



CSCI 4250/6250 – Fall 2013

Computer and Networks Security

INTRODUCTION TO CRYPTO

CHAPTER 8 (Goodrich)

CHAPTER 2-6 (Kaufman)

CHAPTER 8 (Kurose)

- ▶ Slides adapted from Kurose et al., Goodrich et al., and Kaufman et al.

Public Key Cryptography

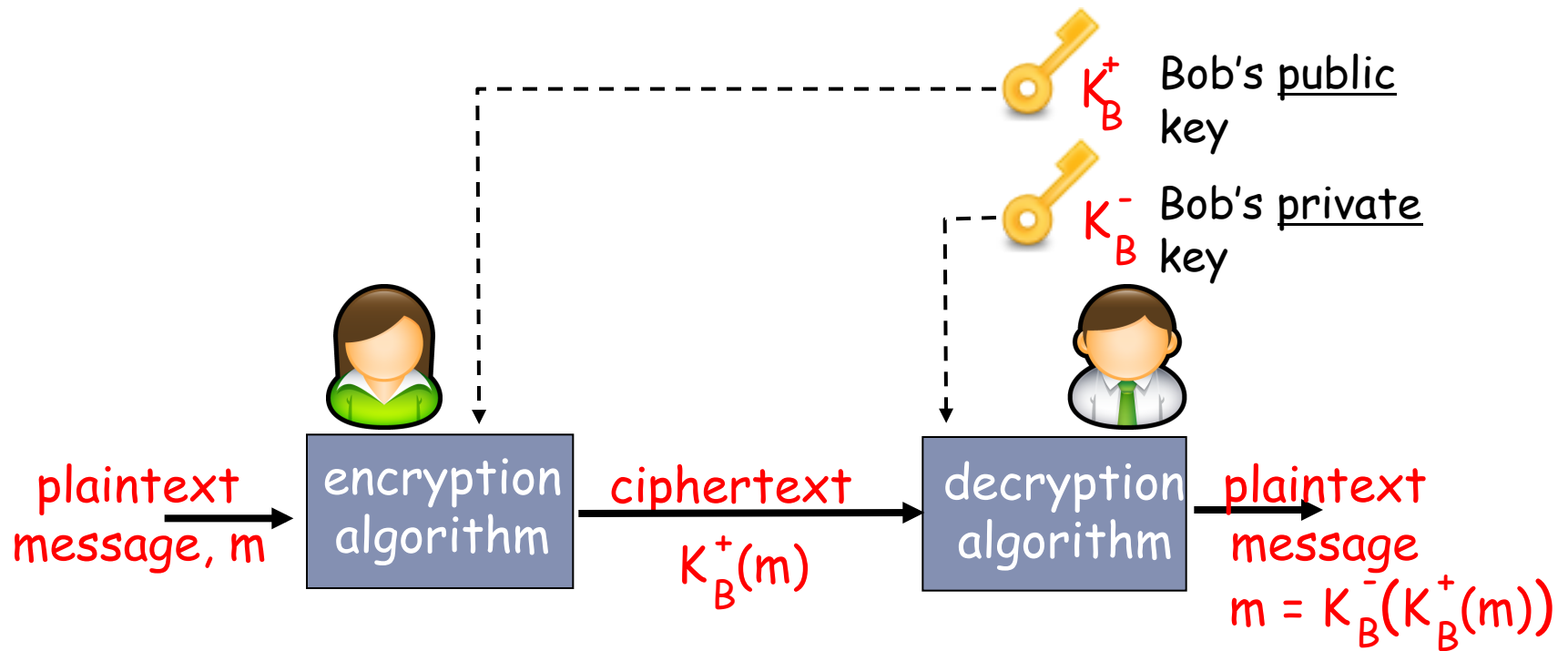
symmetric key crypto

- ▶ requires sender, receiver know shared secret key
- ▶ Q: how to agree on key in first place (particularly if never “met”)?

public key cryptography

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

Public key cryptography



Public key encryption algorithms

Requirements:

- ① need K_B^+ () and K_B^- () 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


RSA: getting ready

- ▶ A message is a bit pattern.
- ▶ A 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 cyphertext).

