CS 3251- Computer Networks 1: Authentication

Professor Patrick Traynor

11/19/13

Lecture 25

Announcements

- Homework 3 was due 30 seconds ago.
- Project 3 is being graded.
    - I hope to have scores posted no later than next week.
- Project 4 is due next week...
    - ...‘nuff said...

Last Time

- What are the four (general) properties security tries to provide?
- The Caeser Cipher is an example of what kind of cryptographic cipher?
- What are the differences between symmetric and asymmetric (public key) cryptography?

Diffie-Hellman - Class Exercise

- Select a partner.
- Setup: Pick a prime number p and a base g ($<p$)
  - $p=13$, $g=4$
- Each partner chose a private value x ($<$p-1)
- Generate the following value and exchange it.

$$y = g^x \; mod \; p$$

- Now generate the shared secret z:

$$z = y^x \; mod \; p$$

- You should have both calculated the same value for z. This is your key!

Chapter 8 roadmap

8.1 What is network security?

8.2 Principles of cryptography

8.3 Message Integrity, Authentication

8.4 Securing e-mail

8.5 Securing TCP connections: SSL

8.6 Network layer security: IPsec

8.7 Securing wireless LANs

8.8 Operational security: firewalls and IDS

Message Integrity

- Bob receives msg from Alice, wants to ensure:
  - message originally came from Alice
  - message not changed since sent by Alice
- Cryptographic Hash:
  - takes input m, produces fixed length value, H(m)
    - e.g., as in Internet checksum... but a bit different...
  - computationally infeasible to find two different messages, x, y such that H(x) = H(y)
    - equivalently: given m = H(x), (x unknown), can not determine x.
    - note: Internet checksum fails this requirement!

Internet Checksum: Poor Crypto Hash Function

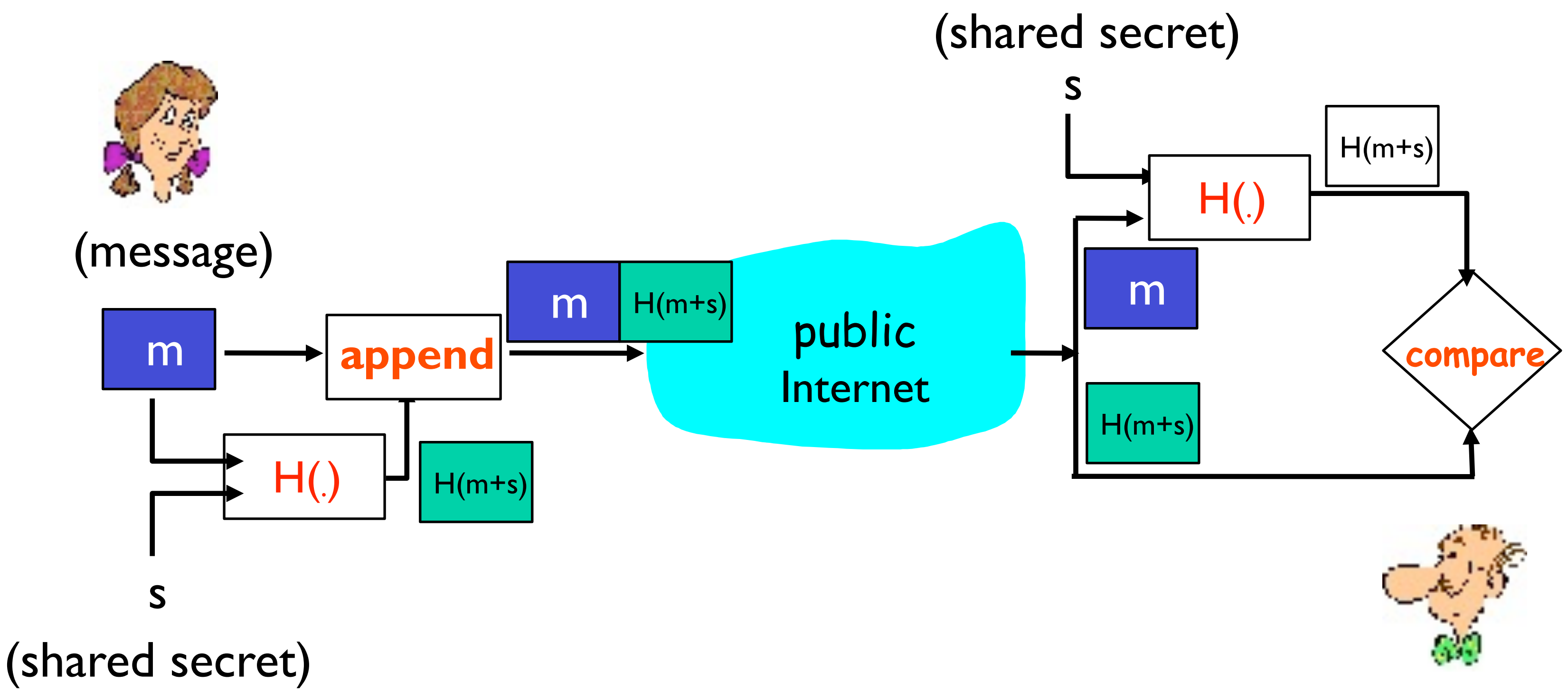
- Internet checksum has some properties of hash function:
  - produces fixed length digest (16-bit sum) of message
  - is many-to-one
- But given a message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 4F 42
	B2 C1 D2 AC

message	ASCII format
I O U 9	49 4F 55 39
0 0 . 1	30 30 2E 31
9 B O B	39 42 4F 42
	B2 C1 D2 AC

different messages but identical checksums!

Message Authentication Code (MAC)

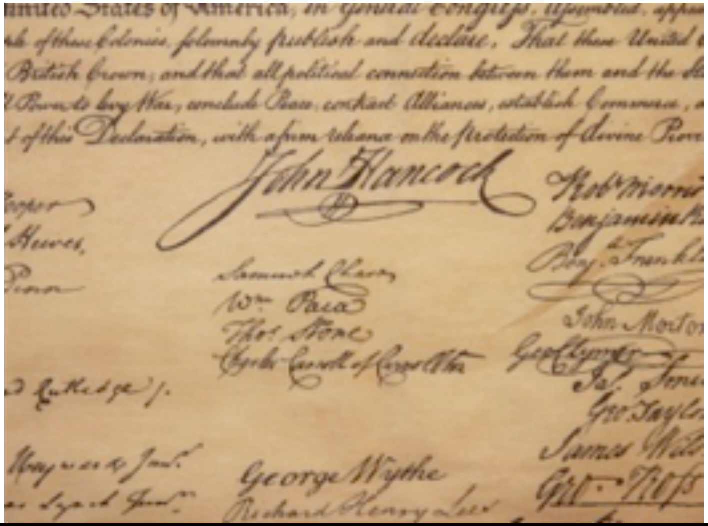


MACs in Practice

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit MAC in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
    - recent (2005) attacks on MD5
- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit MAC
    - Brute-force attacks on SHA now require 2^{63} operations to find a collision.
- General consensus that we should move to SHA2, SHA3

