南京大學
NANJING UNIVERSITY

Computer Networks

Wenzhong Li, Chen Tian

Nanjing University

Material with thanks to James F. Kurose, Mosharaf Chowdhury, and other colleagues.

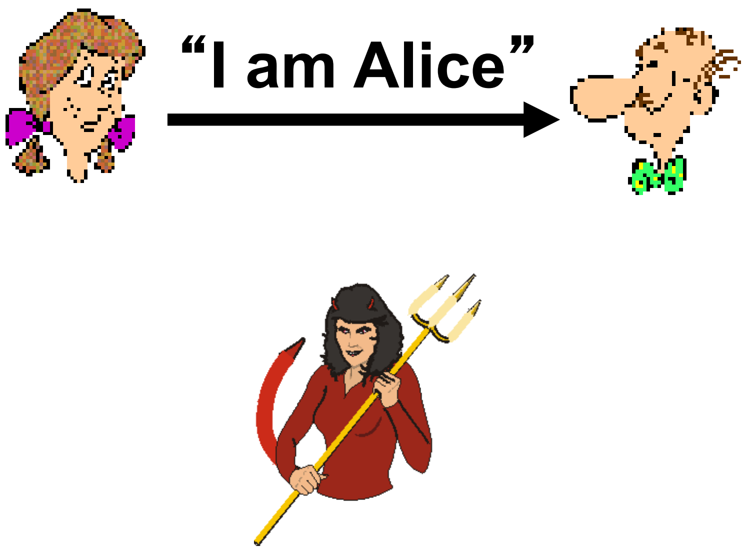
Chapter 5. Network Security

- Network Attacks
- Cryptographic Technologies
- Authentication
- Message Integrity
- Key Distribution
- Security in Different Network Layers
- Firewalls

Authentication

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

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

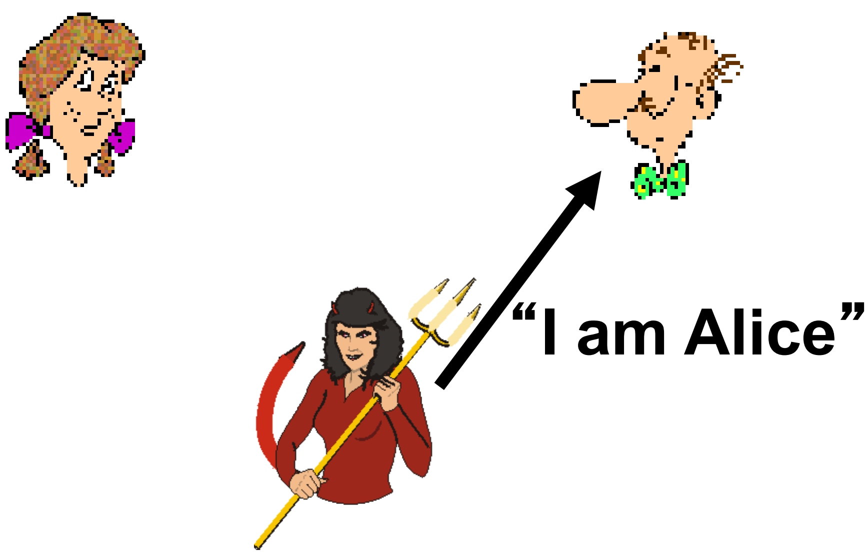


Failure scenario??

Authentication

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

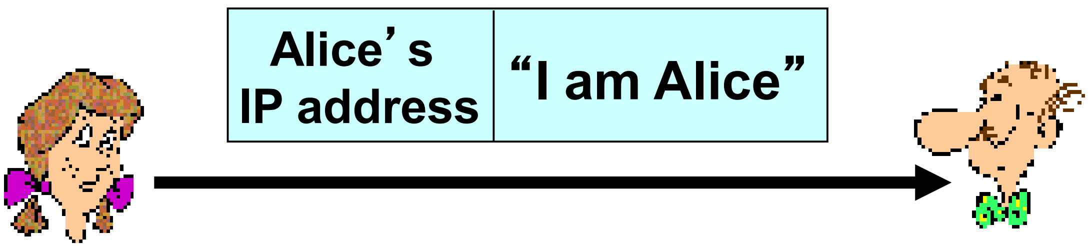
Protocol ap1.0: **Alice says "I am Alice"**



in a network,
Bob can not "see" Alice, so Trudy simply declares herself to be Alice

Authentication: another try

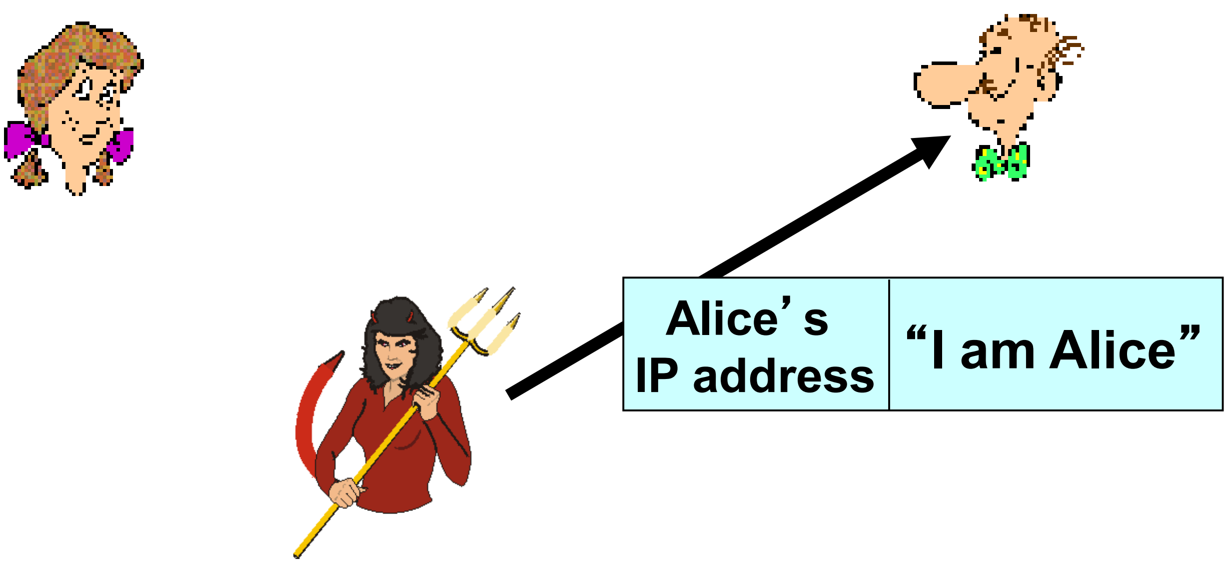
Protocol ap2.0: **Alice says "I am Alice" in an IP packet containing her source IP address**



Failure scenario??

Authentication: another try

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

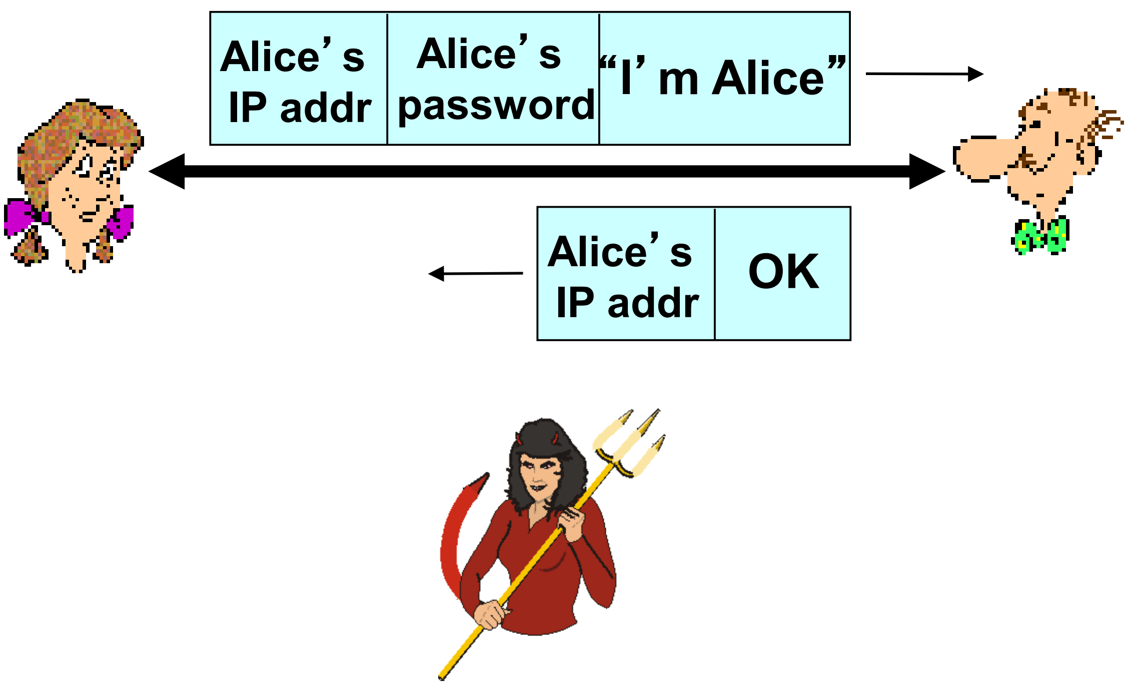


Trudy can create a packet "spoofing" Alice's address

Authentication: another try

Protocol ap3.0:

Alice says "I am Alice" and sends her secret password to "prove" it.

