CSCE 560
Introduction to Computer Networking

Dr. Barry Mullins
AFIT/ENG
Bldg 642, Room 209
255-3636 x7979

Agenda

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity and digital signatures
8.4 Authentication
8.5 Securing email

What is Network Security?

Confidentiality: only sender and intended receiver should "understand" message contents

- Sender encrypts message
- Receiver decrypts message

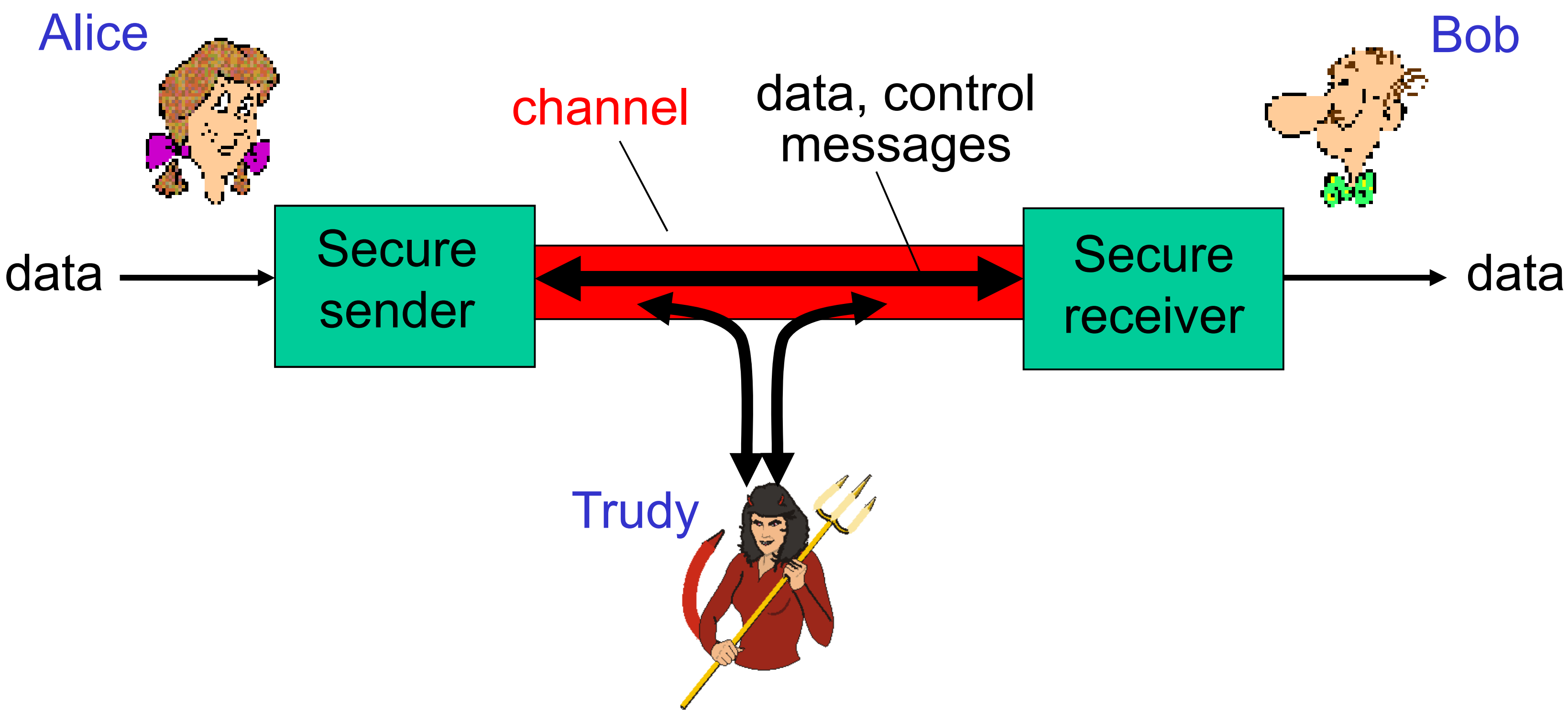
Message Integrity: sender and receiver want to ensure message not altered (in transit or afterwards) without detection

Authentication: sender and receiver want to confirm identity of each other

Availability: services must be accessible and available to users

Friends and Enemies: Alice, Bob, Trudy

- Well known in network security world
- Names used by Ron Rivest in 1978 article and stuck
- Alice and Bob want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who Might Alice and Bob Be?

- … well, real-life Alices and Bobs!
- Web browser/server for electronic transactions
  - On-line purchases
  - On-line banking client/server
- DNS servers
- Routers exchanging forwarding table updates
- Wireless client and access point

What Can A Bad Actor Do?

- *Eavesdrop:* intercept messages
- Actively *insert* messages into connection
- *Impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *Hijacking:* "take over" ongoing connection by removing sender or receiver inserting himself in place
- *Denial of service:* prevent service from being used by others (e.g., by overloading resources)
- Details discussed in CSCE 629

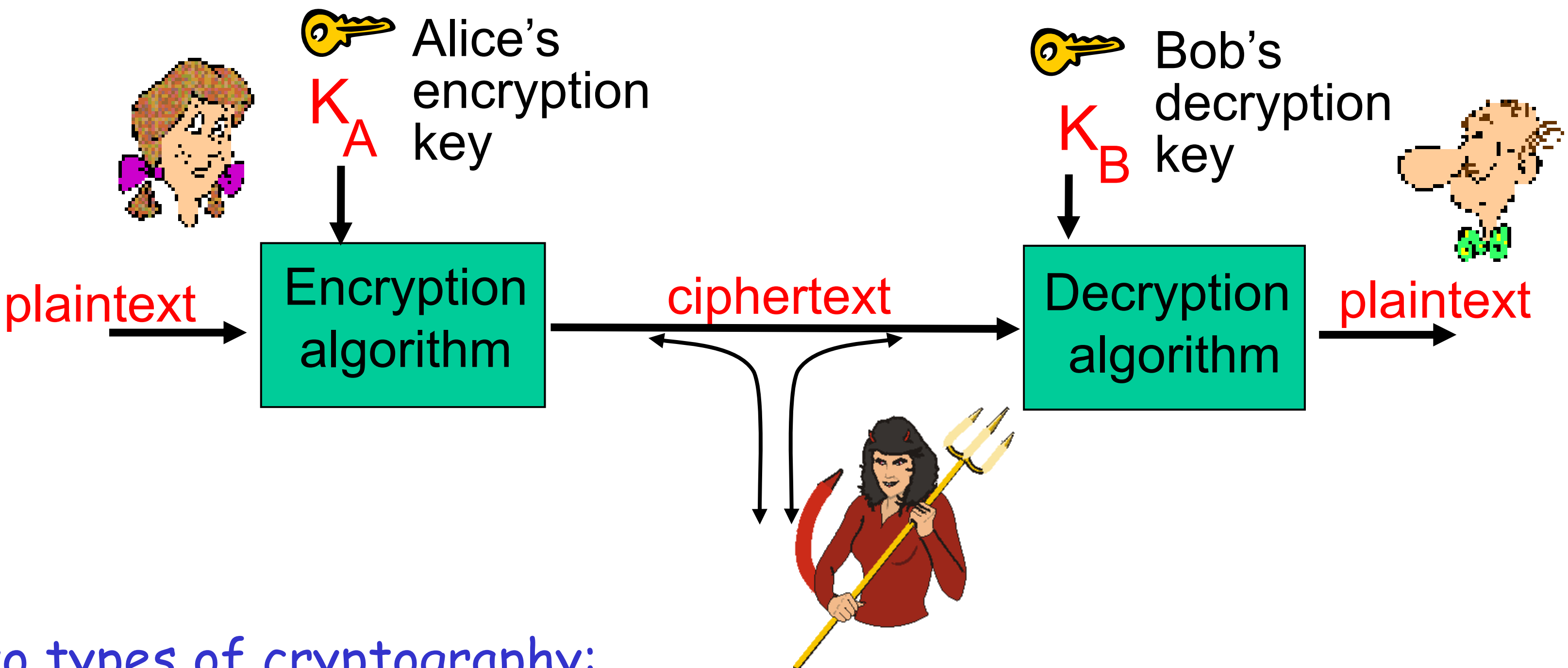
Agenda

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity and digital signatures
8.4 Authentication
8.5 Securing email

Basic Terminology

- Plain Text - Original data: not disguised
- Cipher (Encrypted) Text - Unintelligible to intruder
  - Data disguised using encryption algorithm
- Algorithm
  - Should be public and known to all
    - Inspires trust that the algorithm works
    - Proprietary algorithms are <u>not</u> the answer
- Key
  - String of bits represented as alphanumerics used as input to control how encryption algorithm disguises plain text
  - Should be long enough to prevent easy breaking yet short enough to keep algorithm efficient
  - Typical key lengths in bits: 56, 128, 256, 512

The Language of Cryptography



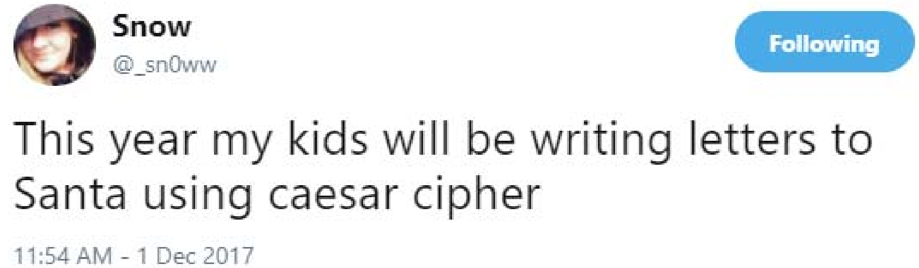
Two types of cryptography:

Symmetric key crypto: sender and receiver keys *identical*

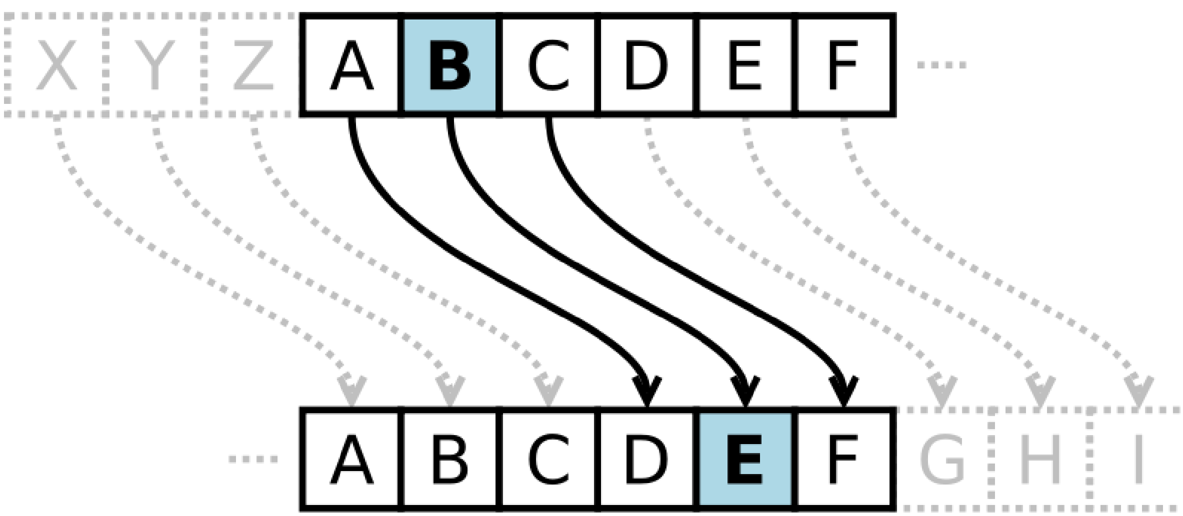
Public-key crypto:

- encryption key → public
- decryption key → private (secret)

Symmetric Key Crypto Substitution Ciphers



- Every letter (or group of letters) is replaced by another letter (or group of letters)
- Example
  - Caesar's (ROTn) cipher: substitute with letter "n" positions later
  - "n" is the key
    - A/D, B/E, C/F, D/G, …, Z/C → n = 3
    - WARNING! → ROT26 has been cracked



Once it is known that Caesar cipher is being used, it is easy to break the code (only 25 possible key values)

Symmetric Key Crypto
Substitution Ciphers

- Monoalphabetic cipher: substitute one letter for another

Plaintext letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext letter:	m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

