

Course information

Security in Computer Networks and Systems

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Official Page: Fénix

The screenshot shows the homepage of the Fénix system. At the top left is the logo of Instituto Superior Técnico de Lisboa (TÉCNICO LISBOA). The main title "Network and Computer Security" is centered above a blue horizontal bar. To the right of the title are links for "+ Info" and "PT / EN". Below the bar, there are two sections: "Latest Announcements" on the left and "Initial Page" with a refresh icon on the right.

TÉCNICO LISBOA

Network and Computer Security + Info PT / EN

Latest Announcements Initial Page

- All information in this presentation is superseded by whatever is in Fénix

General Information

- meic-sirs@disciplinas.tecnico.ulisboa.pt
 - Subject: **[SIRS]** ...
- Lectures
- Labs
 - Guides
 - Have them prepared before class
 - Project
- Grades
 - Theory (50%) + Practice (50%)
 - Passing grade: 9,5 values (out of 20)

Teaching staff

- Theoretical lectures:
 - David R. Matos
 - Ricardo Chaves
- Lab/project:
 - Ricardo Chaves
 - David R. Matos
 - Christof Torres
 - Mafalda Ferreira
 - Martim Monis
 - Miguel Eleutério



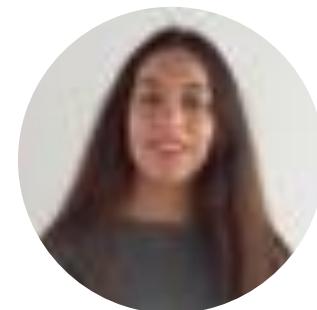
DRM



RC



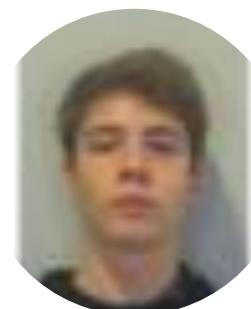
CT



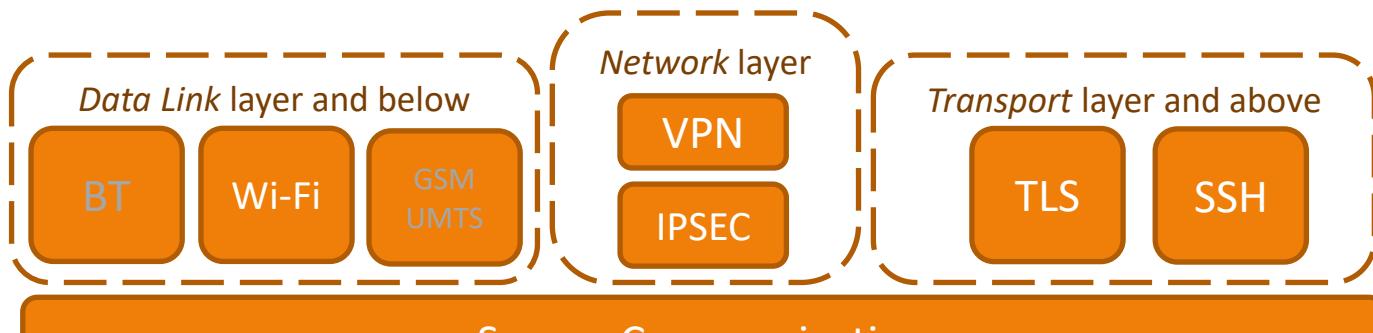
MF



MM



ME



Authorization

Authentication
Protocols

Secret Key
Management

Public Key
Management

Cryptography

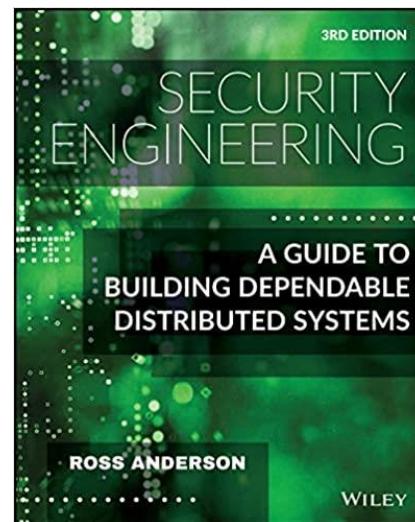
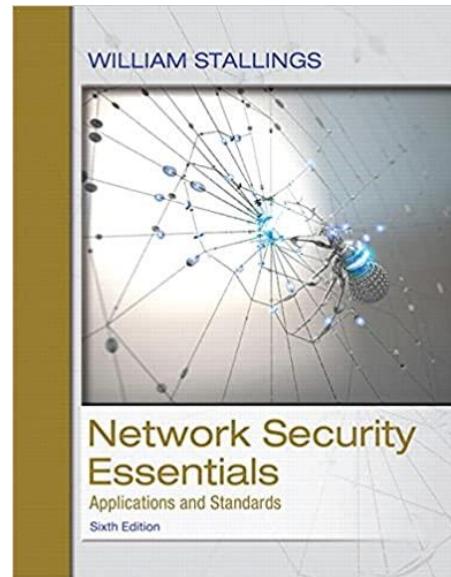
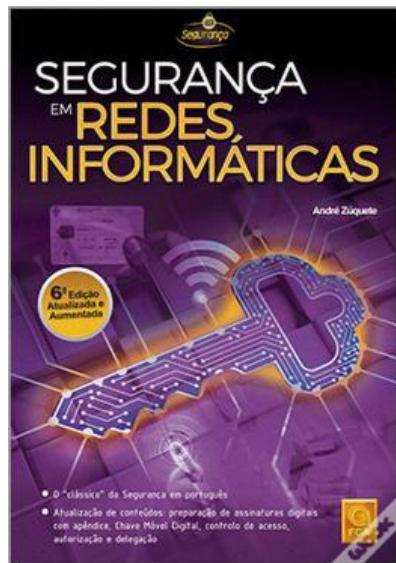
Firewalls &
Intrusion detection

Network
vulnerabilities

Introduction & Overview

Bibliography

- Primary:
 - Zúquete21 (PT)
 - Stallings17 (EN)
- Secondary:
 - Anderson21 (EN)



Grade assessment (theory)

- 1 exam
 - January 23rd, 10:30
 - Exam has a minimum grade of 8 out of 20 values
- There is a recovery exam
 - February 6th, 18:00
 - Theoretical grade is the best of the two exams

Ethics and law

- The purpose of the course is to learn how to protect computer systems from cyber-attacks
 - but some of the things you learn may also be used to attack them
- Notice that
 - Attacking systems is unethical and punished by law
 - Even just “testing” systems without written permission is punished by law
- ~~“Do not try this at home”~~
→ “Try this only at home”

Labs

- Labs
 - Laboratory guides
 - Should be prepared before the lab session
 - Group teamwork
 - To learn and build the project
 - Feedback to individual progress is recorded (not graded)
 - We expect you to **proactively show your work**
 - **Red** – no evidence of progress
 - **Yellow** – progress partially demonstrated
 - **Green** – progress demonstrated

Feedback sheet

Alameda	26				
Alameda	26				
Alameda	26				
Alameda	27				
Alameda	27				
Alameda	27				
Alameda	63				
Alameda	63				
Alameda	63				
Tagus	4				
Tagus	4				
Tagus	4				
Tagus	5				
Tagus	5				
Tagus	5				
Tagus	9				
Tagus	9				
Tagus	9				
Tagus	9				
Tagus	10				
Tagus	10				
Tagus	10				
Tagus	11				

Project

- Overview

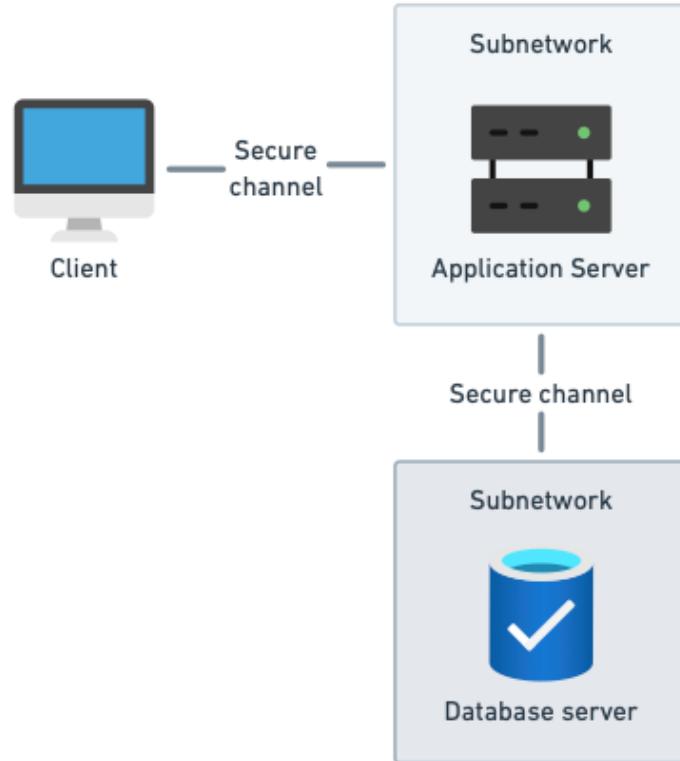
- Choice of business scenario
- Secure document/message library
- Server Infrastructure
- Configuration of secure channels
- Security challenge

- Feedback

- Demonstrate progress in labs
- Build report gradually

- Dates

- Mid point status: November 29th
- Submission: December 20th
- Presentations/Discussions: January 6th – 12th



Grade assessment (practice)

- Project
 - 3 Students per group (of the same class slot)
 - Enrollment is done in lab of first week
 - Minimum grade: 8 out of 20
 - Can be reused from last year (only)
 - If you want to reuse, **do not** enroll in the lab
- Each group member will answer individual questions

Assessment: Special Season

- Isolated from the regular period
- Grades from the normal period cannot be reused
- Exam (50%)
- Individual project (50%)

Summary Plan

Week	Theoretical	Practical
1	Introduction	(groups) + Virtual Machine
2	Cryptography	Project: cryptography
3	Network vulnerabilities	Virtual Networks + Traffic analysis
4	Firewall and Intrusion Detection	Project: infrastructure + firewalls
5	Key management and distribution	Project: security challenge
6	Authentication and authorization	Project
	Secure Communication: Wi-Fi, TLS	finalization
	<i>(Christmas break)</i>	
7		Project presentations/discussions

Introduction to Network and Computer Security

Segurança Informática em Redes e Sistemas

2024/25

David R. Matos, Ricardo Chaves

w/ Miguel Pardal, Carlos Ribeiro, Miguel Correia

We live in a digital world...

- There are more than 5 billion individuals using the Internet
 - That is 5 000 000 000 people
 - Around 2/3 of the world population
- And the number is still increasing...
 - 45% increase in Internet usage since 2018

Sources:

Statista <https://www.statista.com/statistics/617136/digital-population-worldwide/>
ITU Statistics <https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>

... but an insecure world ...

- Just in the first half of 2023...
 - Data Breaches:
 - 694 data breaches, impacting 612.4 million records
 - Identity Theft:
 - 1.4 million identity theft cases reported to the FTC
 - Ransomware Attacks:
 - Attackers extorted at least € 420 million
 - Bitcoin represents about 98% of ransomware payments

Sources:

<https://sites.udel.edu/threat/2023/08/08/major-security-breaches/>

<https://identitytheft.org/statistics/>

<https://www.stationx.net/ransomware-statistics/>

... and costs are rising

- The cost of cyber-crime is projected to be approximately €7 billion at the end of 2023
 - 48% of organizations report an increase in cyberattacks
 - Compared to the previous year
 - Attack frequency is also on the rise
 - Estimate of a cost of over €9 billion by 2025

Sources:

Forbes <https://www.forbes.com/sites/chuckbrooks/2023/03/05/cybersecurity-trends--statistics-for-2023-more-treachery-and-risk-ahead-as-attack-surface-and-hacker-capabilities-grow/>

Cybersecurity

- A secure digital infrastructure is required for an open society
 - To provide personal, social, and economic confidence
- Cybersecurity is crucial for protecting **people**
 - Their **data**
 - The **systems** that store it

Cybersecurity definition

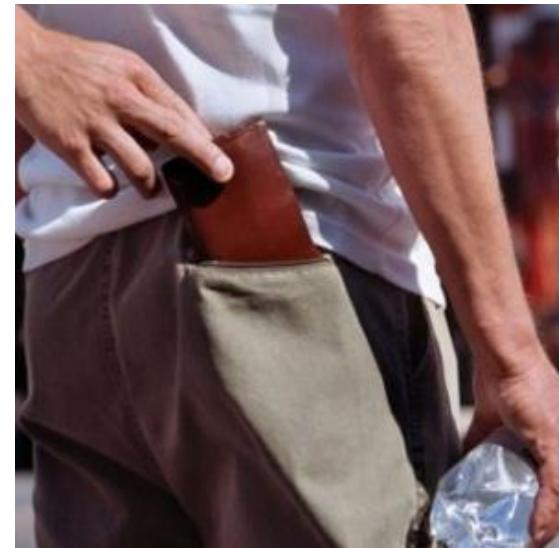


*“the prevention of damage to, unauthorized use of, exploitation of, and the restoration of electronic information and communications systems, and the **information** they contain, in order to strengthen the **confidentiality, integrity and availability** of these systems.”*

(U.S. National Institute of Standards and Technology)

Fundamental Problem

- We live in a **shared environment**
 - Public spaces
 - Shared physical spaces
 - Use of common infrastructures
 - Resource sharing



Sharing in Computer Systems

- Computer systems are **designed to share data**
- Using shared resources
 - Files
 - Memory
 - Program code
 - Peripherals
 - **Networks / Internet**
 - Physical communication medium
 - Switching mechanisms

Sharing Violations

- **Information leakage**
 - Acquisition of information by unauthorized agents
- **Information corruption**
 - Unauthorized tampering of information
- **Vandalism**
 - Interference with the correct operation of the system without benefits to the attacker

Computers make security harder

- Attacks can be **automated**
 - Ability to reproduce an action millions of times, quickly
- Attacks can be **remote**
 - Distance is not a limiting factor due to the Internet
 - Rapid propagation of techniques

Isolation can limit sharing

- Most attack opportunities are enabled by sharing
 - So, we limit sharing with **isolation**
- Physical isolation
 - Safes, walls
- People isolation
 - Only a certain group is informed
- Logical isolation
 - Encrypting a document makes the information unintelligible

Computer Security

Main security properties / attributes (CIA):

- Confidentiality
- Integrity
- Availability

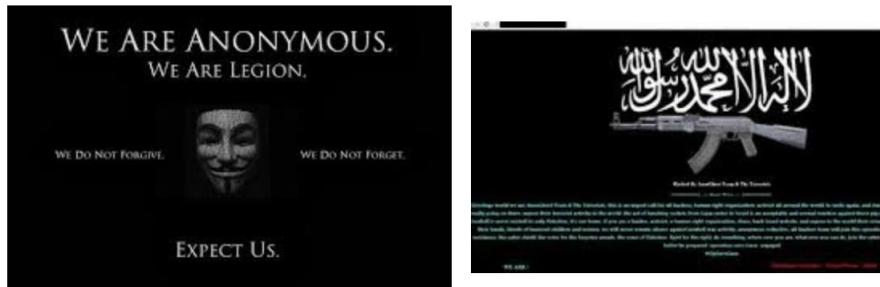
CIA: Confidentiality

- Confidentiality – absence of disclosure of **data** by non-authorized parties - “non-authorized” requires a security policy

The image shows a screenshot of an Ars Technica news article. The header features the site's logo 'ars' in white on a red circle followed by 'TECHNICA' in white. A navigation bar above the article includes categories: BIZ & IT, TECH, SCIENCE, POLICY, CARS, GAMING & CULTURE, and SITE. Below the header, a green banner reads 'HACKED AGAIN —'. The main title of the article is 'T-Mobile discloses 2nd data breach of 2023, this one leaking account PINs and more'. A subtitle below the title states 'Hack affecting 836 subscribers lasted for more than a month before it was discovered.' The author's name, 'DAN GOODIN', and the date, '5/2/2023, 12:40 AM', are also visible. At the bottom of the image is a photograph of a dark bird perched on a large, illuminated pink 'T' logo, which is part of a T-Mobile storefront sign.

CIA: Integrity

- Integrity – absence of invalid **data** or **system** modifications by non-authorized parties



CIA: Availability

- Availability – readiness of **system** to provide service



Computer Security

Main security properties / attributes (CIA):

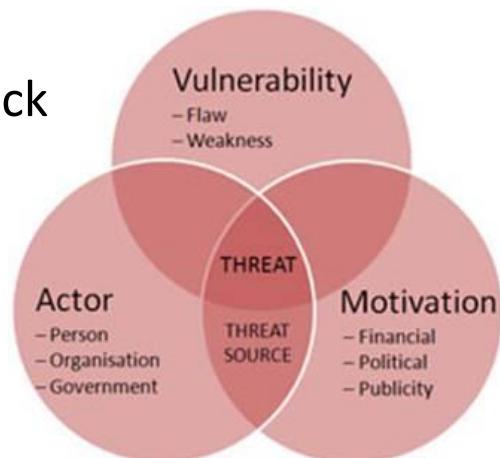
- **Confidentiality**
 - Privacy
 - Segregation of privileges
- **Integrity**
 - Authenticity – integrity of content and origin
 - Non-repudiation – do not deny action or authorship
 - Verifiable by others
- **Availability**

Excerpt from the GDPR: Confidentiality, integrity, availability?

- 'Personal data breach' is a security breach that accidentally or **unlawfully** leads to:
 - the destruction,
 - the loss,
 - alteration,
 - the disclosure, or
 - the unauthorized access
 - to **personal data** that has been transmitted, stored, or otherwise processed

Definitions

- **Vulnerability**
 - Characteristic of a system that makes it susceptible to attacks
- **Attack**
 - Actions that lead to the violation of a security attribute, often by exploiting vulnerabilities
- **Threat**
 - A *threat source* is an actor motivated to attack
 - A *threat* is a potential attack from a source facilitated by one or more vulnerabilities of the system



Attacker/Adversary



Attacker/Adversary

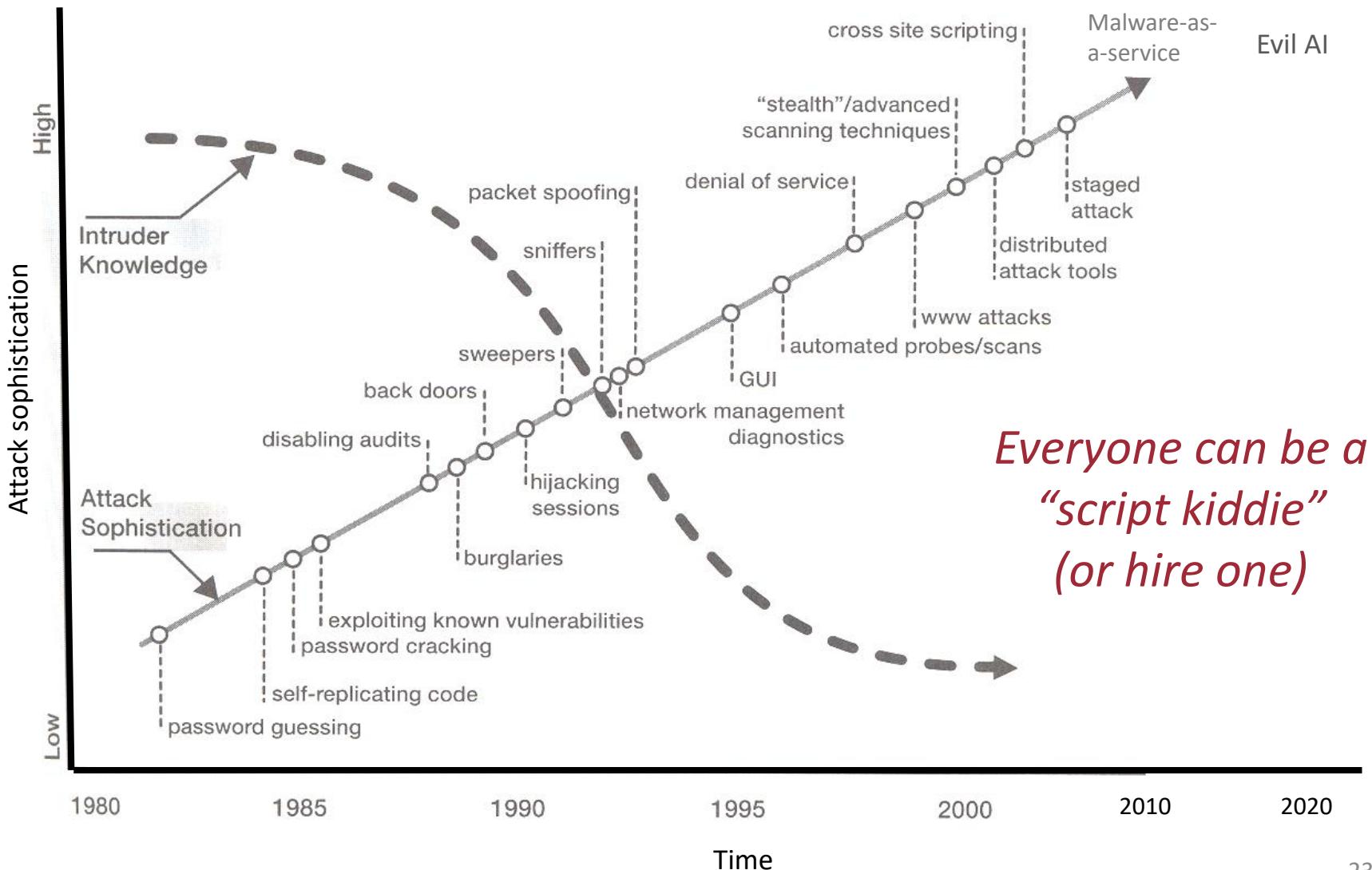
- In Cybersecurity, there is someone else, acting against us
 - From individual hackers to state entities
 - Attackers have varied motives and means
- Attackers adapt to defenses, creating new attacks
 - Continuous cycle
- Challenge: defenses must anticipate, understand, and adapt to ever-evolving threat sources

Possible attackers (with increasing capability)

- Journalists
- Hackers
- Individual criminals
- Organized crime
- Internal staff
- Terrorists
- Police
- Military organizations
- Industrial spies
- National security organizations



Attacks are becoming easier



Threats / attack effects

- Unauthorized access to data (Disclosure)
 - Extracting data from repositories
 - Inference by aggregation or concentration of information
 - Covert channels
 - Viruses, Trojans, worms, logic bombs
(also Hijacking, Disruption)
 - Concentration of responsibilities

Threats / attack effects

- Infrastructure
 - Equipment failures
 - Buggy software or operating systems
 - Network failures
- Performance
 - Reduced productivity
 - Delay in delivery of invoices
- Defective applications
 - *Bugs* causing procedural errors, etc.

Threats / attack effects

- Theft
 - Physical destruction (vandalism)
 - Theft of equipment or information
- Environmental
 - Failures of services
 - Natural disasters

Threats / attack effects

- Personnel
 - Unauthorized or uncontrolled internal access (impersonation)
 - Incorrect data entry (**Deception**)
 - Unhappy workers (Current or former)
- Warfare (**Disruption**)
 - Cyberattacks
 - Economical or military espionage
 - Computer terrorism

Top Threats

- Malware
 - Virus, worms, spyware, **ransomware**, crypto jacking, ...
- Social networks and WWW
 - Accessing the site e.g. similar but fake bank website
 - Obtaining private information through social networks
- Internal
 - Intentional and accidental
- Sophisticated distributed denial of service (DDoS)
 - Faster networks; asymmetry of the threat
- DNS attacks
 - Cache poisoning; domain theft; etc.
- Attacks on routers
 - For use in other attacks (e.g., disclosure, disruption)
 - Exploring the trust relationship between routers (BGP)

Our challenge

- How to ensure security properties for a system?
- Answer: **security mechanisms**
a.k.a. security controls

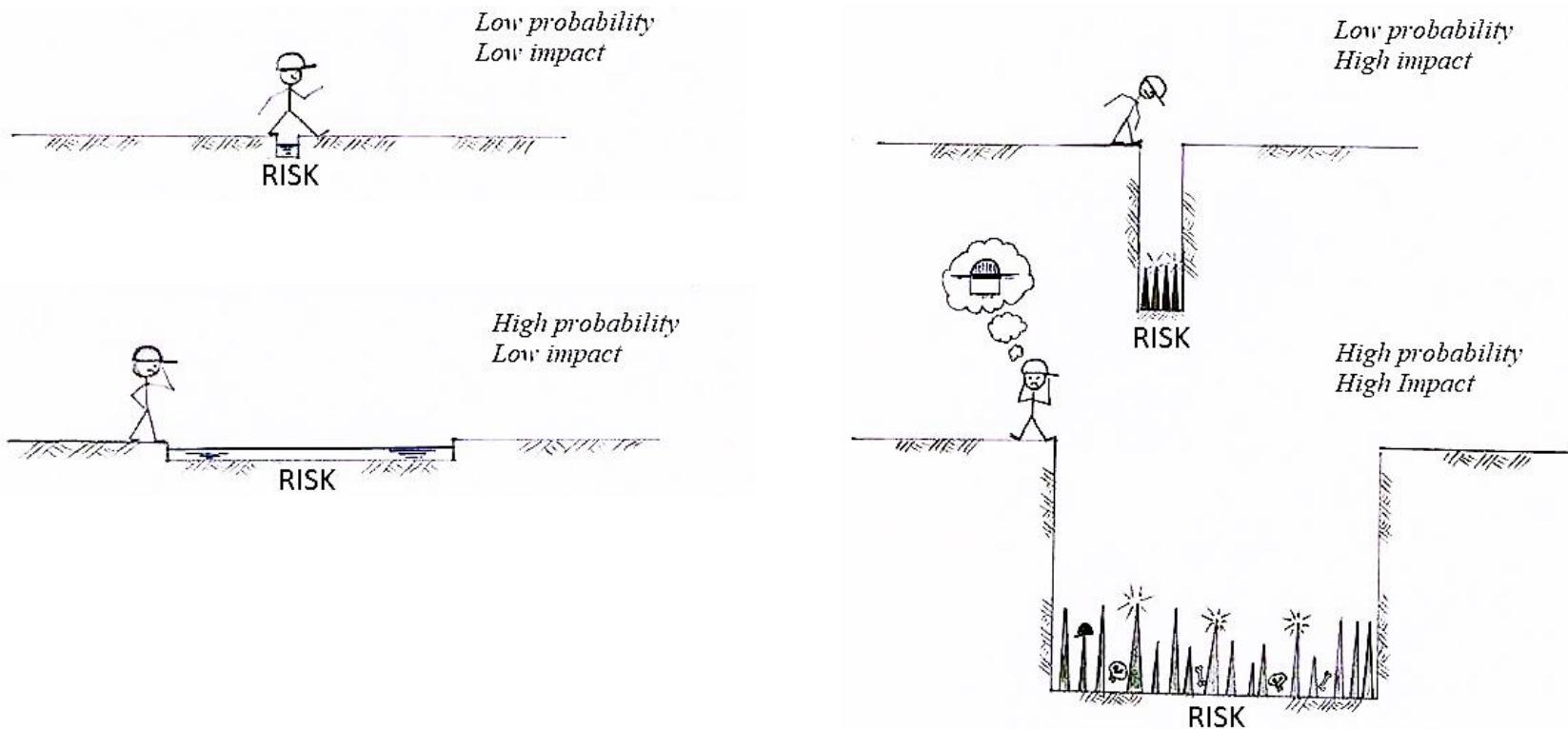
Defense / Protection

- Set of policies and security mechanisms aimed at
 - Reducing the vulnerability of a system
 - Detecting attacks as quickly as possible
 - Past or current
 - Reducing the risk level of a system

Risk

$$\text{Risk} = (\text{level of}) \text{ Threat} \times (\text{level of}) \text{ Vulnerability} \times \text{Impact}$$

Probability



Security Policy

- A security policy ensures protection of the asset against expected attacks, within constraints
- When is a security policy necessary?
 - When there is an **asset** in a shared space



Defining a Security Policy

- What do we want to protect?
- What are the potential threats?
- Who can execute them?
 - Who are the attackers/adversaries?
- How are threats materialized?
- What are the attacks?
- Which procedures and protection mechanisms can prevent these attacks?
- What is the cost of the security policy?
 - Ultimately: the cost of security must be lower than the asset's value

Security Mechanisms

- Specific tools, techniques, or methods implemented to detect, prevent, and respond to security threats



Security policies are **enforced** (i.e., made effective) by an appropriate use of security mechanisms

Security Policy and Mechanism

simple example

- Policy:
 - Only I can enter this room
- Mechanism:
 - Put a lock on the door and only I have the key

Security Policy and Mechanism

technical example

- Policy:
 - A company policy stating that all sensitive data must be encrypted when stored or transmitted
- Mechanism:
 - Implement SSL/TLS encryption for data transmission

Security mechanisms

- What are they?
- How do they work?
- What are they used for?

Services mechanisms: What are they?

- Confidentiality mechanisms
 - Access control
 - Encryption
 - Steganography
 - Confinement
 - etc.
- Integrity mechanisms
 - Cryptography
 - Authentication
 - Repudiation
 - Identification
 - etc.
- Availability mechanisms
 - Fault tolerant replication
 - Crypto puzzles
 - etc.

Security mechanisms: How do they work?

- Prevention
 - Prevent the attack from succeeding
 - Very intrusive
 - Easy management
- Detection
 - Important for unpredictable attacks
 - Complex management
 - Not much intrusive
- Recovery
 - Restitution of the state before the attack
 - Tolerance to attacks

Security mechanisms: What are they used for?

- Defend ourselves against threats
- Against which threats?