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 (with $d < n$) so that $ed-1$ is divisible by z .
(in other words: $ed \bmod z = 1$).
5. Public key is (n, e) . Private key is (n, d) .


RSA: Creating public/private key pair

1. Choose two large prime numbers p, q .
(e.g., 1024 bits each, to avoid brute force given n)
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 (with $d < n$) so that $ed-1$ is divisible by z .
(in other words: $ed \bmod z = 1$).

5. Public key is (n, e) . Private key is (n, d) .

$\underbrace{(n, e)}_{K_B^+}$ $\underbrace{(n, d)}_{K_B^-}$

e can be relatively small
 d should be large

RSA: Encryption, decryption

public private

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)

$ed-1 = 144$, $144/24=6$

Encrypting 8-bit messages.

encrypt: bit pattern m m^e $c = m^e \bmod n$
 00001000 12 24832 17

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

Prerequisite: modular arithmetic

▶ $x \bmod n =$ remainder of x when divide by n

▶ **Facts:**

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$$

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$$

▶ **Thus**

$$(a \bmod n)^d \bmod n = a^d \bmod n$$

▶ **Example: $x=14, n=10, d=2$:**

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

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

Multiplicative Inverses (1)

- ▶ The **residues** modulo a positive integer n are the set

$$\mathbf{Z}_n = \{0, 1, 2, \dots, (n - 1)\}$$

- ▶ Let x and y be two elements of \mathbf{Z}_n such that

$$xy \bmod n = 1$$

We say that y is the **multiplicative inverse** of x in \mathbf{Z}_n and we write $y = x^{-1}$

- ▶ **Example:**
 - ▶ Multiplicative inverses of the residues modulo 10

x	0	1	2	3	4	5	6	7	8	9
x^{-1}		1		7				3		9

Multiplicative Inverses (2)

Theorem

An element x of \mathbb{Z}_n has a multiplicative inverse if and only if x and n are relatively prime

▶ Example

- ▶ The elements of \mathbb{Z}_{10} with a multiplicative inverse are 1, 3, 7, 9

Corollary

If p is prime, every nonzero residue in \mathbb{Z}_p has a multiplicative inverse

▶ Example:

- ▶ Multiplicative inverses of the residues modulo 11

x	0	1	2	3	4	5	6	7	8	9	10
x^{-1}		1	6	4	3	9	2	8	7	5	10

Euler's Theorem

- ▶ The multiplicative group for Z_n , denoted with Z_n^* , is the subset of elements of Z_n relatively prime with n
- ▶ The totient function of n , denoted with $\phi(n)$, is the size of Z_n^*
- ▶ Example

$$Z_{10}^* = \{ 1, 3, 7, 9 \} \quad \phi(10) = 4$$

- ▶ If p is prime, we have

$$Z_p^* = \{ 1, 2, \dots, (p - 1) \} \quad \phi(p) = p - 1$$

Euler's Theorem

For each element x of Z_n^* , we have $x^{\phi(n)} \bmod n = 1$

- ▶ Example ($n = 10$)

$$3^{\phi(10)} \bmod 10 = 3^4 \bmod 10 = 81 \bmod 10 = 1$$

$$7^{\phi(10)} \bmod 10 = 7^4 \bmod 10 = 2401 \bmod 10 = 1$$

$$9^{\phi(10)} \bmod 10 = 9^4 \bmod 10 = 6561 \bmod 10 = 1$$

- ▶ Consequence

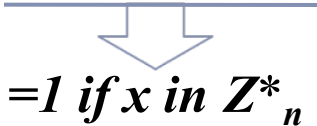
- ▶ $x^y \bmod n = x^{y \bmod \phi(n)} \bmod n$

Why?

