Computer Networks COL 334/672

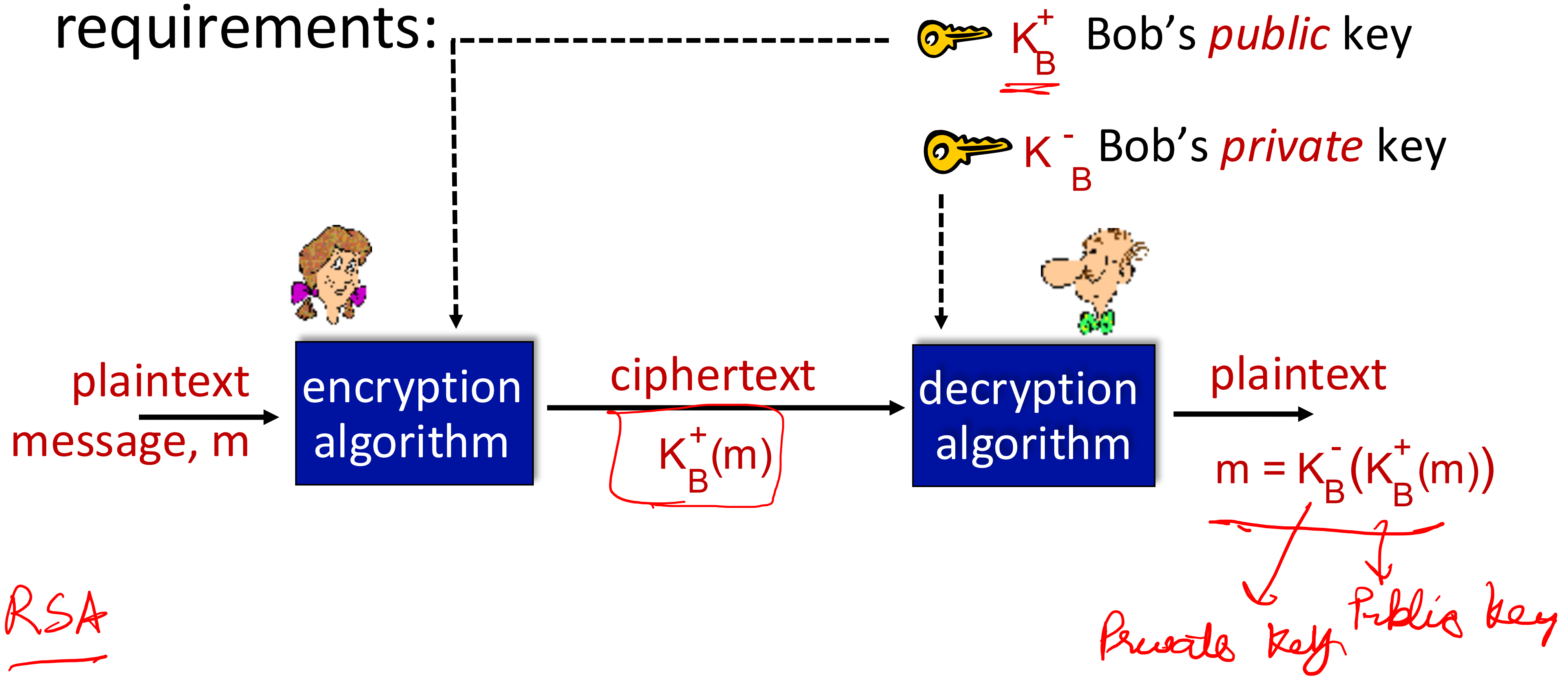
Network Security

Tarun Mangla

Slides adapted from KR

Sem 1, 2024-25

Public Key Cryptography



Prerequisite: modular arithmetic

6 mod 4
= 2

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

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

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

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: Creating public/private key pair

if a and n are co-prime

$a^z \equiv 1 \mod n$

Euler's theorem

if p is prime and a and p are co-prime

1. choose two large prime numbers p, q. (e.g., 1024 bits each)

$a^p \equiv a \mod p$ → Fermat's theo

2. compute $\boxed{n} = pq$, $z = (p-1)(q-1)$ → # of co-primes of n that are less than n

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: $ed \mod z = 1$).

$ed \mod z = 1$ or $ed \equiv 1 \mod z$

5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

public key private key

1001 = 9

n = 35

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

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

   $$\boxed{\boxed{c} = m^e \bmod n}$$

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:	00001000	12	24832	17

$17^{29} \bmod 35$

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

$ed = 2k+1$

~~c~~ $c = m^e \bmod n$

$(m^e \bmod n)^d \bmod n$

Why does RSA work?

