These Slides are prepared from
Matt Bishop slides and book "Introduction to Computer Security"
Benefiting from the Slides posted by Ahmad Al-Mulhem

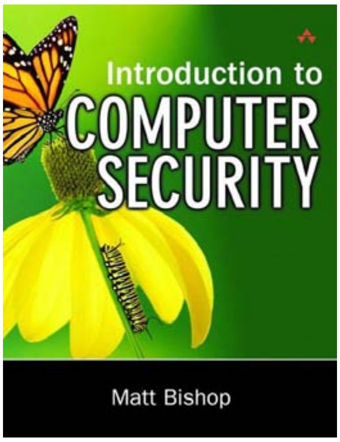
# Cryptography II

## ***Cipher Techniques - Ch 10***

## Key Management - Ch 9

Adnan Gutub
*gutub@kfupm.edu.sa*

*Computer Engineering Department*
*King Fahd University of Petroleum & Minerals*
*Dhahran, Saudi Arabia*






# Chapter 10: Cipher Techniques

- Some Problems
- Types of Ciphers
- Networks
- Examples

# Overview

Problems
- What can go wrong if you naively use ciphers
- Three Attacks, as simple examples !!

Cipher types
- Stream or block ciphers?

Networks
- Link vs end-to-end use

Examples
- Privacy-Enhanced Electronic Mail (PEM)
- Security at the Network Layer (IPsec)



# Network & Cryptography

- Cryptography >>> foundation for secure communication
- Encryption algorithms and protocols are valuable components/tools
- Cryptosystems over a network >>> many problems!
- Cryptography is sensitive to environment:
  - Using cipher requires knowledge of environment, and threats in the environment, in which cipher will be used
  - Is the set of possible messages small?
  - Do the messages exhibit regularities that remain after encipherment?
  - Can an active wiretapper rearrange or change parts of the message?

## Attack #1: Precomputation

Set of possible messages *M* small

Public key cipher *f* used

Idea: precompute set of possible ciphertexts *f*(*M*), build table (*m*, *f*(*m*))

When ciphertext *f*(*m*) appears, use table to find *m*

Also called *forward searches*



## Attack #1: Precomputation (Example)

- Cathy knows Alice will send Bob one of two messages: enciphered BUY, or enciphered SELL: {BUY, SELL}
- Using public key $e_{Bob}$, Cathy precomputes
  - $c_1$ = { BUY, $e_{Bob}$}
  - $c_2$ = { SELL, $e_{Bob}$ }
- Cathy sees Alice send Bob $c_2$
- Cathy knows Alice sent SELL

## Attack # 2: Misordered Blocks

Alice sends Bob message

– $n_{Bob} = 77$, $e_{Bob} = 17$, $d_{Bob} = 53$
– Message is LIVE (11 08 21 04)
– Enciphered message is 44 57 21 16

Eve intercepts it, rearranges blocks

– Now enciphered message is 16 21 57 44

Bob gets enciphered message, deciphers it

– He sees EVIL



## Notes

Digitally signing each block won't stop this attack

Two approaches:

– Cryptographically hash the *entire* message and sign it

– Place sequence numbers in each block of message, so recipient can tell intended order
  - Then you sign each block

## Attack # 3: Statistical Regularities

If plaintext repeats, ciphertext may too

Example using DES:

- input (in hex):
  3231 3433 3635 3837 3231 3433 3635 3837
- corresponding output (in hex):
  ef7c 4bb2 b4ce 6f3b ef7c 4bb2 b4ce 6f3b

Fix: cascade blocks together (chaining)



## Summary

What These Mean:

Use of:

- strong cryptosystems
- well-chosen (or random) keys

–Is this enough to be secure?? NO….

# Stream & Block Ciphers

$E$ encipherment function

- $E_k(b)$ encipherment of message $b$ with key $k$
- In what follows, $m = b_1 b_2 \ldots$, each $b_i$ of fixed length

Block cipher

- $E_k(m) = E_k(b_1)E_k(b_2) \ldots$

Stream cipher

- $k = k_1 k_2 \ldots$
- $E_k(m) = E_{k1}(b_1)E_{k2}(b_2) \ldots$
- If $k_1 k_2 \ldots$ repeats itself, cipher is *periodic* and the key-length of its period is one cycle of $k_1 k_2 \ldots$



# Examples

DES - Block cipher

- $b_i$ = 64 bits, $k$ = 56 bits
- Each $b_i$ enciphered separately using $k$

Vigenère cipher - Stream cipher

- $b_i$ = 1 character, $k = k_1 k_2 \ldots$ where $k_i$ = 1 character
- Each $b_i$ enciphered using $k_{i \bmod \text{length}(k)}$

One time pad - Stream cipher ------- Good example

- XOR'ing each bit of key with one bit of message
- Not periodic – key period is never supposed to repeat

## Self-Synchronous Stream Cipher
### key drawn from plaintext

Take key from message itself (*autokey*)

Example: Vigenère

- *key* XTHEBOYHASTHEBA
- *plaintext* THEBOYHASTHEBAG
- *ciphertext* QALFPNFHSLALFCT

Problem:

- Statistical regularities in plaintext show in key
- Once you get any part of the message, you can decipher more



## Self-Synchronous Stream Cipher
### key drawn from ciphertext

Take key from ciphertext (*autokey*)

Example: Vigenère

- *key* XQXBCQOVVNGNRTT
- *plaintext* THEBOYHASTHEBAG
- *ciphertext* QXBCQOVVNGNRTTM