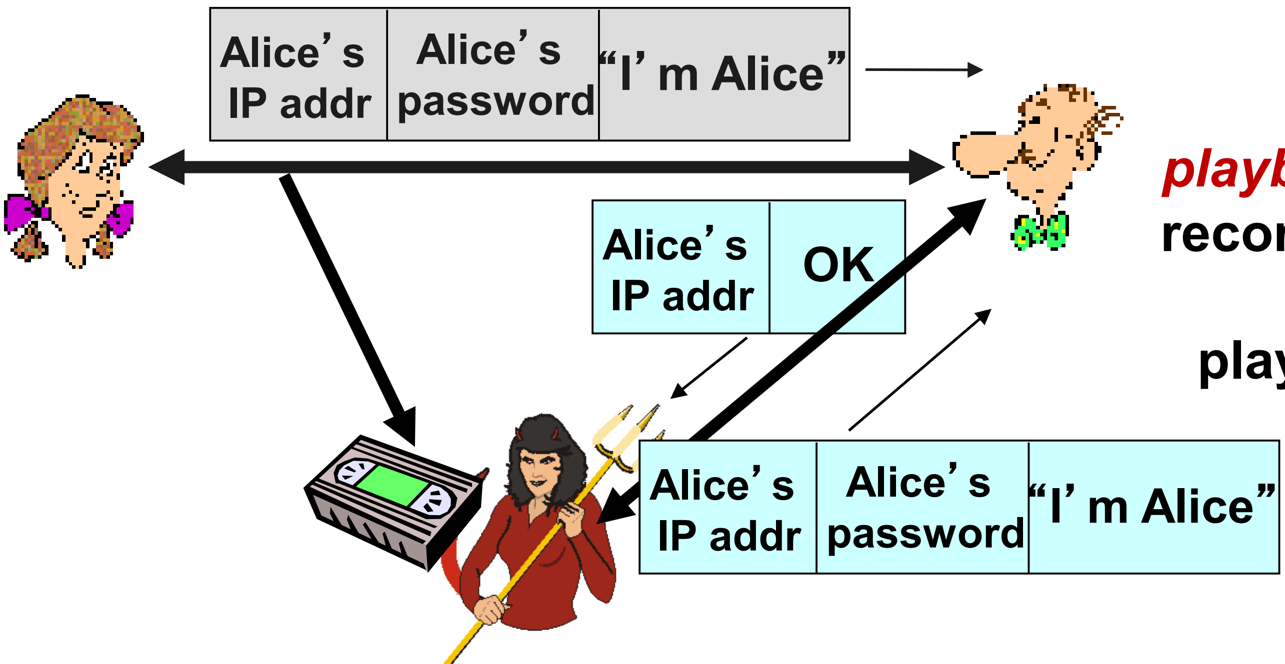


Failure scenario??

Authentication: another try

Protocol ap3.0:

Alice says "I am Alice" and sends her secret password to "prove" it.

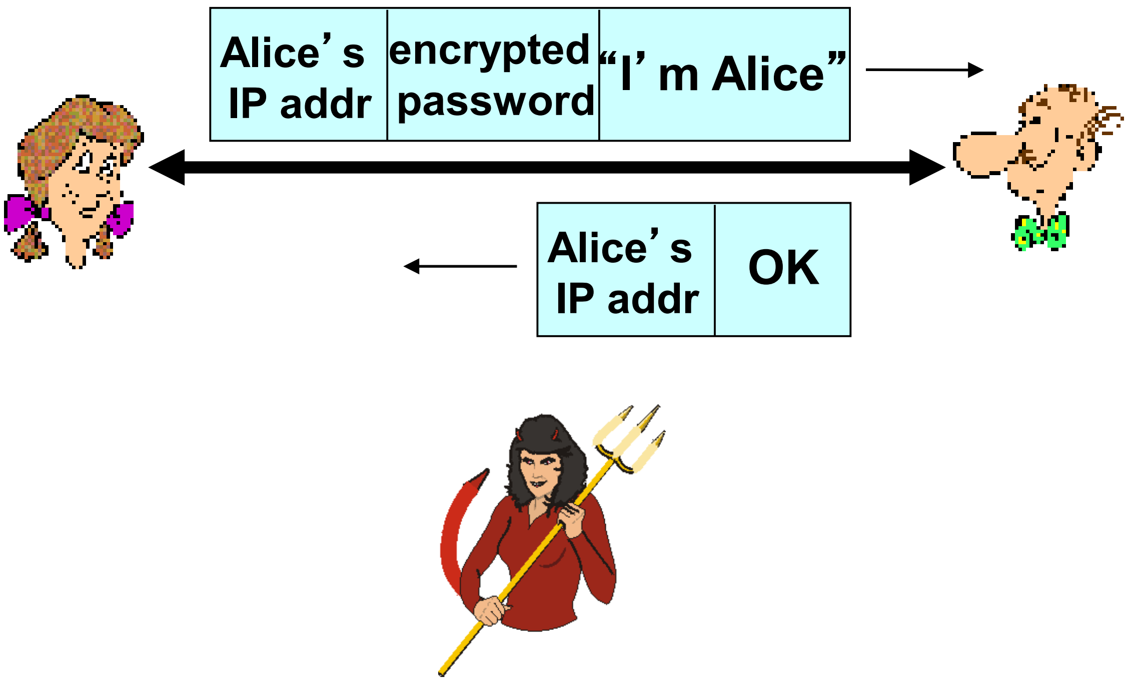


playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

Protocol ap3.1:

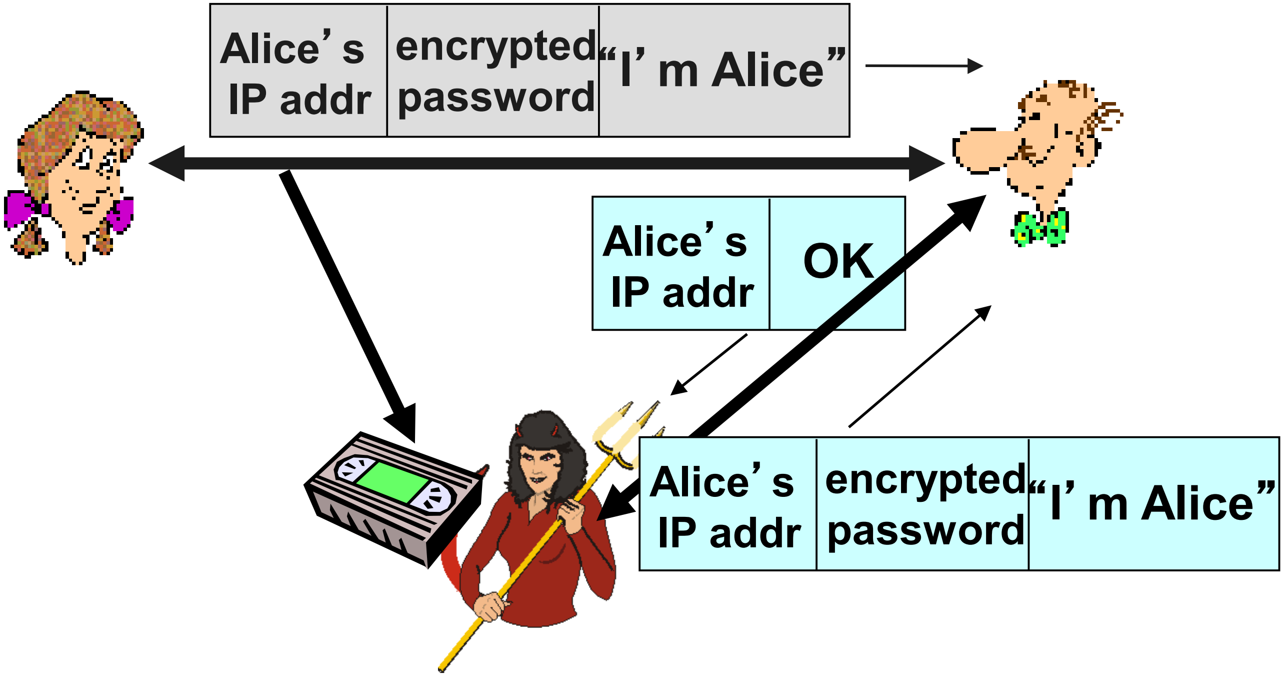
Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: yet another try

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



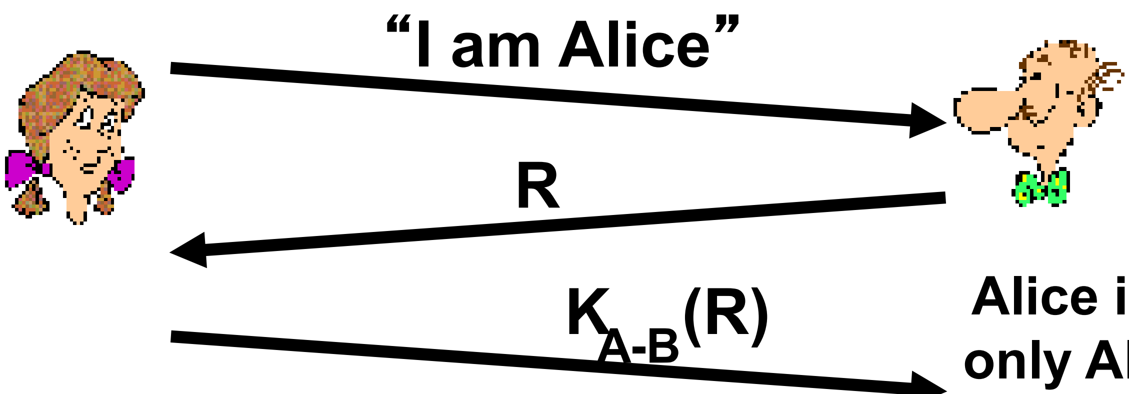
record and playback *still* works!

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!

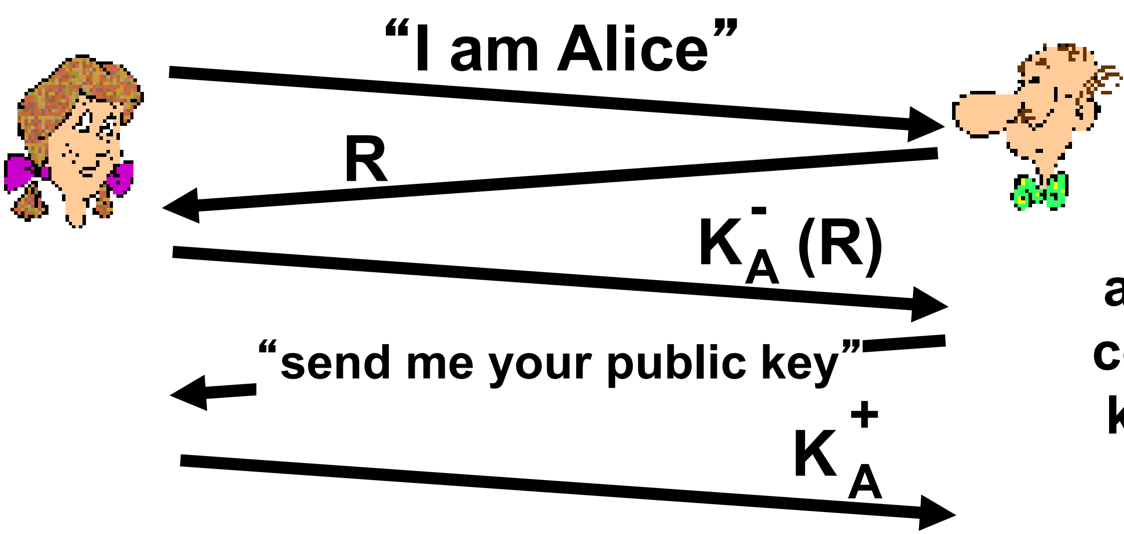
Failures, drawbacks?