Summary

- Sharing vs Isolation
- Main security properties
 - CIA
- Definitions
 - Vulnerabilities, Attacks, Threats
- Defense / protection
 - Security policy and mechanisms
 - Driven by risk assessment

Cybersecurity: Think Maliciously

- “Traditional” Computer Engineering
 - Focus: building for specific functionalities
 - Design, construct, and optimize
- Cybersecurity:
 - Focus: Understand vulnerabilities, potential breaches
 - Unbuild, reverse engineer to uncover flaws
 - Alter original intent to exploit
 - Build defenses against attacks

Network Attacks Overview

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

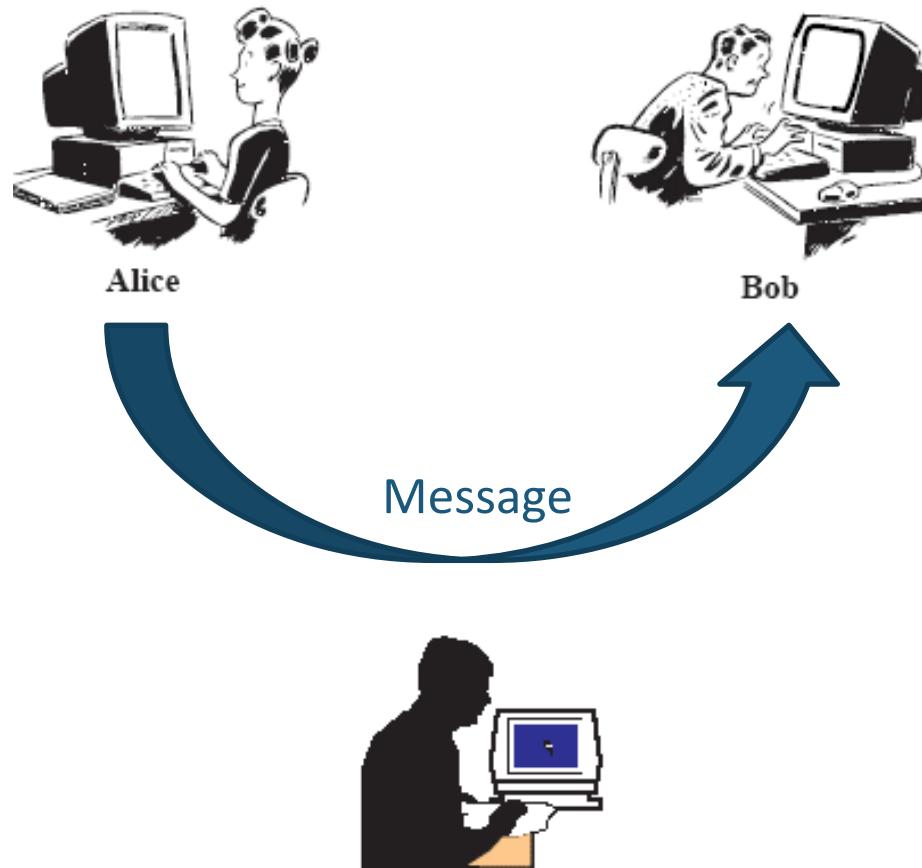
Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Roadmap

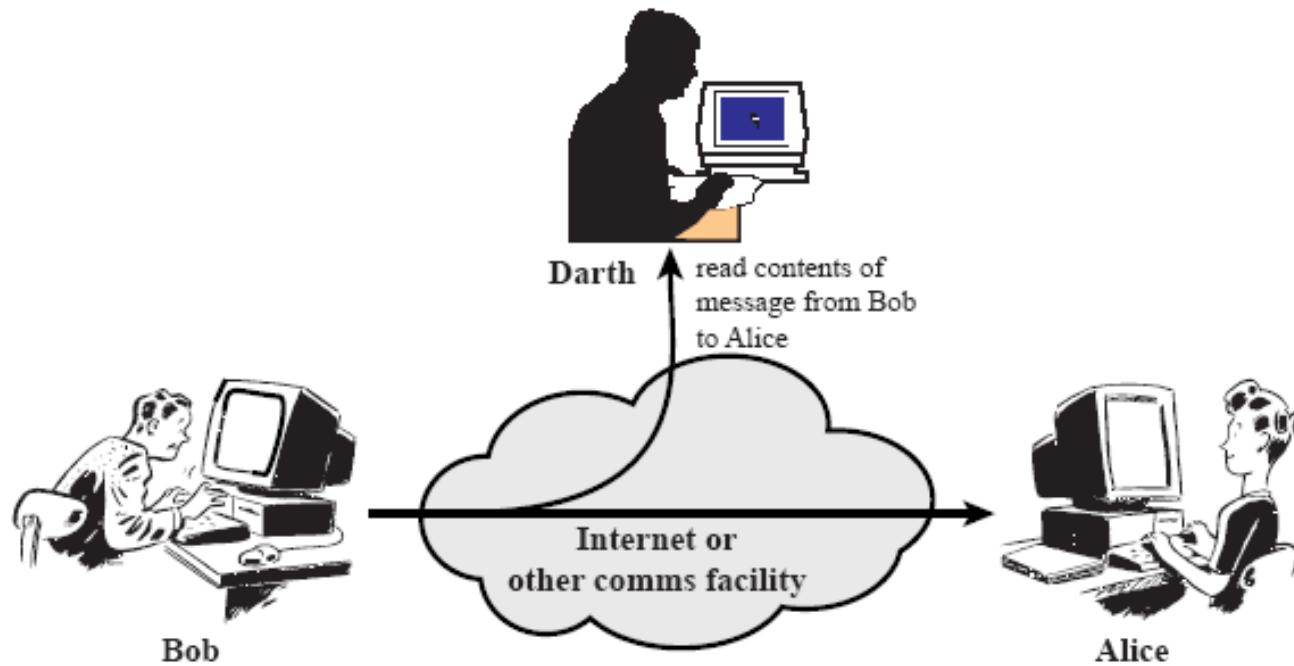
- Network attacks
- Security models

Network communication parties

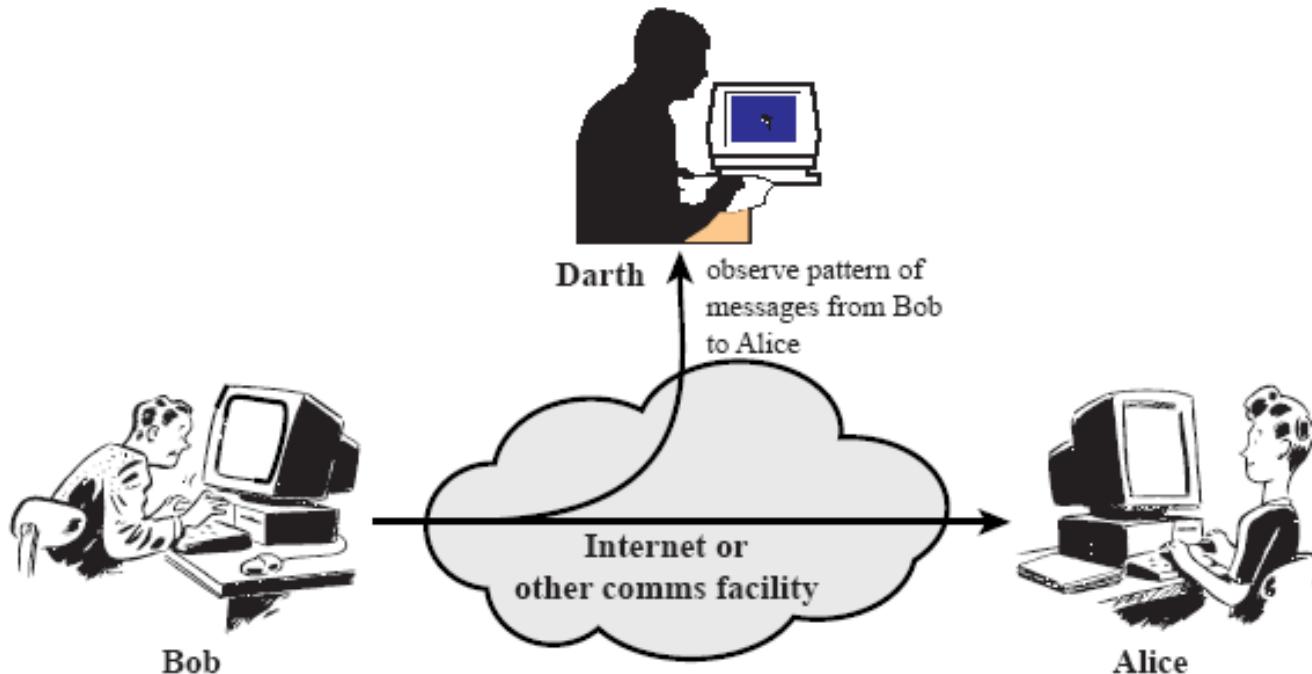
- Alice
 - Bob
 - Charlie
-
- Darth
 - Eve
 - Mallory



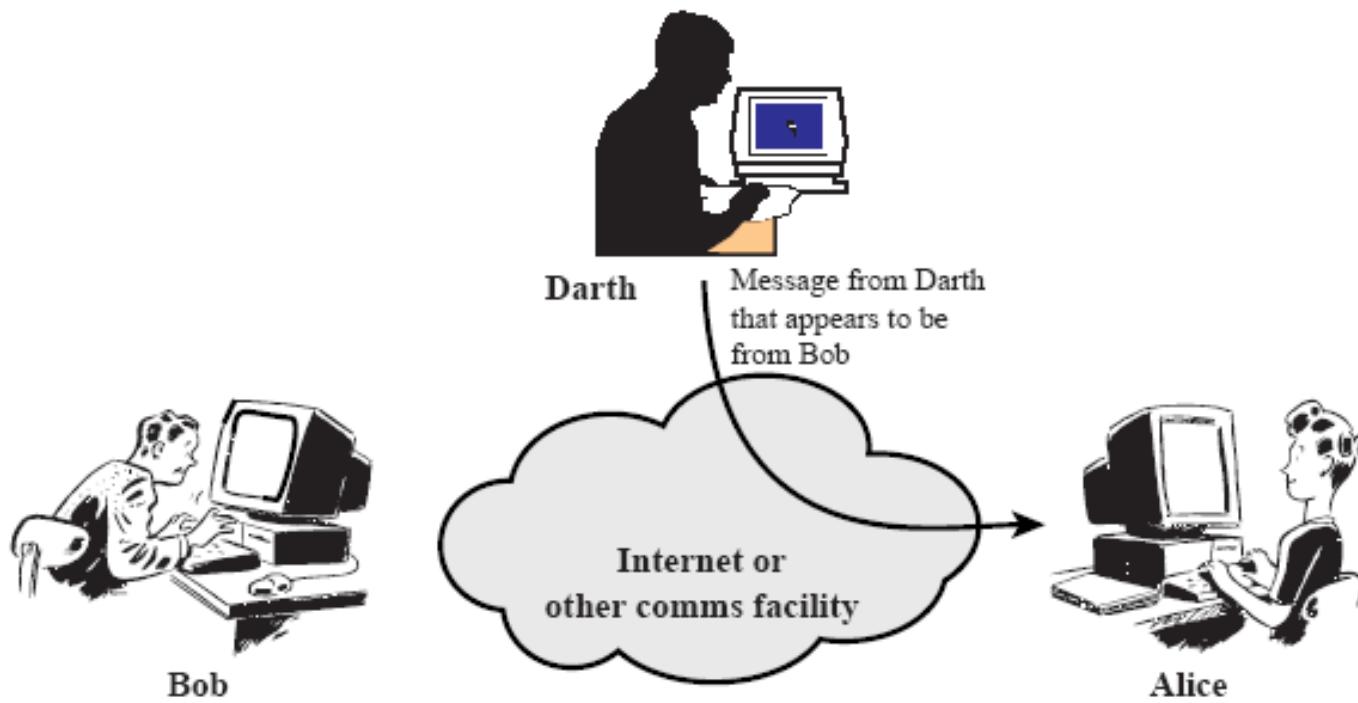
Passive attacks: Listen (information disclosure)



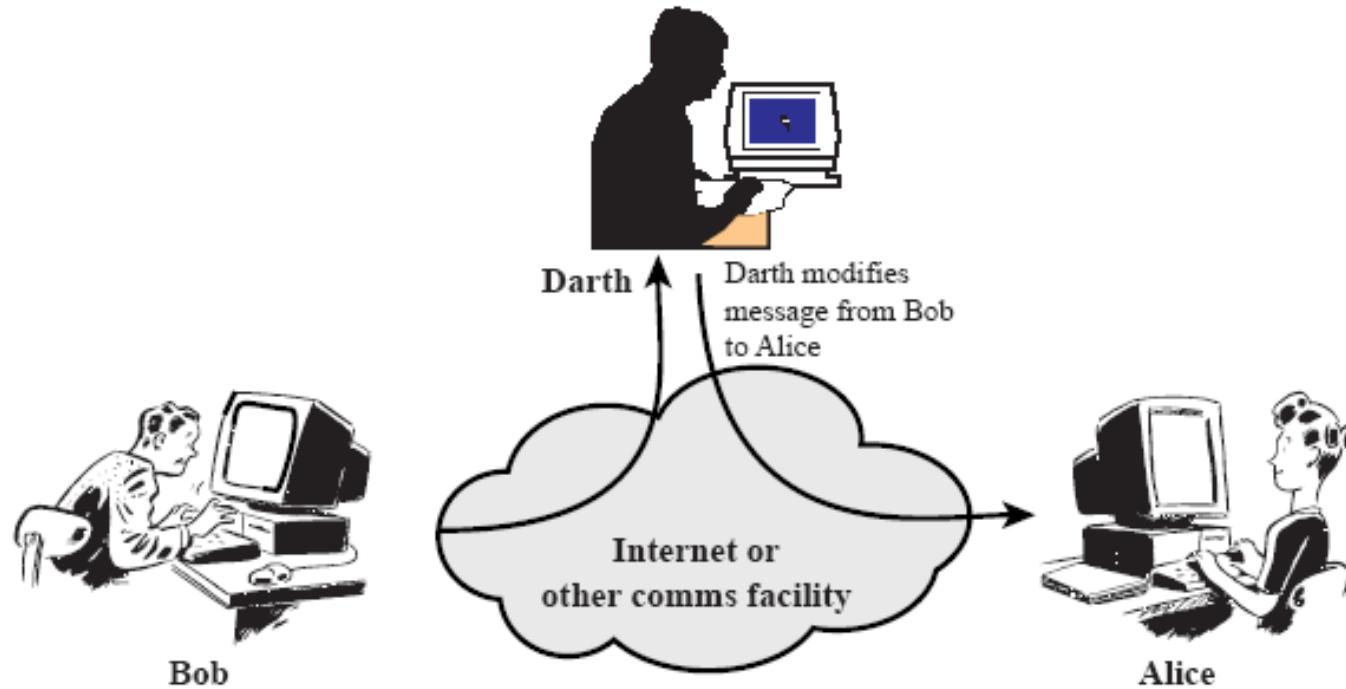
Passive attacks: Listen (traffic analysis)



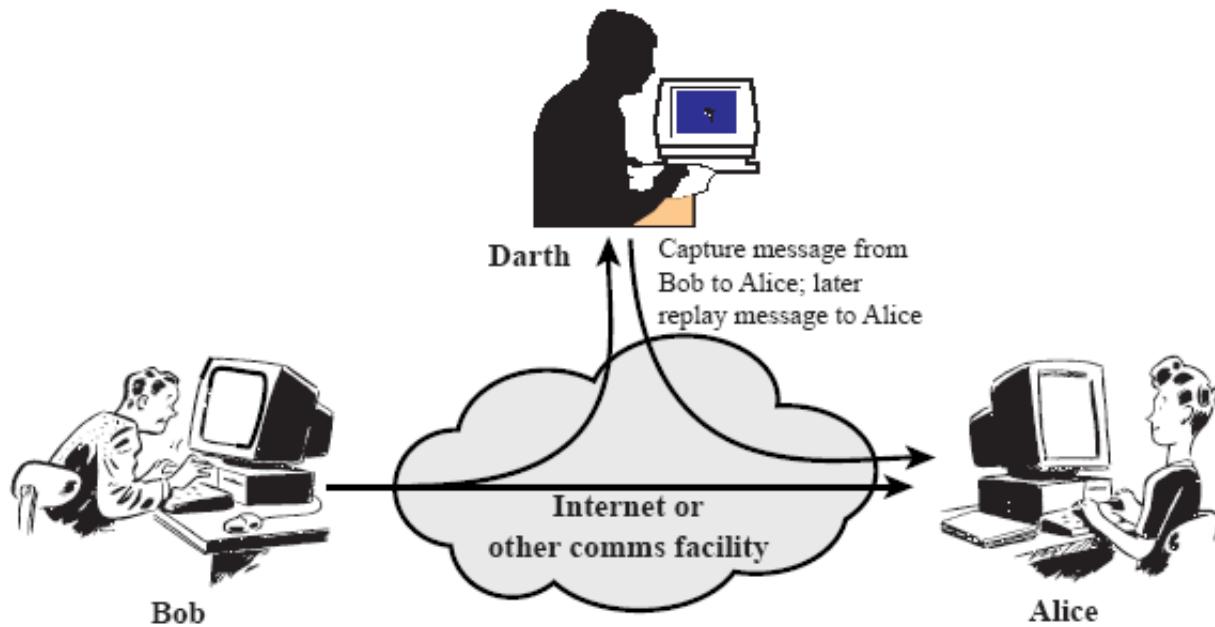
Active attacks: Impersonation



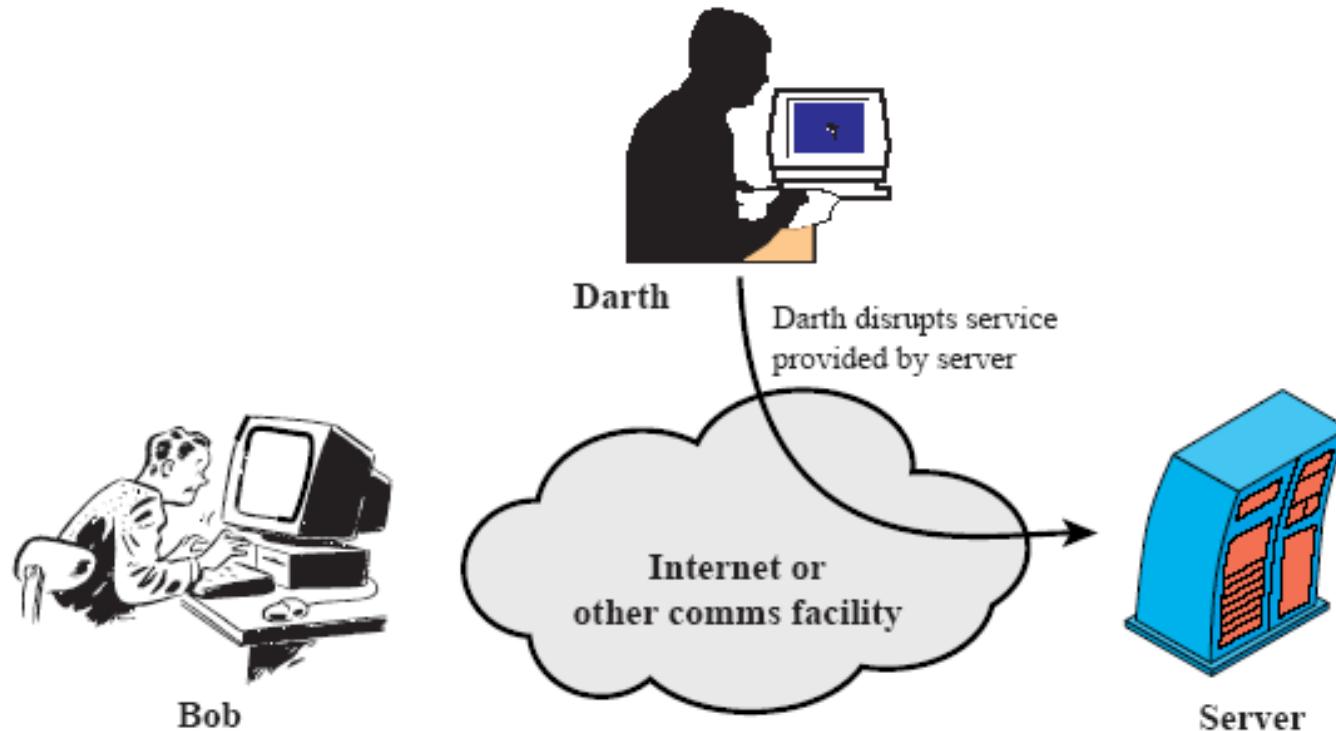
Active attacks: Tampering



Active attacks: Replay

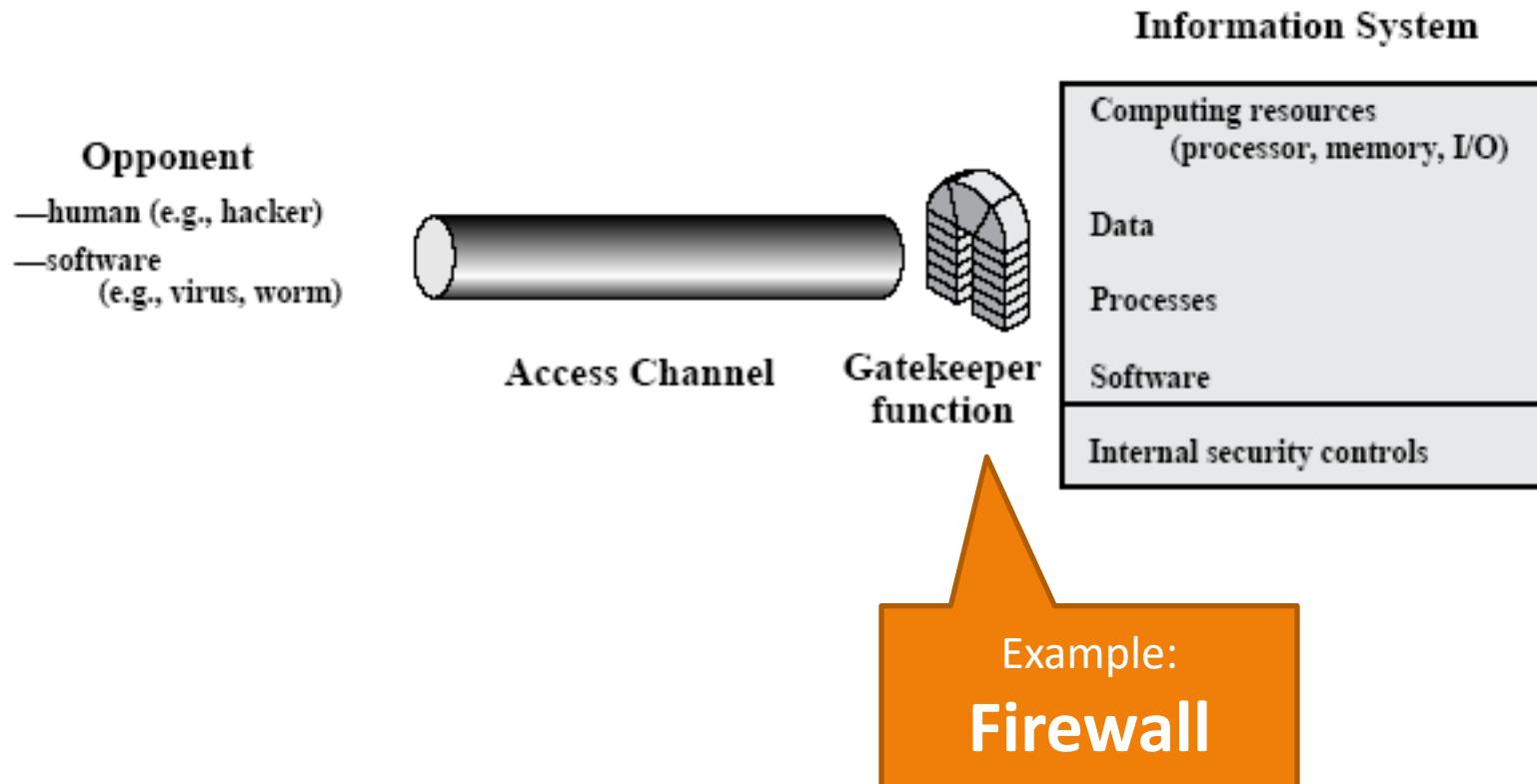


Active attacks: Denial of Service (DoS)

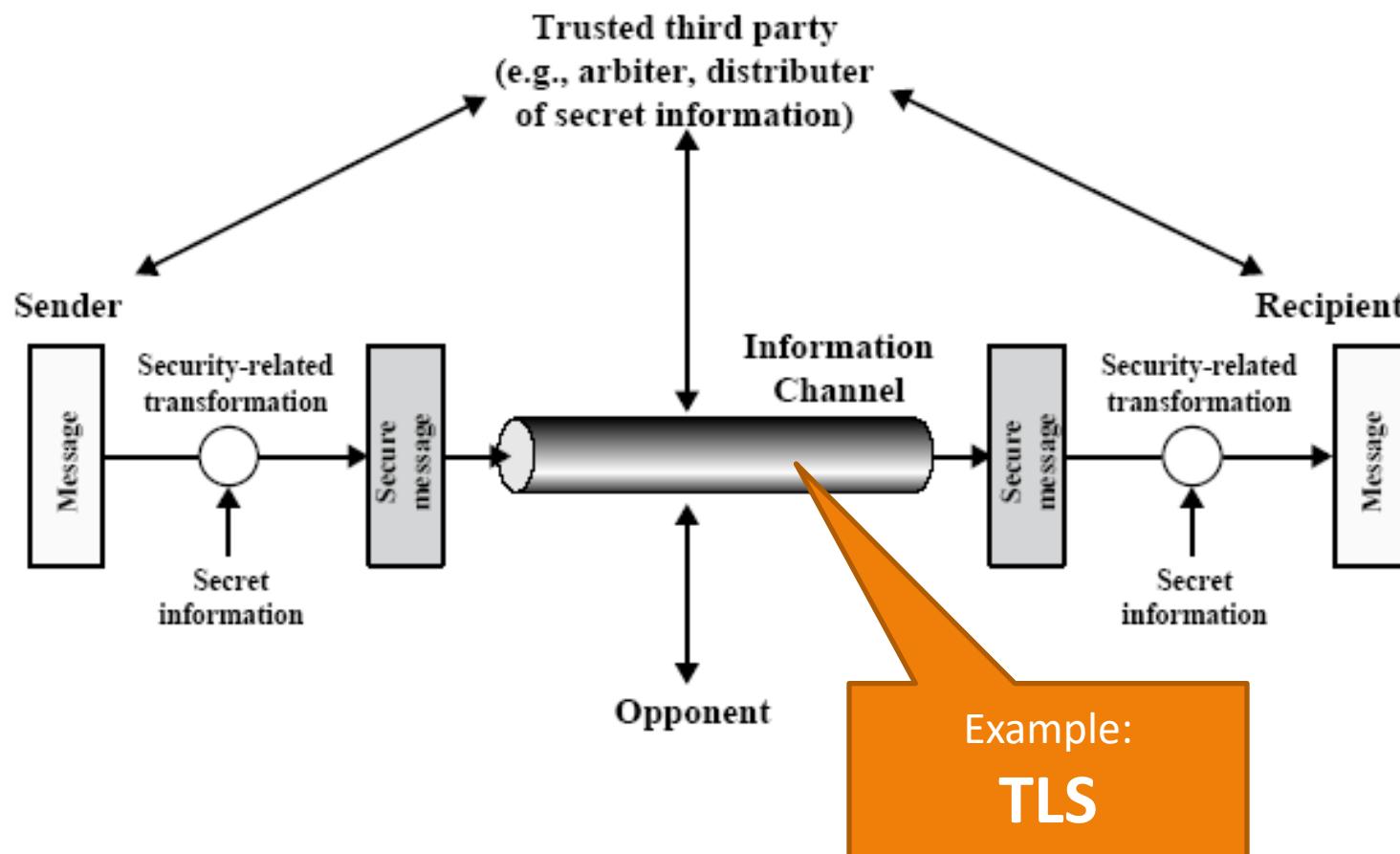


NETWORK SECURITY MODELS

Network Security Model I: Gatekeeper for access control



Network Security Model II: Secure communication channel



Cryptographic Services

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Roadmap

- Cryptography
- Cryptographic services
 - What we want to provide
- Cryptographic building blocks
 - Primitive and composite functions
 - No technology details (yet)
- How to provide the services using the functions

Cryptography: terminology

- Cryptography
 - Art or science of writing in a concealed form
 - from Greek: kryptós, hidden + graph, r. de graphein, write
 - Used to ensure confidentiality of data (until the 1970s)
 - Steganography
 - from Greek: steganós, covered + graph, r. de graphein, write
- Cryptanalysis
 - The art or science of breaking cryptographic systems or ciphered data
- Cryptology
 - Cryptography + Cryptanalysis

Steganography in ancient history

*In ancient Greece, **Histiaeus**, the ruler of Miletus, shaved a slave's head, tattooed it with a message, and waited for the hair to grow back.*

He then sent the messenger on the long journey from Persia to Greece to urge revolt.

*Upon arrival, the messenger's head was shaved again to read the message.**



*<https://thereader.mitpress.mit.edu/a-brief-history-of-secret-communication-methods/>

Cryptography in ancient history

Scytale: used for transposition cipher

It is a Cylinder with a strip of parchment wound around it on which is written a message

*The **key** is a rod with the right diameter*



Cryptography in ancient history

Caeser ciphers are simple substitution ciphers. Each letter in the plaintext is shifted a certain number of places down the alphabet.



Cryptography

- Widespread and dangerous belief:
 - Encrypting everything provides protection against anything
- A simple example to prove the contrary:
 - Money transfer from one bank to the other
 - The bank encrypts the whole message
 - The attacker:
 - Might not be able to understand the message! (or can he?)
 - But he might be able to:
 - Divert the message into his account (maybe not!)
 - Could get rich by:
 - Diverting or stopping debit messages
 - Allow the passage of all credit messages
 - He might be able to distinguish the two merely by looking at their size
 - Crash the bank by:
 - Injecting random messages

Cryptanalysis: what cryptography must protect from

- Basic assumption: the algorithm is known
 - If not public, might be obtained (e.g., stolen)
- Attacks:
 - **Ciphertext-only**: cryptanalyst has access to ciphertexts
 - Without them, no cryptanalysis is possible
 - **Known-plaintext**: cryptanalyst has a set of ciphertexts to which he knows the corresponding plaintext
 - Often easy to get at least partial plaintext, e.g., message beginning
 - **Chosen-plaintext**: cryptanalyst can obtain the ciphertexts corresponding to plaintexts of his choice; or:
 - **Chosen-ciphertext**: cryptanalyst can obtain the plaintexts corresponding to ciphertexts of his choice

Easier for attacker but harder to get



Attacks on information

- We want to protect the information against:
 - Unauthorized insertion of information
 - Unauthorized modification of information in transit
 - Unauthorized replay of information
 - From an earlier legitimate data transmission
 - Unauthorized access to information
- Which cryptographic services can we use to prevent this?

CRYPTOGRAPHIC SERVICES

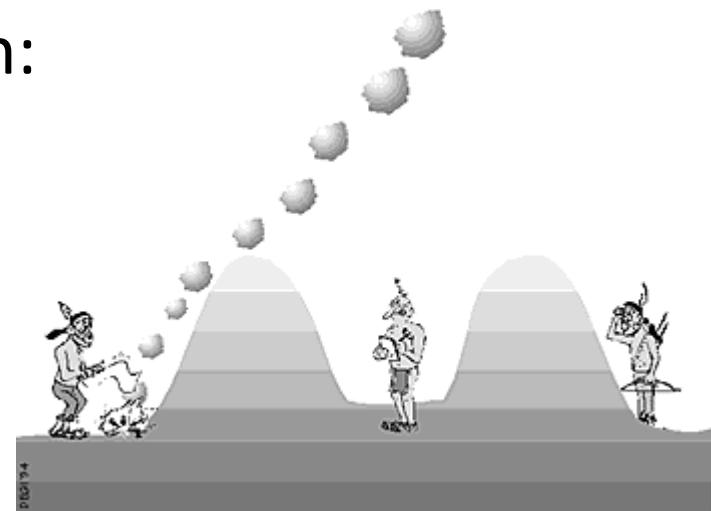
Cryptographic services

- We need the following cryptographic services:
 - Confidentiality
 - Integrity
 - Authenticity
 - Entity authentication
 - Data origin authentication
 - Non-Repudiation



1 - Confidentiality

- Is a service used to keep the content of the information from all, but those entities authorized to have it
 - i.e. making the information unintelligible to all but those who possess some secret
- Typically achieved by encryption:
 - Process of converting plaintext to ciphertext using
 - Cryptographic algorithms
 - Cryptographic key



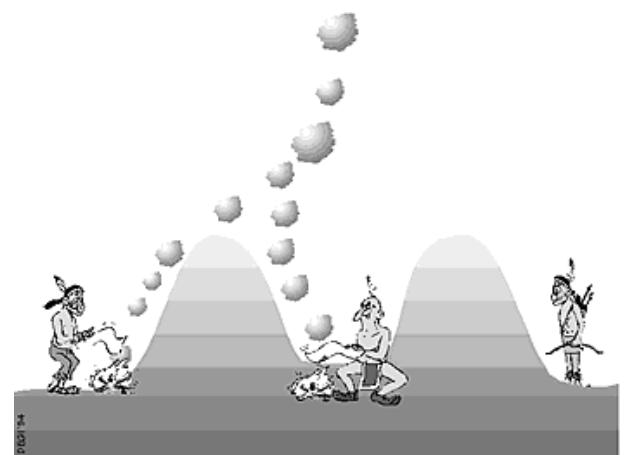
Confidentiality challenges

- Makes debugging harder
 - Software
 - Systems
 - Protocols
- Information loss
 - If the key is permanently lost, so is the information
- Sometimes misused
 - Other, more appropriate, services can be used



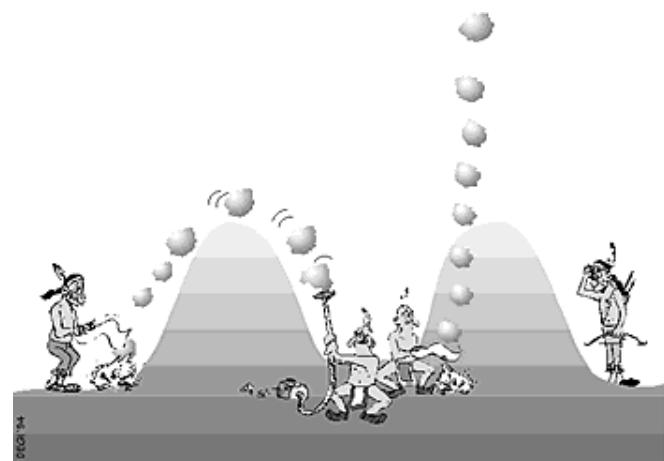
2 - Integrity

- Is a service that detects data manipulation by unauthorized entities
 - An intruder should not be able to substitute a false message for a legitimate one
- Not the same thing as error detection codes
 - e.g. Cyclic Redundancy Codes (CRC) are not cryptographically strong
 - do not protect against intentional alterations of the message



3 - Authenticity

- Is a service used to ascertain the identity or the origin of a message:
 - Guarantees that entities are who they claim to be
 - Verified identification
 - Data origin authentication
 - Requires *message integrity* and *freshness*
 - Tamper detection
 - Replay detection



Authentication

- Entity authentication
 - Verify the identity of an entity
 - Ensure legitimacy of parties involved in a communication
 - Sender authenticates itself to Receiver
 - Receiver checks evidence and decides to accept identity
 - Spoofing/impersonation must be infeasible
- Data origin authentication
 - Confirm the originator/creator of the message
 - Detect message tampering and replay
- All these features must remain true
 - Even after a large number of honest message exchanges

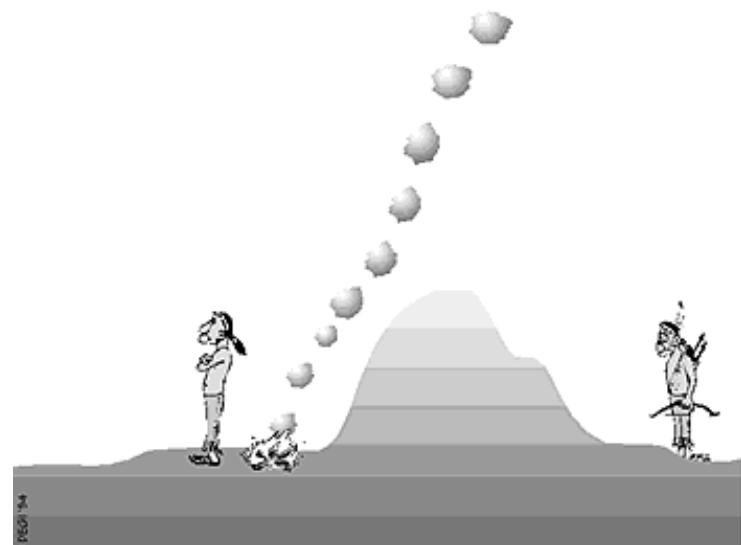
Device versus User Authentication

- Device Authentication
 - Validating a specific device
 - Often using digital certificates, network addresses, or device-specific keys
 - Not bound by human limitations
- User Authentication
 - Verifying the identity of an individual user
 - through credentials
 - Passwords
 - Biometrics
 - Hardware tokens
- We will come back to this later in the course



Non-Repudiation

- It is a service which prevents an entity from denying previous commitments or actions
 - Such as:
 - a sent message
 - a signed document
 - ...



