SECURITY IN NETWORKS

https://www.youtube.com/watch?v=RfAdux3XidM

GOALS

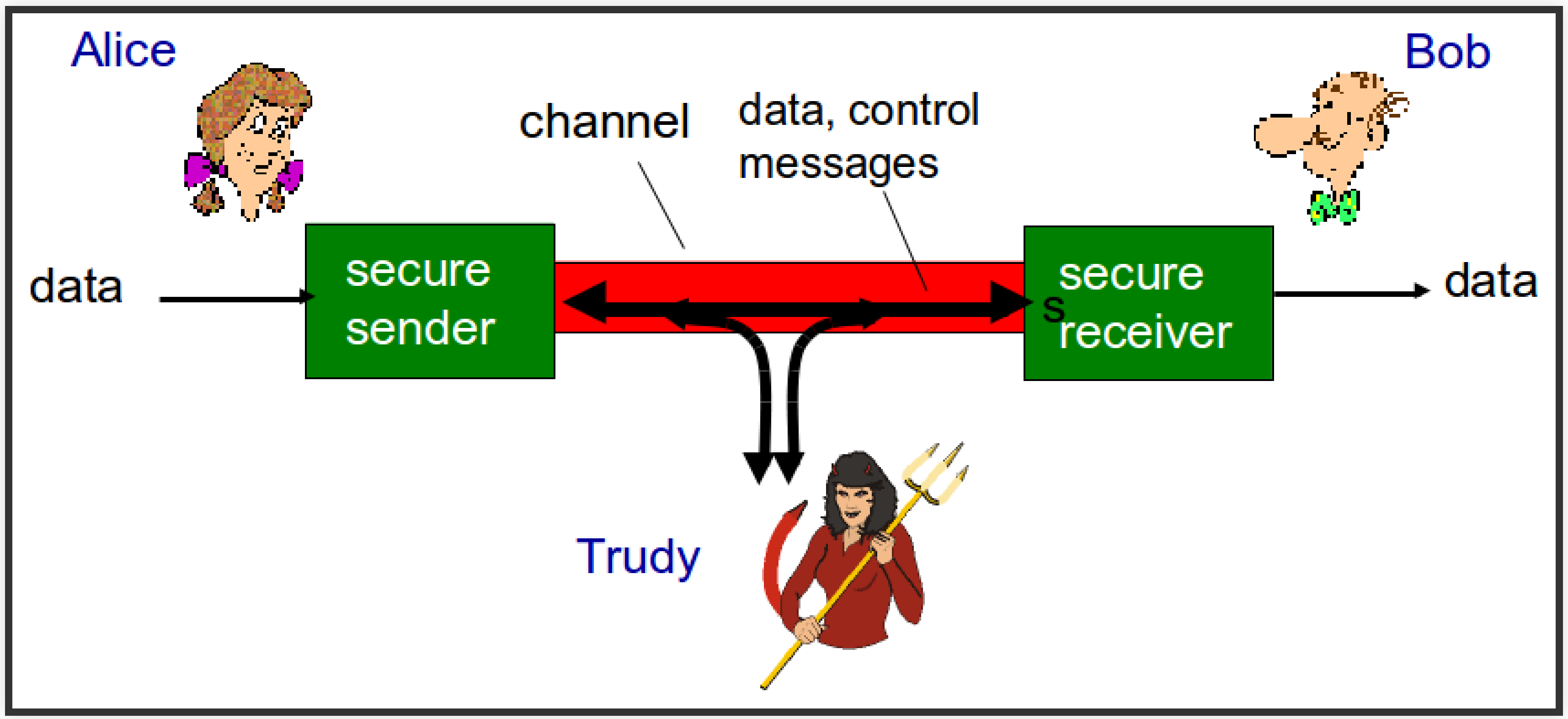
- Understand principles of network security:
  - Cryptography and its many uses beyond "confidentiality"
  - Authentication
  - Message integrity
- Securing Email - PGP

WHAT IS NETWORK SECURITY?

- **Confidentiality:** only sender, intended receiver should "understand" message contents
  - sender encrypts message
  - receiver decrypts message
- **Authentication:** sender, receiver want to confirm identity of each other
- **Message integrity:** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- **Access and availability:** services must be accessible and available to users

FRIENDS AND ENEMIES: ALICE, BOB, TRUDY

- Well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely” (now you could use Angela and Barrack)
- Trudy (intruder) may intercept, delete, add messages

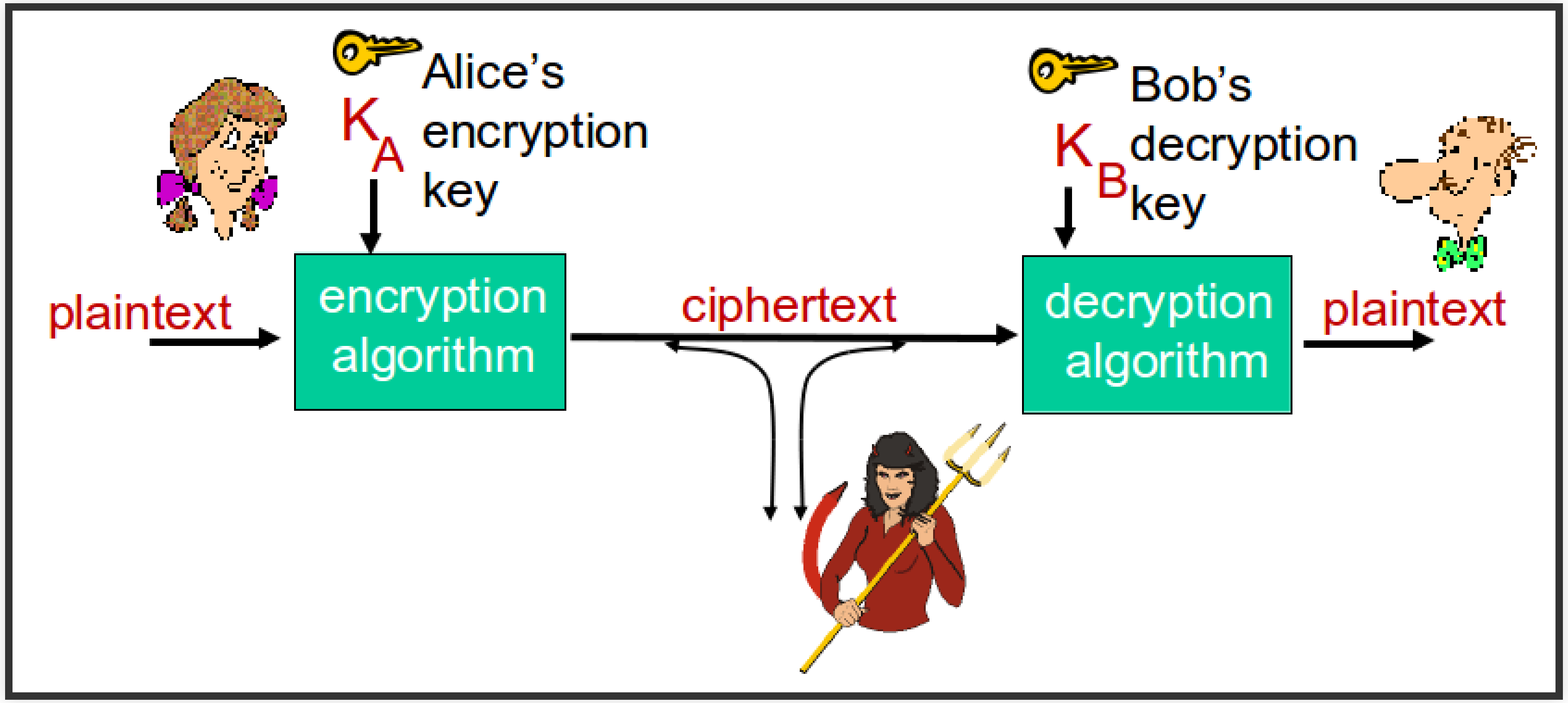


WHO MIGHT BOB, ALICE BE?

- … well, real-life Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

PRINCIPLES OF CRYPTOGRAPHY

THE LANGUAGE OF CRYPTOGRAPHY



- **m** plaintext message
- $\mathbf{K_A(m)}$ ciphertext, encrypted with key $\mathbf{K_A}$
- $\mathbf{m = K_B(K_A(m))}$

SIMPLE ENCRYPTION SCHEME

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz
ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

