Securing Wireless Networks, COMP4337/9337
RSA/DH, Certification Authority, Authentication

Never Stand Still

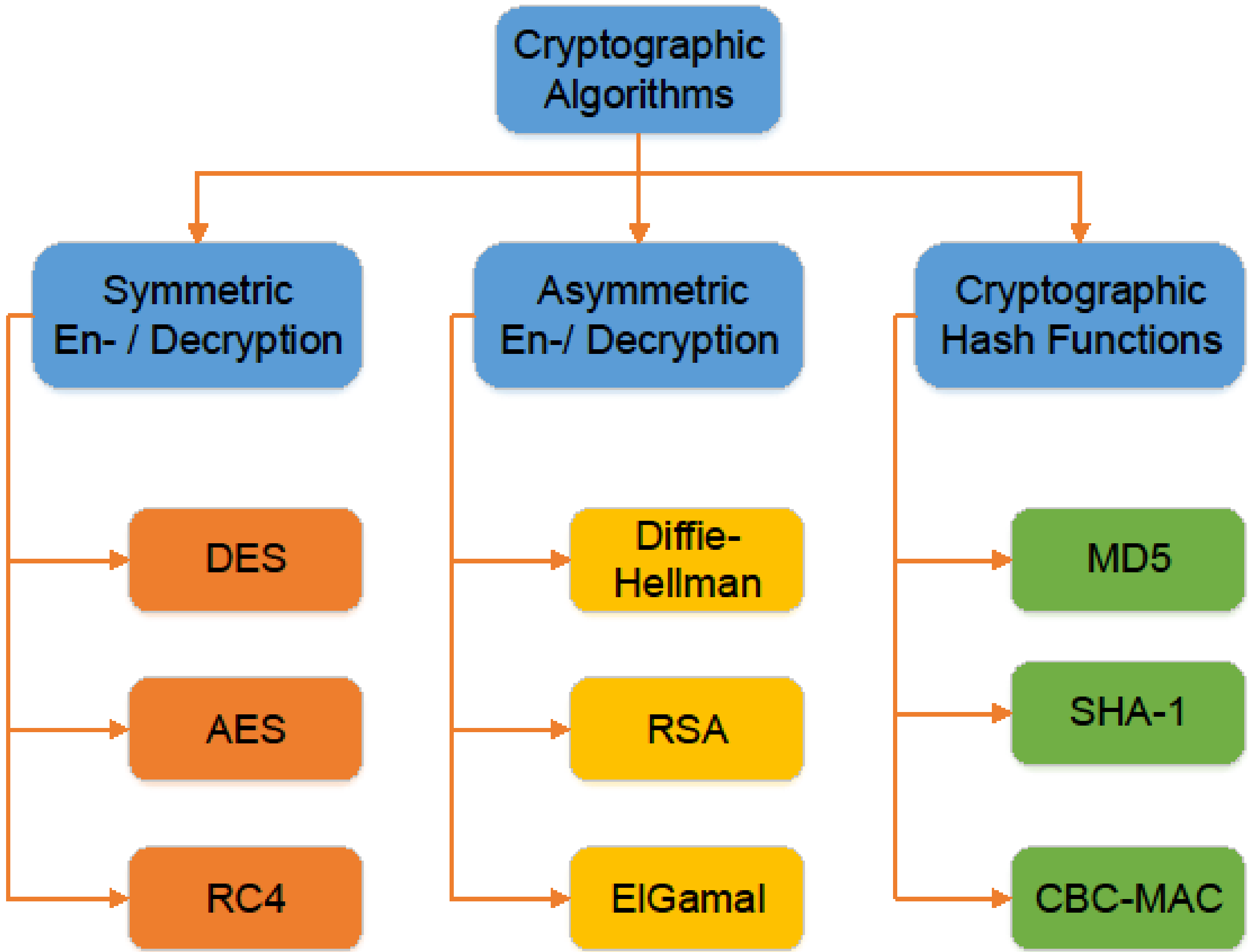
Professor Sanjay K. Jha

School of Computer Science and Engineering, UNSW

Today's Agenda – Part 1

- Key Distribution RSA, Deffie Hellman
- Certification Authority and X.509
- Authentication Recap
- Application Layer Security (PGP)

Recap



Cryptographic Algorithms
Symmetric En- / Decryption
Asymmetric En-/ Decryption
Cryptographic Hash Functions
DES
AES
RC4
Diffie-Hellman
RSA
ElGamal
MD5
SHA-1
CBC-MAC

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^-

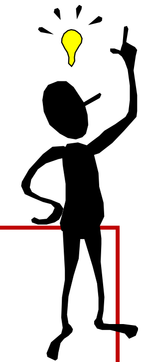
RSA: Rivest, Shamir, Adelson algorithm

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 crypto



- 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

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 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"). E.g.: 4 and 9 are relatively prime. 6 and 9 are not.

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 bit pattern, m $(m < n)$, compute

   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by 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).

encrypt:

letter	m	m^e	$c = m^e \bmod n$
l	12	1524832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	481968572106750915091411825223071697	12	l

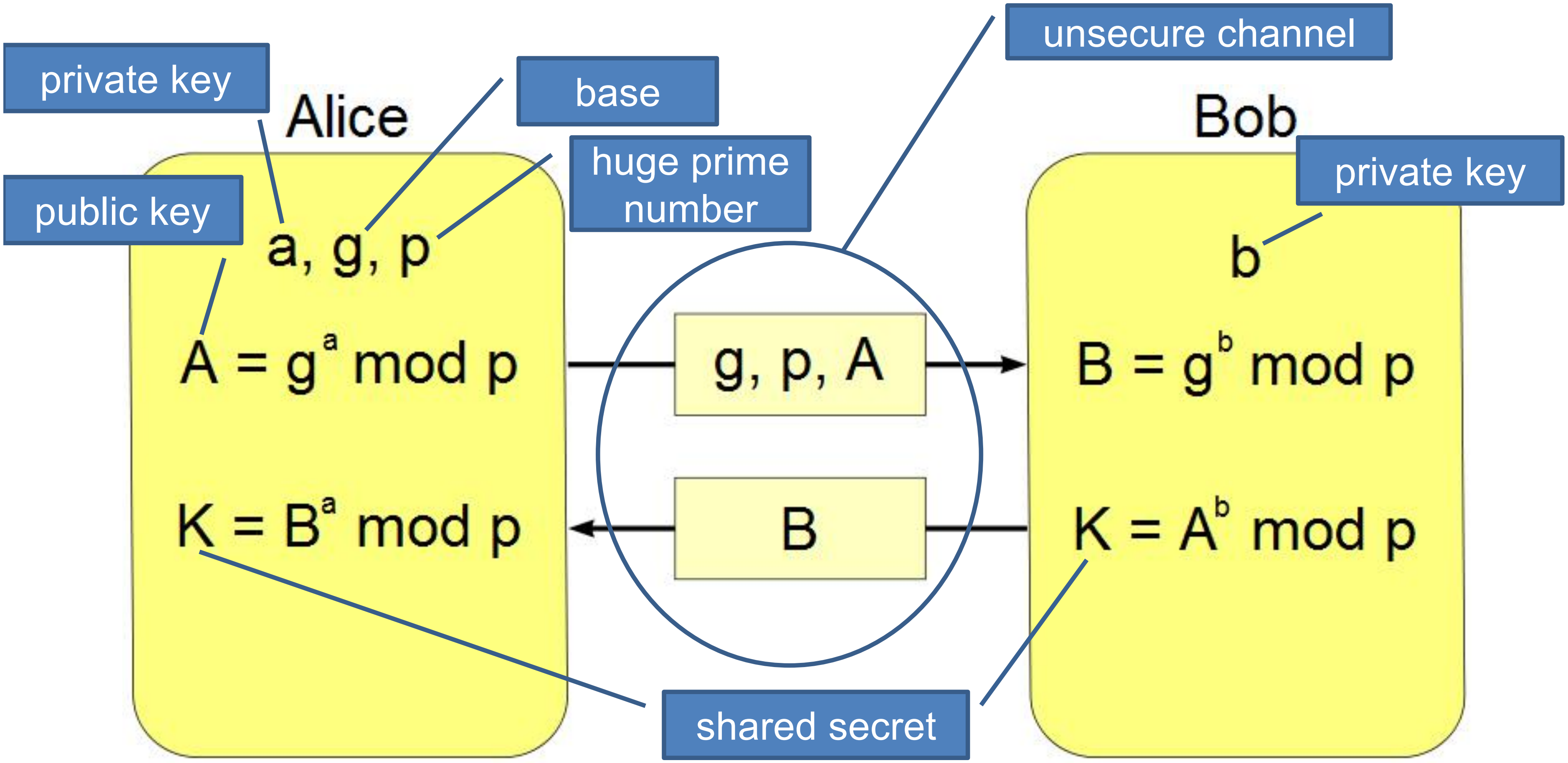
RSA: another important property

The following property will be *very* useful later:

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

Result is the same!