Problem:

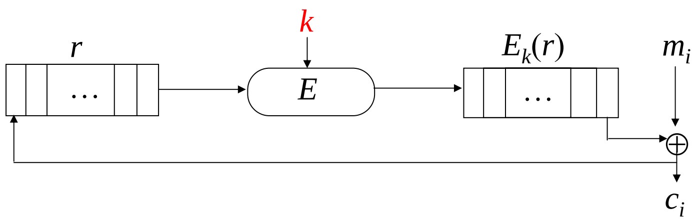
- Attacker gets key along with ciphertext, so deciphering is trivial

# Improving Autokey Stream Cipher

## Cipher feedback mode

Cipher feedback mode: 1 bit of ciphertext fed into $n$ bit register

- Self-healing property: if ciphertext bit received incorrectly, it and next $n$ bits decipher incorrectly; but after that, the ciphertext bits decipher correctly
- Need to know ***k***, $E$ to decipher ciphertext






# Block Ciphers - problem

Encipher, decipher multiple bits at once ----Advantage

- Each block enciphered independently
- Software implementation: block ciphers run faster than software implementation of stream ciphers

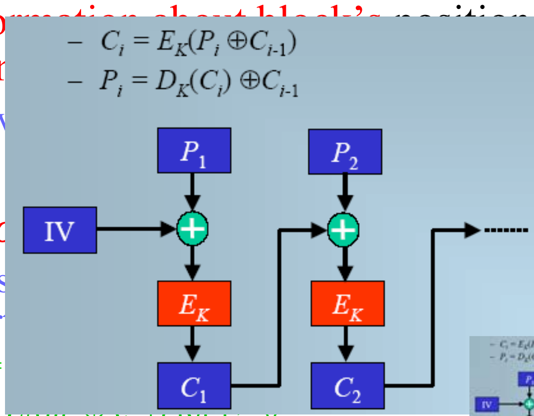
Problem: identical plaintext blocks produce identical ciphertext blocks

- Example: two database records
  - `MEMBER: Basem INCOME $100,000`
  - `MEMBER: Salem INCOME $100,000`
- Encipherment:
  - `ABCQZRME GHRSB CTXUVYSS RMGRPFQN`
  - `ABCQZRME ORPRZ CTXUVYSS RMGRPFQN`

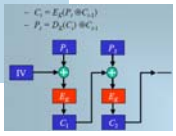
## Solution to Block Cipher Problem

Insert info … into
the plain …
– Disadv … reduced

*Cipher blo…*
– Exclus … previous
cipher …
• $c_0$ = …
• $c_i$ = …




where *IV* is the initialization vector





## Multiple Encryption

Double encipherment: $c = E_{k'}(E_k(m))$
– Effective key length is $2n$, if $k$, $k'$ are length $n$
– Problem: breaking it requires $2^{n+1}$ encryptions, not $2^{2n}$ encryptions

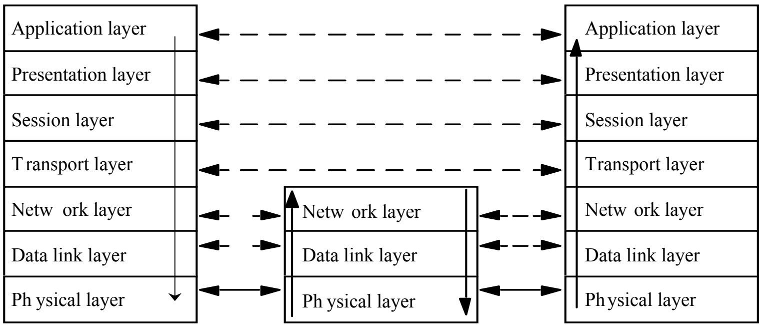
Triple encipherment:
– EDE mode: $c = E_k(D_{k'}(E_k(m))$
• Used in Financial Application: ANSI X9.17 & ISO 8732
– Triple encryption mode: $c = E_k(E_{k'}(E_{k''}(m))$

# Networks & Cryptography

ISO/OSI model

Conceptually, each host has peer at each layer

- Peers communicate with peers at same layer

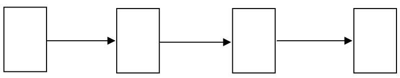





# Link & End-to-End (E2E) Protocols
## Encryption

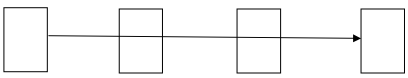
Link encryption

- Each host enciphers message so host at "next hop" can read it
- Message can be read at intermediate (in-between) hosts



End-to-end (E2E) encryption

- Host enciphers message so host at other end of communication can read it
- Message cannot be read at intermediate (in-between) hosts



# Examples & Crypto Considerations

## PPP Encryption Control Protocol - Link protocol

- Host gets message, deciphers it
  - Figures out where to forward it – which neighbor to send it to
  - Re-Enciphers it in appropriate key with that neighbor and forwards it
    - Secure among attackers monitoring the network
    - Vulnerable among attackers within the intermediate hosts
    - Each host shares key with neighbor

## TELNET protocol - End-to-end (E2E) protocol

- Application layer protocol – virtual terminal on a remote host
- Messages between client, server enciphered
  - Encipherment, decipherment occur only at the end hosts
    - » message encipherd throughout its journey
  - Secure among attackers monitoring the network & within the hosts
  - Each host shares key with destination
  - But, Attackers can read the routing information used to forward the message