▶ Remember

- ▶ $[(a \bmod n)(b \bmod n)] \bmod n = (ab) \bmod n$
- ▶ $(a \bmod n)^d \bmod n = a^d \bmod n$

▶ Then

- ▶ $x^y \bmod n = x^{(k\phi(n)+r)} \bmod n = x^{k\phi(n)} x^r \bmod n =$
 $[(x^{k\phi(n)} \bmod n)(x^r \bmod n)] \bmod n = x^{y \bmod \phi(n)} \bmod n$

 $=1 \text{ if } x \text{ in } Z_n^*$

Why does RSA work?

- ▶ Remember that
 - ▶ p and q are two large primes
 - ▶ $n = pq$; $z = (p-1)(q-1) = \phi(n)$
 - ▶ $ed \bmod z = 1$
- ▶ z is equal to the *totient* of n
 - ▶ the number of *numbers* $< n$ that are relatively prime to n
- ▶ Fact: for any x and y , $x^y \bmod n = x^{(y \bmod z)} \bmod n$
- ▶ We need to show that $c^d \bmod n = m$, where $c = m^e \bmod n$

$$\begin{aligned}c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(ed \bmod z)} \bmod n \\ &= m^1 \bmod n \\ &= m \implies \text{(notice that } m \text{ in } [0, n-1])\end{aligned}$$

RSA: another important property

The following property will be *very* useful later:

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

use public key
first, followed
by private key

use private key
first, followed
by 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.

Algorithmic Issues

- ▶ The implementation of the RSA cryptosystem requires various algorithms
- ▶ Overall
 - ▶ Representation of integers of arbitrarily large size and arithmetic operations on them
- ▶ Encryption
 - ▶ Modular power
- ▶ Decryption
 - ▶ Modular power
- ▶ Setup
 - ▶ Generation of random numbers with a given number of bits (to generate candidates p and q)
 - ▶ Primality testing (to check that candidates p and q are prime)
 - ▶ Computation of the GCD (to verify that e and $\phi(n)$ are relatively prime)
 - ▶ Computation of the multiplicative inverse (to compute d from e)

Session keys

- ▶ Exponentiation is computationally intensive
- ▶ DES is at least 100 times faster than RSA

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

Diffie-Hellman

- ▶ **Public key cryptosystem**

- ▶ First known public key-based system
- ▶ Useful to perform key exchange when communication channel is not private

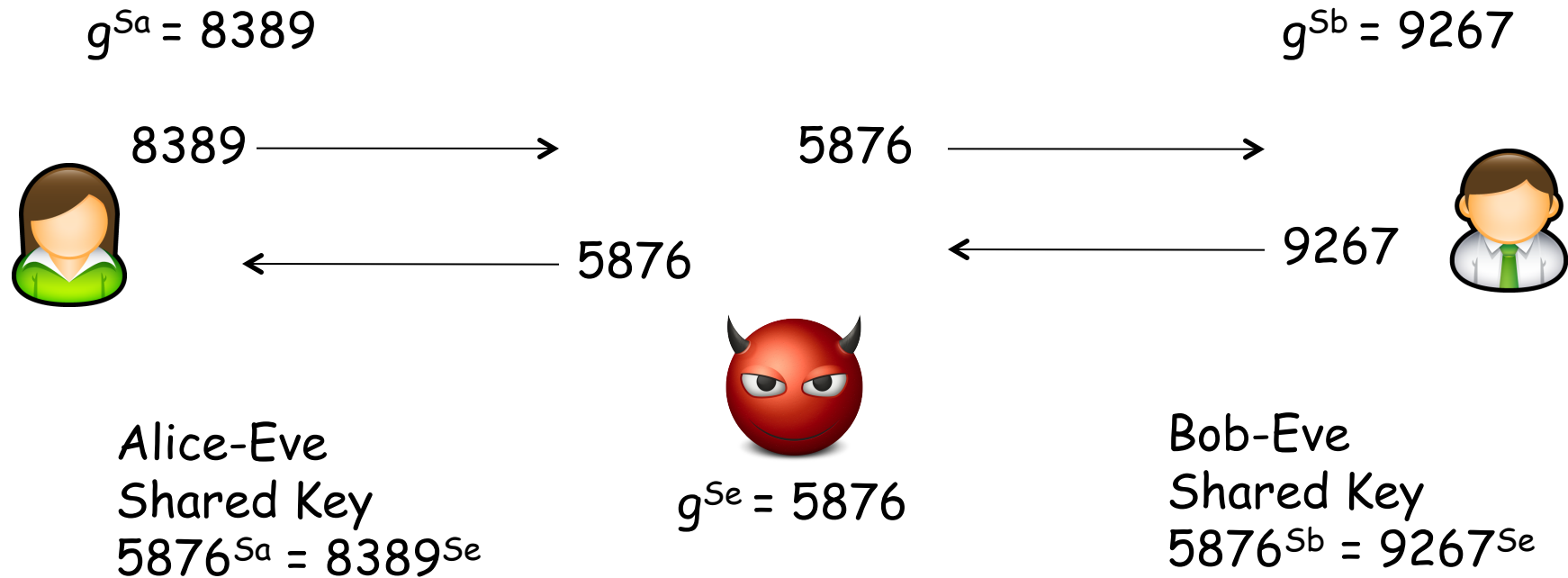
- ▶ **Alice and Bob first agree on a large prime p and another number $g < p$ (some subtle restrictions apply...), then**