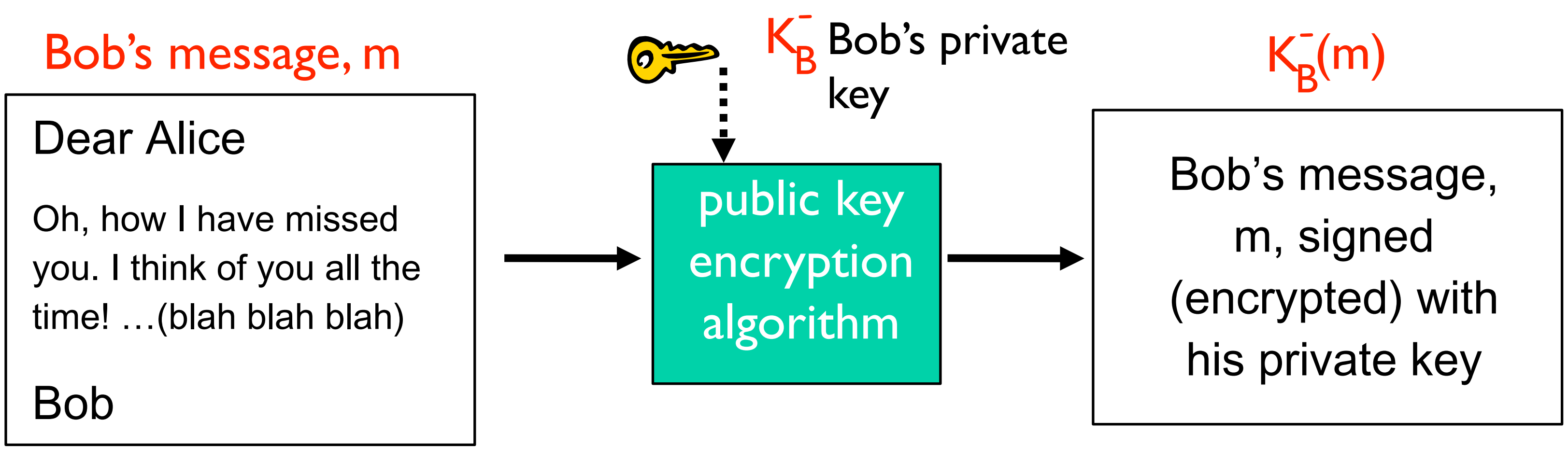
Digital Signatures

- Cryptographic technique analogous to hand-written signatures.
  - sender (Bob) digitally signs document, establishing he is document owner/creator.
  - verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document



Digital Signatures

- simple digital signature for message m:
  - Bob "signs" m by encrypting with his private key KB, creating "signed" message, KB(m)



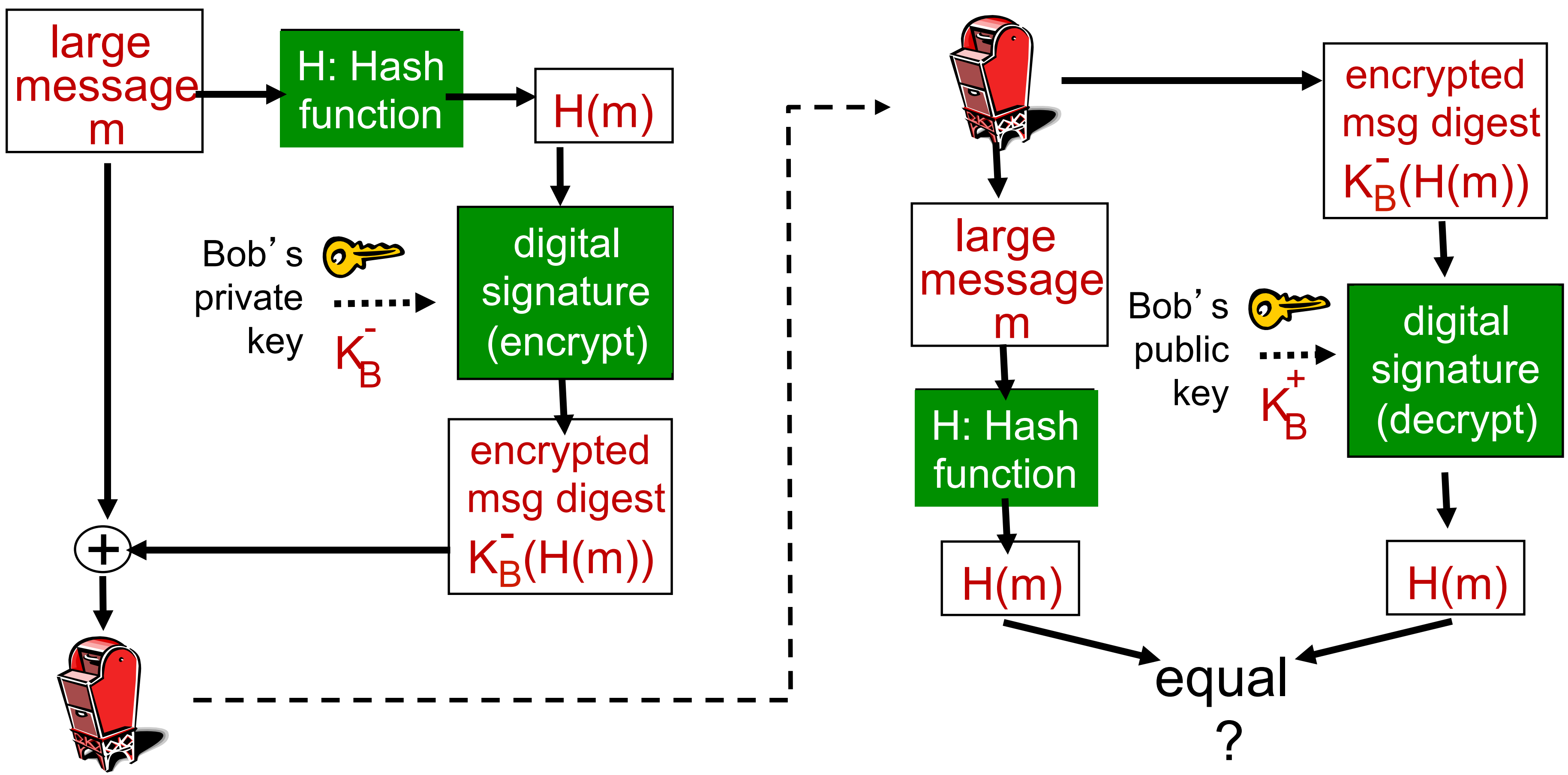
Digital Signatures (more)

- Suppose Alice receives msg m, digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.
- Alice thus verifies that:
  - Bob signed m.
  - No one else signed m.
  - Bob signed m and not m'.
- non-repudiation:
  - Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m.