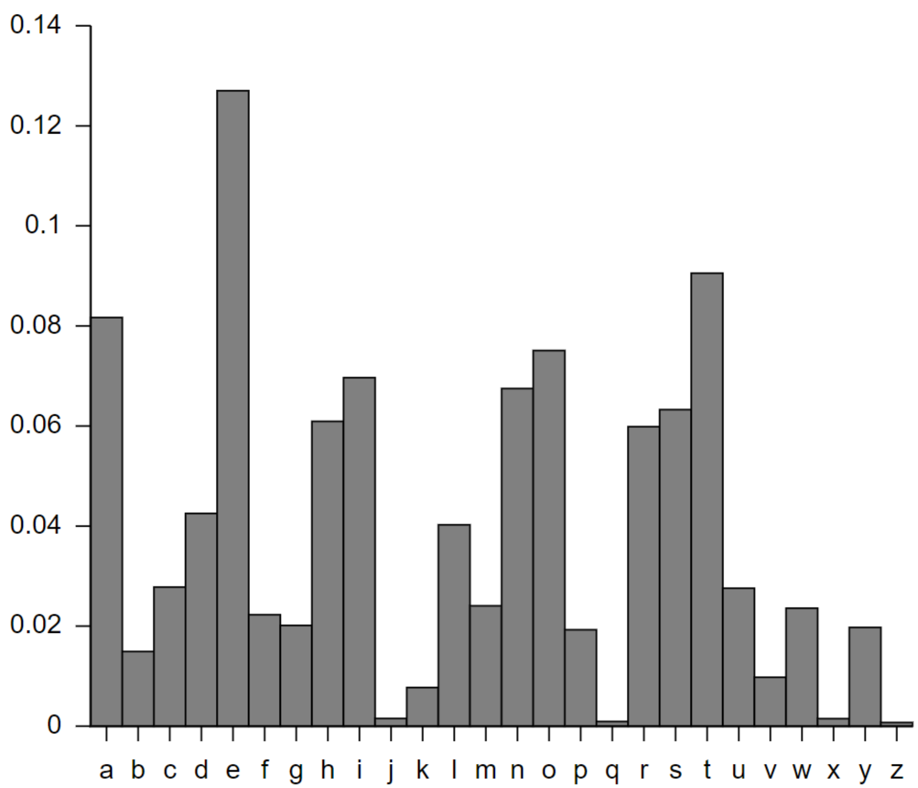
`Plaintext: bob, i love you. alice`

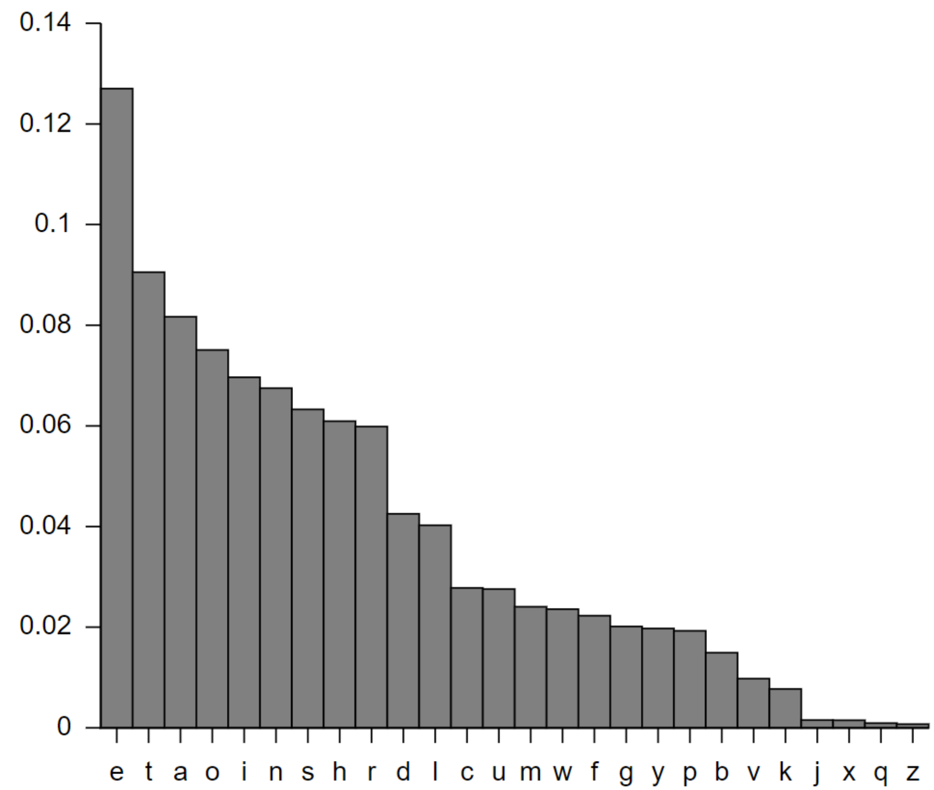
`ciphertext: nkn, s gktc wky. mgsbc`

Symmetric Key Crypto
Substitution Ciphers

Q: How hard to brute force this simple cipher?

- 26! (403×10^{24}) possible pairings of letters - so breaking code is not as easy as Caesar cipher
- Statistical analysis of plain text language can help in breaking the code faster





Symmetric Key Crypto
Substitution Ciphers

- Polyalphabetic cipher: use multiple monoalphabetic ciphers
- Choose either C1 or C2 based on some predetermined sequence (e.g., C1, C2, C2, C1, C2, and repeat)

Plaintext letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
$C_1(k = 5)$:	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
$C_2(k = 19)$:	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s

Plaintext: bob, i love you.

ciphertext: ghu, n etox dhz.

- Note the first b is encrypted using C_1 while the second b uses C_2

Symmetric Key Crypto
Transposition Ciphers

- Instead of substituting letters in the plaintext, we change their order

A	N	D	R	E	W
1	4	2	5	3	6
t	h	i	s	i	s
a	m	e	s	s	a
g	e	i	w	o	u
l	d	l	i	k	e
t	o	e	n	c	r
y	p	t	n	o	w

Key = ANDREW

Plaintext = thisisamessageiwouldliketoencryptnow
Ciphertext = tagltyieiletisokcohmedopsswinnsauerw

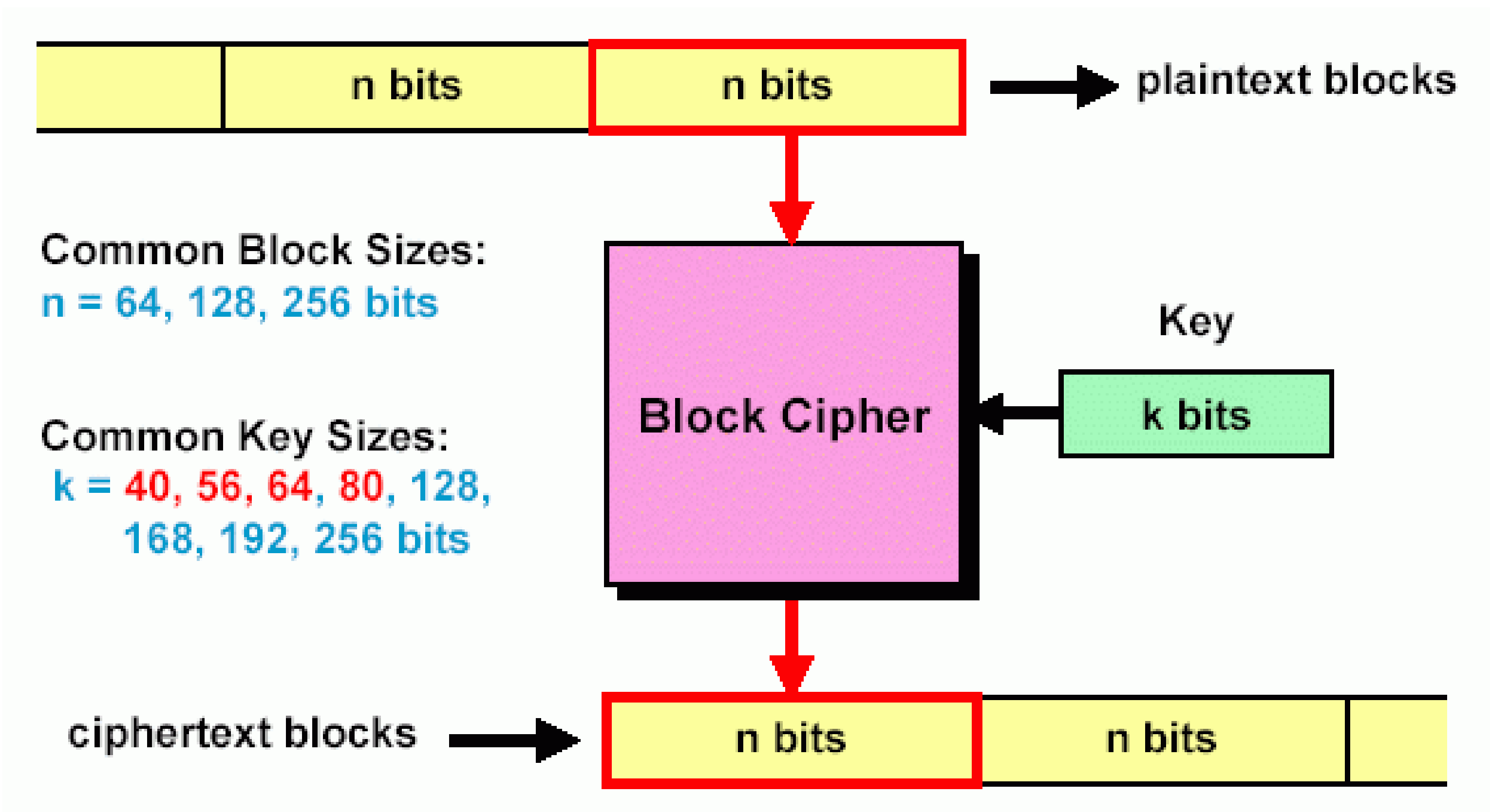
Symmetric Key Crypto: DES

- Most encryption algorithms use a complex combination of substitution and transposition

- DES: Data Encryption Standard – Became standard in 1977
  - Multiple iterations of substitution and transposition using a 56-bit symmetric key and 64-bit blocks
  - How secure is DES?
    - DES Challenge: 56-bit-key-encrypted phrase brute forced:
      - 4 months – 1997
      - 41 days – Feb 1998
      - 56 hours – Jul 1998
      - 22 hours and 15 minutes – Jan 1999
    - No known "backdoor" decryption approach
- Making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys
    - Actually: encrypt → decrypt → encrypt

Block Ciphers

- Divide input bit stream into n-bit sections, encrypt only that section, no dependency/history between sections
- In a good block cipher, each output bit is a function of all n input bits and all k key bits



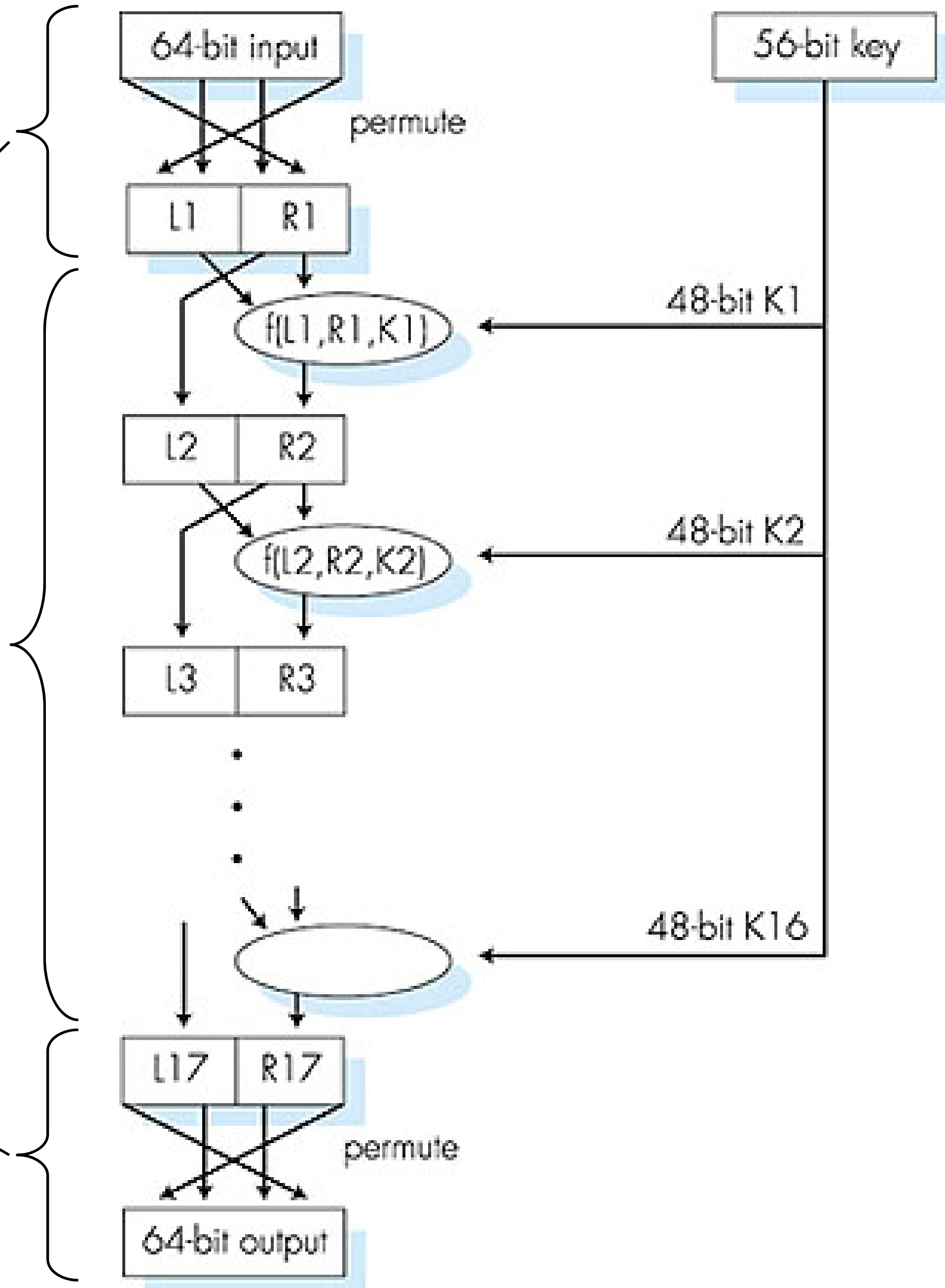
Symmetric Key Crypto: DES

DES operation

Initial permutation

16 identical "rounds" of function application, each using different 48 bits of key material