# Encryption Protocols & Traffic Analysis

## Link encryption

- Shares crypto key with neighbors
- Can protect headers of packets
- Me
  eve
  enc
  for
- Att
  -
  -
- Pos
  destination
  - Note: may be able to deduce this from traffic flows

## End-to-end encryption

-each host & destination share crypto key

- No deciphering is in the

et headers

ediate nodes

route packet

monitoring

mediate hosts

tination

**Traffic Analysis:**

Crypto-analyst can sometimes gain information from the sender and recipient information and from the routing information; without benefiting from the content of the message

# Example Protocols

Privacy-Enhanced Electronic Mail (PEM)

– Applications layer protocol

Internet Protocol (IP) Security (IPSec)

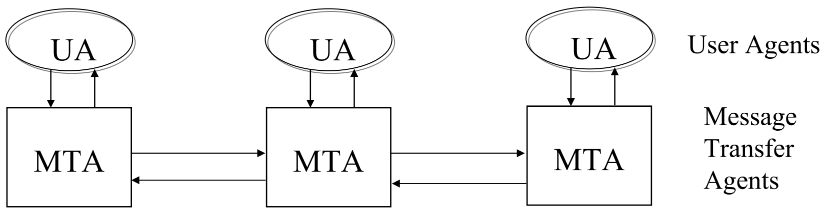
– Network layer protocol

- Successor of the NLSP (Network Layer Security Protocol) that was standardized by ISO



# Message Handling System

***need for Privacy-Enhanced Electronic Mail (PEM)***




Ma
•UA
•UA
•UA
•MT
unti

Assumptions:

Interchange Key – Available

- Asymmetric Crypto System:
  Authentic Public keys are with all parties
- Symmetric Crypto System:
  Authentic Secret keys are with all parties

rabilities:
essage at any MTA
essage at network
message fooling
e letters and inject it
Sender can deny having sent a letter

## Goals of
## Privacy-Enhanced Electronic Mail (PEM)

Confidentiality
- Only sender and recipient(s) can read message

Origin authentication
- Identify the sender precisely

Data integrity
- Any changes in message are easy to detect

Non-repudiation of origin
- Whenever possible …



## PEM Design Principles

Do not change related existing protocols
- Cannot alter SMTP

Do not change existing software
- Need compatibility with existing software

Make use of PEM optional & independent
- Available if desired, but email still works without them
- Some recipients may use it, others not

Enable communication without pre-arrangement
- Out-of-bands authentication, key exchange problematic

# PEM Basic Design: Key Types

**will be revisited at the Key-Management Chapter 9**

Two key types

- *Interchange keys* tied to sender, recipients and is static (for some set of messages)
  - Like a public/private key pair
  - Must be available *before* messages sent
- *Data exchange keys* generated for each message
  - Like a session key, session being the message

**Key Types**

Interchange Keys : associated with user

- long-term
- compromising is catastrophic

Session Keys : associated with communication

- short-term
- compromising does not affect long-term security



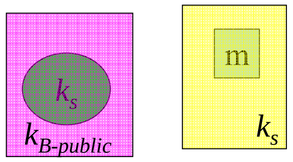
# PEM Basic Design: Sending

PEM Confidentiality

- $m$ message
- $k_s$ data exchange key - DEK
  - Session Key (Short term – used once)
- $k_{B\text{-}public}$ Bob's interchange key

$\{ m \} k_s \,||\, \{ k_s \} k_{B\text{-}public}$

Alice ⟶ Bob

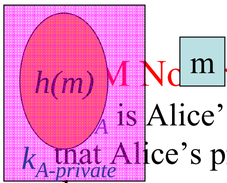


# PEM Basic Design: Integrity

PEM Integrity & authentication:

- *m* message
- *h*(*m*) hash of message *m* —*h(m)*=Message Integrity Check (MIC)
- $k_{A\text{-}private}$ Alice's interchange key

$$\text{Alice} \xrightarrow{\; m \,||\, \{\, h(m) \,\}\, k_{A\text{-}private} \;} \text{Bob}$$




PEM Non-repudiation:

- If $k_A$ is Alice's private key, this establishes that Alice's private key was used to sign the message

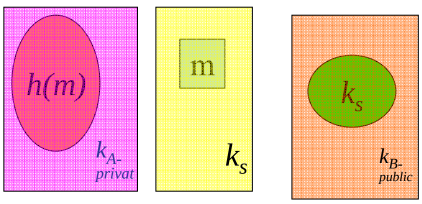


# PEM Basic Design: Everything

Confidentiality, integrity, authentication:

- Notations as in previous slides
- If $k_{A\text{-}private}$ is private key, get non-repudiation too

$$\text{Alice} \xrightarrow{\; \{\, m \,\}\, k_s \,||\, \{\, h(m) \,\}\, k_{A\text{-}private} \,||\, \{\, k_s \,\}\, k_{B\text{-}public} \;} \text{Bob}$$



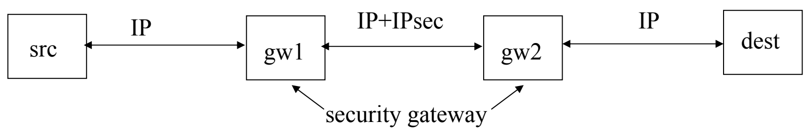
# Internet Protocol Security (IPSec)

IP is the primary protocol in the Internet Network Layer having the task of *delivering* datagrams (packets) from the source host to the destination host solely based on its address.