CRYPTOGRAPHIC BUILDING BLOCKS

Primitive Building Blocks

- Cipher
 - Symmetric
 - Asymmetric
- Hash

Cipher and decipher functions

- Defines two functions: cipher, decipher
- Cipher function
 - Receives data and key
 - Outputs cryptogram
- Decipher function
 - Receives cryptogram and key
 - Outputs original data, if the key is correct
 - Otherwise, returns something else

Symmetric Cipher

- Uses the same key to cipher and decipher
- In cryptographic function notation:
 - $E(M, K)$ – cipher message M with key K
 - Produces cryptogram C
 - $D(C, K)$ – decipher cryptogram C with key K
 - Produce message M'
 - $M' = M$ if the key is correct
 - $D(E(M, K), K) = D(C, K) = M$

Asymmetric Cipher

- Instead of a key, we have a key pair:
 - One part we call the **private** key – KR
 - Only known by one entity
 - The other part we call **public** key – KU
 - Can be known by everybody else
- In cryptographic function notation:
 - AE() – Asymmetric Encryption
 - AD() – Asymmetric Decryption
- We can cipher with one key and decipher with the other

Cipher with private, decipher with public

- In cryptographic function notation:
 - $\text{AE}(M, KR)$ – cipher message M with private key KR
 - Produce cryptogram C
 - $\text{AD}(C, KU)$ – decipher cryptogram C with public key KU
 - Produce message M'
 - $M' = M$ if both keys belong to the same pair
- What do we know about the cryptogram C ?
 - Only the owner of the private key KR can produce it
 - Anyone with the public key KU can decipher it

Cipher with public, decipher with private

- In cryptographic function notation:
 - $\text{AE}(M, \text{KU})$ – cipher message m with public key KU
 - Produce cryptogram C
 - $\text{AD}(C, \text{KR})$ – decipher cryptogram c with private key KR
 - Produce message M'
 - $M' = M$ if both keys belong to the same pair
- What do we know about C in this case?
 - Anyone with the public key KU can produce it
 - Only the owner of the private key KR can decipher it

Cryptographic Hash

- A cryptographic hash function receives an input message and returns a digest of the data
 - Does not use a key
- In cryptographic function notation:
 - $H(M)$ – hash message M
 - Produce digest DT

Digest value produced by hash

- What do we know about the digest value DT?
 - Deterministic
 - The same input always produces the same digest value
 - Fixed Size
 - Digest values are of a fixed length, independent of the input size
 - Unique Representation
 - Ideally, each input produces a unique digest, though collisions can occur
 - Non-reversible
 - Hash is a one-way function
 - It is computationally infeasible to derive the original input from digest
 - Sensitive to Input Changes
 - Small changes in input significantly alter the digest (avalanche effect)

Composite Building Blocks

- Hybrid Cipher
- Integrity Check
 - Message Integrity Code
 - Digital Signature

Hybrid Cipher

- Typically, symmetric ciphers are 100 to 1000 times faster than asymmetric ciphers
 - Mathematical operations used in symmetric cryptography are simpler
- How can we have the best of both?
 - Generate random symmetric key KM
 - Cipher (large) message M with symmetric cipher
 - Cipher (small) key KM with asymmetric cipher
 - We get the same properties of asymmetric cipher with the performance of symmetric cipher
- Functions:
 - HE (Hybrid Encryption) and HD (Hybrid Decryption)

Hybrid Cipher in detail

- In cryptographic function notation:
 - Generate random key for message: $RND()$
 - Produce message key KM
 - Cipher the message key with public key of receiver: $AE(KM, KU)$
 - Produce cryptogram of key CK
 - Cipher the message: $E(M, KM)$
 - Produce cryptogram of message CM
 - Transmit CK, CM
 - Decipher the message key with receiver private key: $AD(CK, KR)$
 - Obtain received key KM'
 - Decipher the message: $D(CM, KM')$
 - Obtain received message M'

Message Integrity Code

- Is it possible to detect changes to a message?
 - Using a hash function H and a secret K
 - Compute a value that can be used to detect changes in received message M'
- Function: MIC (Message Integrity Code)
 - With freshness, can be used to provide authenticity, so, very often, it is called a MAC (Message Authentication Code)

MIC in detail

- In cryptographic function notation:
 - $E(H(M), K)$ – digest the message and cipher result
 - Produces the MIC value
 - Transmit message M and MIC value
 - To verify:
 - Compute $E(H(M'), K)$ and compare with received MIC
 - Same? Then the message did not change
 - Another approach, using decryption:
 - Compute DT' from $D(MIC', K)$ and compare with $H(M')$
 - Same? Then the message did not change

HMIC

- HMIC stands for Hash-based Message Integrity Code
 - Also called HMAC
- Is another approach, without using ciphers
 - Better performance
- Function MIX combines the data with the secret
 - For example, with XOR or some specific concatenation
- How to use the HMIC?
 - Compute $H(MIX(M, K))$ and compare with $H(MIX(M', K))$
 - Same? Then the message did not change

Digital Signature

- Is it possible to detect changes to a message and confirm the sender?
 - Still using a hash function H but now with asymmetric keys K_R K_U
 - Compute a value that can be used to detect changes in received message M'
 - Function: DS (Digital Signature)

DS in detail

- In cryptographic function notation:
 - $\text{AE}(\text{H}(M), \text{KR})$ – digest the message and cipher result with the private key
 - Produces the DS value
 - Transmit message M and DS value
 - To verify:
 - Compare deciphered hash with recomputed hash
 - Compute $\text{AD}(\text{DS}', \text{KU})$ to obtain DT' and compare with $\text{H}(M')$
 - Same? Then the message did not change and was sent by a holder of the private key

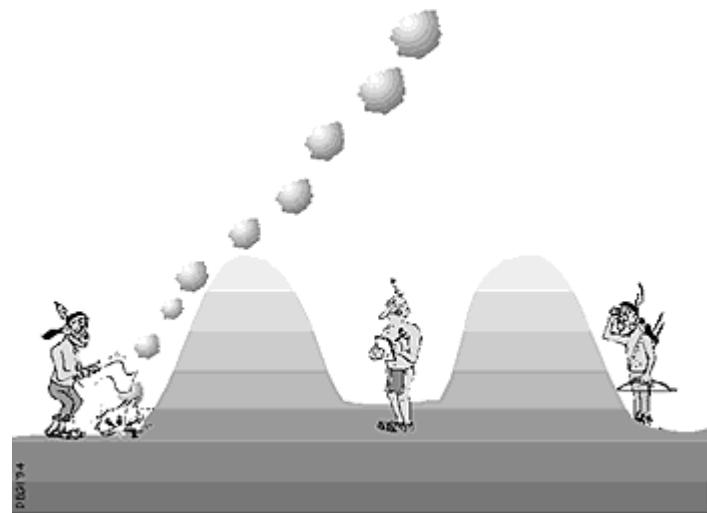
PROVIDING CRYPTOGRAPHIC SERVICES

Cryptographic services (revisited)

- We can now design cryptographic services:
 - Confidentiality
 - Integrity
 - Authenticity
- To protect against:
 - Unauthorized insertion of information
 - Loss of authenticity
 - Unauthorized modification of information in transit
 - Loss of integrity
 - Unauthorized replay of information
 - Loss of authenticity
 - Unauthorized access to information
 - Loss of confidentiality

1 - Confidentiality

- Use symmetric cipher
 - If a secret is shared
- Use asymmetric cipher
 - If public keys are shared
 - More efficient with hybrid cipher



Confidentiality in detail

- Alice wants to send a message to Bob that cannot be read by anyone else



Alice

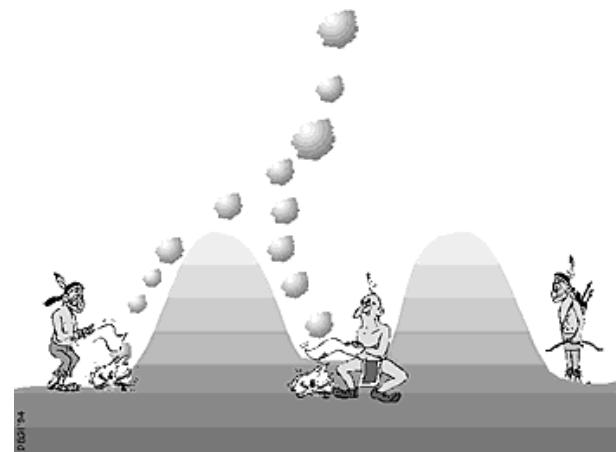


Bob

- How?
 - With shared secret key
 - E? K?

2 - Integrity

- Use MIC
 - If a secret is shared
- Use DS
 - If public keys are shared



Integrity in detail

- Alice wants to send a message to Bob that cannot be written by anyone else without detection



Alice

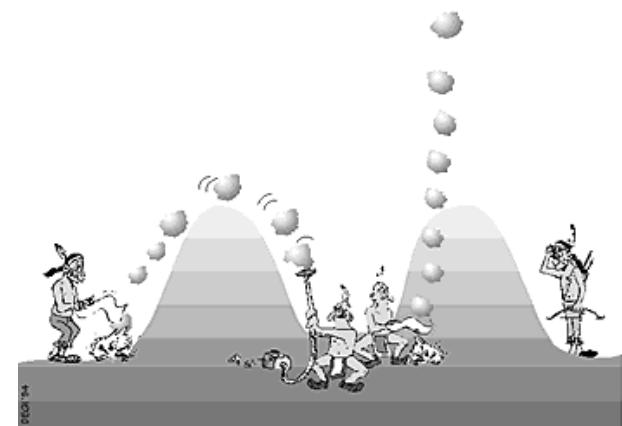


Bob

- How?
 - With shared public keys
 - AE? KRa? KRb?

3 - Authenticity

- *Integrity* assured with MIC or DS
- Freshness requires adding a nonce N to the message
 - Number used Once
 - Random number RN
 - But... receiver needs to memorize them to detect replays
 - Counter CTR
 - But... messages must be received in order
 - Timestamp TS
 - But... clocks must be synchronized
 - Combination of two of the above



Authenticity in detail

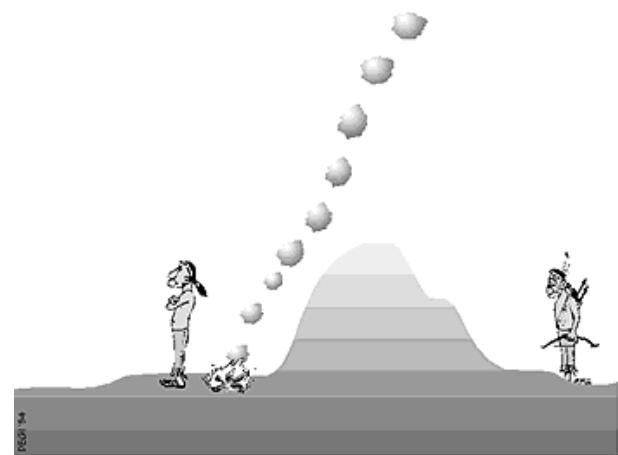
- Alice wants to send an authentic message to Bob



- How? Also need freshness
 - Add nonce
 - Which one?
 - AE? KR_a? KR_b?

Non-Repudiation

- A digital signature can provide non-repudiation, if...
 - the signer is only entity that knows the private key



Confidentiality + Authenticity?

- Alice wants to send a confidential and authentic message to Bob



Alice



Bob

- KUa? KUb?

Summary

- Cryptography allows us to protect information
 - With ciphered data to prevent reads
 - With digests to allow detection of writes
- We can use cryptographic functions to provide cryptographic services for:
 - Confidentiality
 - Integrity
 - Authenticity
- Next: cryptographic technology

Symmetric Ciphers

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Roadmap

- Introduction
- Symmetric Ciphers
- Hash Functions
- Message Integrity Codes

Ciphers: terminology

- Cipher
 - Specific cryptographic technique
- Cipher Procedure
 - Cipher: plaintext → cryptogram (aka ciphertext)
 - Decipher: cryptogram → plaintext
 - Algorithm: data transformation procedure
 - Key: algorithm parameter



Old (broken) ciphers

- Caesar cipher
 - Shift by 3
- Substitution cipher
 - $A \rightarrow C$
 - ...
- Vigenere cipher (1500s)
 - $+ \text{mod } 26$
- Rotor machines (1870-1943)
 - Single rotor: Hebern
 - 3-5 rotors: Enigma
- DES (Digital Encryption Standard) - 1974
 - 56 bits key, 64 bits block



Hebern machine



Enigma machine

Modern cipher types

- Regarding the procedure
 - Stream
 - Block
- Regarding the type of key
 - Symmetric (secret key, a shared secret)
 - Asymmetric (public key and private key)

Ciphers	Block	Stream
Symmetric	yes	yes
Asymmetric	yes	no

Roadmap

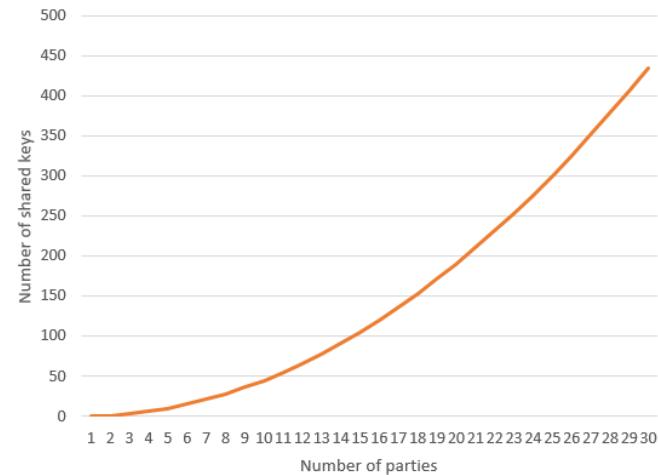
- Introduction
- **Symmetric Ciphers**
- Hash Functions
- Message Integrity Codes

Symmetric Ciphers

- Symmetric Ciphers
 - Stream ciphers
 - Block ciphers
 - DES
 - AES
 - Block cipher modes

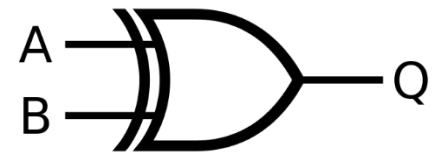
Symmetric Ciphers

- Secret key
 - Shared by 2 or more communicating parties
- Allow for
 - Confidentiality to all who possess the key
 - Message authentication
- Advantages
 - Performance (typically very efficient)
- Disadvantages
 - N communicating parties, 1 to 1 secretly
→ $N \times (N-1)/2$ keys
- Problems
 - Key distribution



Perfectly Secure Cipher: One-Time Pad

- Mauborgne/Vernam [1917]
- XOR (\oplus):
 - $0 \oplus 0 = 0$ $1 \oplus 0 = 1 \rightarrow a \oplus 0 = a$
 - $0 \oplus 1 = 1$ $1 \oplus 1 = 0 \rightarrow a \oplus 1 = \text{not } a$
 $a \oplus b \oplus b = a$
- Encrypt
 - $E(P, K) = P \oplus K = C$
 - P = plaintext; K = key
- Decrypt
 - $D(C, K) = C \oplus K = (P \oplus K) \oplus K = P$



Perfectly Secure Cipher: One-Time Pad

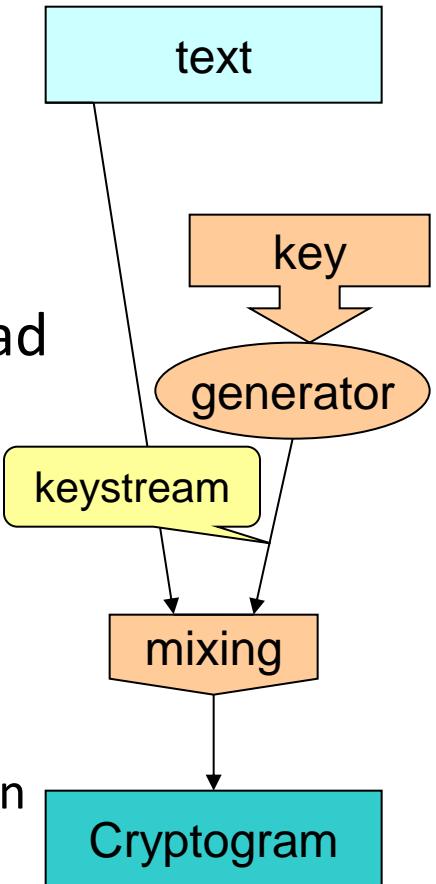
- One-Time Pad (XOR message with key)
- Example:
 - Message: ONETIMEPAD
 - Key: TBFRGFARFM
 - Ciphertext: IPKLPSFHGQ
 - The key TBFRGFARFM decrypts the message to ONETIMEPAD
 - The key POYYAEAAZX decrypts the message to SALMONEGGS
 - The key BXFGBMTMXM decrypts the message to GREENFLUID

One-Time Pad problems

- Security is based on the assumption that K is never reused
 - What if one has two encrypted messages:
$$C_1 = P_1 \oplus K \text{ and } C_2 = P_2 \oplus K$$
$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K$$
$$= P_1 \oplus P_2$$
- Need to generate truly random bit sequence
 - As long as all messages
- Need to securely distribute key bit sequence (!)

Stream ciphers

- Practical approximation to the One-Time Pad
- Keystreams generated in a deterministic way
 - From a fixed size key
 - Approximation to real random sequence generators
- Encryption and decryption as with a one-time pad
 - i.e., by doing XOR with the keystream (**mixing**)
- Practical aspects of stream ciphers security:
 - If the plain text is known, the keystream is exposed
 - The repetition of cycles (reuse of the keystream) facilitates cryptanalysis
 - if the cycle period or part of the plain text is known
 - Integrity control must exist
 - Easy to modify the cryptogram in a deterministic way

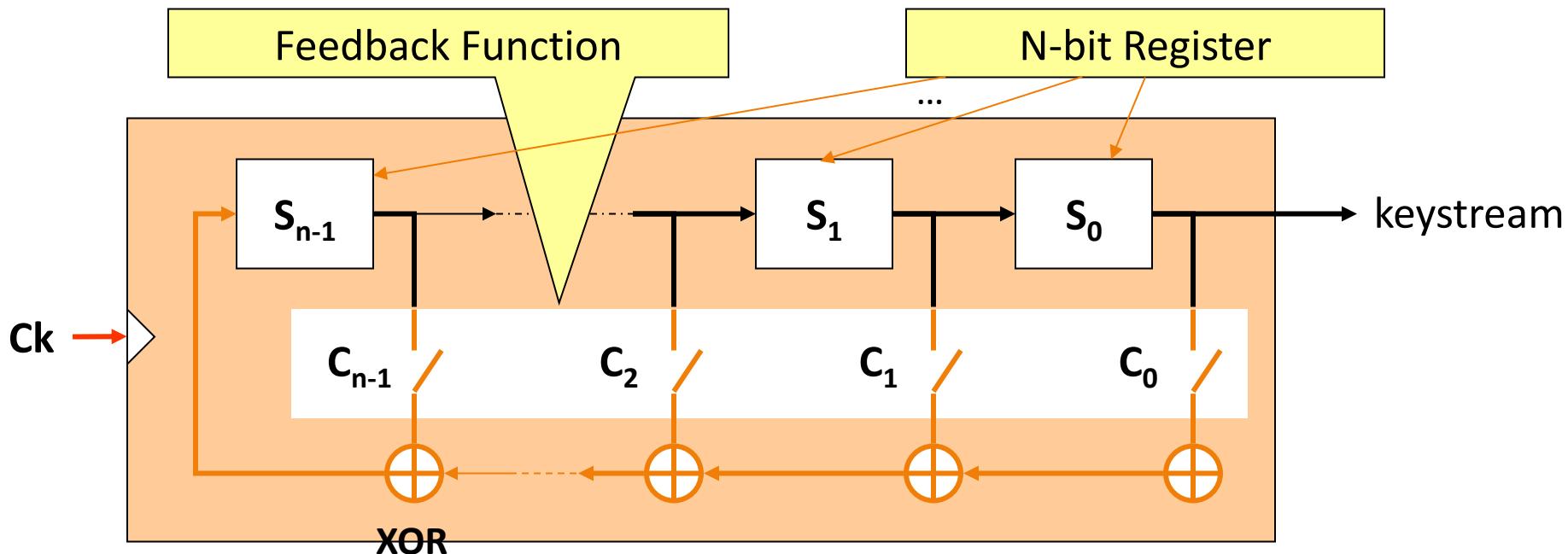


Symmetric stream ciphers

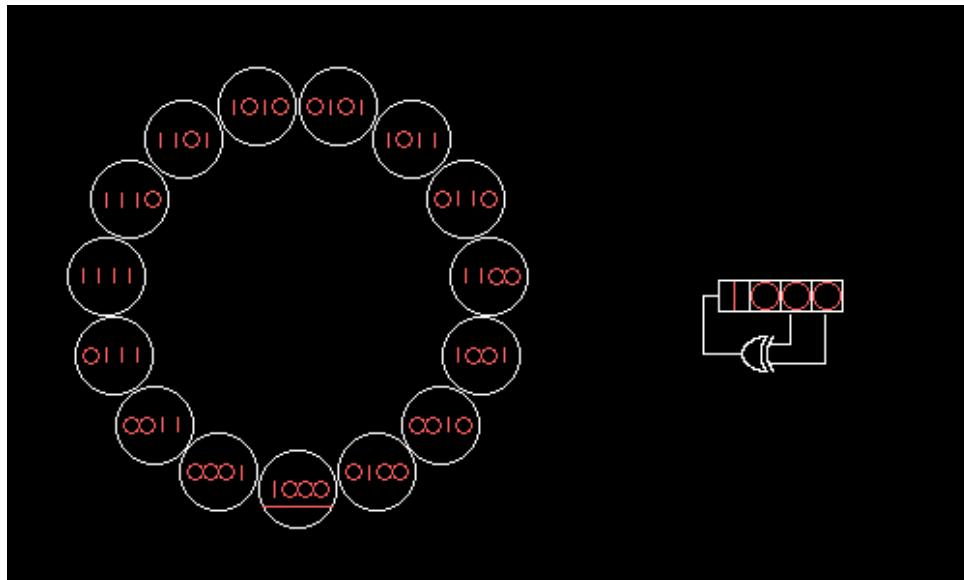
- Used approximations
 - Secure pseudo-random generators
 - Based on LFSRs (Linear Feedback Shift Registers) → next
 - Based on block ciphers → later
 - Other approximations (nonlinear functions, etc.)
 - Usually without self-synchronization
 - Receiver must know when encrypted data begins
 - Typically without the possibility of fast random access
- Most common algorithms
 - A5 (GSM)
 - RC4
 - SEAL (with fast random access)
 - ChaCha

Linear Feedback Shift Register (LFSR)

- State machine that produces a cyclic sequence of bits
 - The sequence depends on the **key** = initial state of the register
 - S_0, \dots, S_{n-1} = **register's bits**; C_0, \dots, C_{n-1} = **coefficients** of the function
 - Max. period of the cycle is $2^n - 1$



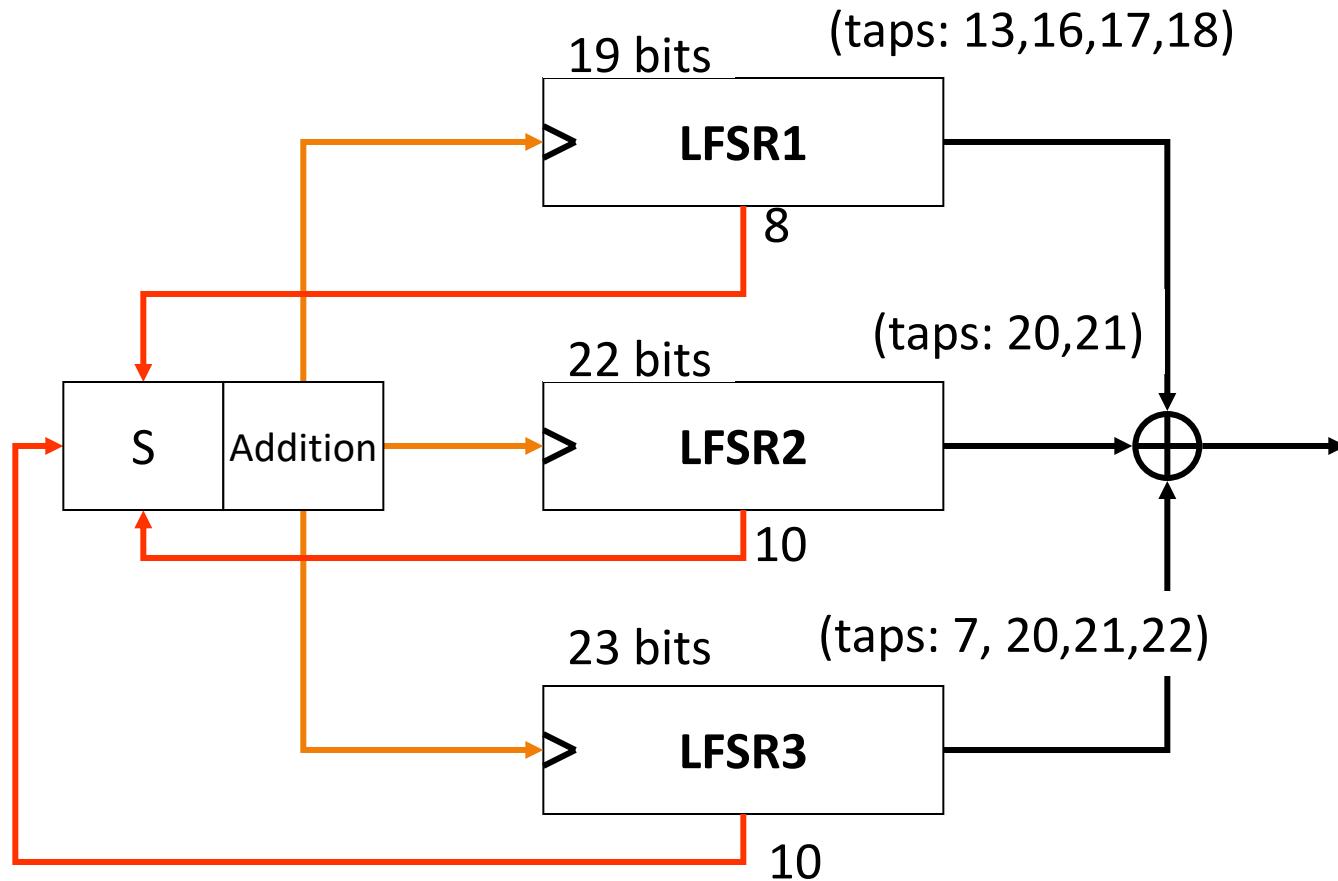
4 bit LFSR example



Bits 2 and 3 are **tapped**

Taps are the bits that affect the next state

LFSR structure: A5/1 (GSM)



Symmetric block ciphers

- Also based on approximations, using Shannon's notions of confusion and diffusion:
 - **Confusion**: repeated application of a complex function to a large block (e.g. 64 bits)
 - **Diffusion**: basic operations:
 - **Permutation**: exchange bits without losing or adding bits
 - **Substitution**: change bits for others using a substitution table
 - **Expansion**: introducing new bits
 - **Compression**: deleting some bits
- Some relevant symmetric encryption algorithms:
 - **DES** – Data=64; Key=56 – insecure, never use
 - **AES** – D=128; K=128, 192, 256 – current standard
 - Others (IDEA, Blowfish, CAST, RC5, etc.)

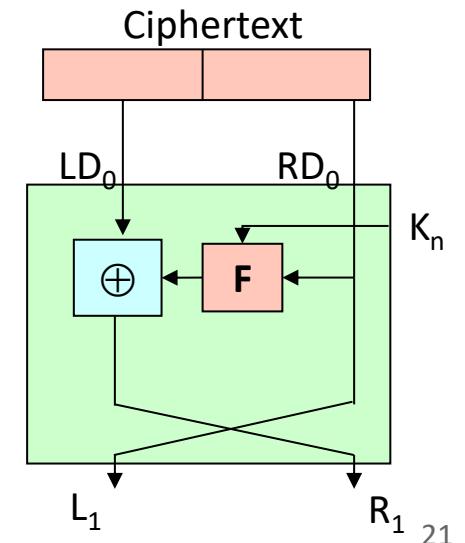
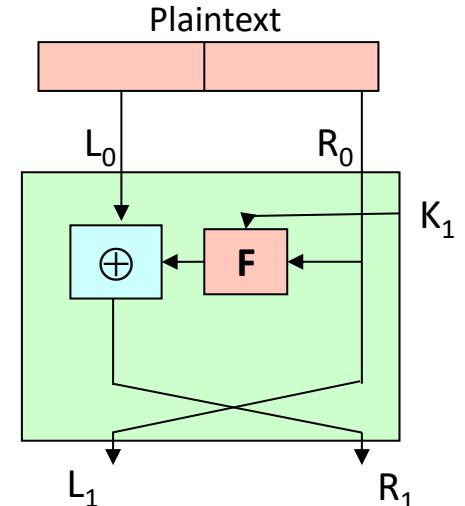
DES Avalanche

Input:	*	1
Permuted:*	.	1
Round 1:*	.	1
Round 2:	*...*...*	*	5
Round 3:	*...*.*.*...*	**	18
Round 4:	..*....*	***	28
Round 5:	*...**...*	***	29
Round 6:*...*	***	26
Round 7:	*****...***	***	
Round 8:	*...*...*	***	
Round 9:	***...***	***	
Round 10:	*...*...*	***	
Round 11:*****	***	
Round 12:	*...***...*	***	
Round 13:	**...*...*	***	
Round 14:	*...**...*	***	
Round 15:	**...*...*	***	
Round 16:*...*	***	
Output:*...*	***	~50%

* is a changed bit that propagates changes

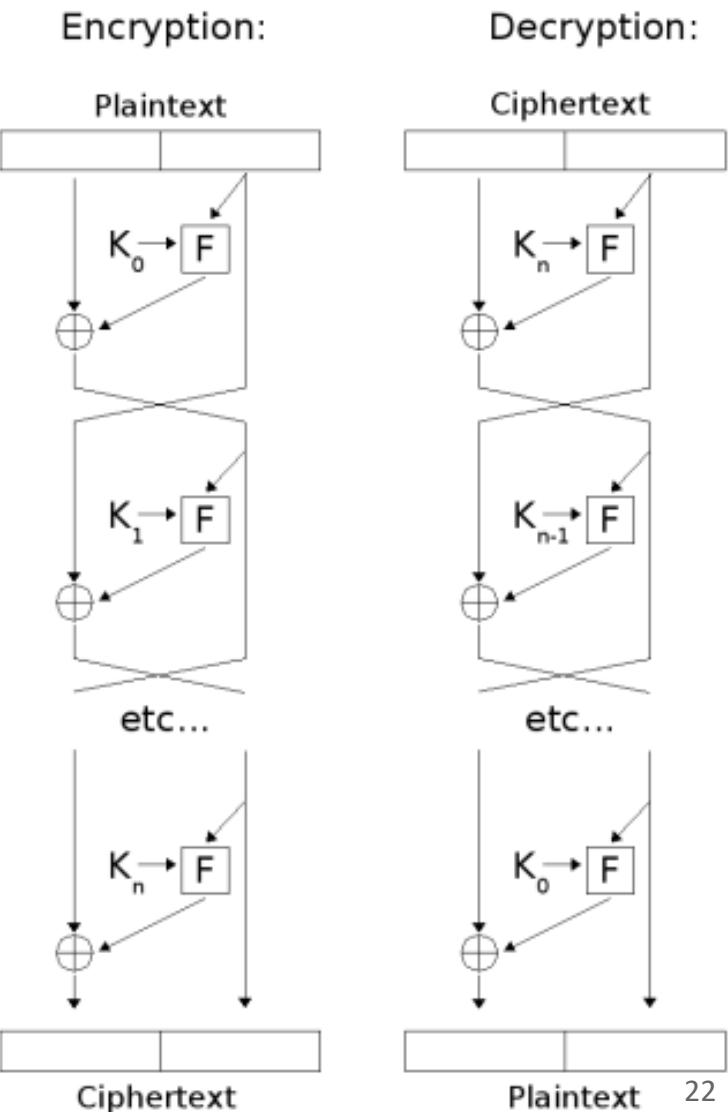
Feistel Network

- Complex function most commonly used in block cipher algorithms
- Applies a **round function (F)** over multiple rounds
 - F can be a pseudo-random generator
 - Each round uses a different **round key (K_i)**
 - Round keys are obtained from the key (K)
 - Text is split in **left (L)** and **right (R)** parts



Feistel Network

- Cipher and decipher processes are the same
 - Keys are used in the inverse order



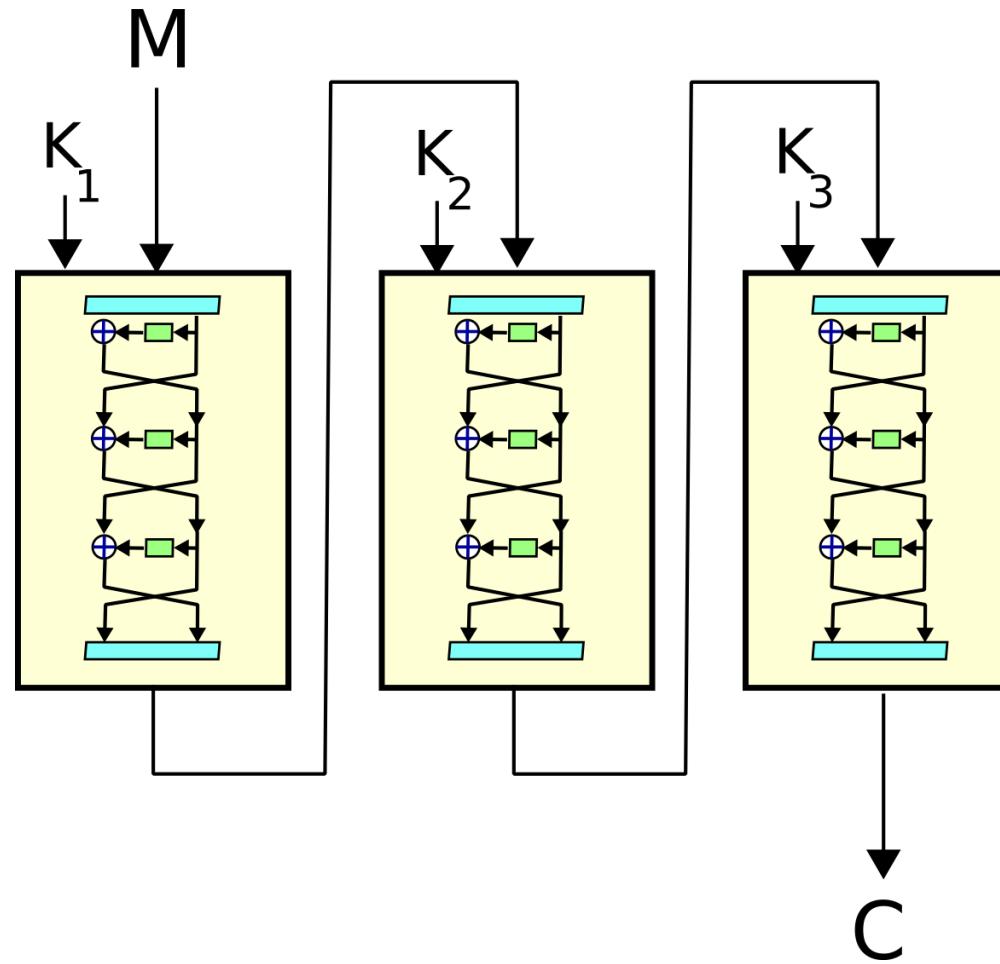
DES Algorithm

- DES:
 - 64-bit blocks
 - Key is 56 bits
- Possible key combinations
 - $2^{56} = 7.2 \times 10^{16} = 72,000,000,000,000,000$
 - Try 1 per second = 9,000 Million years to search the entire key space
- Distributed attacks on DES:
 - RSA's DES challenges:
 - 1997: 96 days (using 70,000 machines)
 - 1998: 41 days (distributed.net)
 - 2008: less than 1 day (array of 128 Spartan FPGAs ~ €5k)
 - 2016: about 15 days on a standard PC with a GPU (~€800)