Final permutation



Symmetric Key Crypto: AES: Advanced Encryption Standard

- DES has small key space
- 3DES has high computational costs

- Enter AES as a NIST standard—2002
  - Processes data in 128-bit blocks
  - 128, 192, or 256 bit keys
  - Brute force decryption
    - DES: Assume a machine that can brute force DES in 1 sec (instead of ~22 hours)
    - AES: To brute force 128-bit AES, same machine would take 149 trillion years

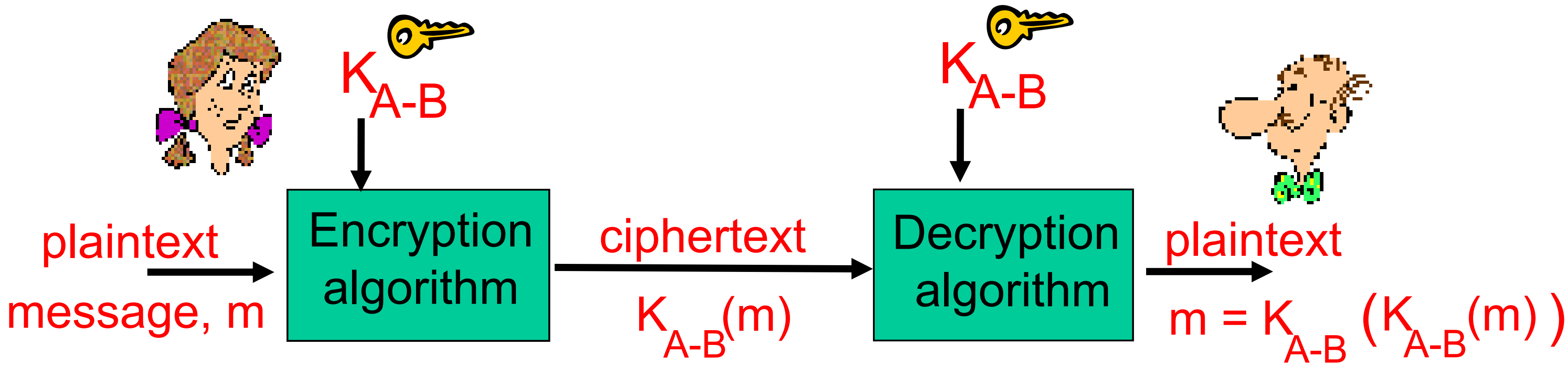
$$\frac{2^{(128bits-56bits)}}{60*60*24*365} = 149.75x10^{12}\ years$$

National Institute of Standards and Technology

Symmetric Key Cryptography

Symmetric key crypto: Bob and Alice share same (symmetric) key: K_{A-B}

Q: How do Bob and Alice agree on key value?



Symmetric Key Crypto

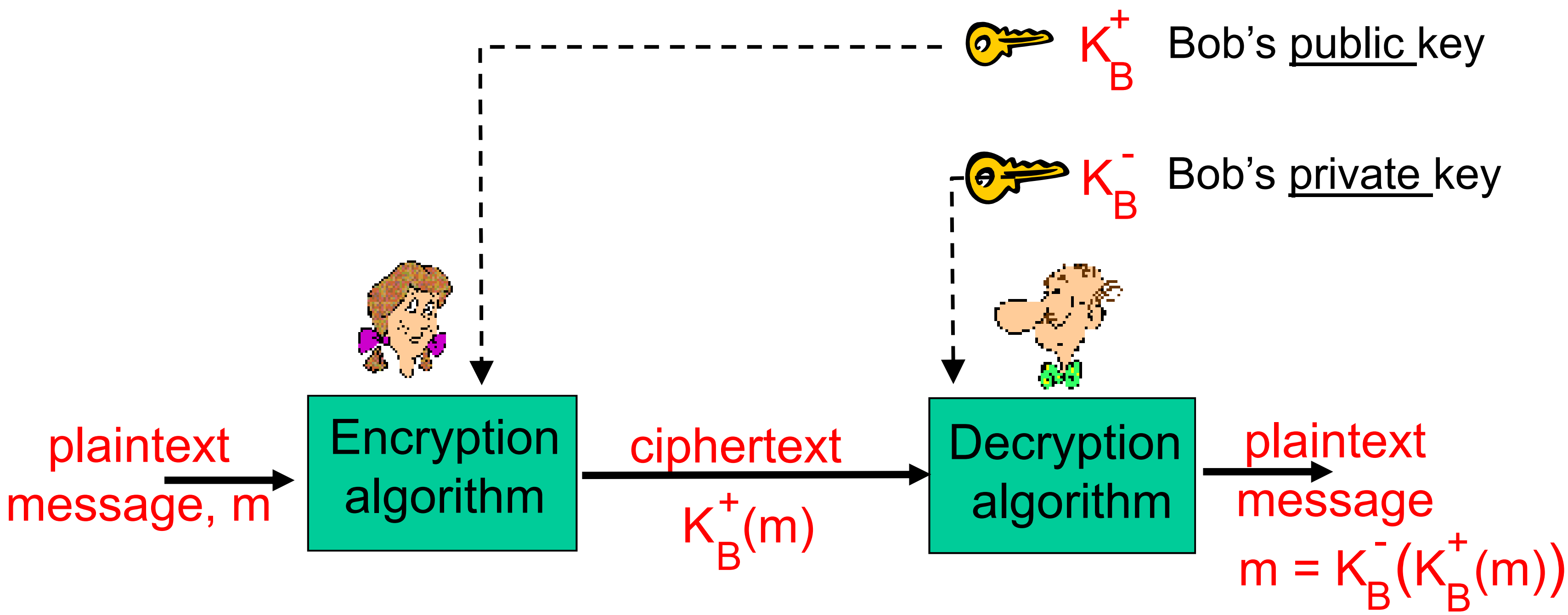
- Problem with all the cryptography algorithms discussed so far
  - If the key is stolen, any message can be decrypted

- Is there a way to do cryptography without worrying about this?

Yes, public key cryptography!

Public Key Cryptography aka Asymmetric Encryption

- Sender and receiver do not share secret key
- Each user (e.g., Bob) holds two different keys
  - Private key to decrypt messages sent to Bob
  - Public key everyone uses to encrypt messages to send to Bob



Public Key Encryption Algorithms

Requirements:

1. Need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ 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, Adleman) algorithm satisfies these requirements

RSA: Choosing Keys

1. Choose two large prime numbers p, q
   Minimum of 1024 bits recommended; CACs use 2048 bits

2. Compute n = pq   and   z (totient) = (p-1)(q-1)

3. Choose e (with 1 < e < z) that has no common factors with z   (e, z are "relatively prime" or gcd(e, z) = 1)

4. Choose d such that ed-1 is exactly divisible by z (i.e., ed mod z = 1)

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

RSA: Encryption, Decryption

0. Given (n,e) and (n,d)

1. To encrypt bit pattern, m, 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

Choose p=5, q=7 → n=35, z=24

Choose e=5 (so e, z relatively prime)

Choose d=29 (so ed-1 exactly divisible by z) → (ed mod z = 1)

[(5*d)-1] / 24 = integer → d = 5, 29, 53, 77…

encrypt:	letter	m	m^e	$c = m^e \bmod n$
	l	12	248832	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:

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

Result is the same!

Session Keys

$c = m^e \bmod n$

- Exponentiation is computationally expensive
- DES is faster than RSA
  - In software → at least 100 times faster
  - In hardware → 1,000 to 10,000 times faster

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 crypto

Agenda

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity and digital signatures
8.4 Authentication
8.5 Securing email

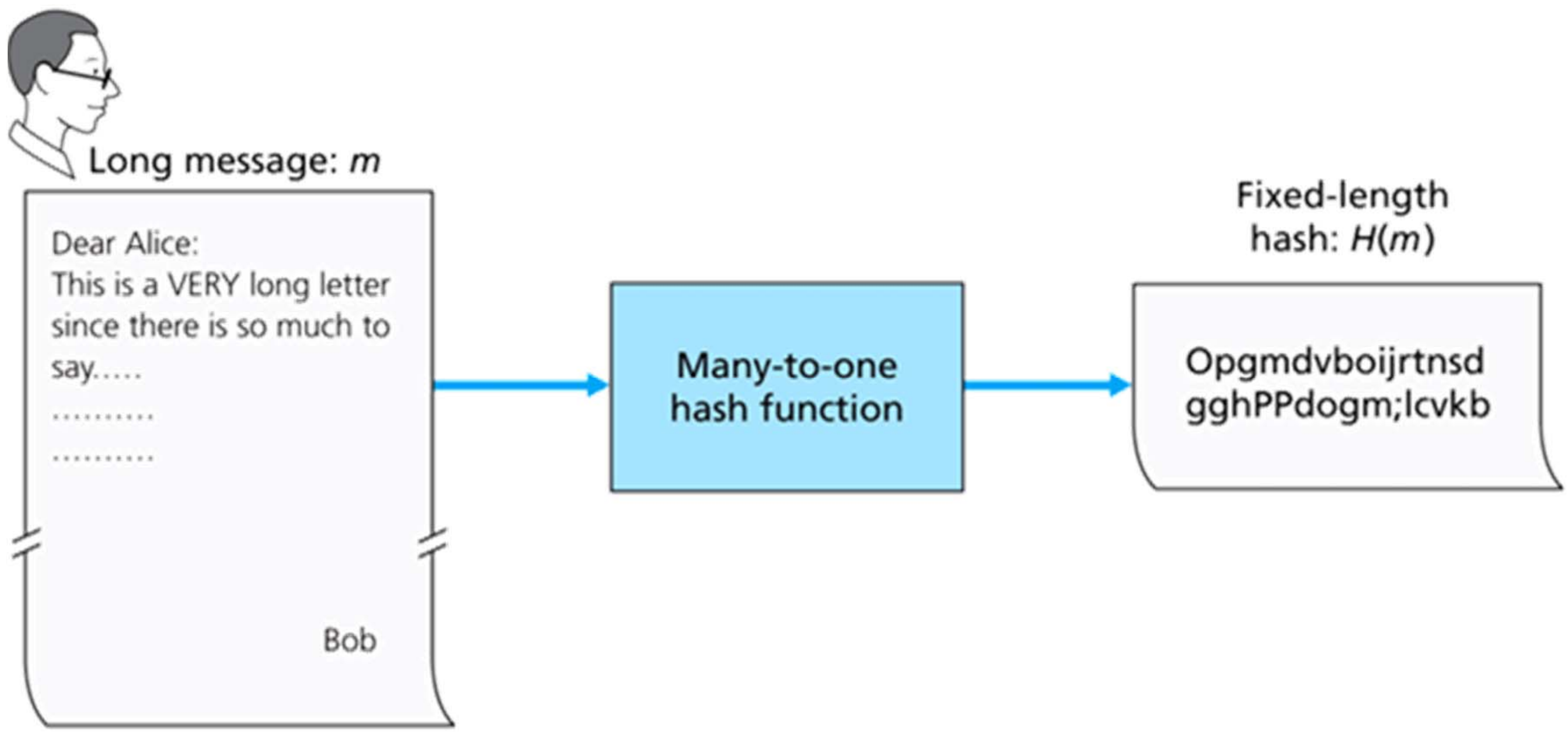
Message Integrity

- Allows communicating parties to verify that received messages are authentic
  - Source of message is who/what you think it is
  - Content of message has not been altered