e.g:

```
Plaintext:  bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

> Encryption key:
>
> Mapping from set of 26 letters to set of 26 letters

BREAKING AN ENCRYPTION SCHEME

CIPHER-TEXT ONLY ATTACK

Trudy has ciphertext she can analyze

Two approaches:

- brute force: search through all keys
- statistical analysis

KNOWN-PLAINTEXT ATTACK

❗ Trudy has plaintext corresponding to ciphertext

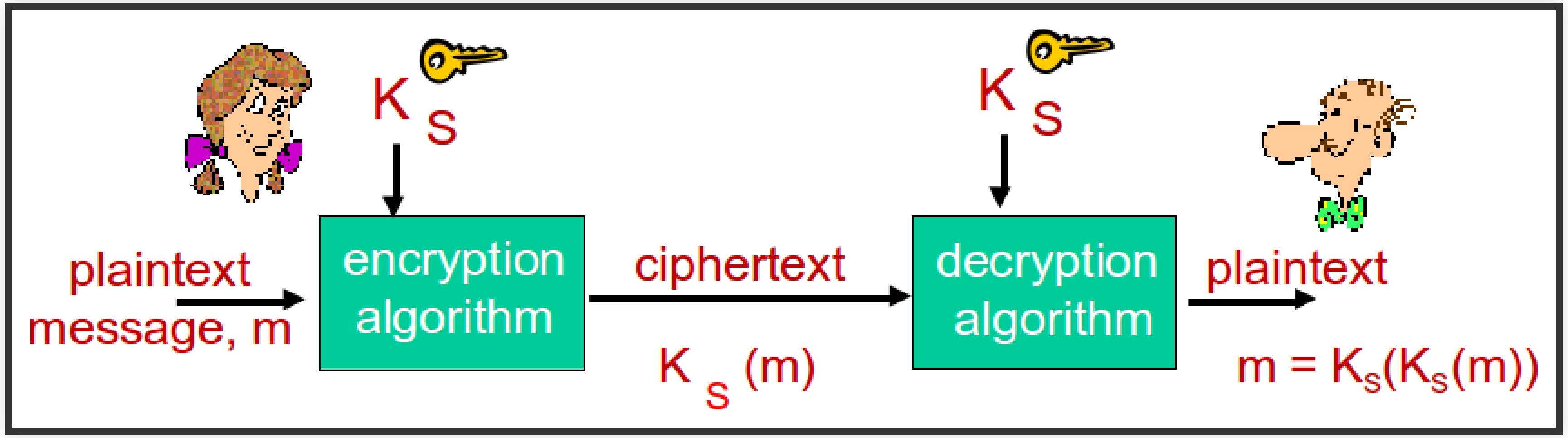
- e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,

CHOSEN-PLAINTEXT ATTACK

Trudy can get ciphertext for chosen plaintext

SYMMETRIC KEY CRYPTOGRAPHY

SYMMETRIC KEY CRYPTOGRAPHY



symmetric key crypto: Bob and Alice share same (symmetric) key: K

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

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

A MORE SOPHISTICATED APPROACH

- n substitution ciphers, M_1,M_2,…,M_n
- cycling pattern:
  - e.g., n=4: M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; …
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

> Encryption key:
>
> n substitution ciphers, and cyclic pattern
>
> - key need not be just n-bit pattern

SYMMETRIC KEY CRYPTO: DES

DES: Data Encryption Standard

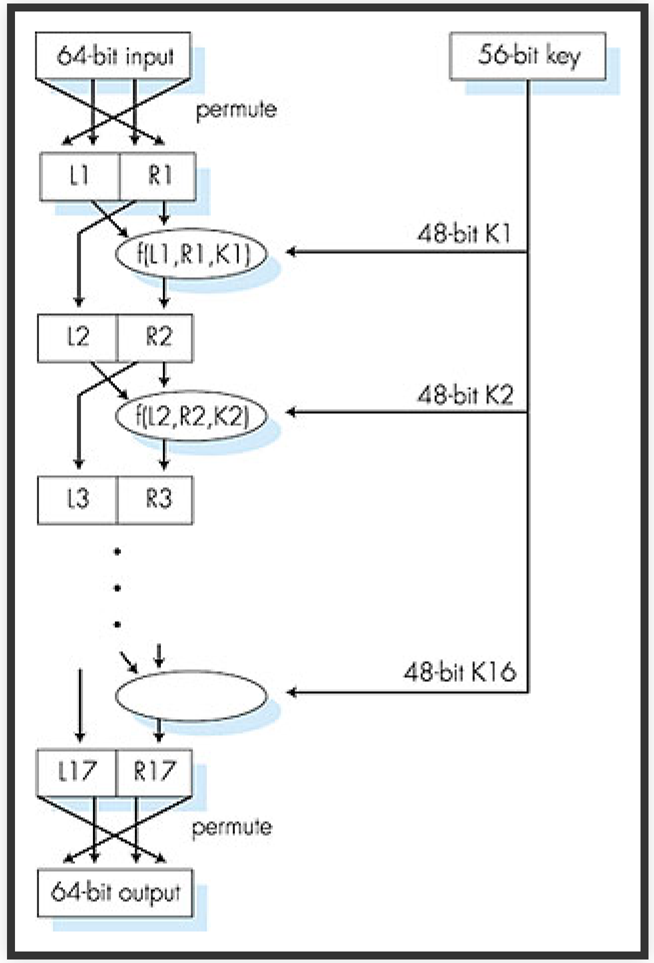
- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

SYMMETRIC KEY CRYPTO: DES

DES operation

- initial permutation
- 16 identical “rounds” of function application, each using different 48 bits of key
- final permutation

SYMMETRIC KEY CRYPTO: DES



AES: ADVANCED ENCRYPTION STANDARD

- Symmetric-key NIST standard, replaced DES (Nov 2001)
- Processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- Iterated cipher
- Brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

AES: ADVANCED ENCRYPTION STANDARD

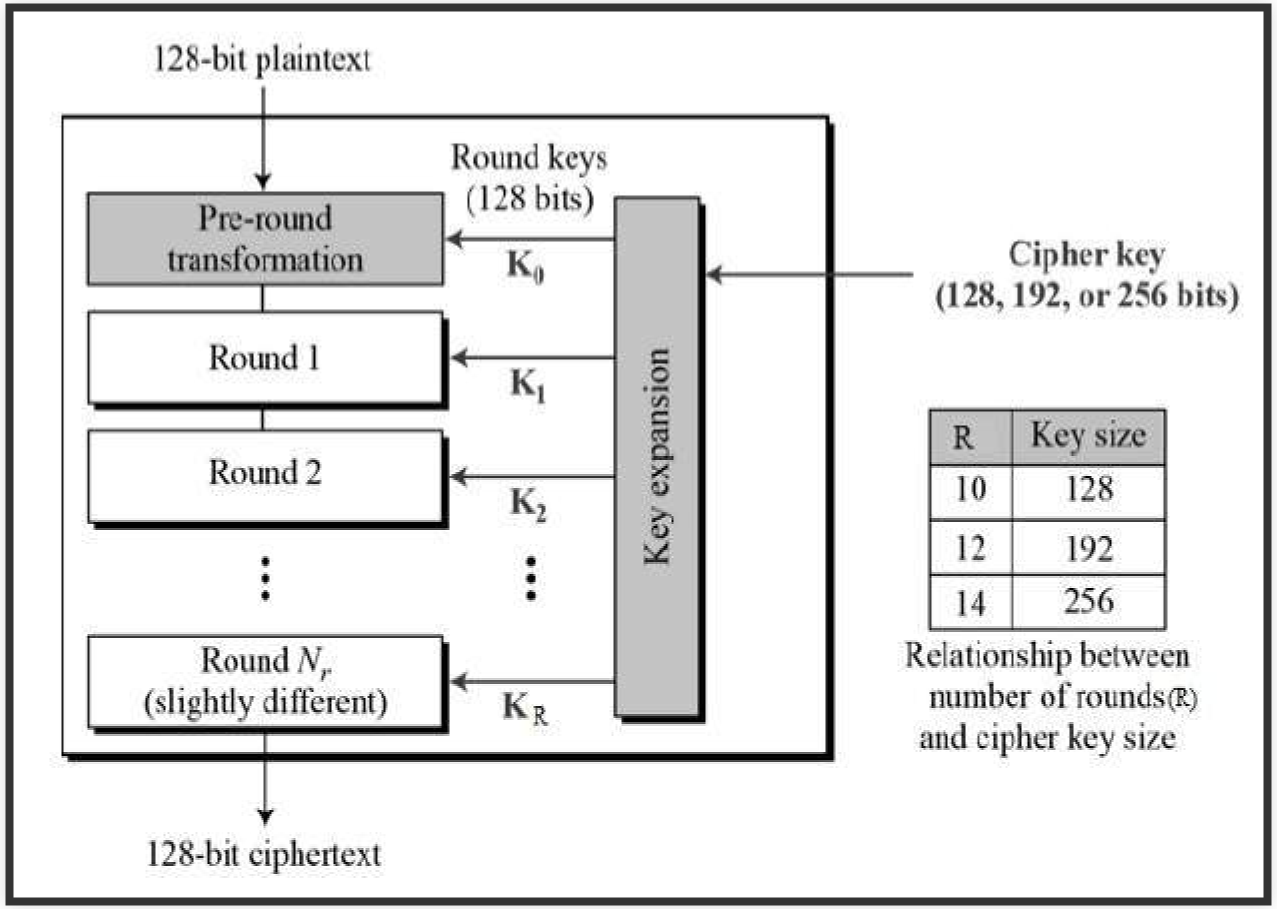
- Number of rounds depend on key length
  - 128 bits → 10 rounds
  - 192 bits → 12 rounds
  - 256 bits → 14 rounds

AES: ADVANCED ENCRYPTION STANDARD

Each round:

1. Round key mixing
2. Substitution step
3. Permutation step

AES: ADVANCED ENCRYPTION STANDARD



AES DEMO

