

# Authentication

Dr. Arjan Durresi  
Louisiana State University  
Baton Rouge, LA 70810  
Durresi@csc.lsu.edu

These slides are available at:

<http://www.csc.lsu.edu/~durresi/csc4601-07/>



- Passwords
- Humans and Computers

## Password

- ❑ Proof by knowledge, sharing
- ❑ Password guessing
  - On-line: limit tries, alarm
  - Off-line: dictionary attack
- ❑ Storing passwords
  - Per-node: /etc/passwd
  - Server: authentication storage server, retrieved by node (yp/NIS)
  - Facilitator: server says yes/no

## Password

- ❑ Storing hash instead of the password, the attacker can guess the password
- ❑ Better store the encrypted password. More difficult to get the key. But if the node itself is compromised ...
- ❑ Combination of both

## Address-based

- ❑ Rely on the identity of the “source”, inferred from the network address of the “source.”
- ❑ .rhosts
  - node, user name
- ❑ /etc/hosts.equiv
  - trusted hosts
- ❑ Threats:
  - break in one, break in all
  - often: A trusts B, then B trusts A
  - address spoofing

## Network Address Impersonation

- ❑ It is easy to transmit a packet claiming any address as source address, either at network layer or the data link layer.
- ❑ Star topology can help, but in small networks
- ❑ Router could check for the real identity of the source, or could filter against a given filter ...
- ❑ We need new ideas

## Humans and Computers

- ❑ Humans:
  - Short, memorable key (8 characters, 48 bits), directly or as key for longer key
- ❑ Computers:
  - (Long) high-quality secret
  - Hidden key (encrypted by password), directly (e.g., hash of the password)

## Passwords as Cryptographic Keys

- ❑ Cryptographic keys, especially public key ones, are large numbers
- ❑ How to generate a key from a password
- ❑ Easier for secret keys. For example do a cryptographic hash of the password and take 56 bits as DES key.
- ❑ More difficult for public key pair. Why?
- ❑ Convert a user password into a seed for a random number generator, which will be used to generate RSA keys.
- ❑ Low performance

## Message Authentication

- ❑ Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- ❑ Will consider the security requirements
- ❑ Then three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function

## Security Requirements: Types of Attacks

- ❑ Disclosure: Release of message contents to any person or process not possessing the appropriate key
- ❑ Traffic analysis: Discover the pattern of traffic between parties
- ❑ Masquerade: Insertion of messages into the network from a fraudulent source
- ❑ Content modification
- ❑ Sequence modification in a series of messages
- ❑ Timing modification: Delay or replay of message
- ❑ Source repudiation: Denial of transmission
- ❑ Destination repudiation: Denial of receipt

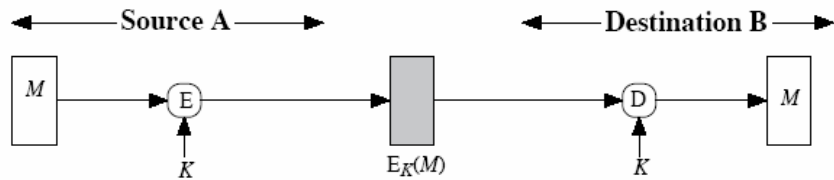
## Authentication Functions

- ❑ Any message authentication needs two levels:
  - Lower level – function that produces an
    - ❑ Message Encryption
    - ❑ Message Authentication Code (MAC)
    - ❑ Hash function
  - High level – Protocol that uses the authenticator as primitive

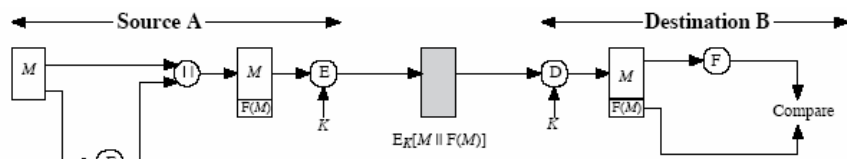
## Message Encryption

- ❑ Message encryption by itself also provides a measure of authentication
- ❑ If symmetric encryption is used then:
  - receiver know sender must have created it
  - since only sender and receiver now key used
  - know content cannot be altered
  - if message has suitable structure, redundancy or a checksum to detect any changes

