# RSA and Digital Signatures

Kurose & Ross, Chapters 8.2-8.3 (5th ed.)

Slides adapted from:
J. Kurose & K. Ross \
Computer Networking: A Top Down Approach (5th ed.)
Addison-Wesley, April 2009.

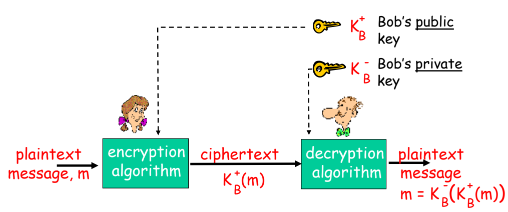

## Public Key Cryptography

*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
- 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

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, Adelson algorithm



## Prerequisite: modular arithmetic

- x mod n = remainder of x when divide 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
  $(a \bmod n)^d \bmod n = a^d \bmod n$
- Example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$



## 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").
4. Choose $d$ such that $ed-1$ is exactly divisible by $z$. (in other words: $ed$ mod $z = 1$ ).
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 message $m$ ($<n$), 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:

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).

Encrypting 8-bit messages.

| | bit pattern | $m$ | $m^e$ | $c = m^e \bmod n$ |
|---|---|---|---|---|
| encrypt: | 00001000 | 12 | 24832 | 17 |

| | $c$ | $c^d$ | $m = c^d \bmod n$ |
|---|---|---|---|
| decrypt: | 17 | 481968572106750915091411825223071697 | 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 $n = pq$ and $z = (p-1)(q-1)$
- Thus,

$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(ed \bmod z)} \bmod n \\ &= m^1 \bmod n \\ &= m \end{aligned}$$



## RSA: another important property

The following property will be *very* useful later:

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

*Result is the same!*



Why $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$ ?

Follows directly from modular arithmetic:

$$\begin{aligned} (m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n \end{aligned}$$

## 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.
- fact: factoring a big number is hard.

## Generating RSA keys

- have to find big primes p and q
- approach: make good guess then apply testing rules (see Kaufman)



## Session keys

- Exponentiation is computationally intensive
- DES is at least 100 times faster than RSA

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



## Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity
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



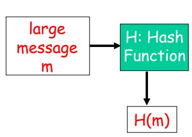
## Message Integrity

- allows communicating parties to verify that received messages are authentic.
  - Content of message has not been altered
  - Source of message is who/what you think it is
  - Message has not been replayed
  - Sequence of messages is maintained
- let's first talk about message digests



## Message Digests

- function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- note that H( ) is a many-to-1 function
- H( ) is often called a "hash function"




desirable properties:
- easy to calculate
- irreversibility: Can't determine m from H(m)
- collision resistance: computationally difficult to produce m and m' such that H(m) = H(m')
- seemingly random output



## Internet checksum: poor message digest

Internet checksum has some properties of hash function:
- ✓ produces fixed length digest (16-bit sum) of input
- ✓ is many-to-one

- but given message with given hash value, it is easy to find another message with same hash value.
  - e.g.,: simplified checksum: add 4-byte chunks at a time:

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

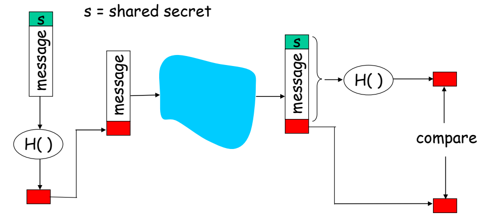
different messages but identical checksums!

## Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest



## Message Authentication Code (MAC)




- ***Authenticates sender***
- ***Verifies message integrity***
- No encryption !
- Also called "keyed hash"
- Notation: $MD_m = H(s||m)$ ; send $m||MD_m$

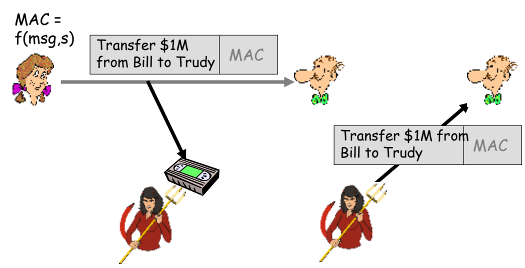


## End-point authentication

- want to be sure of the originator of the message – *end-point authentication*
- assuming Alice and Bob have a shared secret, will MAC provide end-point authentication?
  - we do know that Alice created message.
  - … but did she send it?

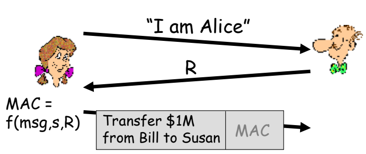


## Playback attack






## Defending against playback attack: nonce






## Digital Signatures

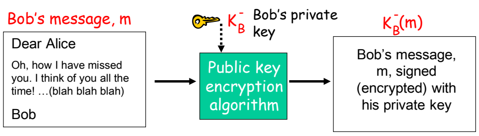
cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- goal is similar to that of MAC, except now use public-key cryptography
- *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 $K_B^-$, creating "signed" message, $K_B^-(m)$

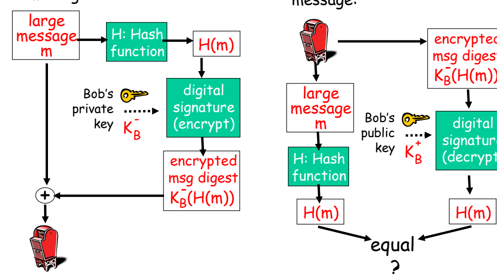





## Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature and integrity of digitally signed message:






## 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.



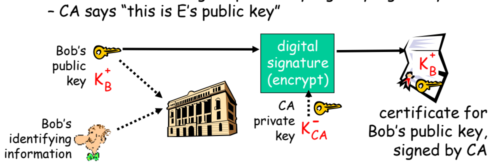
## Public-key certification

- 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 pizzas to Bob.
  - Bob doesn't even like Pepperoni



## Certification Authorities

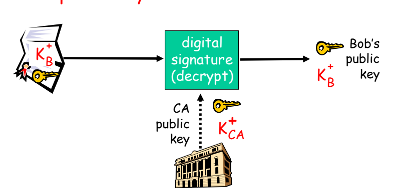
- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "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



## Certificates: summary

- primary standard X.509 (RFC 2459)
- certificate contains:
  - issuer name
  - entity name, address, domain name, etc.
  - entity's public key
  - digital signature (signed with issuer's private key)
- Public-Key Infrastructure (PKI)
  - certificates, certification authorities
  - often considered "heavy"



## Why study computer networks?

- An interface between theory (algorithms, mathematics) and practice
- Understanding the design principles of a truly complex system
- Industry-relevant knowledge
- Fun!

- Challenges in teaching computer networks
- Students' feedback