Block ciphers: reinforcement

- Multiple ciphers
 - Double cipher
 - Breakable by brute force in max. 2^{n+1} attempts instead of expected 2^{2n}
 - Triple cipher, typ. EDE = Encrypt-Decrypt-Encrypt – 3DES-EDE
 - $C_i = E_{K1}(D_{K2}(E_{K3}(P_i)))$
 - $P_i = D_{K3}(E_{K2}(D_{K1}(C_i)))$
 - Typically, $K_1=K_3$ is used
 - Effective key size = 56 + 56 bits = 112 bits
- Key whitening (DESX)
 - DES-X(P_i) = $K_2 \oplus \text{DES}_K(K_1 \oplus P_i)$
 - $C_i = E_K(K_1 \oplus P_i) \oplus K_2$
 - $P_i = K_1 \oplus D_K(K_2 \oplus C_i)$

3DES – Triple DES / TDEA – Triple Data Encryption Algorithm



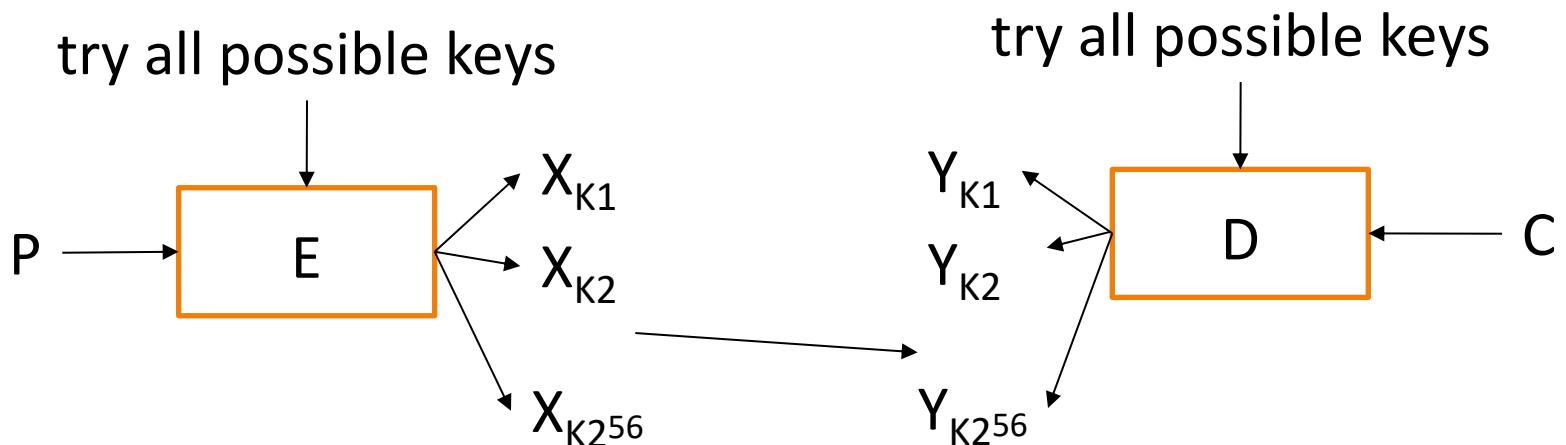
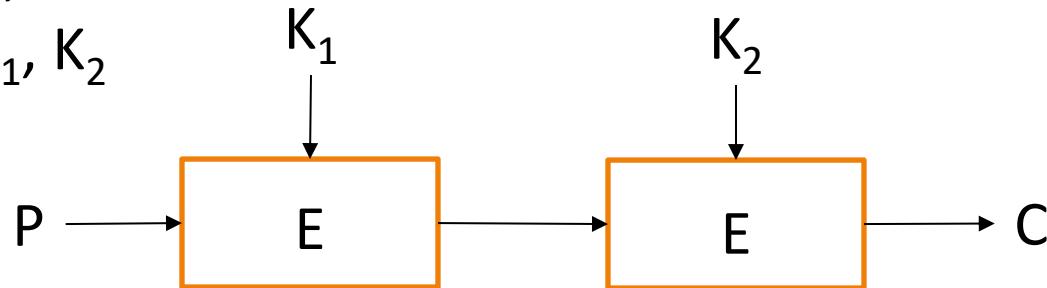
Meet-in-the-middle attack

- Double DES
 - $C = E_{K2}(E_{K1}(P))$
 - Effective key size of Double DES?
 $= 2^{56} * 2^{56} = 2^{112}$ **WRONG!**

Meet-in-the-middle attack

a known-plaintext attack

Attacker knows P, C
does not know K_1, K_2



One $X_{Ki} = Y_{Kj}$ means $K_1 = K_i$ and $K_2 = K_j$
so maximum effort is $2 * 2^{56}$ tries

Meet-in-the-middle attack

- Double DES
 - $C = E_{K2}(E_{K1}(P))$
 - Effective key size of Double DES?
 $= 2^{56} * 2^{56} = 2^{112}$ **WRONG!**
- Meet-in-the-Middle Attack
 - $C = E_{K2}(E_{K1}(P))$
 - $X = E_{K1}(P) = D_{K2}(C)$
 - Brute force attack (given one P/C pair):
 1. calculate $E_{K1}(P)$ for all keys (2^{56} work)
 2. calculate $D_{K2}(C)$ for all keys (2^{56} work)
 3. the match gives the keys
 - Total work = $2 * 2^{56} = 2^{57}$

Cipher modes

- Problem
 - How to use a block cipher with a fixed block size with plaintext of a different size?
 - Plaintext may be much larger than the block size
 - Size of the plaintext may not be a multiple of the size of the block
- The problem is solved by **cipher modes**

Block cipher modes

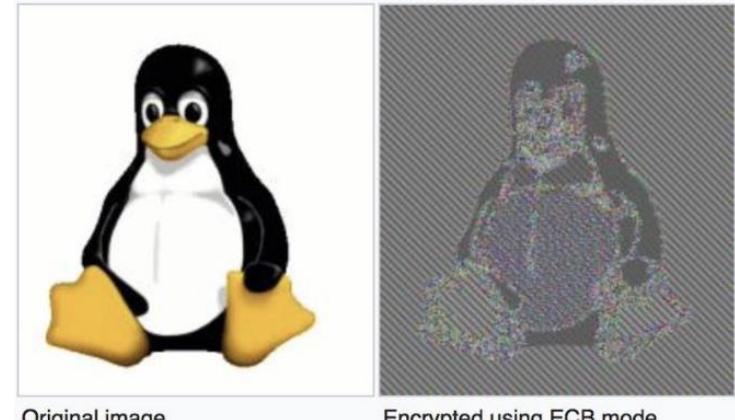
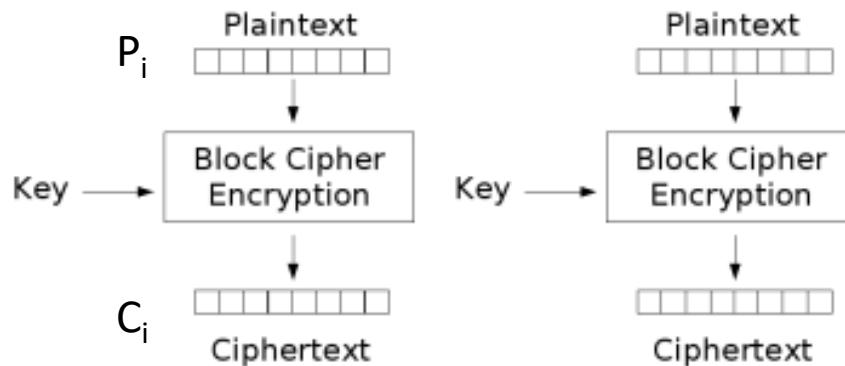
- Initially proposed for DES
 - ECB (Electronic Code Book)
 - CBC (Cipher Block Chaining)
 - OFB (Output FeedBack mode)
 - CTR (CounTeR mode)
 - GCM (Galois Counter Mode)
- Final sub-block processing
 - Alignment with padding
 - Augments the cryptogram by adding padding to the plaintext

Block cipher modes: ECB

ECB (Electronic Code Book)

$$C_i = E_k(P_i)$$

$$P_i = D_k(C_i)$$



- **Strength:**
 - Simple
- **Weakness:**
 - Repetitive information in the plaintext may show in the ciphertext, if aligned with blocks
 - If same message is encrypted with the same key and resent, their cipher texts are the same
- Typical use: sending short pieces of data, e.g., encryption key

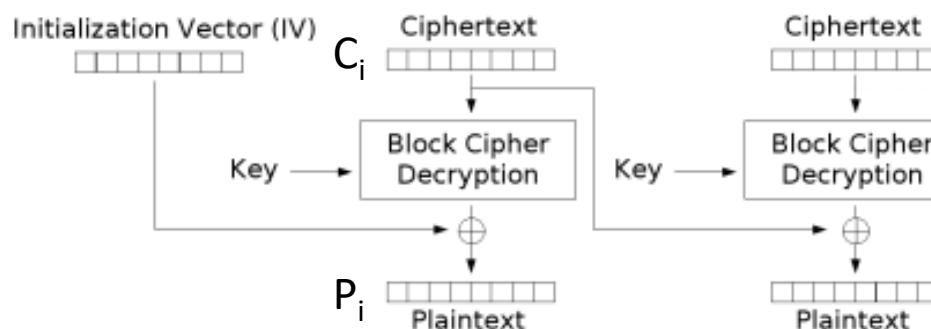
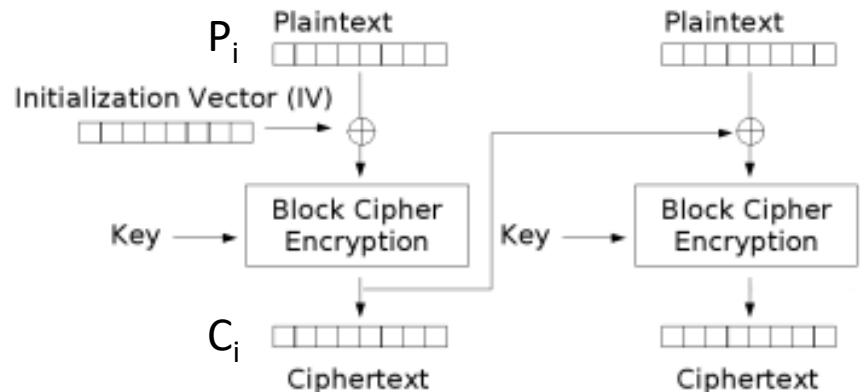
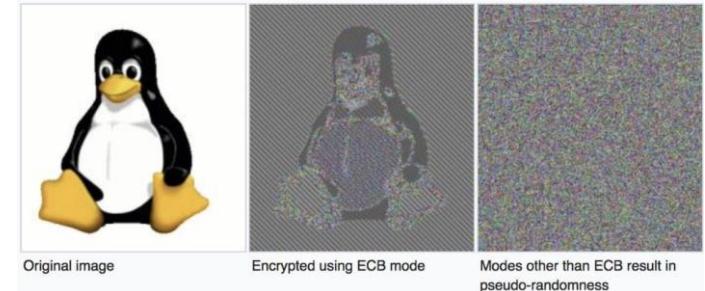
Block cipher modes: CBC

CBC (Cipher-Block Chaining)

$$C_i = E_k(P_i \oplus C_{i-1})$$

$$P_i = C_{i-1} \oplus D_k(C_i)$$

$$C_0 = IV$$



- **Strength:**
 - Repeated plaintext blocks result in different ciphered blocks
 - A ciphertext block depends on **all blocks** before it
- **Weakness:**
 - More complex
 - A corrupted bit in the ciphertext will affect all bits in its block and **one bit** in the next block

Block cipher modes: OFB / CTR

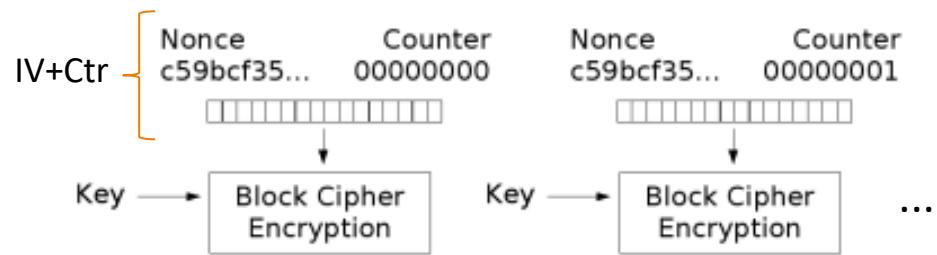
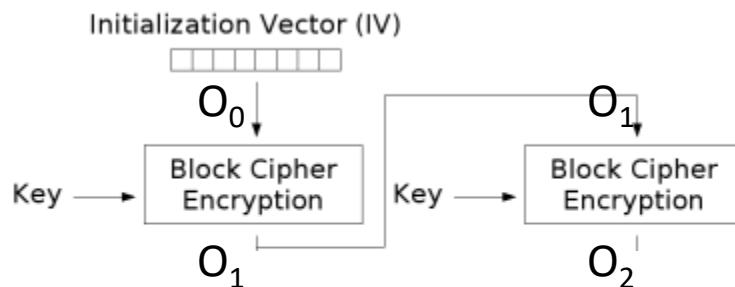
OFB (Output FeedBack)

$$\begin{aligned}C_i &= P_i \oplus O_i \\P_i &= P_i \oplus O_i \\O_i &= E_k(O_{i-1}) \\O_0 &= IV\end{aligned}\quad \text{Transforms block cipher into stream cipher}$$

CTR (CounTer) mode or

ICM (Integer Counter Mode)

$$\begin{aligned}C_i &= P_i \oplus E_k(IV + Ctr_i) \\P_i &= C_i \oplus E_k(IV + Ctr_i) \\Ctr_i &= Ctr_i + 1\end{aligned}\quad \text{Transforms block cipher into stream cipher}$$



Block cipher modes: OFB / CTR

OFB (Output FeedBack)

$$\begin{aligned} C_i &= P_i \oplus O_i \\ P_i &= P_i \oplus O_i \\ O_i &= E_k(O_{i-1}) \\ O_0 &= IV \end{aligned}$$

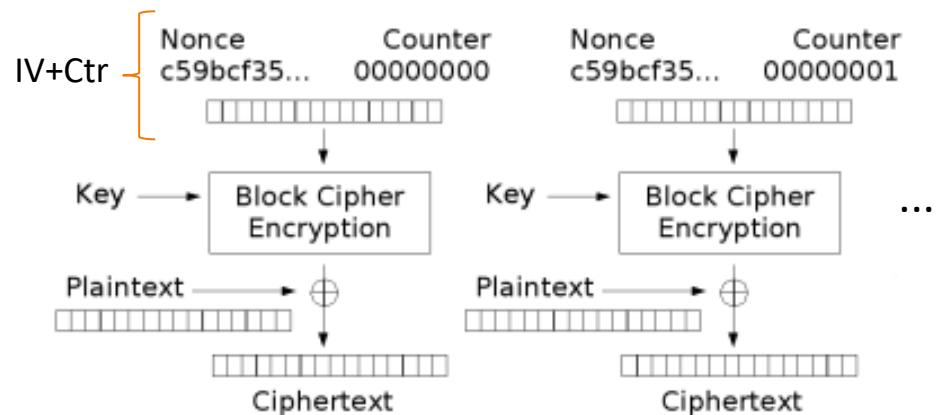
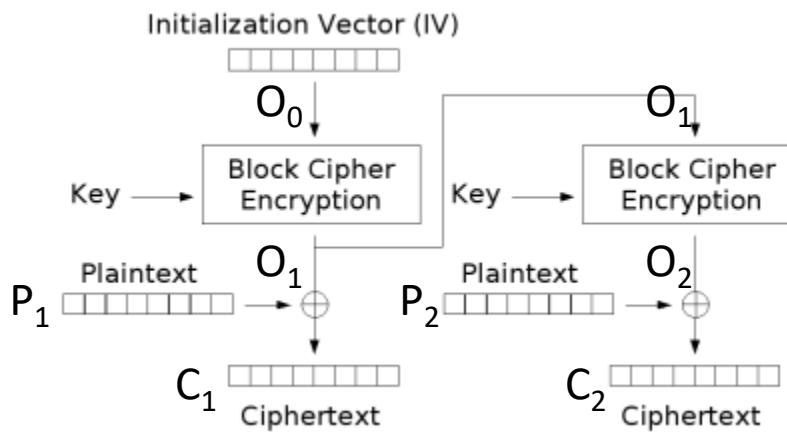
Transforms block cipher into stream cipher

CTR (CounTer) mode or

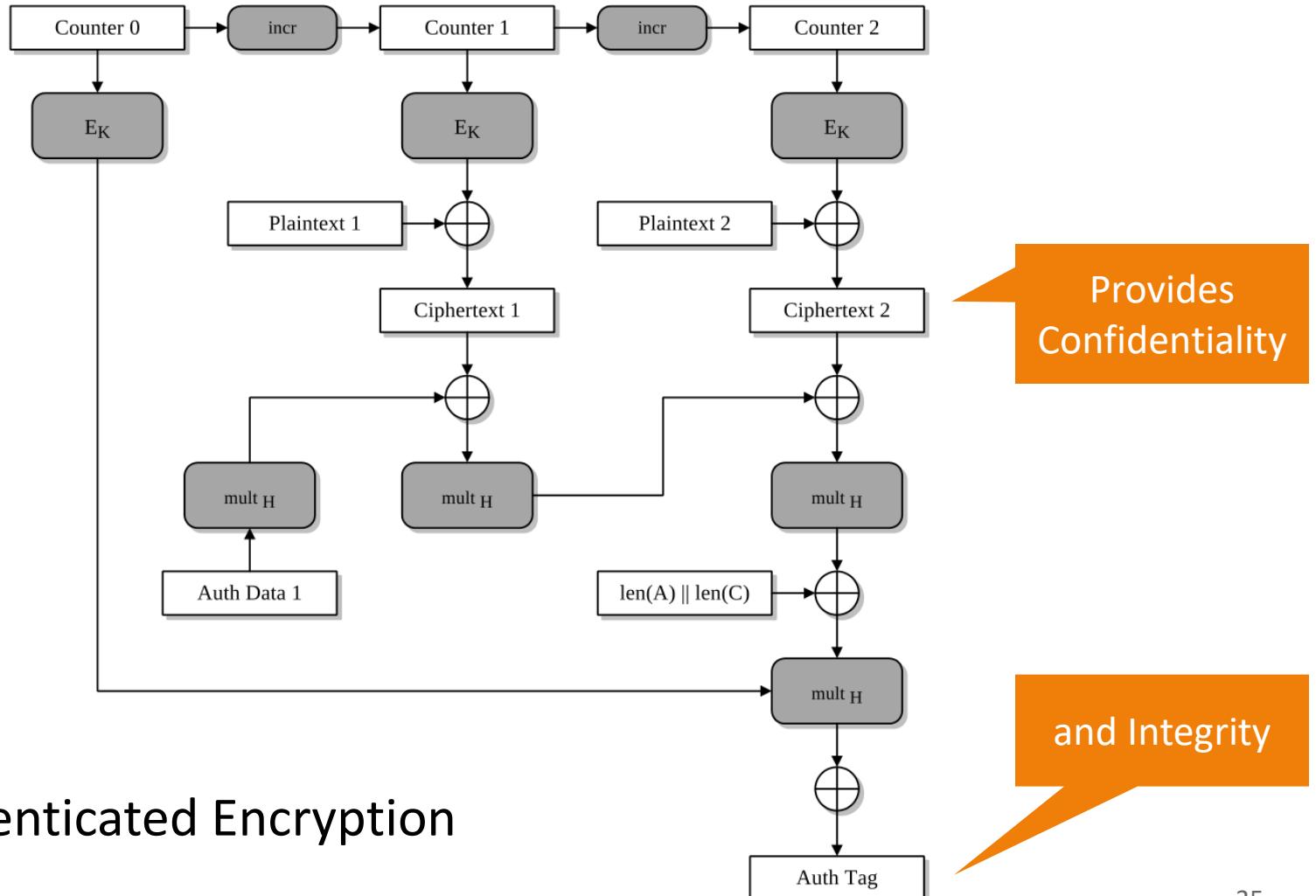
ICM (Integer Counter Mode)

$$\begin{aligned} C_i &= P_i \oplus E_k(IV + Ctr_i) \\ P_i &= C_i \oplus E_k(IV + Ctr_i) \\ Ctr_i &= Ctr_i + 1 \end{aligned}$$

Transforms block cipher into stream cipher



Block cipher modes: GCM Galois Counter Mode



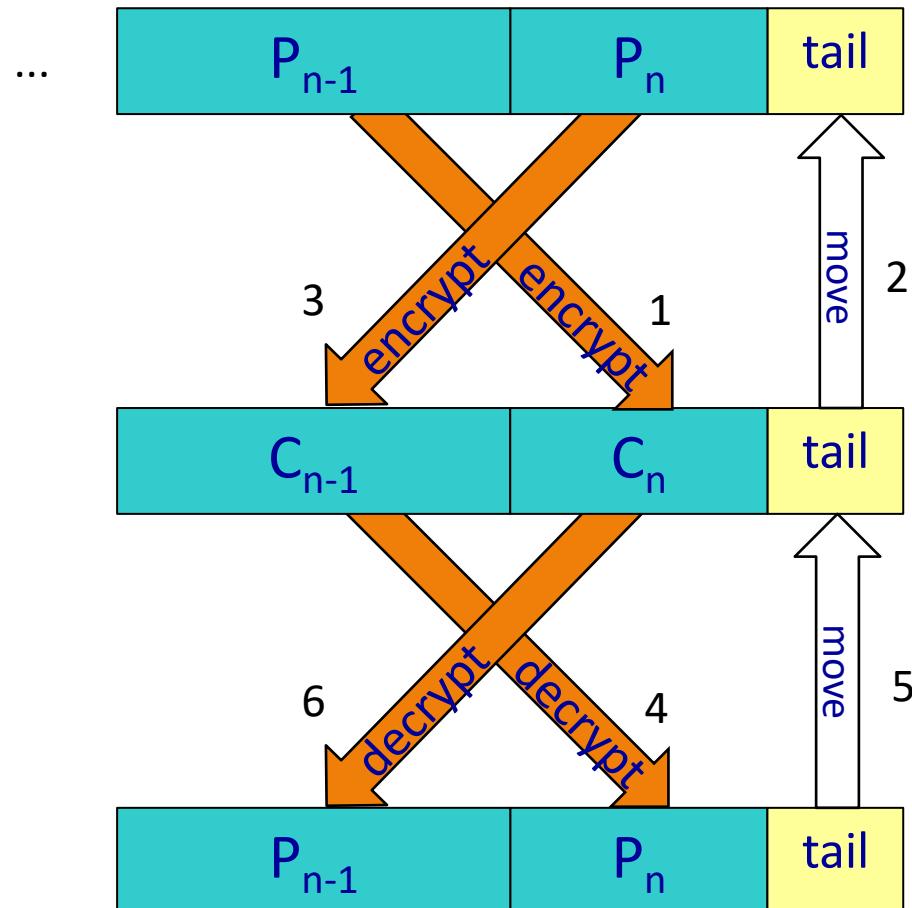
Padding Schemes

- $M \bmod B \neq 0 \Rightarrow$ **padding** of last block (M=message size; B=block size)
- Options:
 - Pad with zero (null) bytes, spaces (0x20), all bytes of the same value
 - Pad with random bits
 - Pad with 0x80 (1000 0000) followed by zero (null) characters
 - **PKCS#5 scheme**
 - Sequence of bytes, each of which equal to the number of padding bytes
 - Example: if 24 bits of padding need to be added, the padding string is "03 03 03" (3 bytes times 8 bits equals 24 bits)



Cipher Text Stealing: ECB mode

- Avoids the need of padding
- Plays with the last 2 blocks
- Benefit: no need to send the padding bits!

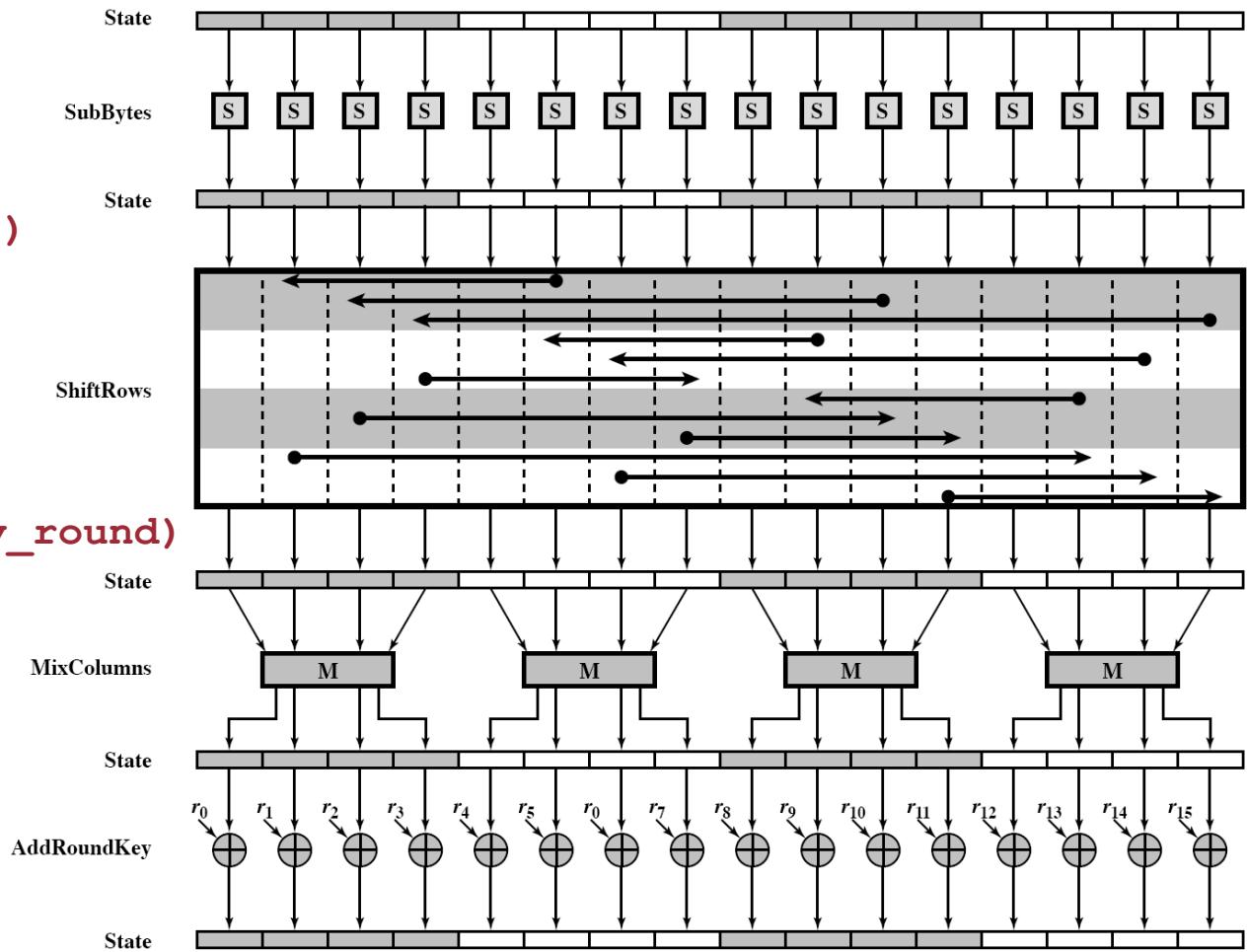


AES – current encryption standard

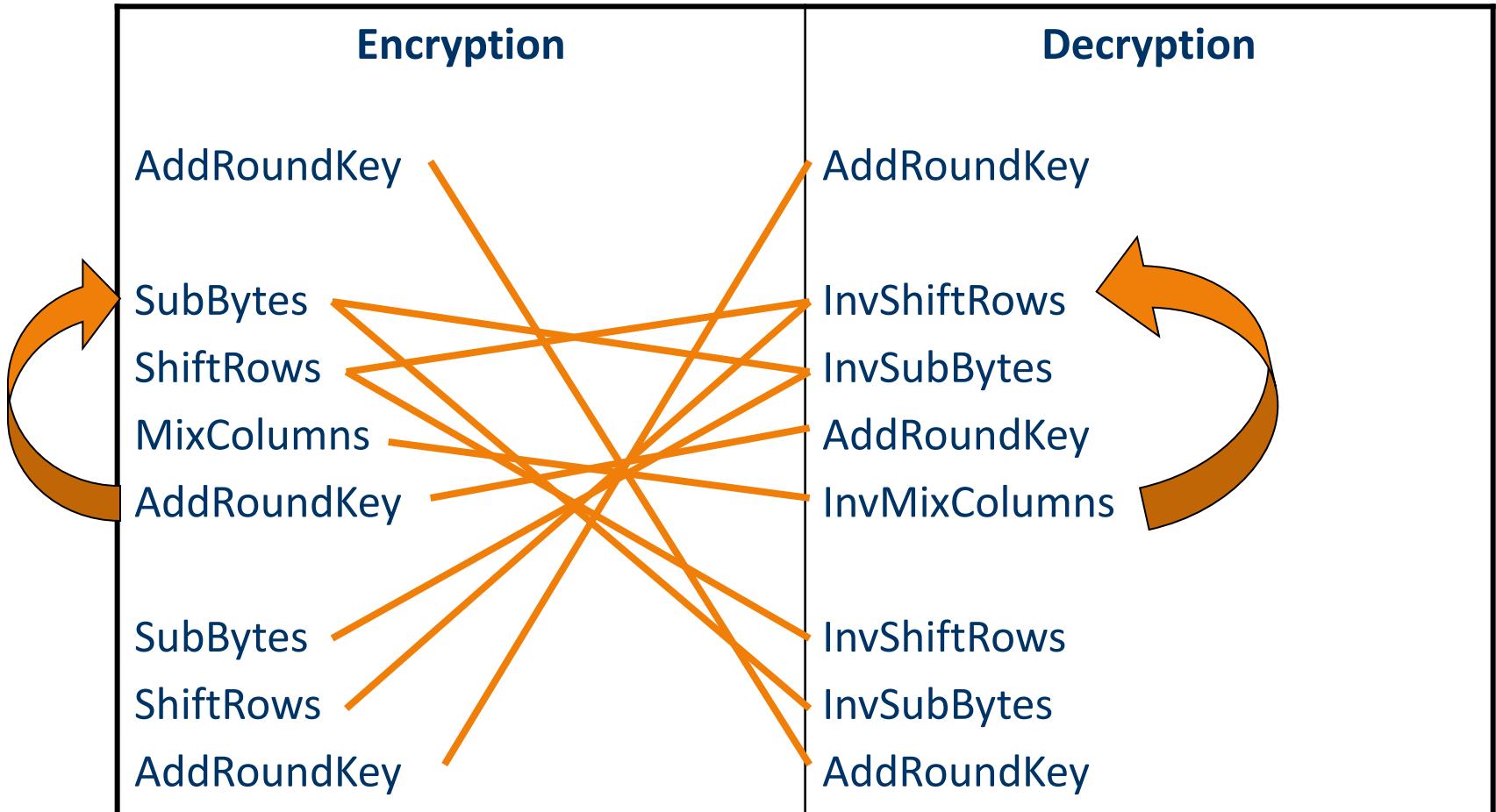
- AES (Advanced Encryption Standard) parameters:
 - Key size: 128, 192, 256 bits
 - Input block length: 128 bits
- Runs 10/12/14 rounds in which it:
 - substitutes bytes (one S-box used on every byte)
 - shifts rows (permute bytes between columns)
 - mixes columns (substitute using column multiplication)
 - adds round key ($\text{XOR } \textit{state}$ with key material)
 - round key size: 128
 - With fast XOR & table lookup implementations

AES - Main Encryption Round

```
state = input  
  
AddRoundKey (Key0)  
  
for #rounds{  
    SubBytes ()  
    ShiftRows ()  
    MixColumns ()  
    AddRoundKey (Key_ round)  
}  
  
out = state
```



AES – Encrypt vs Decrypt



Which algorithm / key size to use?