## Symmetric Encryption: Confidentiality and Authentication

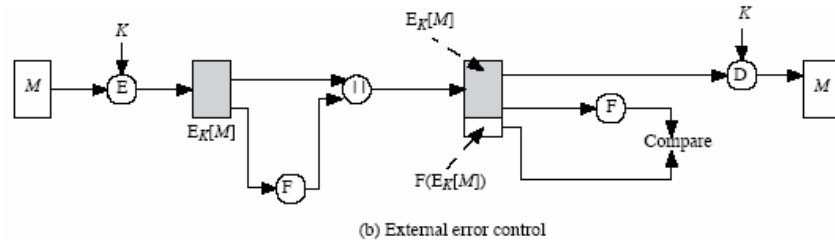


## Internal Error Control



(a) Internal error control

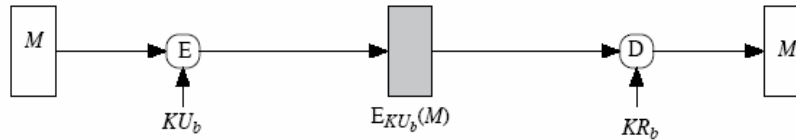
## External Error Control



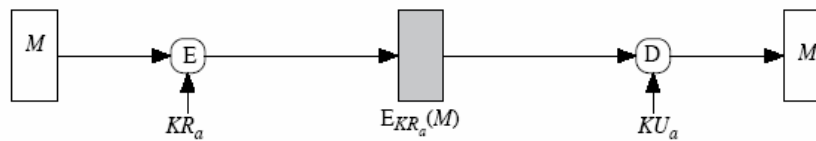
## Message Encryption

- If public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

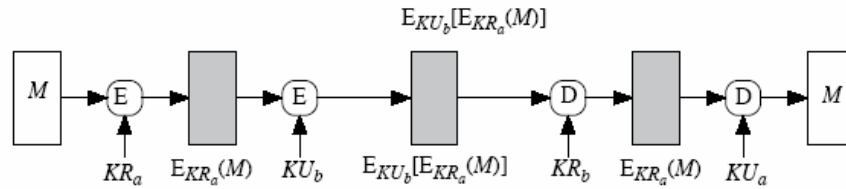
## Public Key Encryption: Confidentiality



## Public Key Encryption: Authentication and Signature



# Public Key Encryption: Confidentiality, Authentication and Signature



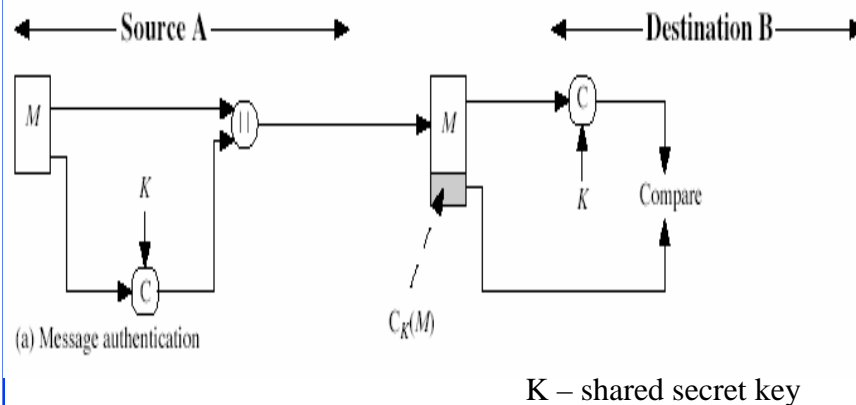
# Confidentiality and Authentication Implications of Message Encryption