Diffie-Hellman key exchange



$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

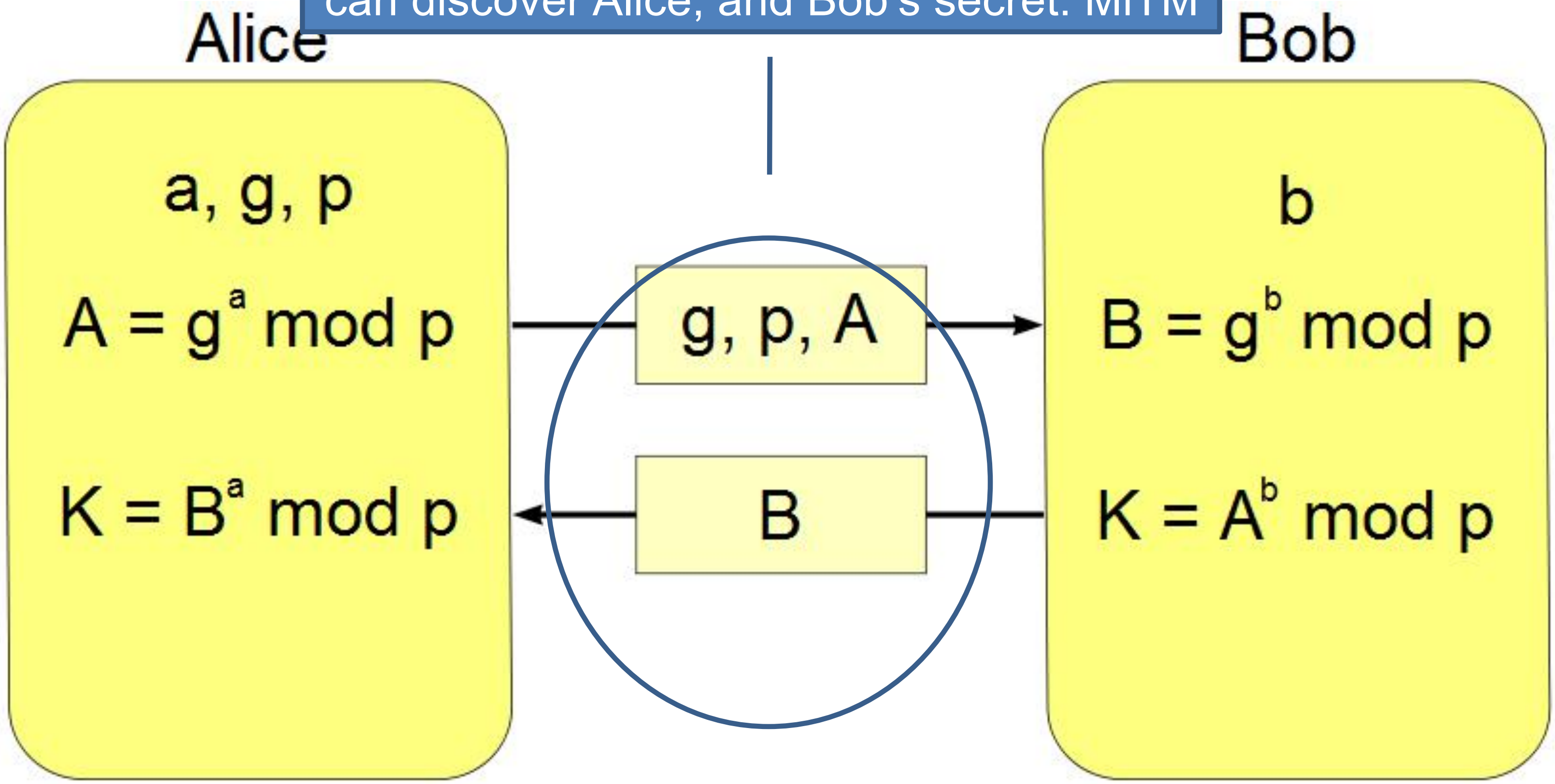
Alice's private key = 5, Bob's private key = 4, g=3, p=7

Alice's public key = 3^5 mod 7 = 5, Bob's public key = 3^4 mod 7 = 4

Alice's shared key = 4^5 mod 7 = 2, Bob's shared key = 5^4 mod 7 = 2

Diffie-Hellman key exchange

If Eve can tamper with the channel, she can discover Alice, and Bob's secret: MiTM



$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

Alice's private key = 5, Bob's private key = 4, g=3, p=7

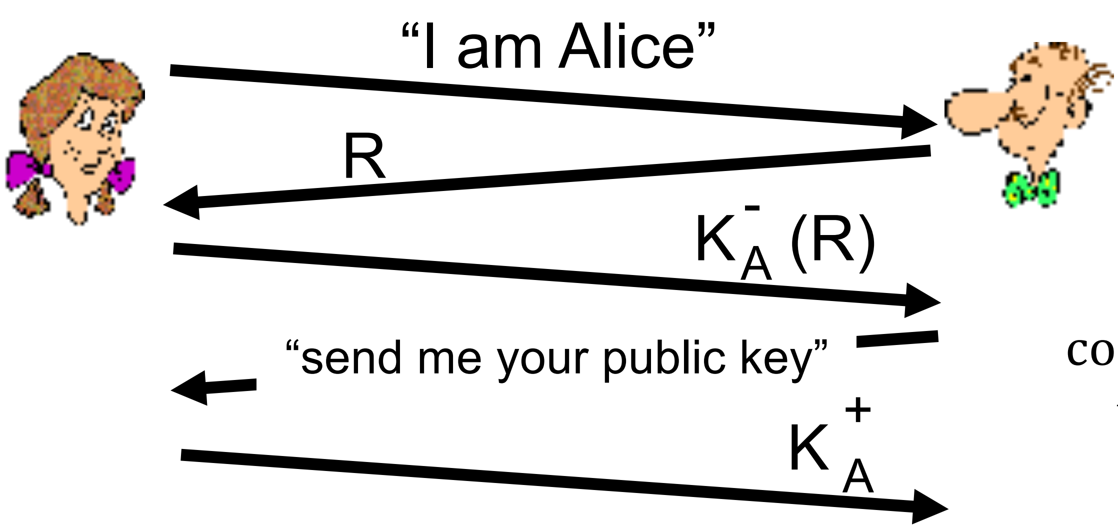
Alice's public key = 3^5 mod 7 = 5, Bob's public key = 3^4 mod 7 = 4

Alice's shared key = 4^5 mod 7 = 2, Bob's shared key = 5^4 mod 7 = 2

Public-key certification

- motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it's Bob's public key
  - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
  - Bob doesn't even like pepperoni