NIST Special Publication 800-131A
Revision 2

Transitioning the Use of Cryptographic Algorithms and Key Lengths

Elaine Barker
Allen Roginsky

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-131Ar2>

C O M P U T E R S E C U R I T Y



March 2019

**Table 1: Approval Status of Symmetric Algorithms Used for
Encryption and Decryption**

Algorithm	Status
Two-key TDEA Encryption	Disallowed
Two-key TDEA Decryption	Legacy use
Three-key TDEA Encryption	Deprecated through 2023 Disallowed after 2023
Three-key TDEA Decryption	Legacy use
SKIPJACK Encryption	Disallowed
SKIPJACK Decryption	Legacy use
AES-128 Encryption and Decryption	Acceptable
AES-192 Encryption and Decryption	Acceptable
AES-256 Encryption and Decryption	Acceptable

Roadmap

- Introduction
- Symmetric Ciphers
- **Hash Functions**
- Message Integrity Codes

Hash functions / message digests

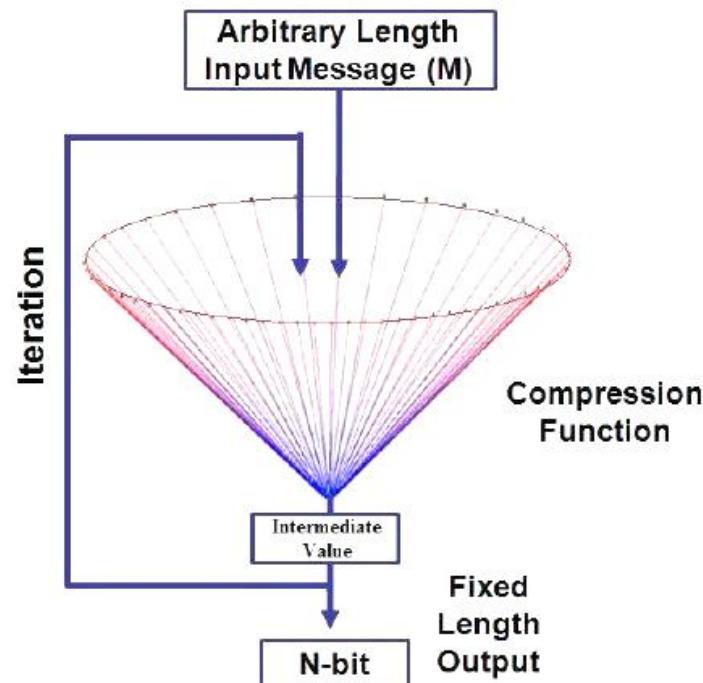
- Aka **cryptographic hash functions** or **collision-resistant hash functions**
 - Do not confuse with hash functions used in hash tables (that you learned in IAED)
- They are cryptographic, but not ciphers
 - not used for encryption

Hash functions properties

- Generate very different output values – **hash** value – for similar inputs
- One-way (non-invertible):
 - Collision resistance
 - Computationally infeasible to find two inputs that give the same hash
 - Preimage resistance (strong collision resistance)
 - Given a hash, it's computationally infeasible to find an input that produces that hash
 - 2nd preimage resistance (weak collision resistance)
 - Given a hash value and the corresponding input, it's computationally infeasible to find a second input that generates that same hash

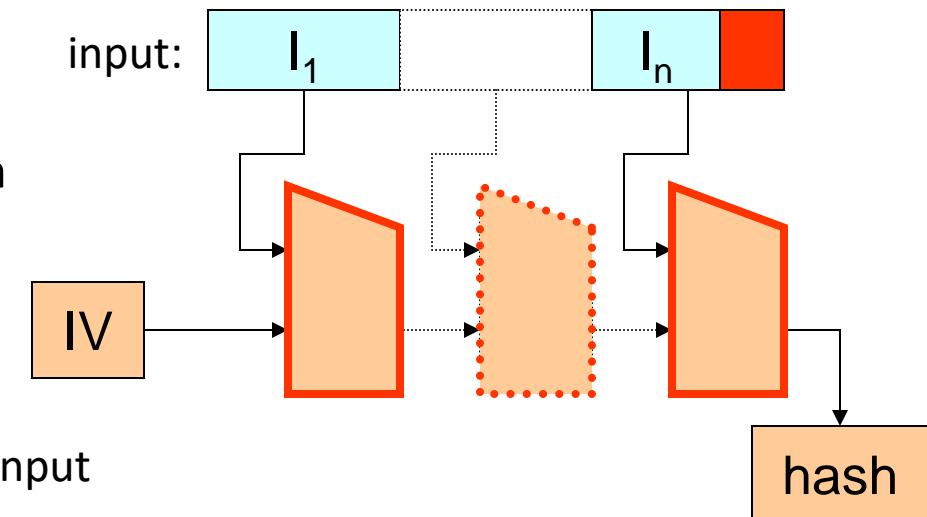
Hash functions

- Generate a fixed size value based on arbitrary size input
 - Iterative usage of a compression function with a fixed parameter input
 - The input text is aligned to the input blocks



Hash functions

- Some mechanisms used:
 - Shannon's diffusion & confusion
 - Iterative compression
 - MD Strengthening
 - Padding with 10000....000
 - Plus the number of bits of the input

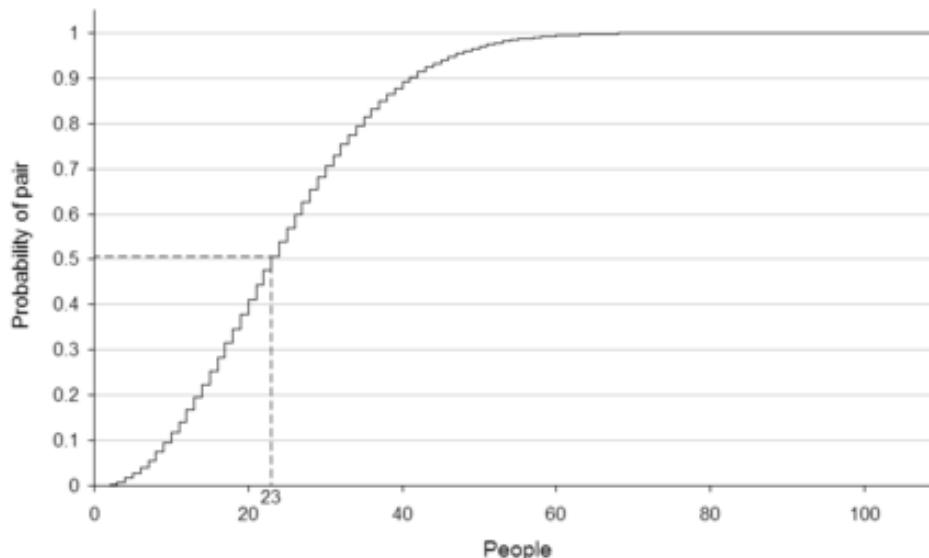


Hash functions

- Most used algorithms
 - Brute force attacks find collisions in $2^{m/2}$ tries, where m is the n. of bits
 - **MD5** (128 bits) *Very weak*
 - Collisions found in 2^{39} tries << 2^{64}
 - **SHA-1** (Secure Hash Algorithm, 160 bits) *Weak*
 - Collisions found in 2^{63} tries << 2^{80}
 - **SHA-2** (256 to 512 bits) *Ok*
 - Collisions found in 2^{128} or 2^{256} tries (secure, for now)
 - **SHA-3** (256 to 512 bits) *Ok*

Birthday paradox

- For there to be a 50% chance that someone in a room shares your birthday, you need 253 people
- However, for 50% chance that any two people in the room have the same birthday, you only need 23 people
 - It only takes 23 people to form 253 pairs when cross-matched



Birthday paradox

- Paradox
 - $P(me)$: probability n students having the same birthday as me (“collision”)
 - Probability A’s birthday date is the same as mine = $1/365 \approx 0.003$
 - Probability A’s birthday date different from mine = $1 - 1/365 \approx 0.997$
 - Probability A and B’s dates different from mine = $(1 - 1/365)^2 \approx 0.994$
 - $P(me) = 1 - (1 - \frac{1}{365})^n$
 - $P(me) > 0,5 \Leftrightarrow n \geq 253$
 - $P(pair)$: probability of pair with the same birthday
 - $P(pair) = 1 - (1 - \frac{1}{365})(1 - \frac{2}{365}) \dots (1 - \frac{n-1}{365})$
 - $P(pair) > 0,5 \Leftrightarrow n \geq 23$!!!

Hash functions attacks

- Objective is to find a **collision**
- **Brute forcing**
 - Attack: pick $H(M)$ and see if it's equal to $H(M')$, $H(M'')$, $H(M''')$, ...
 - $P(\text{collision}) > 0.5 = 2^m/2 = 2^{m-1}$
 - where m is the number of bits of the hash function
- **Birthday attack**
 - Allows finding collision much faster
 - Attack: pick $M, M', M'', M''' \dots$ and obtain hashes until any 2 are identical
 - Finding 2 messages with $H(M) = H(M')$:
 - $P(\text{collision}) > 0.5 \approx 2^{m/2}$ tries (only)
- Cryptanalysis leads to even faster attacks

SHA-1 is no longer secure

SHAttered

The first concrete collision attack against SHA-1
<https://shattered.io>

CWI

Marc Stevens
Pierre Karpman

Google

Elie Bursztein
Ange Albertini
Yarik Markov

SHAttered

The first concrete collision attack against SHA-1
<https://shattered.io>

CWI

Marc Stevens
Pierre Karpman

Google

Elie Bursztein
Ange Albertini
Yarik Markov

Here are two different PDF files,
but with the same SHA-1 hash

<https://shattered.io/>

Transitioning the Use of Cryptographic Algorithms and Key Lengths

Elaine Barker
Allen Roginsky

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-131Ar2>

C O M P U T E R S E C U R I T Y



Table 8: Approval Status of Hash Functions

Hash Function	Use	Status
SHA-1	Digital signature generation	Disallowed, except where specifically allowed by NIST protocol-specific guidance.
	Digital signature verification	Legacy use
	Non-digital-signature applications	Acceptable
SHA-2 family (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256)	Acceptable for all hash function applications	
SHA-3 family (SHA3-224, SHA3-	Acceptable for all hash function applications	

Roadmap

- Introduction
- Symmetric Ciphers
- Hash Functions
- **Message Integrity Codes**

Message Integrity Codes

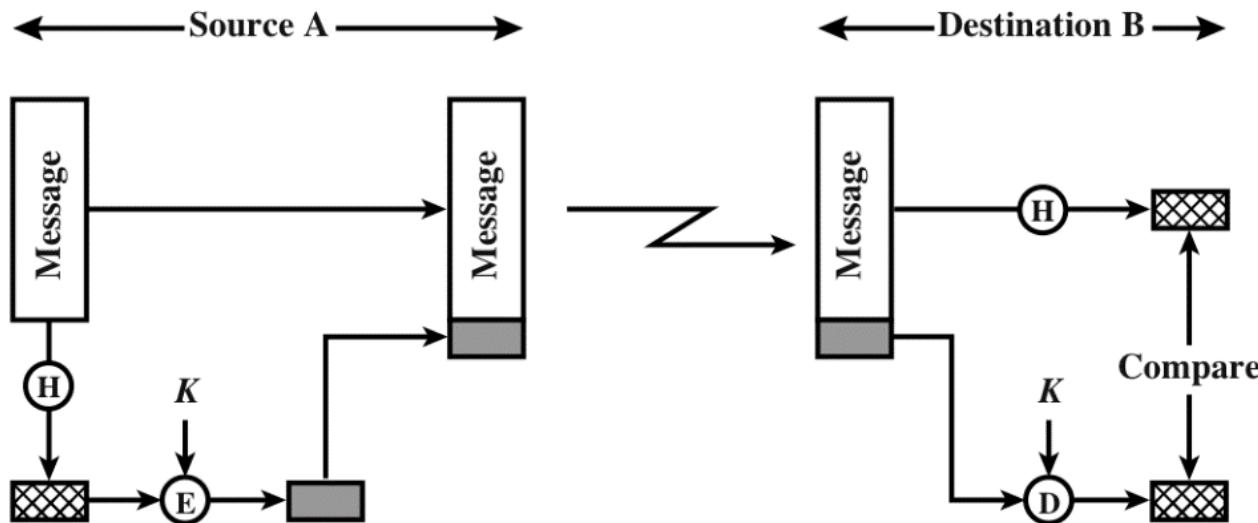
- Objective: detect changes to a message
 - Allows checking its integrity
 - With freshness, can provide authenticity
 - So, many times, called a MAC (Message Authentication Code)
- Assumption: sender and recipient have a **shared secret key K**
- Idea:
 - Send the message + MAC
 - If the message (or the MAC) is modified by an attacker, the recipient will be able to detect it
 - Attacker cannot create a valid MIC because he does not have K
- What about CRCs and checksums?
 - Attacker can modify the message and CRC/checksum accordingly, as he knows the algorithm and there is no secret involved

Message Authentication Code (MAC)

- MAC is a hash generated using a secret key
- Implementation alternatives:
 - 1: Hash the message and encrypt the digest
 - For example, with a symmetric block cipher
 - 2: Using a keyed-function
 - ANSI X9.9 (a.k.a. DES-MAC) with DES-CBC (64 bits) – now using AES
 - Authenticated Encryption
 - Encrypts the data and generates an authentication Tag, e.g GCM
 - 3: Using a keyed-hash
 - Hash the message along with a shared key
 - Keyed-MD5 (128 bits)
 - HMAC (size of the used hash function)

Message Authentication Code (MAC) – 1: hash and encrypt

- Hash the message and encrypt the digest

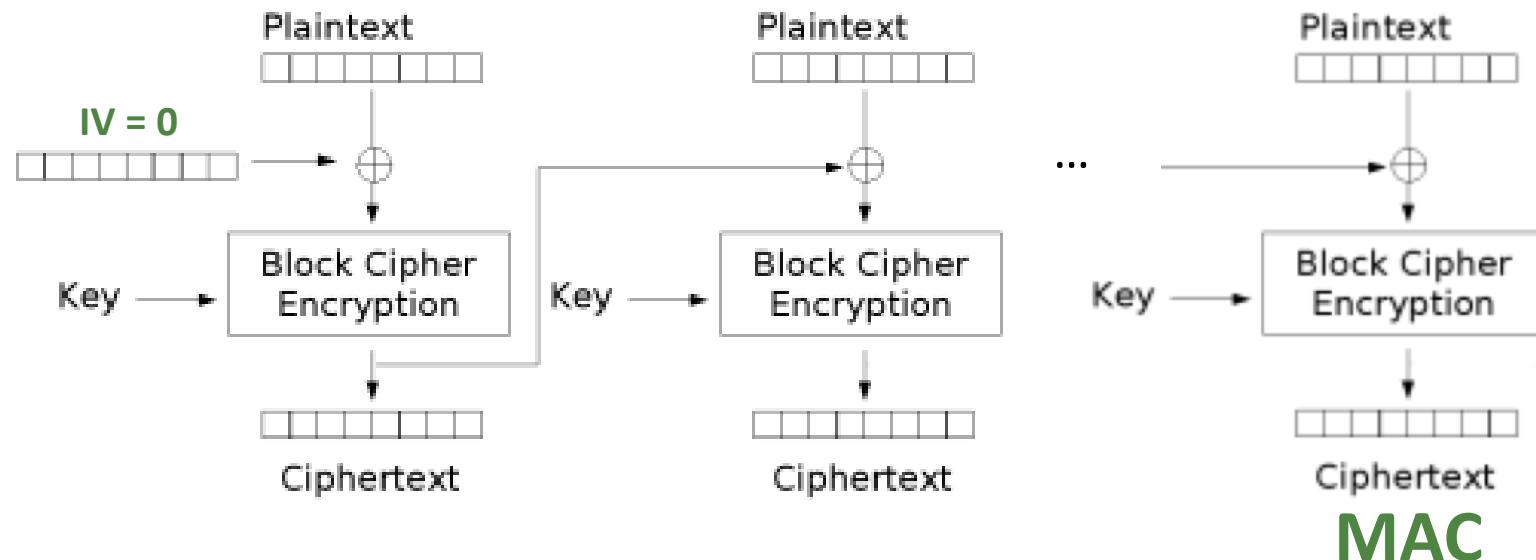


If attacker modifies the message (or the hash), the hash calculated at the destination will not match the hash received

Message Authentication Code (MAC)

2: keyed function

- CMAC (Cipher-based MAC)
 - Use a block cipher in CBC mode
 - MAC is the last block
 - Must be CBC for MAC to depend on the whole message

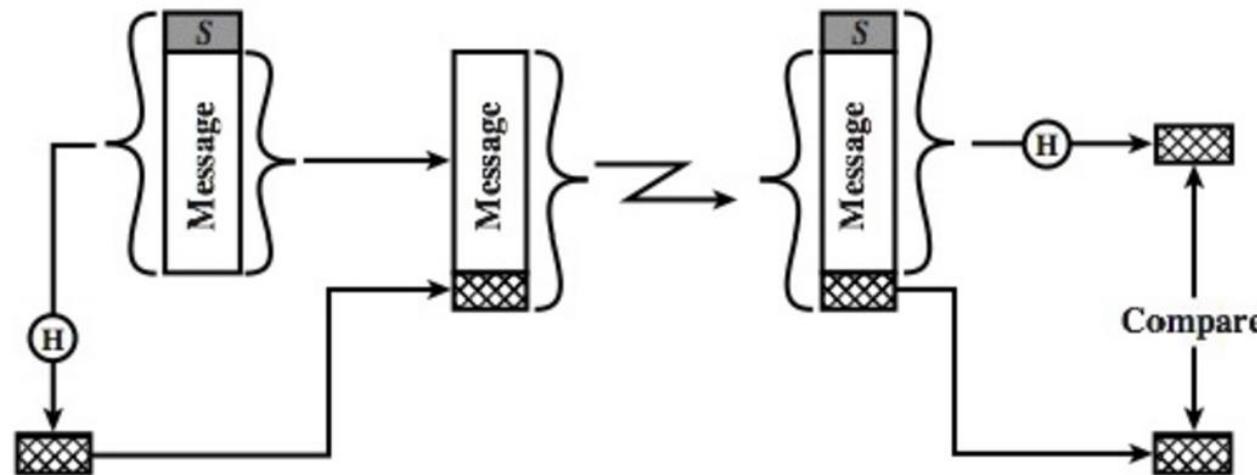


At the destination do the same and check if the MAC obtained is the same

Message Authentication Code (MAC)

3: keyed hash

- Hash the message along with a shared key
 - Put secret in the beginning of message
 - Ad-hoc algorithm showing the basic idea:



Not recommended! Extension attacks are possible

Extension attacks to keyed hash

- Append additional data to the original message without knowing the secret key
 - Attacker intercepts a message M and its valid MAC
 - Without knowing the key, append extra data D to M , creating $M' = M \parallel D$
 - Calculate a new MAC for M' using the intercepted MAC and **the structure of the hash function that continues from the previous value**
 - The recipient, unaware of the alteration, verifies M' with the provided MAC, which appears valid
 - This successfully deceives the recipient into accepting the tampered message $\backslash(M' \backslash)$ as authentic

HMAC (Hash-based MAC)

- HMAC algorithm
 - FIPS Standard / RFC 2104: keyed hash MAC
 - Introduces the following technique to prevent extension attacks
 - $\text{HMAC}(m,k) = \text{hash}(k \oplus \text{opad} \parallel \text{hash}(k \oplus \text{ipad} \parallel m))$
 - ipad – inner padding – 0x36363636
 - opad – outer padding – 0x5c5c5c5c
 - HMAC is used in practice: SSL/TLS, WTLS, IPsec

HMAC processing steps

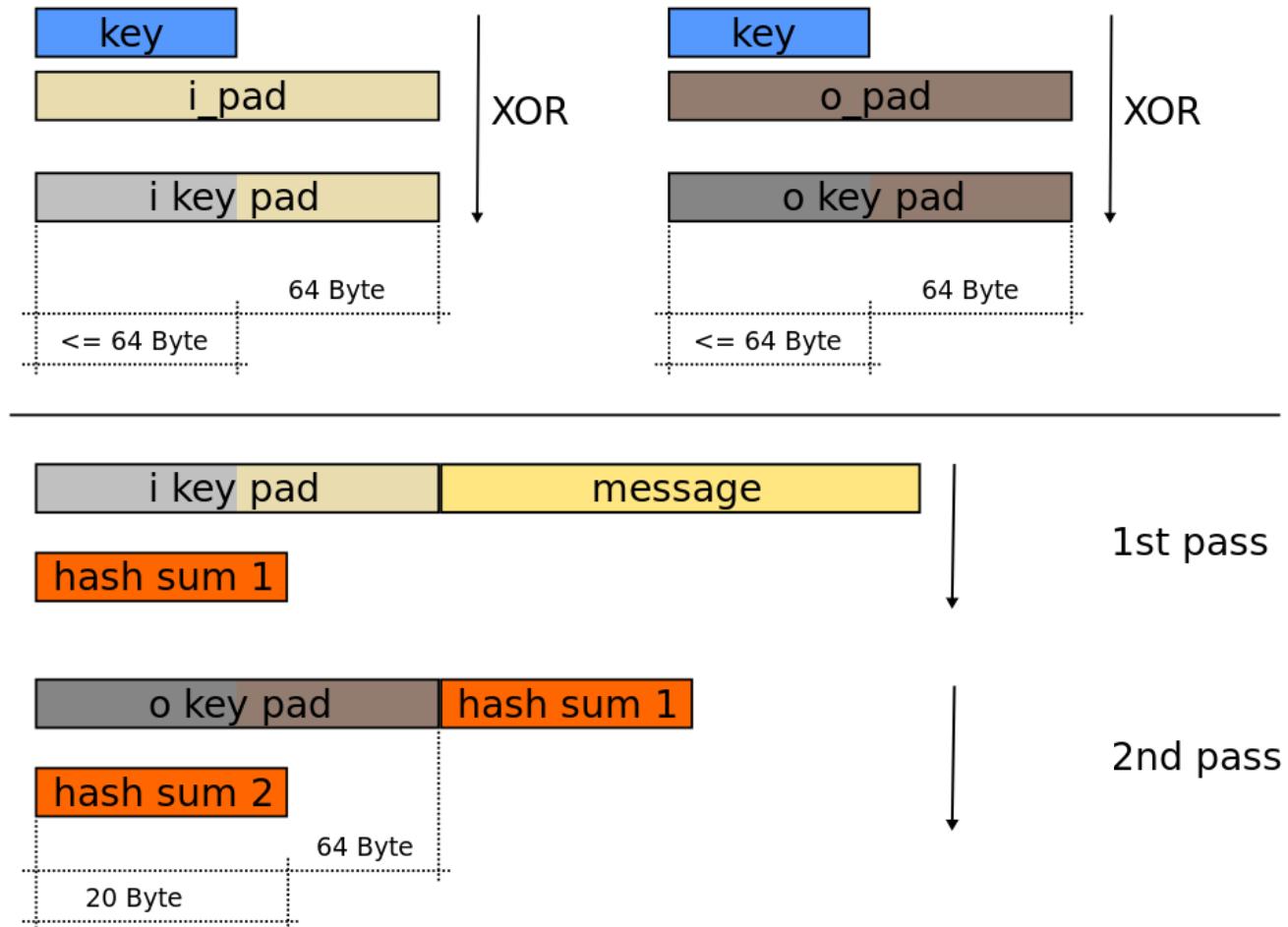


Table 9: Approval Status of MAC Algorithms

NIST Special Publication 800-131A
Revision 2

**Transitioning the Use of
Cryptographic Algorithms and
Key Lengths**

Elaine Barker
Allen Roginsky

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-131Ar2>

C O M P U T E R S E C U R I T Y



CMAC, GMAC, KMAC
são standards do NIST

MAC Algorithm	Implementation Details	Status
HMAC Generation	Key lengths < 112 bits	Disallowed
	Key lengths \geq 112 bits	Acceptable
HMAC Verification	Key lengths < 112 bits	Legacy use
	Key lengths \geq 112 bits	Acceptable
CMAC Generation	Two-key TDEA	Disallowed
	Three-key TDEA	Deprecated through 2023 Disallowed after 2023
	AES	Acceptable
CMAC Verification	Two-key TDEA	Legacy use
	Three-key TDEA	Legacy use
	AES	Acceptable
GMAC Generation and Verification	AES	Acceptable
KMAC Generation and Verification	Key lengths < 112 bits	Disallowed
	Key lengths \geq 112 bits	Acceptable

Summary

- Introduction
- Symmetric Ciphers
- Hash Functions
- Message Integrity Codes

Asymmetric Cryptography

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Roadmap

- Asymmetric Ciphers
- Digital Signatures

Roadmap

- **Asymmetric Ciphers**
- Digital Signatures

Asymmetric ciphers: overview

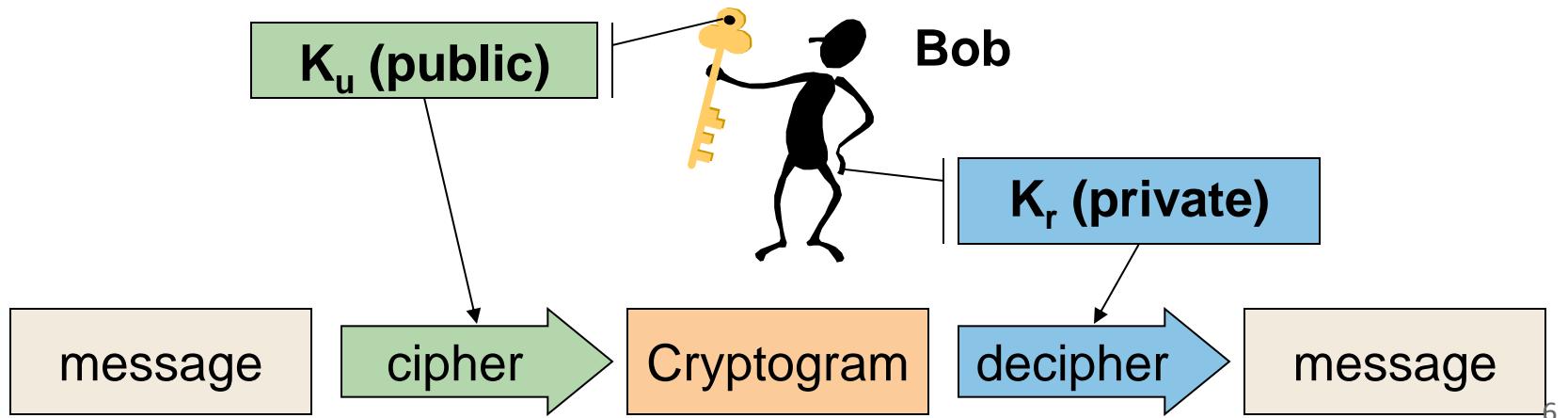
- Also known as **public-key cryptography**
- Appeared with the objective of solving a hard problem:
key distribution
- Use a **pair of keys**
 - One is **private**, personal, non-transmissible
 - One is **public**, can/should be widely known
- Allow for
 - **Confidentiality** with the exchange of secret keys
 - **Authentication** with digital signatures
 - **Integrity** with digital signatures

Asymmetric ciphers: assessment

- **Advantages**
 - N communicating parties $\Rightarrow N$ key pairs
 - Instead of $N(N-1)/2$ as symmetric ciphers
- **Disadvantages**
 - Performance: typically inefficient in comparison to symmetric ciphers (but signatures are not too slow today)
- **Problems**
 - Distribution of public keys
 - Although much simpler than distributing secret keys
 - Life-time of the key pairs (revocation)

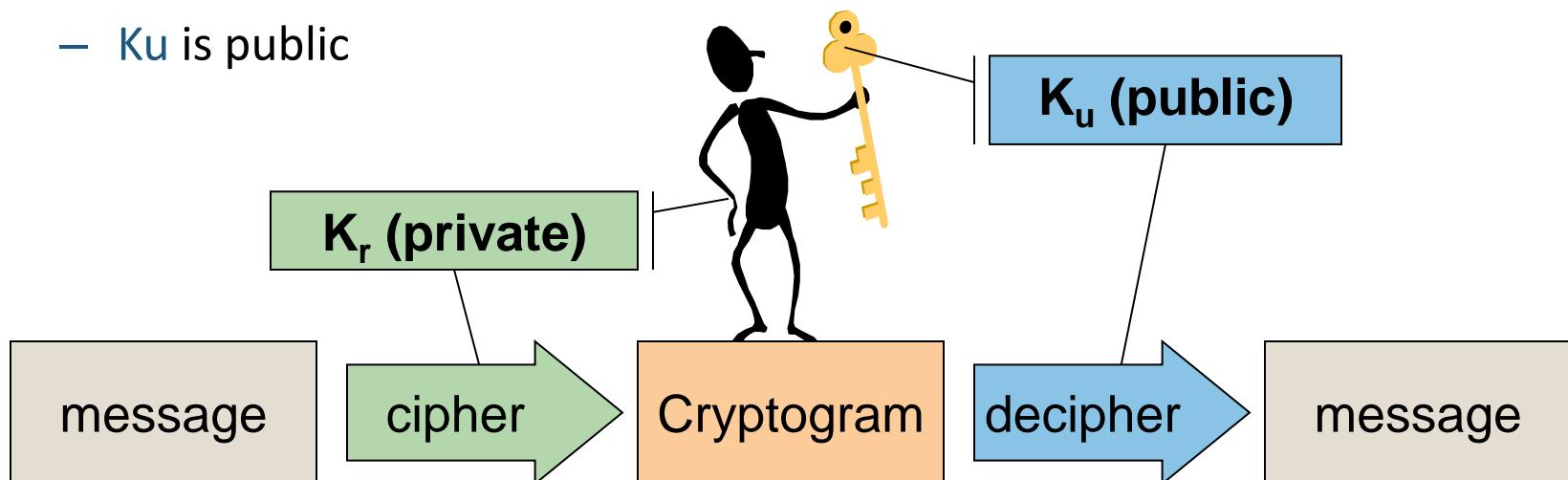
Asymmetric ciphers: Confidentiality

- Cipher / decipher with a pair of keys
 - $C = E(P, Ku)$ $P = D(C, Kr)$
 - To communicate with confidentiality with X we only need to know K_u
 - Very inefficient for large P, so not used like this for encrypting messages/files
- No authentication
 - Bob does not know who generated the cryptogram



Asymmetric ciphers: Authentication

- The cryptogram cannot be modified
 - $C = E(P, Kr)$ $P = D(C, Ku)$
 - Only X knows the key Kr used to generate the cryptogram
 - Better idea to encrypt $H(P)$: $C = E(H(P), Kr)$
- No confidentiality
 - Whoever knows the key Ku is able to decipher the cryptogram
 - Ku is public



Trapdoor functions



- Functions with the properties:
 - Computing in one direction is easy $A \rightarrow B$
 - Computing the inverse is very hard (unless you know a secret) $A \leftarrow B$
- Trapdoor functions are the core concept that enables **asymmetric cryptography**
 - If you find mathematical functions with these properties, you can build a cryptographic scheme around it

Asymmetric Ciphers

- Used mathematical approximations
 - Factorization of large numbers
 - Discrete logarithm problem
 - Knapsack problem
- Most common encryption algorithms
 - RSA (Rivest-Shamir-Adleman) – first and most used asymmetric encryption algorithm
 - ECC (Elliptic Curve Cryptography) – much used today, especially for signatures
- Other public-key techniques (not encryption)
 - Diffie-Hellman (DH), also known as Diffie-Hellman-Merkle – was foundational in the development of asymmetric crypto

Modular arithmetic

- Idea: do arithmetic normally (+ - × ÷) but result is the remainder of dividing by a modulus
 - mod, i.e., the result is the remainder of the integer division of the result by the modulus
 - Performs arithmetic operations not on a line but on a circle
 - mod can be applied “as often as you want”
 - Examples:
 - $1537 \times 4248 \pmod{10} = 6529176 \pmod{10} = 6$
 - $1537 \pmod{10} \times 4248 \pmod{10} = 7 \times 8 \pmod{10} = 56 \pmod{10} = 6$

RSA

- Rivest-Shamir-Adleman
- First **asymmetric encryption** algorithm
 - But not the first asymmetric cryptography algorithm
 - that was Diffie-Hellman-Merkle
- A major step forward for cryptography
 - Simplified key distribution

Programming S.L. Graham, R.L. Rivest*
Techniques Editors

A Method for Obtaining Digital Signatures and Public- Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science
and Department of Mathematics

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

(1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
(2) A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems. A message is encrypted by representing it as a number M , raising M to a publicly specified power e , and then taking the remainder when the result is divided by the publicly specified product, n , of two large secret prime numbers p and q . Decryption is similar; only a different, secret, power d is used, where $e * d = 1 \pmod{(p-1)(q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n .

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

CR Categories: 2.12, 3.15, 3.50, 3.81, 5.25

General permission to make fair use in teaching or research of all or part of this material is granted to individual readers and to nonprofit libraries acting for them provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery. To otherwise reprint a figure, table, other substantial excerpt, or the entire work requires specific permission as does republication, or systematic or multiple reproduction.

This research was supported by National Science Foundation grant MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063.

* Note. This paper was submitted prior to the time that Rivest became editor of the department, and editorial consideration was completed under the former editor, G. K. Manacher.

Authors' Address: MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

© 1978 ACM 0001-0782/78/0200-0120 \$00.75

I. Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II. Public-Key Cryptosystems

In a "public-key cryptosystem" each user places in a public file an encryption procedure E . That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D . These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M . Formally,

$$D(E(M)) = M. \quad (1)$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D . This means that in practice only he can decrypt messages encrypted with E , or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

$$E(D(M)) = M. \quad (2)$$

An encryption (or decryption) procedure typically consists of a *general method* and an *encryption key*. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the *ciphertext* C . Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

When the user reveals E he reveals a very *inefficient* method of computing $D(C)$: testing all possible messages M until one such that $E(M) = C$ is found. If property (c) is satisfied the number of such messages to test will be so large that this approach is impractical.