<p><math>A \rightarrow B: E_K[M]</math></p> <ul style="list-style-type: none"> <li>•Provides confidentiality               <ul style="list-style-type: none"> <li>— Only A and B share <math>K</math></li> </ul> </li> <li>•Provides a degree of authentication               <ul style="list-style-type: none"> <li>— Could come only from A</li> <li>— Has not been altered in transit</li> <li>— Requires some formatting/redundancy</li> </ul> </li> <li>•Does not provide signature               <ul style="list-style-type: none"> <li>— Receiver could forge message</li> <li>— Sender could deny message</li> </ul> </li> </ul> <p>(a) Symmetric encryption</p>
<p><math>A \rightarrow B: E_{KU_b}[M]</math></p> <ul style="list-style-type: none"> <li>•Provides confidentiality               <ul style="list-style-type: none"> <li>— Only B has <math>KR_b</math> to decrypt</li> </ul> </li> <li>•Provides no authentication               <ul style="list-style-type: none"> <li>— Any party could use <math>KU_b</math> to encrypt message and claim to be A</li> </ul> </li> </ul> <p>(b) Public-key encryption: confidentiality</p>
<p><math>A \rightarrow B: E_{KR_a}[M]</math></p> <ul style="list-style-type: none"> <li>•Provides authentication and signature               <ul style="list-style-type: none"> <li>— Only A has <math>KR_a</math> to encrypt</li> <li>— Has not been altered in transit</li> <li>— Requires some formatting/redundancy</li> <li>— Any party can use <math>KU_a</math> to verify signature</li> </ul> </li> </ul> <p>(c) Public-key encryption: authentication and signature</p>
<p><math>A \rightarrow B: E_{KU_b}[E_{KR_a}(M)]</math></p> <ul style="list-style-type: none"> <li>•Provides confidentiality because of <math>KU_b</math></li> <li>•Provides authentication and signature because of <math>KR_a</math></li> </ul> <p>(d) Public-key encryption: confidentiality, authentication, and signature</p>

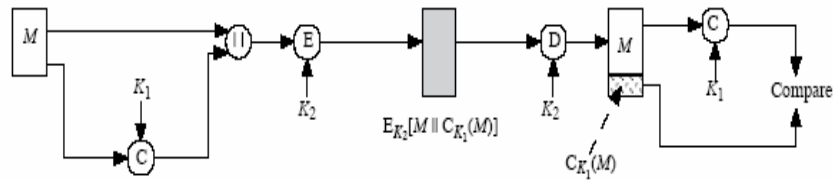
## Message Authentication Code (MAC)

- ❑ Generated by an algorithm that creates a small fixed-sized block
  - depending on both message and some key
  - like encryption though need not be reversible
- ❑ Appended to message as a **signature**
- ❑ Receiver performs same computation on message and checks it matches the MAC
- ❑ Provides assurance that message is unaltered and comes from sender