Need to show that $c^d \bmod n = m$

$\downarrow$

$m^{ed} \bmod n$

$ed \equiv 1 \bmod z$

- 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)

$m^{2k+1} \bmod n$

$m^z \bmod n = 1$

$m \cdot m^{zk} \bmod n$

- thus,

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

$m \cdot (m^z \bmod n)^k \bmod n$

$= m \bmod n$

Why is RSA secure?

public

- 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

$\sqrt{n}$

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 session key K_S
- once both have K_S, they use symmetric key cryptography

What is network security?

confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

authentication: sender, receiver want to confirm identity of each other

access and availability: services must be accessible and available to users

What about integrity and authenticity?

Recall

- message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- authentication: sender, receiver want to confirm identity of each other

Encryption alone can not provide integrity and authentication.

How to provide integrity and authenticity?

Providing integrity

- Given a message m, how can you detect if this has been tampered with?

m , $f(m)$

m'

$f(m') \neq f(m)$

m, $f'(m)$

$m' \neq m$

Alice Bob

Trudy

(Checksum)

Checksum(m') = checksum(m)

computationally hard to find m' such that $f'(m') = f(m)$

Bob sends : m, $H(m)$

Hash

Alice : $H(m)$

Alice Bob

m', $H(m')$ → I am Bob

Trudy

Providing integrity

- Given a message m, how can you detect if this has been tampered with?

Hash function properties:

- many-to-1
- produces fixed-size msg digest
- given message digest x, computationally infeasible to find m such that $x = H(m)$

How to provide authenticity?

Using Shared Secret Key

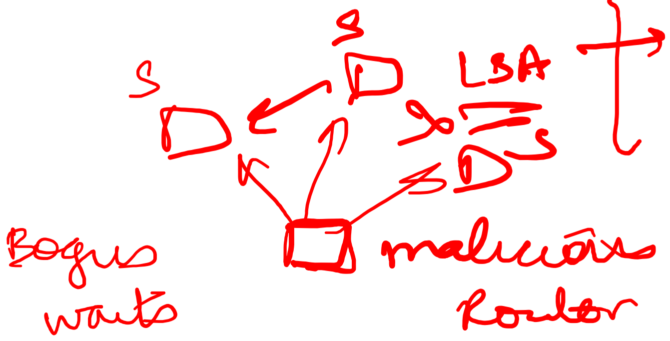
Alice $\xleftarrow{m}$ Bob

(S) (S)

Send: m, $H(m+s)$ → Message Authentication Code (MAC)

Receiver side: $H(m+s)$

How do you have a shared secret key at first place?



→ Integrity (Authenticity)

Using Public Key Cryptography

Public key cryptography

m

Bob (Authentic)

& ensure integrity

$\boxed{m, K_B^-(m)}$ → Share this with everyone

→ Digital Signatures

Bob's message, m

alice

K_B^- Bob's private key

$m, K_B^-(m)$

Dear Alice

Oh, how I have missed you. I think of you all the time! …(blah blah blah)

Bob

Public key encryption algorithm

Dear Alice

Oh, how I have missed you. I think of you all the time! …(blah blah blah)

Bob $K_B^-(m)$

– SHA-1 → 160 bits

– MD5 → 128 bits

$m, K_B^-(H(m))$

Hashing functions

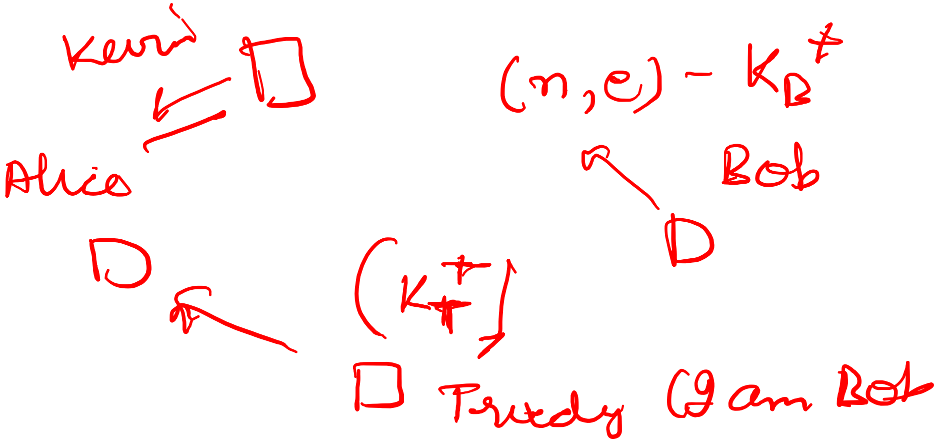
n bits → l bits where l << n

Digital Signatures

How are public keys ~~distributed at~~ first place?