A function E satisfying (a)-(c) is a "trap-door one-way function;" if it also satisfies (d) it is a "trap-door one-way permutation." Diffie and Hellman [1] introduced the concept of trap-door one-way functions but

RSA overview

- **Base values and modulus**
 - p, q – **secret**; large prime numbers
 - $n = p \times q$ – **public**; large (1K, 2K, 3K bits); used as **modulus**; key length
- **Public key** – exponent **e** and **n** ($e < n$)
- **Private key** – exponent **d** ($d < n$) and primes **p** and **q**
- **Key generation:** choose **p, q** large primes
 - **e** chosen such that it is coprime with $z = \phi(n) = (p-1)(q-1)$
 - *coprime* means that the only factor that divides both is 1
 - $\phi(n)$ is the Euler's totient function; counts factors up to n that are coprime with n
 - **d** chosen such that $e.d \equiv 1 \pmod{z} \iff e.d \bmod z = 1$
- **Encryption with public key:** $C = P^e \bmod n$
- **Decryption with private key:** $P = C^d \bmod n$
 $= (P^e \bmod n)^d \bmod n = P^{ed} \bmod n = P^{ed \bmod z} \bmod n = P^1 \bmod n = P$

Public key example

- Taken from a browser

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Public Key
= modulus
256 bytes: D0 18 CF 45 D4 8B CD D3 9C E4 40
EF 7E B4 DD 69 21 1B C9 CF 3C 8E 4C 75 B9 0F
31 19 84 3D 9E 3C 29 EF 50 0D 10 93 6F 05 80
80 9F 2A A0 BD 12 4B 02 E1 3D 9F 58 16 24 FE
30 9F 0B 74 77 55 93 1D 4B F7 4D E1 92 82 10 F6
51 AC 0C C3 B2 22 94 0F 34 6B 98 10 49 E7 0B
9D 83 39 DD 20 C6 1C 2D EF D1 18 61 65 E7 23
83 20 A8 23 12 FF D2 24 7F D4 2F E7 44 6A 5B
4D D7 50 66 B0 AF 9E 42 63 05 FB E0 1C C4 63
61 AF 9F 6A 33 FF 62 97 BD 48 D9 D3 7C 14 67
DC 75 DC 2E 69 E8 F8 6D 78 69 D0 B7 10 05 B8
F1 31 C2 3B 24 FD 1A 33 74 F8 23 E0 EC 6B 19 8A
16 C6 E3 CD A4 CD 0B DB B3 A4 59 60 38 88 3B
AD 1D B9 C6 8C A7 53 1B FC BC D9 A4 AB BC DD
3C 61 D7 93 15 98 EE 81 BD 8F E2 64 47 20 40
06 4E D7 AC 97 E8 B9 C0 59 12 A1 49 25 23 E4
ED 70 34 2C A5 B4 63 7C F9 A3 3D 83 D1 CD 6D
24 AC 07

exponent e

Exponent 65537

Key Size 2 048 bits

= size of the modulus

RSA example with small numbers

- Key generation
 - Pick primes p and q : 3 and 11
 - Calculate modulus $n = p \times q = 3 \times 11 = 33$
 - Calculate $z = \phi(n) = (p-1) \times (q-1) = 2 \times 10 = 20$
 - Choose e such that e is co-prime with z and $e < n$: $e = 7$
 - Public key: $e = 7, n = 33$
 - Calculate d
 - $e.d = 1 \pmod{z}$ and $d < n \Leftrightarrow$
 - $7.d \pmod{20} = 1$ and $d < 33 \Rightarrow$
 - $d = 3$
- Encrypt w/ public key $P = 14$
 - $C = P^e \pmod{n}$
 - $C = 14^7 \pmod{33}$
 - $C = 20$
- Decrypt w/ private key
 - $P = C^d \pmod{n}$
 - $P = 20^3 \pmod{33}$
 - $P = 14$

RSA calculator

- Follow the algorithm steps with an online calculator:
 - <https://www.cs.drexel.edu/~popyack/IntroCS/HW/RSAWorksheet.html>

RSA Calculator

JL Popack, October 1997

This guide is intended to help with understanding the workings of the RSA Public Key Encryption/Decryption scheme. No provisions are made when dealing with large numbers.

Step 1. Compute N as the product of two prime numbers p and q:

p

q

Enter values for p and q then click this button:

The values of p and q you provided yield a modulus N, and also a number r=(p-1)(q-1), which is very important. You will need to find the inverse of the public key modulus N. When you click the Set p, q button, it appears a list of some numbers which equal 1 mod r. You will use this list in Step 2.

N = p*q

r = (p-1)*(q-1)

6865 13729 20593 27457 34321 41185 48049 54913 61777
68641 75505 82369 89233 96097 102961 109825 116689
123553 130417 137281 144145 151009 157873 164737 171601

Why is RSA difficult to break?

- RSA is challenging to break due to the computational difficulty of **integer factorization**
- It is easy to multiply large prime numbers to obtain a modulus **n**
 - But it is extremely hard to perform the inverse operation,
i.e., to deduce the original prime factors from **n** alone
- The encryption process in RSA is straightforward, utilizing the public key
 - But the decryption (inverse RSA) requires knowledge of these prime factors, which constitute the private key
- The complexity of factorizing large numbers into primes ensures that knowing the public key alone is insufficient for decryption, as it does not reveal the prime factors
- This fundamental asymmetry between the ease of multiplication and the difficulty of factorization underpins RSA's security.
- Future crisis: **Shor's quantum algorithm** threatens RSA by efficiently factorizing integers
 - Current quantum technology can only factorize small numbers
 - ≥ 2048 -bitnumbers are safe (for now...)
 - **Post-quantum** crisis; need for **quantum resistance**

RSA signatures

- RSA is often used to produce **signatures**
 - Public key – e, n
 - Private key – d, p, q
- **Signing the message M:**
 - Obtain S such that: $S = (H(M))^d \text{ mod } n$
 - Only the private key owner can generate the signature
- **Verifying the signature:**
 - Check if: $H(M) == S^e \text{ mod } n$
 - Anyone with the public key can verify the signature

Elliptic Curve Cryptography (ECC)

- RSA and DH use integer and polynomial arithmetic with very large numbers/polynomials
 - Impose a significant load in storing and processing keys and messages
- **Elliptic curves** are an alternative
 - Offer same security with smaller bit sizes
 - 224-bit ECC ~ 2048-bit RSA
 - 500-bit ECC ~ 15k-bit RSA
 - Newer, not as well analyzed, but commonly used
 - Many curves exist, each with their tradeoffs
 - Not necessarily faster than RSA for the same equivalent security
 - Normally used for **key agreement or signing**, not ciphering

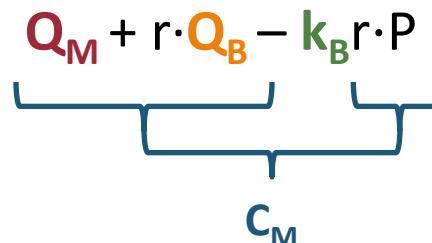
Elliptic Curve Cryptography (ECC)

- Analogy to RSA:
 - ECC addition is analogous to modulus multiplication
 - ECC repeated addition is analogous to modulus exponentiation
- The “hard” problem is the **elliptic curve logarithm problem**
 - $Q = k \cdot P$, where Q, P belong to an elliptic curve
e.g. $y^2 = x^3 - 3x + b$ modulo p
 - Easy to compute Q given k, P
 - Hard to find k given Q, P
- Public Key operations $Q(x,y) = kP(x,y)$:
 - Q = public key
 - k = private key
 - P = base point (a parameter of the curve)

ECC Encryption/Decryption

- There are several alternatives, but we consider the simplest: to encode the message \mathbf{M} as a point \mathbf{Q}_M on the elliptic curve
- Key generation
 - Select suitable curve and base point P
 - Bob chooses a **private key**: $k_B < n$
 - Bob computes the **public key**: $\mathbf{Q}_B = k_B \cdot P$
- Alice **encrypts** \mathbf{Q}_M using Bob's **public key** \mathbf{Q}_B :
$$\mathbf{C}_M = \{r \cdot P, \mathbf{Q}_M + r \cdot \mathbf{Q}_B\}$$
 – r is a random number; C_M is a point in the curve
- Bob **decrypts** \mathbf{C}_M using its **private key** k_B :

$$\mathbf{Q}_M + r \cdot \mathbf{Q}_B - k_B \cdot r \cdot P = \mathbf{Q}_M + r(k_B \cdot P) - k_B(r \cdot P) = \mathbf{Q}_M$$



Porque $\mathbf{Q}_B = k_B \cdot P$

ECC Diffie-Hellman

- ECC Diffie-Hellman:
 - Key exchange is possible (analogous to Diffie-Hellman)
 - users select a suitable curve $E_p(a,b)$ – public information
 - select base point $P=(x_1,y_1)$ – public information
 - with large order n ($nP=O$, and n smallest with this property).
 - A & B select private keys:
 - $k_A < n$ & $k_B < n$
 - compute public keys:
 - $Q_A = k_A P$ & $Q_B = k_B P$
 - compute shared key:
 - $K = k_A Q_B$, or $K = k_B Q_A$
 - same since:
 - $K = k_A k_B P$

Roadmap

- Asymmetric Ciphers
- **Digital Signatures**

Digital signatures – objectives

- Authenticate the content of a document
- Authenticate its signer
- Being able to assure authentication towards a third party
- Signer cannot repudiate the signature



Assinatura Digital

A assinatura digital qualificada permite ao titular de um Cartão de Cidadão, por vontade própria, assinar com a chave pessoal existente no seu Cartão de Cidadão.

Qualquer entidade pode verificar a assinatura digital recorrendo ao uso do certificado digital pessoal do cidadão e a meios de verificação da validade deste certificado.

CMD Assinatura

A assinatura eletrónica qualificada através da Chave Móvel Digital (CMD) permite ao cidadão, português ou estrangeiro, por vontade própria, assinar com a palavra-chave por si escolhida e respetivo código de segurança.

Digital signatures

- Used approximations: asymmetric cipher + hash function
- **Basic algorithm** (K_x^r - private key K_x^u - public key)
 - Signature: $S_x(\text{doc}) = E(K_x^r, \text{hash}(\text{doc}))$
 - Validation: check if $D(K_x^u, S_x(\text{doc})) = \text{hash}(\text{doc})$
- Why cannot an adversary provide a fake signature?
- Another option:
 - Signature: $S_x(\text{doc}) = \text{info} + E(K_x^r, \text{hash}(\text{doc+ info}))$
 - Validation: $D(K_x^u, S_x(\text{doc})) = \text{hash}(\text{doc + info})$
 - Example **info**: timestamp, used algorithm, ...
- Today often **ECDSA** is adopted (with elliptic curve encryption)

Surreptitious forwarding

- A sends B : $\{\{ \text{"I love you"} \}a\}B$
- B deciphers -> $\{ \text{"I love you"} \}a$
- B sends C : $\{\{ \text{"I love you"} \}a\}C$

a – private key
B, C – public keys

Message stealing

- A tries to send | B : {{“my idea”}} B}a
- C intercepts message
- C sends B : {{“my idea”}} B}c

a,c – private keys
B – public key

Solutions

- Sign the name of the receiver
 - A ---> B : $\{\{Bob, msg\}a\}B$
- Cipher the name of the sender
 - A ---> B : $\{\{Alice, msg\}B, \#msg\}a$
- Incorporate both names in the message
 - A ---> B : $\{\{\text{'`A ---> B'}, msg\}B, \#msg\}a$
 - A ---> B : $\{\{\text{'`A ---> B'}, msg\}a\}B$
- Sign + Cipher + Sign*
- Cipher + Sign + Cipher*
- * Not practical
- [Davis2001] Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML

**Transitioning the Use of
Cryptographic Algorithms and
Key Lengths**

Elaine Barker
Allen Roginsky

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-131Ar2>

C O M P U T E R S E C U R I T Y



Table 2: Approval Status of Algorithms Used for Digital Signature Generation and Verification

Digital Signature Process	Domain Parameters	Status
Digital Signature Generation	< 112 bits of security strength: DSA: $(L, N) \neq (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA: $\text{len}(n) < 224$ RSA: $\text{len}(n) < 2048$	Disallowed
	<u>≥ 112</u> bits of security strength: DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA or EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	Acceptable
Digital Signature Verification	< 112 bits of security strength: DSA ³² : $((512 \leq L < 2048) \text{ or } (160 \leq N < 224))$ ECDSA: $160 \leq \text{len}(n) < 224$ RSA: $1024 \leq \text{len}(n) < 2048$	Legacy use
	<u>≥ 112</u> bits of security strength: DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA and EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	Acceptable

Summary

- Asymmetric Ciphers
- Digital Signatures

Cryptographic Services Technology

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Cryptographic services (revisited)

- We can now implement cryptographic services:
 - Confidentiality
 - Integrity
 - Authenticity
- To protect against:
 - Unauthorized insertion of information
 - Loss of authenticity
 - Unauthorized modification of information in transit
 - Loss of integrity
 - Unauthorized replay of information
 - Loss of authenticity
 - Unauthorized access to information
 - Loss of confidentiality

Cipher suite

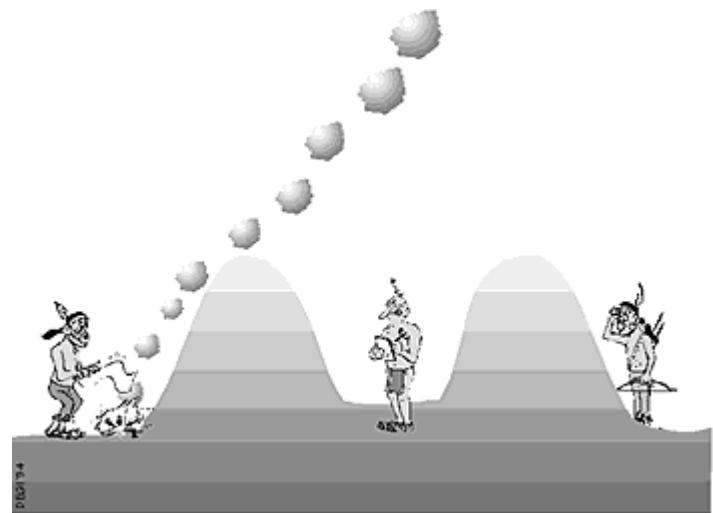
- A collection of cryptographic algorithms and protocols for digital security
- Components include encryption algorithms, message authentication codes and hash functions, like:
 - AES to encrypt data
 - HMAC with SHA-256 to allow data tampering detection
- Used in protocols like SSL/TLS for secure Internet communications:
 - Selection - server and client negotiate to choose the most suitable suite during connection setup
 - Key Exchange - algorithms like RSA or Diffie-Hellman are used to establish a secure connection (*more on this later in the course*)

Example: cipher suites in Java

- JCA (Java Cryptography Architecture):
- Framework for cryptographic functions in Java
 - Supports encryption, key generation, and digital signatures
 - Ensures uniform cryptography across Java applications
- In the code, applications ask for specific implementations from providers:
 - `Cipher.getInstance("AES/CBC/PKCS5Padding");`
 - `Cipher.getInstance("RSA/ECB/PKCS1Padding");`
 - `MessageDigest.getInstance("SHA-256");`
 - `Signature.getInstance("SHA256withRSA");`
 - `Mac.getInstance("HmacSHA256");`

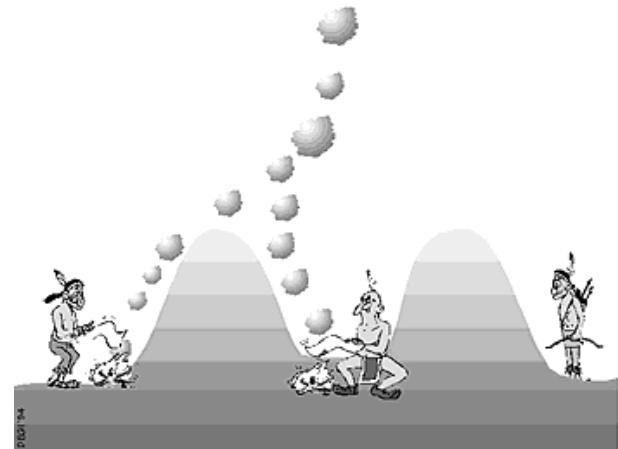
1 – Confidentiality

- Symmetric cipher
 - `Cipher.getInstance("AES/CBC/PKCS5Padding");`
- Asymmetric cipher
 - `Cipher.getInstance("RSA/ECB/PKCS1Padding");`



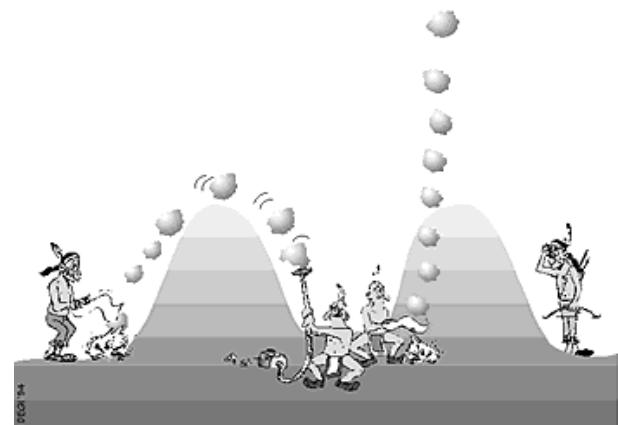
2 – Integrity

- MIC
 - `Mac.getInstance("HmacSHA256");`
- DS
 - `Signature.getInstance("SHA256withRSA");`



3 - Authenticity

- The basis of authenticity is *integrity* using MIC or DS
- Add **metadata** with *origin* information :
 - e.g. sender identification
- and *freshness* elements:
 - e.g. timestamp and random number
 - the random number must be unique
inside the timestamp's acceptance interval
- The metadata and data
must be protected **together!**
 - Same MIC or DS



Summary

- We need **robust** and **well-tested** cryptographic implementations to protect our applications
- Usually there are multiple cipher suites available
 - We saw the Java example, but similar capabilities are available in other mainstream programming languages
 - Algorithms are standardized, so there is compatibility between heterogeneous systems
- Using them correctly, we can provide services of:
 - Confidentiality
 - Integrity
 - Authenticity

Network Vulnerabilities in OSI Layers 1 to 3

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Carlos Ribeiro,
André Zúquete, Miguel P. Correia

Roadmap

- Network models
 - OSI and Internet
 - Address resolution
- Network vulnerabilities
 - Physical layer
 - Data link layer
 - Network layer

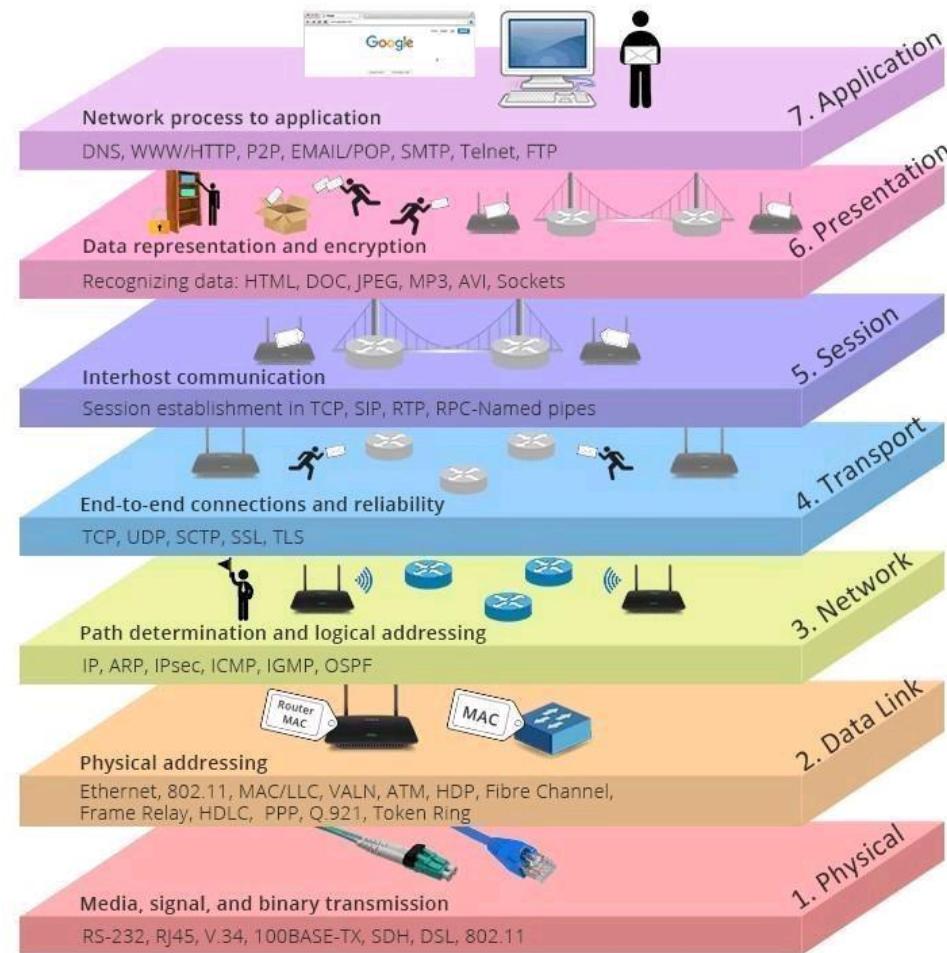
Roadmap

- **Network models**
 - OSI and Internet
 - Address resolution
- Network vulnerabilities
 - Physical layer
 - Data link layer
 - Network layer

OSI model

1. **Physical** (Ethernet, FDDI, B8ZS, V.35, V.24, RJ45)
2. **Data Link** (PPP, FDDI, ATM, IEEE 802.5/ 802.2, IEEE 802.3/802.2, HDLC, Frame Relay)
3. **Network** (IP, IPX, AppleTalk DDP)
4. **Transport** (TCP, UDP, SPX)
5. **Session** (NFS, NetBios names, RPC, SQL)
6. **Presentation** (ASCII, EBCDIC, JPEG, MPEG, GIF, PICT, TIFF)
7. **Application** (HTTP, FTP, SNMP, NFS, Telnet)

OSI model layer 7



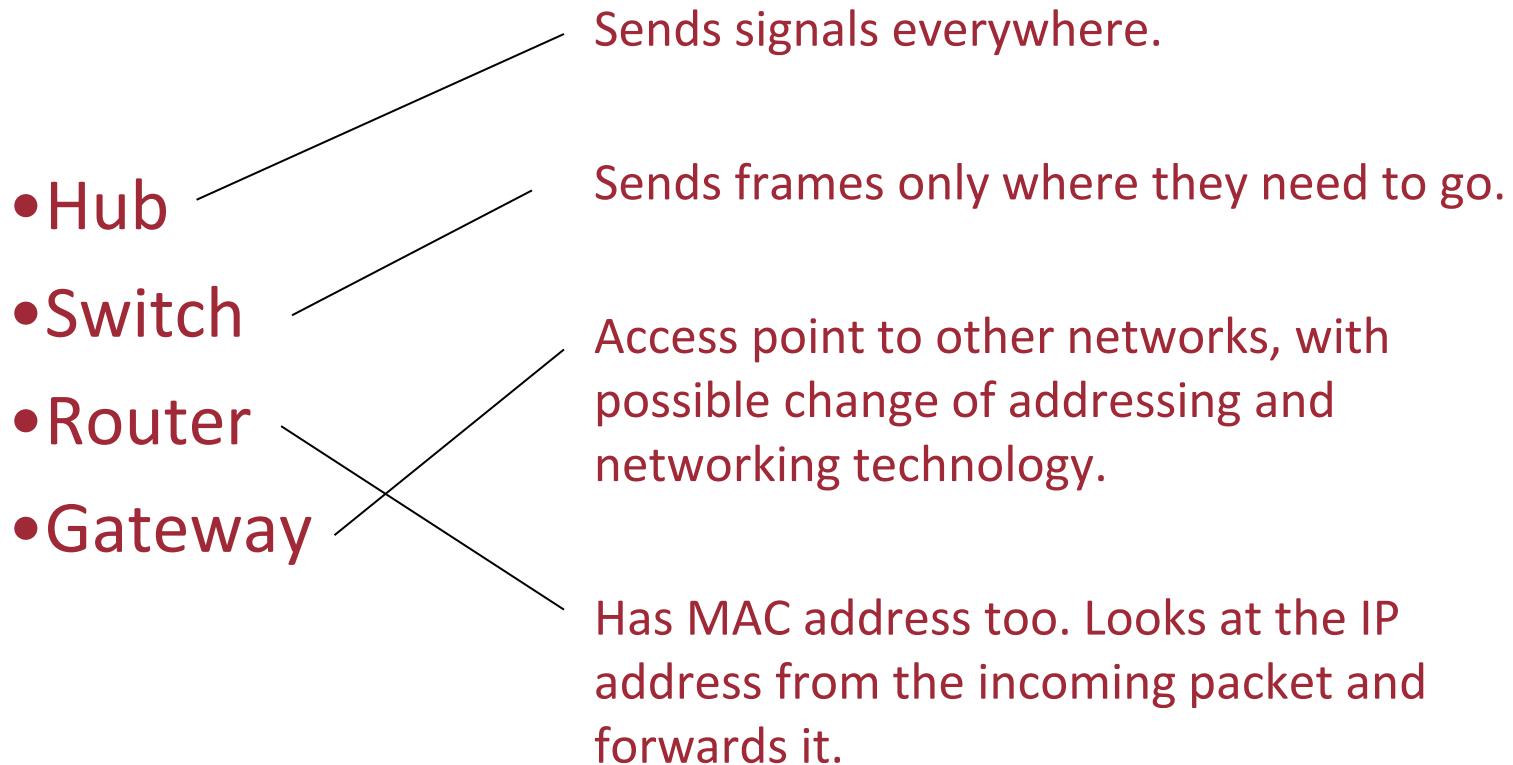
Open System Interconnection

SIRS

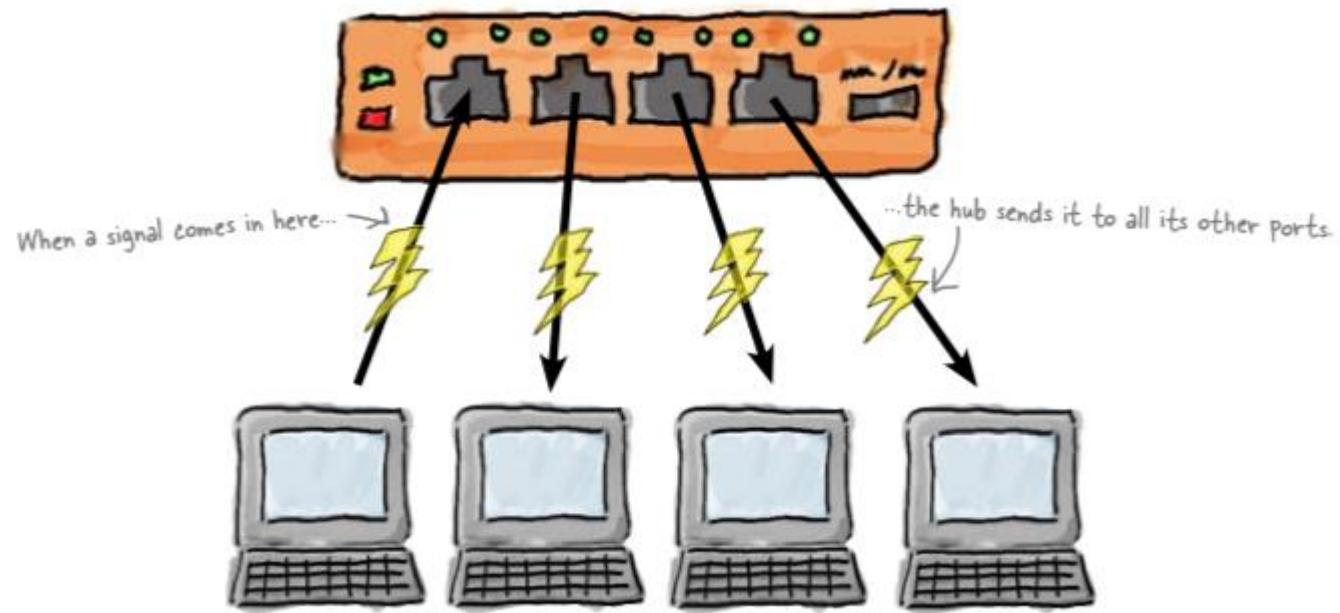
12

Hubs vs Switches vs Routers vs Gateways

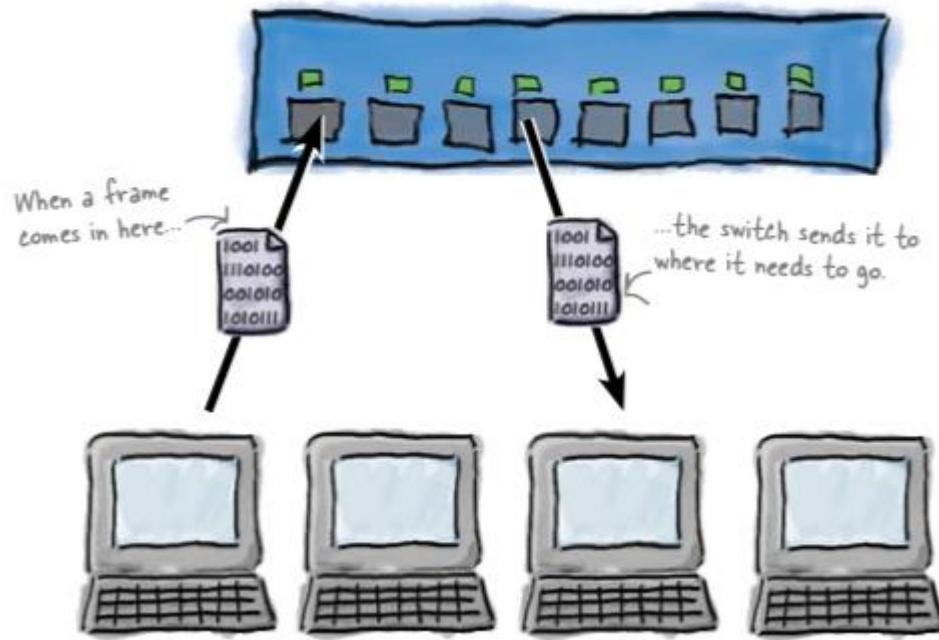
- The network is not just computers and servers
 - There are also network devices such as hubs, switches, routers, and gateways



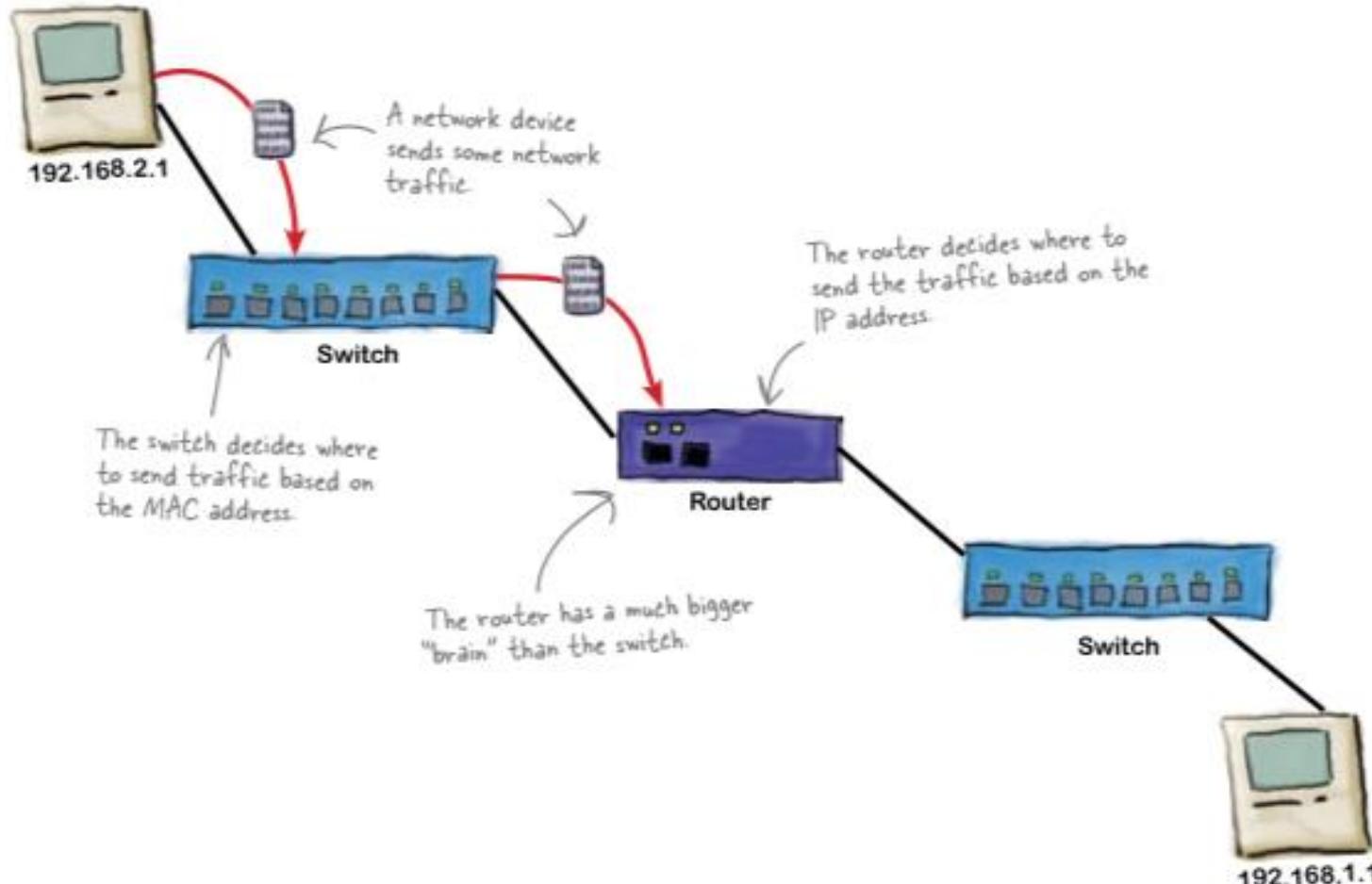
Hubs



Switches



Routers



Roadmap

- **Network models**
 - OSI and Internet
 - **Address resolution**
- Network vulnerabilities
 - Physical layer
 - Data link layer
 - Network layer