Kurose/Ross Authentication: ap5.0



Bob computes

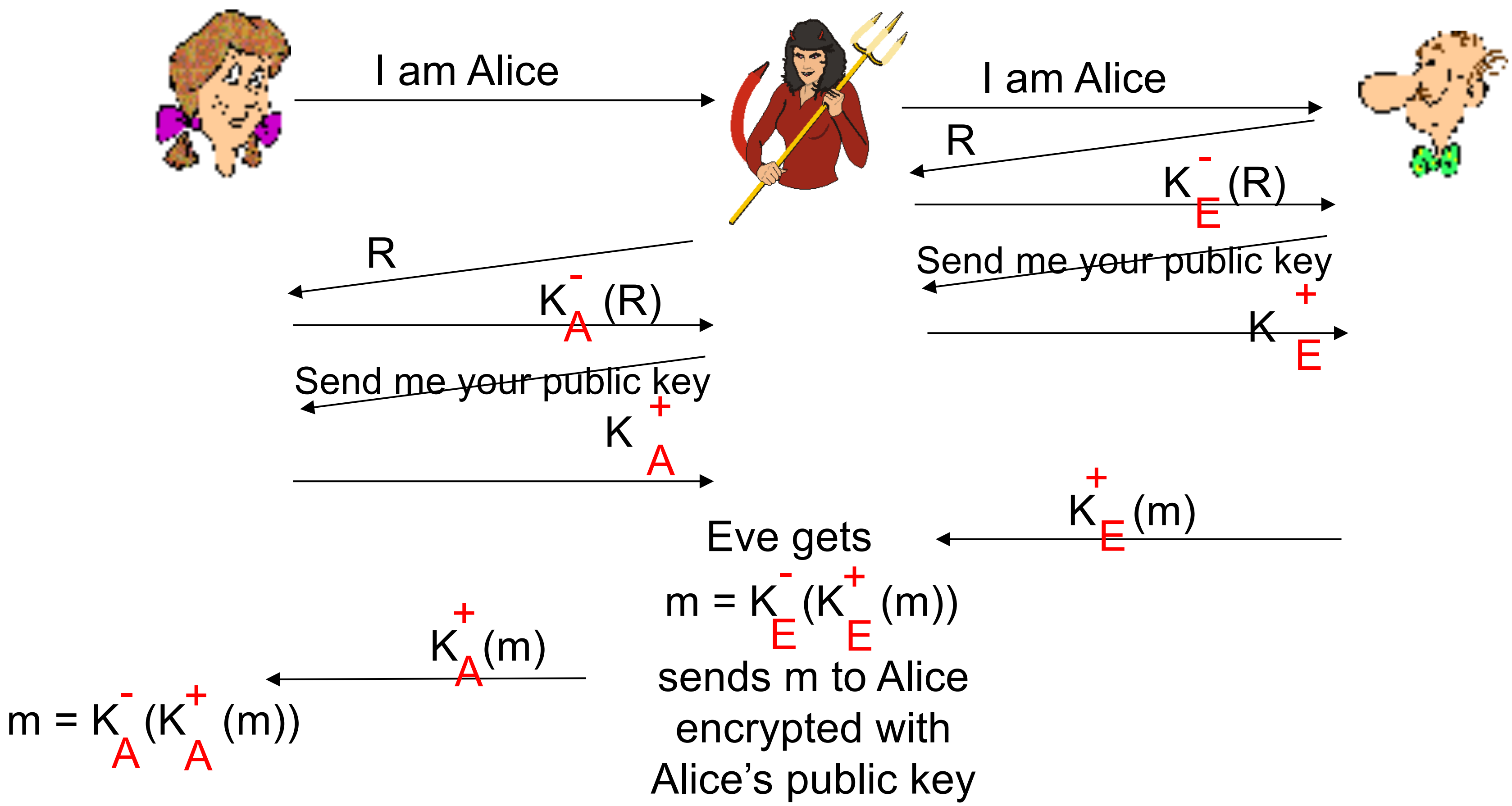
$$K_A^+(K_A^-(R)) = R$$

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

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

MiTM: Man in the Middle

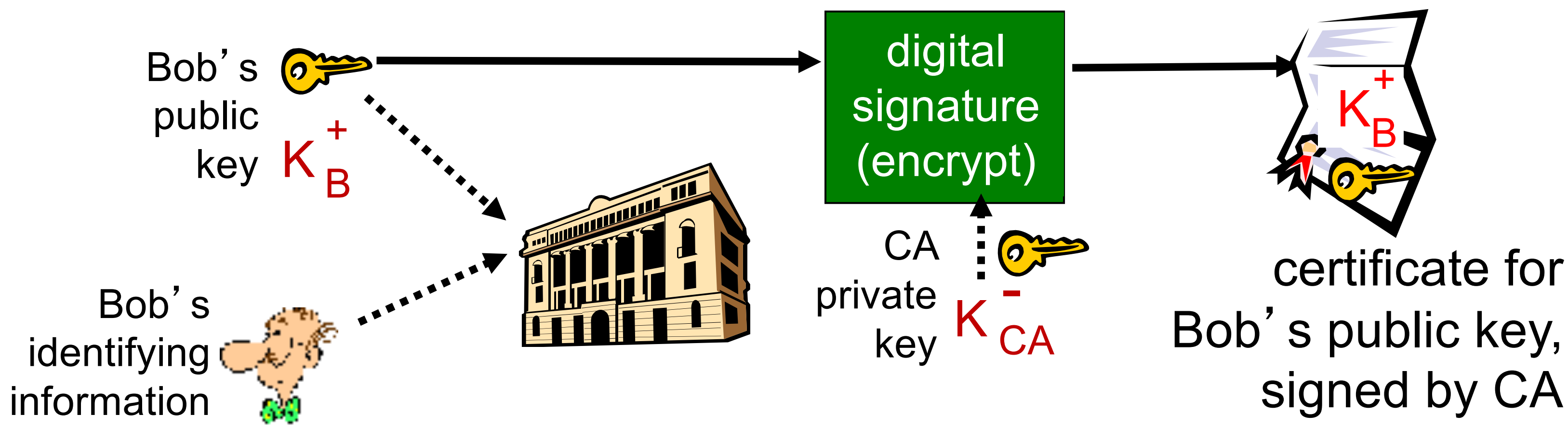
man (or woman) in the middle attack: Eve poses as Alice (to Bob) and as Bob (to Alice)



Certification authorities

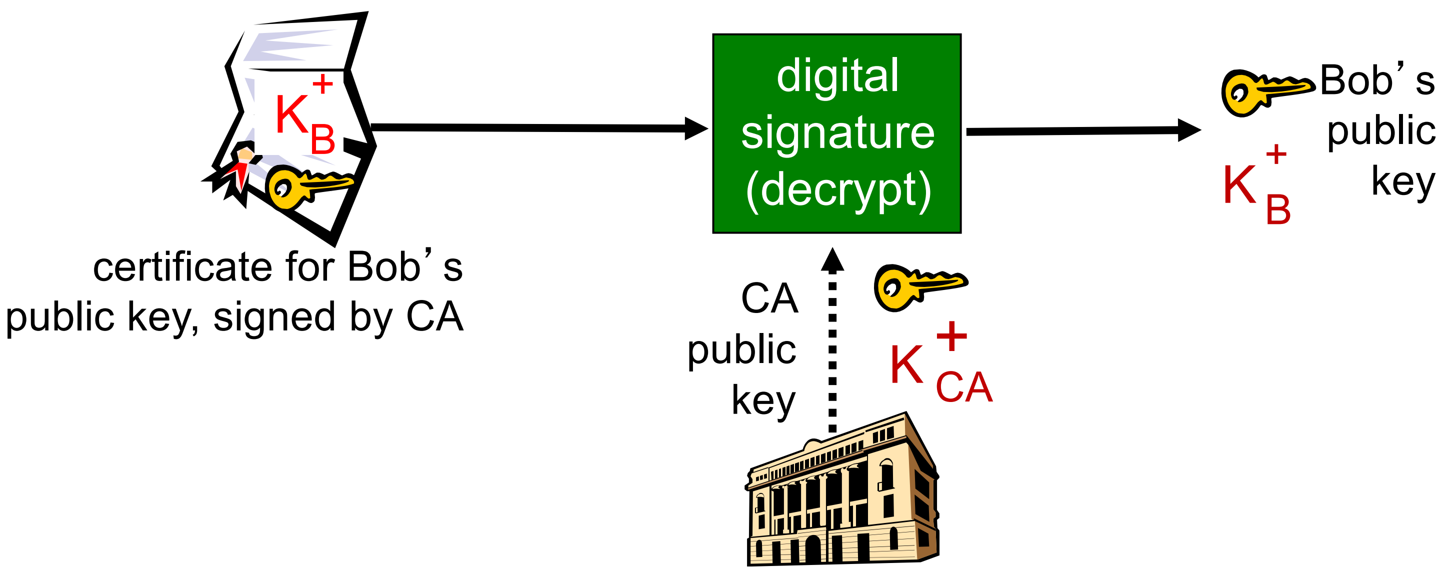
certification authority (CA): binds public key to particular entity, E.

- 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"



Certification authorities

- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



Public Key Distribution of Secret Key

- Prepare a message
- Encrypt that message using conventional encryption using one time session key
- Encrypt the session key using public-key encryption with Alice's **public key**
- Attach the encrypted session key to the message and send it to Alice
- Only Alice can decrypt the session key
- Bob has obtained Alice's public key by means of Alice's **public-key certificate**, must be a valid key

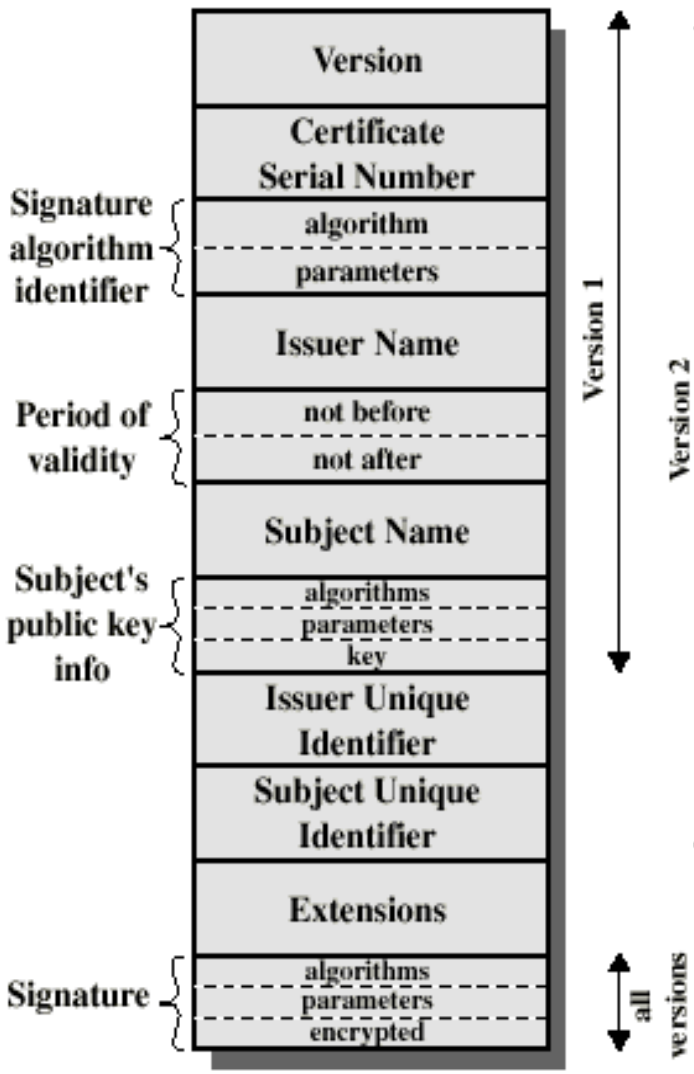
X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use but not mandatory.
- Digital Signature is assumed to use Hash algorithm
- Digital Certificate: user's id, public-key and CA information as input to hash function. Hash is then encrypted with CA's private key to produce **Digital Certificate**