1. g and p can be published (no need to keep them secret)
2. Alice chooses a random number S_a , and Bob a rand num S_b
3. Alice computes $T_a = g^{S_a} \bmod p$, Bob computes $T_b = g^{S_b} \bmod p$
4. Alice and Bob exchange T_a and T_b (in public)
5. Alice and Bob compute $T_b^{S_a} \bmod p$ and $T_a^{S_b} \bmod p$, respectively
6. They will get the same number (the exchanged key)
$$T_b^{S_a} = g^{S_b S_a} \bmod p = g^{S_a S_b} \bmod p = T_a^{S_b}$$

Diffie-Hellman

- ▶ **Why is this secure?**
 - ▶ Nobody else can calculate $g^{S_a S_b}$, even if they separately know $T_a = g^{S_a} \bmod p$ and $T_b = g^{S_b} \bmod p$
 - ▶ To get S_a or S_b an attacker would need to compute discrete logarithms
 - ▶ Discrete logarithms are very hard to compute
 - ▶ Mathematicians have not yet figured out how to do it efficiently
- ▶ **Vulnerable to man-in-the-middle attack in certain scenarios**
 - ▶ Alice and Bob do not authenticate each other
 - ▶ Attacker may intercept and replace T_a and T_b
 - ▶ To solve (or mitigate) problem, T_a and T_b should be stored in a secure repository of “public numbers”

DH – Man-in-the-Middle Attack



Does it help if Alice and Bob try to verify their identity by sending each other a pre-shared password?