- Let's first talk about message digests

Message Digests aka Cryptographic Hash

- Function that takes arbitrary length input m, produces fixed-length string, H(m)
- Computationally infeasible to find two different messages, x, y such that H(x) = H(y)
  - Given m = H(x), (x unknown), cannot determine x



Internet Checksum: Poor Crypto Hash Function

Internet checksum has some properties of hash function:

- ✓ produces fixed length digest (16-bit sum) of message
- ✗ But given message with given hash value, it is easy to find another message with same hash value
- Original message is **IOU100.99BOB**
- Altered message is **IOU900.19BOB**

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

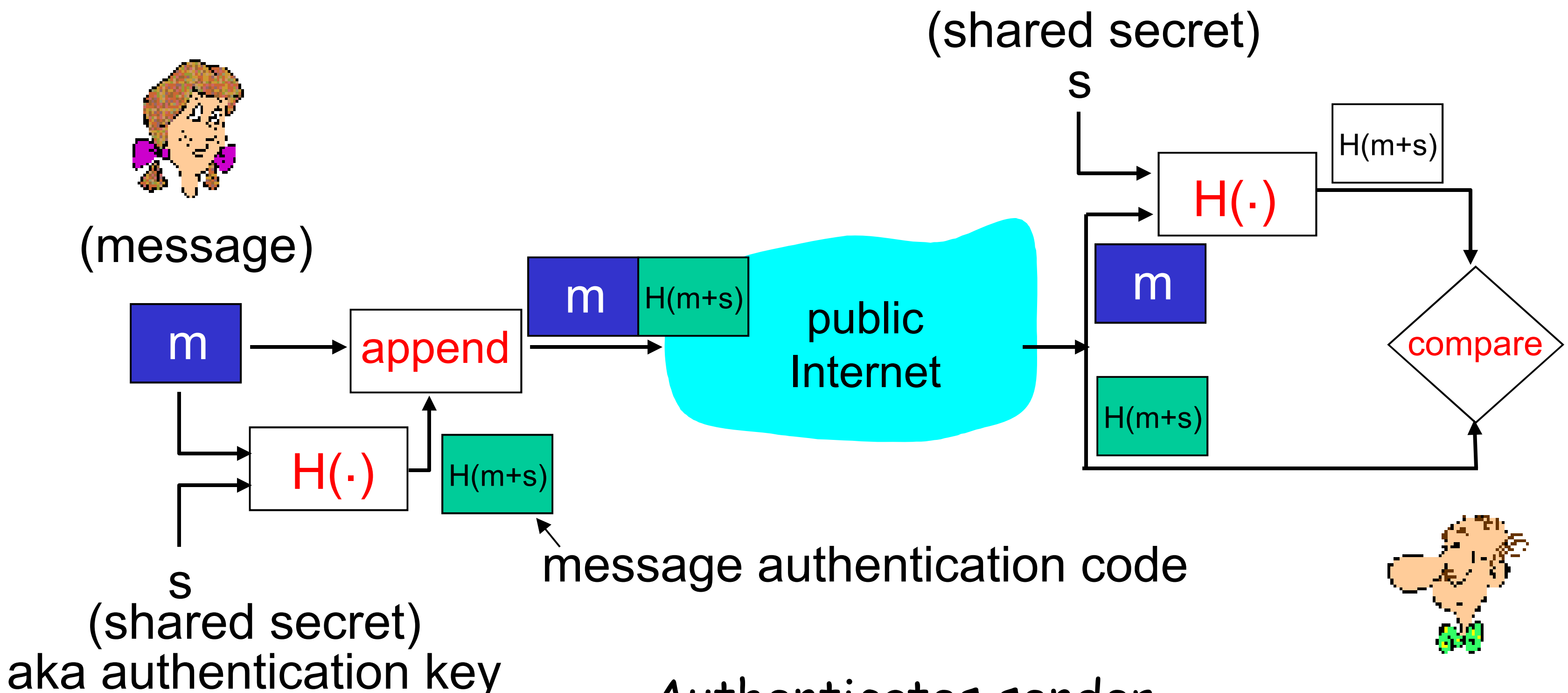
Different messages
but identical checksums!

Hash-based Message Authentication Code (HMAC)

- MD5 hash function designed by Rivest in 1992 (RFC 1321)
  - Computes 128-bit message digest
  - Change anything in the message and hash is vastly different
    - H("Barry Mullins") = 566565478e370616d1f484519a2ea7aa
    - H("barry Mullins") = 1a0aa383eb6cc2793d6ea6e88428104a
  - Security of MD5 hash function "severely" compromised
    - Collision attack can find collisions within seconds

- SHA1 hash function designed by NSA in 1995 (RFC 3174)
  - 160-bit message digest
  - In 2017 Google said they performed a collision attack

Message Authentication Code (MAC)
How We Can Perform Message Integrity



- Authenticates sender
- Verifies message integrity
- No encryption!
- Also called "keyed hash"

Digital Signatures

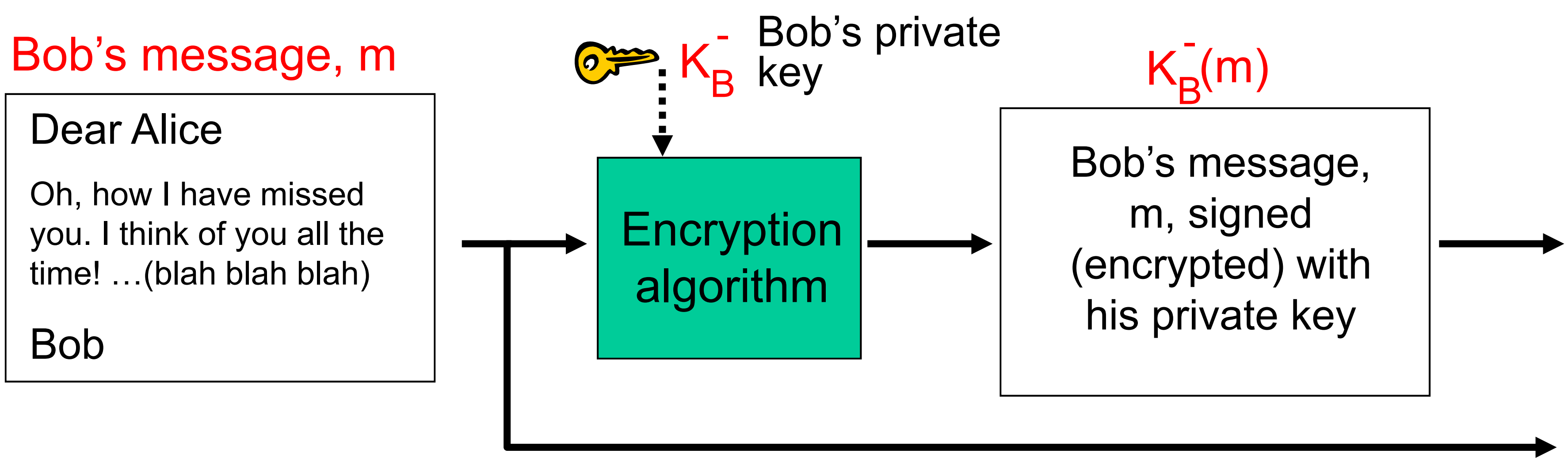
Cryptographic technique analogous to handwritten signatures

- 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
- Non-repudiable: Bob cannot deny signing the document

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital Signatures (More)

- Alice receives msg m and digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ to get $K_B^+(K_B^-(m)) = m$
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key

Alice thus verifies that:

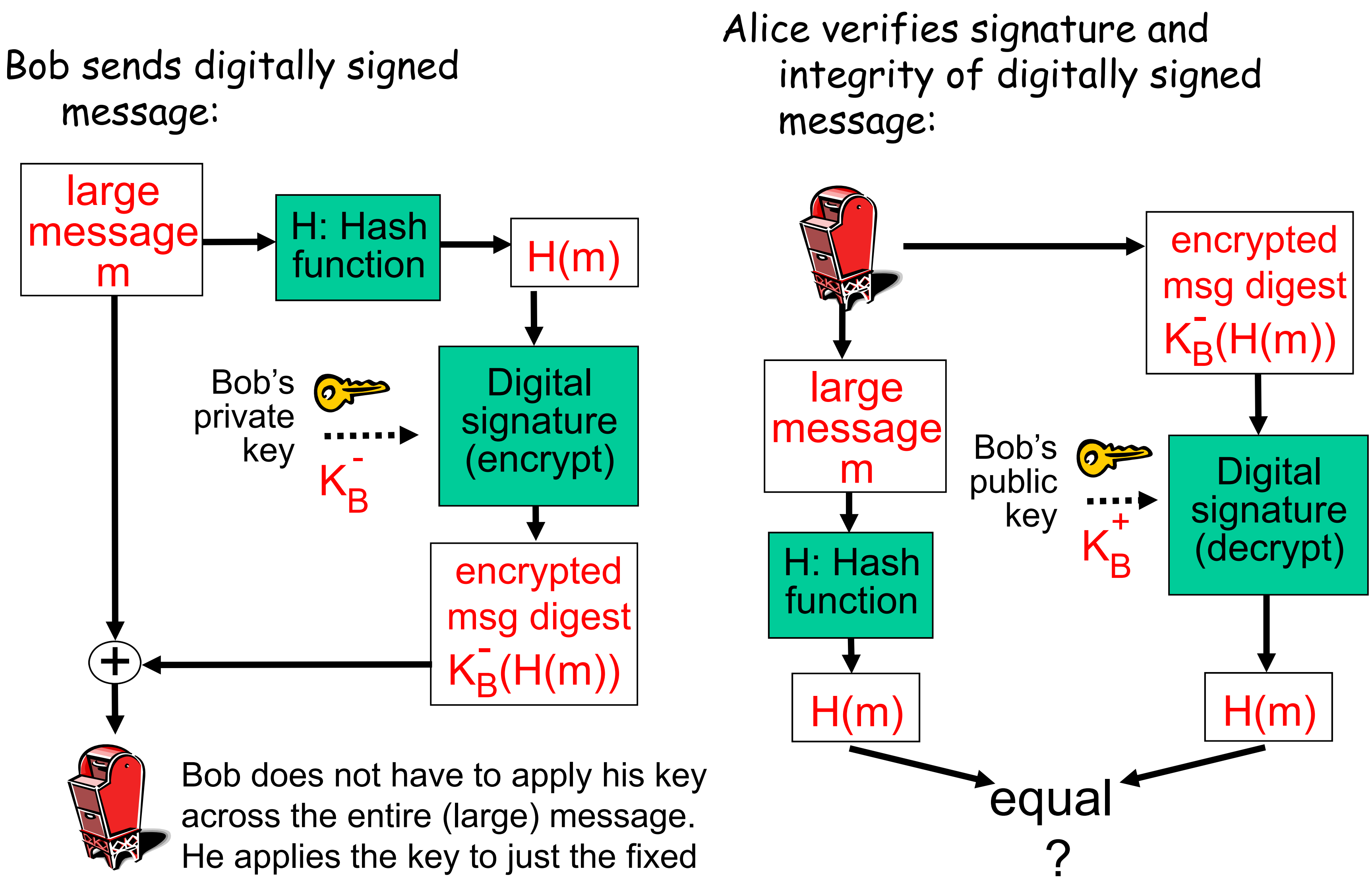
- ✓ Bob signed m
- ✓ No one else signed m
- ✓ Bob signed m and not m'

Non-repudiation:

- ✓ Alice can take m and signature $K_B^-(m)$ to court and prove that Bob signed m

Problem: Computationally expensive (encryption/decryption) to public-key-encrypt long messages