Digital Signature = signed MAC

Bob sends digitally signed message:

Alice verifies signature, integrity of digitally signed message:

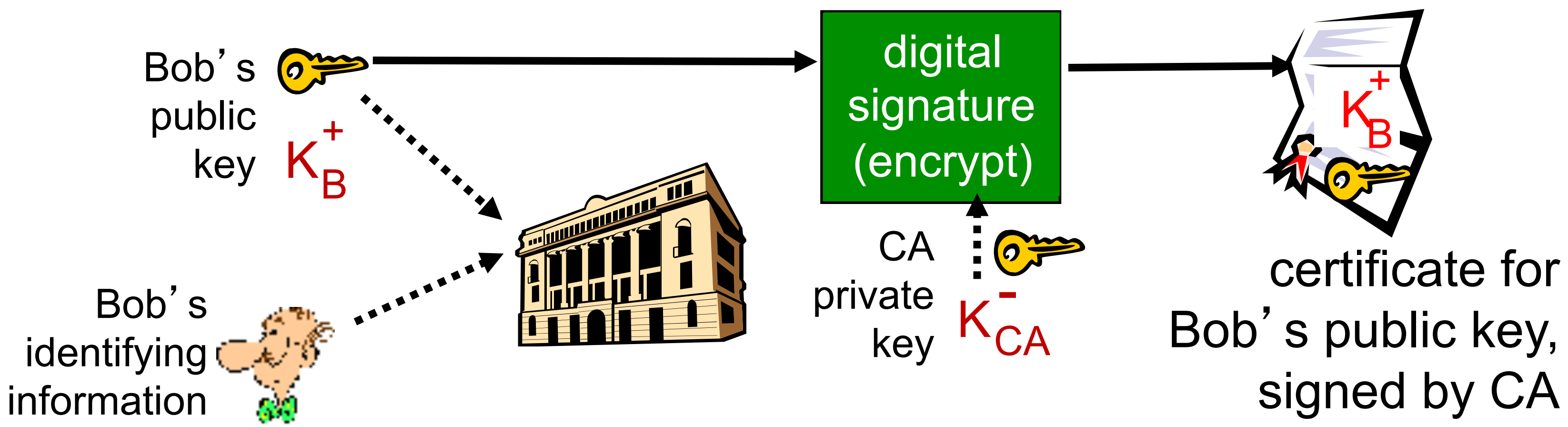


Public Key Certification

- Public Key Problem:
  - When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?
- Solution:
  - Trusted certification authority (CA)

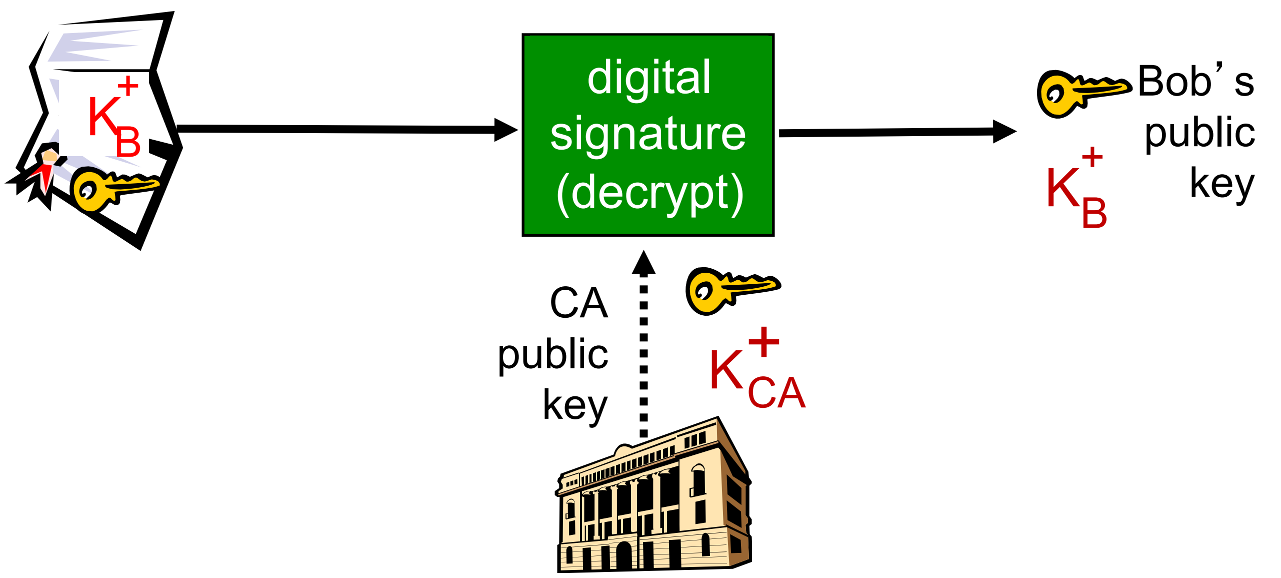
Certificate Authorities

- *Certificate Authority* (CA): binds public key to particular entity, E.
- E 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."