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

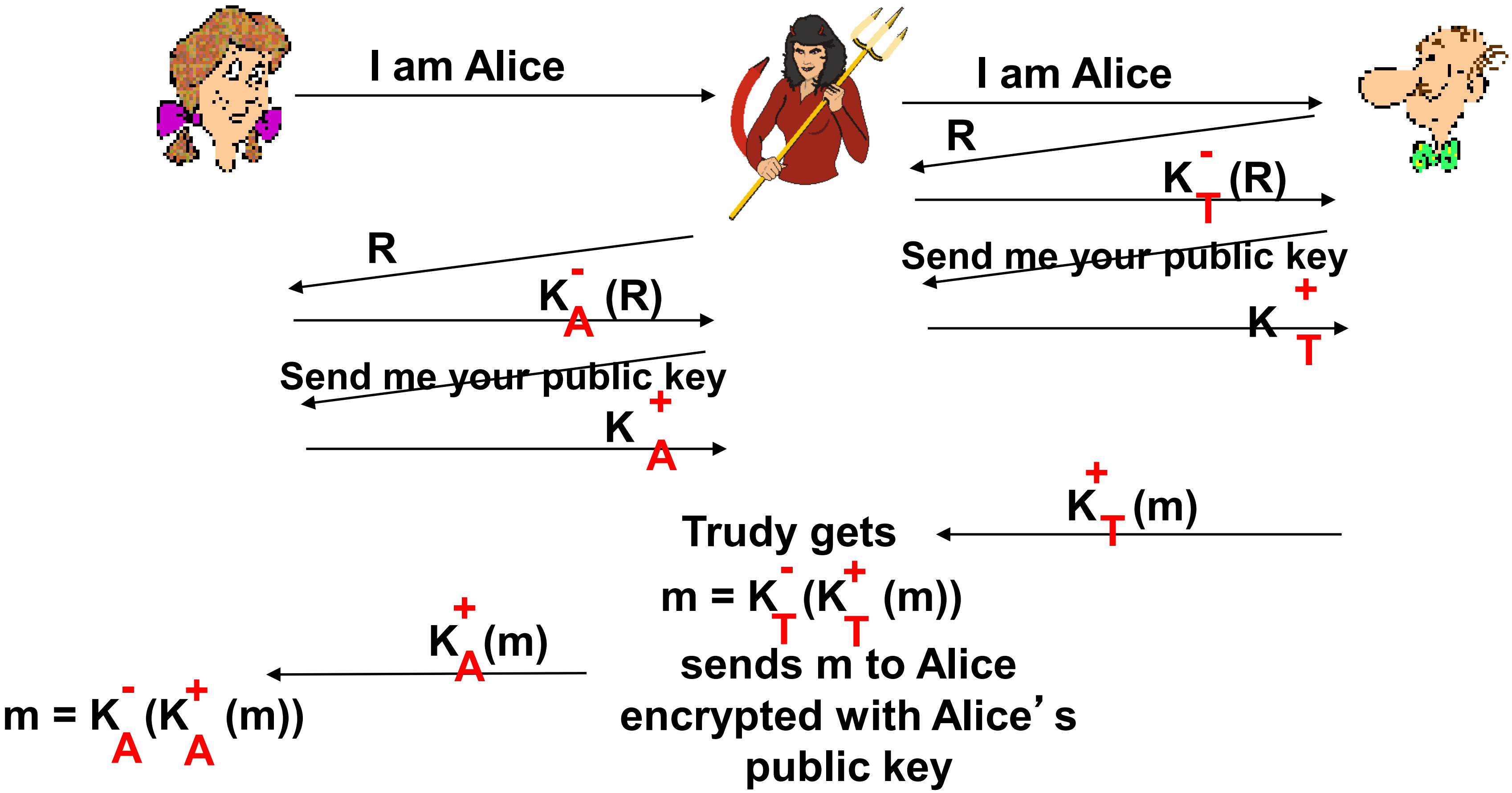
$$K_A^+(K_A^-(R)) = R$$

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

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

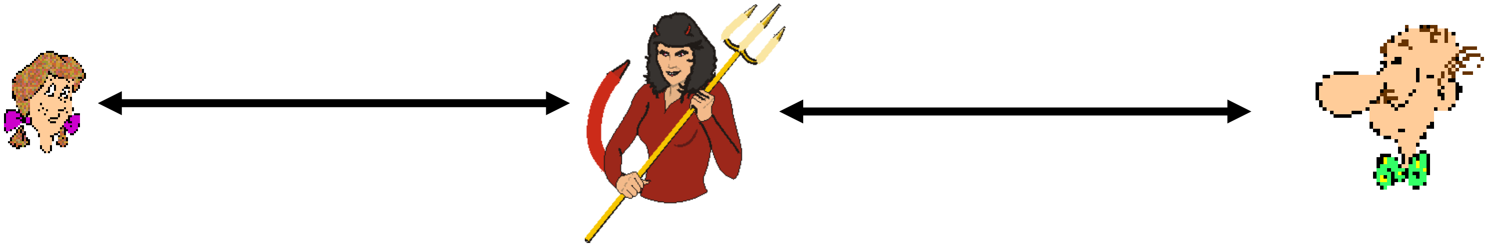
ap5.0: security hole

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



ap5.0: security hole

man (or 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!**

Message Integrity and Authentication

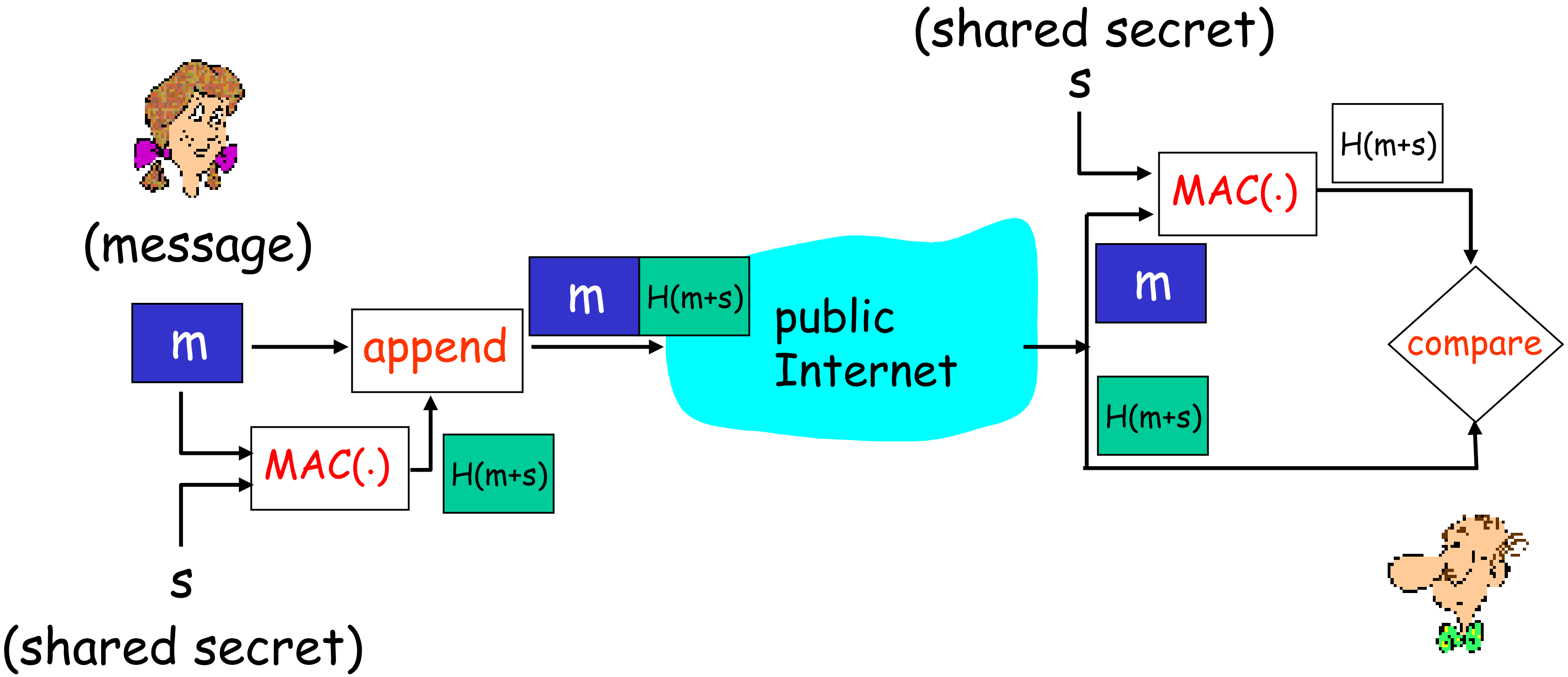
- Receiving msgs from Alice, Bob wants to ensure:
  - Message originally came from Alice
  - Message not changed since sent by Alice

- Security handling
  - Source impersonation / spoofing
  - Message injection / modification
  - Message re-sequencing / replaying

Authentication Functions

- Creating an authenticator which may involve functions of
  - Sender / Message Text
  - Time Stamp / Sequence Number / Random Value
  - Secret Keys
- The sender computes and sends the authenticator as part of the regular message
- The recipient compares the received authenticator with the expected authenticator

Message Authentication Code



Authentication by MAC

- *MAC* is a fixed-size code that is appended to the message
  - Typical sizes of *MAC* range from 64 to 256 bits
- Message can be sent in the clear without encryption
- *MAC* is a function of the message and a secret key
  - Can assure msg not altered, and from alleged sender
- *MAC* should not be reversible, decryption is not needed
- The strength of the *MAC* depends on the function and on the secrecy of the key

Authentication Methods

- Authentication by Crypto
  - Using crypto functions of the text and secret keys
  - CBC-MAC

- Authentication by Hash
  - Using hash functions and involving secret keys in the computations
  - MD5, 128 bit MAC, (RFC 1321)
  - SHA-1, 160 bit MAC, (NIST, FIPS PUB 180-1)

Requirements of MAC Functions

- Operability
  - Work on any input length
  - Produce output of fixed size
  - Should be easy to compute

- Security
  - One-way – given value Y, it is hard to find content X such that $Y = MAC(X)$
  - Weak Collision Resistance – given content X_1 it is hard to find another content X_2 such that $MAC(X_1) = MAC(X_2)$
  - Strong Collision Resistance – it is hard to find any two different contents X_1 and X_2 such that $MAC(X_1) = MAC(X_2)$