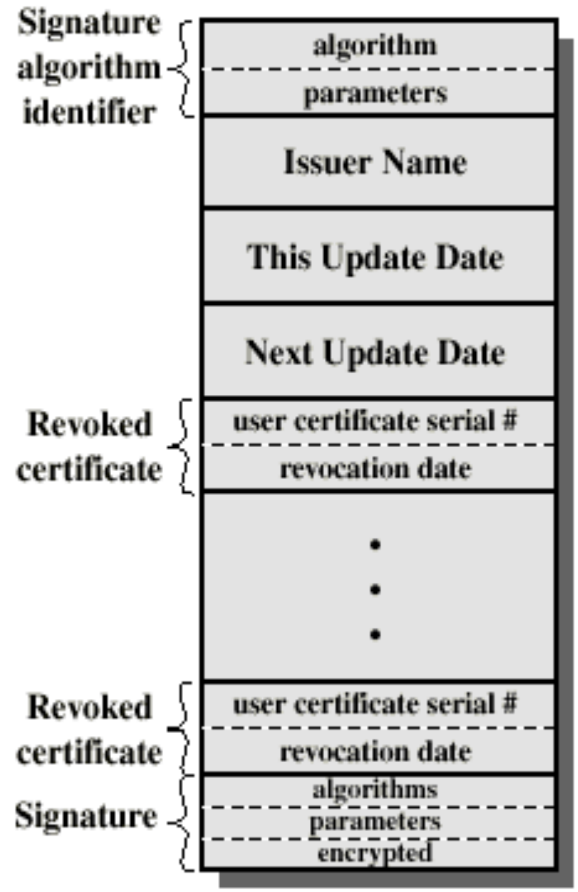
No need to memorise

Read: Stallings ch4 for a quick overview

X.509 Formats



(a) X.509 Certificate



(b) Certificate Revocation List

Obtaining a User's Certificate

- Characteristics of certificates generated by CA:
  - Any user with access to the public key of the CA can recover the user public key that was certified.
    - User can independently calculate hash, decrypt digital certificate using CA's public key, extract hash and compare if hashes match.
  - No part other than the CA can modify the certificate without this being detected.
- Certificates stored in a Directory server – not part of standard.

Distributed Directory

- Users can be registered with a CA and would know its public Key
- Now if A got its certificate from CA X1 and B got it from CA X2.
- If A doesn’t know CA X2’s public key, it can’t trust B’s certificate issued by CA X2.
- However, if the two CA’s have securely exchanged their public keys, then it can work.
  - A obtains the certificate of X2 signed by X1
  - A knows securely X1’s public key
  - A obtains X2’s public key from certificate and can verify using X1’s signature on certificate
  - A can now get B’s certificate from CA X2.
  - Since it has trusted public key for CA X2, things work as usual.

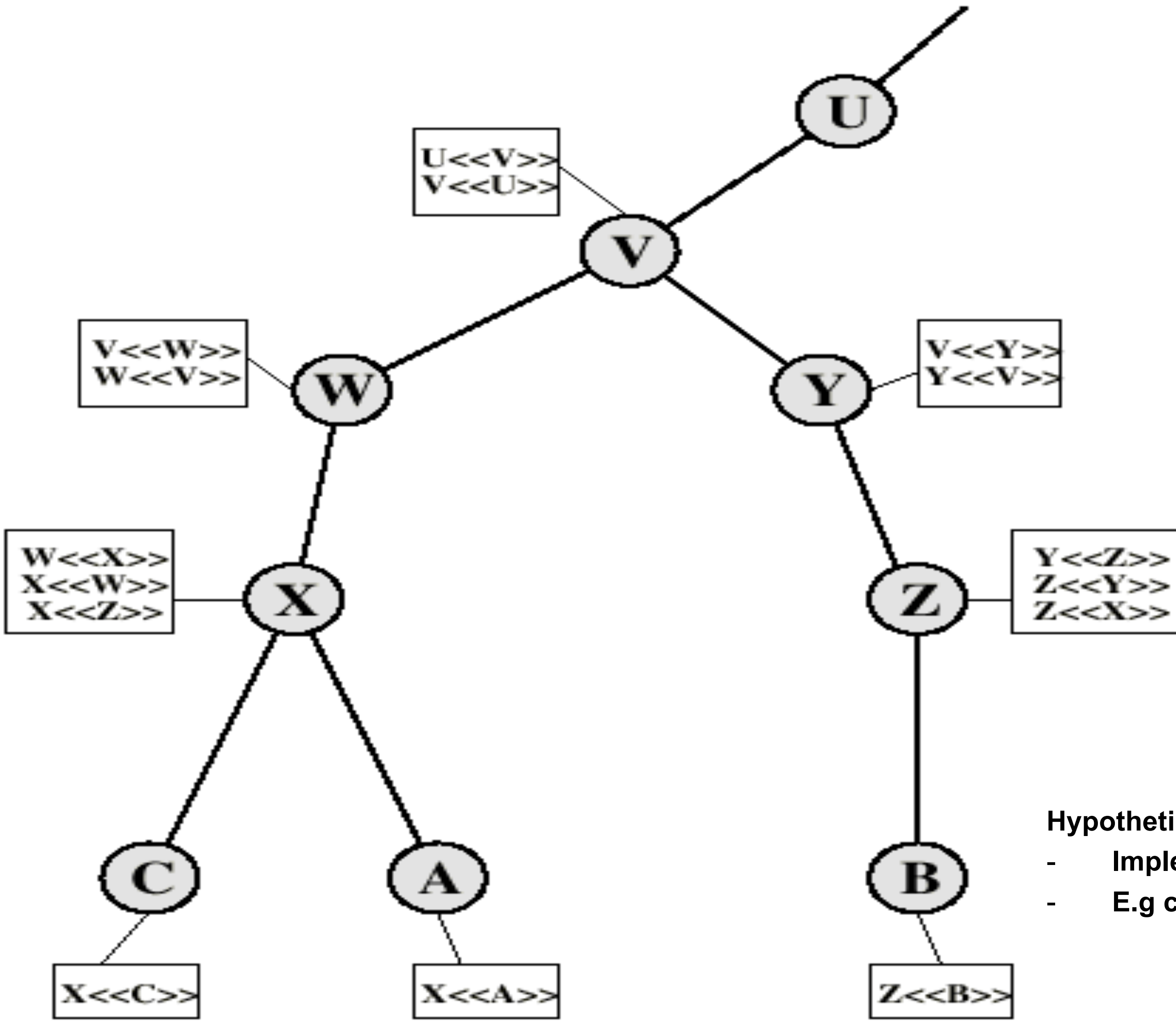
Distributed Directory: Certificate Chain

- Notation $Y <<X>>$ Certificate of user *X* issued by authority *Y*
- A obtains B's public key using the following X.509 notation

  $X_1 << X_2>> X_2 << B>>$
- B obtains A's public key using the following X.509 notation

  $X_2<< X_1>> X_1 << A>>$
  - *Arbitrary chain is possible as long as consecutive pair* (X_n, X_{n+1}) *of CAs have exchanged certificates securely*

X.509 CA Hierarchy



Hypothetical Example

- Implementations may vary
- E.g cache entries

Hierarchy of CAs

- Previous figure: Connected Circles hierarchical relationship, boxes shows certificates maintained in each CA's directory
  - Forward Certs: Certs of X generated by other CAs (e.g at circle X, W<<X>>) – PARENT
  - Reverse Certs: Certs generated by X for others. (e.g. at circle X, X<<C>> X<<A>>) - CHILD
- A can acquire the following Certs from the directory to establish as certification to B

  X <<W>> W <<V>> V <<Y>> Y<<Z>>Z<<B>>

  (Try to get A's certificate)

Revocation of Certificates

- Reasons for revocation:
  - The users secret key is assumed to be compromised.
  - The user is no longer certified by this CA.
  - The CA's certificate is assumed to be compromised.
- X.509 has a new version 3 with some recommendations for improvement – read in your own time if interested

