Distributed Systems

Unit-9: Security

Compiled by Prashant Gautam

Why Security in Distributed Systems?

- Distributed Systems rely on sharing of resources across different networked entities

- Most vital/secret data handled by distributed components

- A single security flaw compromises the whole system
  - Malware and viruses can spread from one part of the system to another easily
  - Users across the world may have access to the system
    - Cyber criminals, hackers

- Security lapses result in
    - Loss of confidence, Claims for damages, Loss of privacy

Overview

Introduction

- Threats, policies and mechanisms
- Cryptography

Secure Channels

- Authentication
- Message Integrity and Confidentiality

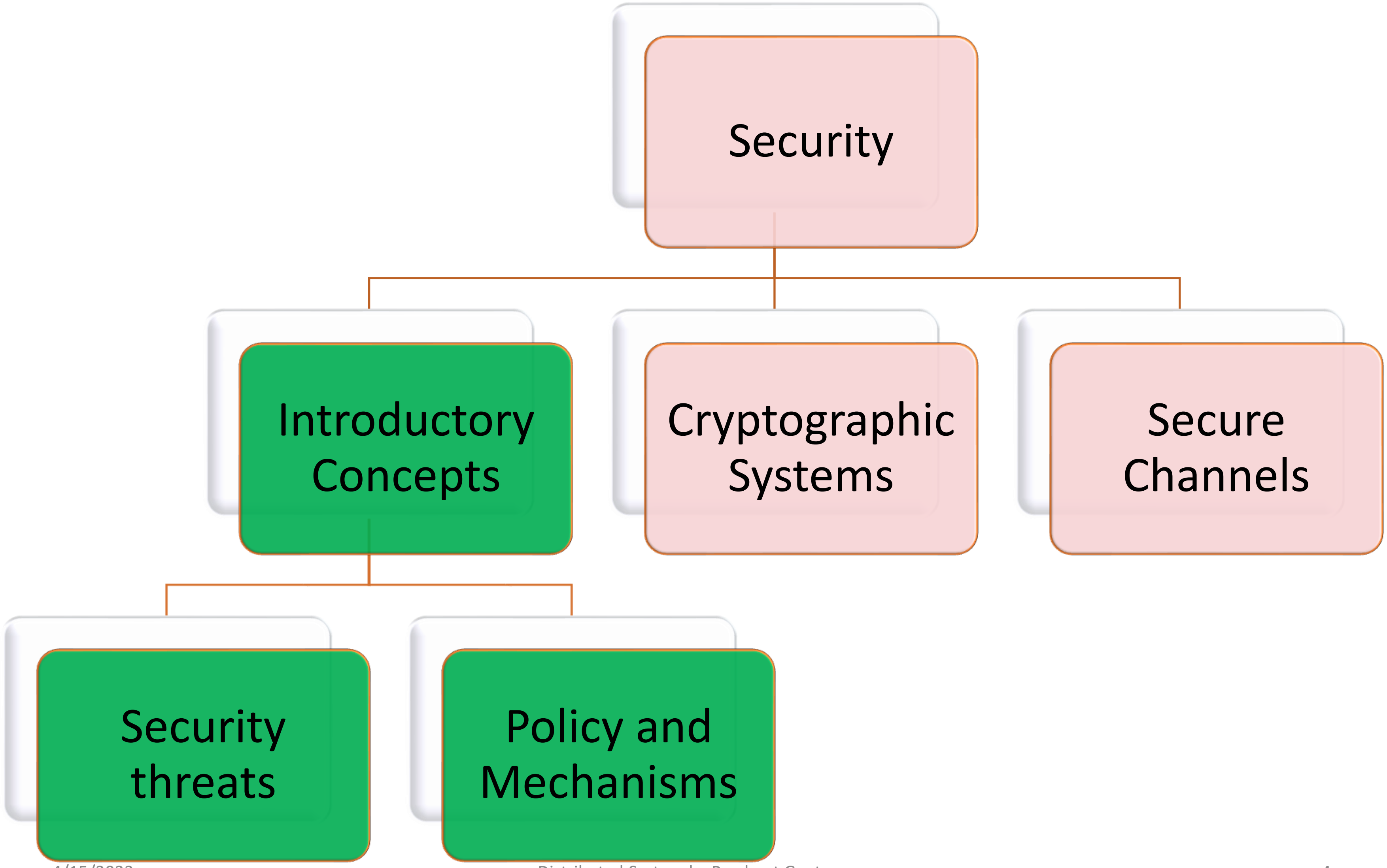
Access Control

- Access Control Matrix
- Protection Domains

Security Management

- Key Management
- Authorization Management

Overview


Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Security threats
Policy and Mechanisms

Introduction to Security

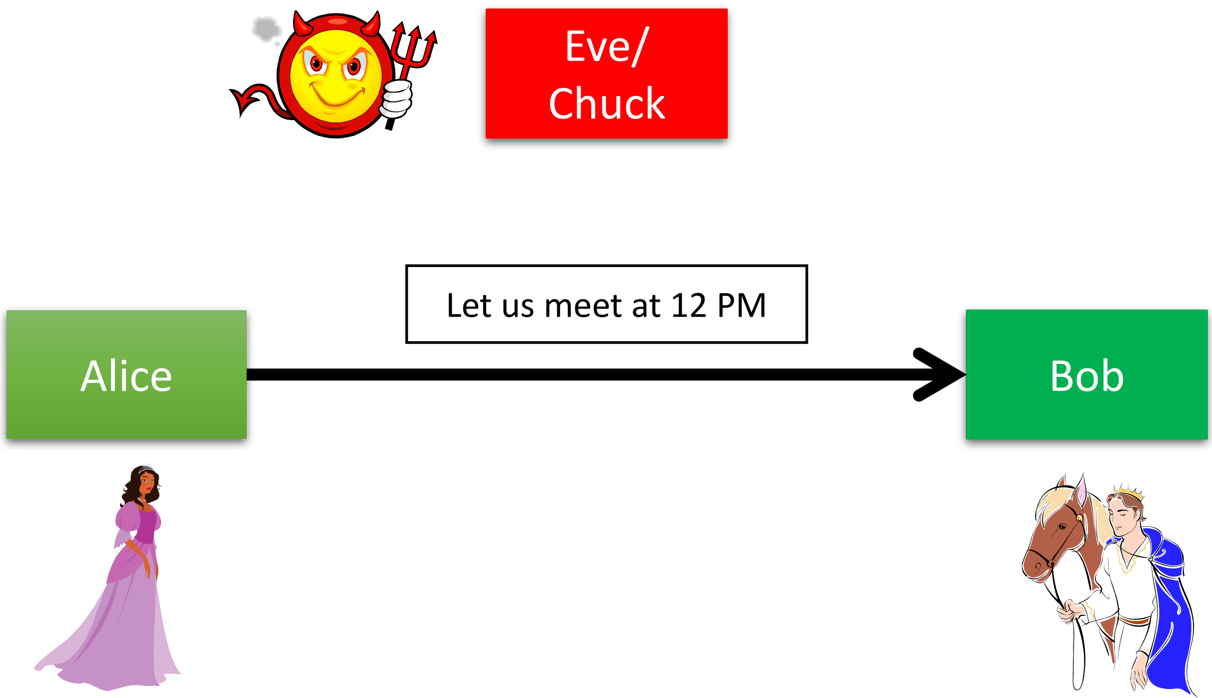
- What services do you expect from secure Distributed Systems?

- Secure DS should provide
  - Confidentiality of Information
    - Information is disclosed only to authorized parties
  - Integrity of Information
    - Alterations to system's assets is made only in an authorized way

Secure DS are immune against possible security threats that compromise confidentiality and integrity

Security Threats

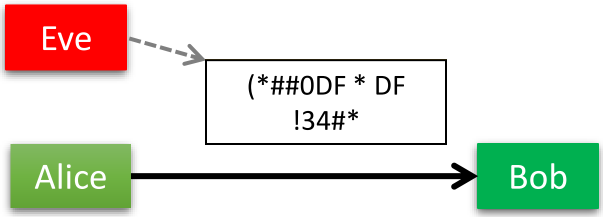
- What are the security threats when two entities communicate?



Types of Security Threats (1)

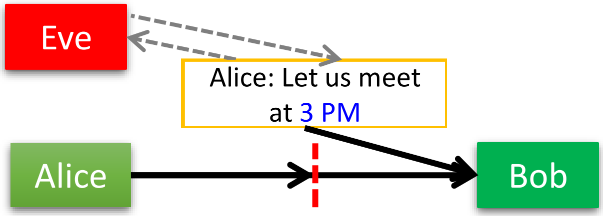
1. Interception
   - Unauthorized party has gained access to a service or data
   - Example: Illegal copying of files, Eavesdropping over network

2. Interruption
   - Services or data become unavailable, unusable or destroyed
   - Example: Denial-of-Service (DoS) Attacks



Types of Security Threats (2)

3. **Modification**
   - Unauthorized changing of data or tampering with services
   - Example: Changing a program to secretly log the activities



4. **Fabrication**
   - Additional data or activity is generated that would normally not exist
   - Example: Replay attacks

Security Policy and Mechanisms

- To build a secure DS, we need to

Formulate Security Policies → **Build Security Mechanisms**

Formulate **Security Policies**	Build **Security Mechanisms**
• Specifies what is to be done	• Specifies how policies are implemented
Describe the security requirements: • which actions the entities are allowed to take • which actions are prohibited	*Implement* mechanisms to: • Protect data transferred • Verify identity of an entity • Secure access permissions

Security Mechanisms

Four core components of security mechanisms

Cryptographic Algorithms and Secure Channels

1. Encryption
   - Transform the data to something that attacker cannot understand

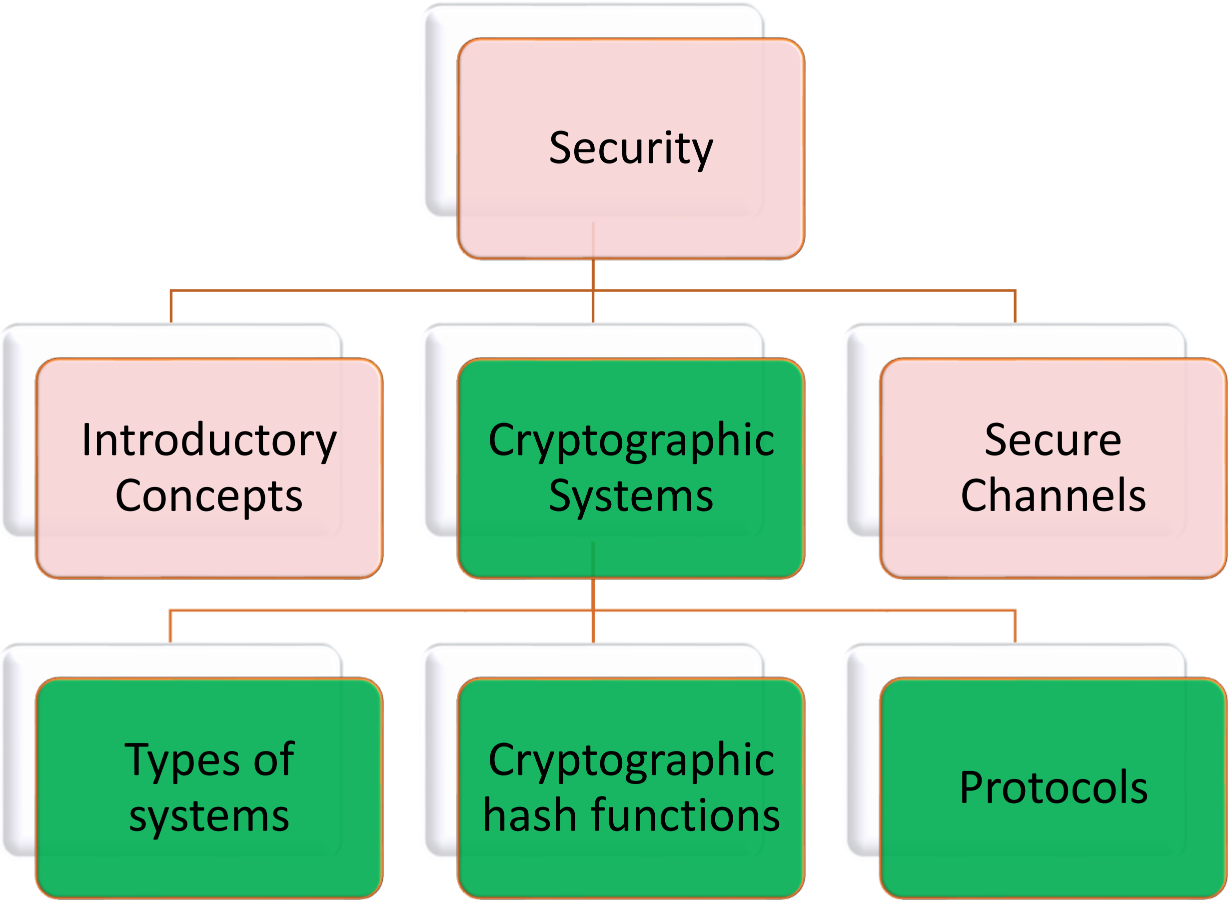
2. Authentication
   - Verifies the claimed identity of the user, host or other entity