CBC-MAC Authentication

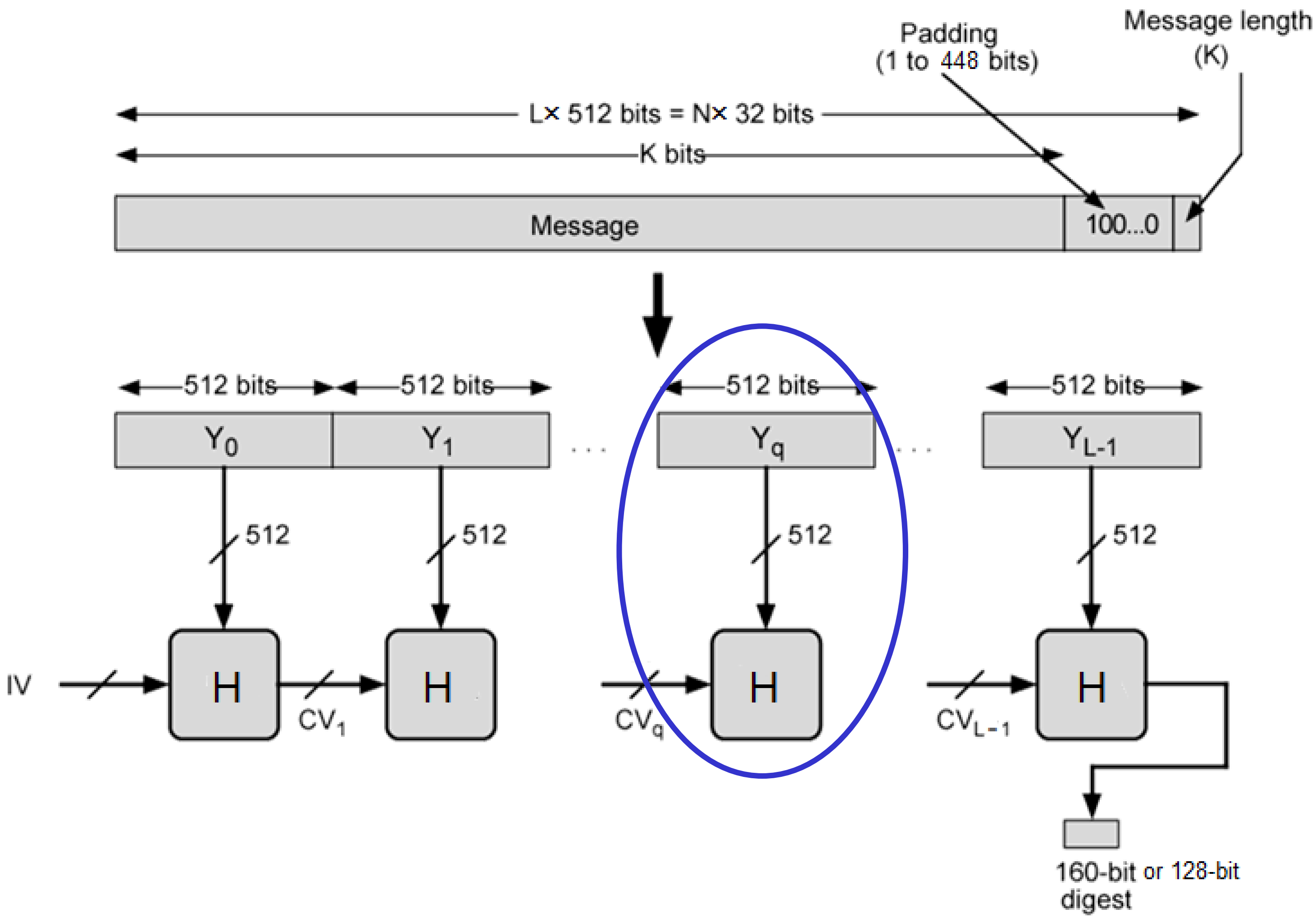
- Cipher block chaining message authentication code
- Divide message M into L blocks of size n bits each

  $M = M_1, M_2, \ldots, M_L$
- Let K be a secret key of the encryption algorithm E

- Let $C_0 = IV$ be a random block of n' bits
- Compute $C_i = E_K(M_i + C_{i-1})$ for $i = 1, 2, \ldots, L$
  - $CBC\text{-}MAC_K(M) = C_L$

- Let $MAC_K(M) = (C_0, C_L) = (IV, CBC\text{-}MAC_K(M))$
  - i.e. the first and last blocks of *CBC* encryption

Common Structure for MD5 and SHA-1



Common Steps

- Input message less than 2^{64} bits
  - Processed in 512 bit blocks
- Appends padding bits
  - Message Length congruent to 448(mod 512)
- Adds length field
  - Original message length is written in last 64 bits

MD5 Processing

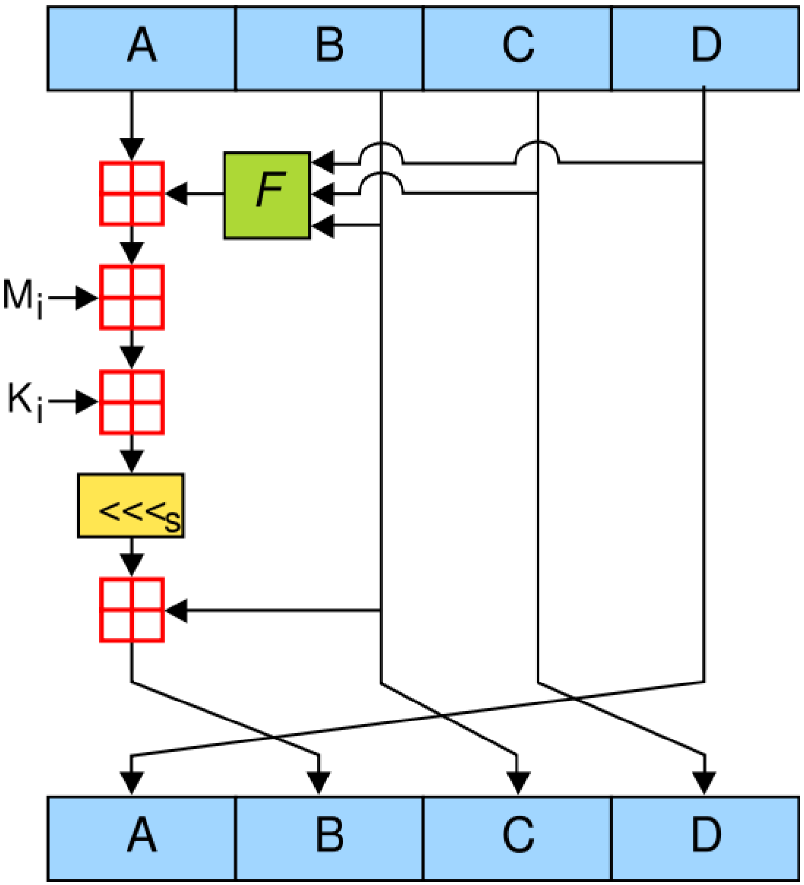
- Uses 4-word state buffer A, B, C, D to compute the message digest
  - Initial value: 01234567, 89abcdef, fedcba98, 76543210
  - Total 128 bits
- Process message in 16-word blocks
  - M_0, M_1, … M_{15}
- Processing of a msg block consists of 4 similar stages
  - Each with a different function F
- Each stage is composed of 16 similar operations
  - Using F, modular +, and left rotation

One MD5 Operation

- A different F is used for each stage

$$F_1(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$F_2(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$F_3(X,Y,Z) = X \oplus Y \oplus Z$$
$$F_4(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

- M_i is a 32-bit word of msg
- K_i is a 32-bit generated constant

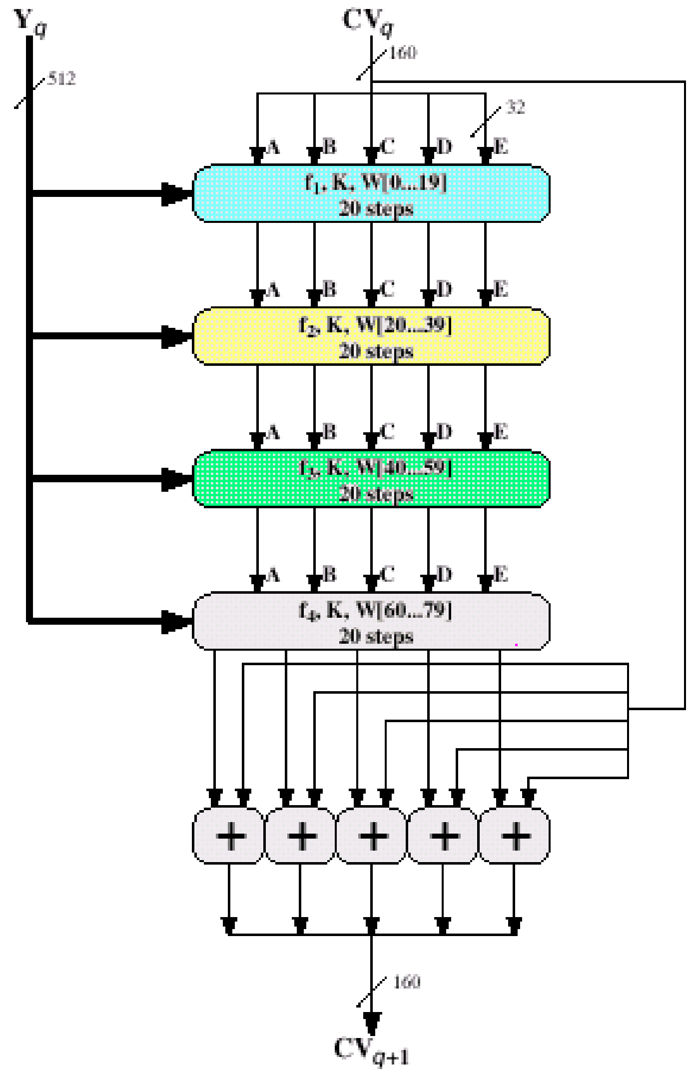


SHA-1 Processing

- Uses 5 word state buffer A, B, C, D, E to compute the message digest
  - Value 67452301, efcdab89, 98badcfe, 10325476, c3d2e1f0
  - Total 160 bits
- Process message in 16-word chunks
  - M_0, M_1, … M_{15}
- Processing of a msg block consists of 4 similar stages
  - Each with a different function F
- Each stage is composed of 20 similar operations