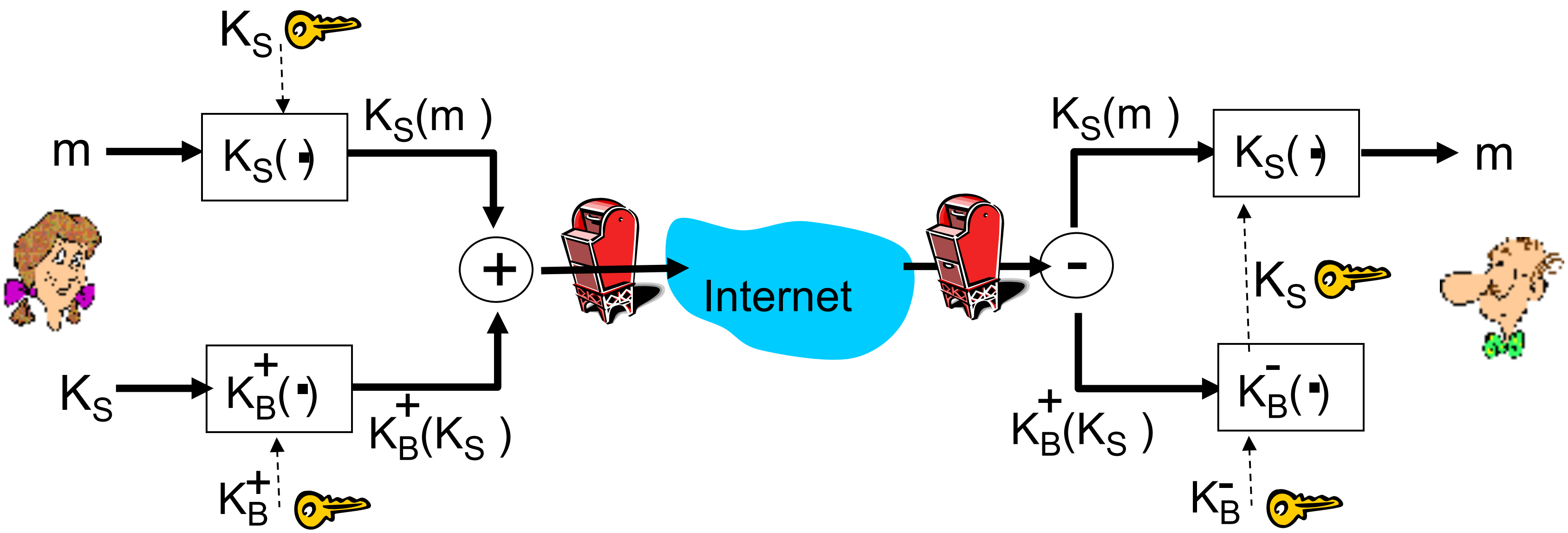
Application Layer Security

- Security can be implemented at various layers
- Application Layer Security
  - *A quick glimpse at secure email (PGP)*

- Secure Socket Layer (SSL) / Transport Layer Security (TLS)
  - Part 2 this week
- Network Layer Security
  - IPSec next week
- Link Layer Security
  - WLAN covered earlier, more on Enterprise later

Secure e-mail: Alice

❖ Alice wants to send confidential e-mail, m, to Bob.



Alice:

❖ generates random *symmetric* private key, K_S

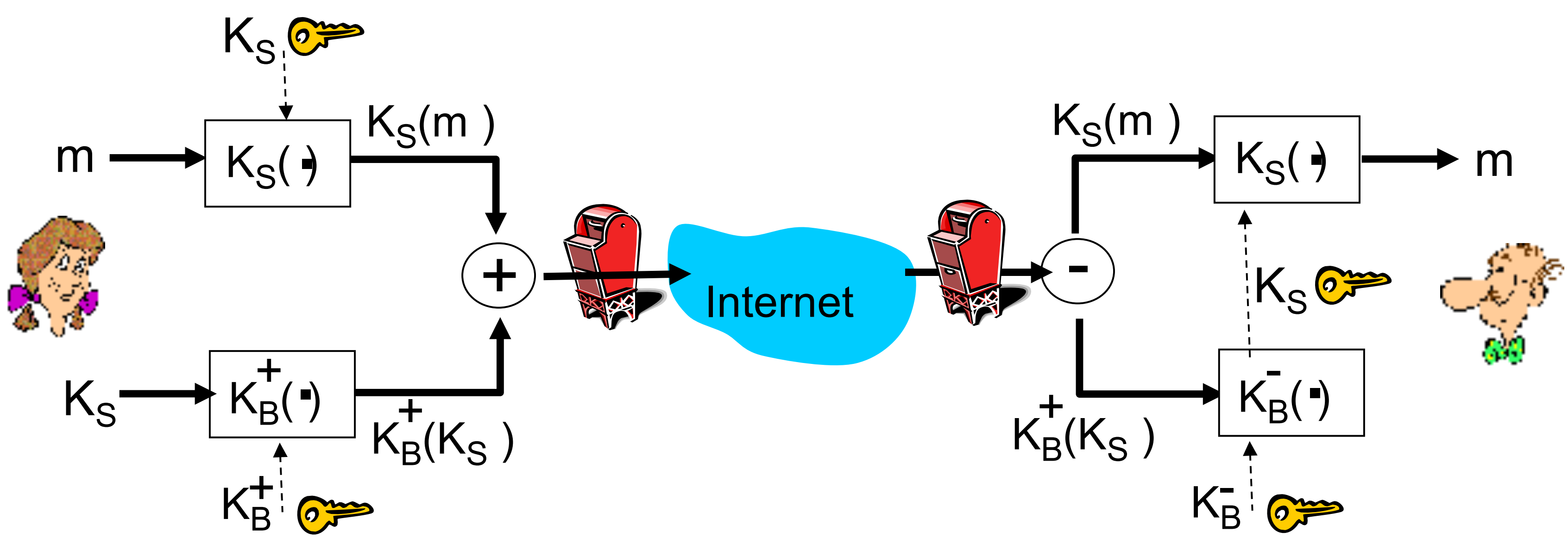
❖ encrypts message with K_S (for efficiency)

❖ also encrypts K_S with Bob's public key

❖ sends both $K_S(m)$ and $K_B(K_S)$ to Bob

Secure e-mail: Bob

- Alice wants to send confidential e-mail, m, to Bob.