```
echo "This is the very secret message" > cleartext.txt
# openssl aes-256-cbc -in cleartext.txt -a
# -a -> Base64 encrypts so easy to copy to email
openssl aes-256-cbc -in cleartext.txt -out ciphertext.txt
cat ciphertext.txt #Verify content
openssl aes-256-cbc -d -in ciphertext.txt
```

PUBLIC KEY CRYPTOGRAPHY

PUBLIC KEY CRYPTOGRAPHY

Symmetric key crypto

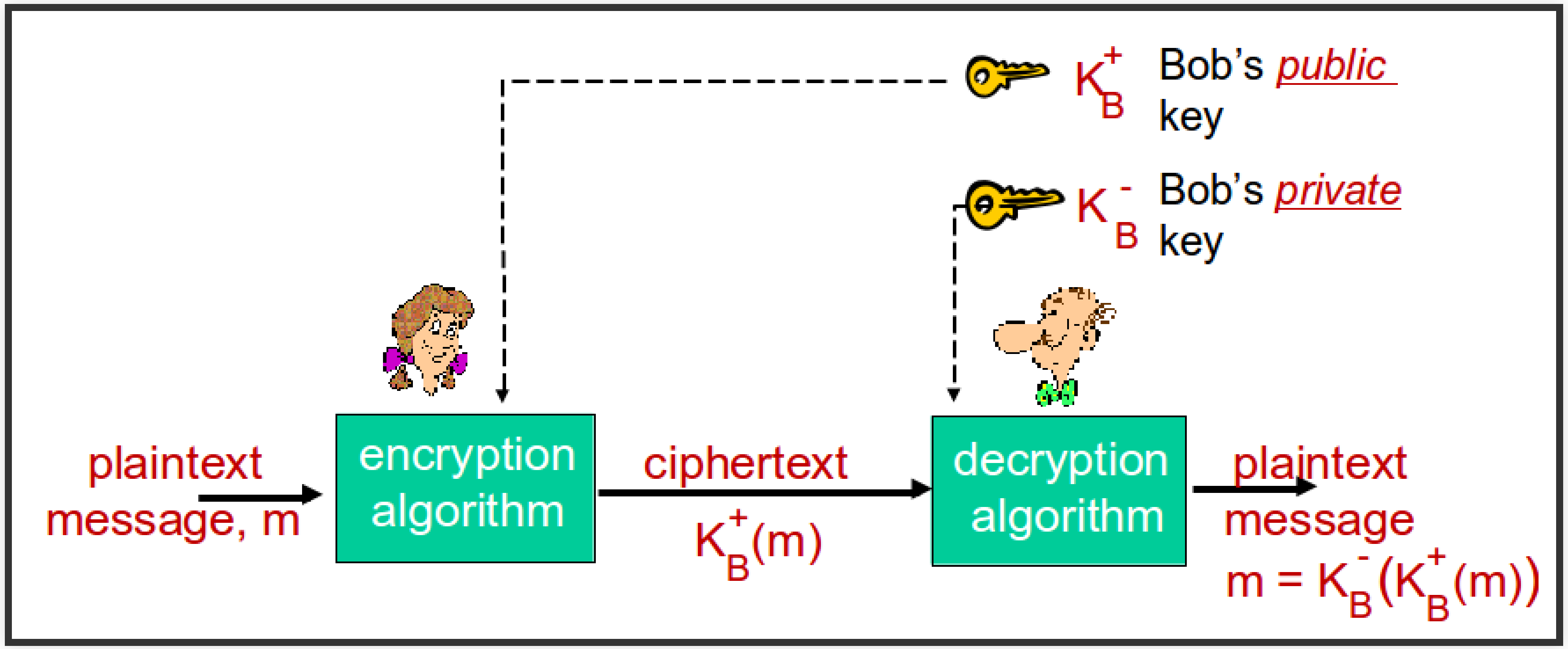
- requires sender, receiver know shared secret key
- **Q:** how to agree on key in first place (particularly if never "met")?

PUBLIC KEY CRYPTOGRAPHY

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

PUBLIC KEY CRYPTOGRAPHY



PUBLIC KEY ENCRYPTION ALGORITHMS

Need

- $K^{+}_{B}(\cdot)$ and $K^{-}_{B}(\cdot)$

such that

- $K^{-}_{B}(K^{+}_{B}(m)) = m$
- given public key K^{+}_{B}, it should be impossible to compute private key K^{-}_{B}

RSA: Rivest, Shamir, AdelMan algorithm

PREREQUISITE: MODULAR ARITHMETIC

PREREQUISITE: MODULAR ARITHMETIC

x mod n = remainder of **x** when divided 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:

- **$(a \text{ mod } n)^d$ mod n = a^d mod n**

EXAMPLE

x=14, n=10, d=2:

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

RSA

RSA: GETTING READY

- message: just a bit pattern
- 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

- choose two large prime numbers p, q. (e.g. at least, 2048 bits each)
- compute **n = pq**, **z = (p-1)(q-1)**
- choose **e** (with **e<n**) that has no common factors with **z** (e, z are "relatively prime").
- choose **d** such that **ed-1** is exactly divisible by **z**. (in other words: **ed mod z = 1**).
- public key is **(n,e)**.
- private key is **(n,d)**.

RSA: ENCRYPTION, DECRYPTION

- Given **(n,e)** and **(n,d)** as computed above