Access Control

3. Authorization
   - Verifies if the entity is authorized to perform an operation

4. Auditing
   - To trace which clients accessed what, and which way

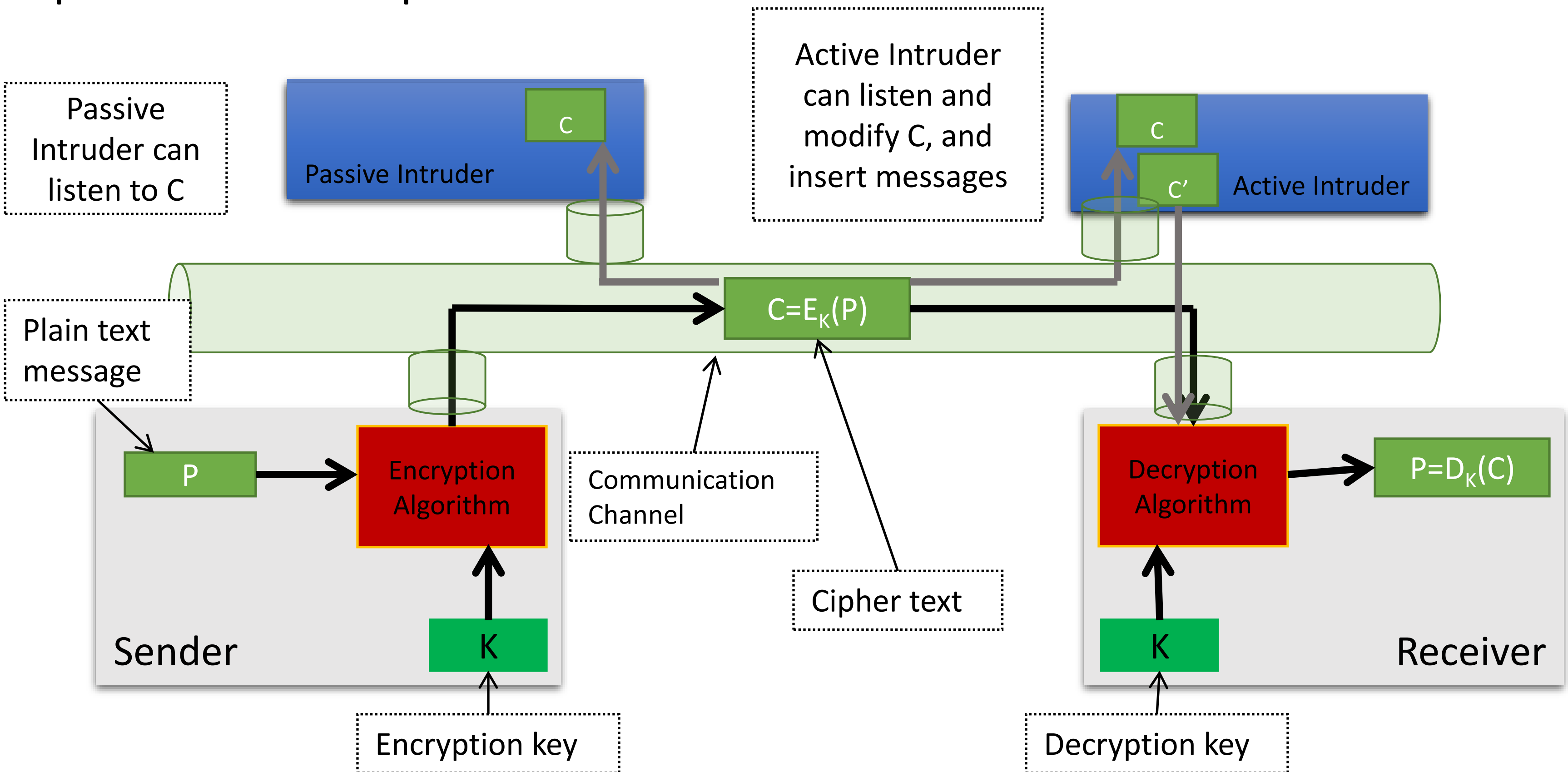
Overview



Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Types of systems
Cryptographic hash functions
Protocols

Cryptographic systems

- Cryptography is the study of techniques for secure communication in the presence of third parties



$C=E_K(P)$ ➔ C is obtained by encrypting the plain text P with key K
$P=D_K(C)$ ➔ P is obtained by decrypting the cipher text P with key K

Types of Cryptographic Systems

- Two types
  1. Symmetric Cryptosystem (Shared-key system)
  2. Asymmetric Cryptosystem (Public-key system)

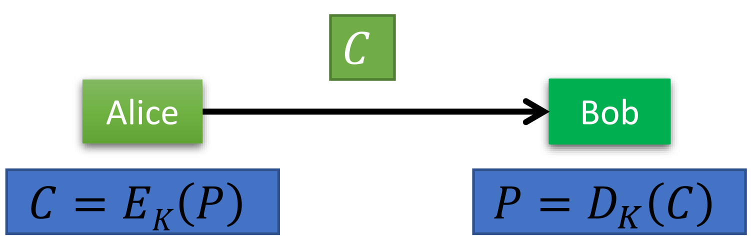
Symmetric Cryptographic System

- Same key is used to encrypt and decrypt the data

$$P = D_K\left(E_K(P)\right)$$

- The shared key $K_{A,B}$ between Alice A and Bob B should be kept secret

- Also known as *secret-key* or *shared-key* systems



Public-key Cryptographic System

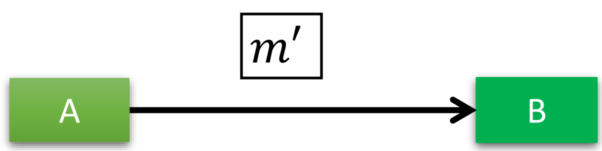
- The key for encryption (K_E) and decryption(K_D) are different
- But, K_E and K_D form a unique-pair

$$P = D_{K_D}\left(E_{K_E}(P)\right)$$

- One of keys is made public, and another made private
- Denotation: Public key of A = K_A^+ ; Private key of A = K_A^-
- Public-key systems can be used for:
  - Encryption
  - Authentication

Encryption in Public-key system

- Scenario: Alice (A) wants to send to Bob (B)
- Problem: Only Bob should be able to decrypt the message
- Approach:
  - At A: Encrypt using B's public key
  - At B: Decrypt using B's private key



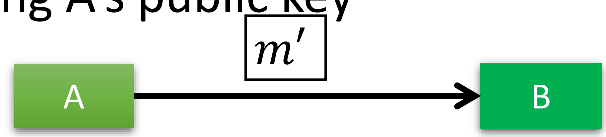
$$m' = E_{K_B^+}(m)$$

$$m = D_{K_B^-}(m')$$

Drawback: How does 'B' know that 'A' sent the message?

Authentication in Public-key system

- Scenario: Alice (A) wants to send a message to Bob (B)
- Problem: Bob wants to make sure that message came from Alice (and not from some intruder)
- Approach:
  - At A: Encrypt using A's private key
  - At B: Decrypt using A's public key



$$m' = E_{K_A^-}(m)$$

$$m = D_{K_A^+}(m')$$

Drawback: How to ensure that ONLY 'B' gets the message?

Combining encryption and authorization

- Scenario: Alice (A) wants to send a message to Bob (B)

- Many algorithms use a combination of the above two methods to:
  - Ensure that only Bob can decrypt the message
  - Bob can verify that Alice has sent the messages

$$A \xrightarrow{\boxed{m''}} B$$

$$m' = E_{K_A^-}(m) \qquad m' = D_{K_B^-}(m'')$$

$$m'' = E_{K_B^+}(m') \qquad m = D_{K_A^+}(m')$$

- Approach: Encrypt/Decrypt using a combination of keys
  - A widely used method in many secure algorithms

• What happens if intruder 'C' modifies message 'm' sent by 'A'?
Some part of message 'm' should contain data that is verifies the message

Cryptographic Hash Functions

- A hash function $H(m)$ maps an input message m to a hash value h
  - Message m is of any arbitrary length
  - Hash h is fixed length

$$h = H(m)$$

- Often, h is called as the "*message digest*" of m

- How does a cryptographic hash function differ from a regular hash function?

Properties of Cryptographic Hash Functions

1.One-Way Function

- Finding hash h from m is easy, but not vice-versa
- Computationally infeasible to find m that corresponds to a given hash h

2.Weak-collision resistance

- Given a message, it is hard to find another message that has the same hash value
- Given m and $h = H(m)$, it is computationally infeasible to find $m' \neq m$ such that $H(m) = H(m')$

3.Strong-collision resistance

- Given a hash function, it is hard to find two messages with the same hash value
- Given $H(.)$, it is computationally infeasible to find two messages m and m' such that $H(m) = H(m')$

Encryption/Decryption Functions

- Recall: An encryption function $E_K(m_p)$ encrypts a plain-text message m_p to an cipher-text message m_c using a key K

$$m_c = E_K(m_p)$$

- Similarly, decryption function $D_K(m_c)$:

$$m_p = D_K(m_c)$$

- Encryption/Decryption Functions have the same properties as Cryptographic Hash Functions
  - One-way functions
  - Weak and Strong collision resistance

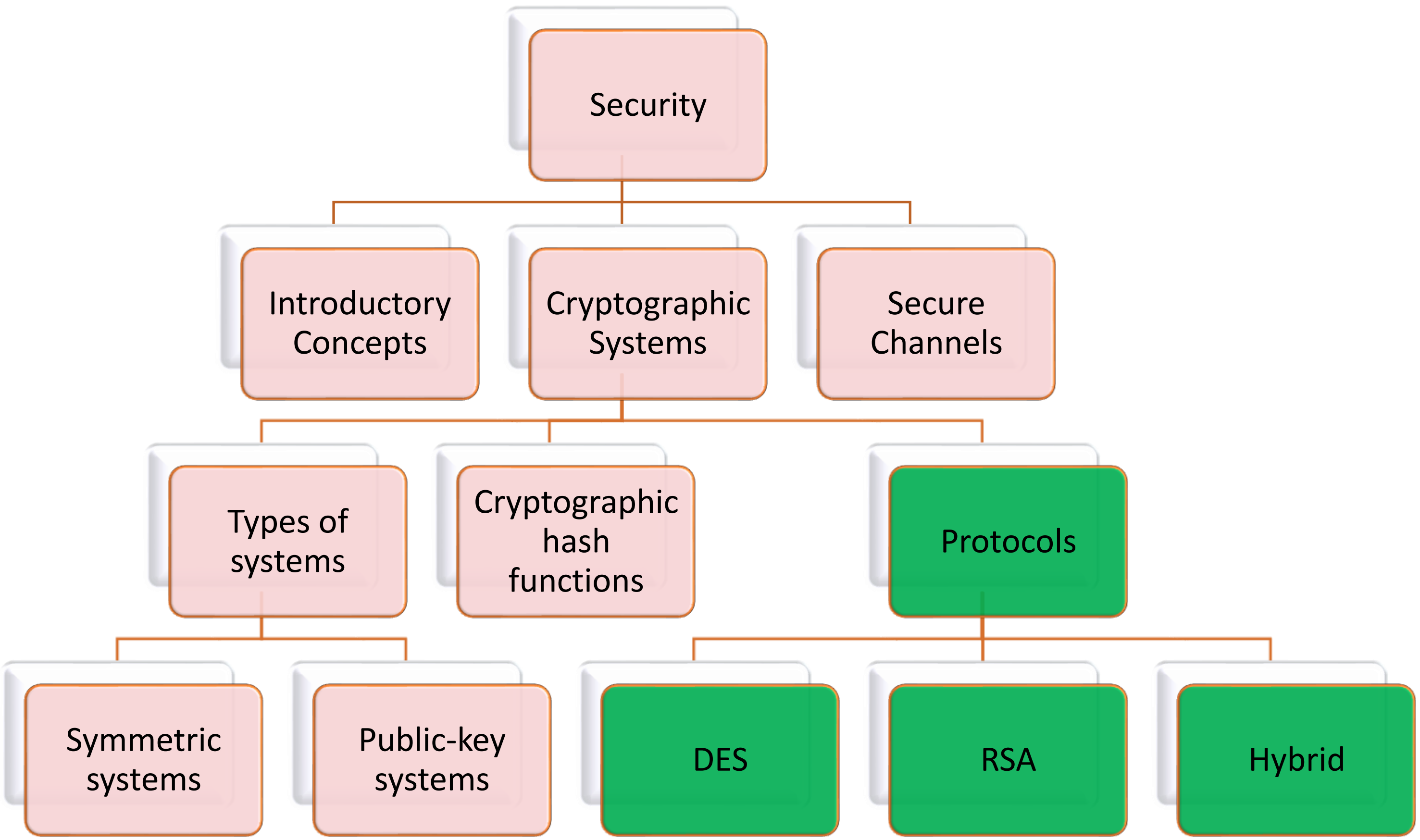
Additional Properties of Encryption/Decryption Functions

- Infeasible Key Extraction *
  - Given a plain-text m_p and its cipher-text m_c, it is hard to find the key K such that $m_c = E_K(m_p)$

- Key Collision Resistance *
  - Given a plain-text and a key K, it is hard to find another key K' that produces the same cipher-text
  - Given a plain-text m_p and a key K such that cipher-text $m_c = E_K(m_p)$, it is hard to find another key K' that produces the same cipher-text m_c

* Property names given are meant for understanding, and is not widely used in the community

Overview



Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Types of systems
Cryptographic hash functions
Protocols
Symmetric systems
Public-key systems
DES
RSA
Hybrid

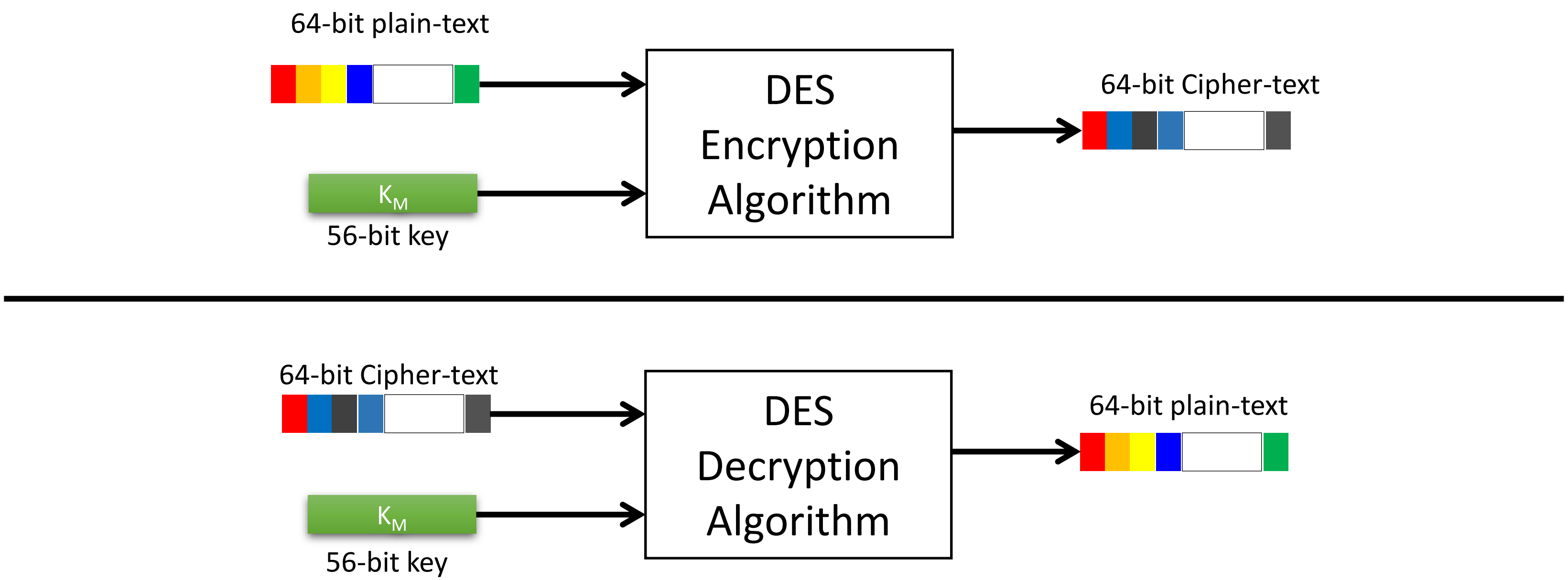
Protocols in Cryptosystems

- We will study three protocols in cryptosystems
  - Data Encryption Standard (DES)
    - Encryption/Decryption in Symmetric Cryptosystems
  - RSA protocol
    - Encryption/Decryption in Public-Key cryptosystems
  - Hybrid Cryptographic protocol
    - A combination of Symmetric and Public-Key based system