## Message Authentication Code

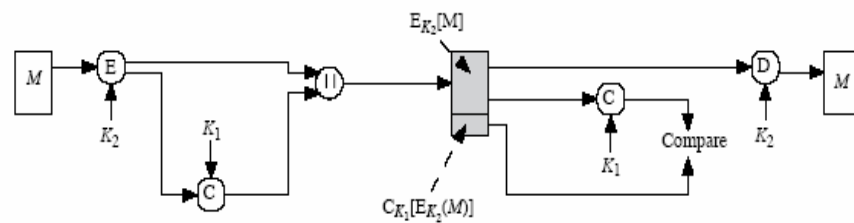


## Message Authentication and Confidentiality



Authentication tied to plaintext

## Message Authentication and Confidentiality



Authentication tied to ciphertext

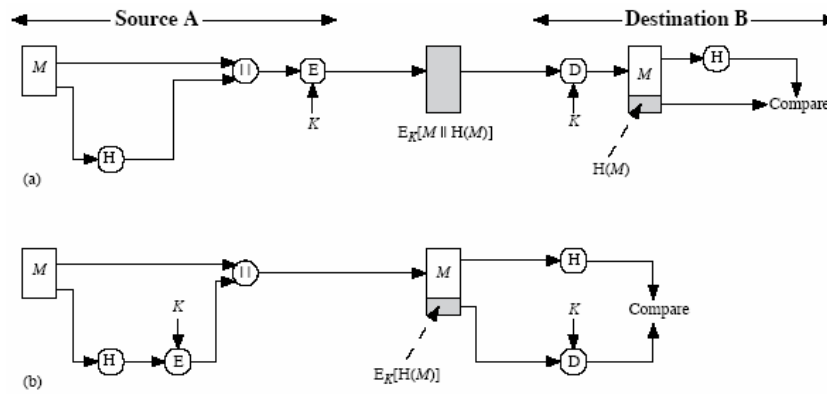
# Message Authentication Codes

- ❑ As shown the MAC provides authentication
- ❑ Can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- ❑ Why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (eg. archival use)
- ❑ Note that a MAC is not a digital signature because both sender and receiver share the key

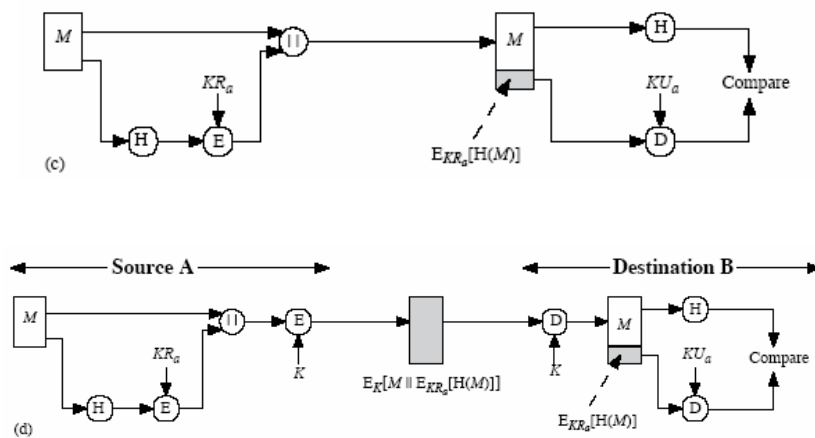
# Basic uses of MAC

$A \rightarrow B: M \parallel C_K(M)$ <ul style="list-style-type: none"><li>•Provides authentication<ul style="list-style-type: none"><li>—Only A and B share <math>K</math></li></ul></li></ul> <p>(a) Message authentication</p>
$A \rightarrow B: E_{K_2}[M \parallel C_{K_1}(M)]$ <ul style="list-style-type: none"><li>•Provides authentication<ul style="list-style-type: none"><li>—Only A and B share <math>K_1</math></li></ul></li><li>•Provides confidentiality<ul style="list-style-type: none"><li>—Only A and B share <math>K_2</math></li></ul></li></ul> <p>(b) Message authentication and confidentiality: authentication tied to plaintext</p>
$A \rightarrow B: E_{K_2}[M] \parallel C_{K_1}(E_{K_2}[M])$ <ul style="list-style-type: none"><li>•Provides authentication<ul style="list-style-type: none"><li>—Using <math>K_1</math></li></ul></li><li>•Provides confidentiality<ul style="list-style-type: none"><li>—Using <math>K_2</math></li></ul></li></ul> <p>(c) Message authentication and confidentiality: authentication tied to ciphertext</p>