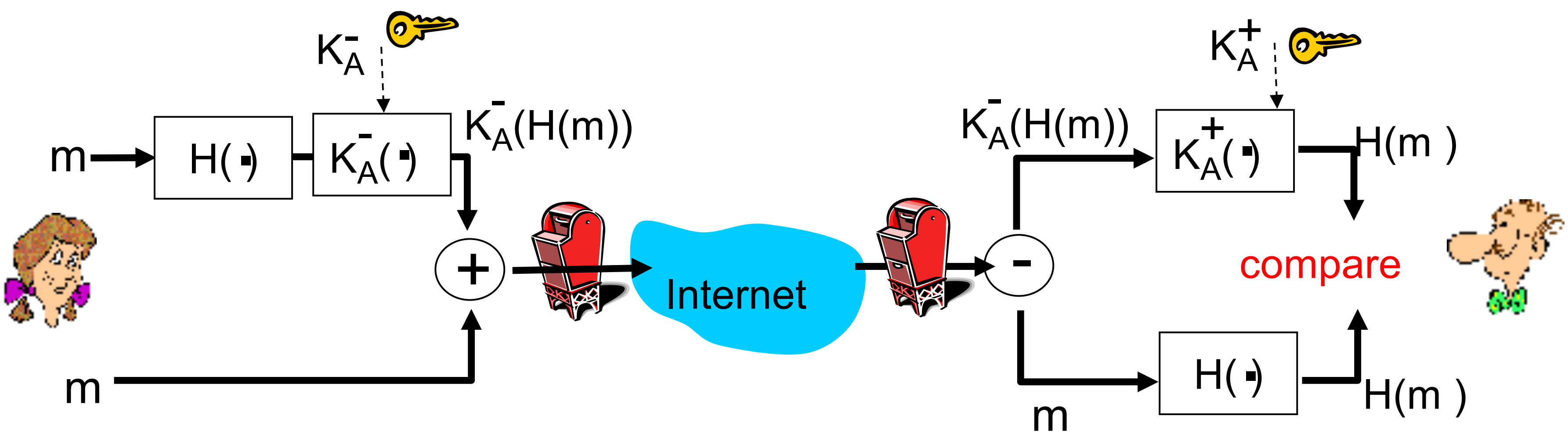


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

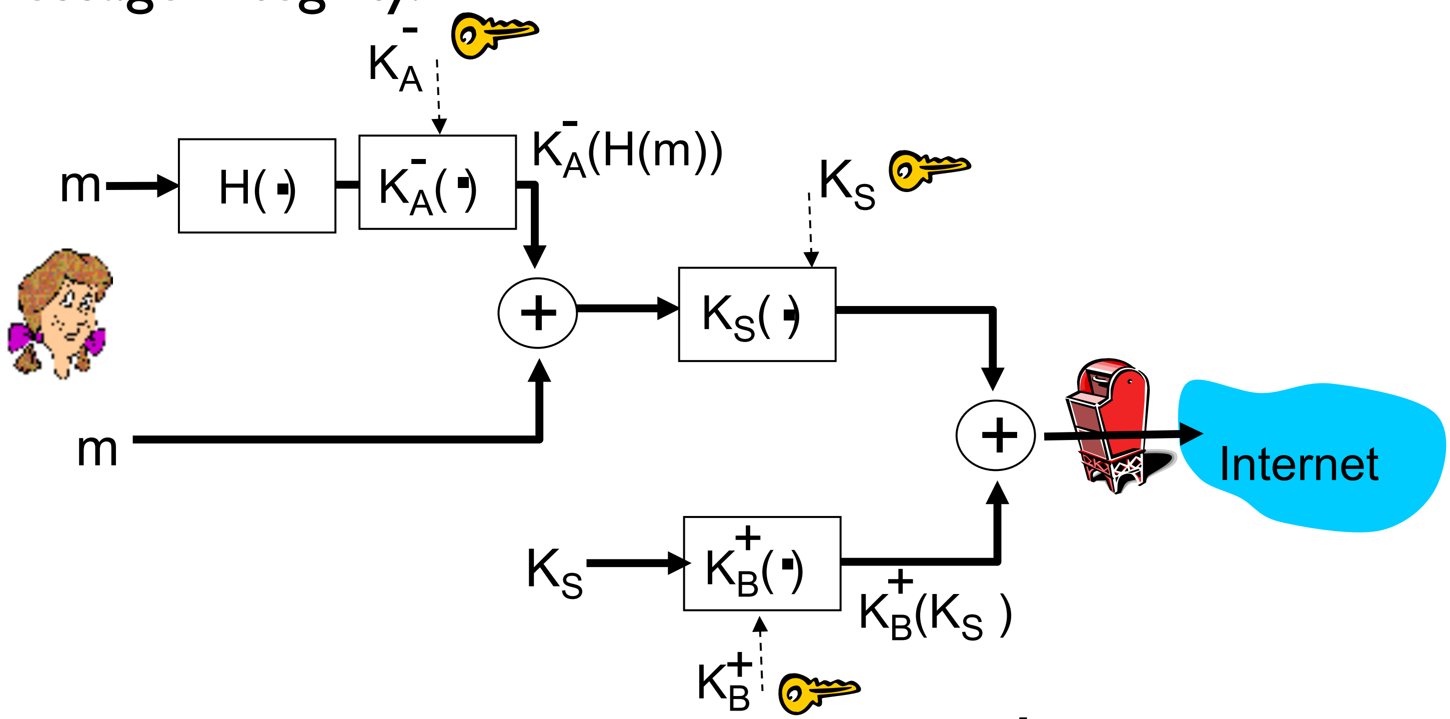
- Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key (Exercise: see how Bob's side works)

UNSW AUSTRALIA CySPri Laboratory

Acknowledgements

- Network Security Essentials: Stallings, Chapter 4 provided by Henric Johnson, Blekinge Institute of Technology,Sweden (Please refer to Section 4.3 and 4.4 from Staillings)

- Computer Networking A top-Down Approach: Jim Kurose and Keith Ross, chapter 8 (several lecture foils provided by authors)

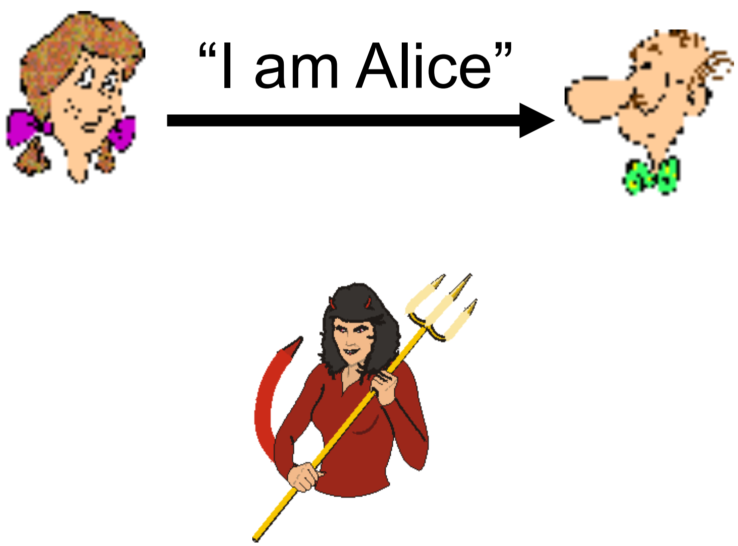
Revise Authentication Basics

- Next set of foils are for self study you have already done in 3331/9331 (Kurose-Ross Ch8)
- These are basic building blocks and please revise them as they will help understand material covered in this subject.

Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”

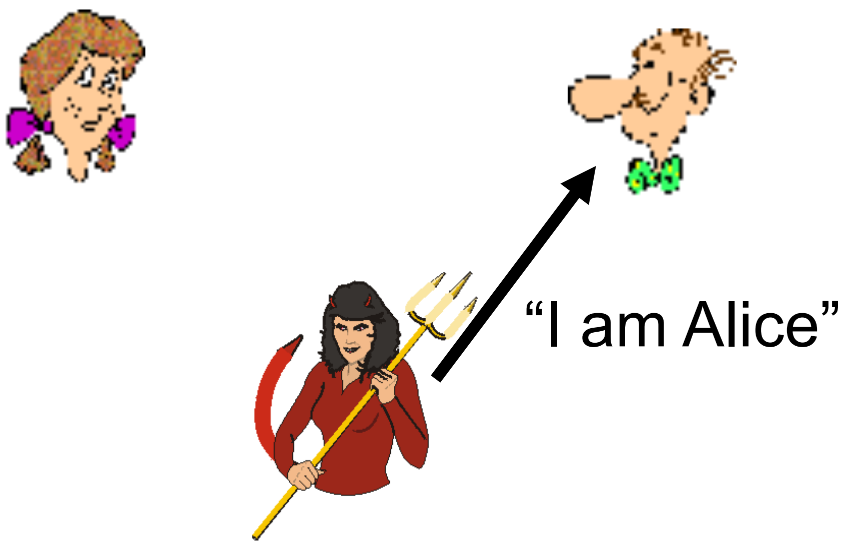


Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"



In a network,
Bob can not "see" Alice, so
Eve simply declares
herself to be Alice

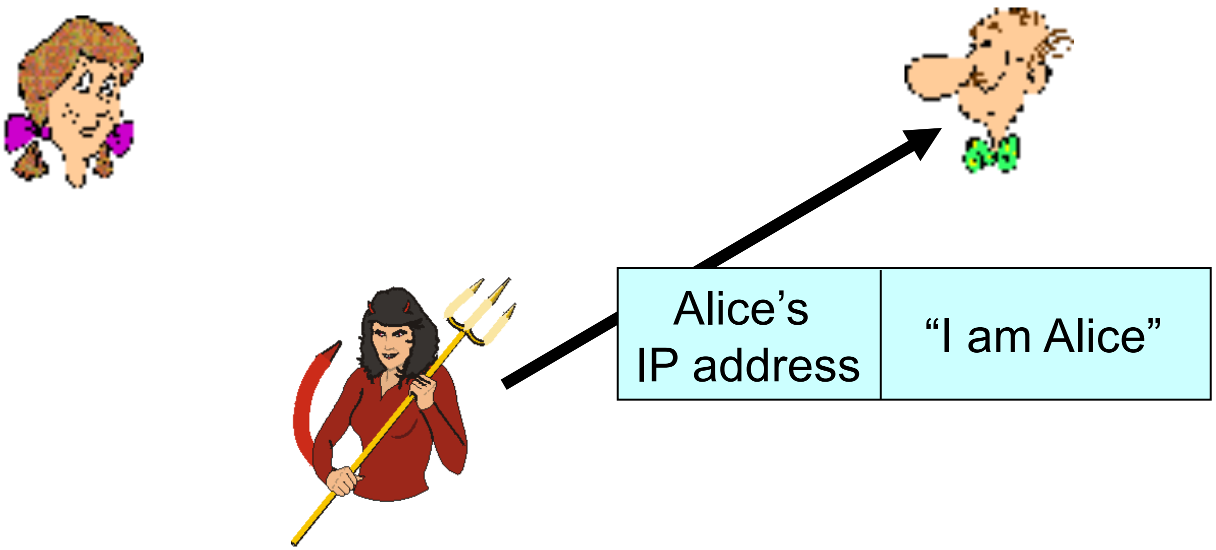
Authentication: another try

Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address

Failure scenario??

Authentication: another try

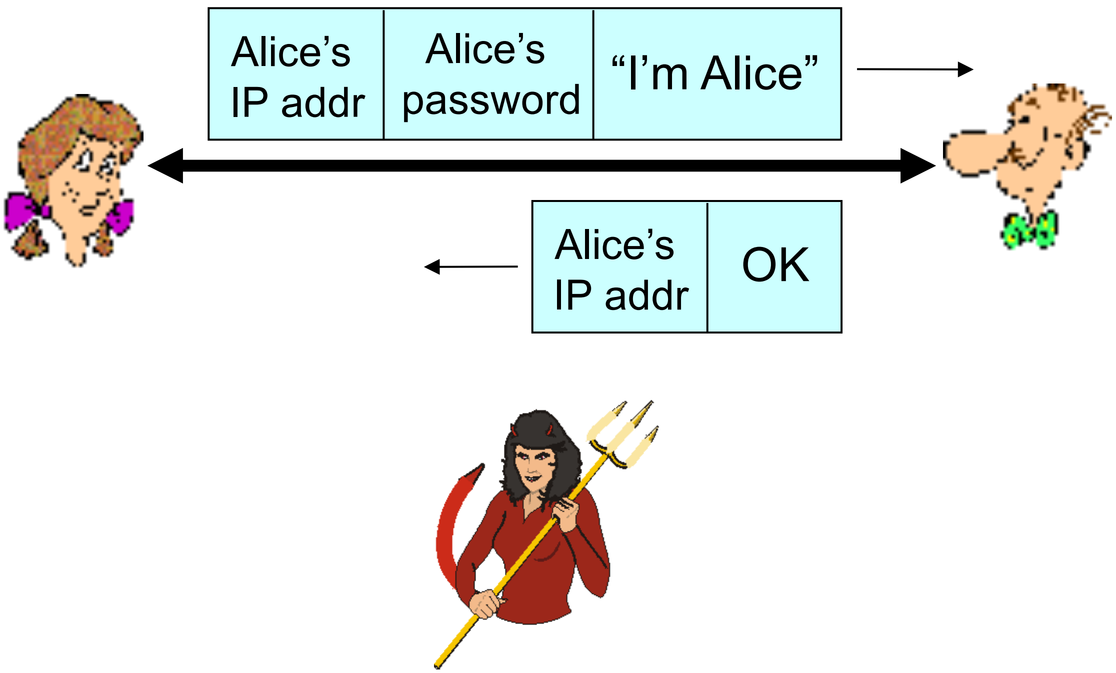
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Eve can create a packet “spoofing” Alice’s address