DES Protocol

- Purpose: Encryption/Decryption in Symmetric Cryptosystems

- Encryption and Decryption relies on a 56-bit master key (K_M)
- Operates on a 64-bit block of input data to encrypt/decrypt



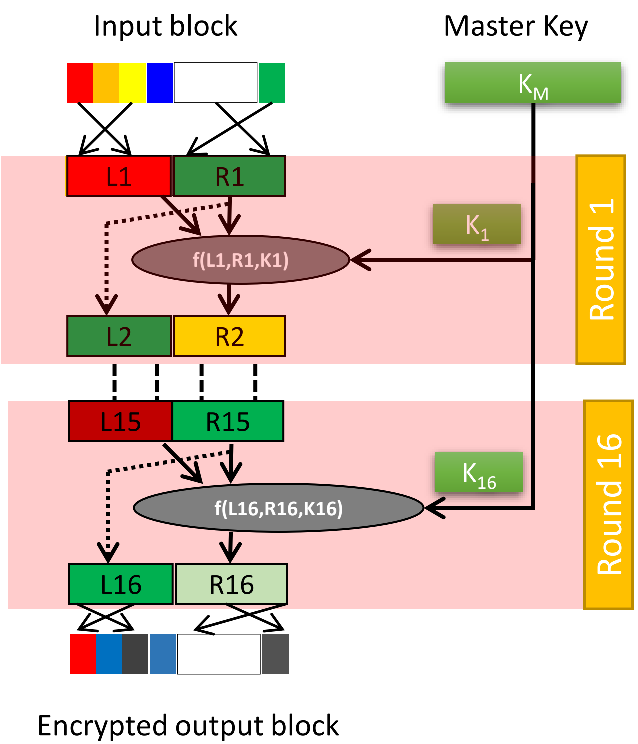
DES Encryption Algorithm

1. Permute a 64-bit block

2. 16 rounds of identical operations

3. In each round *i*
   i. Divide the block into 2 halves L_i and R_i
   ii. Extract 48-bit key K_i from K_M
   iii. Mangle the bits in L_i and R_i using K_i to produce R_{i+1}
   iv. Extract R_i as L_{i+1}

4. Perform an inverse permutation on the block L_{16}-R_{16} to produce the encrypted output block



Discussion about DES

- DES encryption and decryption is relatively fast
- DES has disadvantages of a Symmetric Cryptosystem
  - Requires sender and receiver to exchange K_M
  - For N-user system, DES needs N(N-1)/2 master key pairs
- History of DES:
  - DES was invented in 1974
  - In 1997, it was shown that DES can be easily cracked using brute-force attacks
- Triple-DES is in use in some systems
  - It applies DES three times using two keys

RSA protocol

- Invented by Rivest, Shamir and Adleman (RSA) as a protocol for Public-key systems

- Approach:
  1. Choose two very large prime numbers, p and q
  2. Compute n = pq
  3. Compute z = (p-1)(q-1)
  4. Choose a number e that is relatively prime to z
     - e is co-prime to z
  5. Compute the number d such that de % z = 1
     - This is equivalent to finding de = 1 + kz for some integer k

Example*
p=7; q=19
n = 7*19 = 133
z = 6*18 = 108
e=5
d=65 for m=325

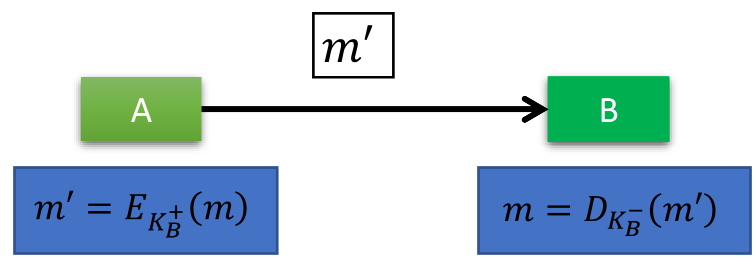
- Depending on the requirement, d and e can be used as public and private keys
  - d is used for decryption; e is used for encryption

* The numbers chosen in the example are for illustration. Prime numbers with 100s of digits are chosen in the actual RSA algorithm

Example: RSA protocol for encryption (1)

- Scenario: Alice (A) wants to send to Bob (B)
- Problem: Only Bob should be able to decrypt the message
- Given d and e are the two keys computed by RSA, which row indicates correct choice of keys?

Correct/ Incorrect	Alice		Bob	
	Private K_A^-	Public K_A^+	Private K_B^-	Public K_B^+
x	d	e		
x	e	d		
Correct			d	e
x			e	d



Example: RSA protocol for encryption (2)

At the sender:

- Split the message into fixed-length blocks of size s ($0 <= s < n$)
- For each block m_i
  - Sender calculates the encrypted message c_i such that $c_i = m_i^e \pmod n$
  - Send c_i to the receiver

At the receiver:

- Receive c_i from sender
- For each block c_i
  - Compute actual message $m_i = c_i^d \pmod n$
- Merge all c_i's to obtain the complete message

Calculated Values

$p=7$; $q=19$; $n=133$; $d=65$; $e=5$

Example
$s = 132$
$m_i = 6$
$c_i = 6^5 \pmod{133}$ $= 62$
$m_i = 62^{65} \pmod{133}$ $= 6$

Discussion about RSA

- RSA has advantages of Public-key system
  - Harder to break the code
  - For a N-user system, RSA needs only 2N keys
- Computation time of RSA is much larger than DES
  - Approximately 100-1000 times slower

Hybrid Cryptographic protocols

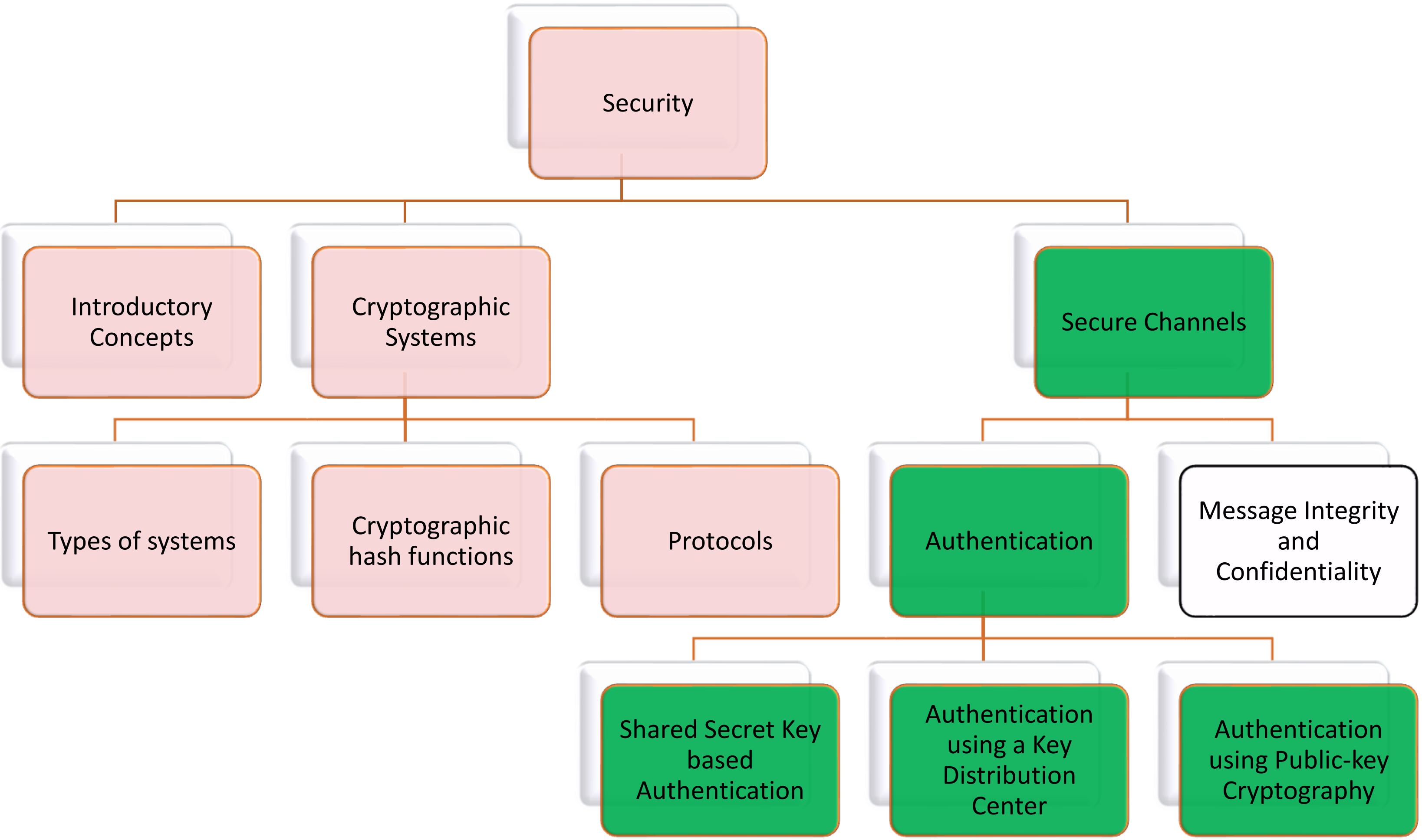
- Large scale distributed systems use a combination of symmetric and public-key protocols

- Leveraging the advantages of both schemes
  - Encryption based on Public-key are more secure
    - Authenticate using RSA
    - Exchange the “secret key” using RSA for a session
  - Encryption based on Symmetric keys are faster
    - Exchange large data using the above “secret key”

Beyond Cryptographic Mechanisms

- Many users, clients and servers need to dynamically send messages in a distributed system
  - How can an end-to-end secure distributed system be built using the cryptographic mechanisms?
  - How can each message and user be protected against security threats?
  - How do clients and processes authenticate?
  - What protocols are needed for these? What is their complexity?

Overview



Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Types of systems
Cryptographic hash functions
Protocols
Authentication
Message Integrity and Confidentiality
Shared Secret Key based Authentication
Authentication using a Key Distribution Center
Authentication using Public-key Cryptography

Secure Channels

- A *Secure Channel* is an abstraction of secure communication between communication parties in a DS

- A Secure Channel protects senders and receivers against:
  - Interception
    - By ensuring *confidentiality* of the sender and receiver

  - Modification and Fabrication of messages
    - By providing *mutual authentication* and *message integrity* protocols

- We will study
  - Authentication
  - Confidentiality and Message Integrity

Authentication

- Consider a scenario where Alice wants to set up a secure channel with Bob
  - Alice sends a message to Bob (or trusted third party) for mutual authentication
  - Message integrity should be ensured for all communication between Alice and Bob
    - Generate a “session key” to be used between Alice and Bob
    - Session-keys ensure integrity and confidentiality
  - When the channel is closed, the session key is destroyed

Types of Mutual Authentication Protocols

1. Shared Secret Key based Authentication
2. Authentication using a Key Distribution Center
3. Authentication using Public-key Cryptography

Types of Mutual Authentication Protocols

1. Shared Secret Key based Authentication
2. Authentication using a Key Distribution Center
3. Authentication using Public-key Cryptography

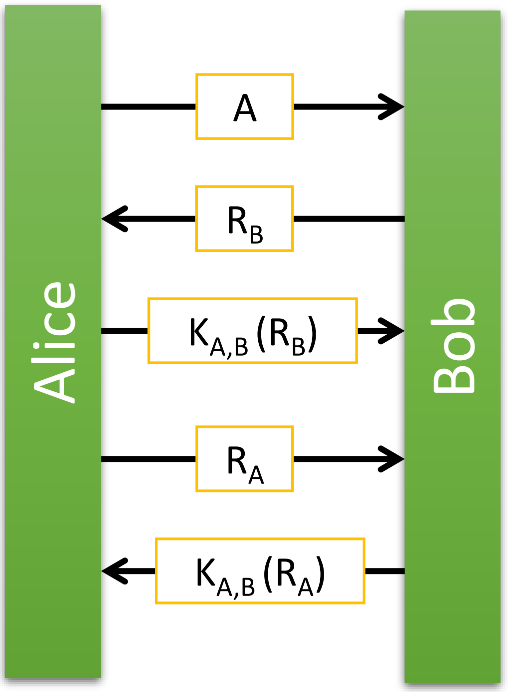
Shared Secret Key based Authentication

- The scheme is also known as "*Challenge-Response protocol*"
- Let $K_{A,B}$ be the shared secret key between Alice and Bob

The Challenge-Response Protocol

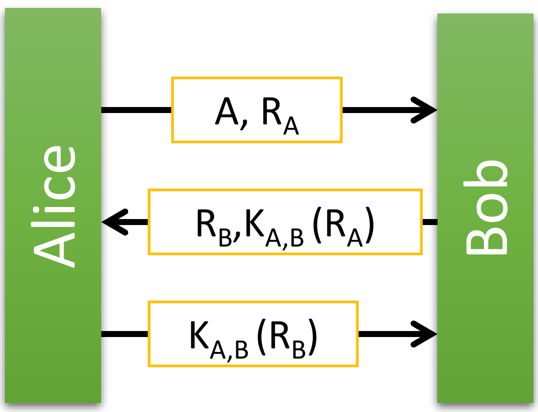
1. 'A' sends her identity to 'B'
2. 'B' sends a challenge R_B back to 'A'
3. 'A' responds to the challenge by encrypting R_B with $K_{A,B}$ (denoted by $K_{A,B}(R_B)$), and sending it back to 'B'
4. 'A' challenges 'B' by sending R_A
5. 'B' responds to the challenge by sending the encrypted message $K_{A,B}(R_A)$