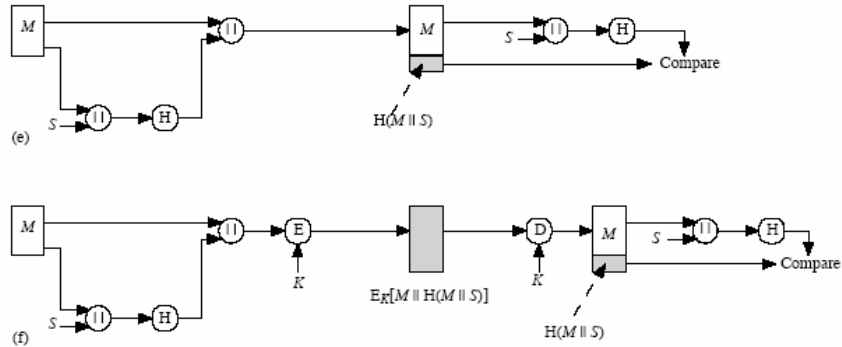
## Basic Uses of Hash Function



## Basic Uses of Hash Function



## Basic Uses of Hash Function



## Basic Uses of Hash Function

$A \rightarrow B: E_K[M \parallel H(M)]$ <ul style="list-style-type: none"> <li>•Provides confidentiality                             <ul style="list-style-type: none"> <li>– Only A and B share <math>K</math></li> </ul> </li> <li>•Provides authentication                             <ul style="list-style-type: none"> <li>– <math>H(M)</math> is cryptographically protected</li> </ul> </li> </ul> <p>(a) Encrypt message plus hash code</p>	$A \rightarrow B: E_K[M \parallel E_{KR_a}[H(M)]]$ <ul style="list-style-type: none"> <li>•Provides authentication and digital signature</li> <li>•Provides confidentiality                             <ul style="list-style-type: none"> <li>– Only A and B share <math>K</math></li> </ul> </li> </ul> <p>(d) Encrypt result of (c) - shared secret key</p>
$A \rightarrow B: M \parallel E_K[H(M)]$ <ul style="list-style-type: none"> <li>•Provides authentication                             <ul style="list-style-type: none"> <li>– <math>H(M)</math> is cryptographically protected</li> </ul> </li> </ul> <p>(b) Encrypt hash code - shared secret key</p>	$A \rightarrow B: M \parallel H(M \parallel S)$ <ul style="list-style-type: none"> <li>•Provides authentication                             <ul style="list-style-type: none"> <li>– Only A and B share <math>S</math></li> </ul> </li> </ul> <p>(e) Compute hash code of message plus secret value</p>
$A \rightarrow B: M \parallel E_{KR_a}[H(M)]$ <ul style="list-style-type: none"> <li>•Provides authentication and digital signature                             <ul style="list-style-type: none"> <li>– <math>H(M)</math> is cryptographically protected</li> <li>– Only A could create <math>E_{KR_a}[H(M)]</math></li> </ul> </li> </ul> <p>(c) Encrypt hash code - sender's private key</p>	$A \rightarrow B: E_K[M \parallel H(M) \parallel S]$ <ul style="list-style-type: none"> <li>•Provides authentication                             <ul style="list-style-type: none"> <li>– Only A and B share <math>S</math></li> </ul> </li> <li>•Provides confidentiality                             <ul style="list-style-type: none"> <li>– Only A and B share <math>K</math></li> </ul> </li> </ul> <p>(f) Encrypt result of (e)</p>

## MAC Properties

- ❑ A MAC is a cryptographic checksum
$$\text{MAC} = C_K(M)$$
  - condenses a variable-length message  $M$
  - using a secret key  $K$
  - to a fixed-sized authenticator
- ❑ Is a many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

## Attack on MAC

- ❑ If  $k > n$ ,  $k$  – length of key,  $n$  – length of message
- ❑  $M_1$  and  $\text{MAC}_1$  are known
- ❑ Attacker has to perform  $\text{MAC}_i = C_{k_i}(M_1)$  for all possible  $k_i$ .
- ❑ At least one key is guaranteed to produce a match
$$\text{MAC}_i = \text{MAC}_1$$
- ❑ A total of  $2^k$  MACs will be produced
- ❑ But there are only  $2^n < 2^k$  different MAC values
- ❑ The attacker has no way to know which is the correct key

## Attack on MAC (cont.)

- ❑ The attacker has to iterate:
  - Round 1
    - ❑ Given  $M_1$ ,  $MAC_1 = C_K(M_1)$
    - ❑ Compute  $MAC_i = C_{k_i}(M_1)$  for all  $2^k$  keys
    - ❑ Number of matches  $\approx 2^{(k-n)}$
  - Round 2
    - ❑ Given  $M_2$ ,  $MAC_2 = C_K(M_2)$
    - ❑ Compute  $MAC_i = C_{k_i}(M_1)$  for all  $2^{(k-n)}$  keys
    - ❑ Number of matches  $\approx 2^{(k-2n)}$
- ❑ How many rounds?
  - $\alpha = k/n$
  - For example if  $k = 80$  bit and  $n = 32$ 
    - ❑ First round :  $2^{48}$  keys
    - ❑ Second round:  $2^{16}$
    - ❑ Third round – find the key