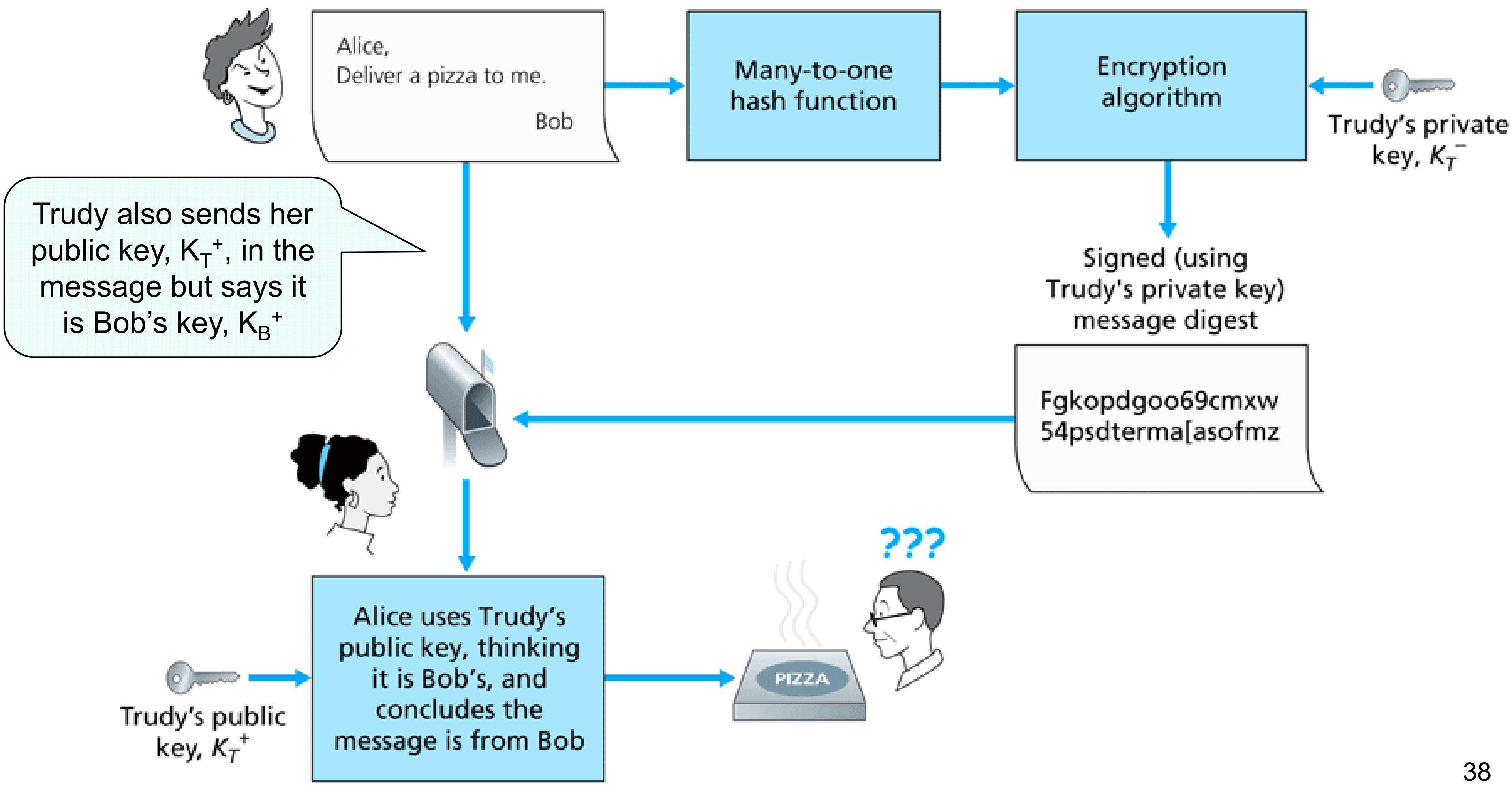
Digital Signature: Signed Message Digest



Bob does not have to apply his key across the entire (large) message. He applies the key to just the fixed length hash.

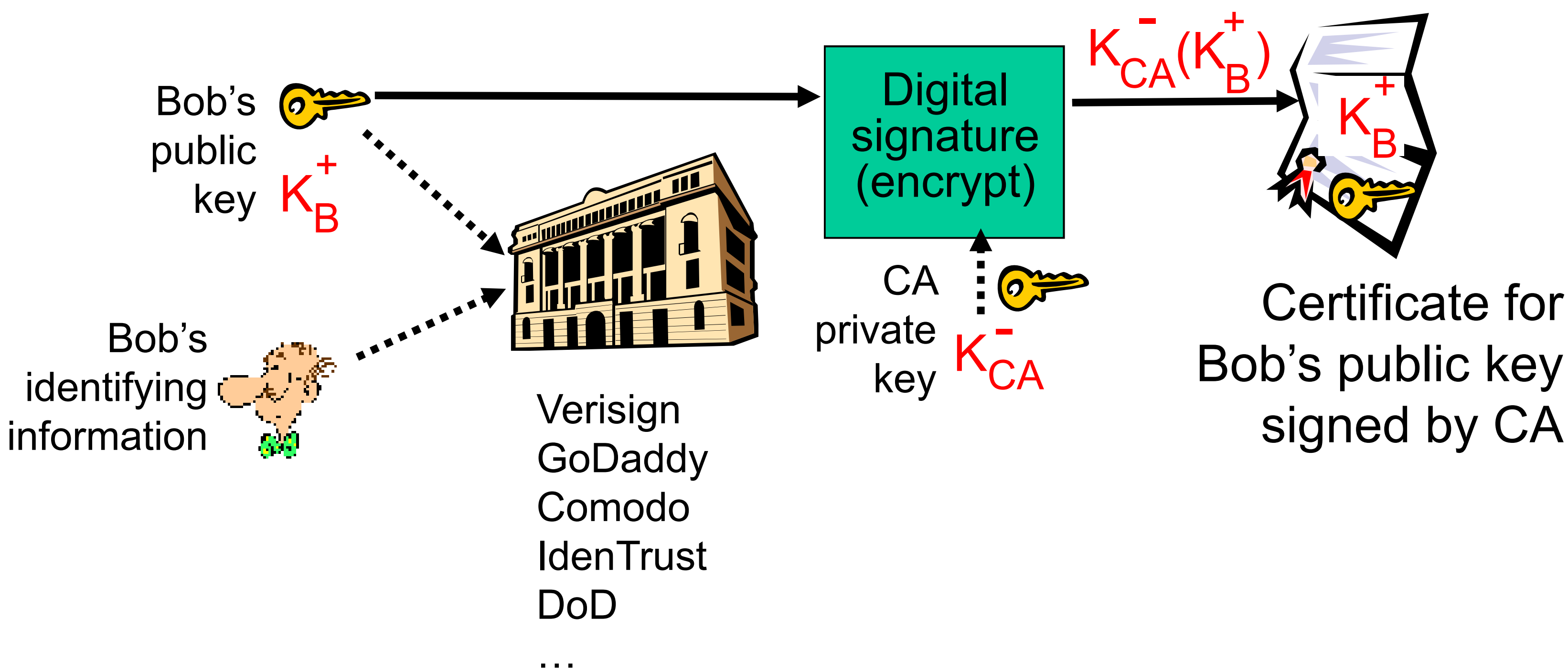
Certification Authorities

- When Alice obtains Bob's public key (from web site, e-mail, CD, thumbdrive), how does she know it is Bob's public key, not Trudy's?
- We need to know for sure we have the correct key!



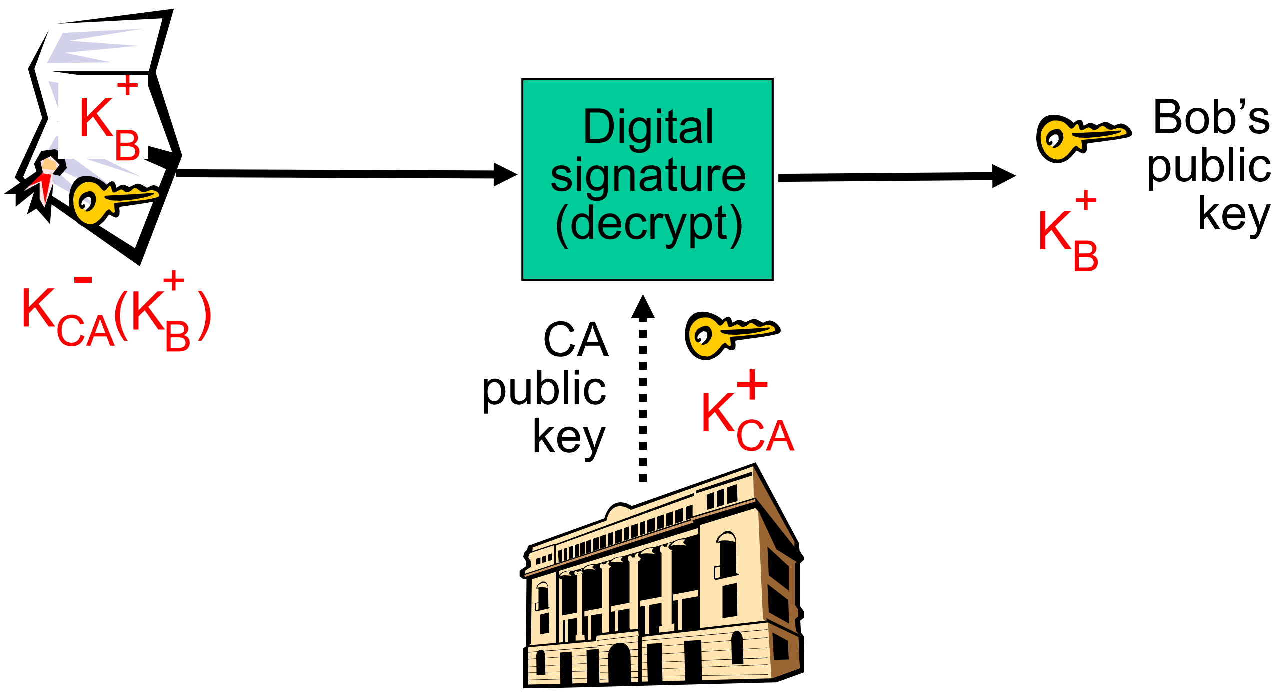
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E
- Bob registers his public key with CA
  - Bob provides "proof of identity" to CA
  - CA creates certificate binding Bob to his public key
  - Certificate containing Bob's public key digitally signed by CA
    - CA says "This is Bob's public key"



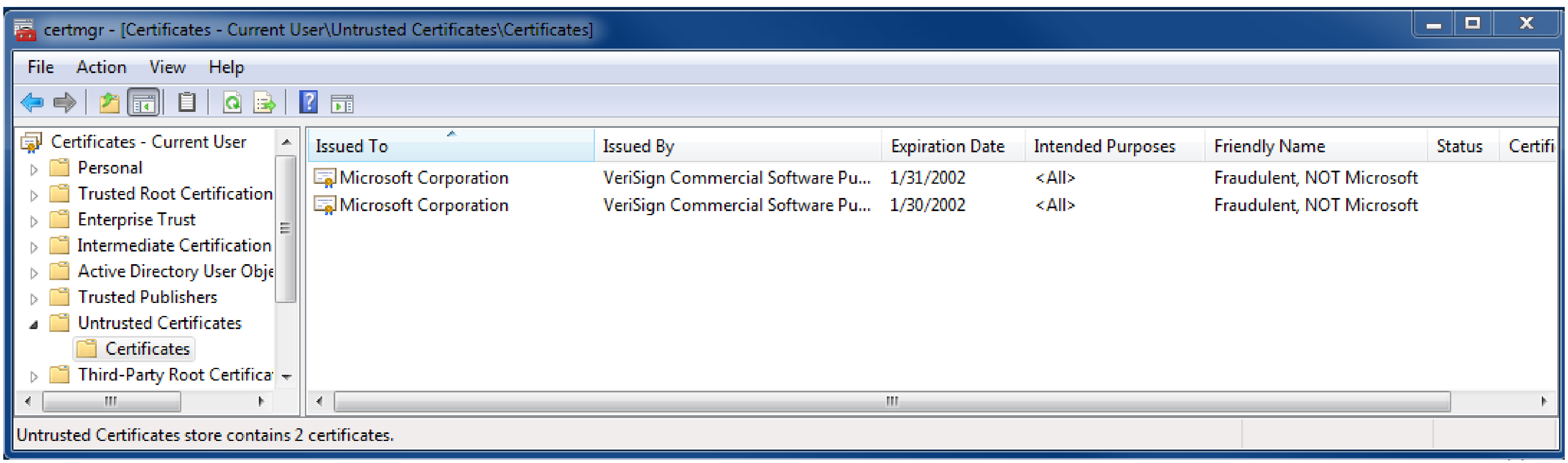
Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere)
  - Apply CA's public key to Bob's certificate to get Bob's public key



It's All About Trust…

- You can trust the identity associated with a public key only to the extend that you can trust a CA and its ID verification process
- In mid-March 2001, VeriSign advised Microsoft that on January 29 and 30, 2001, it issued two VeriSign Class 3 code-signing digital certificates to an individual who fraudulently claimed to be a Microsoft employee. The common name assigned to both certificates is "Microsoft Corporation".
- To view certificates
  - Start → Run → certmgr.msc