IPSec: Network layer security

- Provides confidentiality, integrity, authentication of endpoints, replay detection

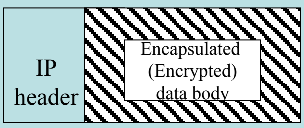
IPsec: used to protect data flows between a pair of hosts (e.g. computer users or servers), between a pair of security gateways (e.g. routers or firewalls), or *between security gateway and host*.






# IPsec Modes: Transport Mode / Tunnel Mode

## Transport Mode



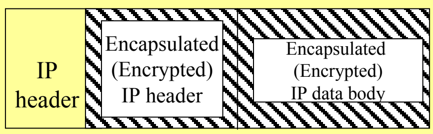

Encapsulate IP packet *data area* only

Use IP to send IPsec-wrapped data packet

Note: IP header not protected

Used when both end-points supports IPsec

## Tunnel Mode




Encapsulate *IP packet* (header and data)

Use IP to send IPsec-wrapped packet

Note: IP header protected

The unencrypted IP header is used to deliver the encrypted packets to a system where can be decrypted and forwarded

Used when any or both end-points do not support IPsec but at least two intermediate hosts do

# Two Protocols of IPsec
## for Message Security

Authentication Header (AH)
- Message integrity
- Origin authentication
- Anti-replay

Encapsulating Security Payload (ESP)
- Confidentiality ----extra: added to AH
- Beside all others provided by AH
  - Message integrity
  - Origin authentication
  - Anti-replay

Both are based on Cryptography supplied by the Internet Key Exchange (IKE)



# IPsec Architecture

Security Policy Database (SPD)
- Says how to handle messages based on *IP & transport layer headers* to do one of the following:
  1. discard them
  2. add security services
  3. forward message unchanged
- SPD associated with network interface
- SPD determines appropriate entry from packet attributes:
  - Including source
  - Including destination
  - Including transport protocol

## Without detailing the IPsec!! – left for a HW

# Which to Use: PEM, IPsec

What do the security services apply to?

- If applicable to one application *and* application layer mechanisms available, use that
  - PEM for electronic mail
- If more generic services needed, look to lower layers
  - IPsec for network layer, either end-to-end or link mechanisms, for connectionless channels as well as connections
- If endpoint is host, IPsec sufficient; if endpoint is user, application layer mechanism such as PEM needed



# Key Points

Key management critical to effective use of cryptosystems

- Different levels of keys (session *vs.* interchange)

Keys need infrastructure to identify holders, allow revoking

- Key escrowing complicates infrastructure

Digital signatures provide integrity of origin and content

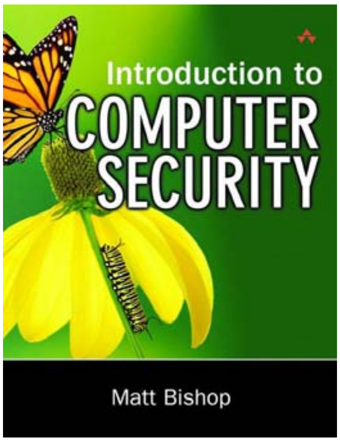
Much easier with public key cryptosystems than with classical cryptosystems

These Slides are prepared from
Matt Bishop slides and book "Introduction to Computer Security"
Benefiting from the Slides posted by Ahmad Al-Mulhem

# Cryptography II

Cipher Techniques - Ch 10

***Key Management - Ch 9***




Adnan Gutub
*gutub@kfupm.edu.sa*

*Computer Engineering Department*
*King Fahd University of Petroleum & Minerals*
*Dhahran, Saudi Arabia*



# Chapter 9: Key Management

- Key Distribution Problem
- Session and Interchange Keys
- Key Exchange
- Cryptographic Key Infrastructure
- Storing and Revoking Keys
- Digital Signatures

# Overview

- Key Distribution Problem
- Key exchange
  - Session vs. interchange keys
  - Classical, public key methods
- Cryptographic key infrastructure
  - Certificates
- Key storage
  - Key revocation
- Digital signatures

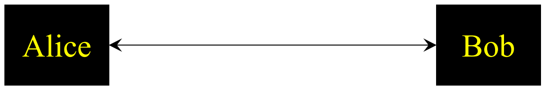


# Classical Key Exchange

- Bootstrap problem: how do Alice, Bob begin?
  - Alice can't send it to Bob in the clear!
- Assume trusted third party, Cathy
  - Alice and Cathy share secret key $k_A$
  - Bob and Cathy share secret key $k_B$
- Use this to exchange shared key $k_s$

# Key Distribution Problem

Algorithm like DES, Rijndael requires a shared a key!
Bootstrap problem: how do Alice and Bob begin?
Alice can't send the key to Bob in the clear!