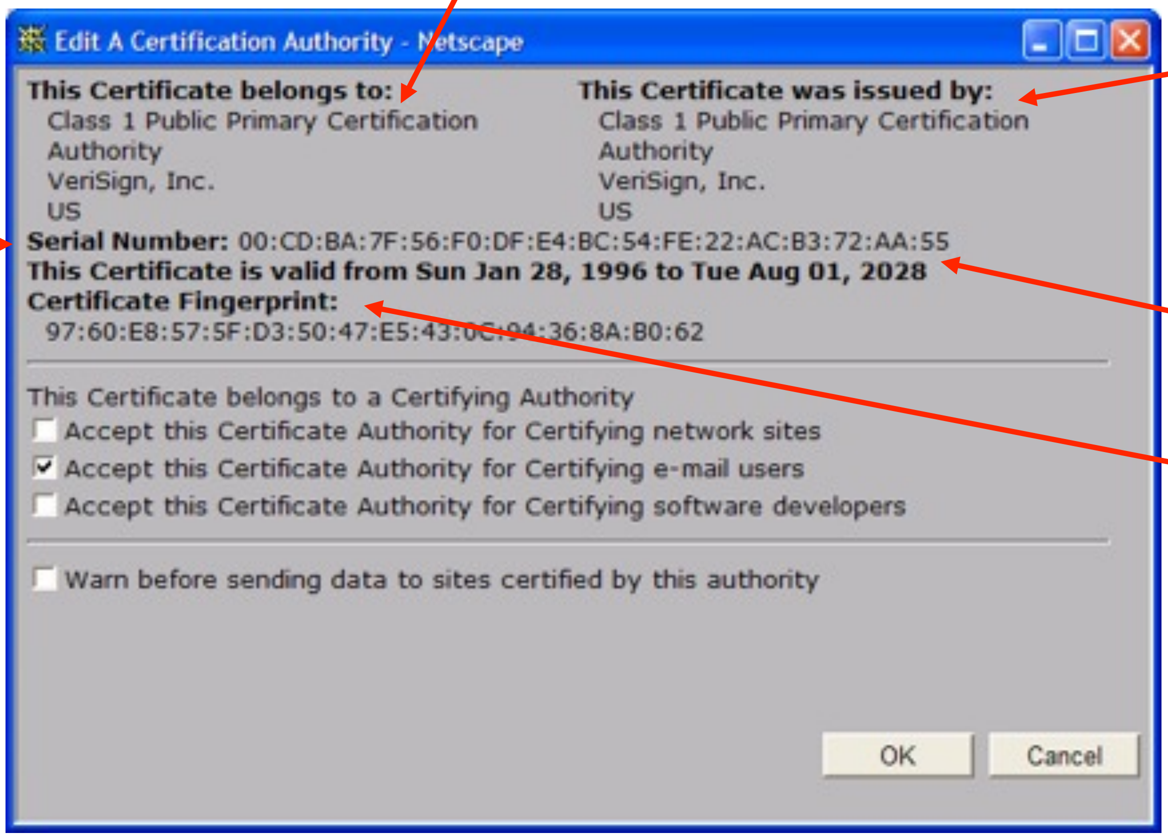
Certificate Authority

- 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



A Certificate Contains:

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- info about certificate issuer
- valid dates
- digital signature by issuer

Problems with PKI

- Why exactly do you trust a CA?
  - Anyone have any idea how many you actually trust?
- If two CAs present you with a certificate for Microsoft, which one is right?
- What prevents a CA from making up a key for you?
- What happens when keys are compromised?

Authentication

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

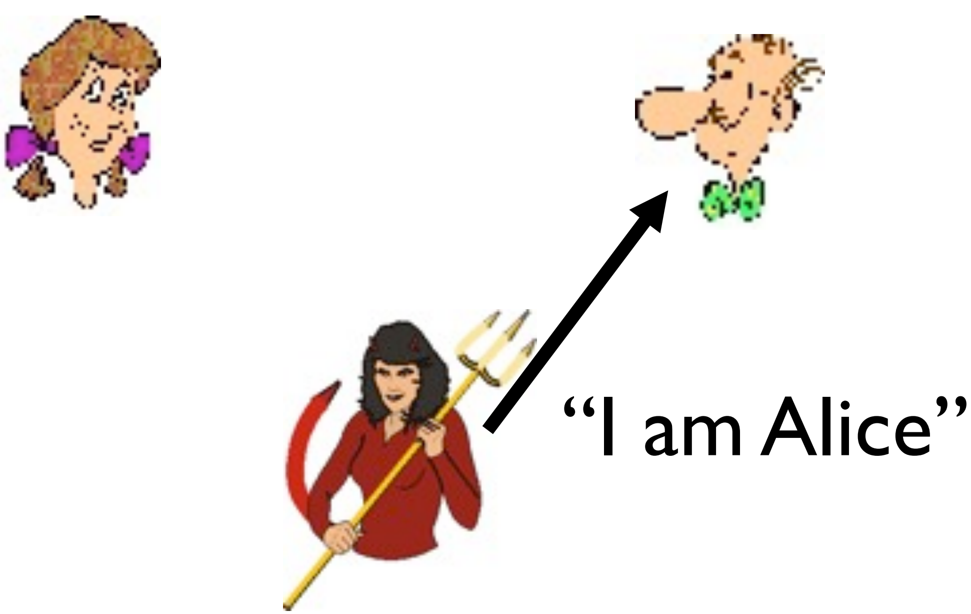
Protocol ap1.0: Alice says "I am Alice"

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 Trudy simply declares
herself to be Alice

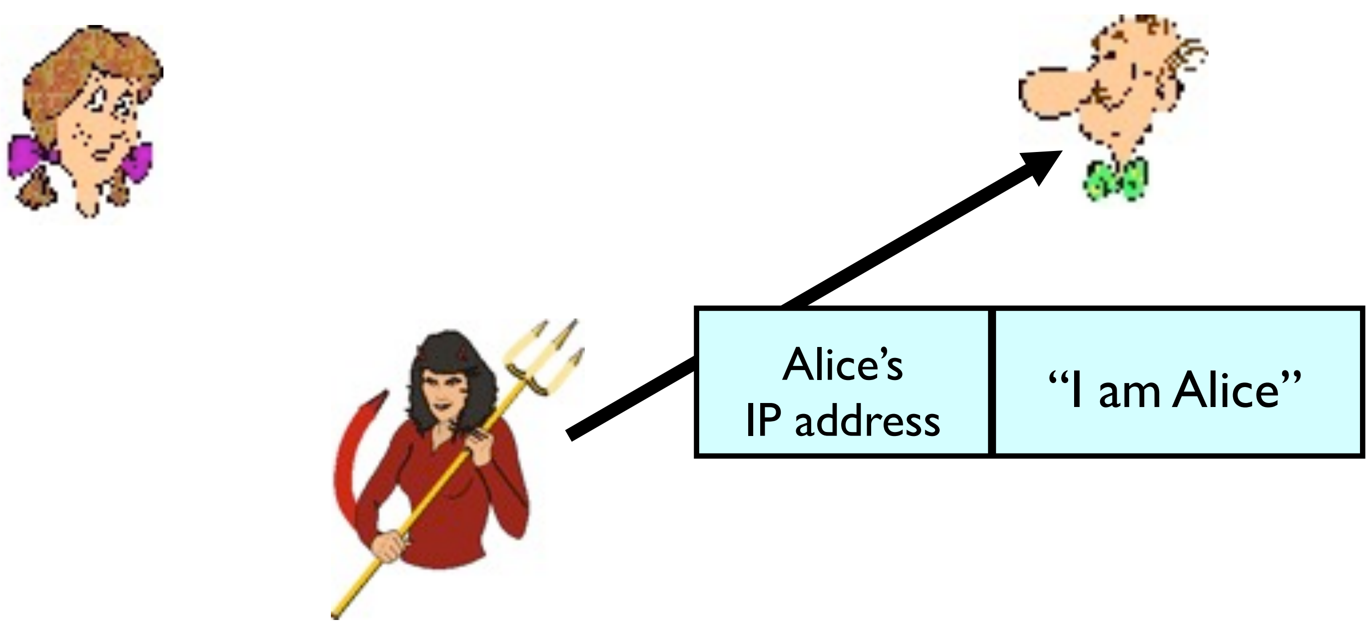
Authentication: another try

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

Failure scenario??