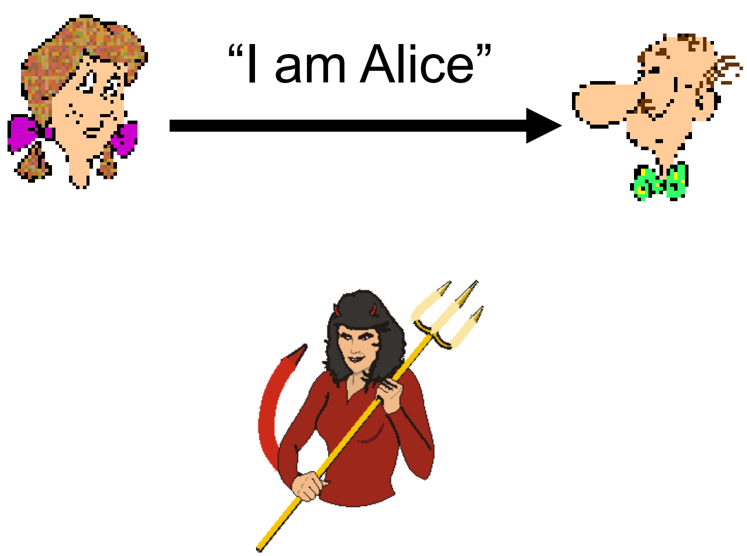
Agenda

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity and digital signatures
8.4 Authentication
8.5 Securing email

Authentication

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

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

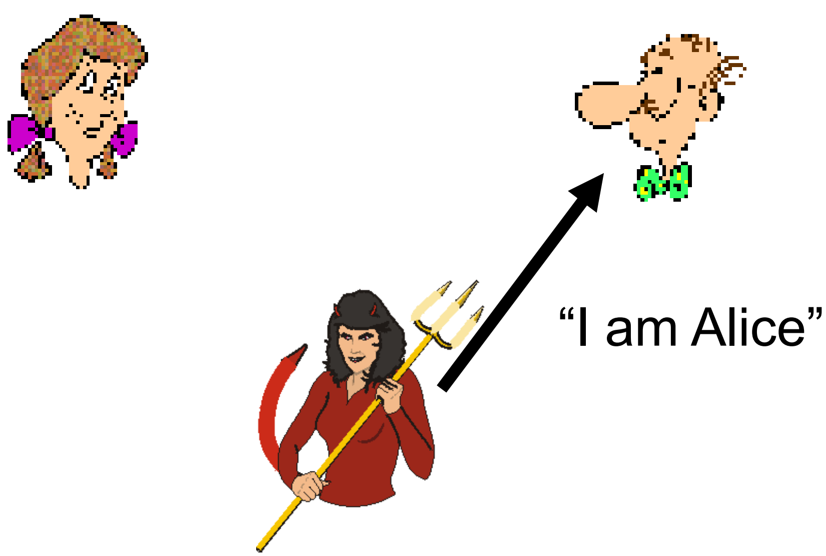


Failure scenario??

Authentication

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

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



In a network,
Bob cannot "see" Alice, so
Trudy can simply declare
herself to be Alice

Authentication: Another Try

Protocol ap2.0

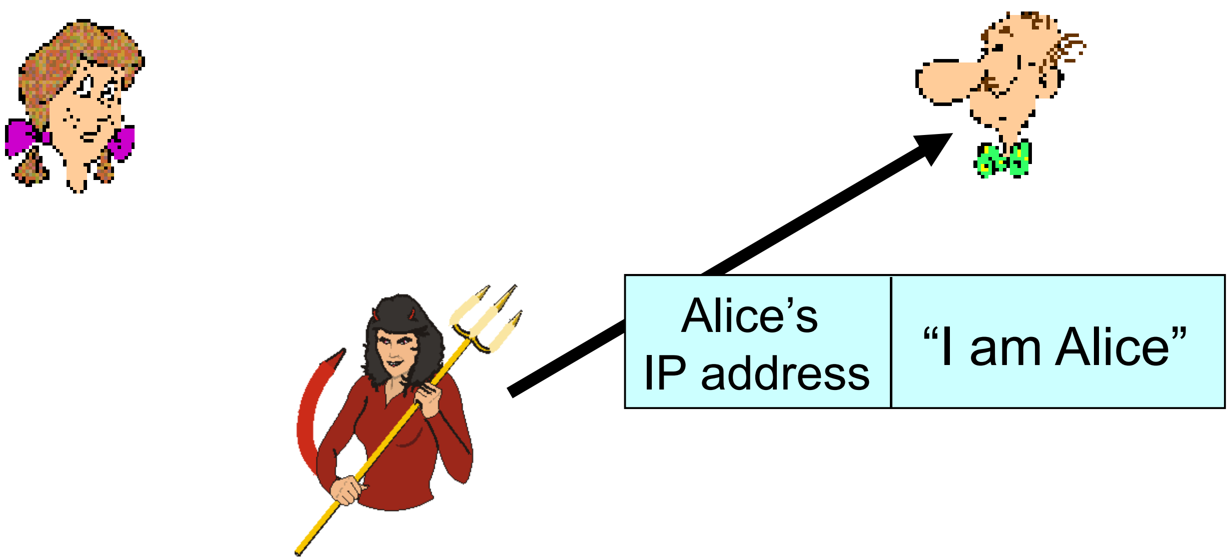
- Alice says "I am Alice" in an IP packet containing her source IP address and
- Bob already knows Alice's IP address

Failure scenario??

Authentication: Another Try

Protocol ap2.0

- Alice says "I am Alice" in an IP packet containing her source IP address and
- Bob already knows Alice's IP address

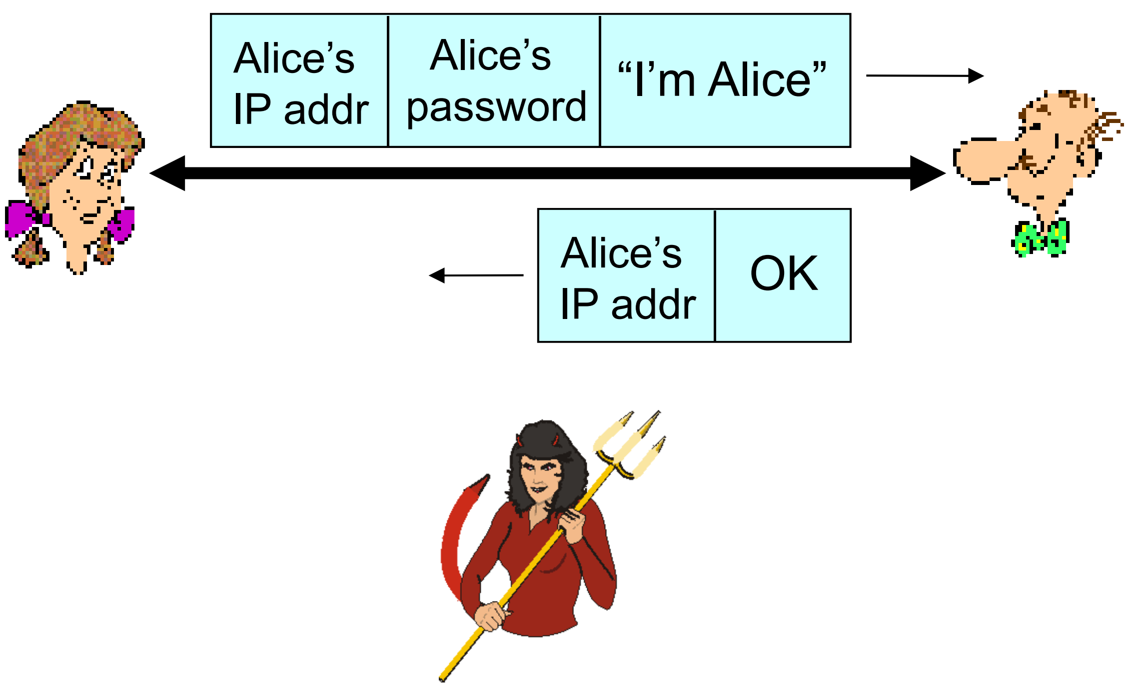


Trudy can create a packet "spoofing" Alice's address

Authentication: Another Try

Protocol ap3.0

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

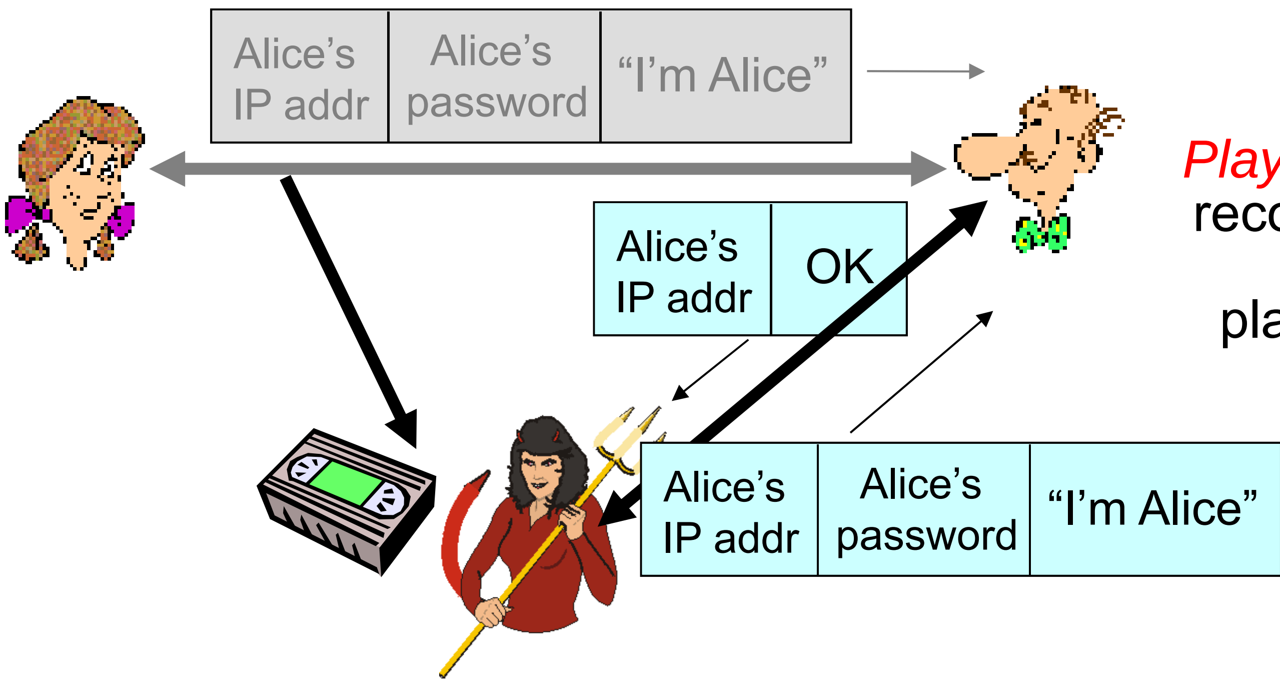


Failure scenario??