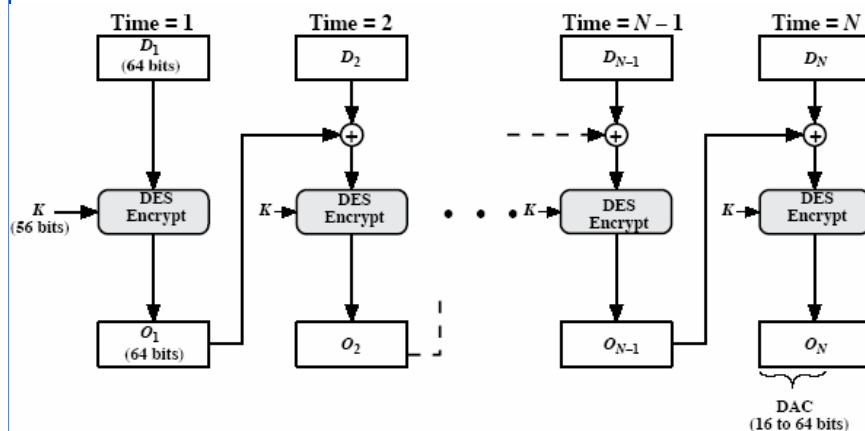
## Requirements for MACs

- ❑ Taking into account the types of attacks
- ❑ Need the MAC to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message

## Using Symmetric Ciphers for MACs

- ❑ Can use any block cipher chaining mode and use final block as a MAC
- ❑ **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - ❑ or the leftmost M bits ( $16 \leq M \leq 64$ ) of final block
- ❑ But final MAC is now too small for security

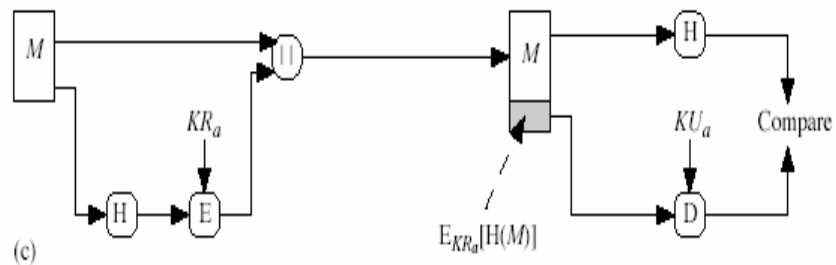
## Data Authentication Algorithm



## Hash Functions

- ❑ Condenses arbitrary message to fixed size
- ❑ Usually assume that the hash function is public and not keyed
  - cf. MAC which is keyed
- ❑ Hash used to detect changes to message
- ❑ Can use in various ways with message
- ❑ Most often to create a digital signature

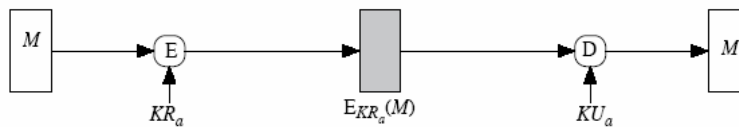
## Hash Functions & Digital Signatures



# Eavesdropping and Server Database Reading

- ❑ Public key:
  - Need to protect private key
- ❑ Use good password
  - Eavesdropping
- ❑ Use random challenge with signing/encryption using secret
  - Server database reading
- ❑ Lamport hash

# Public key



# Password

Alice  $\xrightarrow{\text{Alice, instant\_pwd}}$  Bob

Bob knows hash of Alice's pwd  
Bob computes hash of instant\_pwd and compares it with the stored value

# Random challenge

Alice  $\xrightarrow{\text{Alice}}$  Bob  $\xrightarrow{\text{R}}$  Alice  $\xrightarrow{\text{X}}$  Bob