How to share secret key?

How to securely share public key?



① Conf
② Auth
③ Integ

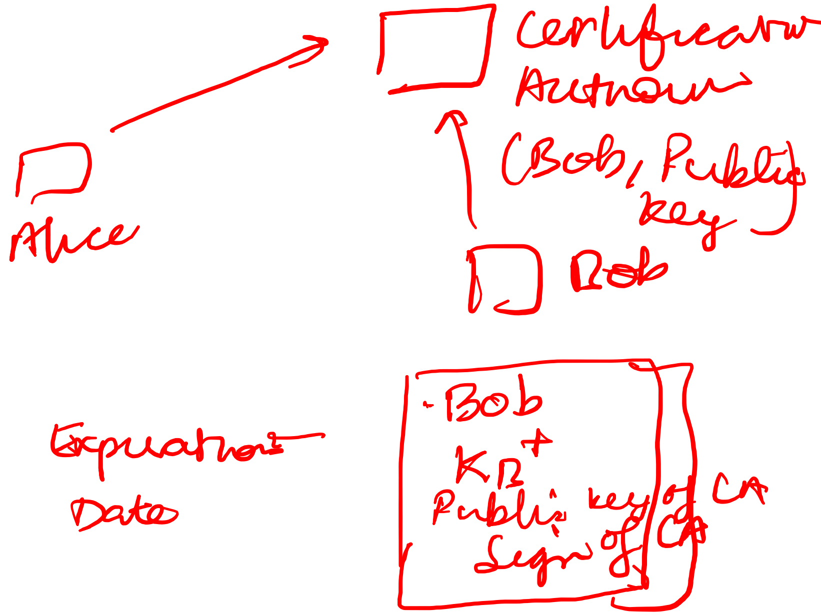
→ Maybe Alice knows IP addr of Bob (spoofing)