Authentication: Another Try

Protocol ap3.0

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

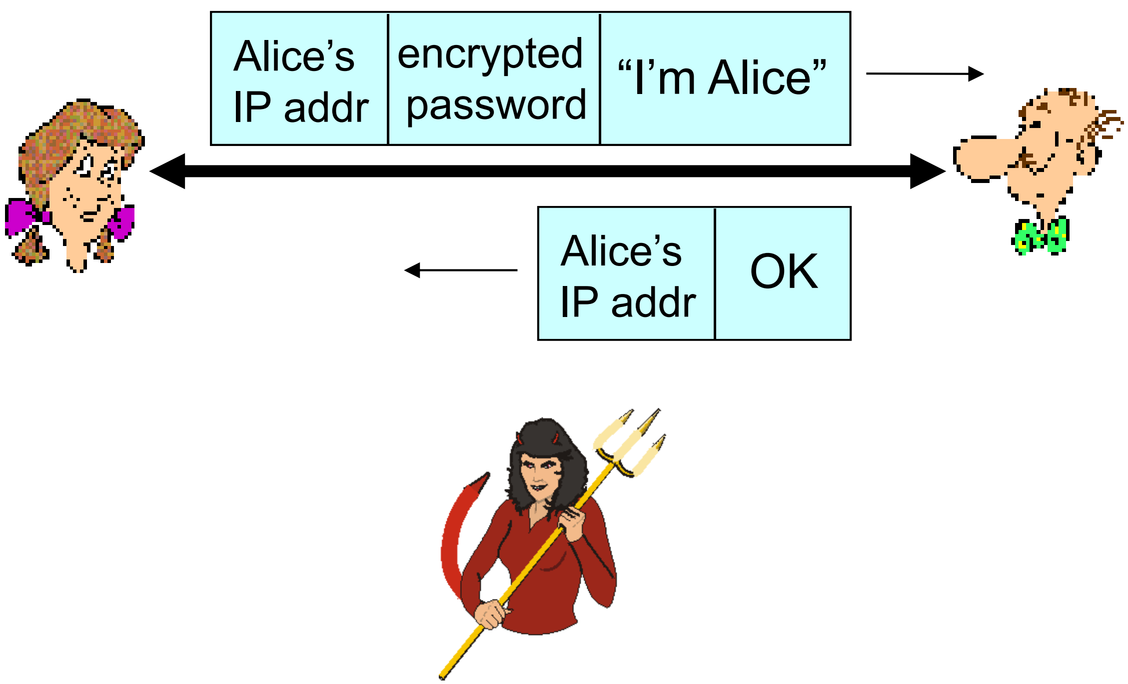


Playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: Yet Another Try

Protocol ap3.1

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

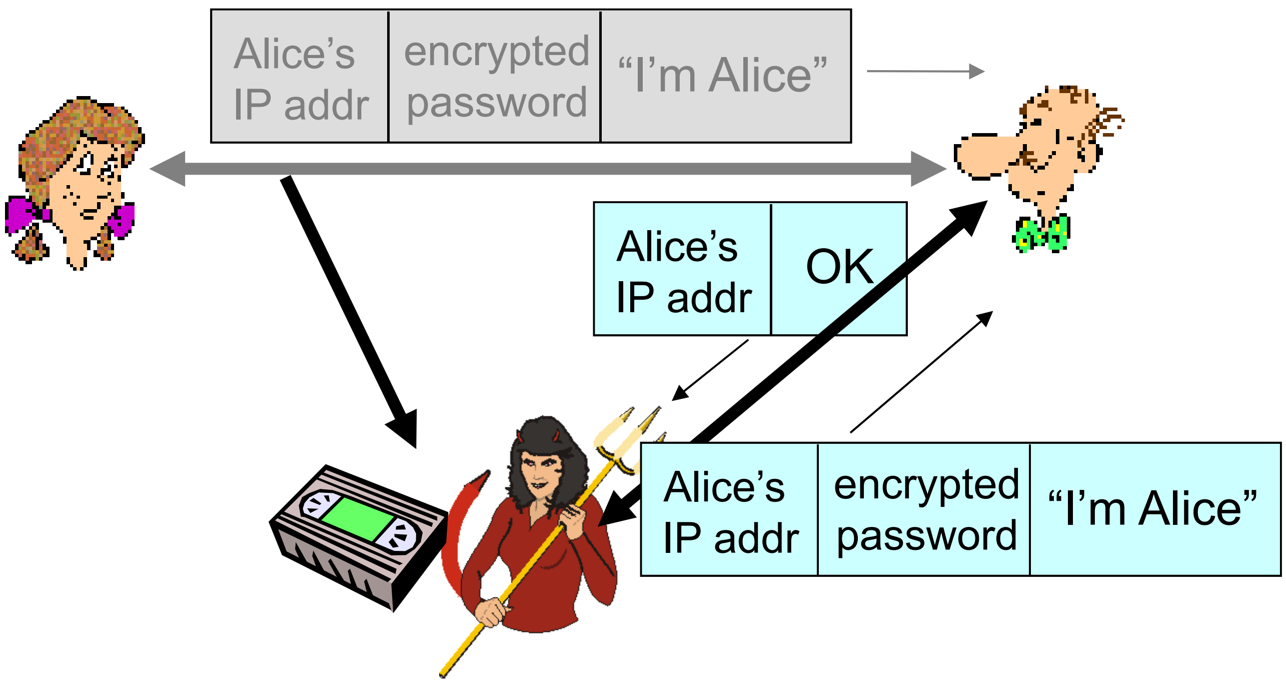


Failure scenario??

Authentication: Another Try

Protocol ap3.1

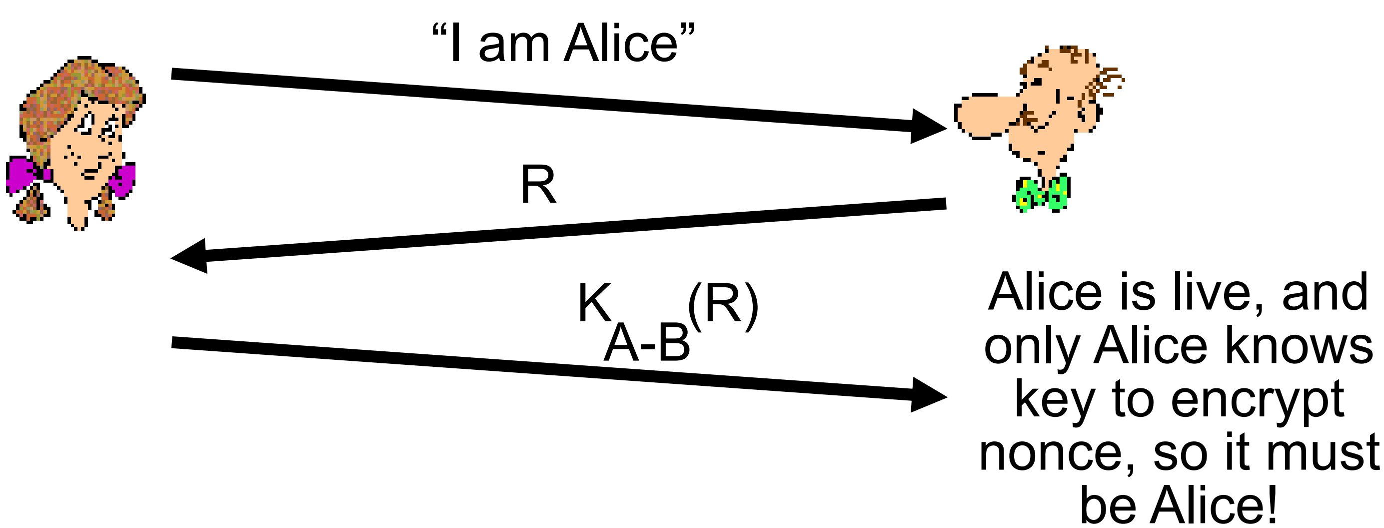
- Alice says "I am Alice" and sends her encrypted secret password to "prove" it.



Record and playback still works!

Authentication: Yet Another Try

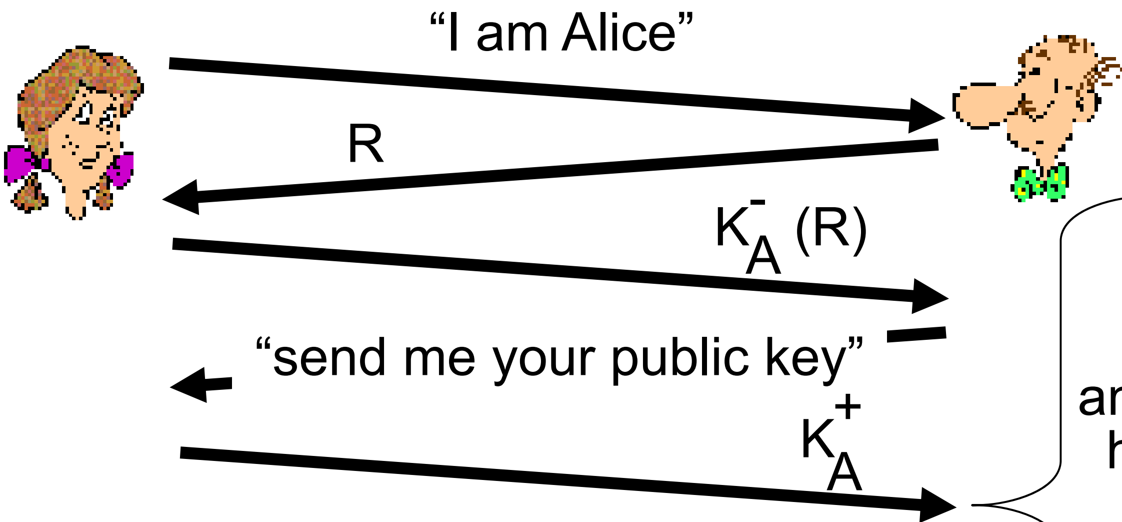
- Goal: Avoid playback attack
- Nonce: Number (R) used only *once-in-a-lifetime*
- ap4.0: To prove Alice is "live", Bob sends Alice nonce, R
  Alice must return R, encrypted with shared secret key



Failures, drawbacks?

Authentication: ap5.0

- ap4.0 requires shared symmetric key
- Can we authenticate using public key techniques?
- ap5.0: Use nonce and public key cryptography



Bob computes

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

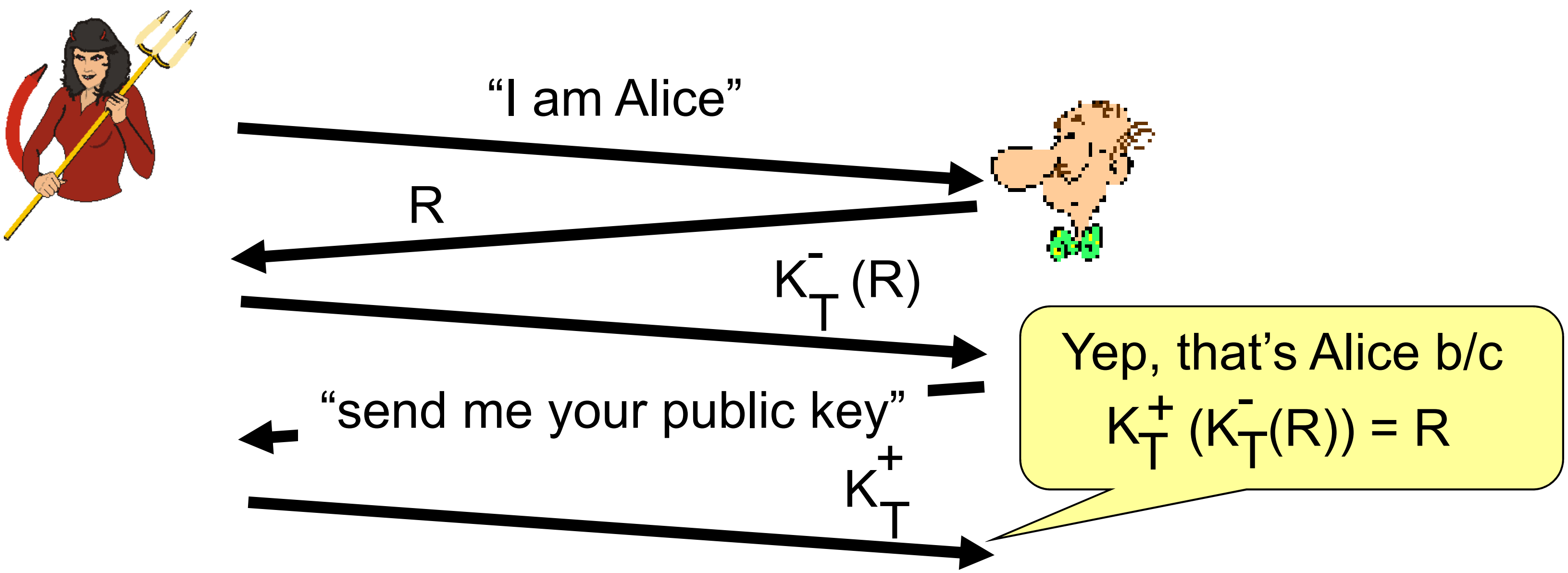
and knows only Alice could have the private key that encrypted R such that