Authentication: another try

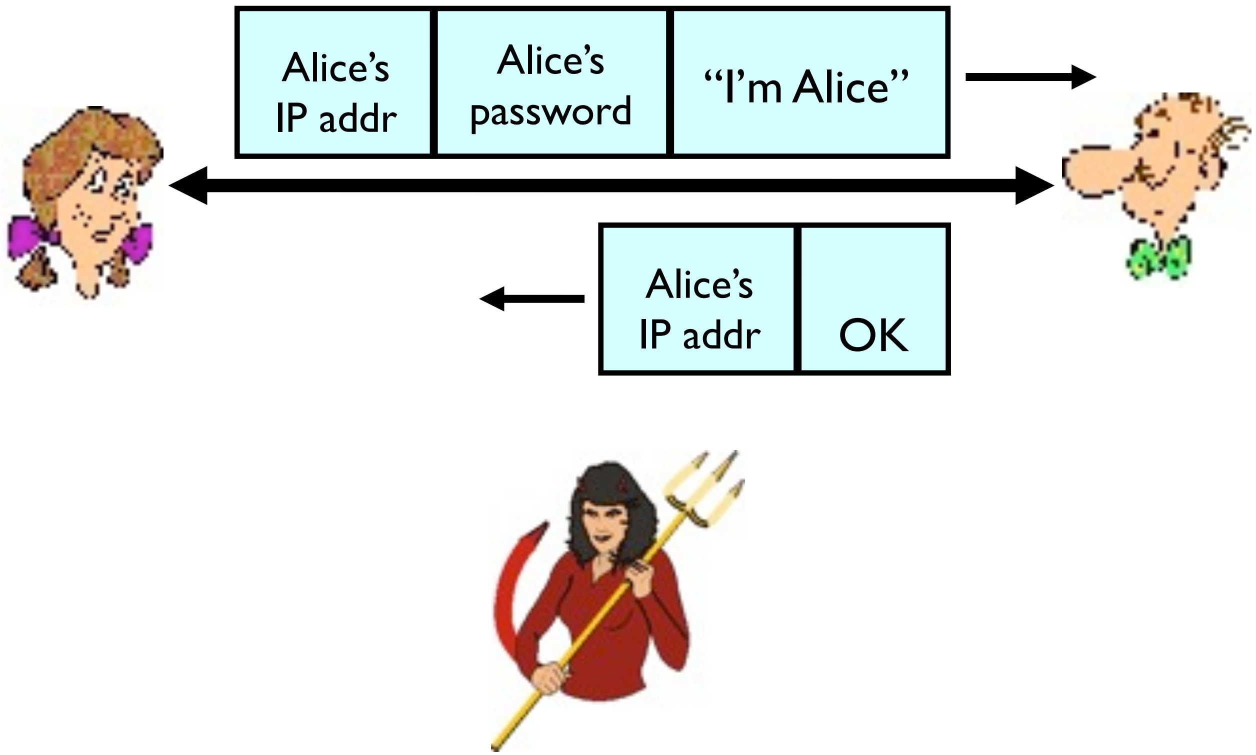
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

Authentication: another try

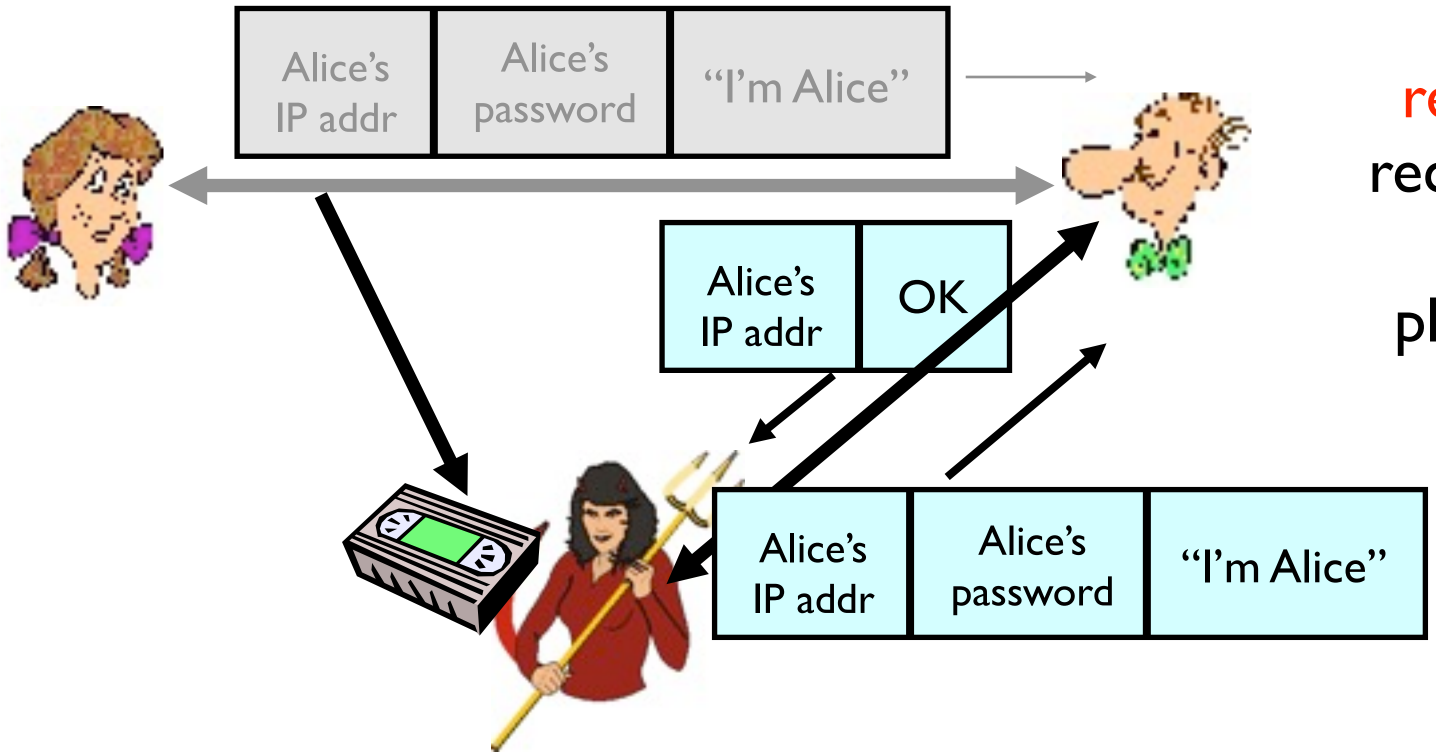
<u>Protocol ap3.0:</u> Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