Authentication: another try

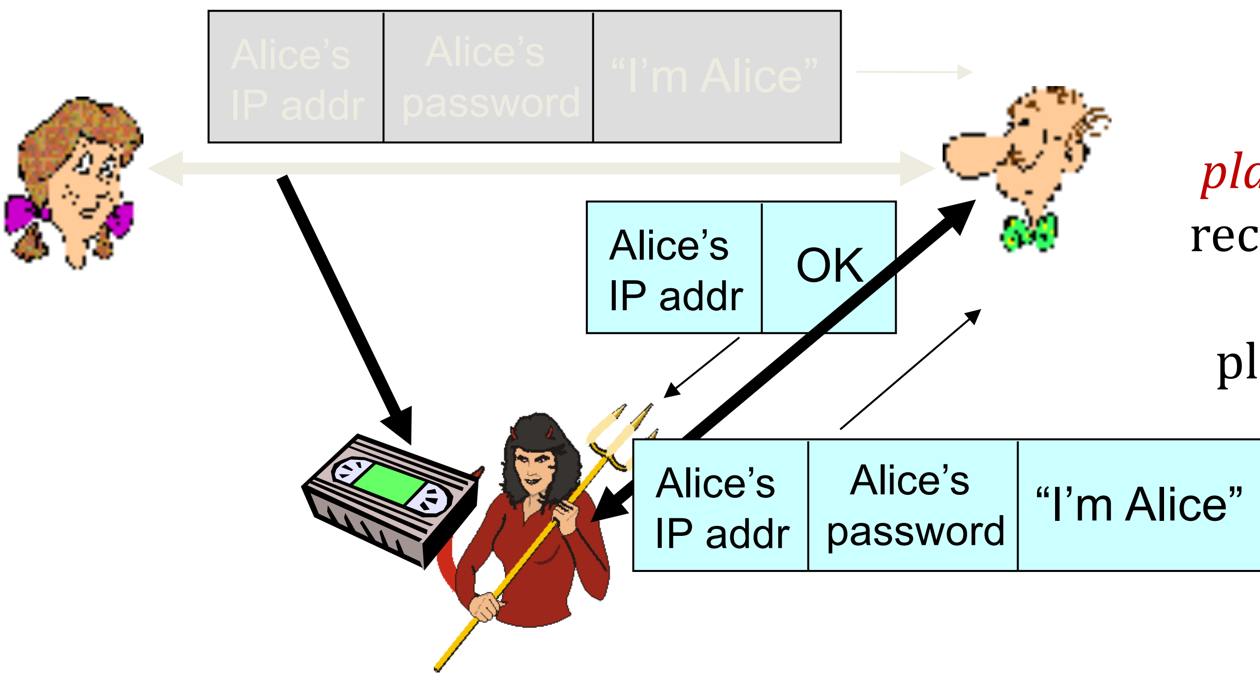
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

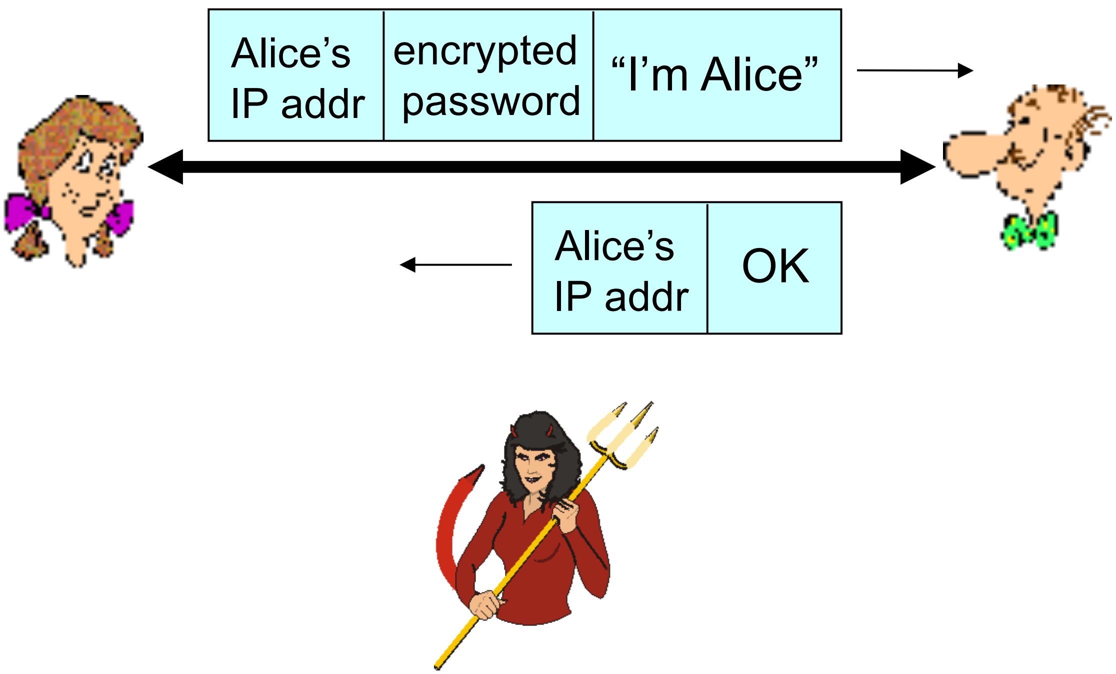
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



playback attack: Eve records Alice's packet and later plays it back to Bob

Authentication: yet another try

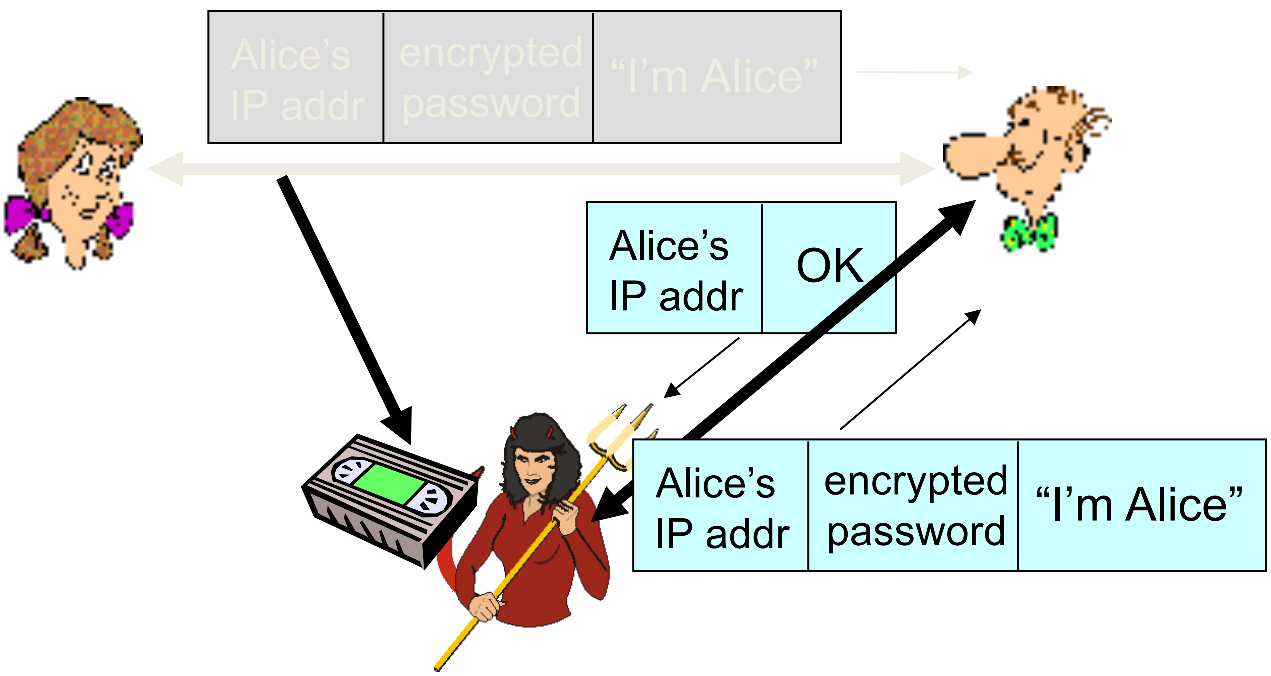
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