Certificates → Trusted entity

X → Y → Z

X → Z

Public key Infrastructure



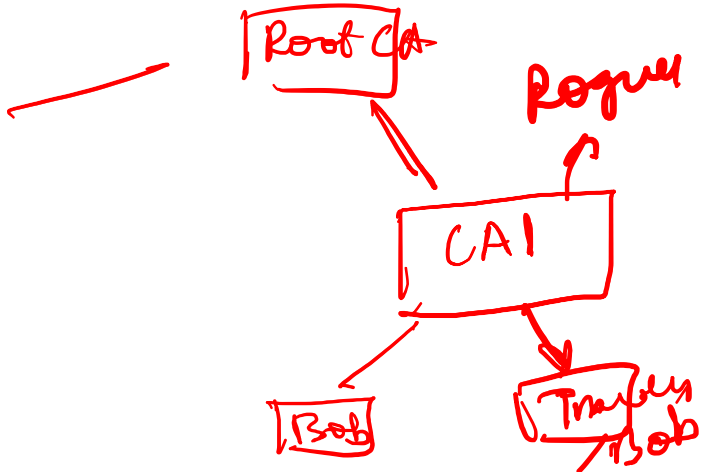
→ Some trust b/w Bob and Alice

Browsers

Pre-installed CA

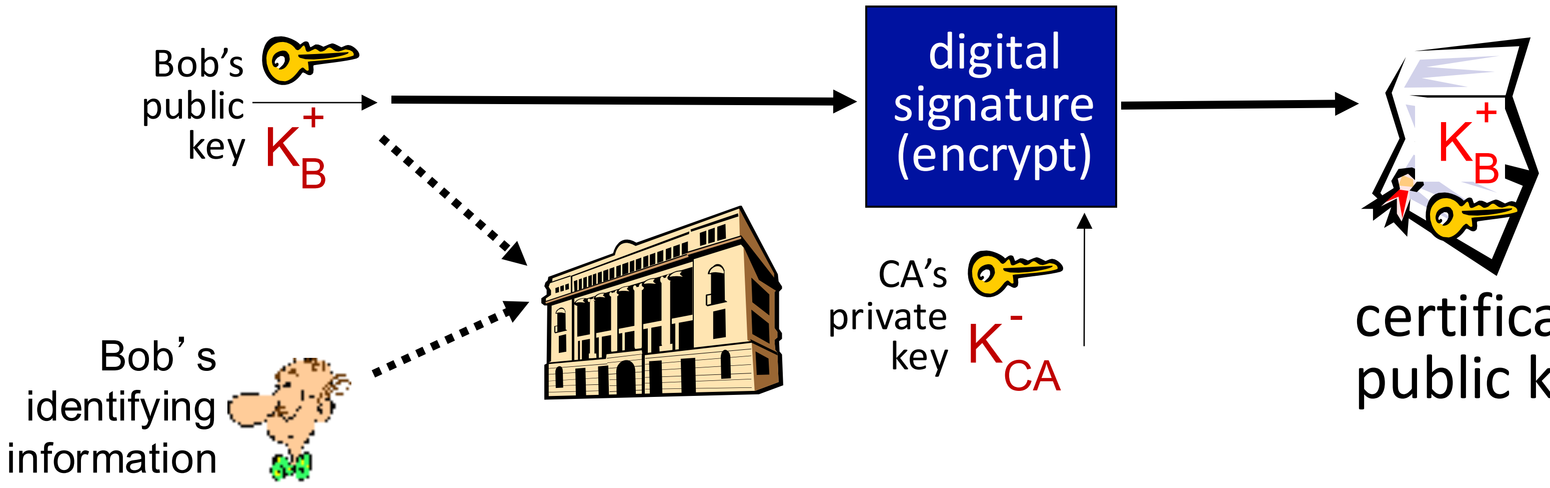
Root CA

CA1



Public key Certification Authorities (CA)

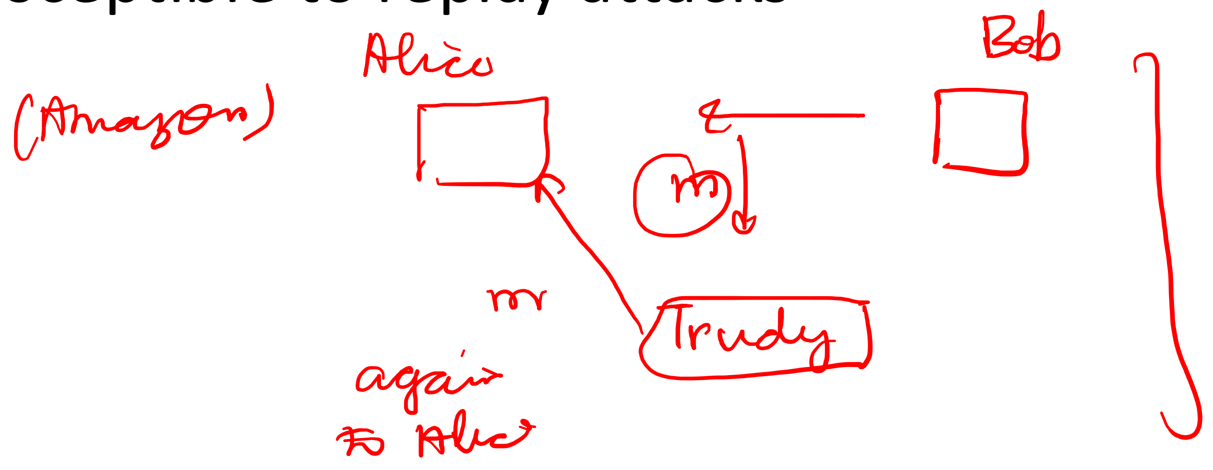
- certification authority (CA): binds public key to particular entity, E
- entity (person, website, router) registers its public key with CE provides "proof of identity" to CA
  - CA creates certificate binding identity E to E's public key
  - certificate containing E's public key digitally signed by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