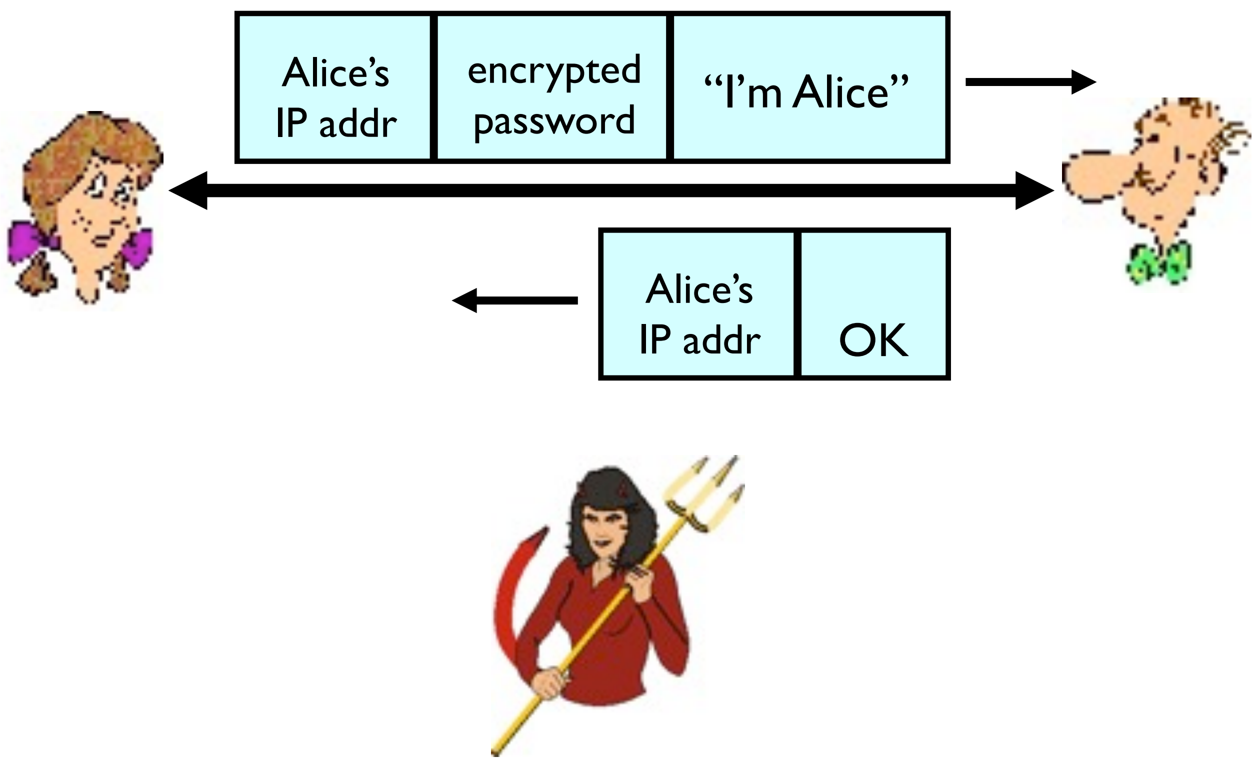
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



replay attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

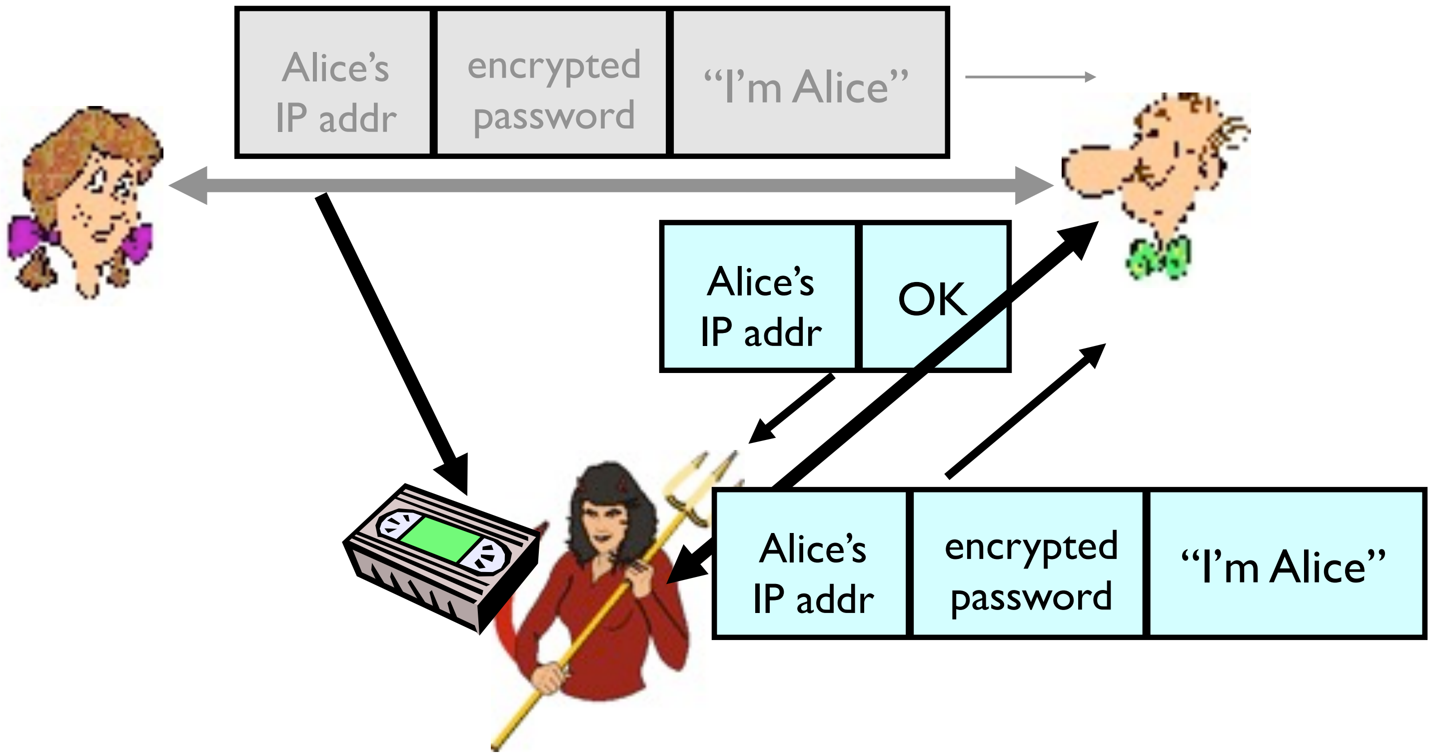
Protocol ap3.1: Alice says "I am Alice" and sends her encrypted secret password to "prove" it.



Failure scenario??

Authentication: another try

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



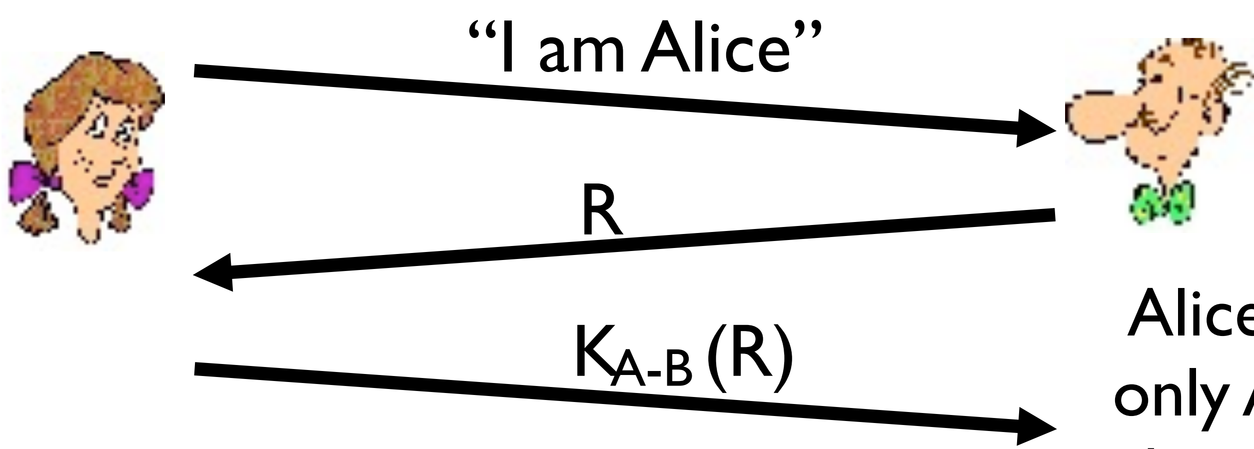
record and playback still works!