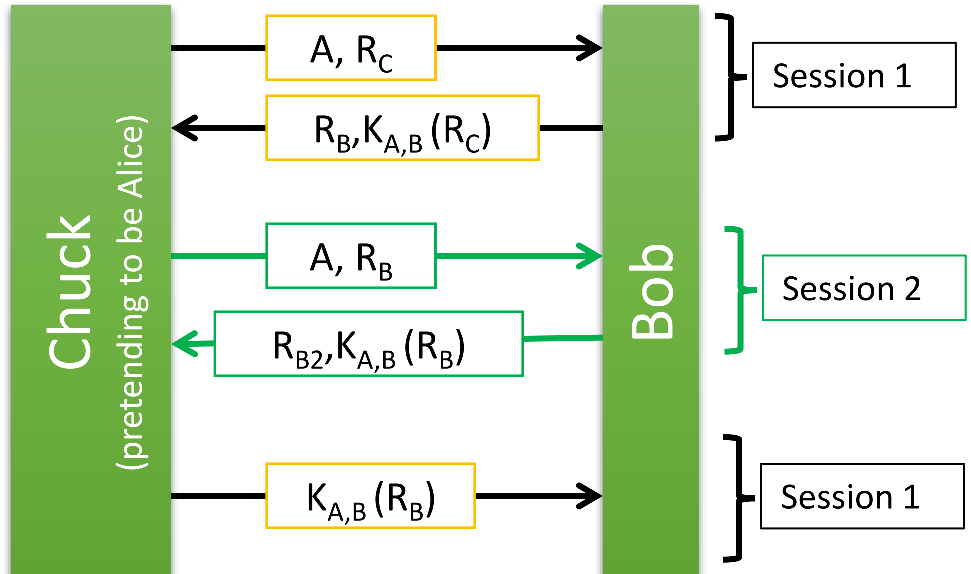
A and B are mutually authenticated



A Possible Optimization (?)

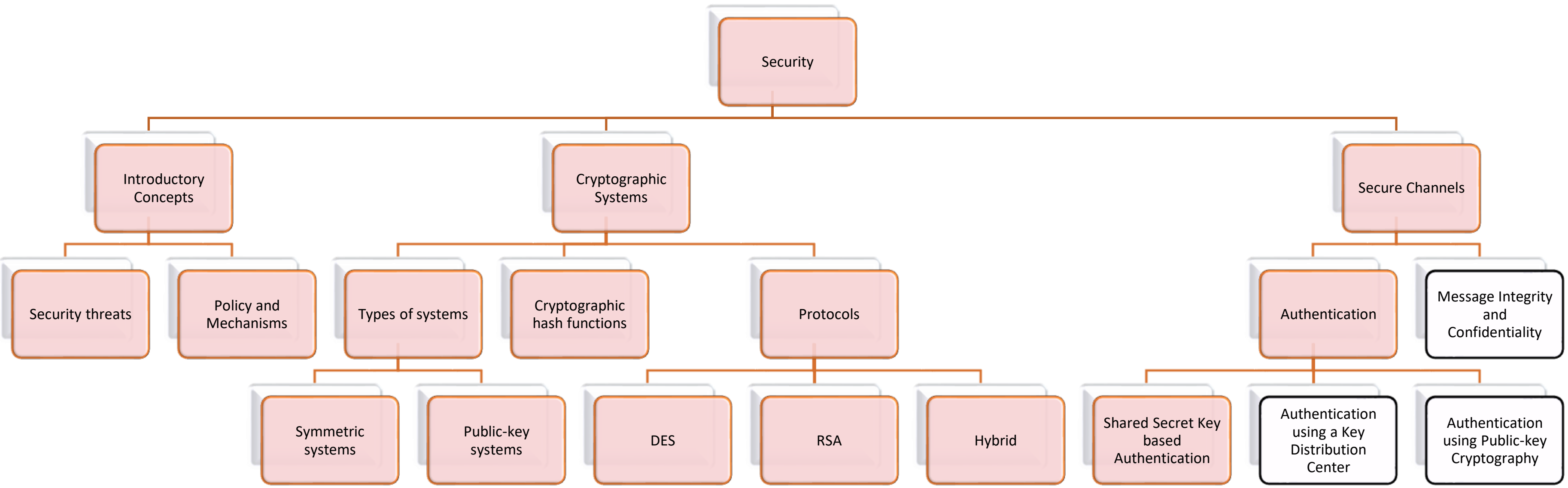
- Will the below 3-step protocol work?





Reflection Attack

Summary



Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Security threats
Policy and Mechanisms
Types of systems
Cryptographic hash functions
Protocols
Authentication
Message Integrity and Confidentiality
Symmetric systems
Public-key systems
DES
RSA
Hybrid
Shared Secret Key based Authentication
Authentication using a Key Distribution Center
Authentication using Public-key Cryptography

Overview

Introduction

- Threats, policies and mechanisms
- Cryptography

Secure Channels

- Authentication
- Message Integrity and Confidentiality

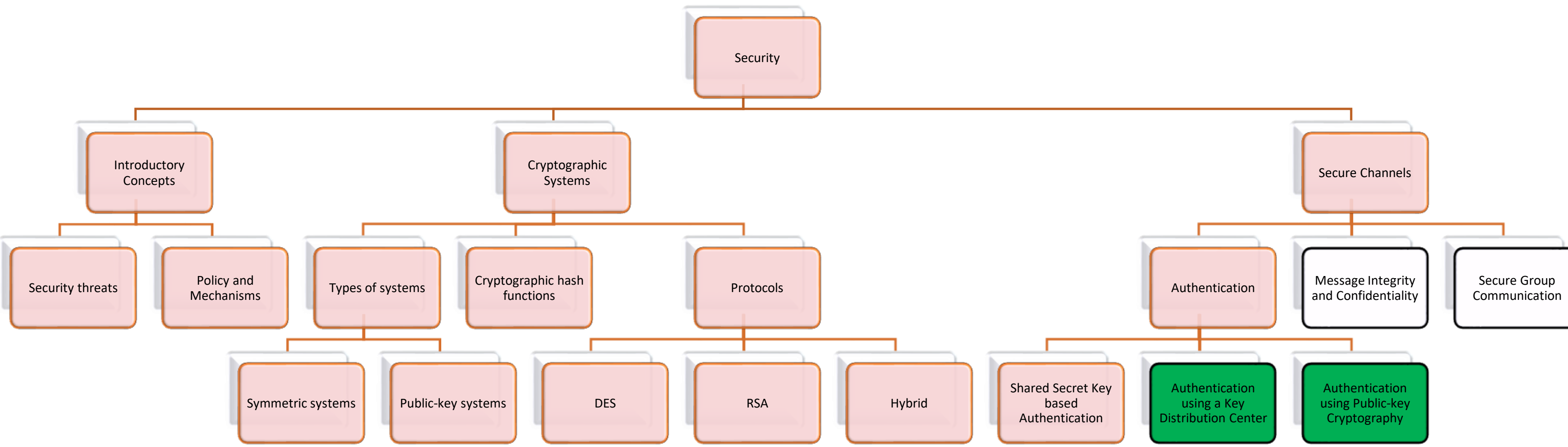
Access Control

- Access Control Matrix
- Protection Domains

Security Management

- Key Management
- Authorization Management

Overview


Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Security threats
Policy and Mechanisms
Types of systems
Cryptographic hash functions
Protocols
Authentication
Message Integrity and Confidentiality
Secure Group Communication
Symmetric systems
Public-key systems
DES
RSA
Hybrid
Shared Secret Key based Authentication
Authentication using a Key Distribution Center
Authentication using Public-key Cryptography

Types of Mutual Authentication Protocols

1. Shared Secret Key based Authentication
2. Authentication using a Key Distribution Center
3. Authentication using Public-key Cryptography

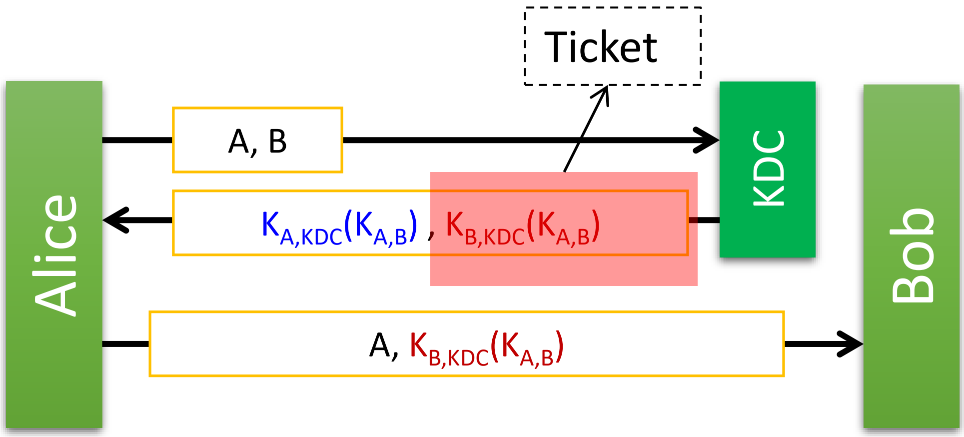
Types of Mutual Authentication Protocols

1. Shared Secret Key based Authentication
2. Authentication Using a Key Distribution Center
3. Authentication Using Public-key Cryptography

Authentication Using a Key Distribution Center

- Shared secret key based authentication is not scalable
  - Each host has exchange keys with every other host
    - Complexity: $O(N^2)$
- The complexity is reduced by electing one node as "*Key Distribution Center*" (KDC)
  - KDC shares a secret-key with every host
  - No pair of hosts need to share a key
  - Complexity: O(N)

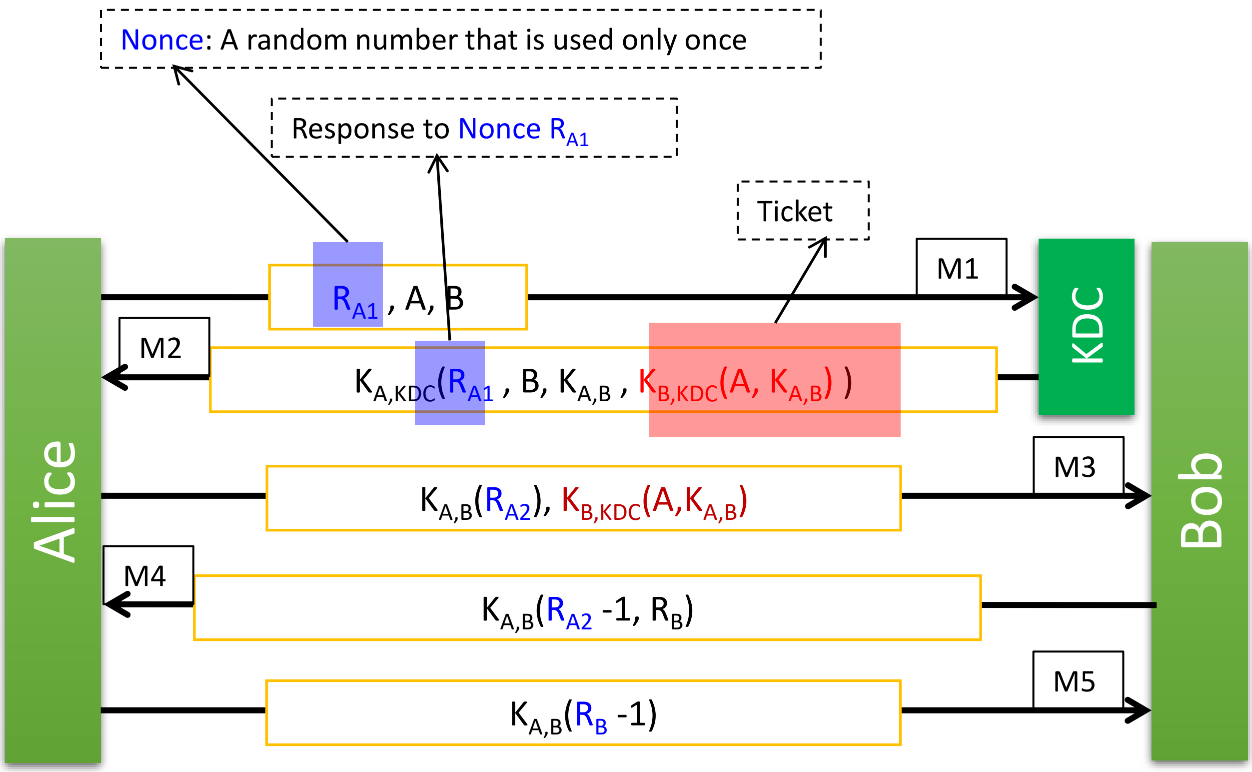
Basic Model of a KDC based Authentication



- We will study a widely used KDC protocol called Needham-Schroeder Authentication Protocol

Needham-Schroeder Authentication Protocol

- A multi-way challenge response protocol



What happens if:

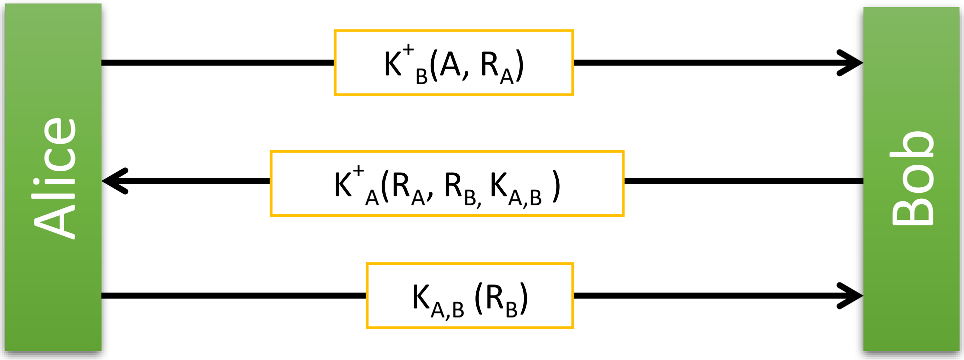
1. Nonce is not included in M2?
2. B is not included in M2?
3. R_{A2} is returned in M4, instead of $(R_{A2}-1)$?
4. Chuck has an old key $K_{A,B}$ and intercepts M3, and replays M3 at a later point of time to Alice?