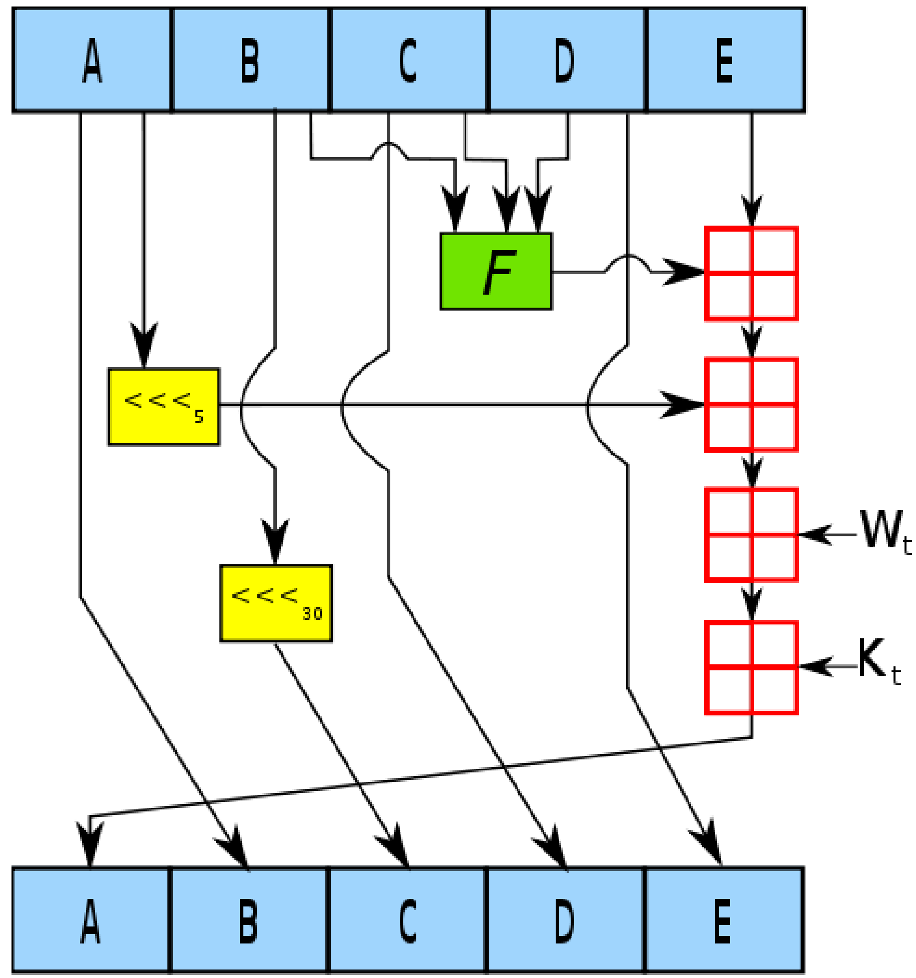
SHA for a Single Chunk

- $M_0, M_1, \ldots, M_{15}$: 16 words of input chunk
- For t = 0 to 15, $W_t = M_t$
- For t = 16 to 79, $W_t = S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$
- F_1, F_2, F_3, F_4 : 4 different elementary functions
- K : distinct set of constants for each F_i



One SHA Operation

- F is a nonlinear function that varies
- W_t is the expanded message word of step t
- K_t is the constant of step t

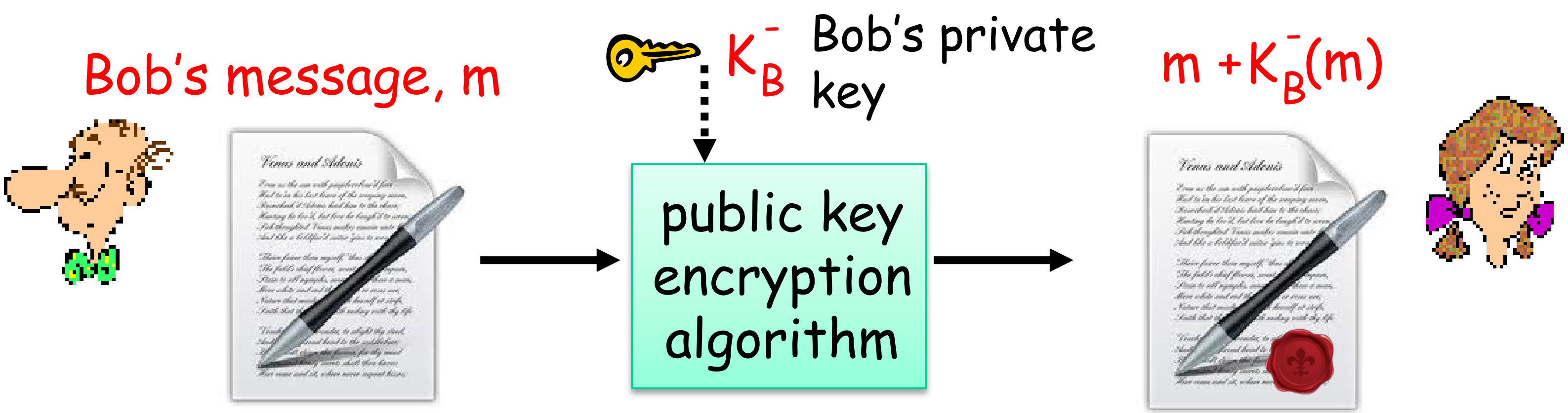


Breaking MD5 & SHA-1

- 2004年，山东大学数学系王小云首次展示MD5产生碰撞的高效算法。
- 2005年2月，王小云提出SHA-1产生碰撞的算法，其复杂度从$O(2^{80})$降为$O(2^{69})$，同年8月，该复杂度进一步降为$O(2^{63})$。
- 但是，产生碰撞并不等于可以随意产生所需要的内容，更不能随意篡改内容并通过哈希校验，所以MD5和SHA-1至今仍被广泛使用。

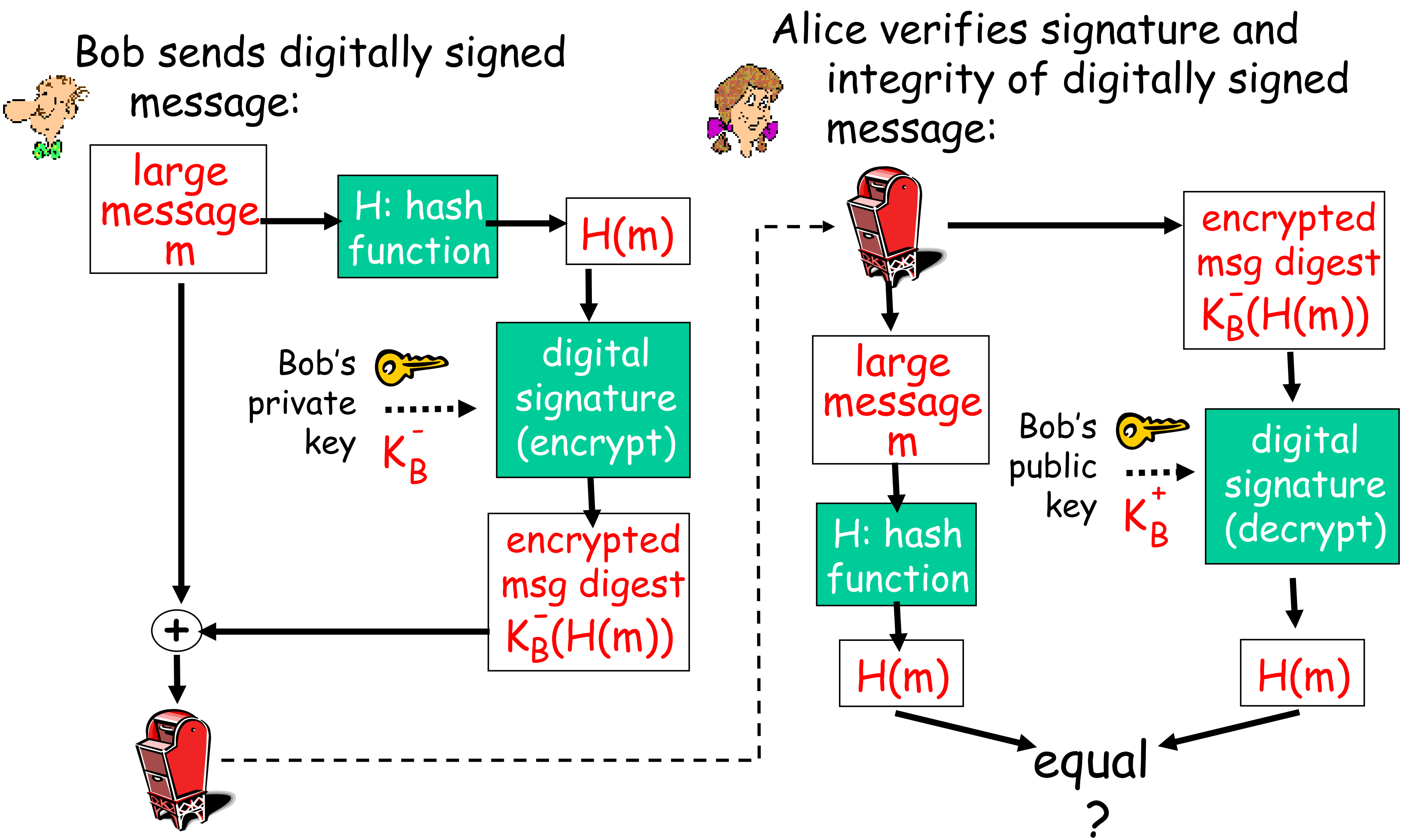
Digital Signature

- Sender (Bob) digitally signs document, making him document owner/creator
- Recipient (Alice) can prove to someone that Bob, and no one else, must have made the document



- Bob's private key is essential

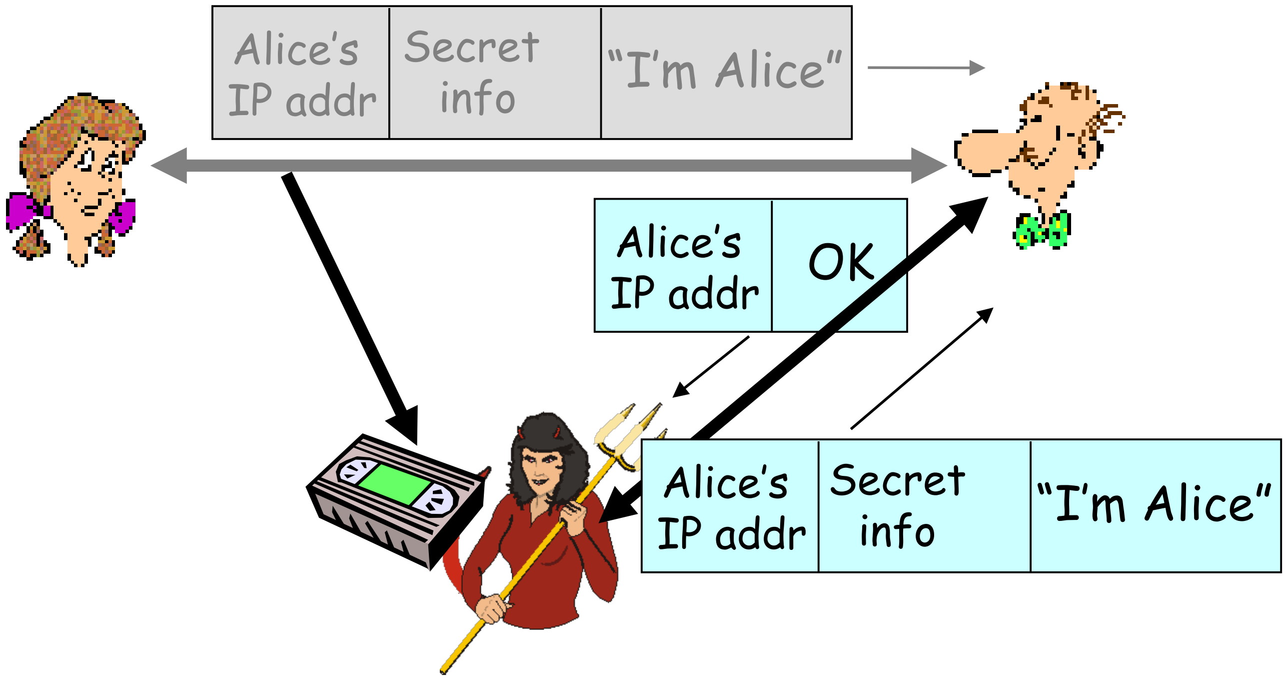
Digital Signature is Signed MAC



Key Distribution

- Problem
  - How can Alice and Bob share the common secret key
  - How does Alice know Bob’s public key does be Bob’s public key
- Solution
  - Diffie-Hellman Key Exchange
  - Trusted certification authority (CA)
  - Certificate for public key