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



Failures, drawbacks?

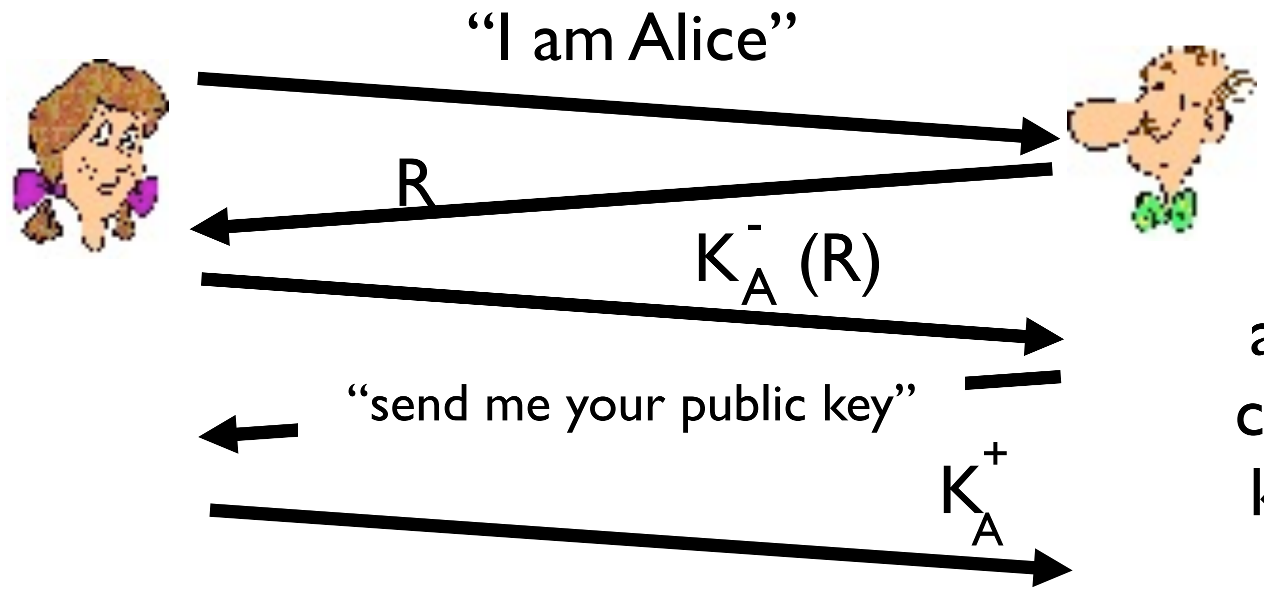
Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

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

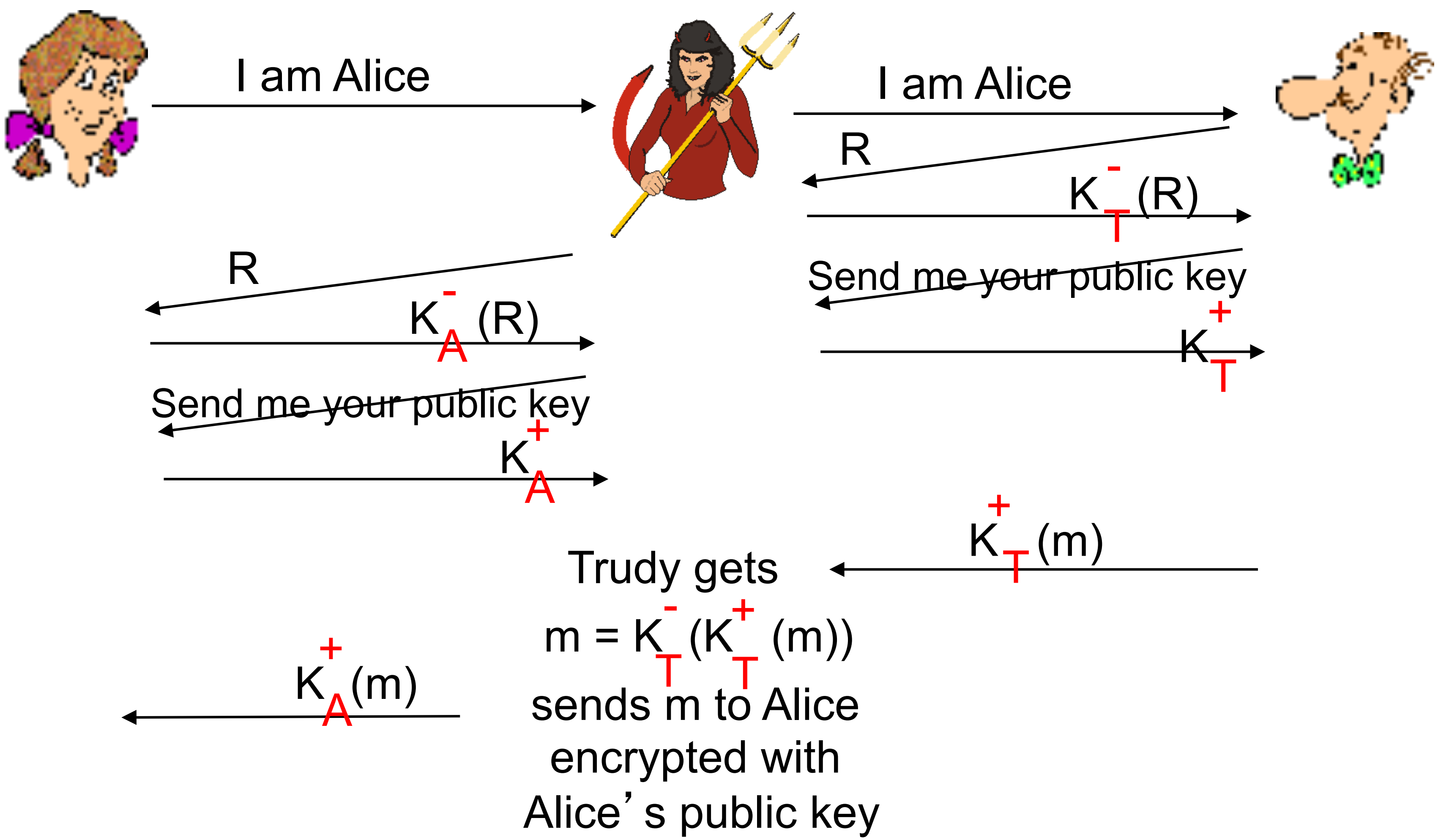
$$K_A^+(K_A^-(R)) = R$$

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 (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

Man (woman) in the middle attack: Trudy 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 Trudy receives all messages as well!