Network Addresses

- MAC address (layer 2)
 - MAC = Medium Access Control
 - Address of NIC (Network Interface Card)
 - Unique identifier with 48 bits
 - The first 24 identify the manufacturer
- IP address (layer 3)
 - IP = Internet Protocol ≈ Inter-connect Net-works Protocol
 - IPv4 address has 32 bits
 - Usually represented as 4 separate decimal numbers
 - 131.159.15.24
 - IPv6 address has 128 bits
 - Represented as 8 groups of 4 hex digits (16 bits)
 - 2001:4ca0:2001:0013:0250:56ff:feba:37ac



IP address

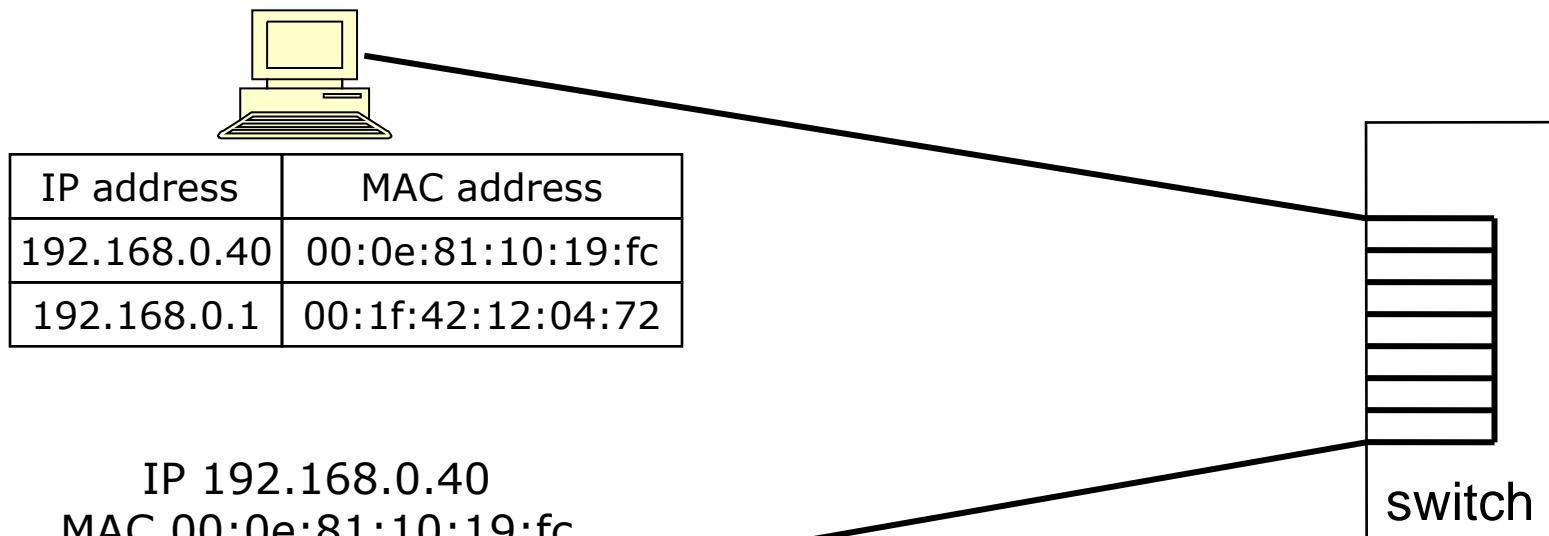
- IP addresses identify the network and the machine
- Example: 192.168.0.22 address:
 - In CIDR notation: 192.168.0.22 / 24
 - First 24 bits of IP address are significant for network routing
 - Network mask is 255.255.255.0
 - 192.168.0.* identifies the network
 - *.*.*.22 identifies the machine

Address Resolution: MAC to IP

- Address Resolution Protocol (ARP)
 - Layer 3 Protocol (Network)
 - Translates an IP address into a MAC address
- ARP query
 - Who has the IP 192.168.0.40? Answer to 192.168.0.20
- ARP reply
 - 192.168.0.40 is at 00:0e:81:10:19:FC
- ARP caches:
 - Stores previous answers
 - When the answers are too old, they are removed

ARP tables

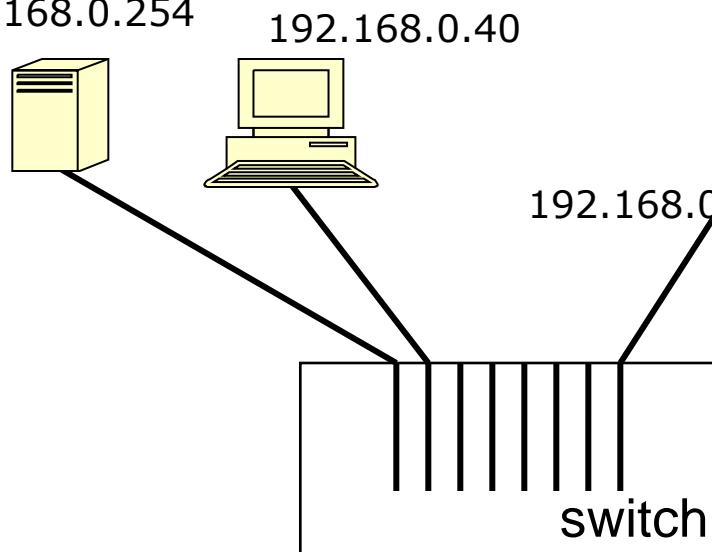
IP 192.168.0.20
MAC 00:0e:81:10:17:d1



IP address	MAC address
192.168.0.20	00:0e:81:10:17:d1
192.168.0.1	00:1f:42:12:04:72

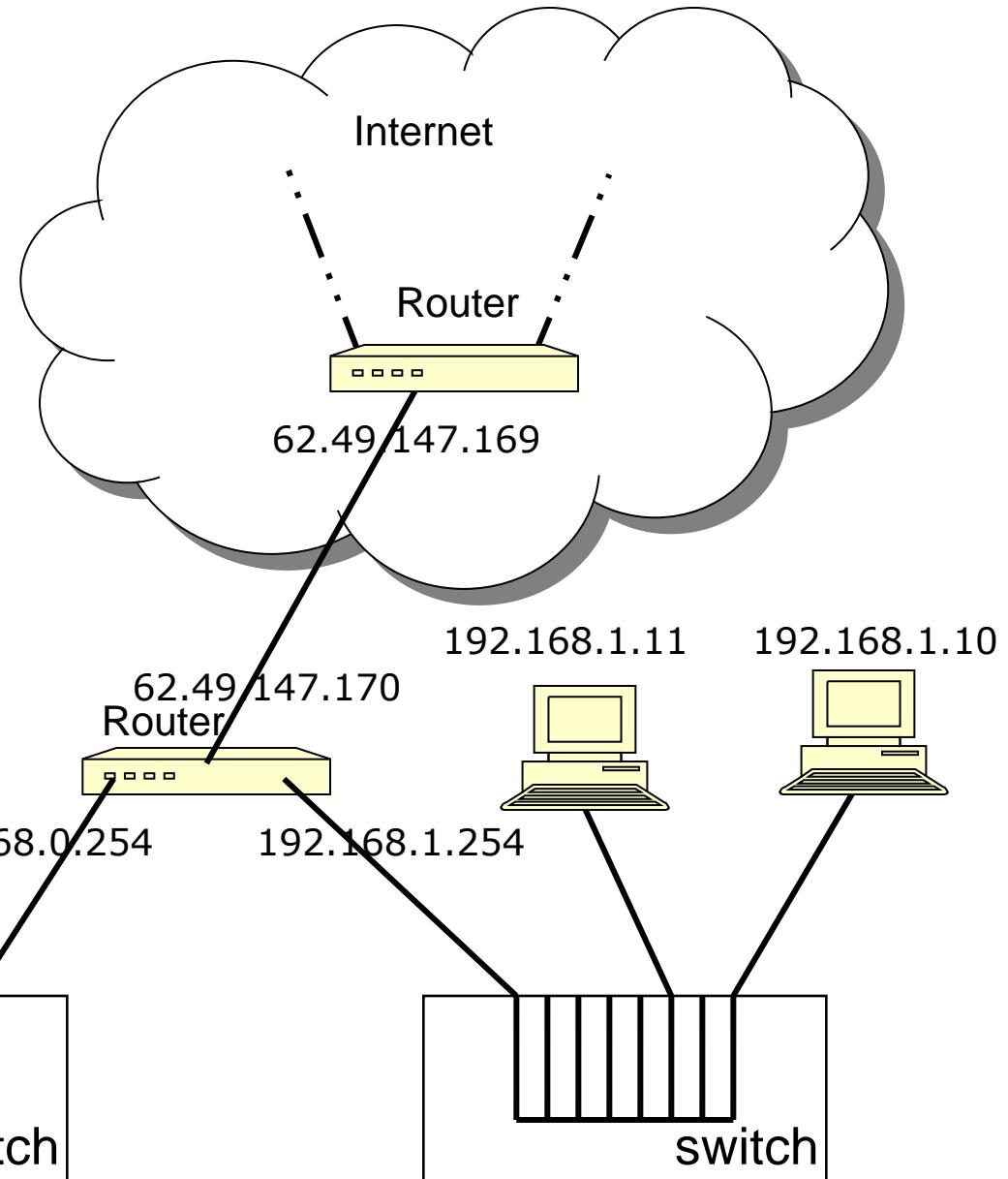
Routing

IP address
192.168.0.20
Network mask
255.255.255.0
Default router
192.168.0.254



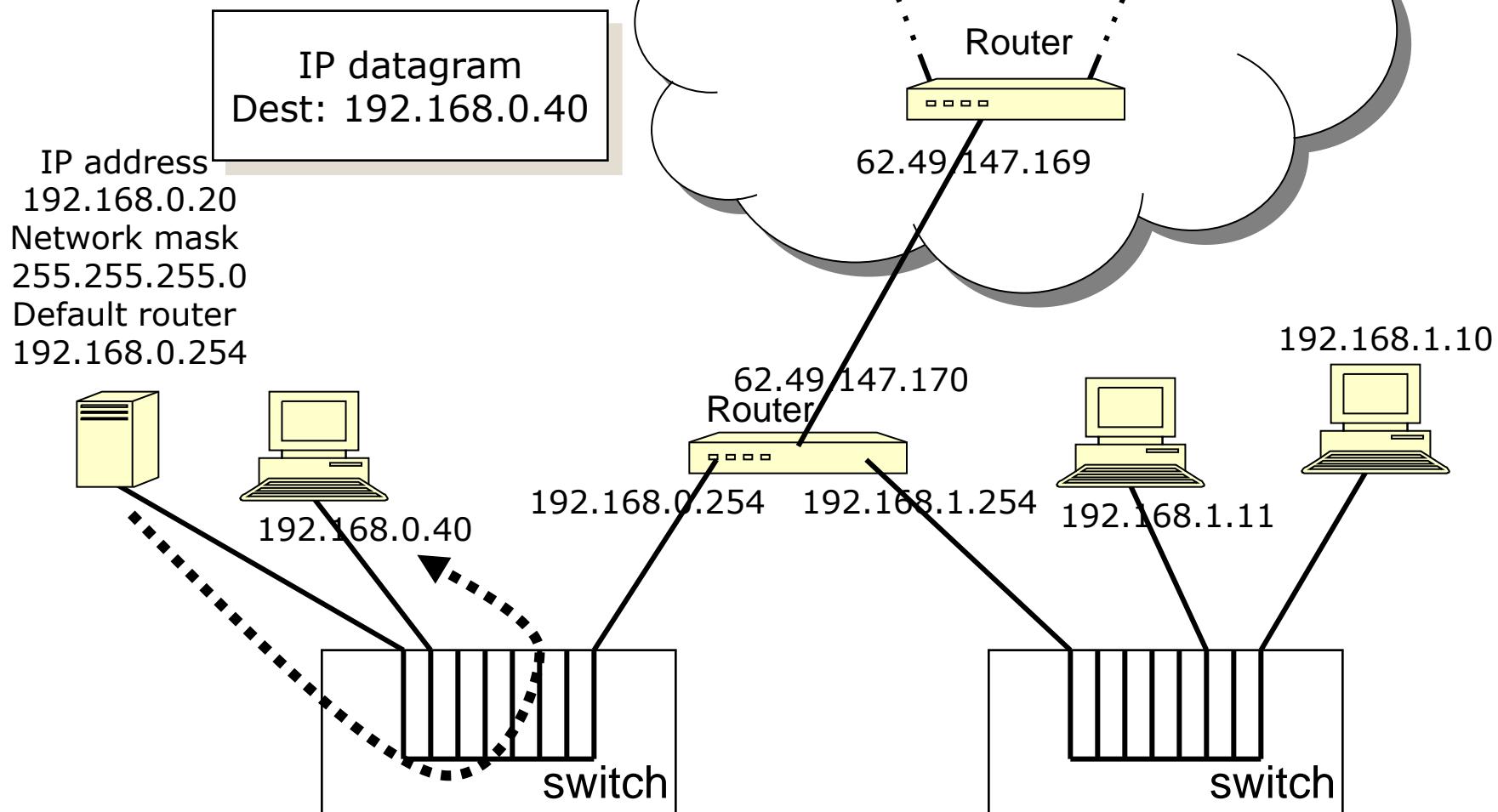
192.168.0.254

switch



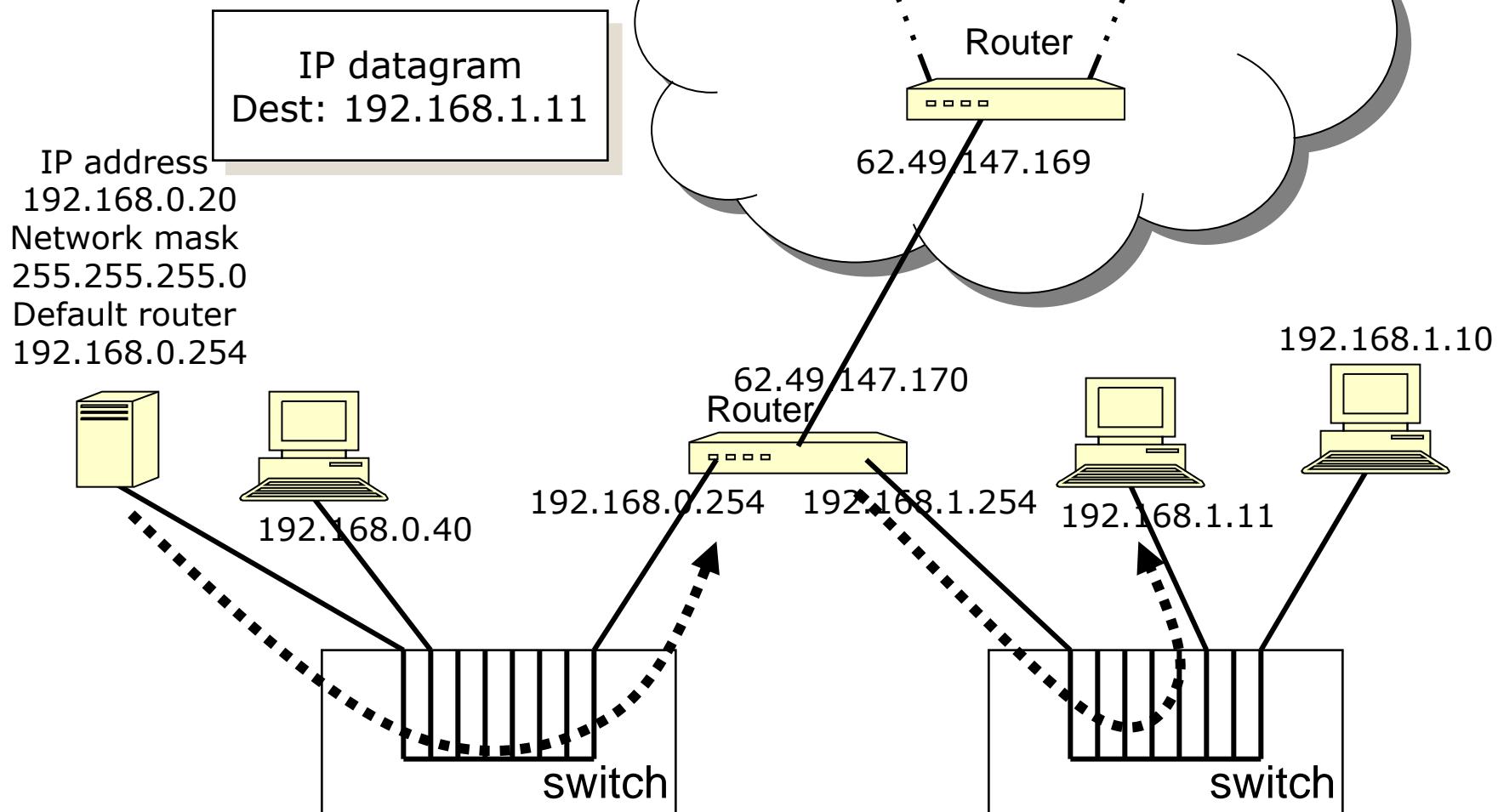
Routing

Direct delivery



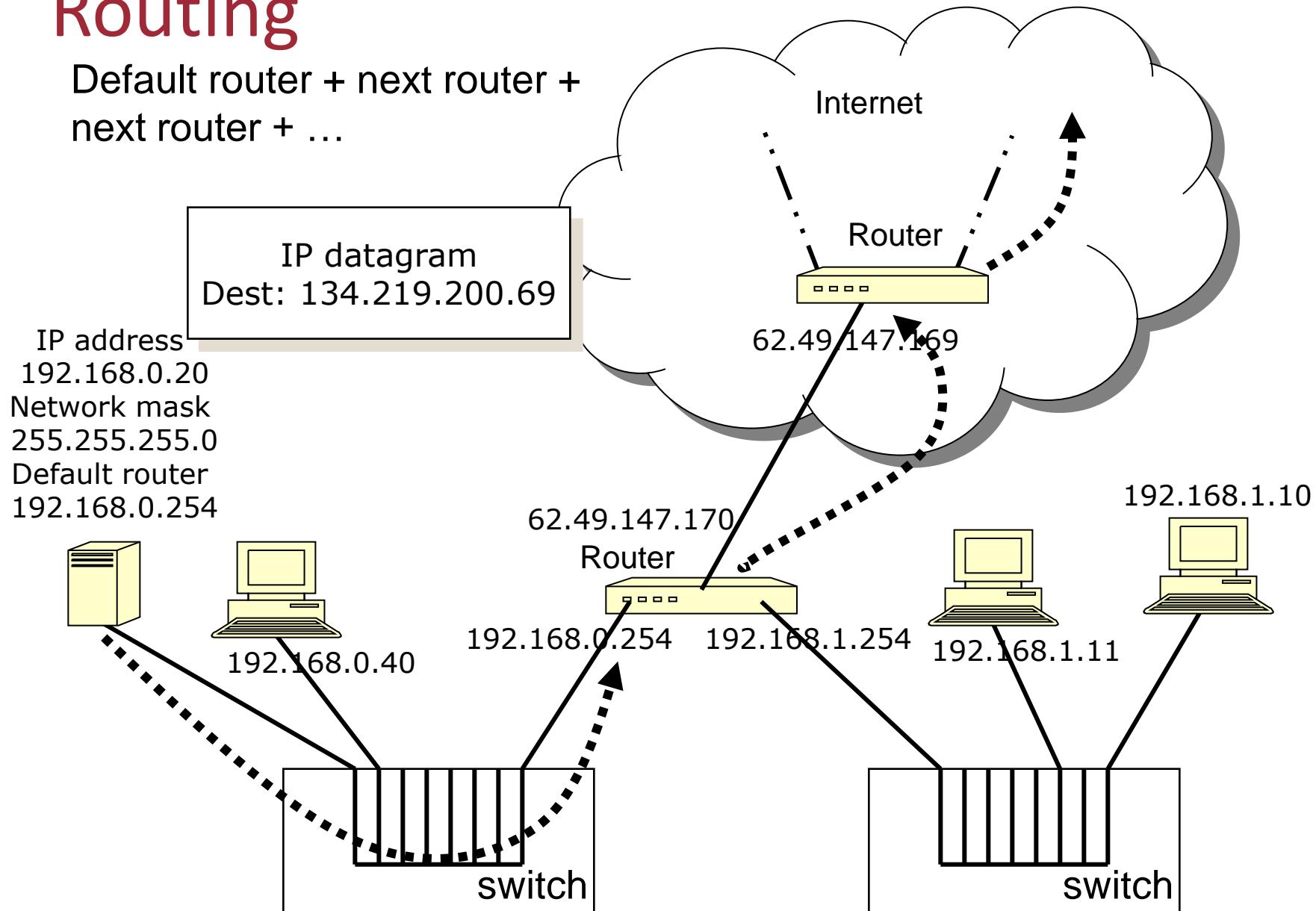
Routing

Default router +
direct delivery



Routing

Default router + next router +
next router + ...



Private Addresses

- Some network ranges were reserved for private addressing (IETF RFC 1918):
 - 10.0.0.0 to 10.255.255.255 (1 network, 2^{24} machines)
 - 172.16.0.0 to 172.31.255.255 (16 networks, 2^{16} machines each)
 - 192.168.0.0 to 192.168.255.255 (256 network, 2^8 machines each)
- Packets with these addresses (origin or destination) should never be sent outside the network itself
 - An attempt to solve the lack of IP addresses
 - Adds security because machines cannot be addressed from outside the network
- In the previous example, the router has:
 - one public IP address: 62.49.147.170 and
 - two private addresses: 192.168.0.254 and 192.168.1.254

Roadmap

- Network models
 - OSI and Internet
 - Address resolution
- **Network vulnerabilities**
 - Physical layer
 - Data link layer
 - Network layer

(Layer 1) Physical Layer: Hubs

- Topics:
 - Behavior
 - Problems
 - Sniffers and anti-sniffers

Hub behavior

- Information broadcast on a shared medium
 - Threats: Information Leakage (sniffers)
- Easy to install more devices
 - But anyone can connect
 - Even if the Hub is physically secure

Sniffers

- Usually, network adapters operate in a non-promiscuous mode
 - Network adaptors only listen to what is sent to their MAC
- Sniffers work in a promiscuous mode
 - Read all frames, with any MAC
- Some sniffer tools:
 - Tcpdump
 - Wireshark (Ethereal)
 - Snort



Identifying sniffers

- AntiSniff tool
 - Latency Method
 - Send high volume of packets to target
 - Compare time needed to answer to 1 packet vs N packets
 - DNS Method
 - Detect large volume of reverse lookup DNS queries from Tcpdump, Wireshark running at sniffer machine
 - OS-specific Method
 - Sends packets to target system which certain operating systems respond to
 - Example: Windows in promiscuous mode always responds to MAC = ff:00:00:00:00:00

Identifying sniffers using ARP

- ARP method
 - Machines cache ARPs
 - Send a non-broadcast ARP with our correct MAC address
 - Then send a broadcast ping with the right IP but wrong MAC address
 - Only a machine which has our correct MAC address from the sniffed ARP will respond
 - i.e., the sniffer machine!

Preventing Sniffing

- Solutions:
 - Prevent the use of network adapters in promiscuous mode
 - Use of switches instead of hubs
 - But does not fully solve (as we will see later)
- Prevent effectiveness of sniffing:
 - One-time passwords
 - e.g. SecurID, S/Key
 - Use of encryption

Roadmap

- Network models
 - OSI and Internet
 - Address resolution
- **Network vulnerabilities**
 - Physical layer
 - **Data link layer**
 - Network layer

(Layer 2) Data Link

- Topics:
 - Switches
 - Behavior
 - MAC flooding
 - ARP spoofing/poisoning

Switch behavior

- Switches *typically* send frames only to the destination MAC address
 - They have a table with the MAC reachable from each of their ports

Port	MAC
1	00:0e:81:10:19:fc
2	00:1f:42:12:04:72
...	...

- When a frame reaches the switch:
 - Searches for the port where the device with that MAC is at
 - Sends the frame to that port
- Switches reduce the *sniffing* problem
 - The network adapter *typically* only sees what is meant for it

ARP Vulnerabilities

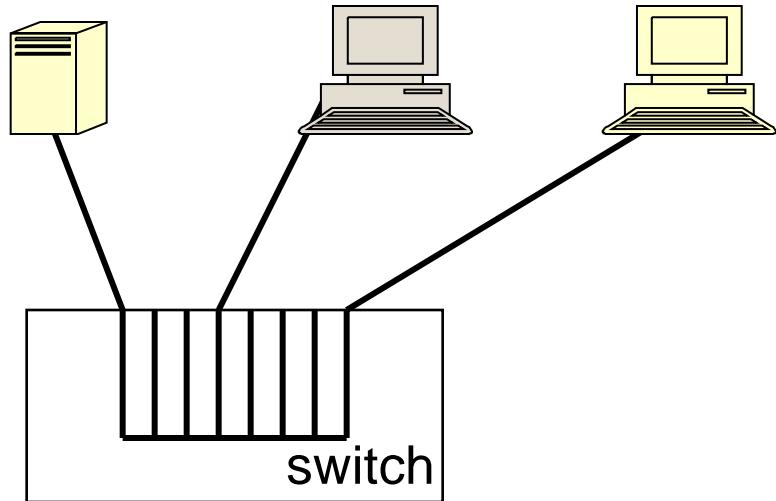
- MAC flooding
 - Overwhelm the switch with entries
- ARP spoofing/poisoning:
 - An attacker sends a non-requested ARP message with a false IP-MAC address correspondence
 - ARP messages are in no way signed, so it is easy to falsify a message from any given MAC

MAC Flooding

- Attacker sends several unsolicited ARP messages
 - Each ARP message is sent with a different MAC
- When the table is filled up:
 - Some switches stop accepting new connections (DoS)
 - Most switches revert to a Hub mode:
 - Allowing standard sniffing attacks to work again!

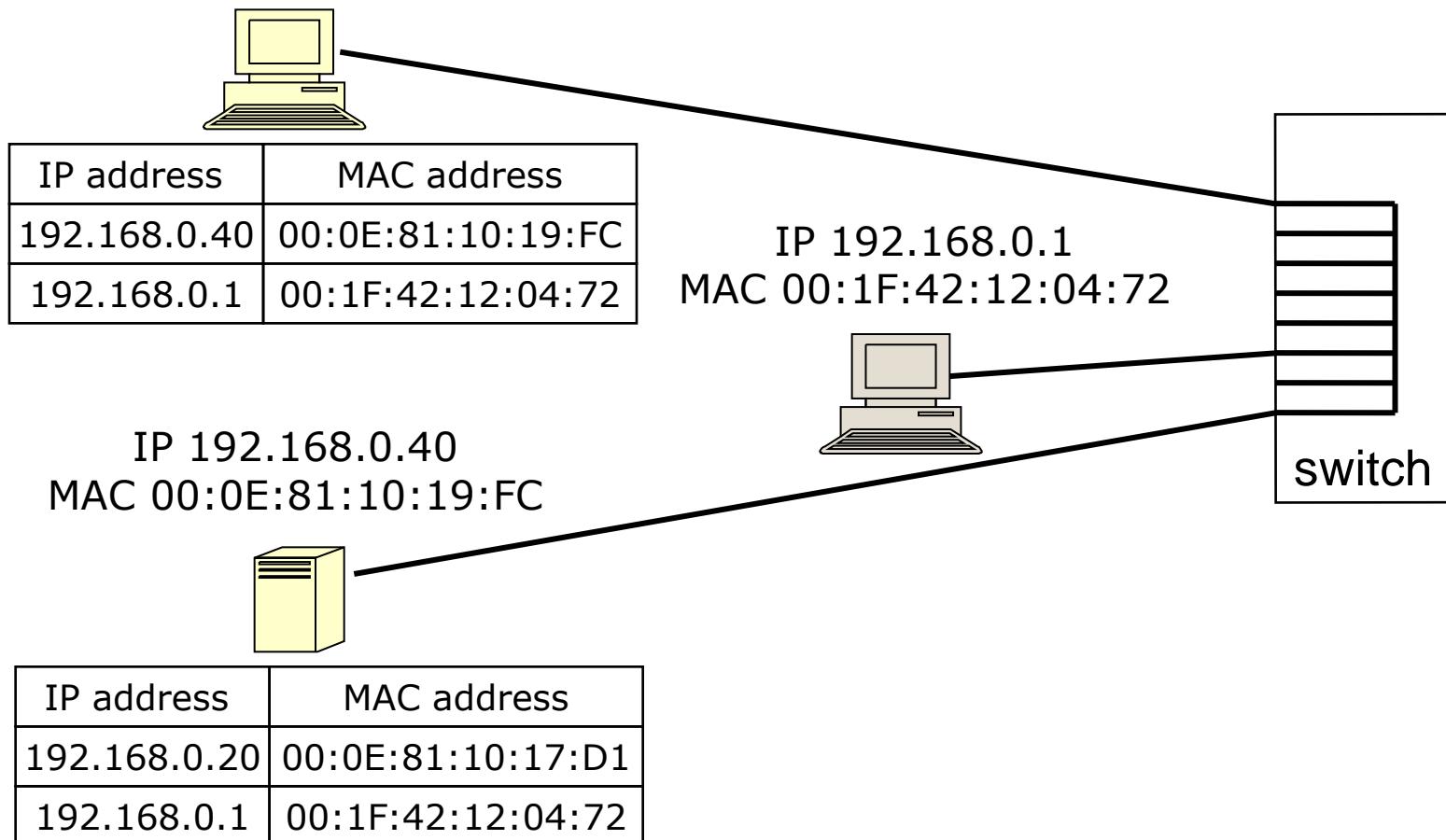
	Device	MAC address
1	1	00:0e:81:10:19:fc
2	4	00:0e:81:32:96:af
3	4	00:0e:81:32:96:b0
4	4	00:0e:81:32:96:b1

9999	4	00:0e:81:32:97:a4



ARP Tables OK

IP 192.168.0.20
MAC 00:0E:81:10:17:d1

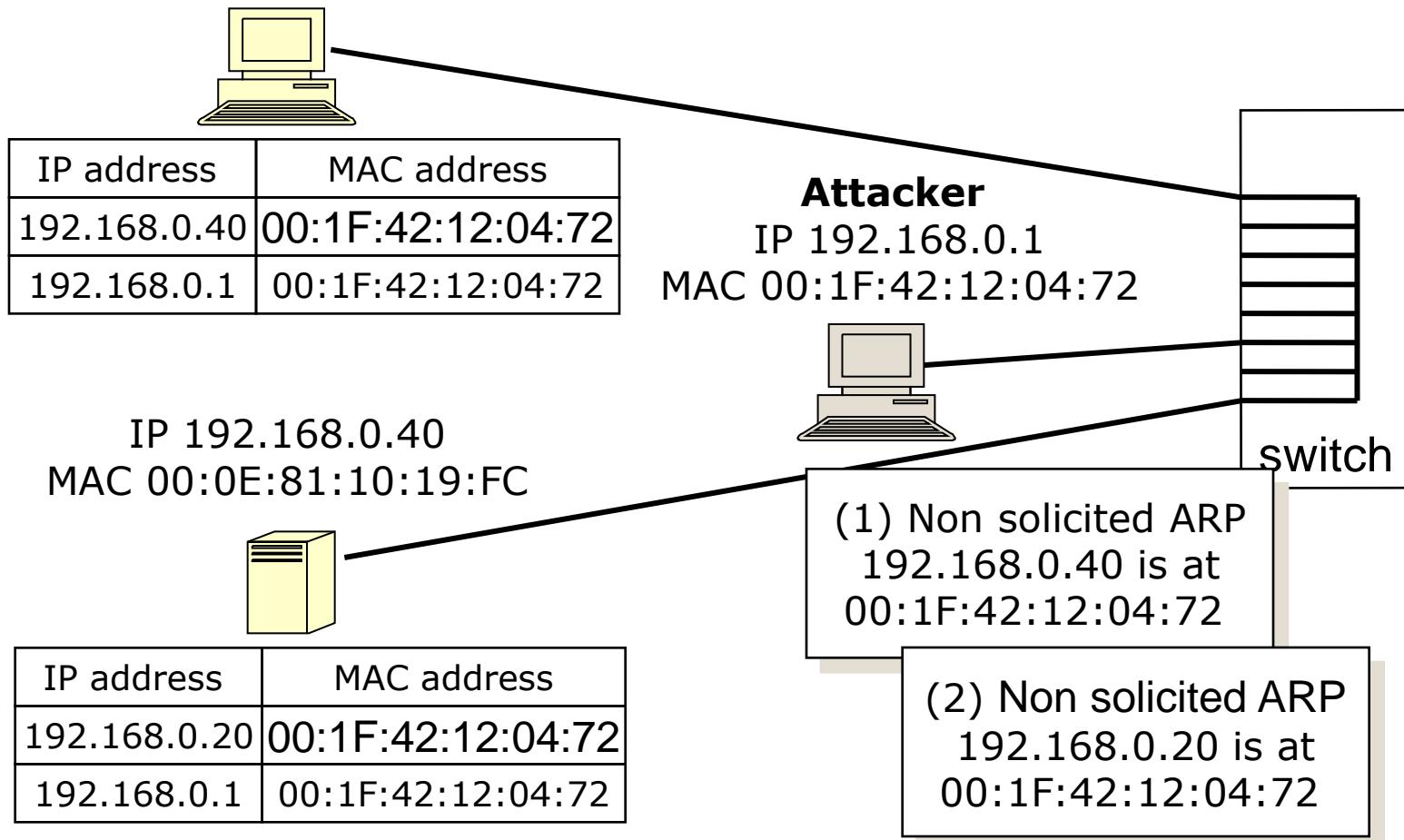


ARP Spoofing/Poisoning Attack Steps

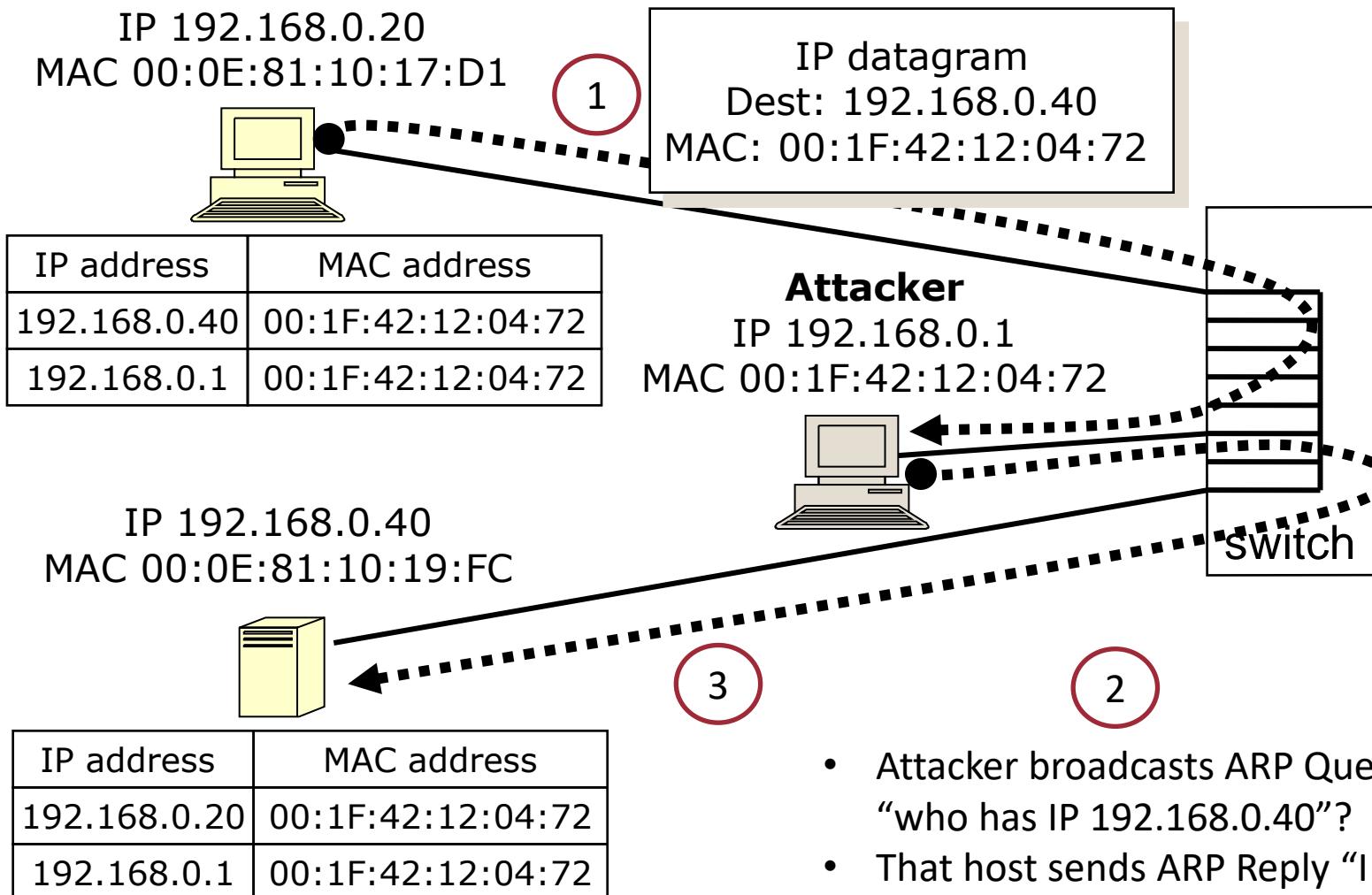
- Intercepting Traffic
 - The attacker sends forged ARP messages onto a local network
- Associating MAC Address
 - These messages associate the attacker's MAC address with the IP address of a legitimate network member, typically a router or gateway
- Redirecting Data
 - Consequently, data intended for the legitimate member is misdirected to the attacker
- Data Interception or Modification
 - The attacker can intercept, modify, or block data
- Resending Data
 - The attacker typically forwards the data to the legitimate recipient, maintaining the illusion of a normal flow, to avoid detection
 - Typically done by sending the packets to the real MAC address associated with the IP address in the packet, which the attacker has knowledge of