Types of Mutual Authentication Protocols

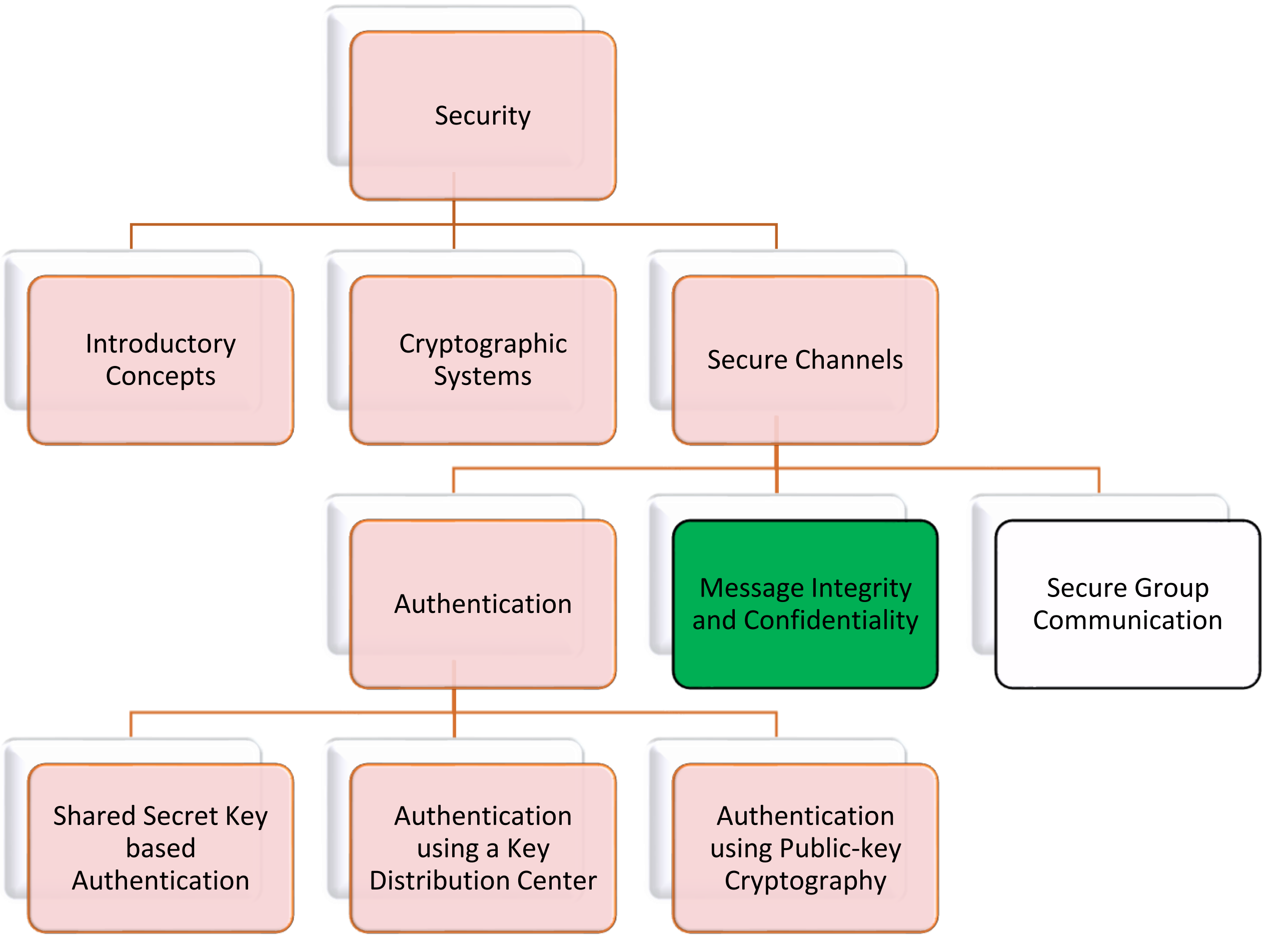
1. Shared Secret Key based Authentication
2. Authentication using a Key Distribution Center
3. Authentication using Public-key Cryptography

Authentication Using a Public-Key Cryptography

- Recall:
  - K^{+}_{N}: Public key of user N
  - K^{-}_{N}: Private key of user N



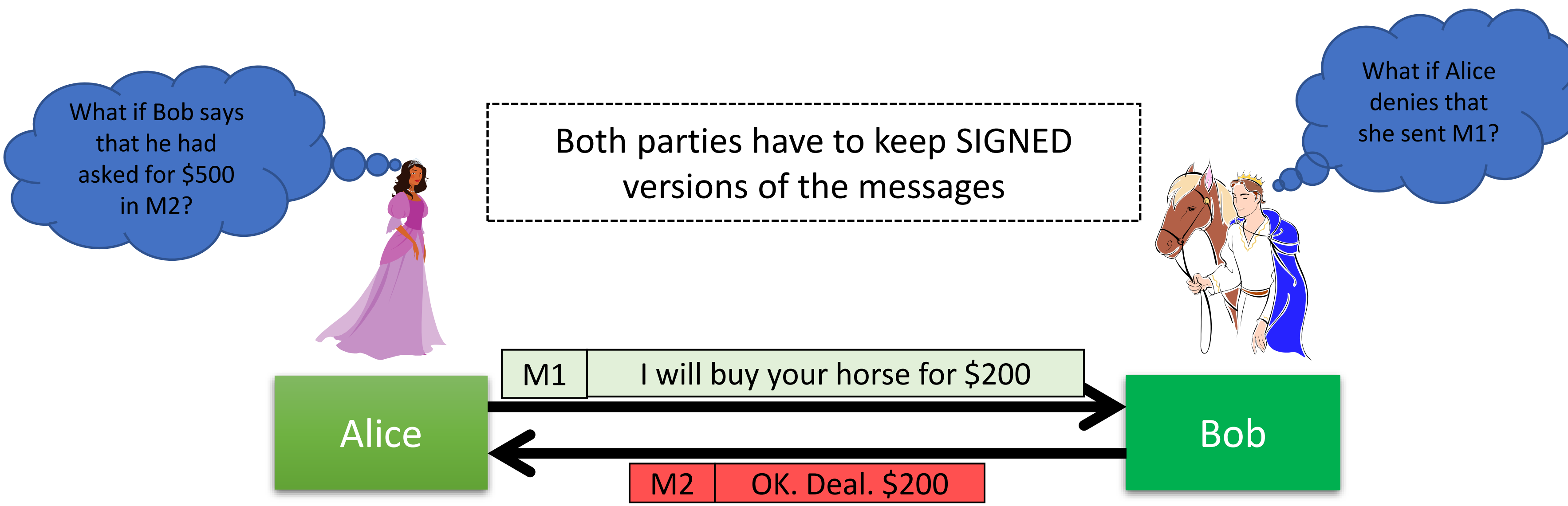
Overview



Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Authentication
Message Integrity and Confidentiality
Secure Group Communication
Shared Secret Key based Authentication
Authentication using a Key Distribution Center
Authentication using Public-key Cryptography

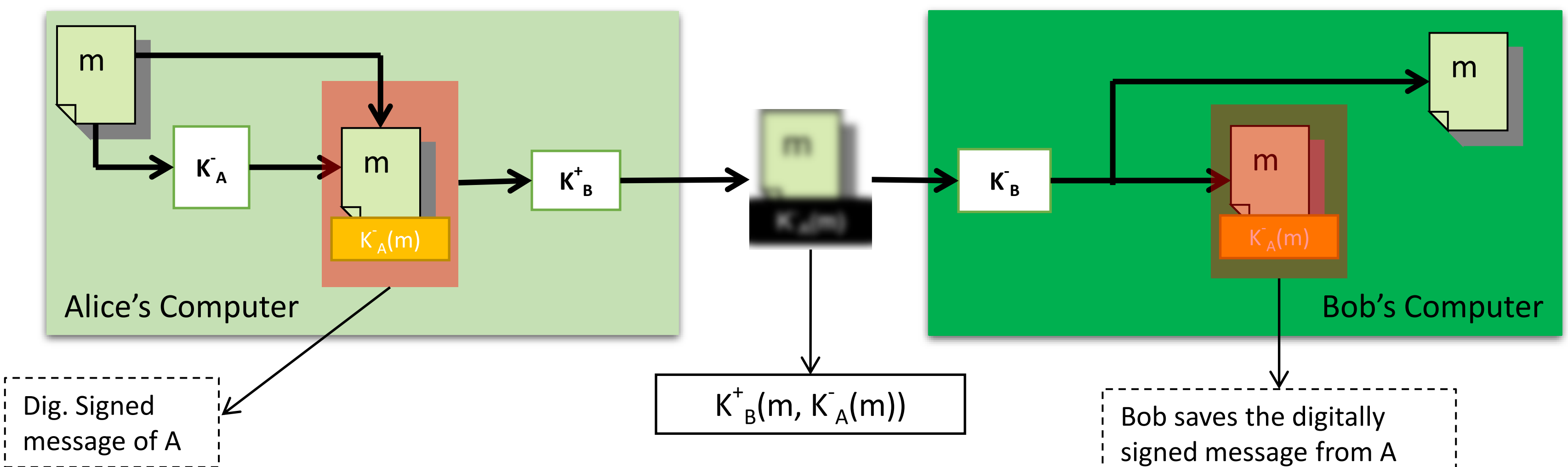
Message Integrity and Confidentiality

- Encryption is used for providing Confidentiality
- But, how to provide Message Integrity?
  - Encryption protects message modification by third party adversaries
  - How to protect modification at senders or receivers?



Digital Signatures

- The sender and the receiver will keep ***digitally signed*** copies of the messages
- One way of creating digital signatures is by using RSA

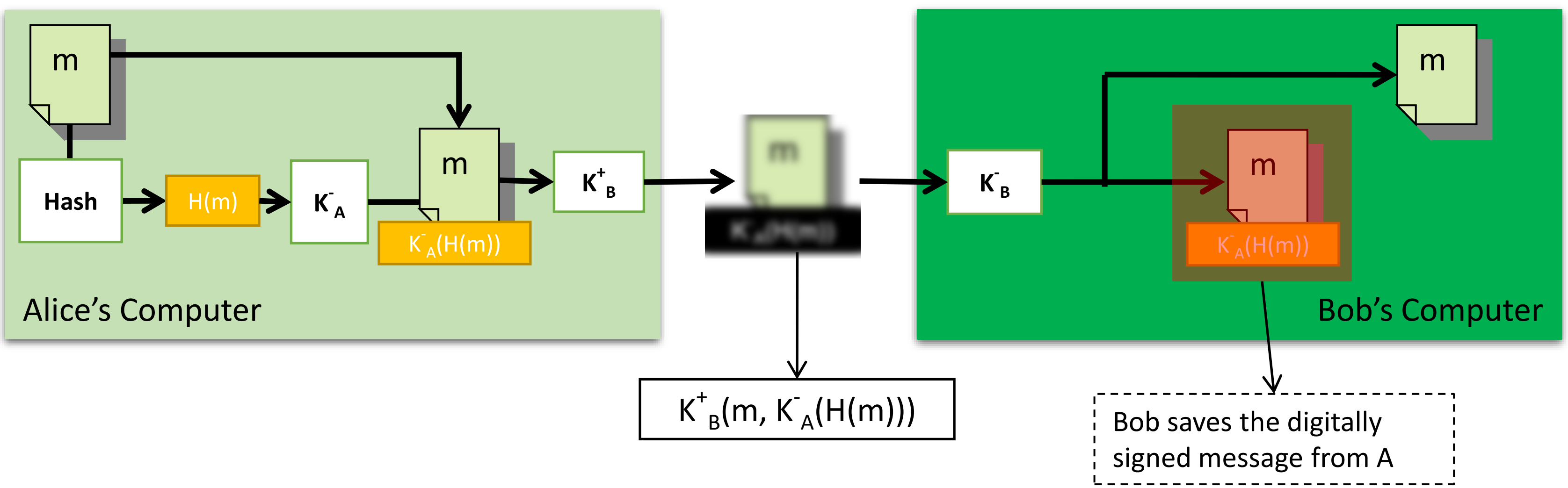


Issues with Naïve Digital Signatures

- The validity of the digitally signed message holds as long as A's private key remains a secret
  - 'A' can claim that her private key was stolen
- If 'A' changes her private key, the signed message becomes invalid
  - A centralized authority can keep track of when keys are changed
- Encrypting the entire message with private key is costly
  - RSA is slower, and encryption of long data is not preferred
  - Any ideas on how to improve?