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

Authentication: ap5.0

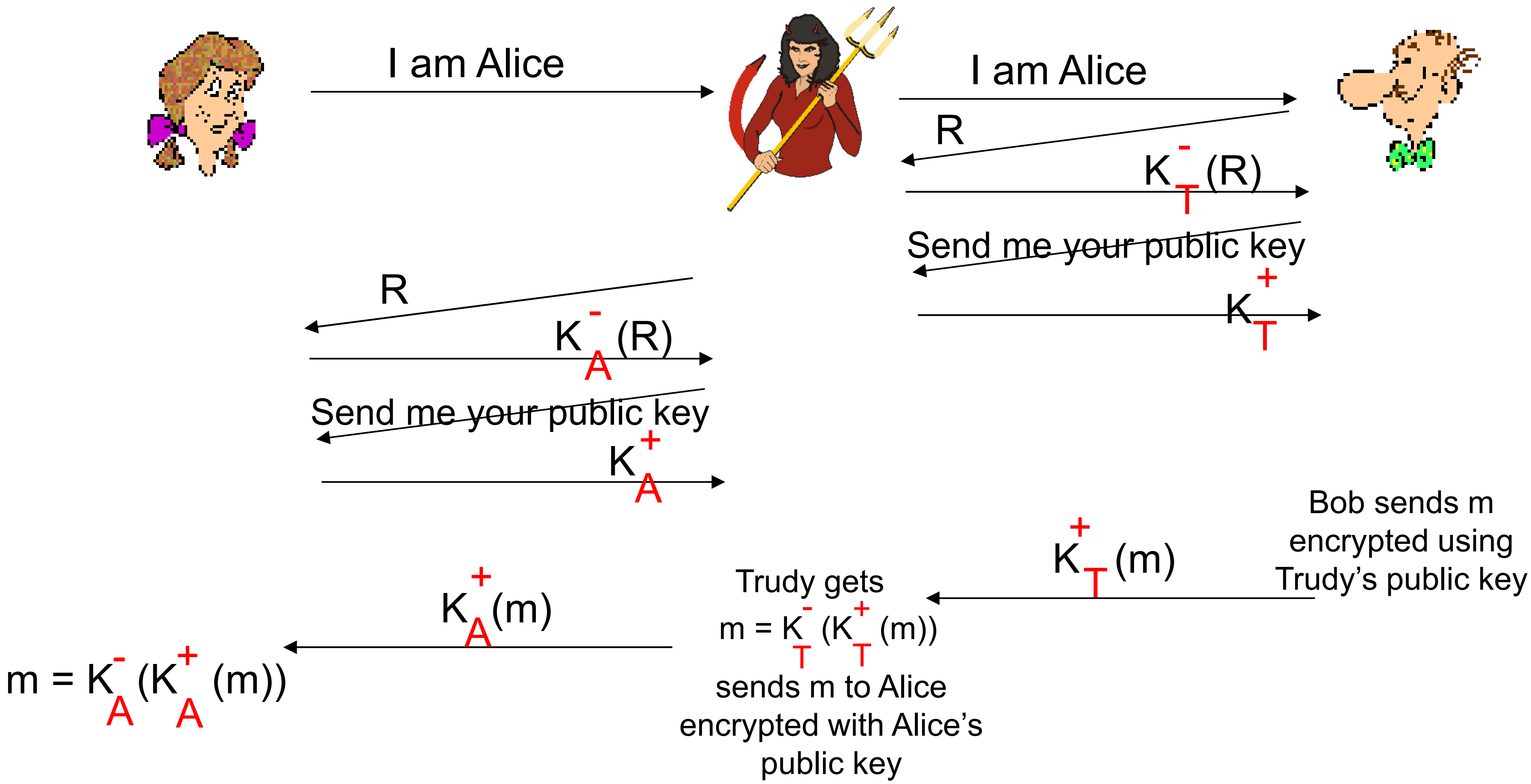
ap5.0: Use nonce and public key cryptography

Now Trudy impersonates Alice and Bob is clueless until he talks with Alice later and she denies talking with Bob



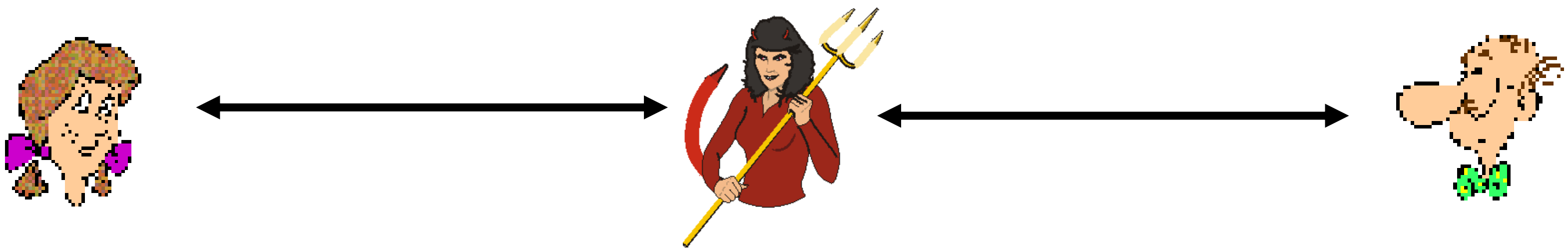
ap5.0: Security Hole

- Man in the middle attack
  - Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: Security Hole

- Man in the middle attack
  - Trudy poses as Alice (to Bob) and as Bob (to Alice)



Difficult to detect:

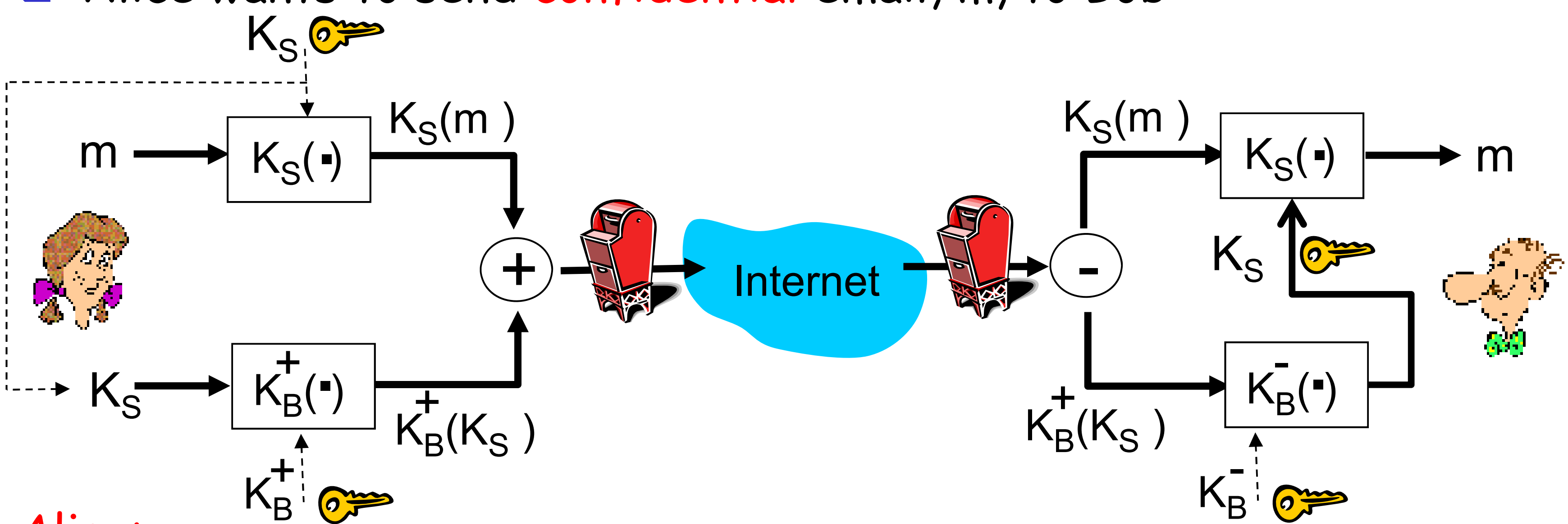
- Bob receives everything that Alice sends and vice versa
  - So Bob and Alice can meet one week later and recall the entire conversation
- Problem is that Trudy receives all messages as well!

Agenda

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity and digital signatures
8.4 Authentication
8.5 Securing email

Securing Email - C

- Alice wants to send confidential email, m, to Bob



Alice:

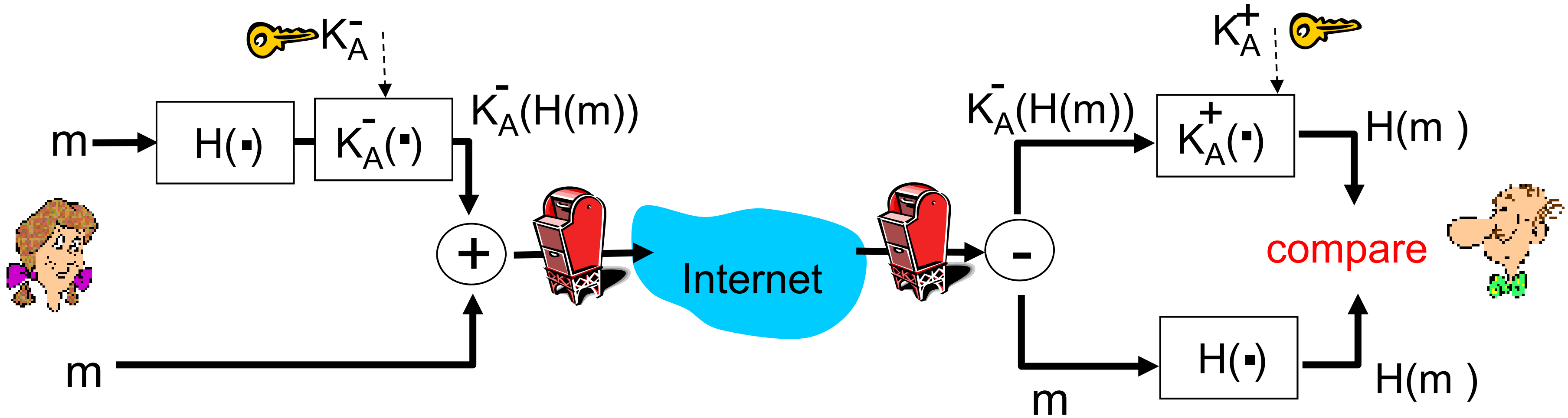
- Generates random symmetric session key, K_s, and encrypts m with K_s
  - Why use K_s? Public key encryption is inefficient for long msgs
- Also encrypts K_s with Bob's public key
- Sends both $K_S(m)$ and $K_B^+(K_S)$ to Bob

Bob:

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

Securing Email - I and A

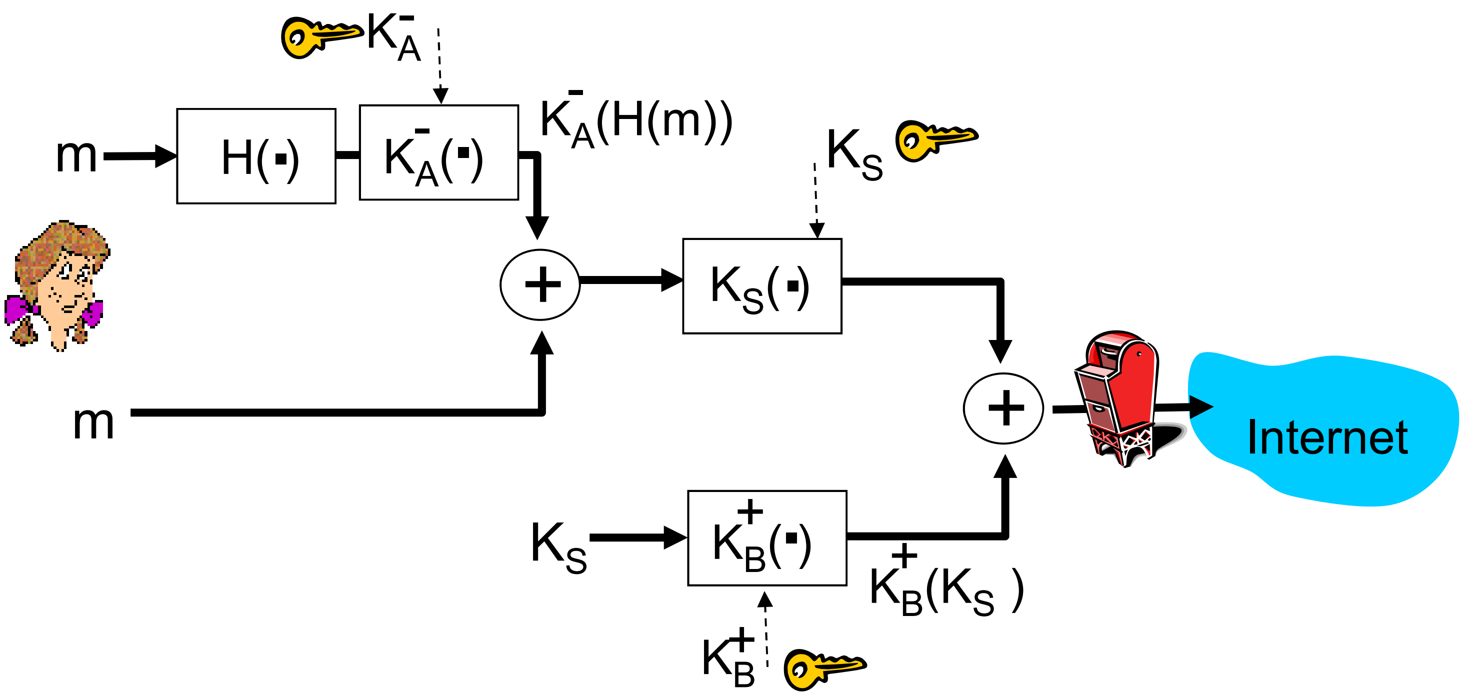
- Alice wants to provide message integrity and sender authentication



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

Securing Email - CIA

- Alice wants to provide confidentiality (secrecy), sender authentication, and message integrity



- Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Network Security (Summary)

The following questions from Chapter 8 are representative of those you may see in the future:

Review questions: 1, 2, 3, 9, 10, 11, 12, 13, 16, 17

Surveys

- Please take a few moments to complete course & instructor surveys

Within AFIT : https://cf.afit.edu/OES/
External : https://www.afit.edu/en/OES/