certificate for Bob's public key, signed by CA

Revisiting Authentication

- Susceptible to replay attacks



→ Replay Attack

m | ( )] → Timeliness of the message

timestamp

1. Conf
2. Auth
3. Int
4. Timeliness

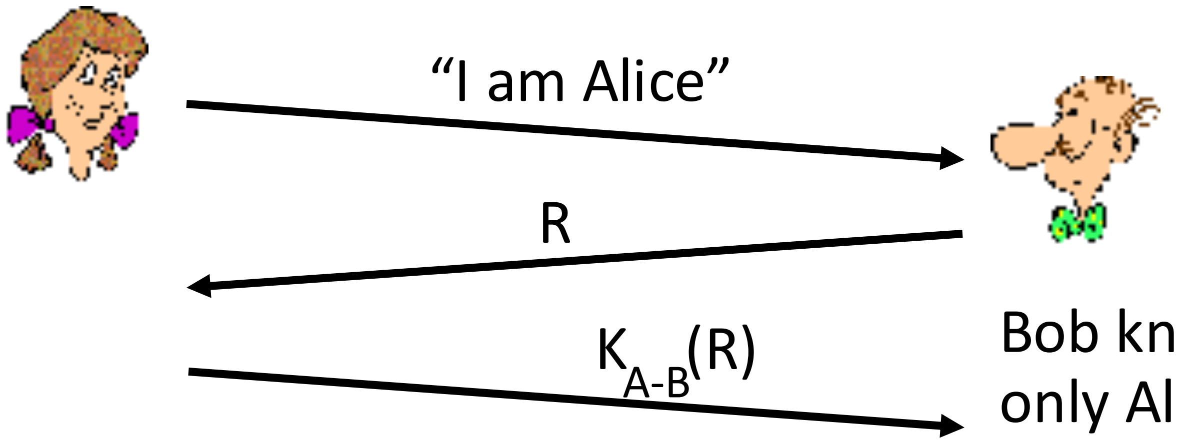
Authentication

Goal: avoid playback attack

nonce: number (R) used only once-in-a-lifetime

protocol: to prove Alice “live”, Bob sends Alice nonce, R

- Alice must return R, encrypted with shared secret key



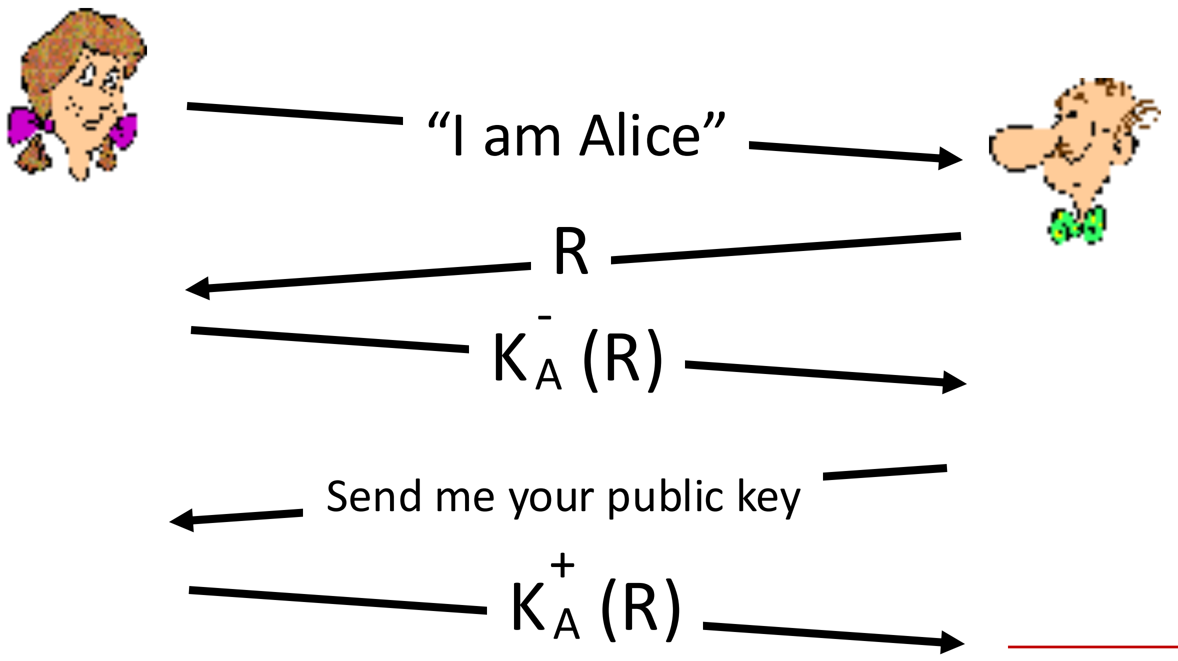
Bob know Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Failures, drawbacks?

requires shared symmetric key - can we authenticate using public key techniques?

Live Authentication using Public Keys

use nonce, public key cryptography



Bob computes

$$K_A^+(K_A^-(R)) = R$$

and knows only Alice could have the private key, that encrypted R such that

$$K_A^+(K_A^-(R)) = R$$