- to encrypt message **m (<n)**, compute $\mathbf{c = m^e \bmod n}$
- to decrypt received bit pattern, **c**, compute $\mathbf{m = c^d \bmod n}$

❗ magic happens!

$$m = (\underbrace{m^e \mod n}_{c})^d \mod n$$

RSA EXAMPLE:

- Bob chooses **p=5, q=7**. Then **n=35, z=24**.
- **e=5** (so **e, z** relatively prime).
- **d=29** (so **e · d - 1** exactly divisible by **z**).
- encrypting 4-bit messages.

RSA EXAMPLE:

Encrypt

$$\underbrace{bit\ pattern}_{1100} \quad \underbrace{m}_{12} \quad \underbrace{m^e}_{24832} \quad \underbrace{c = m^e \mod n}_{17}$$

RSA EXAMPLE:

Decrypt

$$\underbrace{c}_{17} \quad \underbrace{c^d}_{4819685721067509150914118252230716970} \quad \underbrace{m = c^d \mod n}_{12}$$

WHY DOES RSA WORK?

- 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 $n = p \cdot q$ and $z = (p-1)(q-1)$

thus,

- $c^d \bmod n = (m^e \bmod n)^d \bmod n$

RSA: ANOTHER IMPORTANT PROPERTY

The following property will be very useful later:

$$K^{-}_{B}(K^{+}_{B}(m))$$

$$= m =$$

$$K^{+}_{B}(K^{-}_{B}(m))$$

Using public or private key first: **Result is the same**

Why?

Follows directly from modular arithmetic:

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

WHY IS RSA SECURE?

- 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

💡 Factoring a big number is hard - no known efficient algorithm

RSA IN PRACTICE: SESSION KEYS

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key cryto 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 key K_S
- once both have K_S, they use symmetric key cryptography

MESSAGE INTEGRITY, AUTHENTICATION

MESSAGE INTEGRITY, AUTHENTICATION

> ❗ When Bob receives a message (plain or ciphertext) and he believe it was sent by Alice

To authenticate this message, Bob must verify:

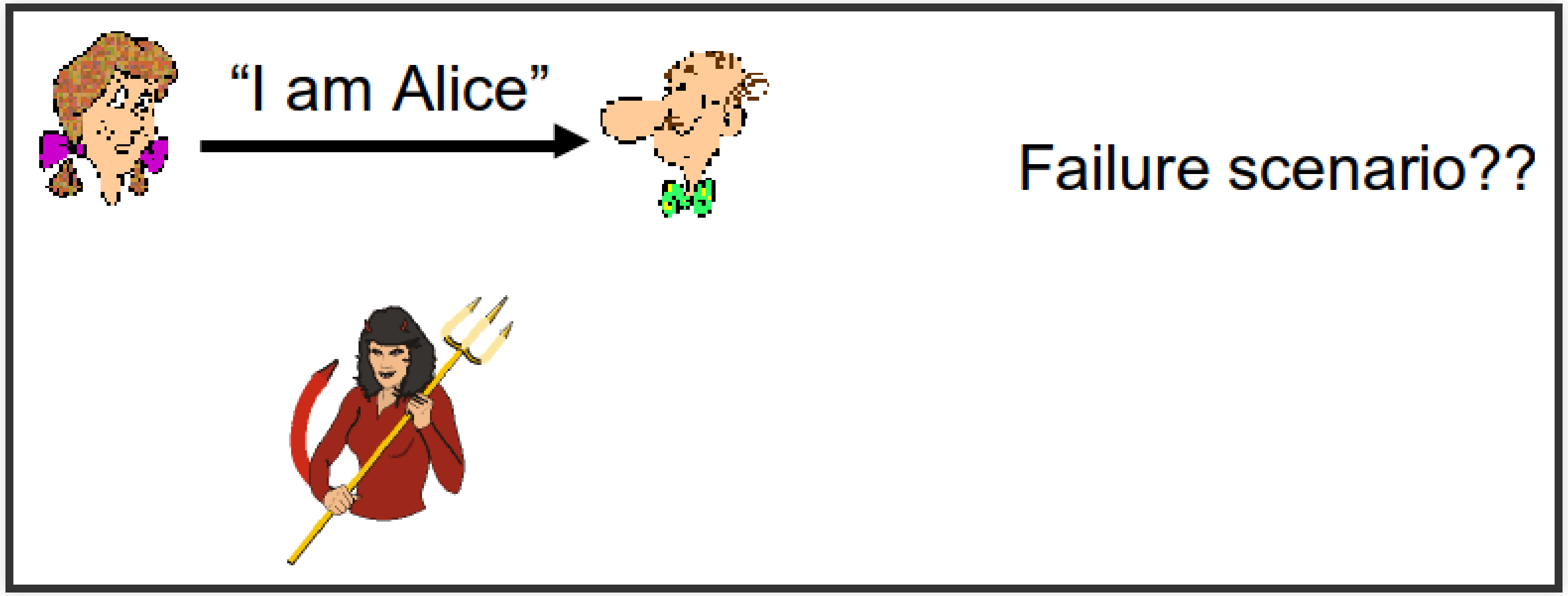