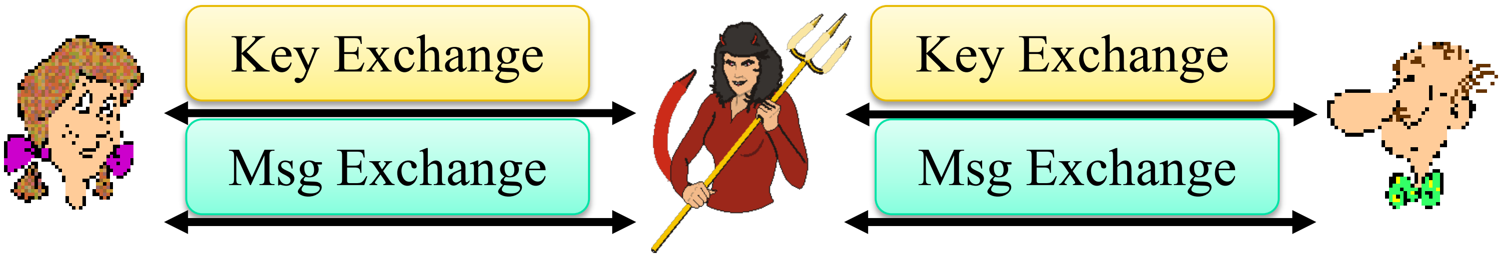
Attack Key Distribution



- Record and playback
  - Still account for large part of secret holes
  - Needs proper use of timestamp and nonce

Nonce：不重数

Attack Key Distribution



- Middle attack (Man-in-the-middle attack)
  - Trudy poses as Alice (to Bob) and as Bob (to Alice)
- Hard to detect
  - Bob receives everything that Alice sends, and vice versa
  - But Trudy receives all messages as well!

- 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

Diffie-Hellman Key Exchange

- Diffie-Hellman Key Exchange, 1976
- Diffie, Hellman (Turing Award 2015)

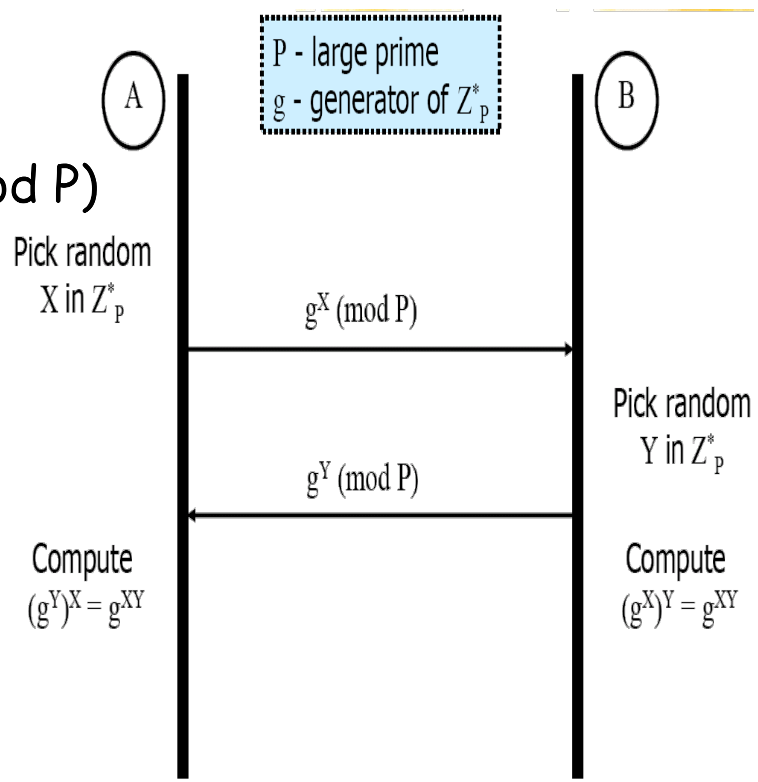
Diffie-Hellman Key Exchange (1)

- Preliminary
  - Large prime P known to the world
  - Generator g of Z_p^* known to the world
  - A and B do not share any secret value

$Z_p^* = \{0, 1, 2, \ldots, p\text{-}1\}(\text{mod } p)$

Diffie-Hellman Key Exchange (2)

- The D-H Protocol
  - A picks at random a number $X \in \{1, 2, \ldots, P-1\}$ and sends to B the value $g^X (\text{mod } P)$
  - B picks at random a number $Y \in \{1, 2, \ldots, P-1\}$ and sends to A the value $g^Y (\text{mod } P)$
  - A computes $(g^Y)^X\ (\text{mod } P) = g^{XY} (\text{mod } P)$
  - B computes $(g^X)^Y\ (\text{mod } P) = g^{XY} (\text{mod } P)$
  - A and B now share the secret value $g^{XY} (\text{mod } P)$
- Note:
  - $Z_P^* = \{1 \le a \le P-1 : gcd(a,P)=1\}$
    - Each $[a]$ denote a set $[a] = \{a + k \times P : k \in Z\}$
    - For a prime P, $Z_P^* = \{1, 2, \ldots, P-1\}$
  - Generator g of Z_P^*: $g \in Z_P^*$
    - $\forall a \in Z_P^*, \exists k \in Z, a = g^k (\text{mod } P)$

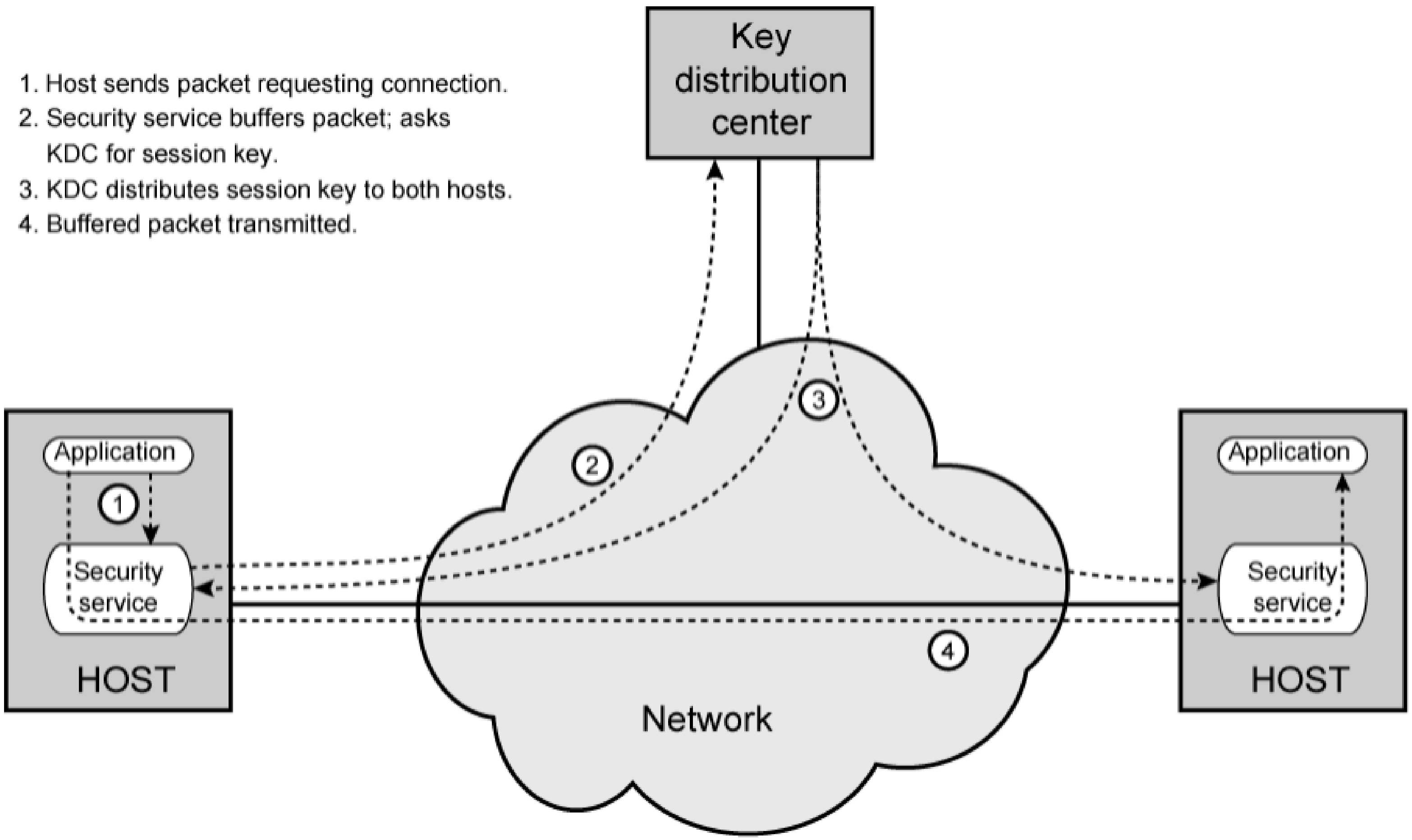


Alice	
knows	doesn't know
$p = 23$	$b = ?$
base $g = 5$	
$a = 6$	
$A = 5^a \bmod 23$	
$A = 5^6 \bmod 23 = 8$	
$B = 19$	
$s = B^a \bmod 23$	
$s = 19^6 \bmod 23 = 2$	
$s = 2$	