DH – Man-in-the-Middle Defense

▶ Published DH numbers

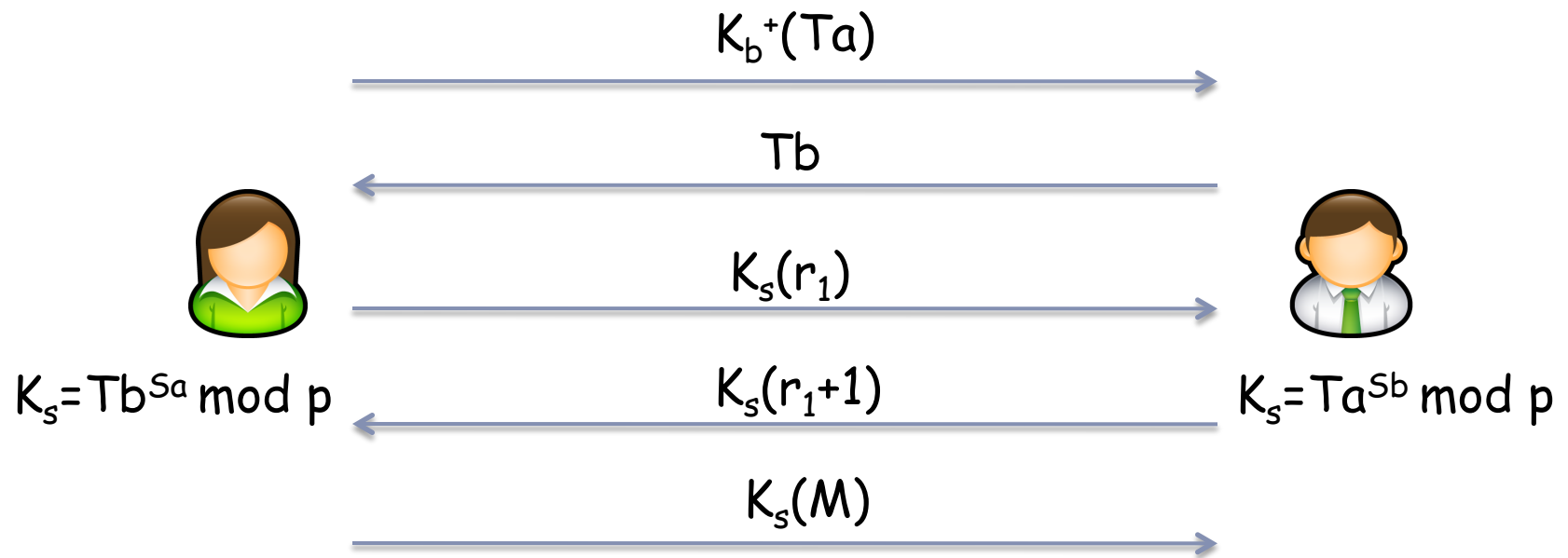
- ▶ p and g are agreed upon
- ▶ Each party chooses a fixed secret number S_i and publishes her ($T_i = g^{S_i} \bmod p$) in a reliable place
- ▶ Assumption: the attacker cannot change/forge p and g

▶ Authenticated DH, examples

- ▶ Alice can sign her T_a
- ▶ Alice can encrypt her T_a with Bob's pub key
- ▶ After DH, Alice sends Bob a hash $H(S|T_a)$, where S is a pre-shared secret (e.g., a password)