- The message indeed originated from Alice
- The message was not tampered with on its way to Bob

AUTHENTICATION

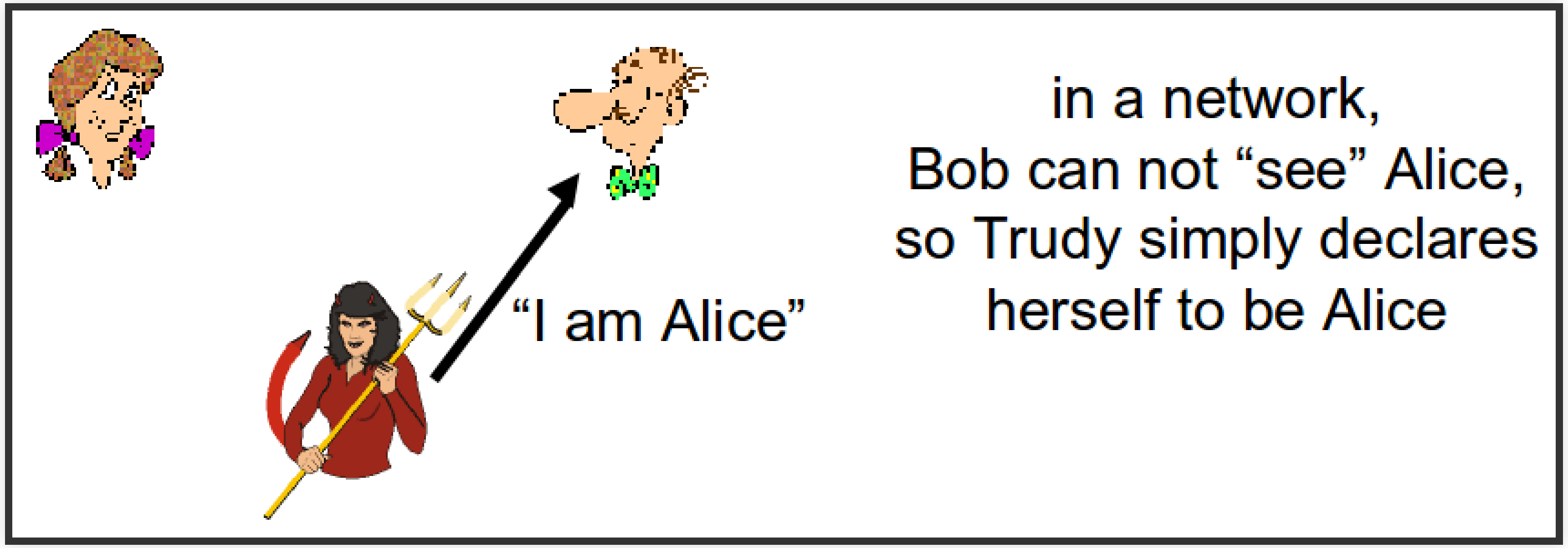
Goal: Bob wants Alice to “prove” her identity to him

PROTOCOL 1.0

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

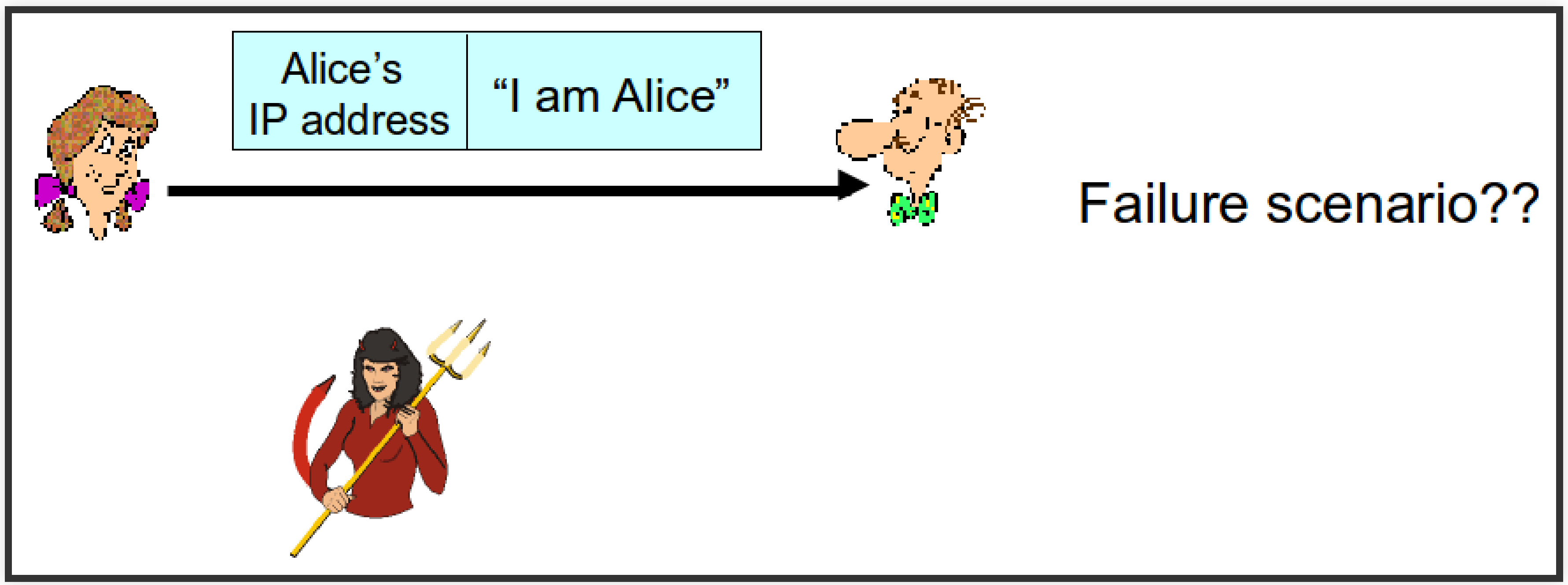


PROTOCOL 1.0

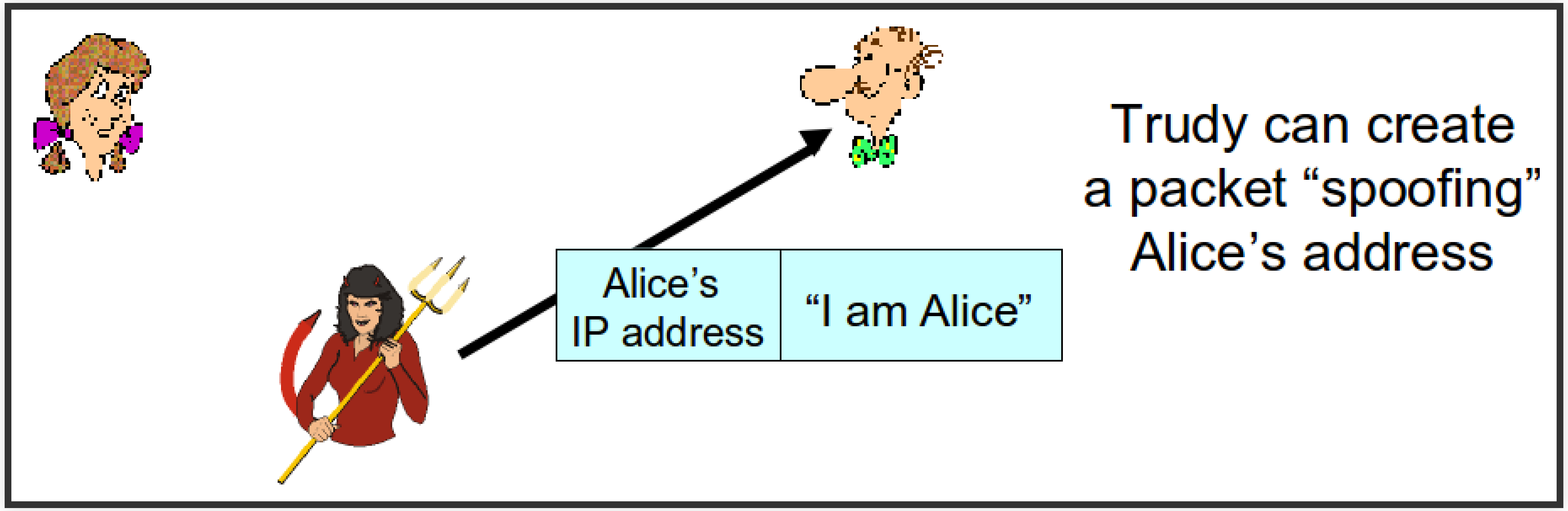

"I am Alice"
in a network,
Bob can not "see" Alice,
so Trudy simply declares
herself to be Alice

PROTOCOL 2.0

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

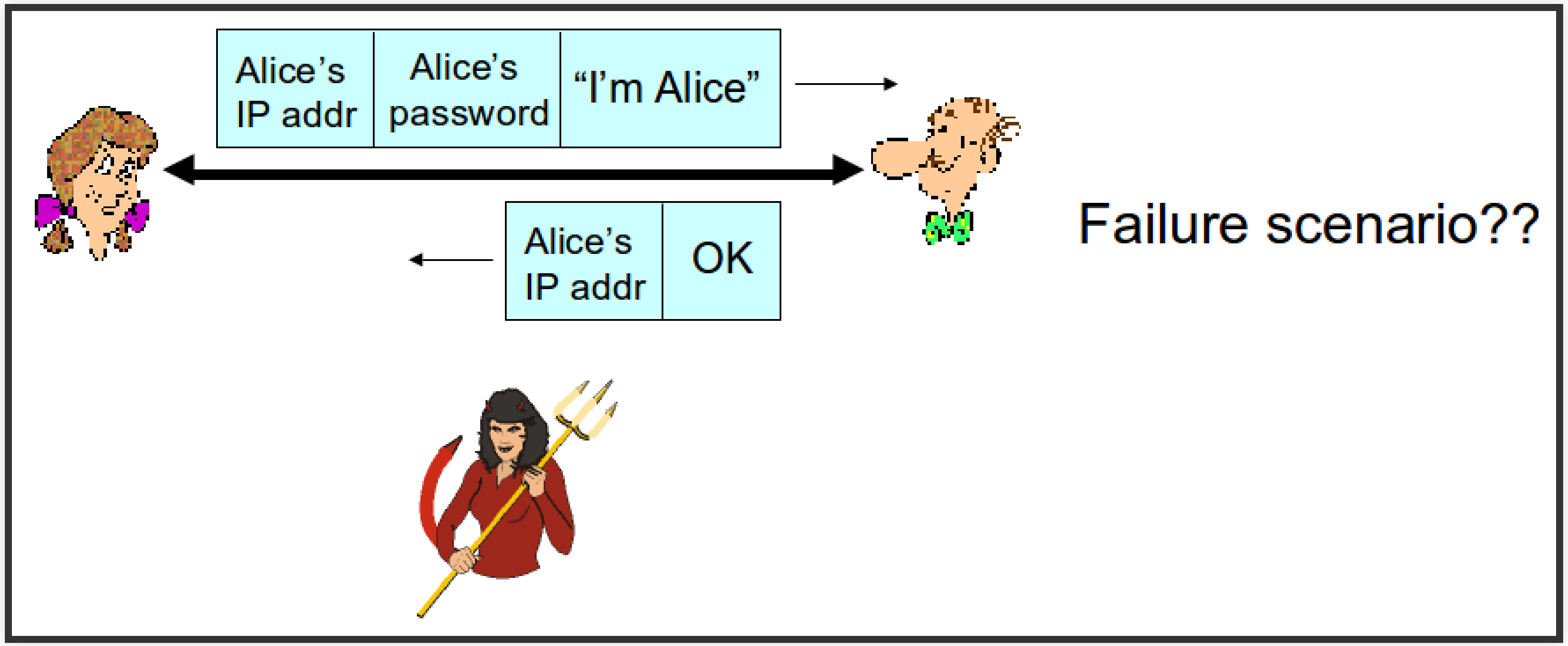


PROTOCOL 2.0

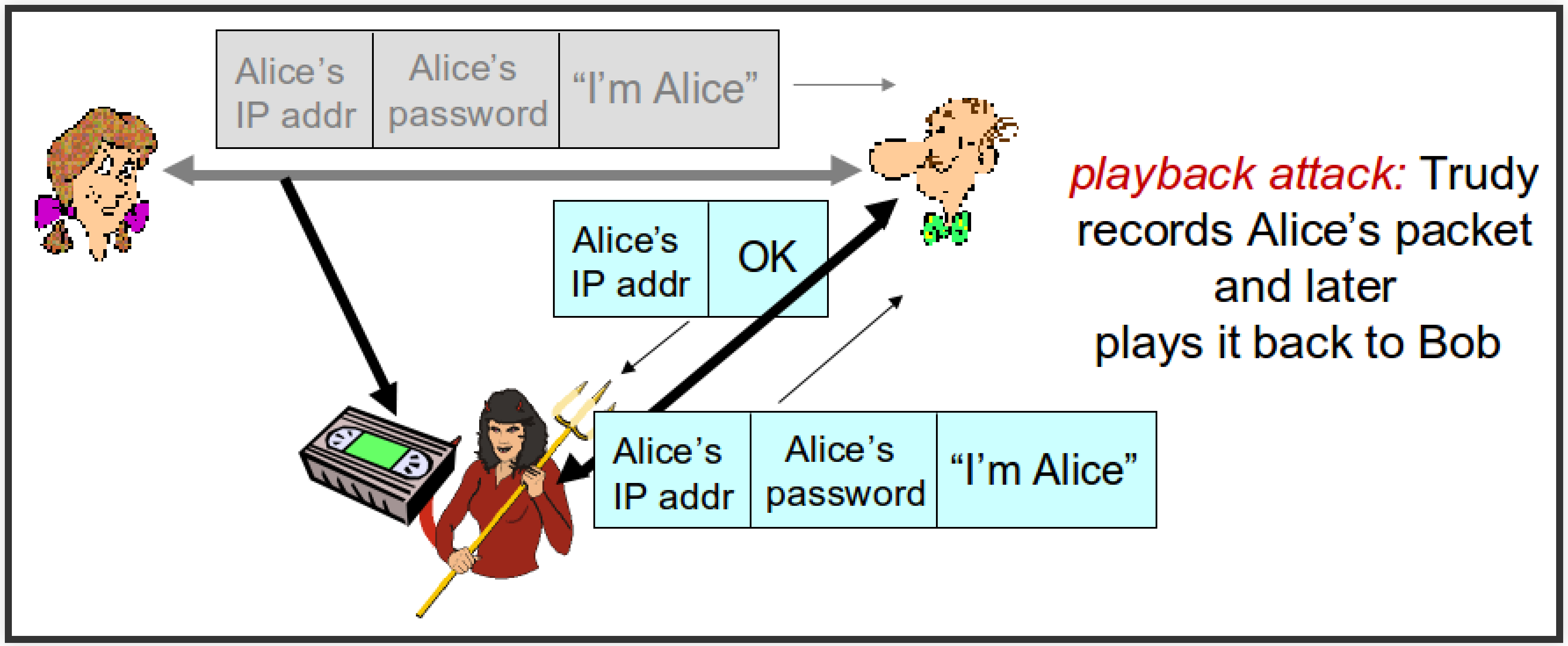

Alice's
IP address
"I am Alice"
Trudy can create
a packet "spoofing"
Alice's address

PROTOCOL 3.0

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

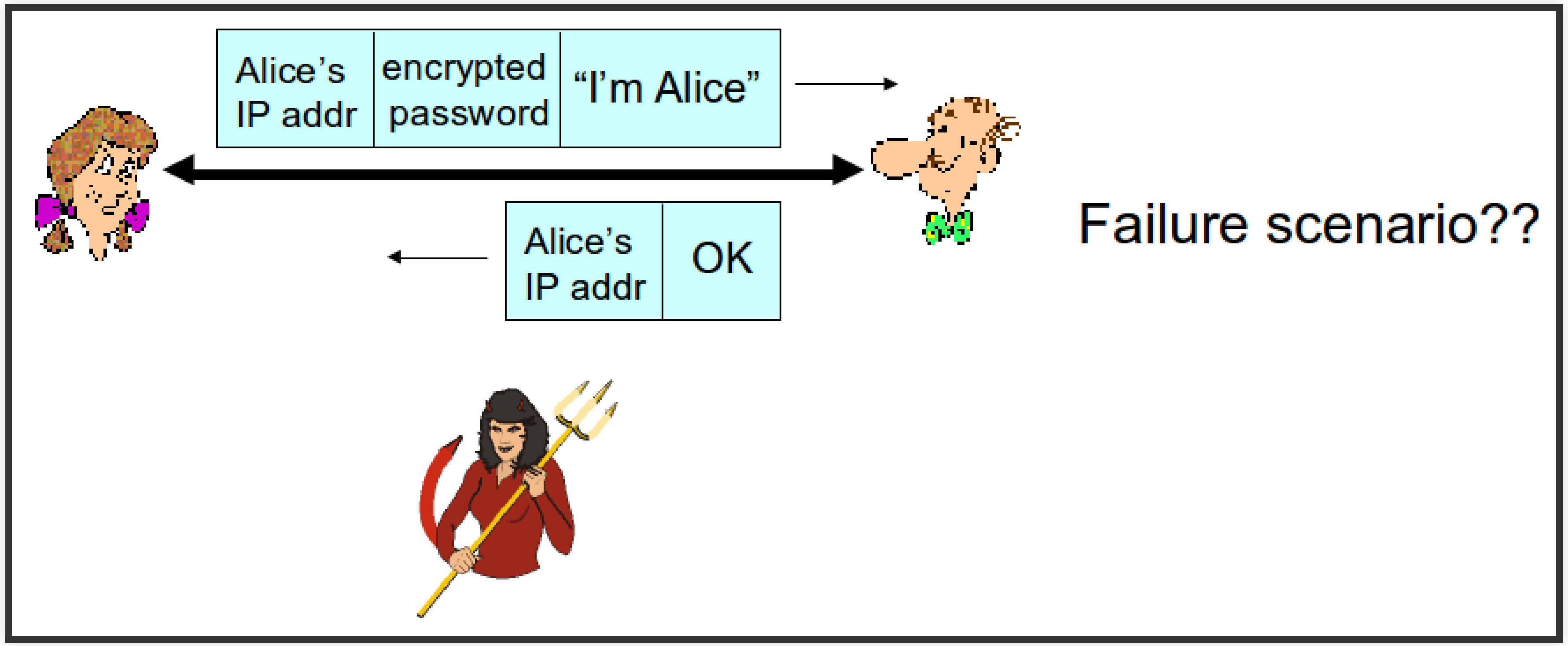


PROTOCOL 3.0

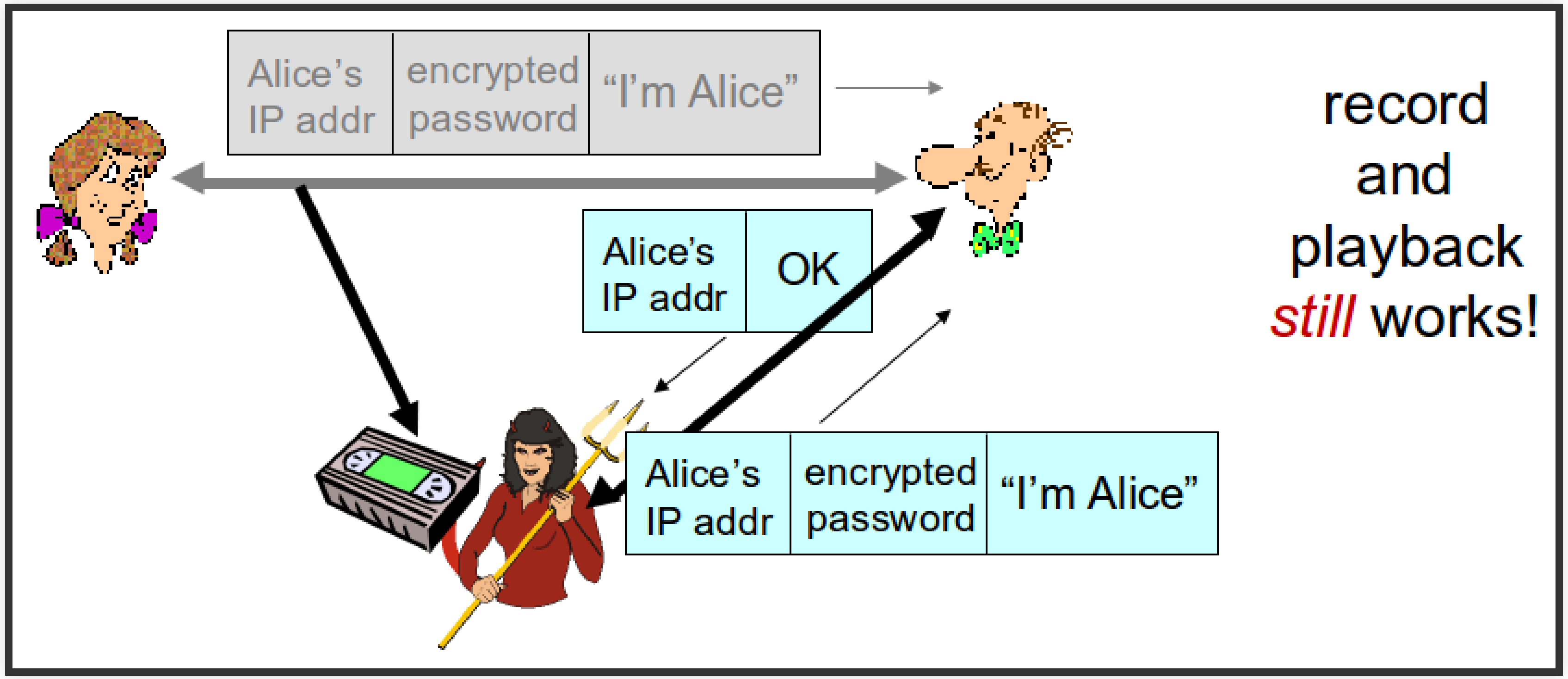

Alice's IP addr
Alice's password
"I'm Alice"
Alice's IP addr
OK
Alice's IP addr
Alice's password
"I'm Alice"
playback attack: Trudy records Alice's packet and later plays it back to Bob

PROTOCOL 3.1

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



PROTOCOL 3.1



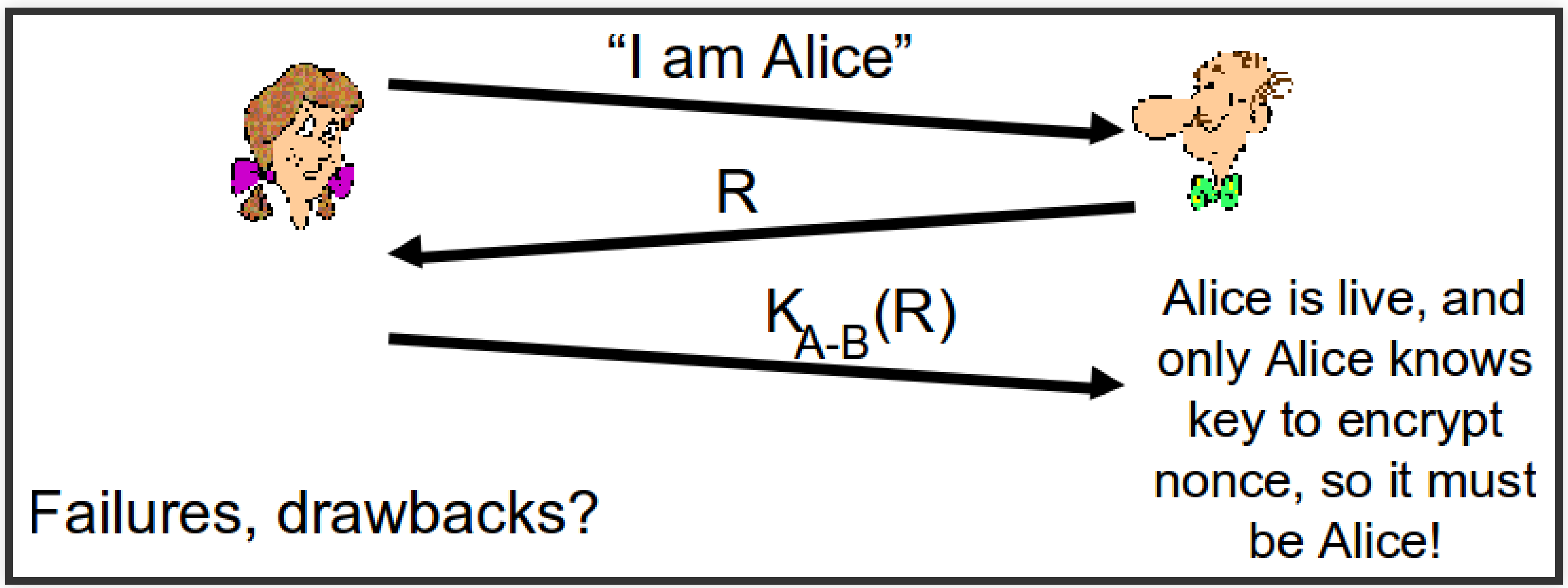
Alice's IP addr
encrypted password
"I'm Alice"
Alice's IP addr
OK
Alice's IP addr
encrypted password
"I'm Alice"
record and playback *still* works!

PROTOCOL 4.0

Goal: avoid playback attack

Nonce number **R** used only **once-in-a-lifetime**

Protocol ap4.0: To prove Alice “live”, Bob sends Alice **nonce**, R. Alice must return R, encrypted with shared secret key



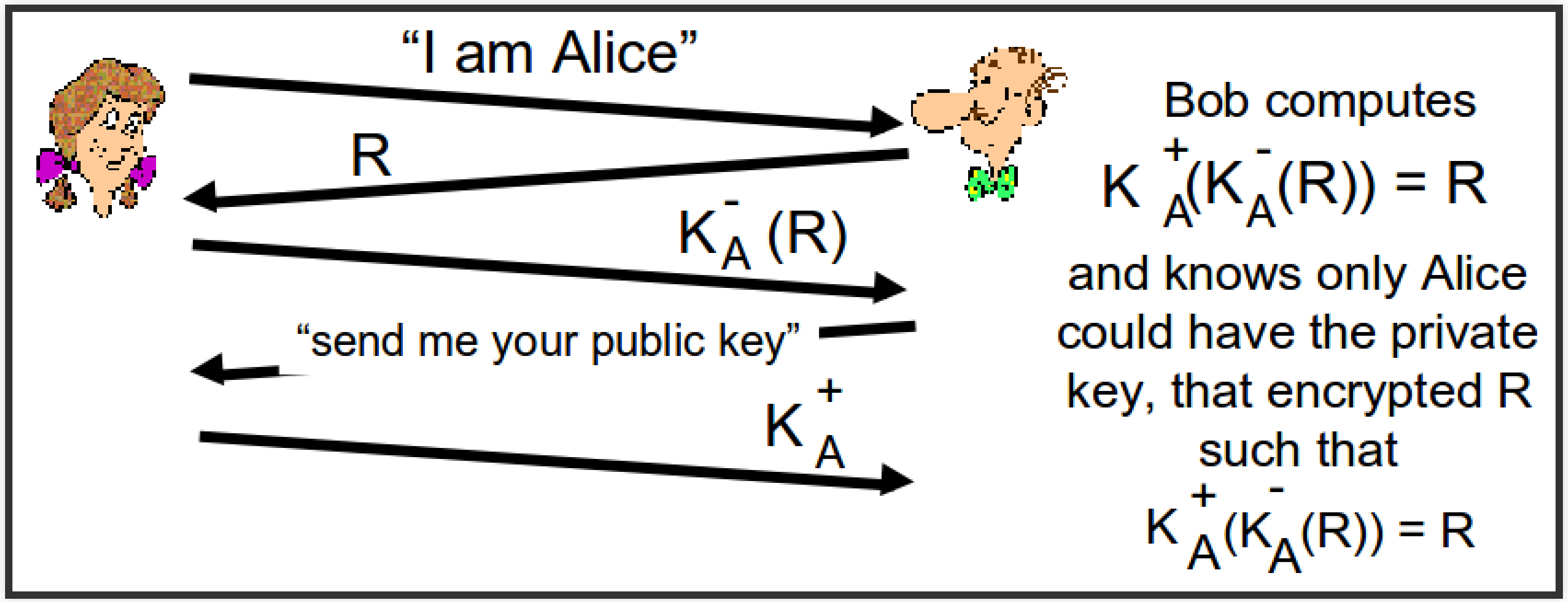
PROTOCOL 4.0

ap4.0 requires shared symmetric key

- Can we authenticate using public key techniques?

PROTOCOL 5.0