**Key Types**

Interchange Keys : associated with user
- long-term
- compromising is catastrophic

Session Keys : associated with communication
- short-term
- compromising does not affect long-term security



# Benefits

## Limits amount of traffic enciphered with single key

– Standard practice, to decrease the amount of traffic an attacker can obtain

## Prevents some attacks

– Example: *Alice* will send *Bob* message that is either "BUY" or "SELL". *Eve* computes possible ciphertexts { "BUY" } $k_B$ and { "SELL" } $k_B$. *Eve* intercepts enciphered message, compares, and gets plaintext at once

# Key Exchange Algorithms

Goal: Alice, Bob get shared key

- Key cannot be sent in clear
  - Attacker can listen in
  - Key can be sent enciphered, or derived from exchanged data plus data not known to an eavesdropper
- Alice, Bob may trust third party
- All cryptosystems, protocols publicly known
  - Only secret data is the keys, ancillary information known only to Alice and Bob needed to derive keys
  - Anything transmitted is assumed known to attacker



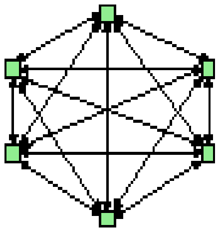
# Key Distribution Problem

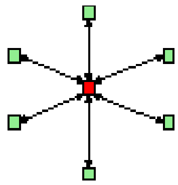
**Possible Solutions:**

1. Physical Distribution:
   - use a trusted courier (secure channel)
   - used widely until 1970s
2. Distribution Protocol:
   - assume a trusted 3rd party
3. Public Key Cryptography:
   - most widely used technique

# Key Distribution Problem

For n users, [n(n−1)]/2 keys!
10000 students, 50 million keys!
How do you manage them?
What if compromised?!

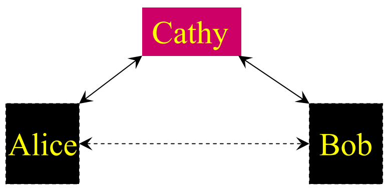




For n users, n keys
For 10000 students, 10000 keys
Session keys generated as needed
Needs protocol and trusted server



# Key Exchange Protocols




Assumptions

– Alice & Bob cannot arrange the session key in the clear
– Alice & Bob trust a 3rd part Cathy
– Alice & Bob already have interchange keys with Cathy
– Cryptosystem and protocol are public; keys are secret
– Attacker is the network!

# Notation

$X \rightarrow Y : \{M\}\ k$

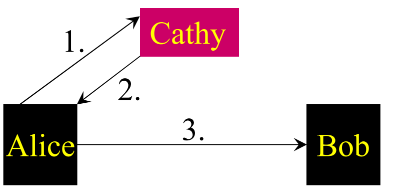
– $X$ sends to $Y$ message M enciphered by key $k$

$A \rightarrow B : r_1 || \{ M || N_a\}\ k_{ab} || \{ T_a \}\ k_{bs}$

– communicating parties: A, B, S
– message: M
– concatenation: ||
– nonce numbers (numbers used once; random): $r_1$, $N_a$, $N_b$, . . .
– timestamps: $T_a$, $T_b$, . . .
– shared keys: $K_{ab}$, $K_{bs}$



# Simple Protocol




## Steps

1. A→C: $\{B\}K_{ac}$
2. C→A: $\{K_{ab}\}K_{ac} || \{K_{ab}\}K_{bc}$
3. A→B: $\{K_{ab}\}K_{bc}$

## Problems

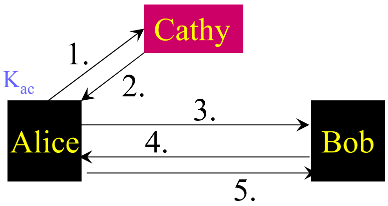
- How does Bob know he is talking to Alice?
- Replay attack (3, msg)
  - Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
- msg = "deposit $199 in my account"

Protocols must provide authentication and defense against replay

# Needham-Schroeder Protocol

Steps :

1. $A \rightarrow C : \{A \parallel B \parallel N_a\}$
2. $C \rightarrow A : \{A \parallel B \parallel N_a \parallel K_{ab} \parallel \{ A \parallel K_{ab} \}K_{bc}\}K_{ac}$
3. $A \rightarrow B : \{A \parallel K_{ab}\}K_{bc}$
4. $B \rightarrow A : \{N_b\}K_{ab}$
5. $A \rightarrow B : \{N_b - 1\}K_{ab}$




Argument: Alice talking to Bob

Second message:

- Enciphered using key only she and Cathy knows (So Cathy enciphered it)
- Response to first message ($N_a$ in it matches $N_a$ in first message)

Third message:

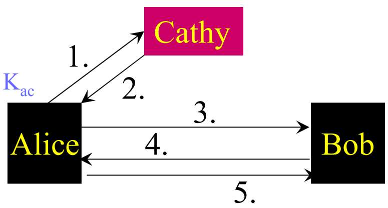
- Alice knows only Bob can read it (only Bob can derive session key from message)
- Any messages enciphered with that key are from Bob



# Needham-Schroeder Protocol

Steps :

1. $A \rightarrow C : \{A \parallel B \parallel N_a\}$
2. $C \rightarrow A : \{A \parallel B \parallel N_a \parallel K_{ab} \parallel \{ A \parallel K_{ab} \}K_{bc}\}K_{ac}$
3. $A \rightarrow B : \{A \parallel K_{ab}\}K_{bc}$
4. $B \rightarrow A : \{N_b\}K_{ab}$
5. $A \rightarrow B : \{N_b - 1\}K_{ab}$




Argument: Bob talking to Alice

Third message:

- Enciphered using key only he and Cathy knows (So Cathy enciphered it)
- Cathy provided session key and says Alice is other party

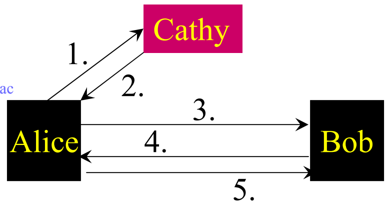
Fourth & Fifth message:

- Uses session key to determine if it is replay from Eve
- If not, Alice will respond correctly in fifth message
- If so, Eve cant decipher $N_b$ and so cant respond, or responds incorrectly

# Needham-Schroeder Protocol

Steps :

1. $A \rightarrow C : \{A \parallel B \parallel N_a\}$
2. $C \rightarrow A : \{A \parallel B \parallel N_a \parallel K_{ab} \parallel \{A \parallel K_{ab}\}K_{bc}\}K_{ac}$
3. $A \rightarrow B : \{A \parallel K_{ab}\}K_{bc}$
4. $B \rightarrow A : \{N_b\}K_{ab}$
5. $A \rightarrow B : \{N_b - 1\}K_{ab}$




Discussion

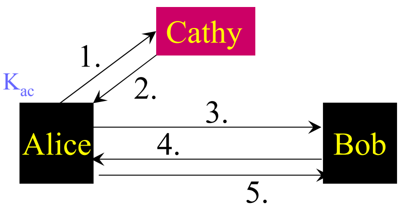
- Prevent eavesdropping, replay, modification, masquerading
- Fails if the session key ($K_{ab}$) is compromised!
  - Eve can replay the last 3 messages
  - Eve can pretend to be Alice
- Variations:
  - use timestamps (Denning and Sacco 81)
  - use an identification-number (Otway-Rees 87)



# Needham-Schroeder Protocol + Timestamps

Steps :

1. $A \rightarrow C : \{A \parallel B \parallel N_a\}$
2. $C \rightarrow A : \{A \parallel B \parallel N_a \parallel K_{ab} \parallel \{A \parallel T \parallel K_{ab}\}K_{bc}\}K_{ac}$
3. $A \rightarrow B : \{A \parallel T \parallel K_{ab}\}K_{bc}$
4. $B \rightarrow A : \{N_b\}K_{ab}$
5. $A \rightarrow B : \{N_b - 1\}K_{ab}$




Discussion

- Adding timestamps prevent replaying old session keys
- Needs clock synchronization!
  - may either reject valid messages or accept replays
- Forms the basis for Kerberos protocol (Ticket - issuer proofs identity)
  - Used by MS Window OS

## Key Exchange through Public Key Crypto

Here interchange keys known

- $e_A$, $e_B$ Alice and Bob's public keys known to all
- $d_A$, $d_B$ Alice and Bob's private keys known only to owner

Simple protocol (Version 1)

- Alice and Bob exchange session key $k_{ab}$

Alice —— $\{ k_{ab} \} e_B$ ——→ Bob



## Problem and Solution

Vulnerable to forgery or replay

- Because $e_B$ known to anyone, Bob has no assurance that Alice sent message

Simple fix uses Alice's private key

Simple protocol (Version 2)

- Alice and Bob exchange session key $k_{ab}$

Alice —— $\{ \{ k_{ab} \} d_A \} e_B$ ——→ Bob

## Man In the Middle Attack (Spoofing)

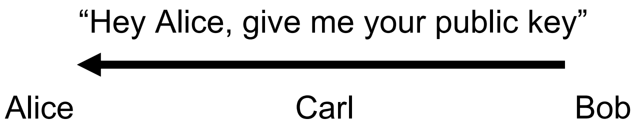
Cautions

Assumes Bob has Alice's public key, and *vice versa*

- If not, each must get it from public server
- If keys <u>*not*</u> *bound to identity of owner*, attacker Eve can launch a *man-in-the-middle* attack
  - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)

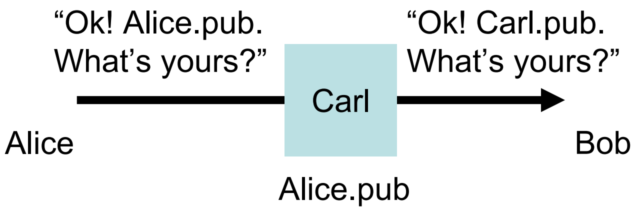


## Man In the Middle Attack (Spoofing)




SSL-Like Example

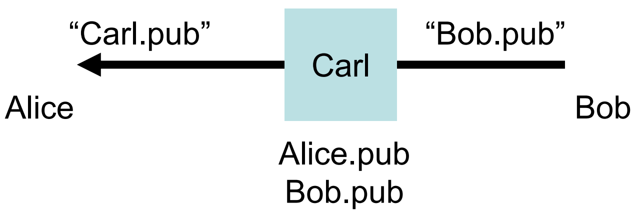
# Man In the Middle Attack (Spoofing)