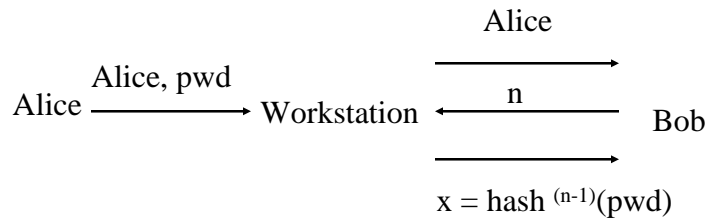
Knows Alice's secret  
Picks random R

Computes same function  
and compares X

## Lamport hash

Defend against eavesdropping and reading the database

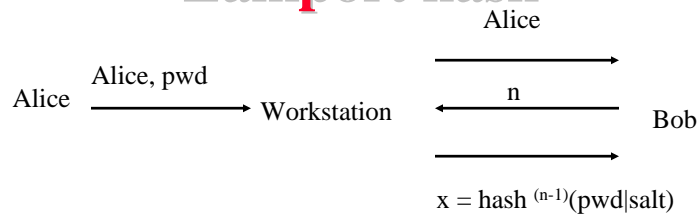
$\text{hash}^n(\text{pwd}) = \text{hash}(\text{hash} \dots (\text{hash}(\text{pwd})) \dots)$



Bob knows  $\langle n, \text{hash}^n(\text{pwd}) \rangle$

Bob compares  $\text{hash}(x)$  to  $\text{hash}^n(\text{pwd})$ , if equals replaces  $\langle n, \text{hash}^n(\text{pwd}) \rangle$  with  $\langle n-1, x \rangle$

## Lamport hash



- ❑ Add **salt**, chosen at password installation:
  - $\text{Hash}^n(\text{password}|\text{salt})$
  - Salt is stored at the server and it is unique for each user.
- ❑ The same password can be used in more than one server
- ❑ No need to change password when  $n=1$
- ❑ Adding salt prevents intruder to precompute  $\text{hash}^k$  for all passwords in a dictionary and then compare with stolen password hashes of all users

## Lamport hash

- ❑ Can be used limited number of times
- ❑ No mutual authentication
- ❑ **Small  $n$  attack**
  - An intruder impersonates Bob and sends to Alice a small value  $m$ , for example  $m=50$ .
  - Salt – known, eavesdropped from previous authentication
  - When Alice responds with  $\text{hash}^{50}(\text{password})$ , the intruder can impersonate her for some time ( $n>m$ )
  - Protection: Alice's workstation could display  $n$ , and Alice could check it

## Lamport hash – Human and paper

- ❑ Used when the workstation cannot be trusted
- ❑ During password installation  $\langle n, \text{hash}^n(\text{password}) \rangle$  are stored in the server and all values of  $\text{hash}^i(\text{password})$  are calculated and given to Alice
- ❑ When Alice logs in: She uses the string at the top of the page. Next time she uses the next value

## Strong Password Protocols

- ❑ Encrypted Key Exchange **EKE**
- ❑ Alice and Bob share a weak secret key  $W$ , which is a hash of Alice's password
- ❑ They do a Diffie-Hellman exchange, encrypting the Diffie-Hellman numbers with  $W$
- ❑ Do a mutual authentication based on the agreed Diffie-Hellman shared secret, which is strong secret.

## Trusted Intermediaries

- ❑ Can't do pair-wise authentication with secret key: key explosion
- ❑ Key distribution center (KDC)
  - Single point of failure, performance bottleneck
- ❑ Certification authorities (CAs)
  - Can be off-line
  - Single point of failure
  - Need to manage revocation list (CRL)

## Authentication of People

- ❑ What you know (passwords)
- ❑ What you have (keys)
- ❑ What you are (biometric devices)
- ❑ Where you are (physical)

## Passwords

- ❑ An eavesdropper might see the password when used
- ❑ An intruder might read the file where the computer stores password information
- ❑ The password might be easy to guess by someone making direct login attempts to the computer
- ❑ The password might be crackable by an off-line computer search
- ❑ In attempting to force users to chose unguessable password, the system might become inconvenient to users or users might resort to writing passwords down