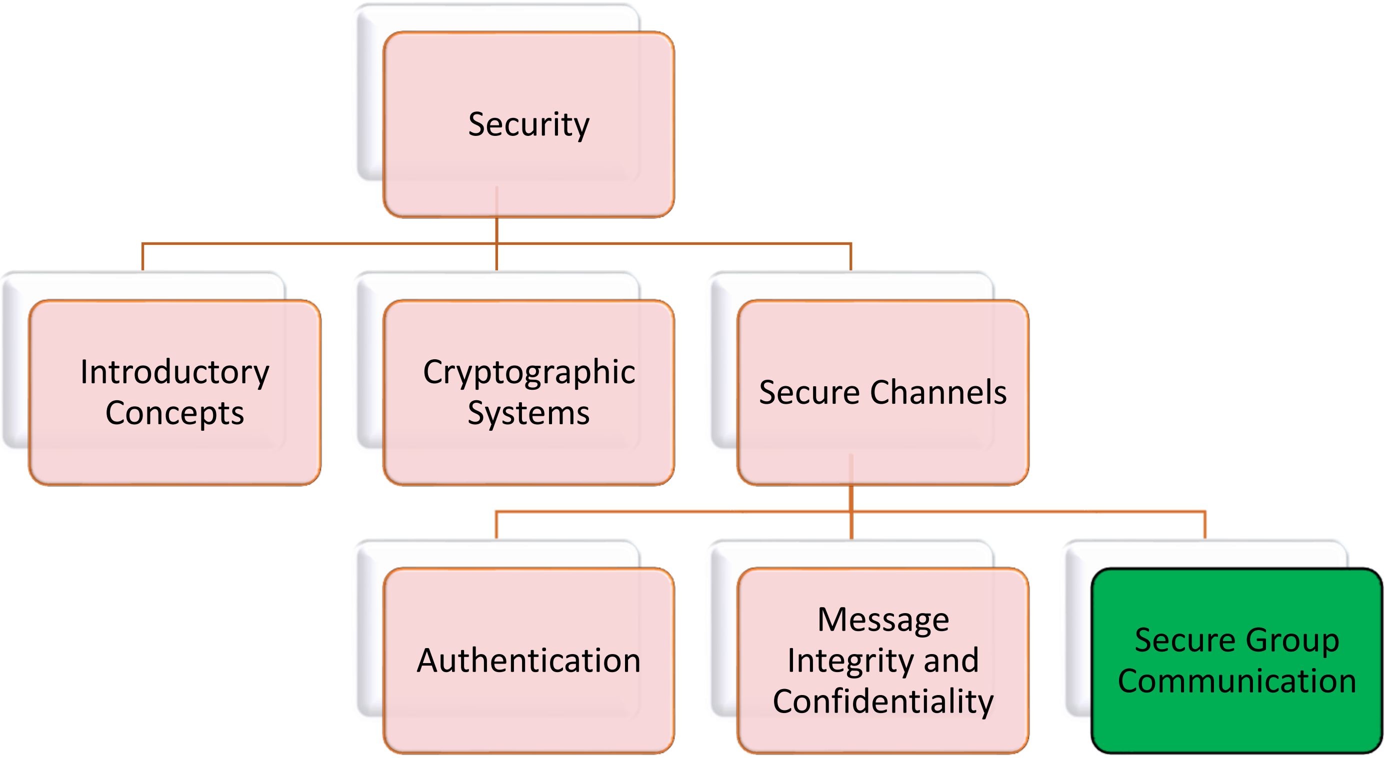
Digital Signature with Message Digest

- Instead of attaching the encrypted message, attach the *message digest*



Bob can verify the signature by comparing the hash of received message, and the hash attached in the digitally signed message

Overview



Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Authentication
Message Integrity and Confidentiality
Secure Group Communication

Secure Group Communication

- Scenario: Client has to communicate with a group of replicated servers
  - Replication can be for fault-tolerance or performance improvement
- Client expects that the response is secure
- Naïve Solution:
  - Client collects the response from each server
  - Client authenticates every response
  - Client takes a majority vote
- Drawback of naïve approach
  - Replicated of servers should be transparent to the client
  - Reiter et al. proposed a transparent and secure group authentication protocol (Secure Group Authentication Protocol)

Secure Group Authentication Protocol

- Problem: Authenticate a group of replicated servers at the client in a transparent way

- Main Idea:
  - Multiple servers share a secret
  - None of them know the entire secret
  - Secret can be revealed only if all/most of them cooperate

- The approach is similar to k-fault tolerant systems
  - Tolerate intrusion (faults) of c servers out of N servers if c <= N

- We will study the algorithm in two phases:
  1. Generating Shared Secret Signatures
  2. Enhancing transparency

1. Generating Shared Secret Signatures

- For each client call:
  - Each server S_i returns the response (r_i)
  - In addition, it sends a signed message digest: $sig(S_i,r_i) = K^-_{s_i}(md(r_i))$
- For each response, client computes

  $md_c(r_i) = H(r_i)$

- Hence, client has N triplets

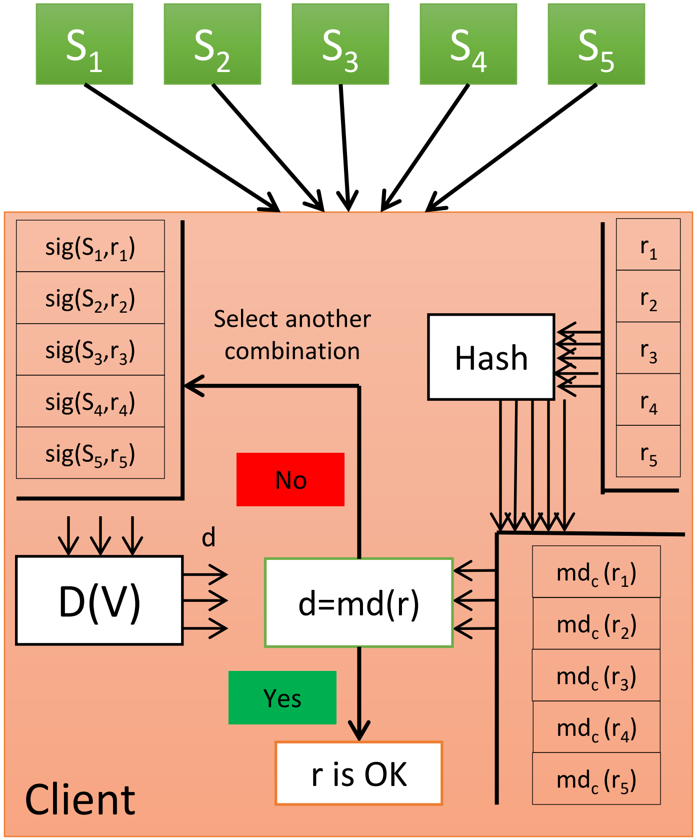
  $R_i = < r_i, md_c(r_i), sig(S_i,r_i)>$

- Client takes c+1 combinations of triplets
- For each combination vector [R, R' and R'']
  - Computes a create a single digest using a special decryption function D(V)

    d = D(V) = D(sig(S,r), sig(S',r'),sig(S'',r''))

    NOTE: d is a vector of responses
  - If d[x] = respective $md_c(r_i)$, then r_i is correct

Parameters: N=5; c=2



2. Enhancing transparency

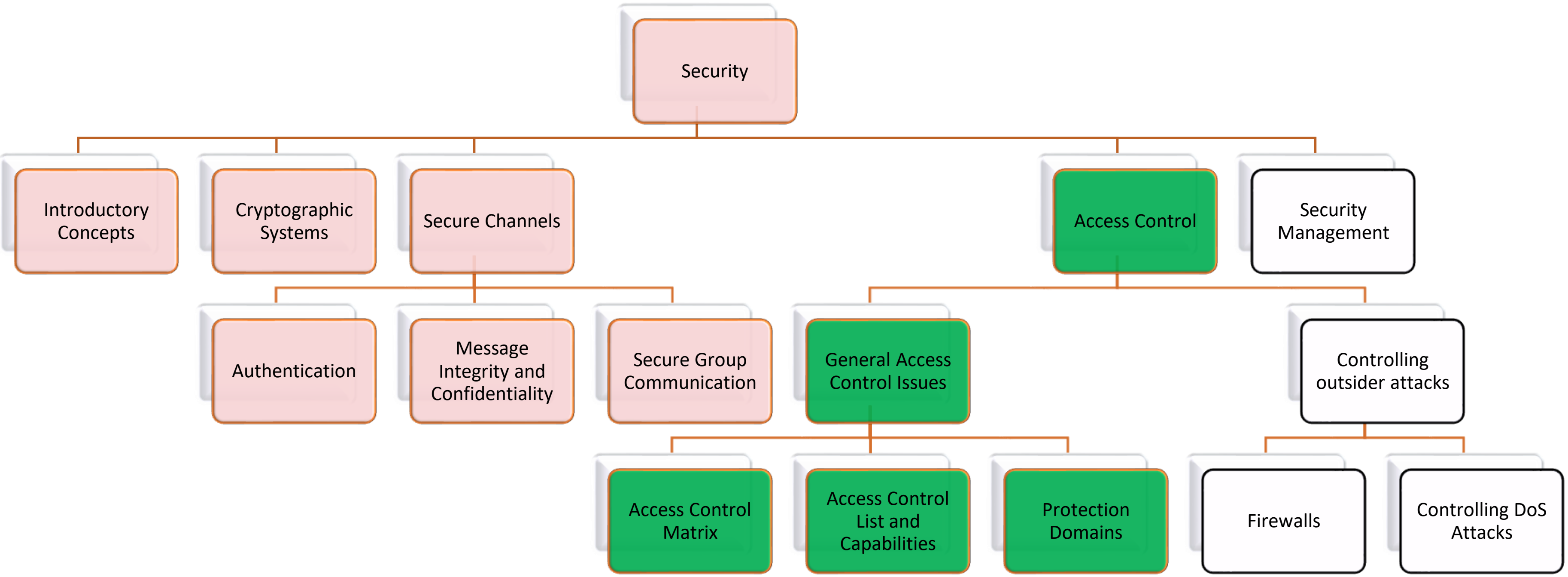
- Each server broadcasts a message $[r_{i,} sig(S_i, r_i)]$ to other servers
- When the server has received k+1 messages
  - Compute valid signature D(V) for one combination of responses r
  - If success, then transmit [V,r] to the client
- When client receives a response, it re-verifies the correctness of r

Discussion: Secure Group Authentication Protocol

- The protocol can be generalized to (m,n)-threshold scheme
  - A message is divided into n-pieces (shadows)
  - m shadows can be used to reconstruct the message
  - (m-1) or fewer shadows cannot reconstruct

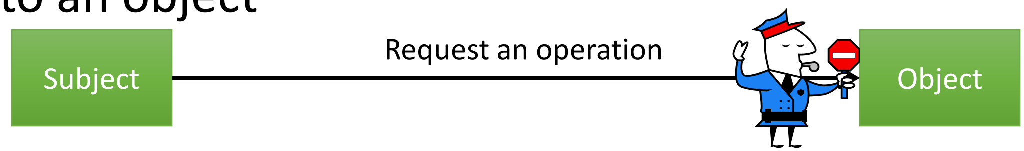
- In the above protocol:
  - m = c+1 = Number of valid signatures required
  - n = N = Number of replicated servers

Overview

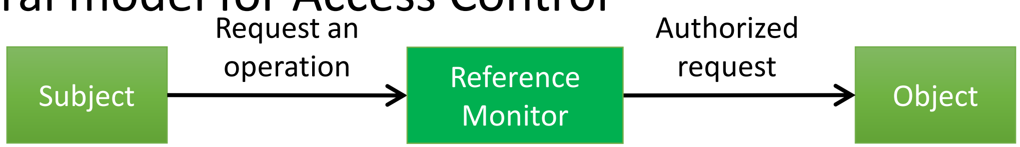

Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Access Control
Security Management
Authentication
Message Integrity and Confidentiality
Secure Group Communication
General Access Control Issues
Controlling outsider attacks
Access Control Matrix
Access Control List and Capabilities
Protection Domains
Firewalls
Controlling DoS Attacks

Access Control

- Access Control (AC) verifies the access rights of a subject requesting an access to an object



- A General model for Access Control

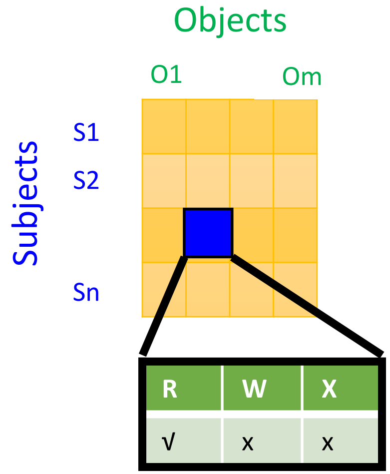


  - A reference monitor records which subjects may do what

- How is Access Control different from Authorization?
  - Authorization = granting access rights
  - Access Control = verifying access rights

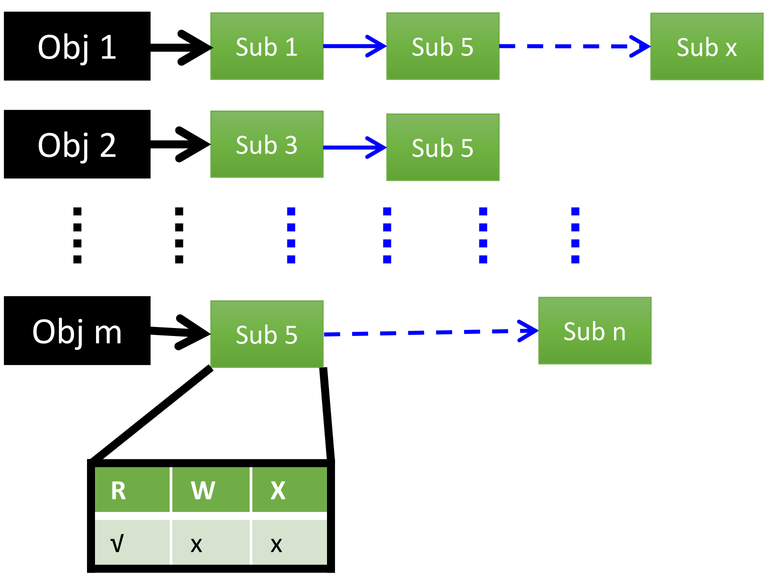
Access Control Matrix

- A matrix M that keeps track of which operations can a subject invoke on an object
  - M [s,o] lists which operations subject s can perform on object o
- The reference monitor will look up AC Matrix before authorizing the request
- Scalability of AC Matrix
  - For a large DS with many users, most of the entries are empty
- Two implementations of AC Matrix:
  1. Access Control Lists
  2. Capabilities



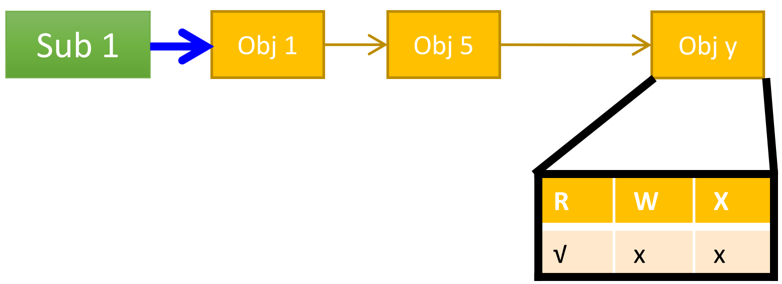
1. Access Control List

- AC Matrix is stored as a list
  - Similar to how a graph of nodes and edges are stored as an adjacency list



2. Capabilities

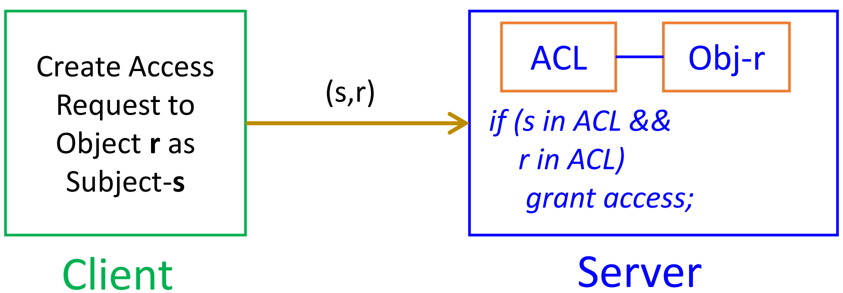
- AC Matrix can also be decomposed row-wise
- ***Capability*** for a user is a list of objects and the associated permissions for a given user