Bob	
knows	doesn't know
$p = 23$	$a = ?$
base $g = 5$	
$b = 15$	
$B = 5^b \bmod 23$	
$B = 5^{15} \bmod 23 = 19$	
$A = 8$	
$s = A^b \bmod 23$	
$s = 8^{15} \bmod 23 = 2$	
$s = 2$	

Trudy	
knows	doesn't know
$p = 23$	$a = ?$
base $g = 5$	$b = ?$
	$s = ?$
$A = 8$	
$B = 19$	
$s = 19^a \bmod 23 = 8^b \bmod 23$	

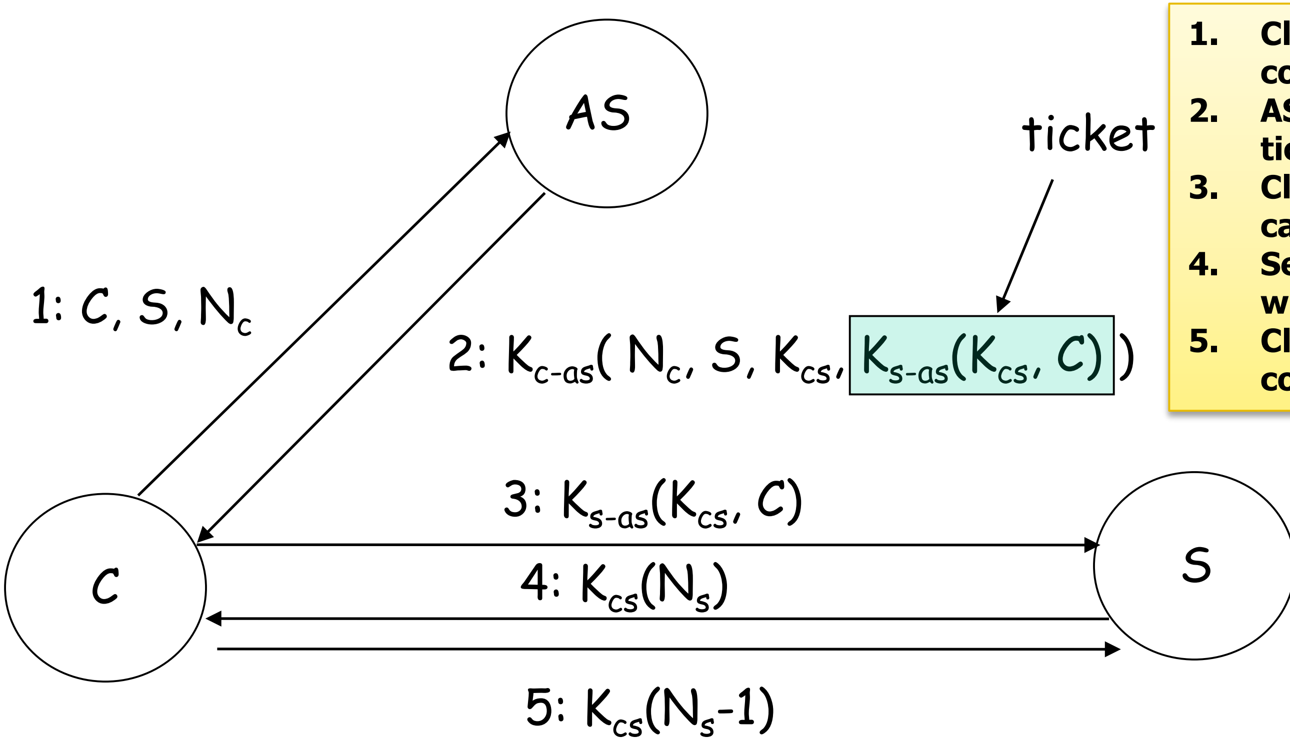
Trusted Certification Authority



Key Distribution via CA

- Session Key
  - Used for duration of one logical connection
  - Destroyed at end of session
- Permanent key
  - Used for distribution of keys
- Key distribution center (CA)
  - Determines validity of sender and receiver
  - Provides one session key for that connection
- Security service module (SSM)
  - Performs end to end encryption
  - Obtains keys for host

The Needham-Schroder Protocol



1. **Client tells AS that it want to communicate with Server.**
2. **AS generate session key K_{cs} and ticket, then sends back to client**
3. **Client sends ticket to server, server can decrypt using $K_{s\text{-}as}$**
4. **Server send a nonce encrypted with the obtained session key**
5. **Client sends back nonce-1 to show communication OK**

AS: Authentication server (KDC)
C: client
S: server

$K_{x\text{-}as}$: key shared between X and AS, where X is C, or S
K_{cs}: session key between client C and server S
N_x: Nonce generated by X

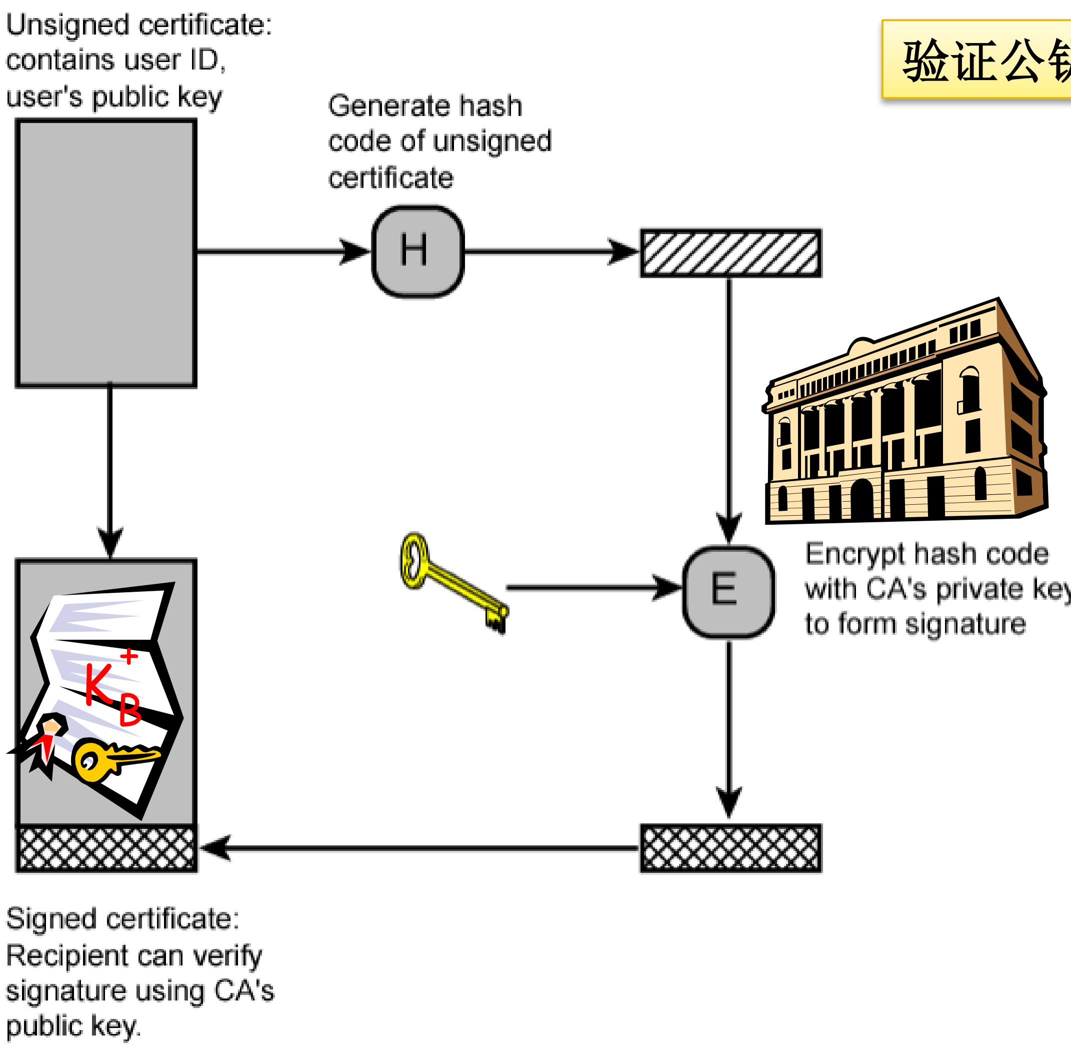
One-Time Session Key

- Public key not suitable for large blocks of message
- Bob communications with Alice by following steps
  - Prepares a message
  - Encrypts the message using symmetric crypto with a one-time session key
  - Encrypts the session key using Alice's public key
  - Attaches the encrypted session key to the message and sends it to Alice
  - Alice gets the session key using her private key, and decrypts the message

Public Key Certificate

- Question
  - How to ensure the published public key does be Alice's public key, not from someone else
- Solution: Public key certificate
  - A public key plus User ID of the key owner
  - Above block signed by a trusted CA with a timestamp
- Others cannot substitute Alice's public key with his own
  - Cannot forge the signature of the trusted CA

Public Key Certificate



验证公钥正确性的过程

Public Key Certificate

- Serial number (unique to this certificate)
- Info about certificate owner, including algorithms and key value