**Carl takes Alice.pub and replaces is it by Carl.pub**
**Bob receives Carl.pub as if it is Alice.pub**

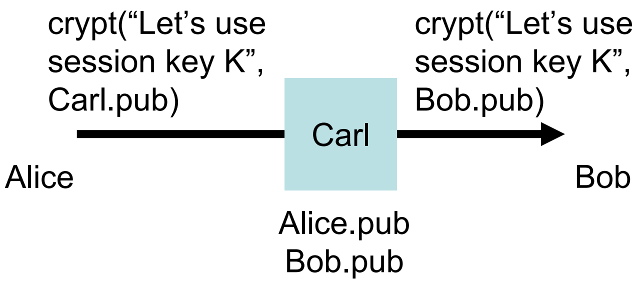


# Man In the Middle Attack (Spoofing)



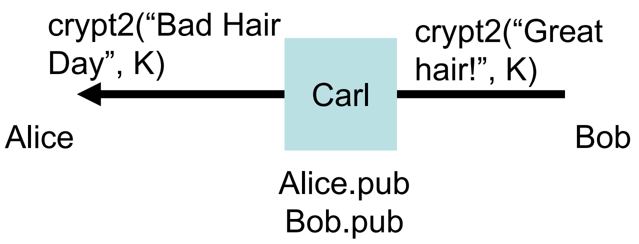

**Carl then, sends to Alice Carl.pub**
**Alice receives Carl.pub as if it is Bob.pub**

# Man In the Middle Attack (Spoofing)






# Man In the Middle Attack (Spoofing)



# Cryptographic Key Infrastructure

Goal: bind identity (Alice) to public key

Classical: not possible as all keys are shared

- Use protocols to agree on a shared key (see earlier)

Public key: bind identity to public key

- Crucial as people will use key to communicate with principal whose identity is bound to key
- Erroneous binding means no secrecy between principals
- Assume principal identified by an acceptable name



# Digital Certificates

Goal: Binding identity (Alice) to public key

Create token (message) containing

- Identity of principal (here, Alice)
- Corresponding public key
- Timestamp (when issued)
- Other information (perhaps identity of signer)

Sign it with public key of trusted authority (here, Cathy)

Simple Certificate: $C_A = \{e_A \parallel \text{Alice} \parallel T\}\, d_C$

# Use

- Bob gets Alice's certificate
  - If he knows Cathy's public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key
- Problem: Bob needs Cathy's public key to validate certificate
  - Problem pushed "up" a level
  - Two approaches: Merkle's tree, signature chains



# Certificate Signature Chains

- Create certificate
  - Generate hash of certificate
  - Encipher hash with issuer's private key
- Validate
  - Obtain issuer's public key
  - Decipher enciphered hash
  - Recompute hash from certificate and compare
- Problem: getting issuer's public key

# X.509 Chains

Issued by a Certification Authority (CA), containing:

- version (1, 2, or 3)
- serial number (unique within CA) identifying certificate
- signature algorithm identifier
- issuer X.500 name (CA)
- period of validity (from - to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2+)
- subject unique identifier (v2+)
- extension fields (v3)
- signature (of hash of all fields in certificate)

Notation CA<<A>> denotes certificate for A signed by CA



# X.509 Certificate Validation

Obtain issuer's public key

- The one for the particular signature algorithm

Decipher signature

- Gives hash of certificate

Recompute hash from certificate and compare

- If they differ, there's a problem

Check interval of validity

- This confirms that certificate is current

# Using Digital Certificates

- The (Certificate Authority) CA owns a public key and a private key
- The CA's public key is put in a self-signed certificate that is distributed through many channels (e.g embedded in browser)
- The CA use its private key to sign certificates containing identity and corresponding public key of requesters after verifying their identities
- Certificates are made available in public databases or exchanged online



# Issuers

*Certification Authority (CA)*: entity that issues certificates

- Multiple issuers pose validation problem
- Alice's CA is Cathy; Bob's CA is Don; how can Alice validate Bob's certificate?
- Have Cathy and Don cross-certify
  - Each issues certificate for the other

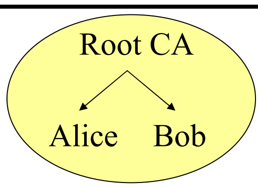
# Communicating with Certificates

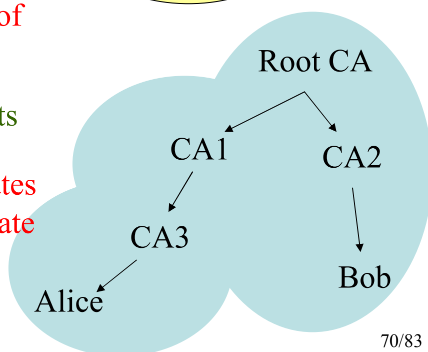
- Both Alice and Bob have the CA self-signed certificate
  - obtained through off-line means
- When Alice wants to send a message to Bob
  - She retrieves Bob's certificate from a public database
  - She verifies the CAs signature on Bobs certificate
  - She extracts Bob's public key
  - She uses the Bob's public key and her own secret key to encrypt the message
- Self-signed (root) certificates



# Certificate Hierarchy

- If both users share a common CA then they are assumed to know its public key
- Otherwise CA's must form a hierarchy
- Use certificates linking members of hierarchy to validate other CA's (cross-certify)
- Each CA has certificates for clients (forward) and parent (backward)
- Each client *trusts* parents certificates
- Enable verification of any certificate from one CA by users of all other CAs in hierarchy



# Validation and Cross-Certifying

Certificates:

- Cathy<<Alice>>
- Dan<<Bob>
- Cathy<<Dan>>
- Dan<<Cathy>>

Alice validates Bob’s certificate

- Alice obtains Cathy<<Dan>>
- Alice uses (known) public key of Cathy to validate Cathy<<Dan>>
- Alice uses Cathy<<Dan>> to validate Dan<<Bob>>



# Certificate Hierarchy

A establishes a certificate path to B:

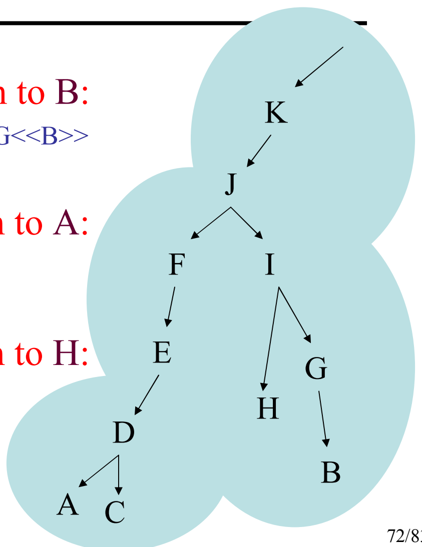
D<<E>>E<<F>>F<<J>>J<<I>>I<<G>>G<<B>>

C establishes a certificate path to A:

D<<A>>

B establishes a certificate path to H:

G<<I>>I<<H>>



# Pretty Good Privacy (PGP)

Created by Philip Zimmermann in 1991 e-mail communications

Use a bottom-up approach; instead of a top-down PKI

- Each user acts as a CA

A certificate is composed of:

- One public key packet
- Zero or more signature packets

Forms a "web of trust" among users



# Storing Keys

Multi-user or networked systems: attackers may defeat access control mechanisms

- Encipher file containing key
  - Attacker can monitor keystrokes to decipher files
  - Key will be resident in memory that attacker may be able to read
- Use physical devices like "smart card"
  - Key never enters system
  - Card can be stolen, so have 2 devices combine bits to make single key

## Key Revocation

- Certificates invalidated *before* expiration
  - Usually due to compromised key
  - May be due to change in circumstance (*e.g.*, someone leaving company)
- Problems
  - Entity revoking certificate authorized to do so
  - Revocation information circulates to everyone fast enough
    - Network delays, infrastructure problems may delay information
- *CRL - Certificate revocation list*
  - lists certificates that are revoked (no longer valid)



## Digital Signature - Again

- Construct that authenticated origin, contents of message in a manner provable to a disinterested third party ("judge")
- Sender cannot deny having sent message (service is "nonrepudiation")
  - Limited to *technical* proofs
    - Inability to deny one's cryptographic key was used to sign
  - One could claim the cryptographic key was stolen or compromised
    - Legal proofs, *etc.*, probably required; not dealt with here

# Common Error

Classical: Alice, Bob share key $k$

– Alice sends $m \,||\, \{ m \}\, k$ to Bob

This is a digital signature

***WRONG***

**This is not a digital signature**

– Why? Third party cannot determine whether Alice or Bob generated message



# Classical Digital Signatures

Require trusted *third party*

– Alice, Bob each share keys with trusted party Cathy

To resolve dispute, judge gets $\{ m \}\, k_{Alice}$, $\{ m \}\, k_{Bob}$, and has Cathy decipher them; if messages matched, contract was signed

Alice → $\{ m \}k_{Alice}$ → Bob

Cathy ← $\{ m \}k_{Alice}$ ← Bob

Cathy → $\{ m \}k_{Bob}$ → Bob

# Public Key Digital Signatures

Alice's keys are $d_{Alice}$, $e_{Alice}$

Alice sends Bob

$$m \parallel \{ m \} d_{Alice}$$

In case of dispute, judge computes

$$\{ \{ m \} d_{Alice} \} e_{Alice}$$

and if it is $m$, Alice signed message

- She's the only one who knows $d_{Alice}$!



# RSA Digital Signatures

Use private key to encipher message

- Protocol for use is *critical*

Key points:

- Never sign random documents, and when signing, always sign hash and never document
  - Mathematical properties can be turned against signer
- Sign message first, then encipher
  - Changing public keys causes forgery

# Attack #1

Example: Alice, Bob communicating

- $n_A = 95$, $e_A = 59$, $d_A = 11$
- $n_B = 77$, $e_B = 53$, $d_B = 17$

26 contracts, numbered 00 to 25

- Alice has Bob sign 05 and 17:
  - $c = m^{d_B} \bmod n_B = 05^{17} \bmod 77 = 3$
  - $c = m^{d_B} \bmod n_B = 17^{17} \bmod 77 = 19$
- Alice computes $05 \times 17 \bmod 77 = 08$; corresponding signature is $03 \times 19 \bmod 77 = 57$; claims Bob signed 08
- Judge computes $c^{e_B} \bmod n_B = 57^{53} \bmod 77 = 08$
  - Signature validated; Bob is toast



# Attack #2: Bob's Revenge & Payback

Bob, Alice agree to sign contract 06

Alice enciphers, then signs:

$(m^{e_B} \bmod 77)^{d_A} \bmod n_A = (06^{53} \bmod 77)^{11} \bmod 95 = 63$

Bob now changes his public key

- Computes $r$ such that $13^r \bmod 77 = 6$; say, $r = 59$
- Computes $re_B \bmod \phi(n_B) = 59 \times 53 \bmod 60 = 7$
- Replace public key $e_B$ with 7, private key $d_B = 43$

Bob claims contract was 13. Judge computes:

- $(63^{59} \bmod 95)^{43} \bmod 77 = 13$
- Verified; now Alice is toast

# Key Points

- Key management critical to effective use of cryptosystems
  - Different levels of keys (session *vs*. interchange)
- Keys need infrastructure to identify holders, allow revoking
  - Key escrowing complicates infrastructure
- Digital signatures provide integrity of origin and content
  - Much easier with public key cryptosystems than with classical cryptosystems