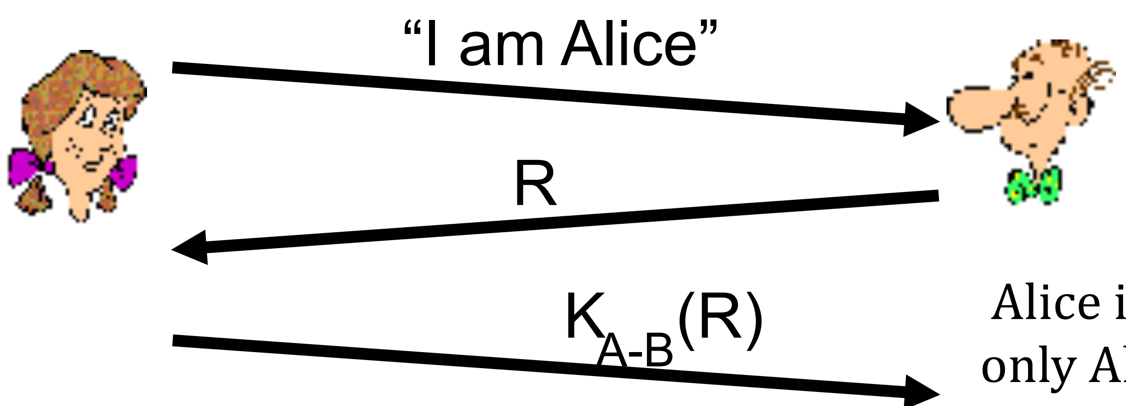
record
and
playback
still works!

Authentication: yet another try

Goal: avoid playback attack

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

ap4.0: to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key



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

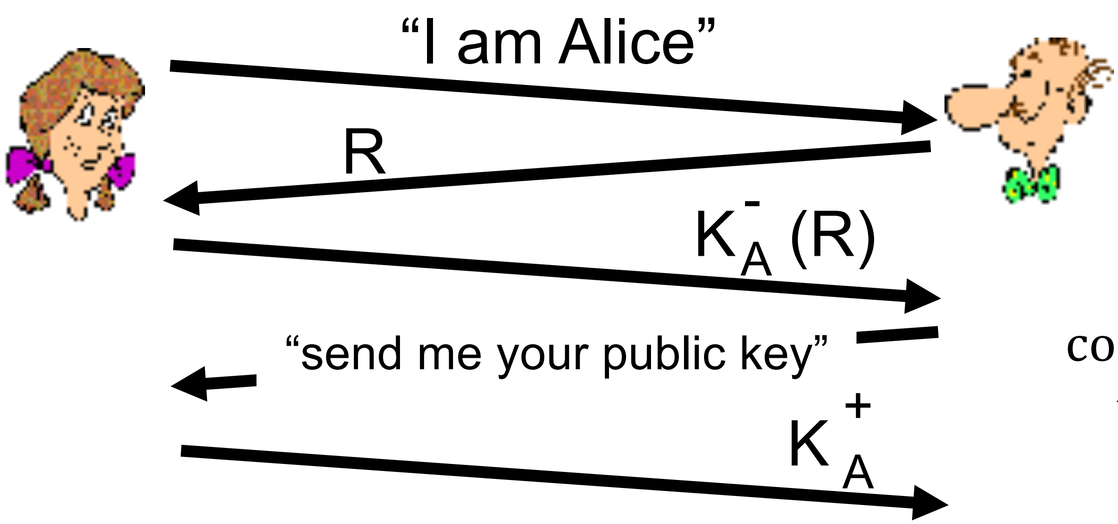
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Bob computes

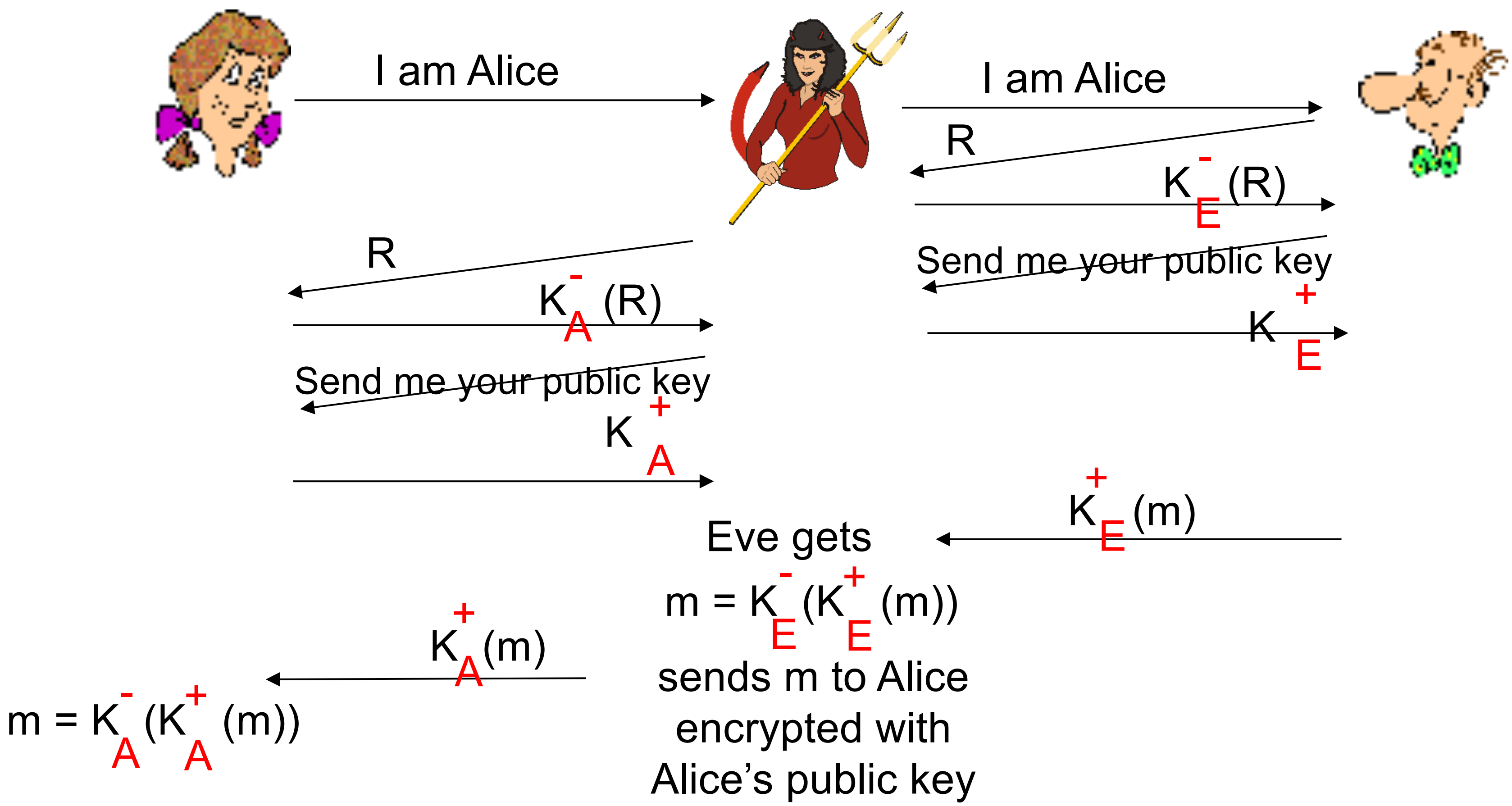
$K_A^+(K_A^-(R)) = R$

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

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

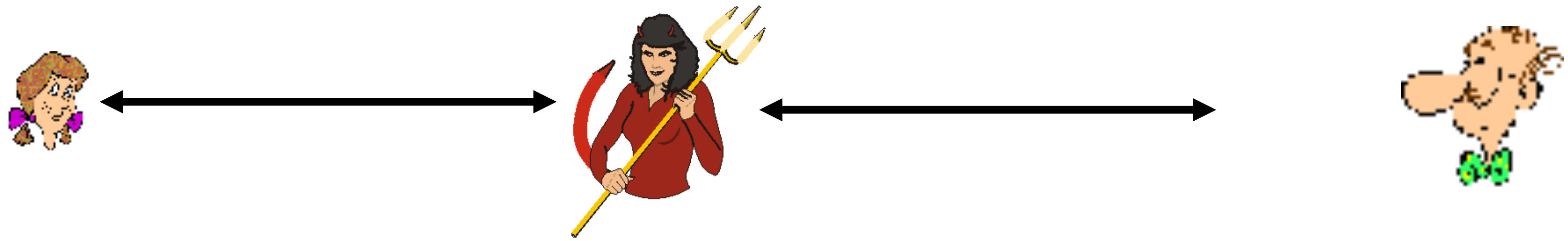
ap5.0: security hole

man (or woman) in the middle attack: Eve poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

man (or woman) in the middle attack: Eve poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Eve receives all messages as well!