DH – Man-in-the-Middle Defense

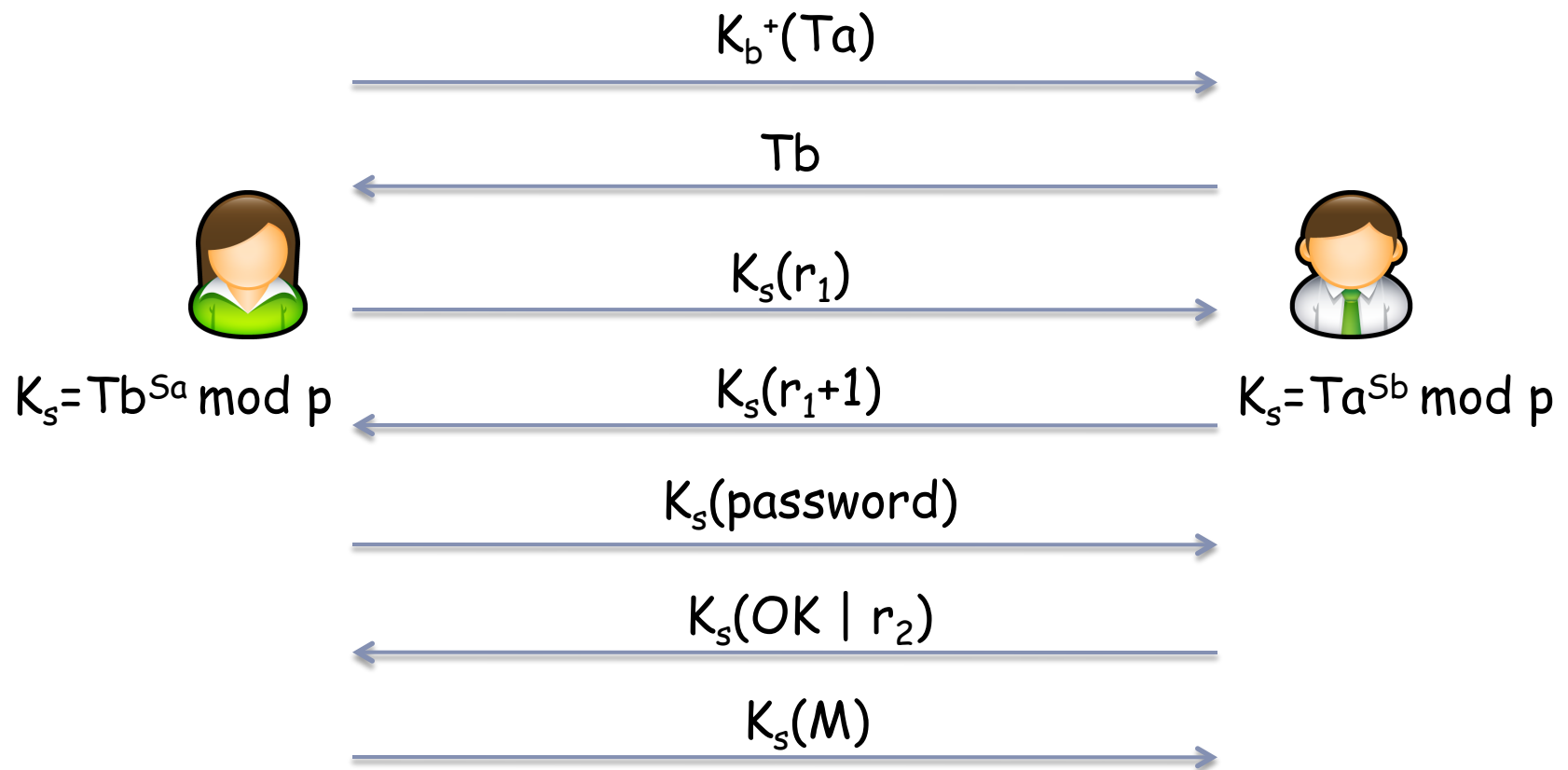
- ▶ Bob is a server, and has a priv/pub key
- ▶ Alice knows (and trusts) Bob's pub key, K_b^+



This seems to have significant problems
(Eve can still pretend to be Alice)

DH – Man-in-the-Middle Defense

- ▶ Bob is a server, and has a priv/pub key
- ▶ Alice knows Bob's pub key, K_b^+



Perfect Forward Secrecy

- ▶ A protocol is said to have PFS if it is impossible for Trudy to decrypt a message m sent between Alice and Bob, even if Trudy, after m is sent, breaks into both Alice's and Bob's machines and steals their private keys
- ▶ This can be achieved by using session keys that
 - ▶ Are chosen independently from the private/public keys
 - ▶ Alice and Bob forget the session key as soon as the communication is over
- ▶ E.g., this can be done using Diffie-Hellman
 - ▶ Alice and Bob forget their S_a and S_b after end of session
 - ▶ To avoid man-in-the-middle, Alice "signs" T_a with her private key, and Bob "signs" T_b with his pub key

Zero-Knowledge Proof Systems

- ▶ Used only for authentication
- ▶ Allows you to prove that you know a secret without actually revealing the secret
- ▶ E.g.: RSA is a zero-knowledge proof system
 - ▶ You can prove you know the “secret” associated with your public key without revealing your private key
- ▶ There exist ZKPSs that are much more efficient than RSA