- The 'capability' object can be issued by Reference Monitor to a user, and can reside at the user side
  - But should be protected (digital signature)

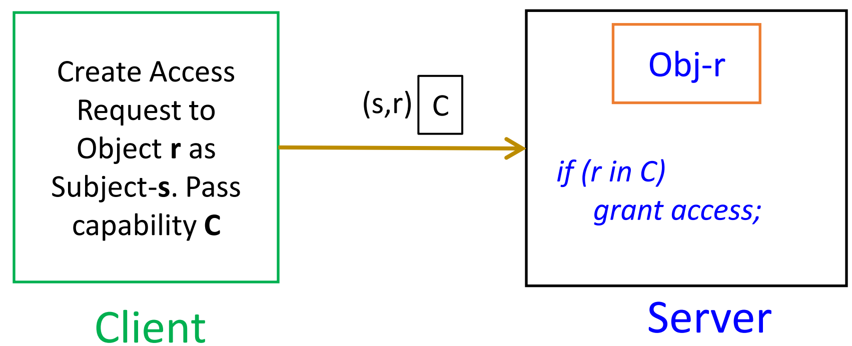
Comparison of ACL and Capabilities

Using ACL



The server stores the ACL

Using Capabilities



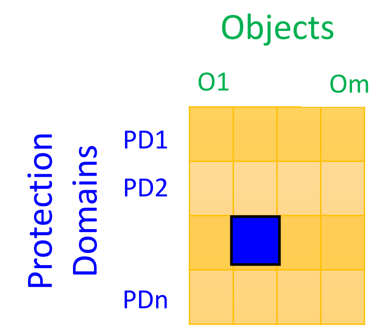
- The server is not interested in whether it knows the client
- Server should check Capability

Protection Domains

- Motivation: Access Control using ACL and Capabilities is still not scalable in very large DS

- Organizing Permissions:
  - Object permission is given to a protection domain (PD)
  - Each PD row consists of (object, permission) pair

- User Management:
  - Group uses into a “protection domain”
  - Users belong to one or more protection domain
    - e.g., employees, admin, guests
  - Before granting the access, check permissions from user’s PDs



PD: Admin

R	W	X
√	√	√

PD: Employees

R	W	X
√	x	√

PD: Guests

R	W	X
x	x	x

A matrix is shown only for representation. Similar to AC Matrix, PD can be implemented using techniques similar to ACL

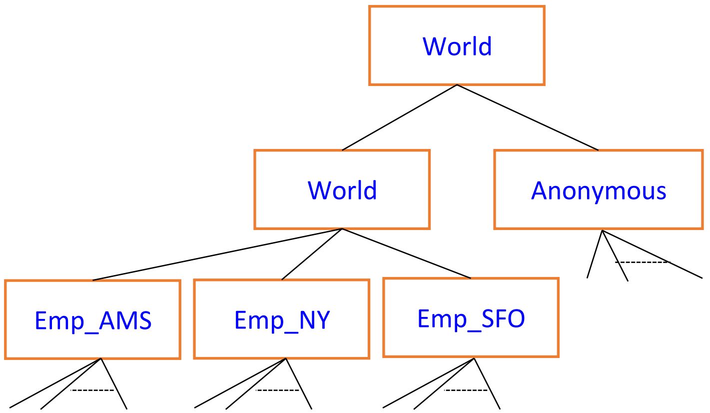
Hierarchical Protection Domains

- Domains can be organized into hierarchical groups

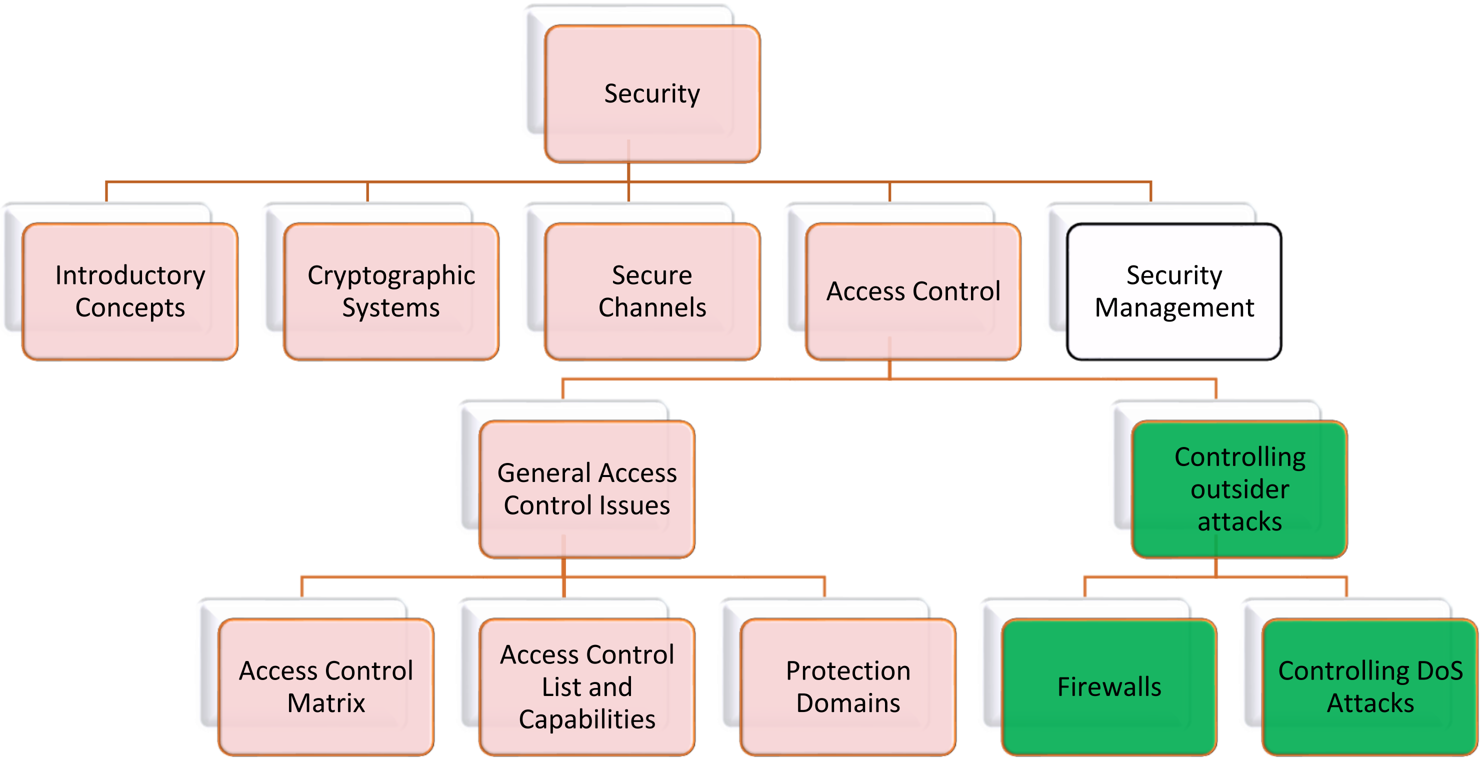
- Advantage
  - Managing group membership is easy

- Disadvantage
  - Looking up for a member is costly

- An optimization:
  - Let each user carry a ***certificate*** of all the groups they belong to
  - Reference monitor validates the certificate (similar to the Capabilities approach)



Overview


Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Access Control
Security Management
General Access Control Issues
Controlling outsider attacks
Access Control Matrix
Access Control List and Capabilities
Protection Domains
Firewalls
Controlling DoS Attacks

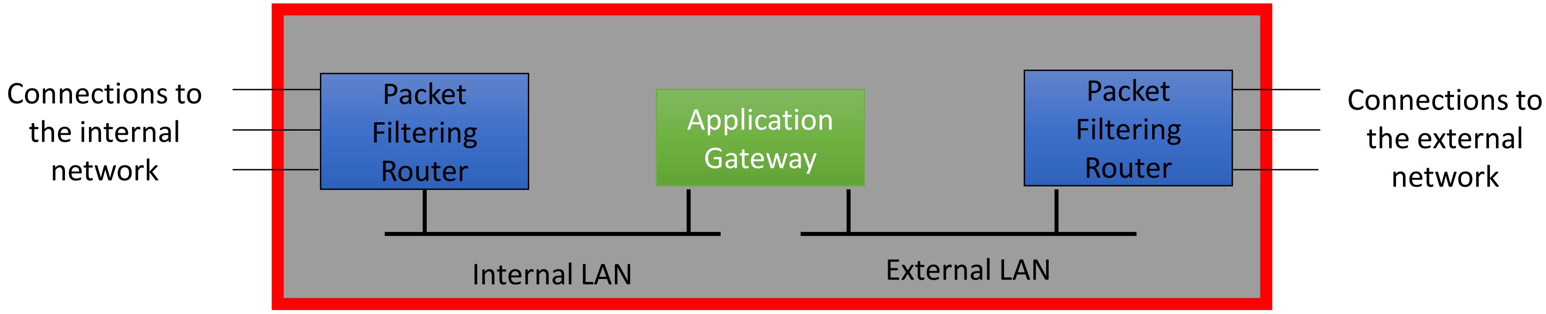
Controlling Outsider Attacks

- Above discussed schemes are good for preventing against small number of users who belong to a DS
- But, what if any user on the Internet can access the resources of DS
  - Search engines, Sending mails, …
- Above approaches are complicated, inefficient and time-consuming when a DS is exposed to the rest-of-the-world
- We discuss two coarse grained Access Control schemes
  - Firewall
  - Controlling Denial of Service (DoS) attacks

Firewall

- Firewall provides access control by regulating the flow of network traffic to/from a LAN

- Firewall disconnects any part of the DS from outside world
  - The internal LANs route packets through Packet Filtering Routers (PFR) and Application Gateway (AG)
  - Incoming/outgoing packet contents (e.g., src/dest IP) are examined by PFRs
  - PFRs selectively forward the incoming/outgoing packets to/from AG



Denial of Service Attacks

- *Denial of Service* (DoS) attacks maliciously prevents authorized processes from accessing the resources
  - Example: Flood a search server with TCP requests
- DoS attacks from a single source can be easily prevented
- Triggering DoS attacks from multiple sources (Distributed DoS or DDoS) is hard to detect
  - Attackers hijack a large group of machines
  - DDoS is hard to detect

Types of DoS Attacks

Two types

1. Bandwidth Depletion
   - Send many messages to a server
   - Normal messages cannot reach the server

2. Resource Depletion
   - Trick the server to allocate large amount of resources
   - Example: TCP SYN flooding
     - Initiate a lot of TCP connections on server (by sending TCP SYN packets)
     - Do not close the connections
     - Each connection request blocks some memory for some time

Preventing DoS Attacks

Idea: Detect DDoS by continuously monitoring network traffic at routers

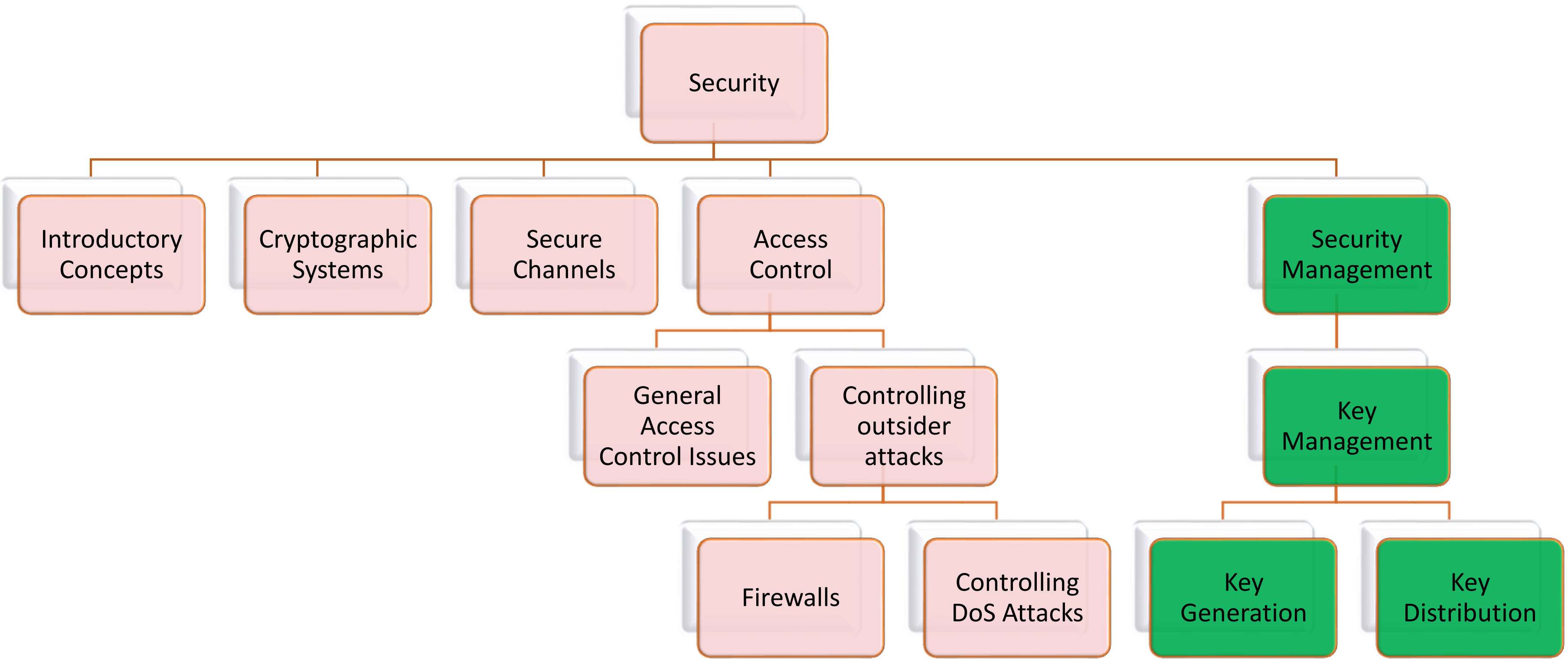
Three types:

- Monitor at ingress routers:
  - Monitor incoming traffic for large flow towards a destination
  - Drawback: Too late to detect since regular traffic is already blocked

- Monitor at egress routers:
  - Monitor traffic when packets leave organization's network
  - e.g., Drop packets whose source IP does not belong to the org's network

- Monitor at core internet routers:
  - Monitor at routers at the core of the Internet Service Providers (ISPs)
  - e.g., Drop packets when rate of traffic <u>to</u> a specific node is disproportionately larger than the rate of traffic <u>from</u> that node

Overview

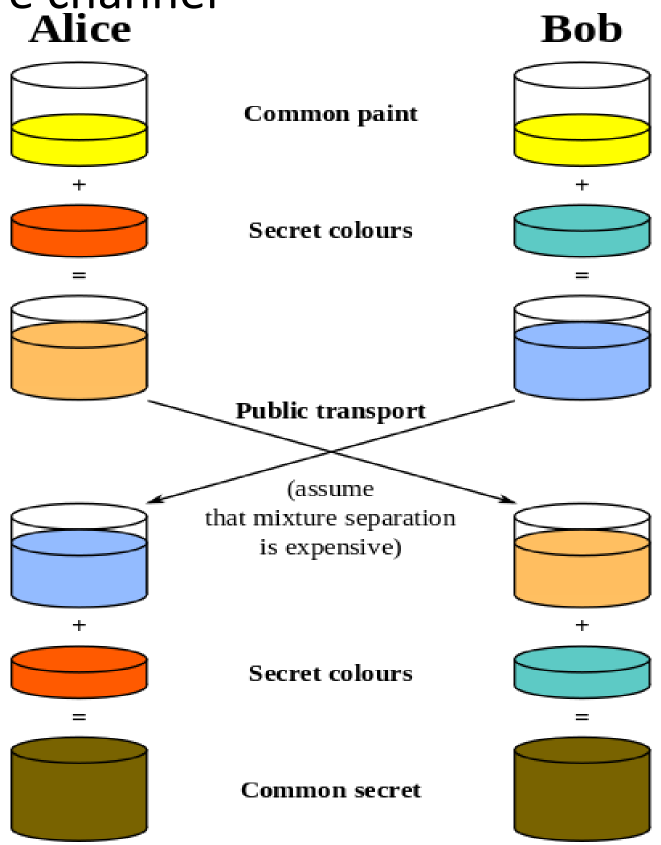

Security
Introductory Concepts
Cryptographic Systems
Secure Channels
Access Control
Security Management
General Access Control Issues
Controlling outsider attacks
Key Management
Firewalls
Controlling DoS Attacks
Key Generation
Key Distribution

Key Management

- We have studied how to use keys for authentication and encryption
- Examples of key-generation that we studied:
  - Session key
    - Generated and distributed by public-private key pair
  - Public-Private Key
    - Generated by algorithms. How is it securely distributed?
- How to boot-strap secure key generation and distribution in a DS?
- There are two main approaches in key management:
  - Diffie-Hellman Algorithm (For key generation over insecure channels)
  - Key distribution

Diffie-Hellman Key Generation

- Diffie-Hellman algorithm establishes a <u>shared key</u> between two entities across an insecure channel



Diffie-Hellman Algorithm

Alice	Eve	Bob
Selects two large numbers **n** and **g** with certain mathematical properties		
Choose a secret number **x**		Choose a secret number **y**
Alice transmits **[n, g, g^x mod n]**	**n, g,** **g^x mod n**	**y, n, g, g^x mod n**
n, g, x, g^x mod n, g^y mod n	**n, g,** **g^x mod n,** **g^y mod n**	Bob transmits **[g^y mod n]**
Alice computes **$(g^y \bmod n)^x = g^{xy} \bmod n$**		Bob computes **$(g^x \bmod n)^y = g^{xy} \bmod n$**
SHARED SECRET KEY = **g^{xy} mod n**		SHARED SECRET KEY = **g^{xy} mod n**

Discussion of Diffie-Hellman Algorithm

- D-H algorithm can be used in:
  - Symmetric Cryptosystems
    - Secret key **g^{xy} mod n** can be generated between A and B even in the presence of insecure channel
  - Public-key Cryptosystems
    - A's private key = **x**
    - A's public key = **g^{x} mod n**
- But, how to securely distribute public keys?

Key Distribution

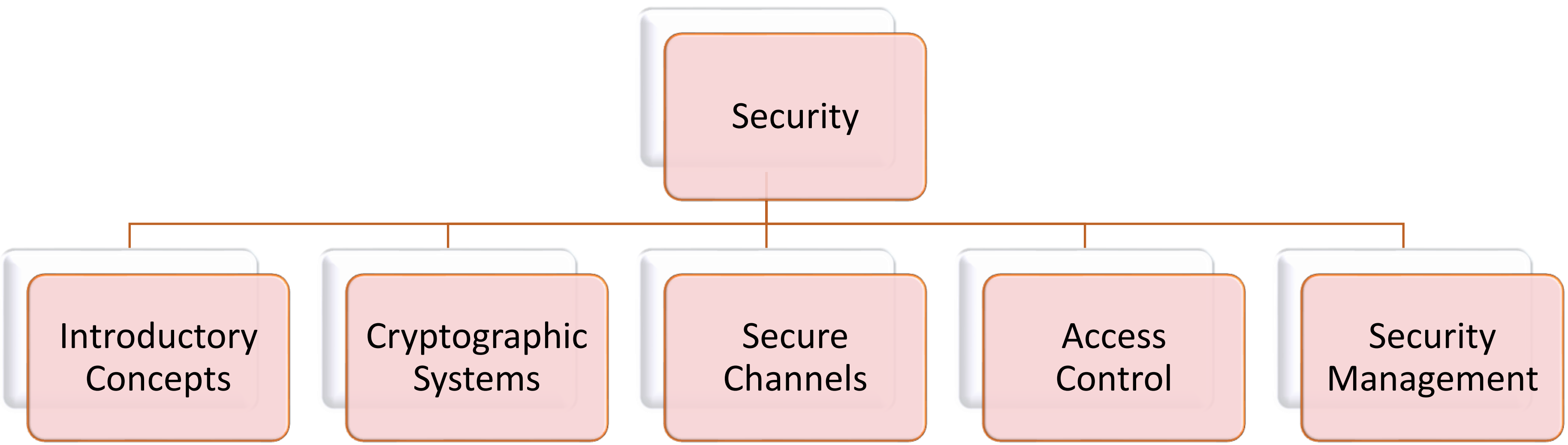
- Key distribution in Symmetric Cryptosystems
  - Out-of-band key exchange
  - Diffie-Hellman algorithm
    - Complex for a large DS

- Key distribution in Public-key Cryptosystems
  - Main problem: How do we authenticate the entity that transmitted the public key?
  - We will study an approach called Public-key certificates

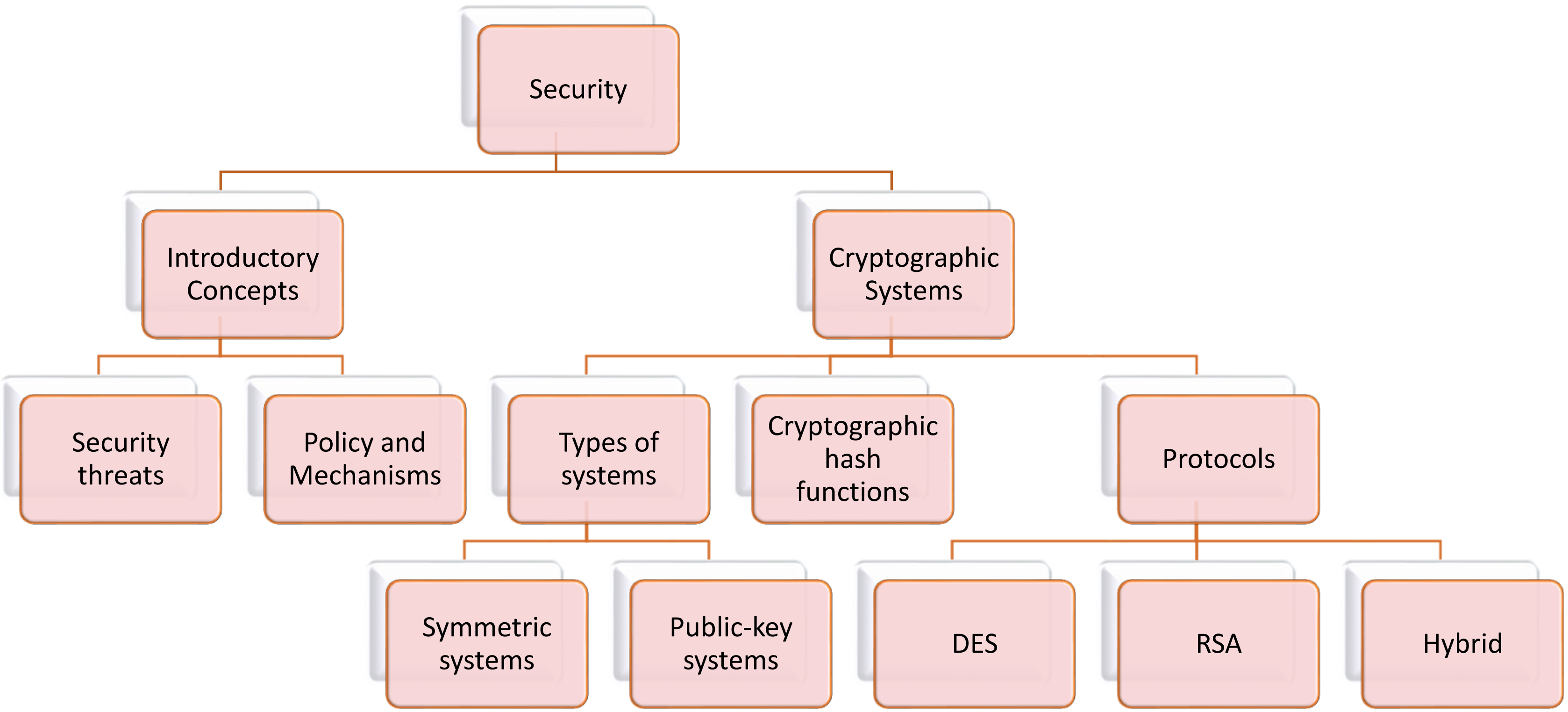
Public-key Certificate

- In practice, we trust a well-know Certification Authority (CA)
  - Example of a CA: VeriSign
- Public key of CA (K^{+}_{CA}) is well-known
  - e.g., It is built-in with browsers
- How does Bob validate public key of Alice?
  - Each user (Alice) has a <u>public-key certificate</u> cert(A, K^{+}_{A}) that issued by CA
  - Bob verifies if the received (A, K^{+}_{A}) matches that in the certificate

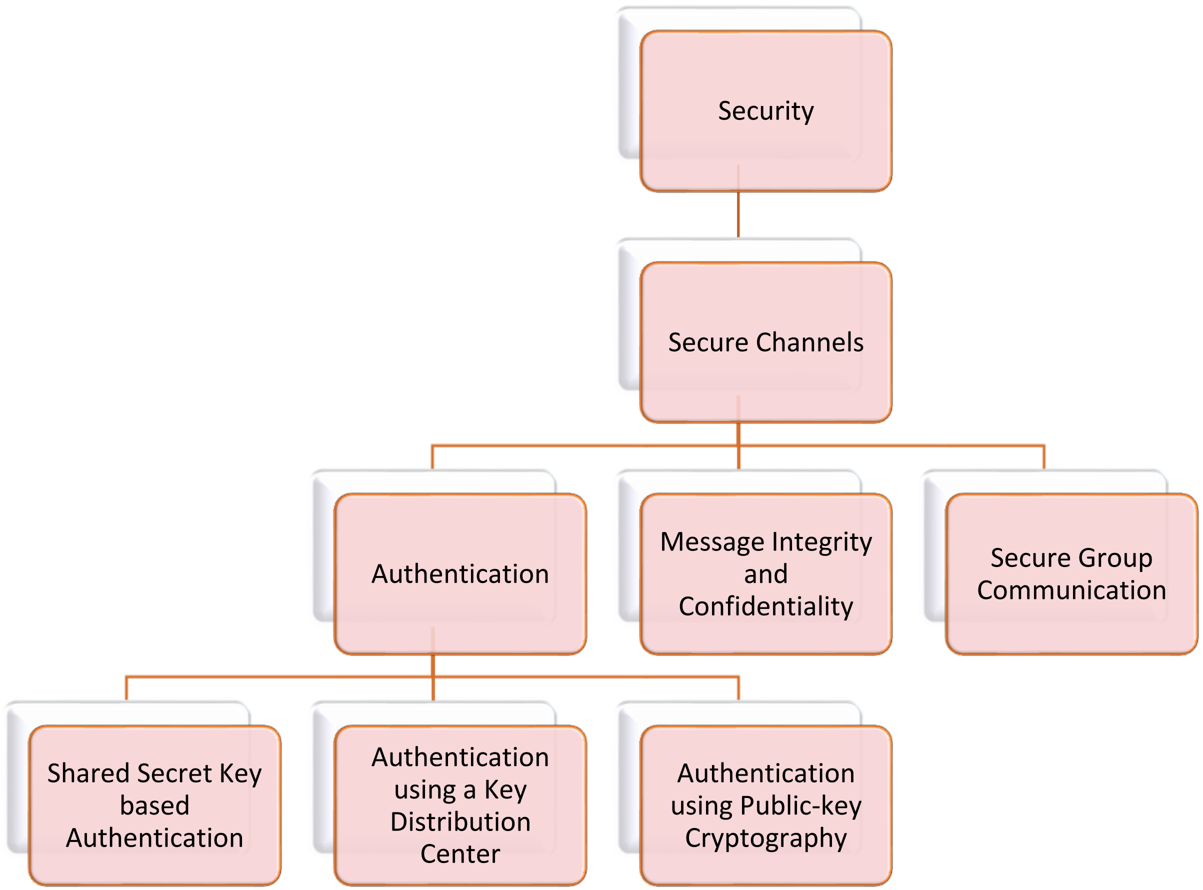
High-level Summary of Security Chapter



Summary of "Introductory Concepts" and "Cryptographic Systems"



Summary of “Secure Channels”



Summary of Security Chapter