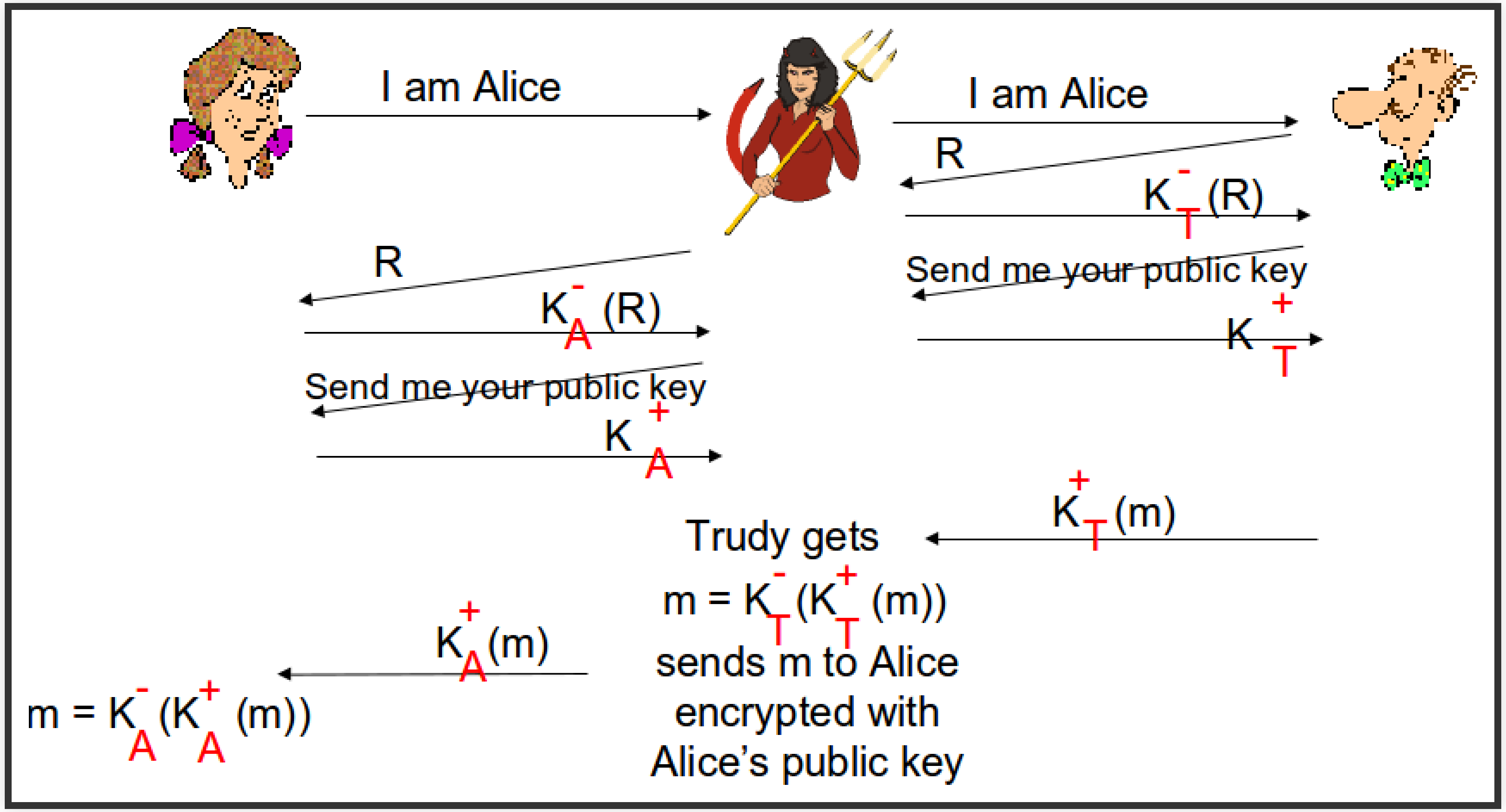
Protocol ap5.0: Use nonce, public key cryptography



PROTOCOL 5.0: SECURITY HOLE

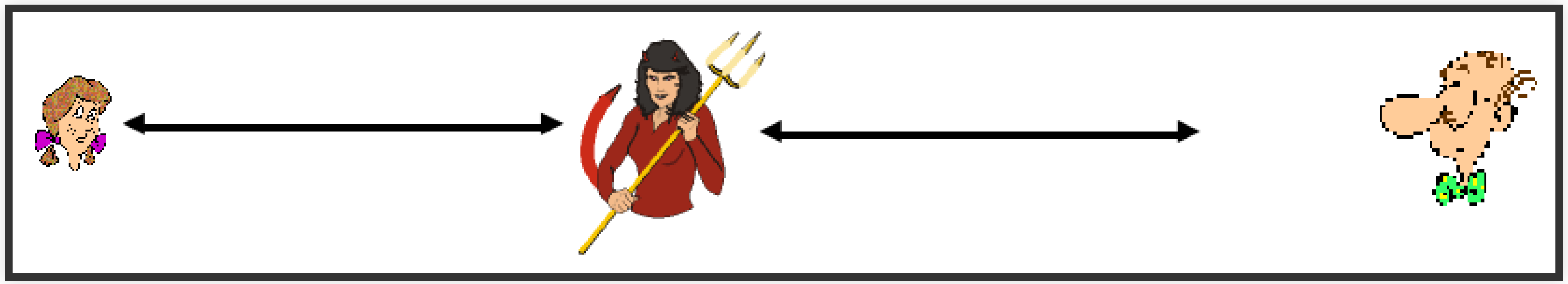
Man (or woman) in the middle attack:

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!

DIGITAL SIGNATURES

DIGITAL SIGNATURES

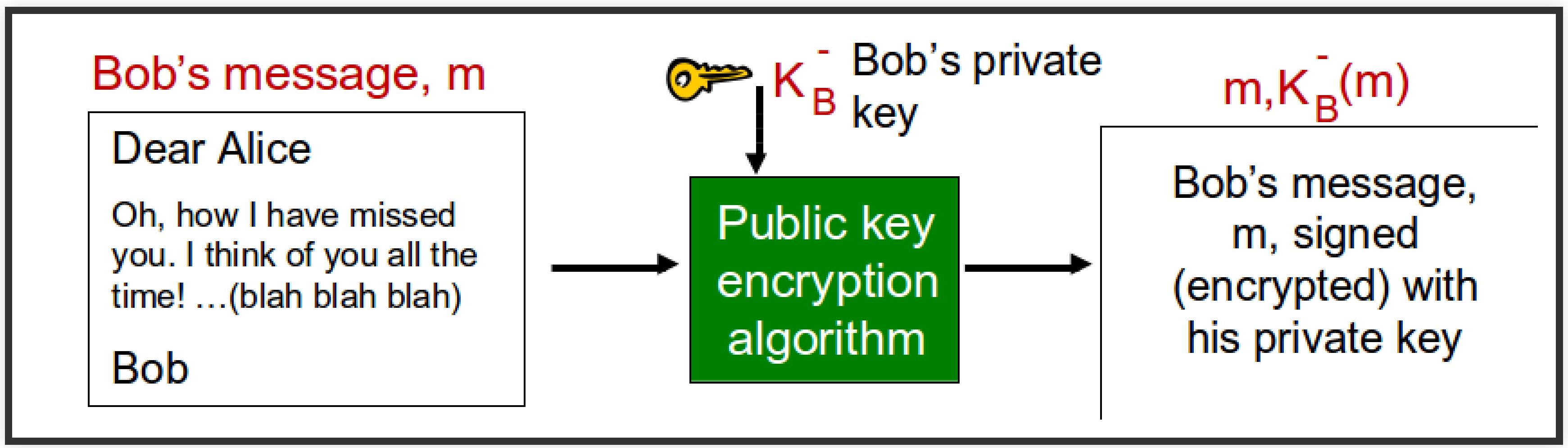
Cryptographic technique analogous to hand-written signatures:

- 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 $\mathbf{K_B^-}$, creating "signed" message, $\mathbf{K_B^-(m)}$



DIGITAL SIGNATURES

- suppose Alice receives msg **m**, with signature: **m, $K_B^-(m)$**
- 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.

DIGITAL SIGNATURES

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 $\mathbf{K_B^-(m)}$ to court and prove that Bob signed **m**

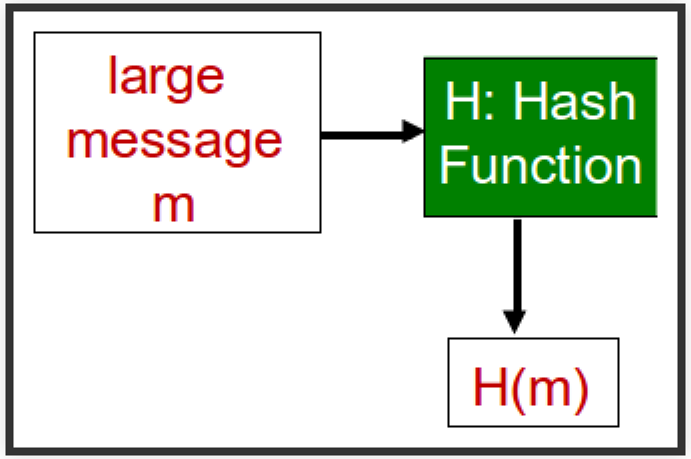
MESSAGE DIGESTS

MESSAGE DIGESTS

Computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy- to-compute digital “fingerprint”

- Apply hash function **H** to **m**, get fixed size message digest, **H(m)**.

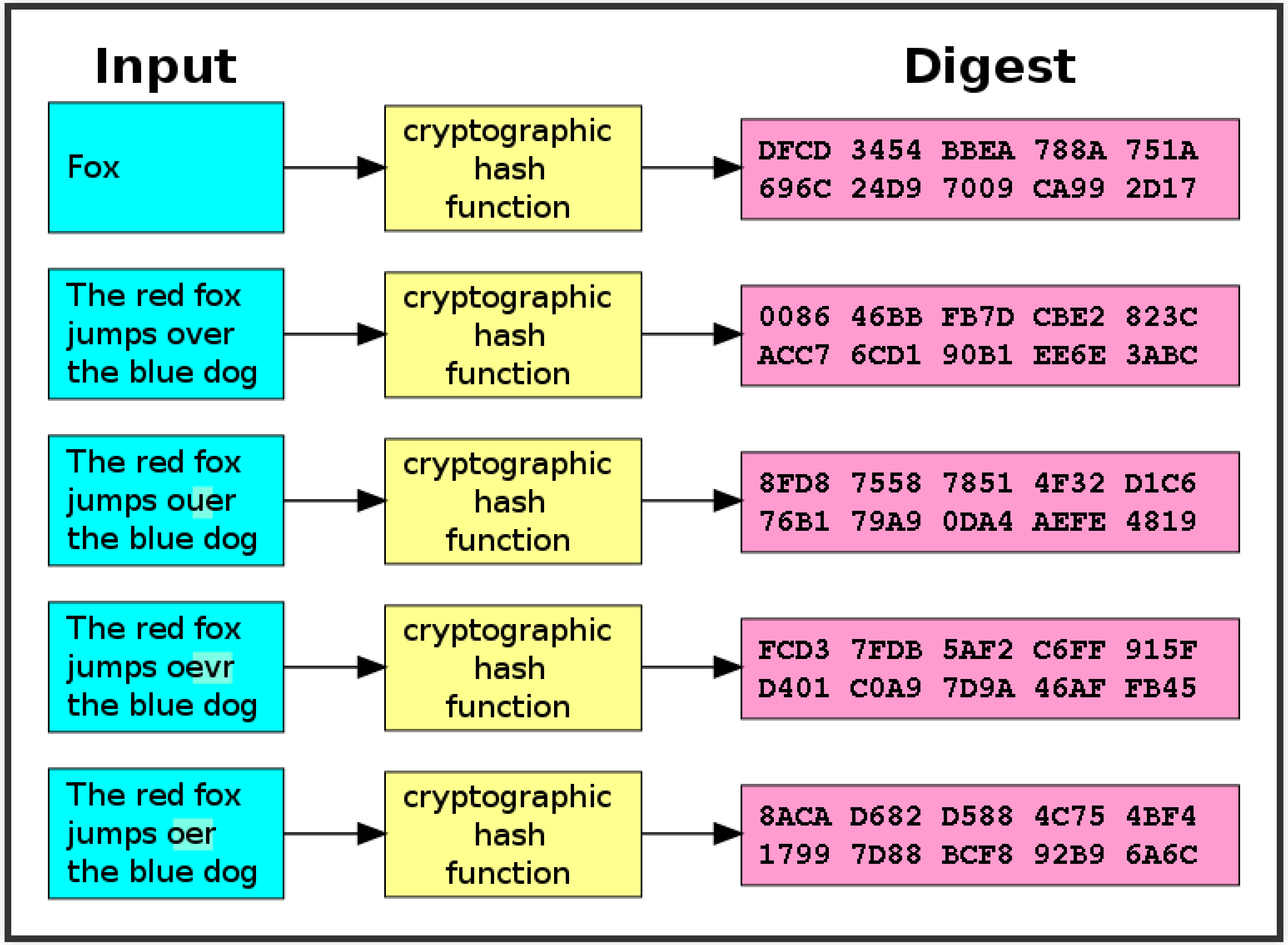


CRYPTOGRAPHIC HASH FUNCTION

Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- computationally infeasible to find messages **x** and **y** such that **H(x) = H(y)**

CRYPTOGRAPHIC HASH FUNCTION



INTERNET CHECKSUM: POOR CRYPTO HASH

Internet checksum has some properties of hash function:

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

INTERNET CHECKSUM: POOR CRYPTO HASH

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

Different messages, but identical checksums

CRC is also a poor crypto hash function

HASH FUNCTION ALGORITHMS

MD5 hash function widely used (RFC 1321)

- computes 128-bit message digest in 4-step process.
- arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
- **It is no longer secure!**
  - Rainbow tables available

HASH FUNCTION ALGORITHMS

SHA-1 is also used

- US standard [NIST, FIPS PUB 180-1]
- 160-bit message digest
- Shown that you can find collisions in 'only' $\mathbf{2^{51}}$ attempts. See http://eprint.iacr.org/2008/469.pdf originally expected $\mathbf{2^{80}}$

HASH FUNCTION ALGORITHMS

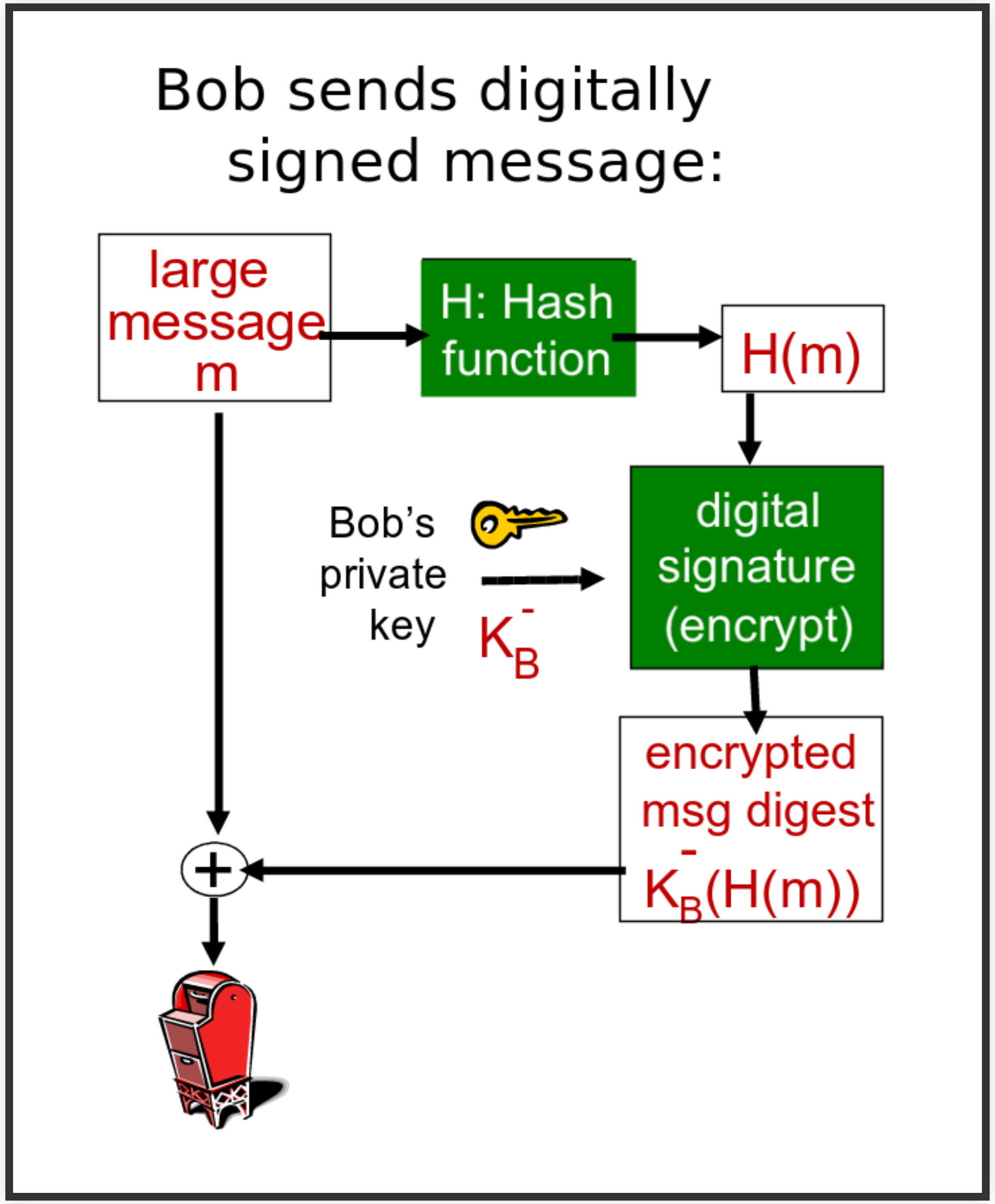
Keccak wins SHA3 competition in 2012

- 5 year competition by NIST for the next cryptographic hash function standard
- Keccak announced the winner in October 2012: http://csrc.nist.gov/groups/ST/hash/sha-3/winner_sha-3.html

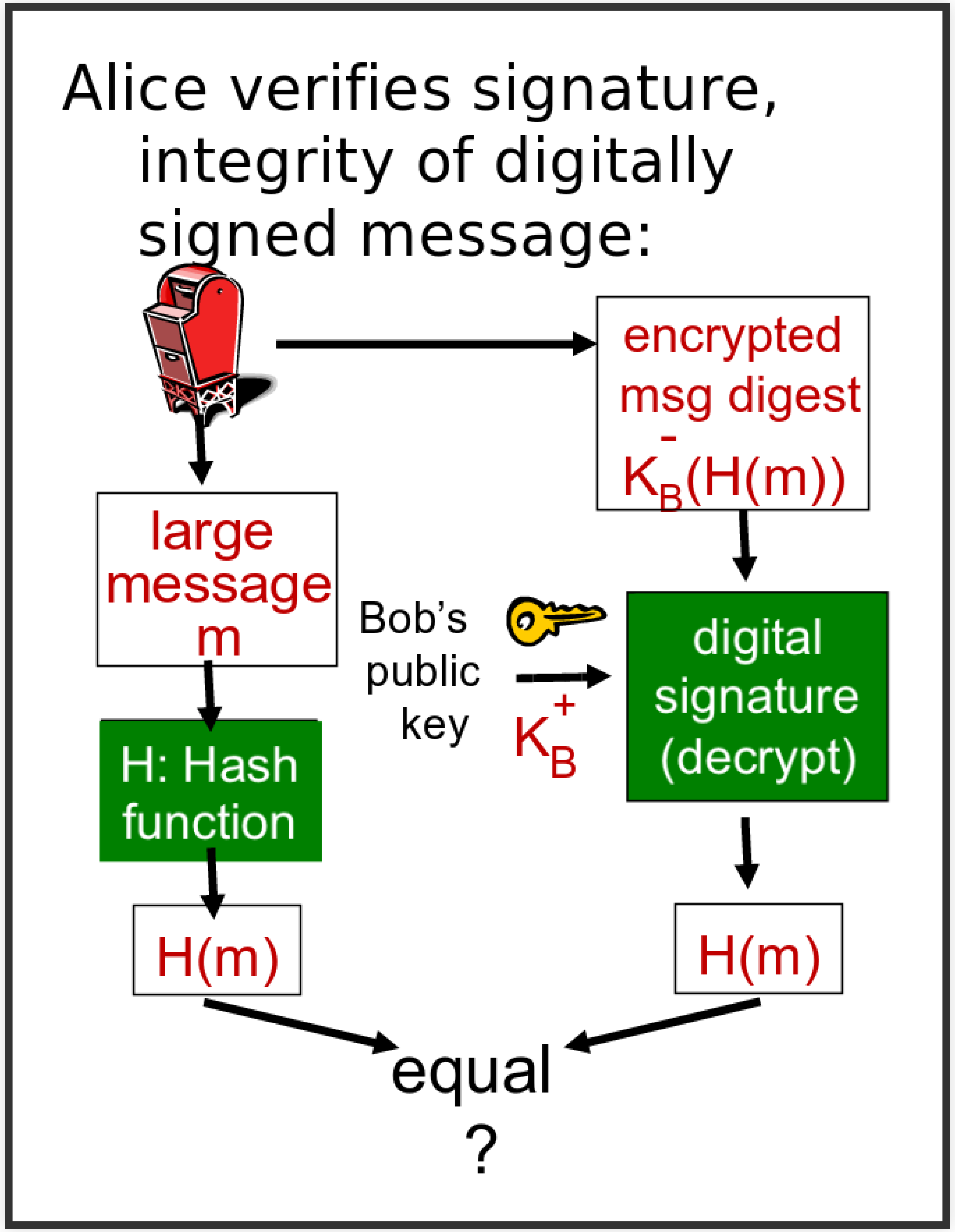
DIGITAL SIGNATURE

Digital signature = signed message digest

DIGITAL SIGNATURE



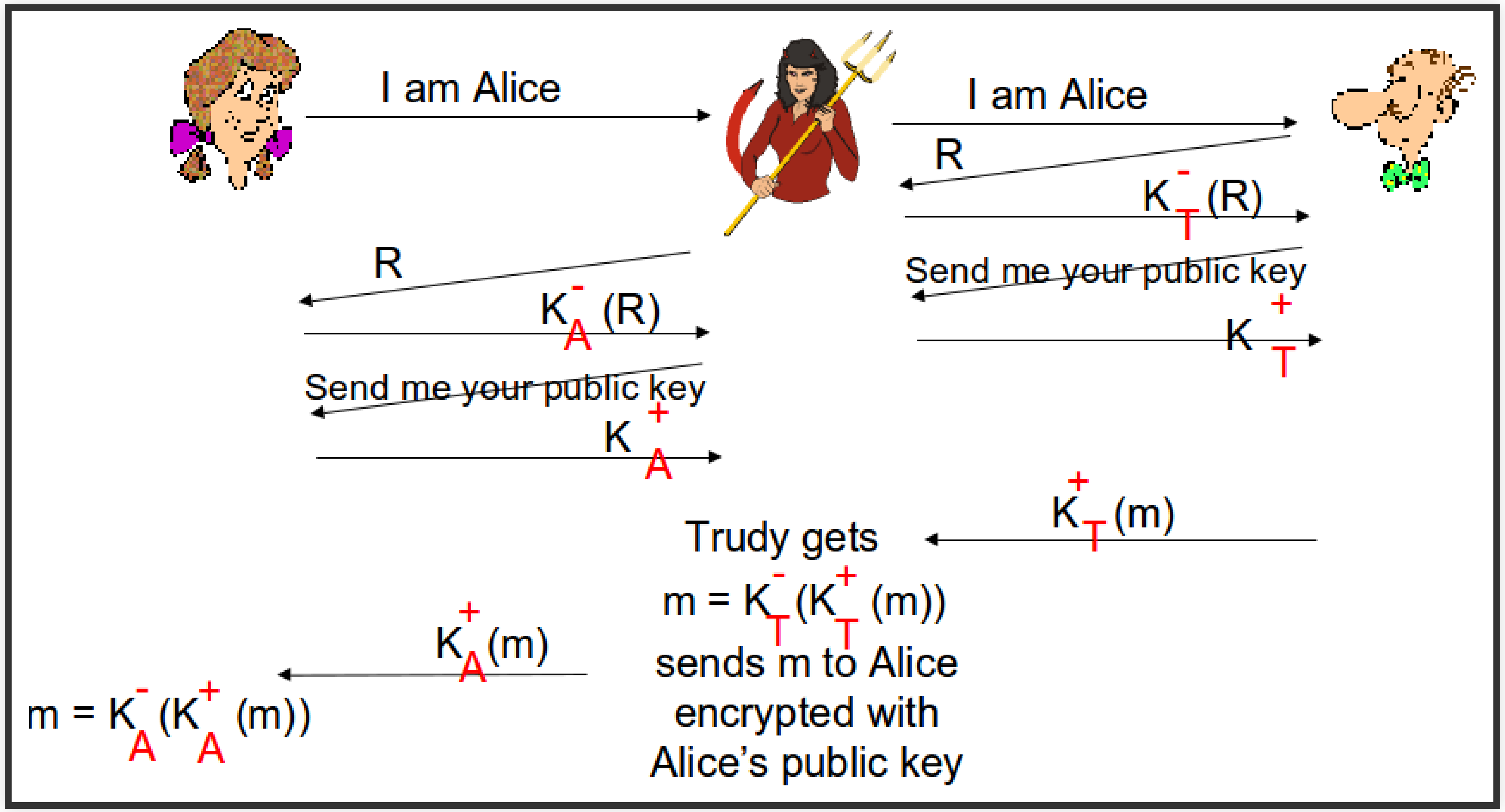
DIGITAL SIGNATURE



RECALL: 5.0 SECURITY HOLE

man (or woman) in the middle attack:

Trudy poses as Alice (to Bob) and as Bob (to Alice)



PUBLIC-KEY CERTIFICATION

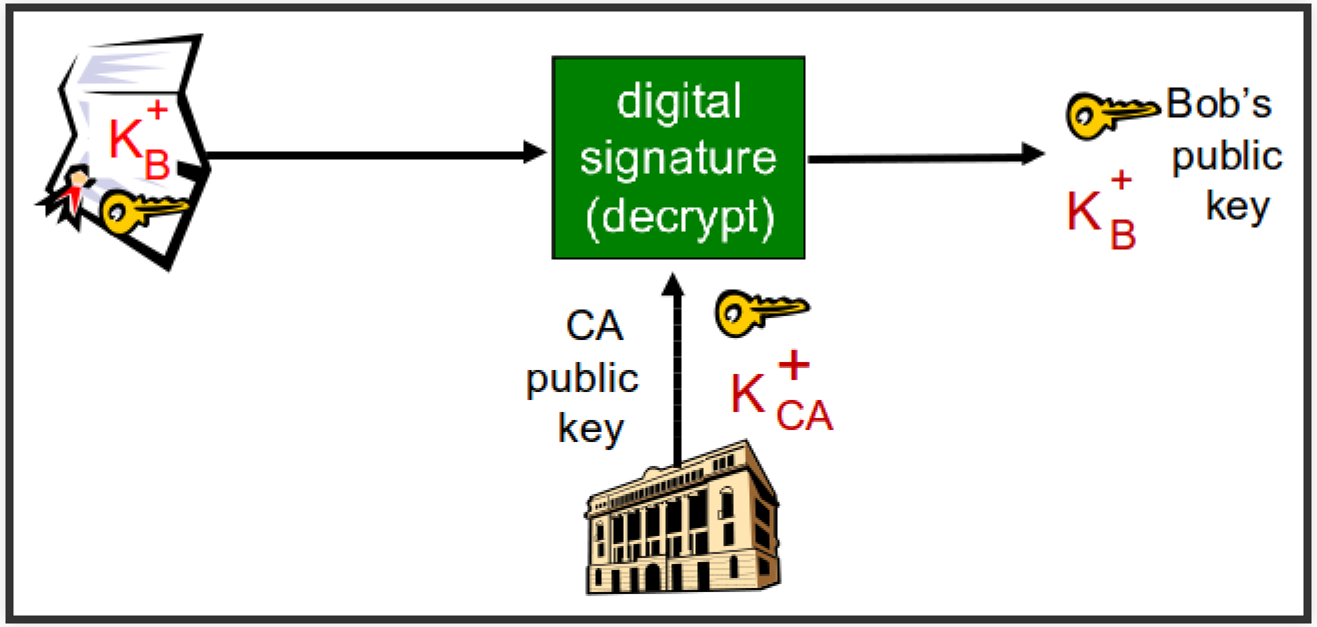
CERTIFICATION AUTHORITIES

- **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



JAVA TRUSTED CERTS

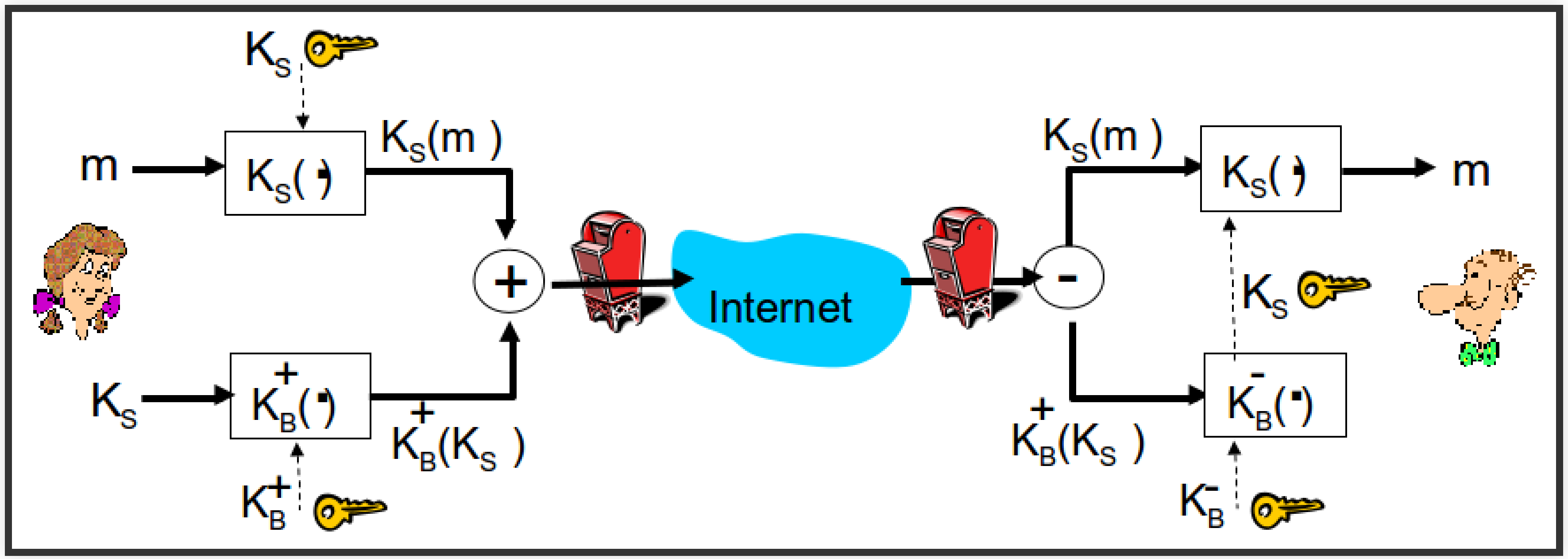
```
keytool -keystore "$JAVA_HOME/jre/lib/security/cacerts" -storepass changeit -list
```

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 104 entries

digicertassuredidrootca, Apr 16, 2008, trustedCertEntry,
Certificate fingerprint (SHA1): 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99
comodorsaca, May 11, 2015, trustedCertEntry,
Certificate fingerprint (SHA1): AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD
thawtepremiumserverca, May 20, 2015, trustedCertEntry,
Certificate fingerprint (SHA1): E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2B
....
```

