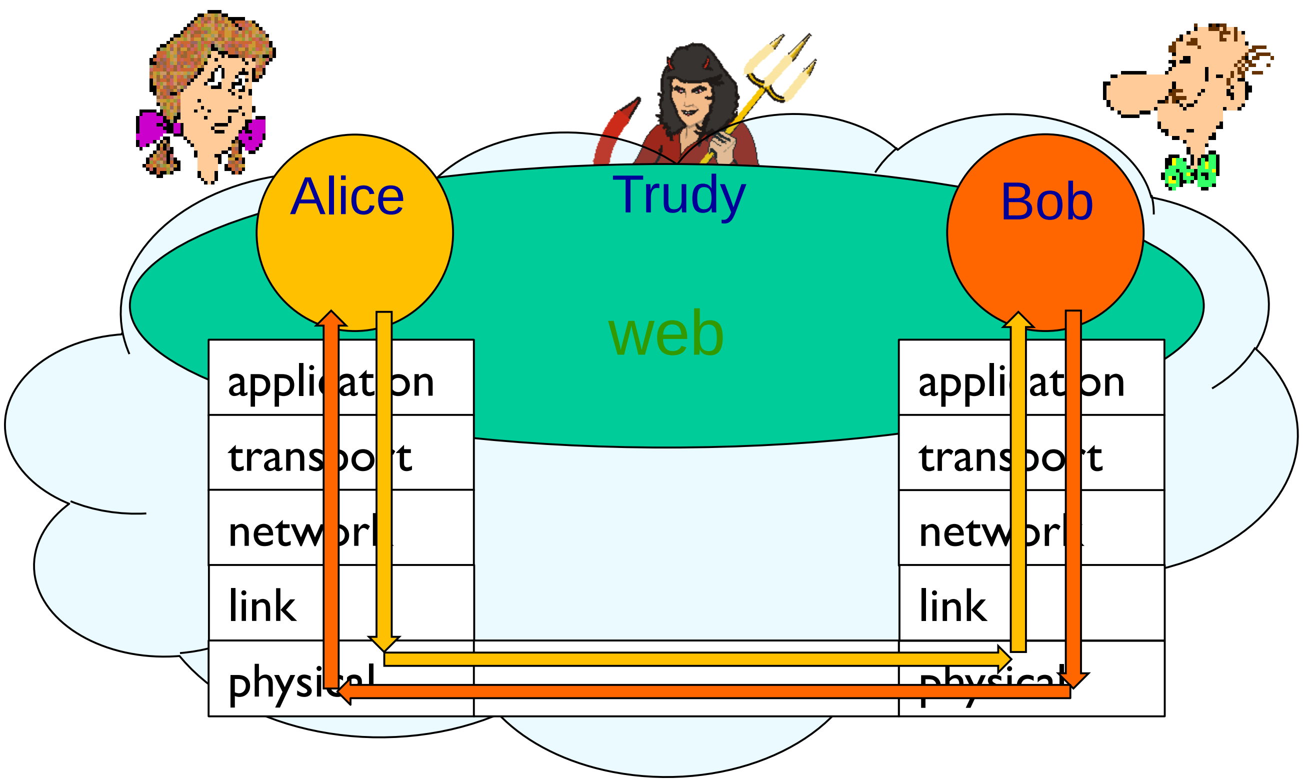


- attack vectors

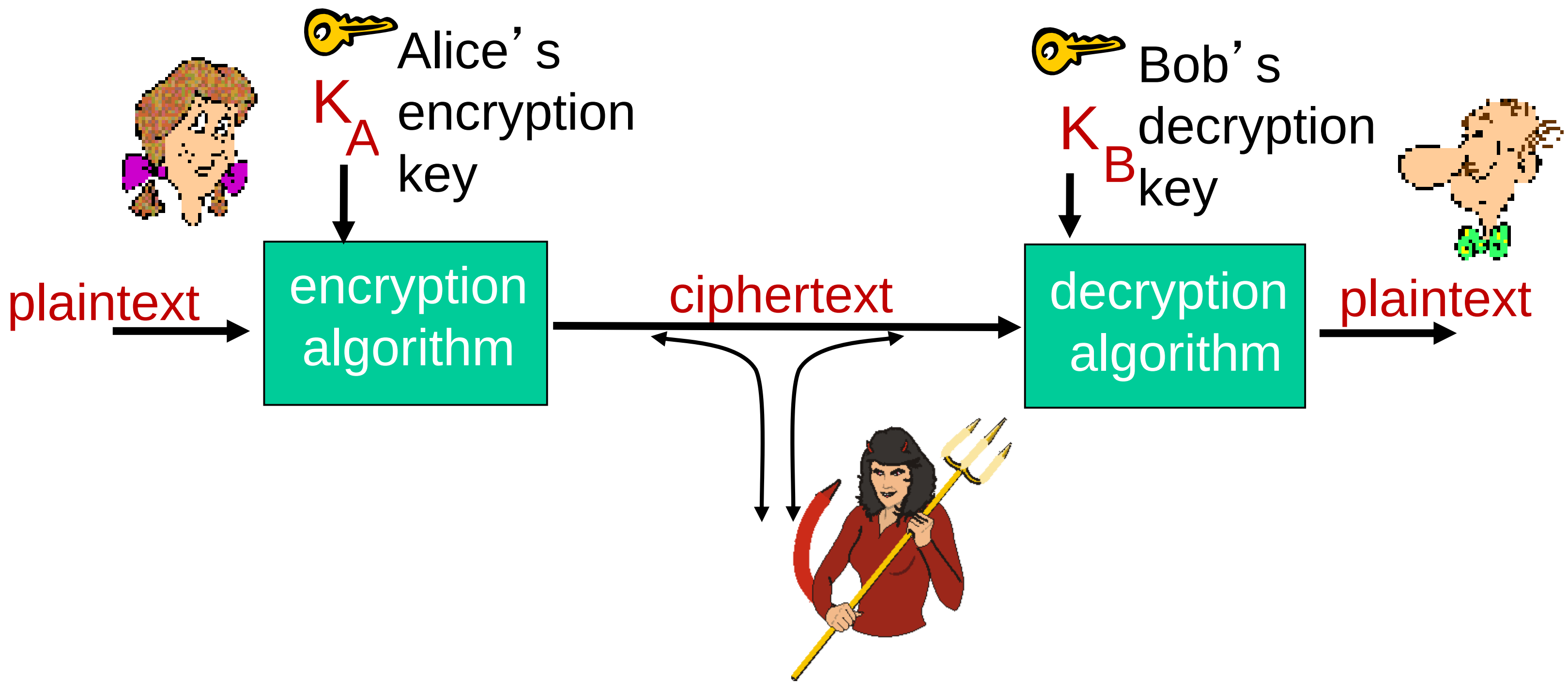
- **this week**
  - **cryptography**
    - to preserve confidentiality and integrity

friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



the language of cryptography



m plaintext message

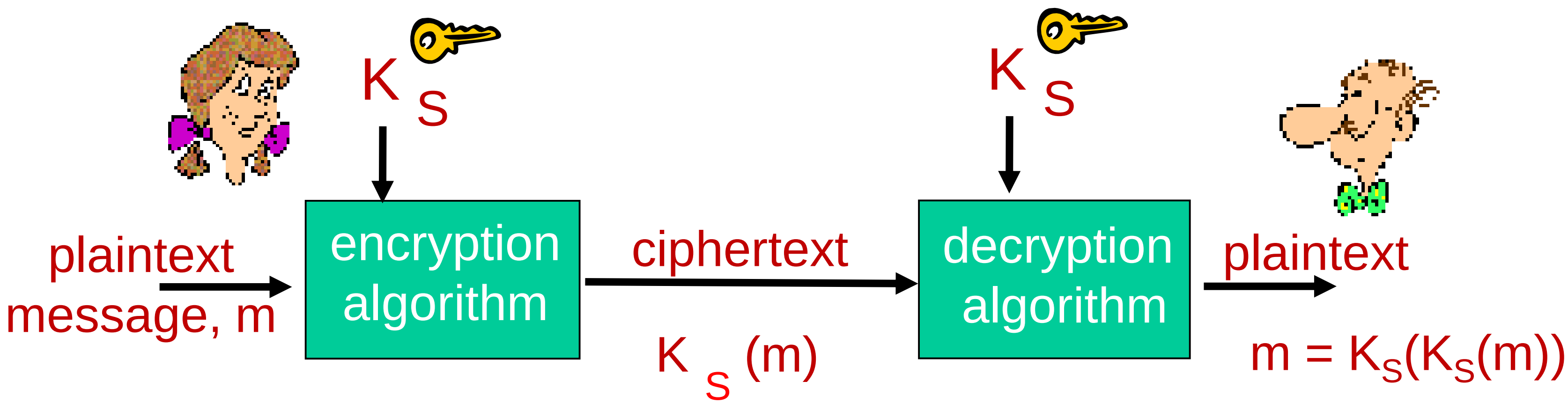
$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

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

- $\mathtt{n}$ substitution ciphers, $M_1, M_2, \ldots, M_n$
- cycling pattern:
  - e.g., $\mathtt{n}$=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: $\mathtt{n}$ 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