X509 Public Key Certificate

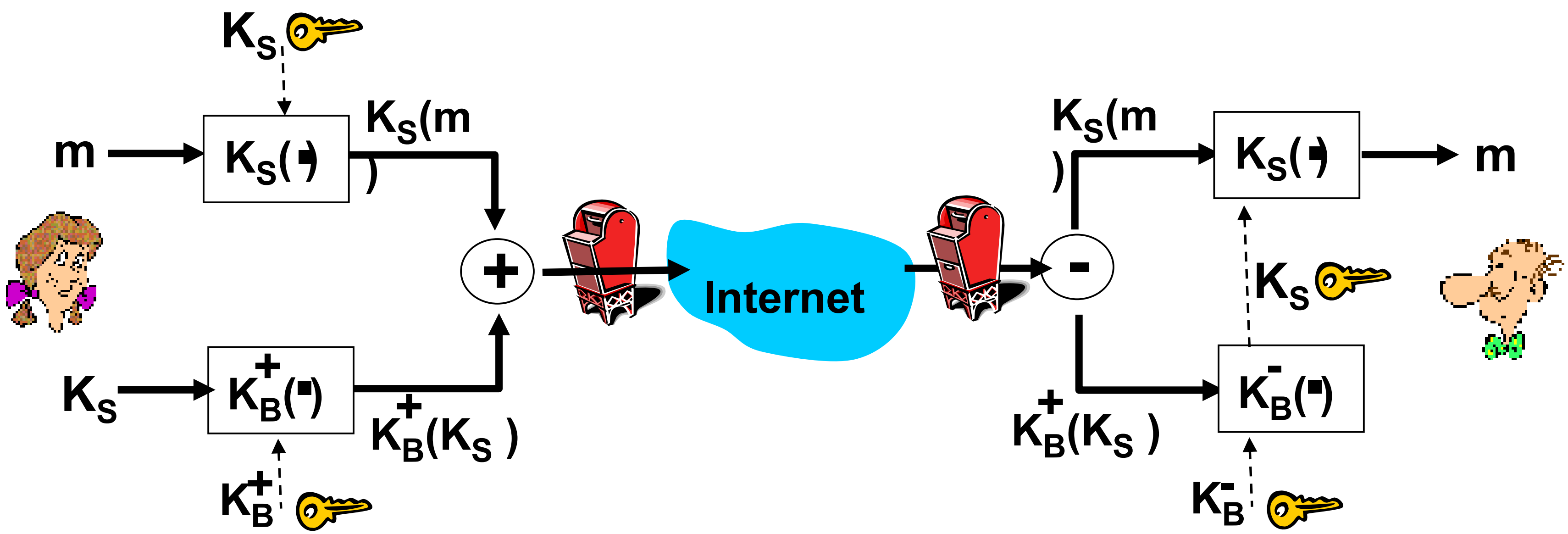
Field	Value
Version	V3
Serial number	3c 8d 3a 64 ee 18 dd 1b 73 0b...
Signature algorithm	sha1RSA
Issuer	Thawte SGC CA, Thawte Cons...
Valid from	2. května 2008 18:02:55
Valid to	2. května 2009 18:02:55
Subject	www.google.com, Google Inc,...
Public key	RSA (1024 Bits)
Enhanced Key Usage	Server Authentication (1.3.6....
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Authority Information Access	[1]Authority Info Access: Acc...
Basic Constraints	Subject Type=End Entity, Pat...
Thumbprint algorithm	sha1
Thumbprint	8a aa 9a 71 f0 5c e7 25 8a 35 ...

Signature	31 0a 6c a2 9e e9 54 ...

- Info about certificate issuer
- Including valid dates, digital signature by issuer (thumbprint / fingerprint)

Secure e-mail

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

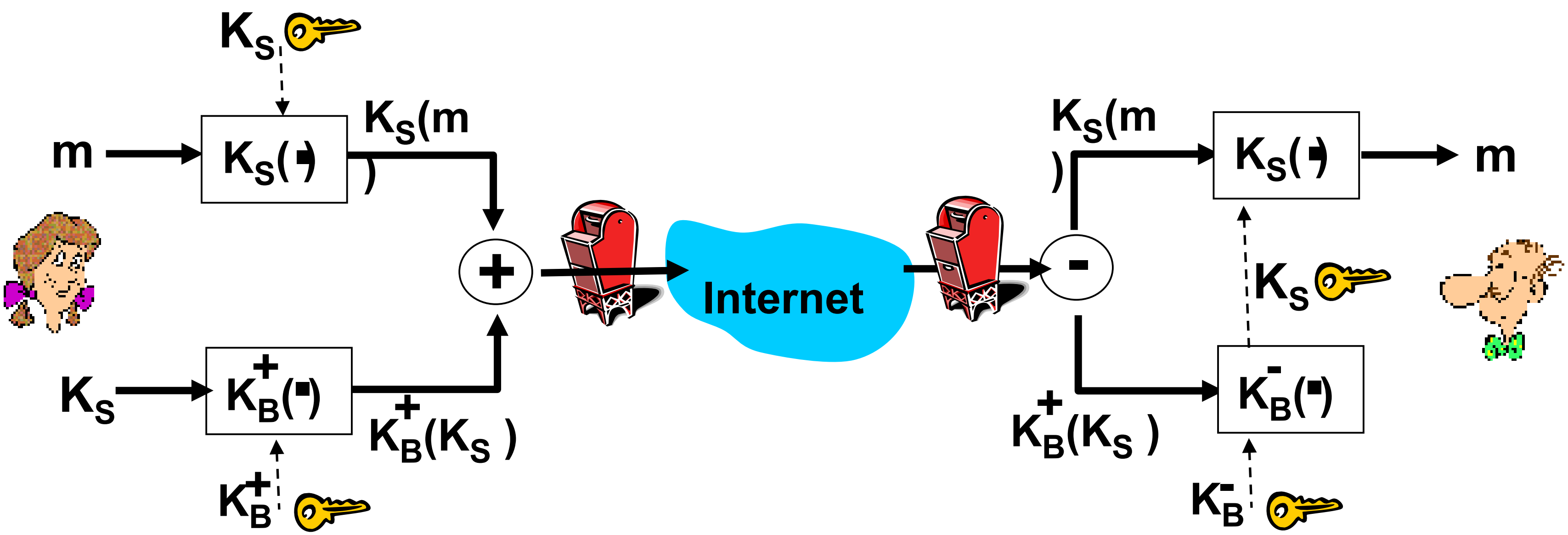


Alice:

- 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

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

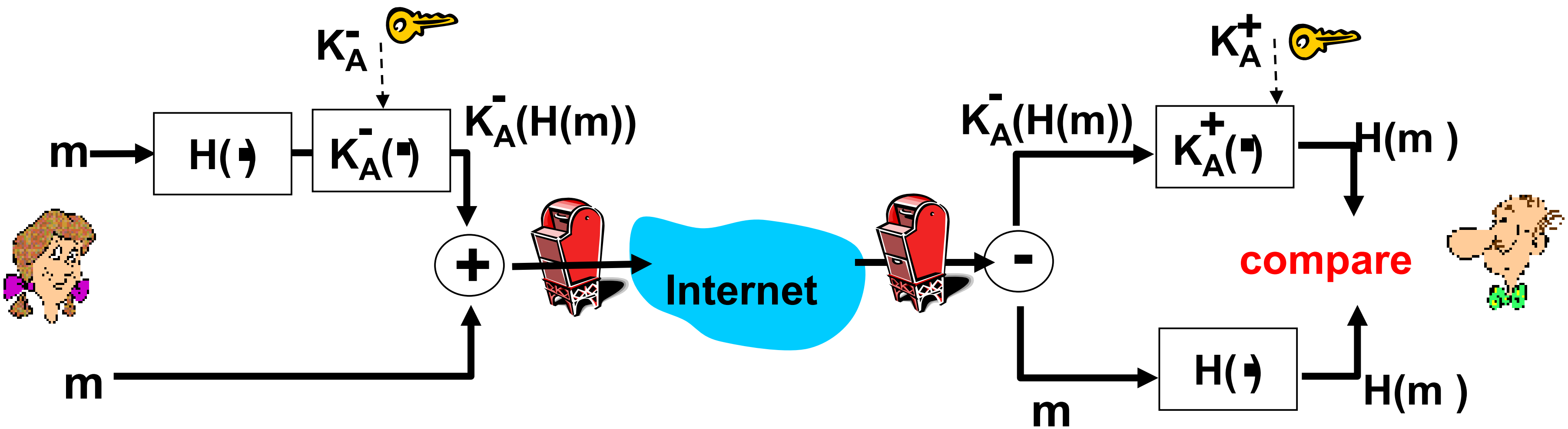


Bob:

- 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)

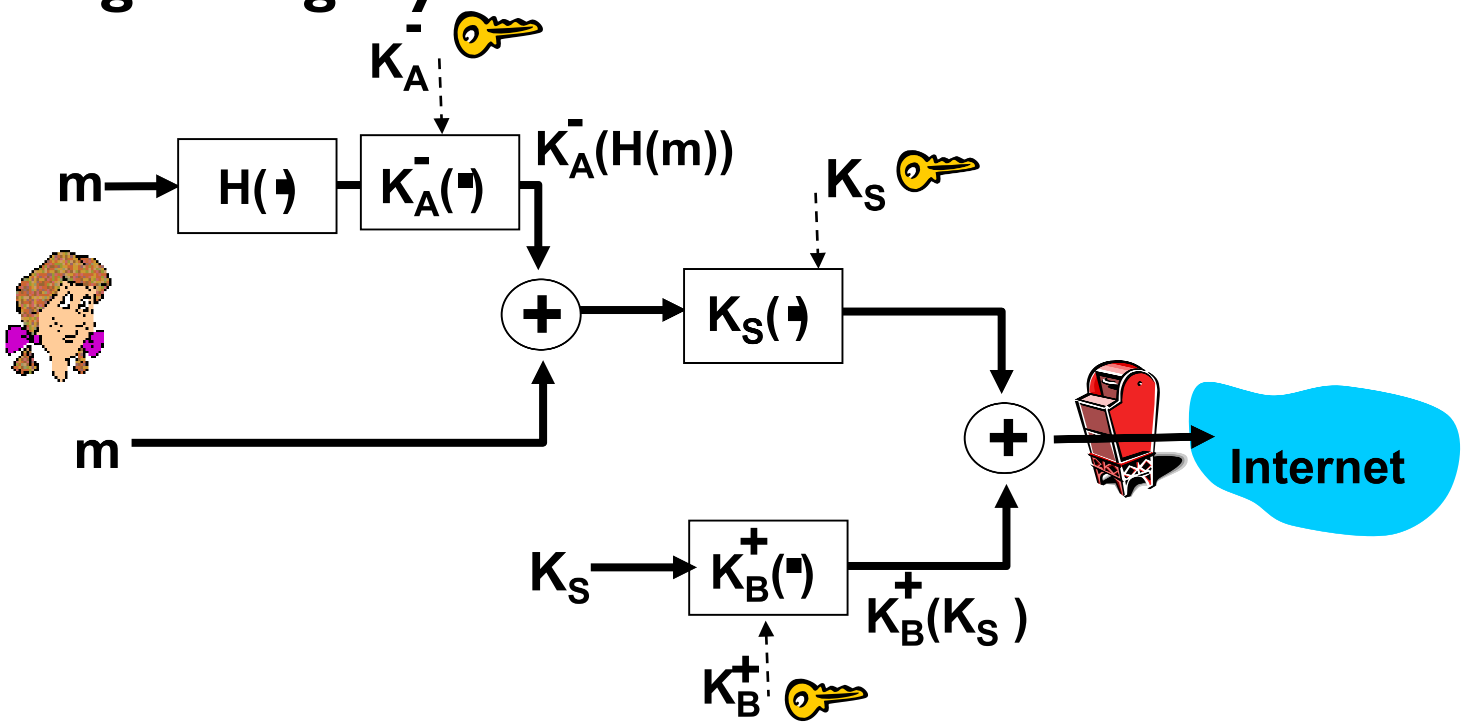
Alice wants to provide sender authentication message integrity



- **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**

Summary

- Authentication
- MAC
  - CBC-MAC
  - MD5
  - SHA-1
- Digital Signature: MAC+Encription
- Key Distribution
  - Diffie-Hellman Key Exchange
  - Trusted certification authority (CA)
  - Certificate for public key

Homework

- 第八章：R15, P9, P16, P18