Remember Diffie-Hellman?

- How does Alice know Bob sent T_A?

Alice picks
her x: S_A

$T_A = g^{SA} \bmod p$

$T_B = g^{SB} \bmod p$

Bob picks
his x: S_B

Alice computes:
$K = T_B^{SA} \bmod p$

$T_B^{SA} = (g^{SB})^{SA} = g^{SB\ SA} = g^{SA\ SB} = (g^{SA})^{SB} = T_A^{SB} \bmod p$

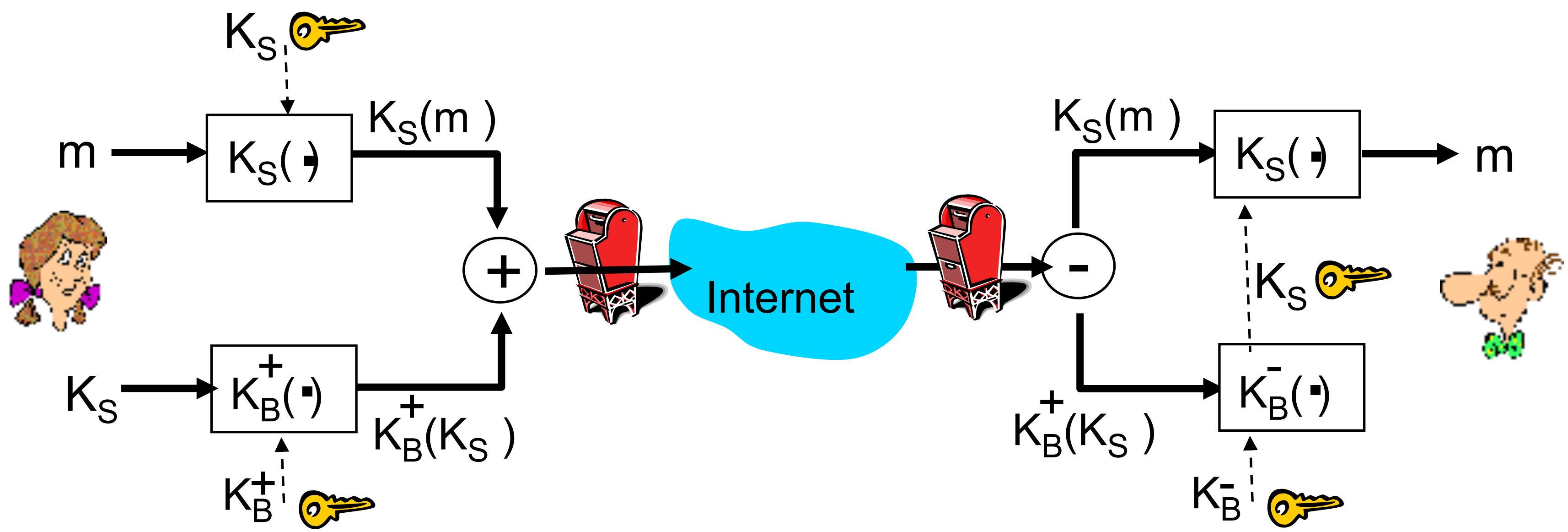
Alice computes:
$K = T_A^{SB} \bmod p$

Encrypted communication with K

- There is nothing to prevent a man-in-the-middle attack against this protocol.

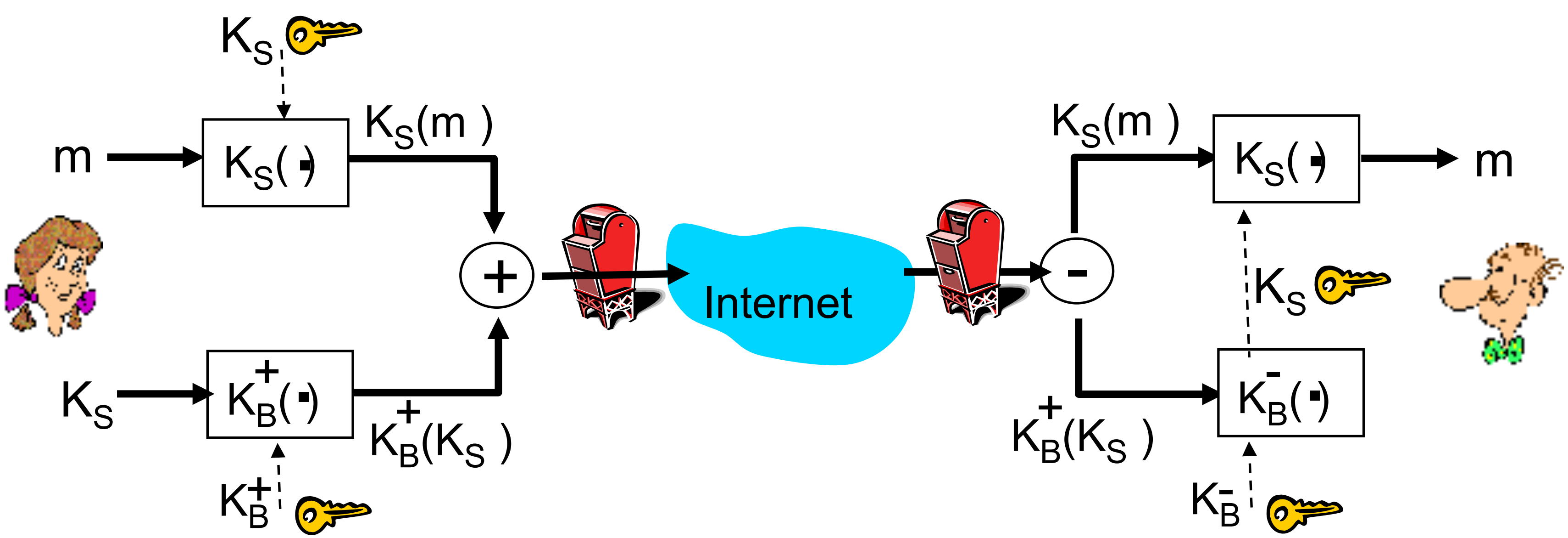
Secure Email

- 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 Email

- 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



Next Time

- Read Sections 8.5-8.6
- Read “Security Problems in the TCP/IP Protocol Suite” by Bellovin.
- Project 4 should be your focus right now
    - Last hurdle before the final - get it done!