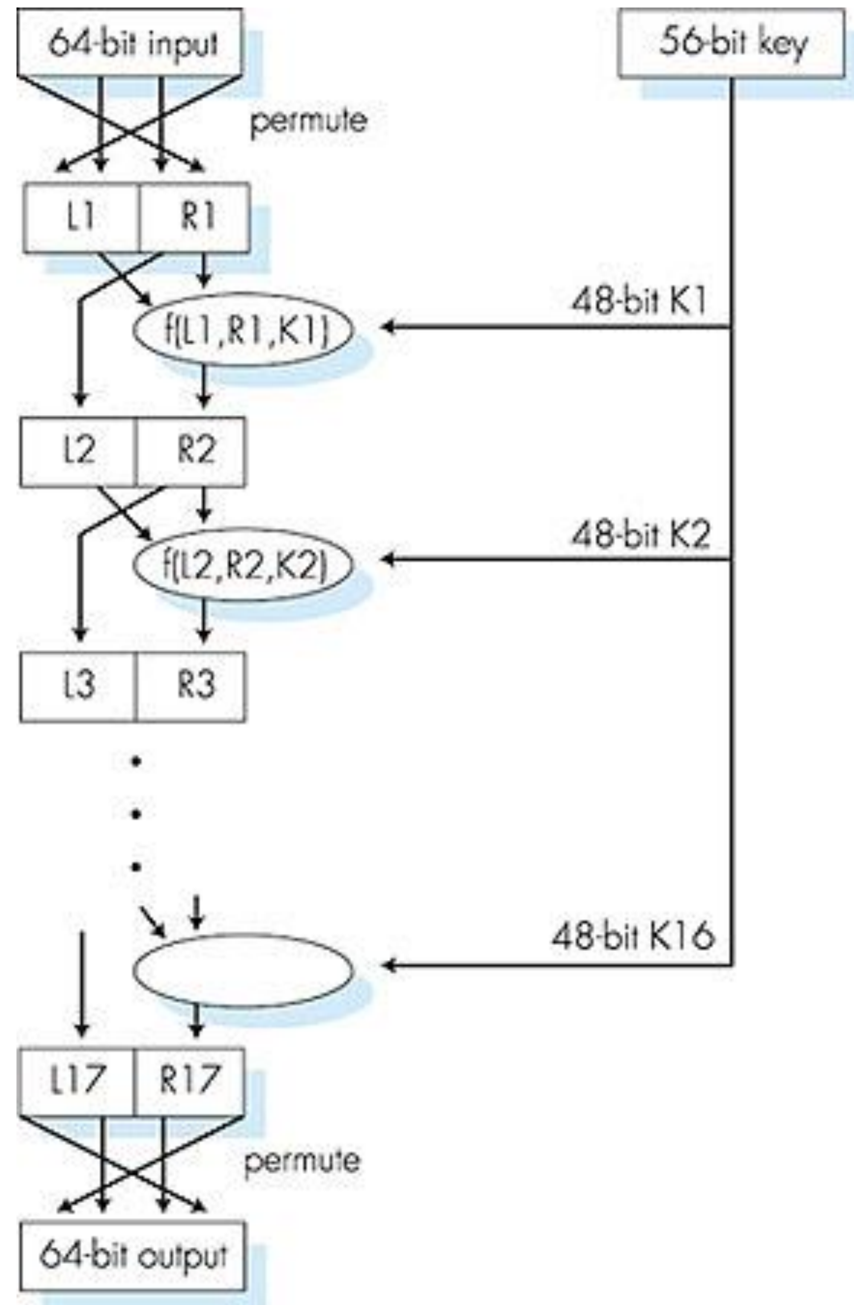
symmetric key crypto: **des**

***des** operation*

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



aes: advanced encryption standard

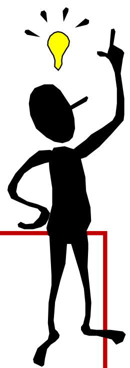
- symmetric-key NIST standard, replaced **des** (nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on **des**, takes 149 trillion years for **aes**