## On-line Password Guessing

- ❑ Many systems set initial passwords related to the SSN
- ❑ Even if passwords are chosen so they are not obvious, they may be guessable if the impostor gets enough guesses
- ❑ How to limit the number of guesses?
- ❑ Design systems so that guesses have actually be typed by a human – reverse Turing
- ❑ Keep track of the number of consecutive incorrect passwords – used in ATM, PINs
  - Can block other people's accounts

## On-line Password Guessing

- ❑ Have incorrect password be processed slowly
- ❑ Try to catch the guesser, for example trying to trace the connection
  - Filter the behavior
- ❑ Trade-off between password and convenience of use
  - A random chosen 8-character string: if the attacker can guess a password every millisecond, it would take over three years to guess it
  - Let the users chose “good” password and have a program check them for example run through a spell checking and reject them if they are spelled correctly

## Off-line Password Guessing

- ❑ The attacker might be able to obtain a cryptographic hash of the password
- ❑ Even though the attacker cannot reverse the hash of the password, the attacker can guess the password, perform same hash
- ❑ An attacker can with a file of hashed passwords might hash all the words in a dictionary and check to see whether any of the passwords matches any of stored values
- ❑ How to protect the passwords in the system
  - Use salt – makes harder the hash
  - Encrypt the password file

## Passwords

- ❑ How long a password should be?
  - 64 bits of randomness is good
  - Humans cannot remember 64-bit random numbers
- ❑ Eavesdropping
  - Most systems do not display passwords as they are being typed
  - Have a list of passwords used only once
  - Have a list of passwords and have the system ask for specific ones on each authentication
- ❑ Careless users and administrators– most user see passwords as nuisances
  - Educate users, helping them to understand the reasons for security

## Trojan Horses

- ❑ A faked login prompt to capture passwords
- ❑ Counter measures:
  - Make it hard to have the appearance of login prompt
  - Use interrupts such as in Windows Ctrl-Alt-Delete
  - Prevent login by user programs

## Initial Password and Distribution

- ❑ Let the user set the password
- ❑ Create an initial strong password and give to users by mail ... the secure as good as the security of mail

## Authentication Tokens

- ❑ What you have
- ❑ Smart cards:
  - Challenge/response – Use PINs
  - Not trivial to reproduce and could keep a secret larger than most people can memorize
  - But can be lost or stolen
- ❑ Cryptographic calculator:
  - Interaction through a user (typing ...)

## Biometrics

- ❑ Accuracy:
  - False acceptance rate.
  - False rejection rate.
  - Can adversary select imposters?
    - ❑ Identical twins, family members, etc.
- ❑ Retinal scanner – The device examines the tiny blood vessels in the back of your eye. The layout is as distinctive as a fingerprint and easier to read
- ❑ Face recognition – Measure facial dimensions
- ❑ Iris scanner – Like retinal scanner, maps the distinctive layout of the iris of the eye. Less intimidating user interface
- ❑ Handprint reader – Measure the dimensions of the hand, not as accurate as fingerprints but also less expensive

# Fingerprints

- ❑ Vulnerability:
  - Dummy fingers and dead fingers
- ❑ Suitability and stability:
  - Not for people with high probability of damaged fingerprints
  - Not for kids growing up

# Voice Recognition

- ❑ Single phrase:
  - Can use tape recorder to fake
- ❑ Stability:
  - Background noise
  - Colds
  - Use with public phones

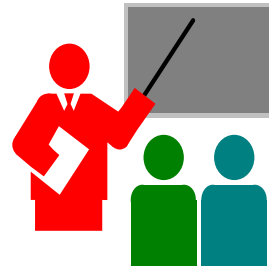
## Keystroke Timing

- ❑ Each person has a distinct typing timing/style
  - Hand/finger movements
- ❑ Suitability:
  - Best done for “local” authentication
    - ❑ Avoid network traffic delay

## Signatures

- ❑ Machines can't match human experts in recognizing shapes of signatures
- ❑ Add information of timing (dynamics) of movements
  - Signing on an electronic tablet

# Summary



- Passwords
- Authentication Functions
- Attacks