SECURING E-MAIL

SECURE E-MAIL

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



SECURE E-MAIL

Alice:

- generates random symmetric private key, $\mathbf{K_S}$
- encrypts message with $\mathbf{K_S}$ (for efficiency)
- also encrypts $\mathbf{K_S}$ with Bob's public key
- sends both $\mathbf{K_S(m)}$ and $\mathbf{K_B(K_S)}$ to Bob

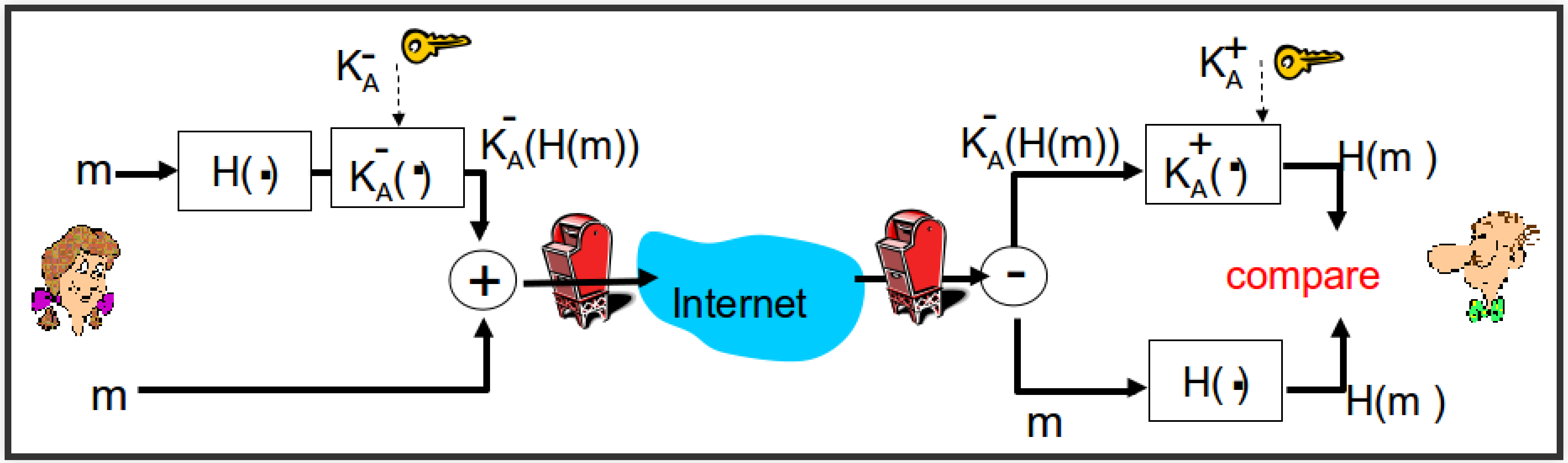
SECURE E-MAIL

Bob:

- uses his private key to decrypt and recover $\mathbf{K_S}$
- uses $\mathbf{K_S}$ to decrypt $\mathbf{K_S(m)}$ to recover **m**

SECURE E-MAIL (CONTINUED)

Alice wants to provide sender authentication and message integrity



Alice digitally signs message

sends both message (in the clear) and digital signature

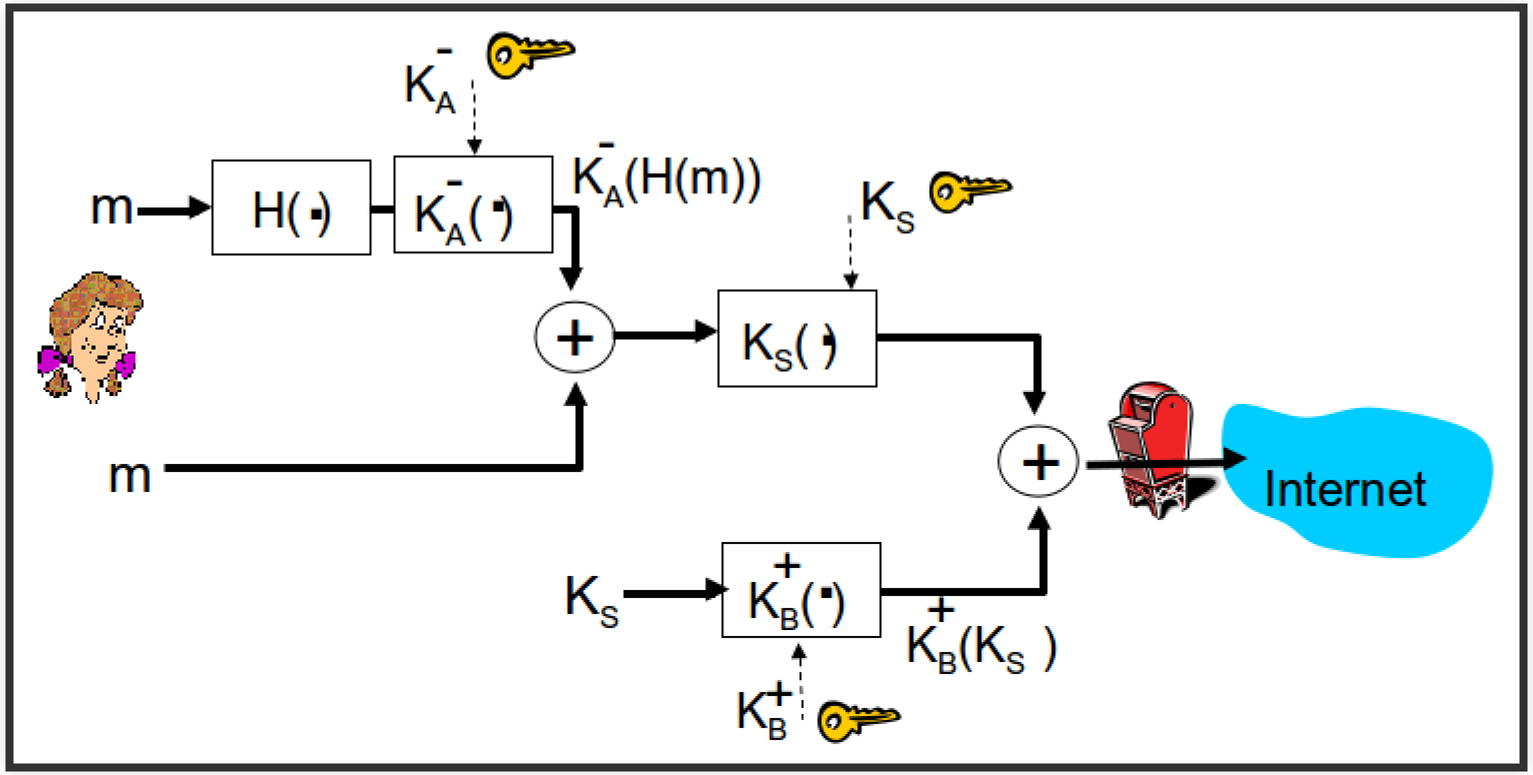
SECURE E-MAIL (CONTINUED)

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

Alice uses three keys:

1. her private key
2. Bob’s public key
3. newly created symmetric key

SECURE E-MAIL (CONTINUED)



PGP

- **Pretty Good Privacy**
- Written by Phil Zimmermann in 1991
- Email encryption scheme

PGP

Depending on the version:

- Uses MD5 or SHA for message digest
- CAST, tripple-DES or IDEA for symmetric key enc.
- RSA for public key enc.

PGP

Provides mechanism for public key verification

- signing others' keys you trust
- Key signing parties
- Users physically gather, exchange public keys and certify others with their private key

PGP HISTORY

- PGP encryption found its way outside the United States
- February 1993 Zimmermann became the formal target of a criminal investigation by US Gov.
- **"Munitions export without a license"**
  - Cryptosystems using keys larger than 40 bits were then considered munitions

PGP HISTORY

- Zimmermann published entire source code in a book - available world wide
- Anyone could OCR/type in and recreate
- The claimed principle was simple: export of munitions—guns, bombs, planes, and software—was (and remains) restricted; but the export of books is protected by the First Amendment.

QUESTIONS