public key crypto

symmetric key crypto

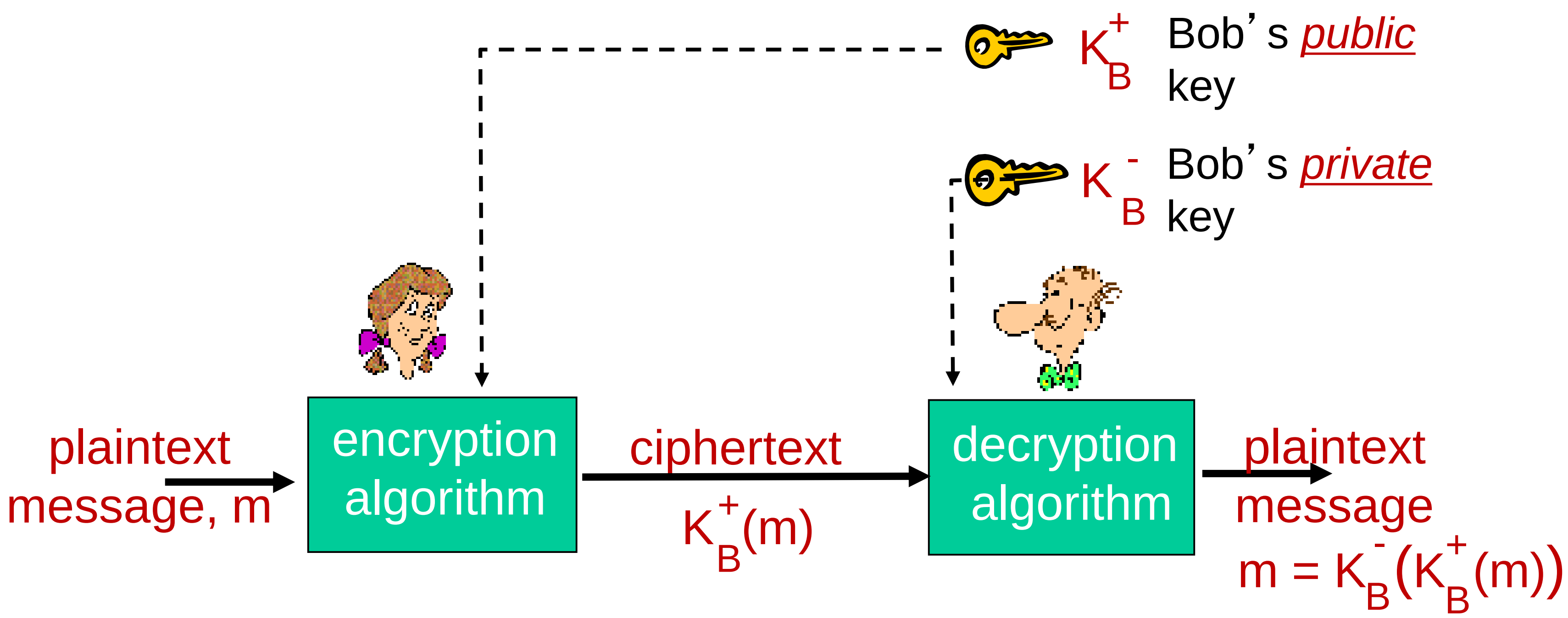
- requires client & server know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key crypto



- radically different approach [Diffie-Hellman76, RSA78]
- client & server do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to server

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: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number

example:

- `m`= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt `m`, we encrypt the corresponding number, which gives a new number (the ciphertext).

rsa: creating public/private key pair

1. choose two large prime numbers p, q (e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose e (with $e<n$) that has no common factors with z (e, z are "relatively prime").
4. choose d such that $ed-1$ is exactly divisible by z (in other words: $ed \bmod z = 1$).
5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

rsa: encryption, decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m $(<n)$, compute

   $c = m^e \bmod n$

2. to decrypt received bit pattern, c, compute

   $m = c^d \bmod n$

magic happens! $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

rsa example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.

	bit pattern	m	m^e	$c = m^e \bmod n$
encrypt:	00001100	12	24832	17

	c	c^d	$m = c^d \bmod n$
decrypt:	17	481968572106750915091411825223071697	12

why does rsa work?

- must show that $c^d \bmod n = m$
  where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where $n= pq$ and $z = (p-1)(q-1)$
- thus,

$$
\begin{aligned}
c^d \bmod n &= (m^e \bmod n)^d \bmod n \\
&= m^{ed} \bmod n \\
&= m^{(ed \bmod z)} \bmod n \\
&= m^1 \bmod n \\
&= m
\end{aligned}
$$

rsa: another important property

The following property will be *very* useful later:

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

result is the same!

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

follows directly from modular arithmetic:

$$
\begin{aligned}
(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\
&= m^{de} \bmod n \\
&= (m^d \bmod n)^e \bmod n
\end{aligned}
$$

why is rsa secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

rsa in practice: session keys

- exponentiation in **rsa** is computationally intensive
- **des** is at least 100 times faster than **rsa**
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

- Bob and Alice use **rsa** to exchange a symmetric key K_S
- once both have K_S, they use symmetric key cryptography