ARP Tables Spoofing/Poisoning

IP 192.168.0.20
MAC 00:0E:81:10:17:D1



ARP Tables Adversary-in-the-Middle attack



- Attacker broadcasts ARP Query “who has IP 192.168.0.40”?
- That host sends ARP Reply “I do, my MAC is 00:0E:81:10:19:FC”
- Switch updates its table

ARP Tables Poisoned

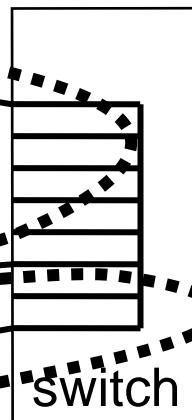
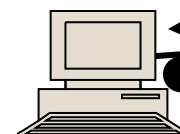
IP 192.168.0.20
MAC 00:0E:81:10:17:D1

IP datagram
Dest: 192.168.0.40
MAC: 00:1F:42:12:04:72

IP address	MAC address
192.168.0.40	00:1F:42:12:04:72
192.168.0.1	00:1F:42:12:04:72

IP 192.168.0.40
MAC 00:0E:81:10:19:FC

Attacker
IP 192.168.0.1
MAC 00:1F:42:12:04:72



IP address	MAC address
192.168.0.20	00:1F:42:12:04:72
192.168.0.1	00:1F:42:12:04:72

Attacker table

IP address	MAC address
192.168.0.40	00:0E:81:10:19:FC
192.168.0.20	00:0E:81:10:17:D1

Results from ARP Spoofing/Poisoning

- The devices 192.168.0.20 and 192.168.0.40 have **poisoned** ARP tables
- All the data sent from 192.168.0.20 to 192.168.0.40 is redirected to the attacker (Layer 2)
- The attacker may redirect the data to the intended receiver
- Neither the attacked machines nor the switch can detect the attack
- Tools example
 - dsniff - auditing and penetration testing tool set
 - Ettercap - packet sniffer and ARP cache poisoning
- In conclusion: **switches do not eliminate the sniffing problem**

A comment on “security tools”

- **dsniff** is one of many tools usable for good and bad:

dsniff

latest release: [dsniff-2.3.tar.gz \(CHANGES\)](#)
[beta snapshots](#)

Abstract

dsniff is a collection of tools for network auditing and penetration testing. dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf, and webspy passively monitor a network for interesting data (passwords, e-mail, files, etc.). arpspoof, dnsspoof, and macof facilitate the interception of network traffic normally unavailable to an attacker (e.g, due to layer-2 switching). sshmitm and webmitm implement active monkey-in-the-middle attacks against redirected SSH and HTTPS sessions by exploiting weak bindings in ad-hoc PKI.

I wrote these tools with honest intentions - to audit my own network, and to demonstrate the insecurity of most network application protocols. Please do not abuse this software.

Preventive Measures

- Do not trust Layer 2 isolation
- Use tools like **arpwatch**
 - Monitor the ARP to IP translation
 - Alert the system administrators
- Use of switches with fixed tables
 - Has a cost in **loss of flexibility**

Roadmap

- Network models
 - OSI and Internet
 - Address resolution
- **Network vulnerabilities**
 - Physical layer
 - Data link layer
 - **Network layer**

(Layer 3) Network Layer

- Topics:
 - Routers and Routing
 - IP Addresses

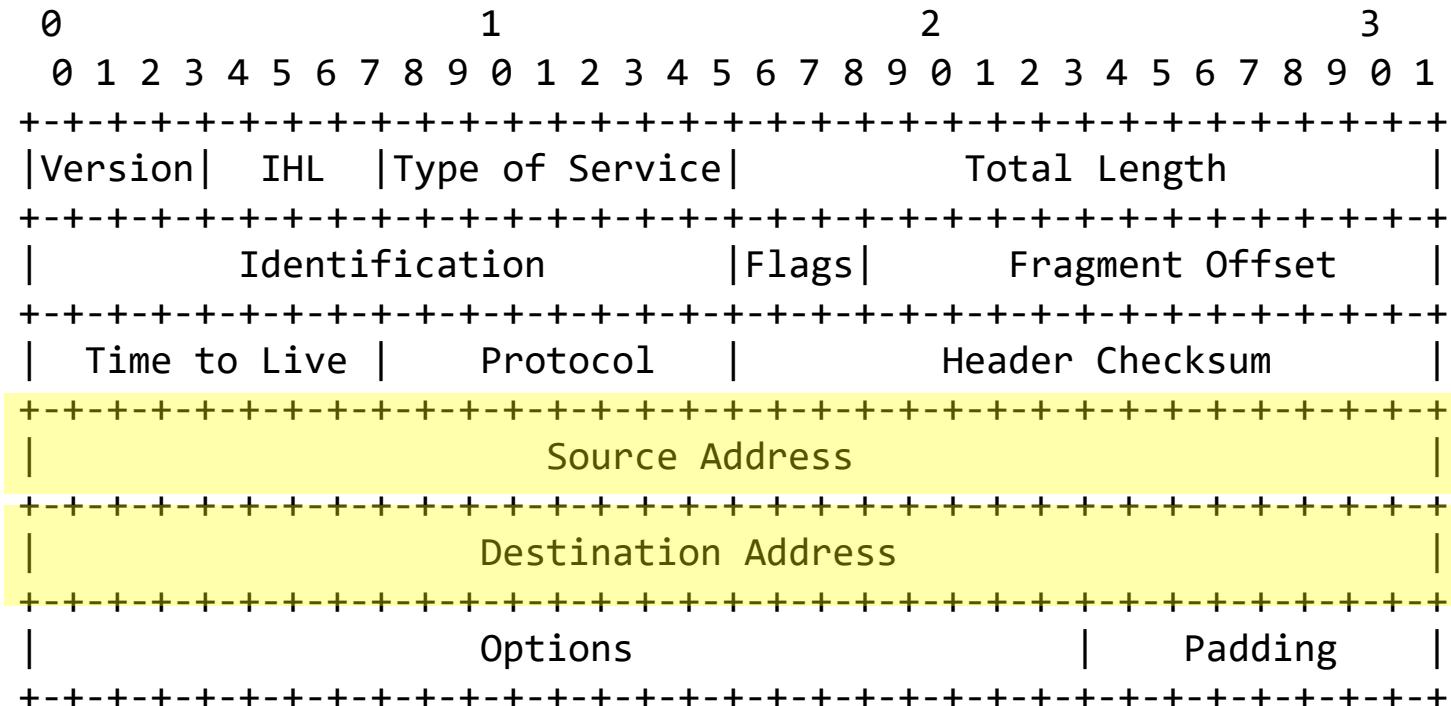
Router behavior

- Routers support the indirect delivery of IP datagrams
- Routing tables are used
- A datagram can usually be sent:
 - Directly to the final destination
 - To the next router in the direction of the destination
 - To the default router

Network Layer threats

- Packet integrity threat
 - IP spoofing
- Information leak threat
- Denial-of-Service (DoS) threat

IP packet header (RFC 791)

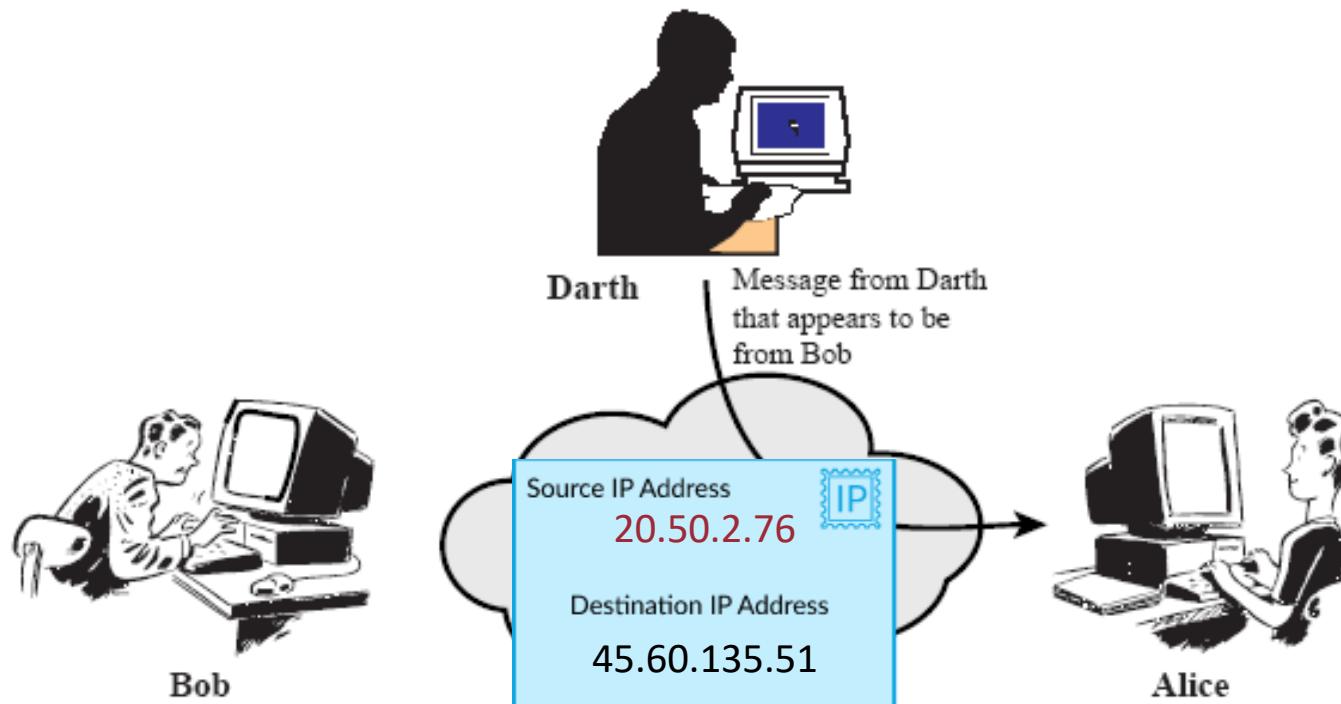


Network Layer attacks (1)

- IP spoofing:
 - Packet integrity threat
 - Data is **not** authenticated
 - Attacker can change the source address of IP packets
 - It is insecure to base access control on IP addresses
 - Attacker can *delay, reorder, replay, modify, or inject* IP packets and any of its fields

IP packet masquerade

Real IP: 45.60.65.43



(a) Masquerade

Real IP: 45.60.135.51

Network Layer attacks (2)

- Users have little to no guarantee concerning the routing path taken by the packets:
 - Information leak threat
 - DoS threat

Route hijacking

Traceroute Path 1: from Guadalajara, Mexico to Washington, D.C. via Belarus



Network Layer attacks (3)

- Route update security
 - An attacker might corrupt the routing tables by sending routing-update messages
 - ICMP redirect packets
 - Intra-domain
 - RIPv1 and IGRP do not have authentication
 - Inter-domain
 - BGP also does not have authentication; based on policy
 - DoS, Man-in-the-Middle attacks are possible

Roadmap (to be continued)

- Network models
 - OSI and Internet
 - Address resolution
- **Network vulnerabilities**
 - Physical layer
 - Data link layer
 - Network layer
 - **Transport layer**
 - **Application layer**
- Network security models

Network Vulnerabilities: OSI Layer 4 and above

Segurança Informática em Redes e Sistemas
2024/25

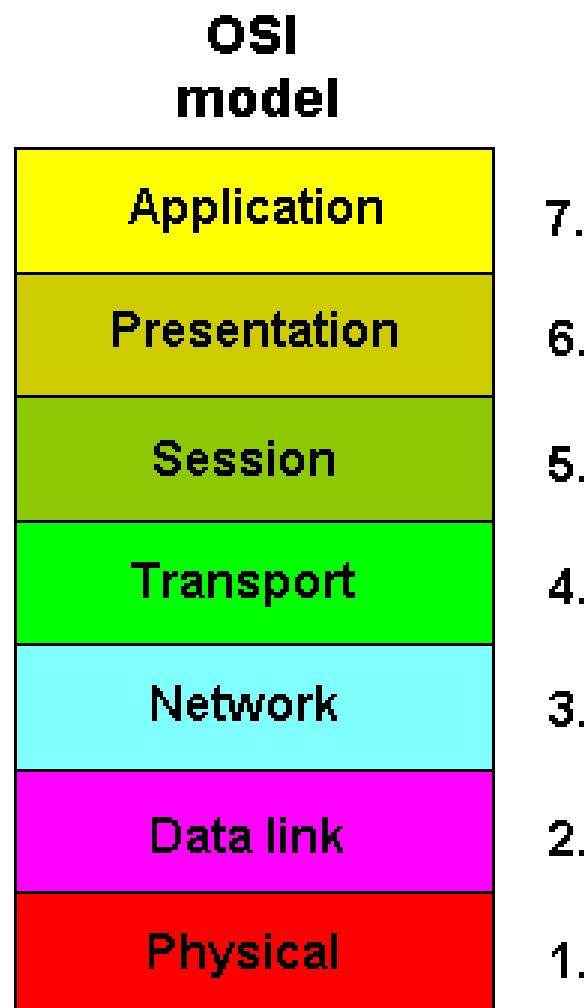
Ricardo Chaves, David R. Matos

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Roadmap

- Network vulnerabilities
 - Physical layer
 - Data link layer
 - Network layer
 - Transport layer
 - Application layer
- Network security models

OSI network model



Roadmap

- **Network vulnerabilities**
 - Physical layer
 - Data link layer
 - Network layer
 - **Transport layer**
 - Application layer
- Network security models

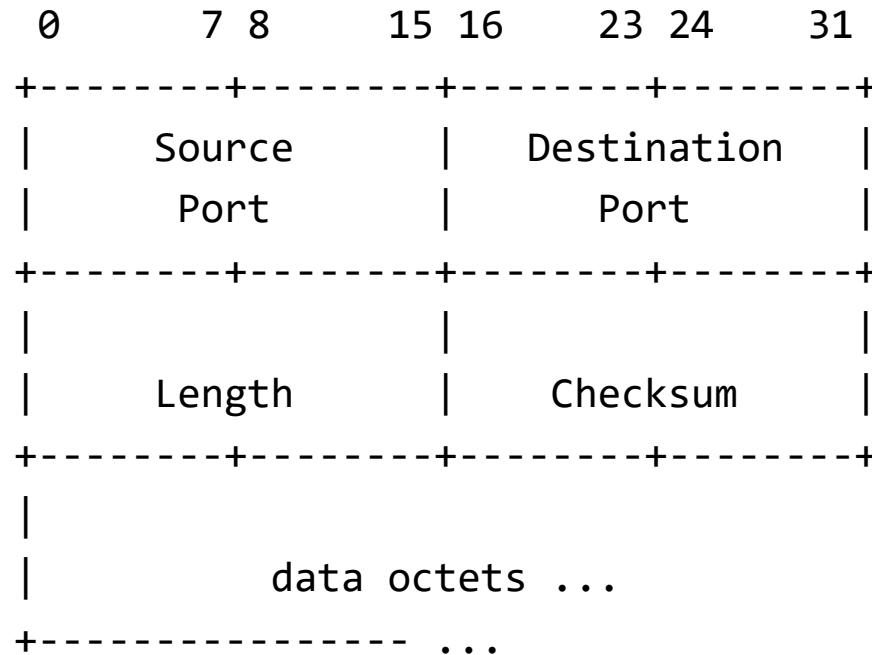
(Layer 4) Transport Layer

- Topics:
 - UDP
 - TCP
 - Handshake
 - Hijacking
 - DoS
 - TCP DoS
 - ICMP DoS
 - Solutions

UDP

- User Datagram Protocol
- This protocol can be used to send and receive individual packets, without an established connection
- It is just a thin addition to IP
 - It is vulnerable to the same attacks
 - The attacker can make any change
 - And recalculate the checksum

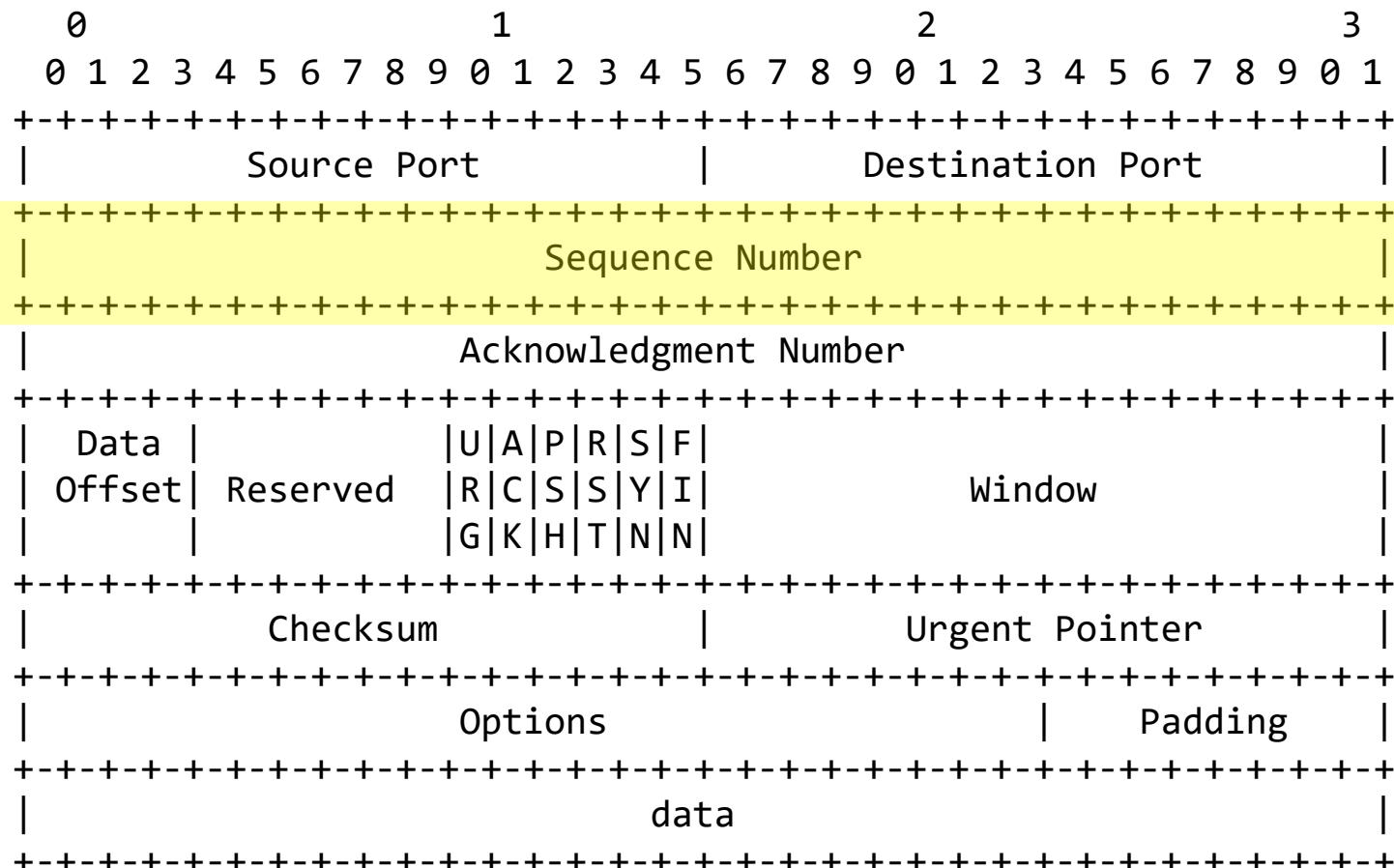
UDP header format (RFC 768)



TCP

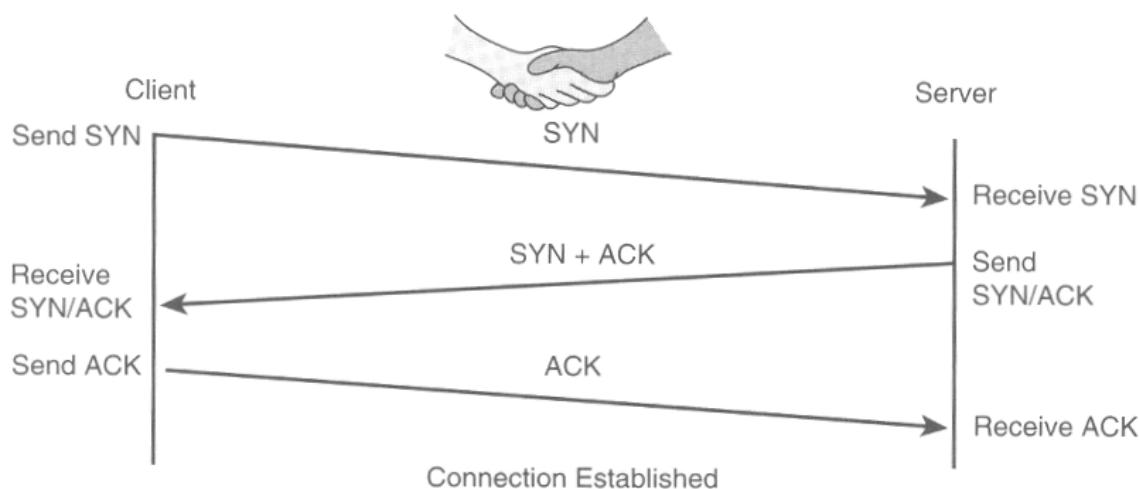
- Transmission Control Protocol
- This protocol can be used establish a connection to send and receive a data stream of bytes
 - Reliable
 - Ordered
 - Error-checked

TCP header format (RFC 793)



TCP/IP 3-way handshake

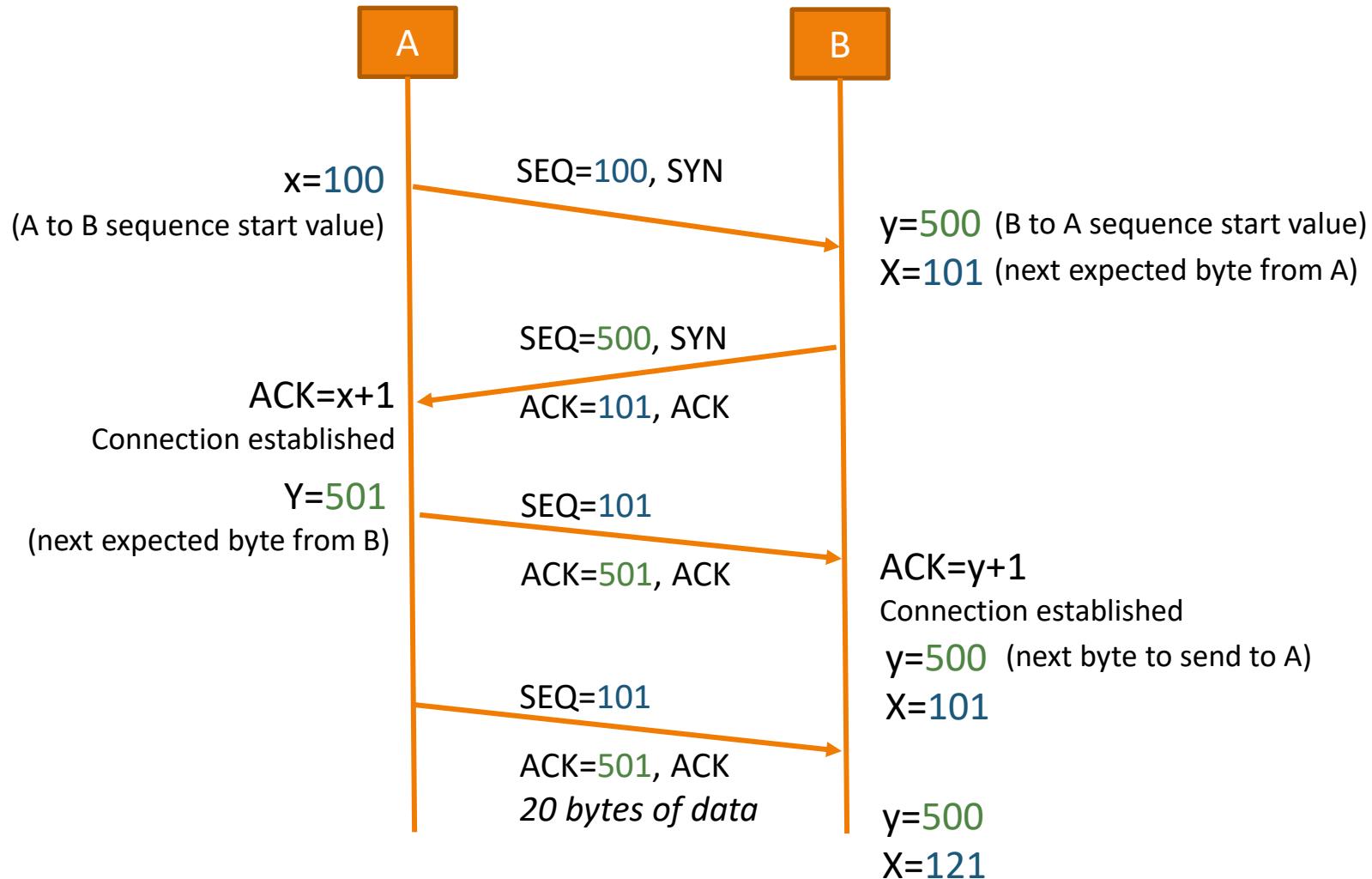
- Process used to make a connection between server and client
- SYN used to initiate and establish a connection
- ACK confirms to the other side that it has received the SYN
 - SYN-ACK is a SYN message from local device and ACK of the earlier packet
- Later, FIN is used for terminating a connection



TCP/IP handshake

- Client sends a SYN request to server with initial sequence number **x**
- Server sends the SYN/ACK packet with its own sequence number SEQ **y** and acknowledgement number ACK **x+1** for client's original SYN packet
 - The ACK indicates the next SEQ number expected from client by the server
- Client acknowledges the receipt of the SYN/ACK packet from server by sending the ACK number **y+1** which will be the next sequence number expected from server
- After the session establishment, packets are sent and received, increasing the sequence and the acknowledgement numbers accordingly

TCP handshake example



TCP connection hijacking

- There are different techniques, depending on the attacker's capability to intercept communications
 - Full adversary-in-the-middle
 - Weak adversary-in-the-middle
 - De-synchronization
 - No interception
 - Blind

Adversary-in-the-middle

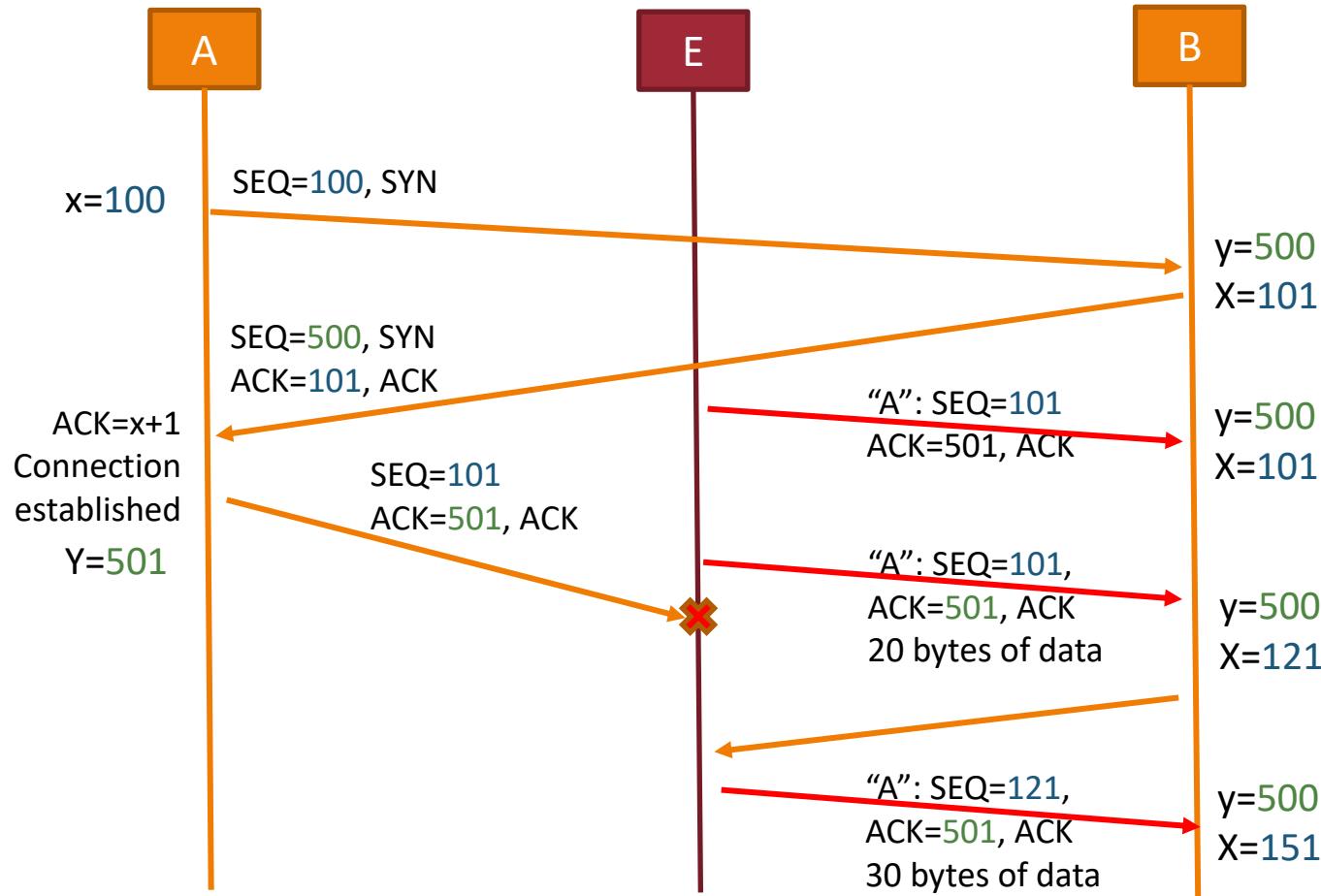
TCP hijack

- The attacker is positioned to fully intercept the communication
 - E.g. is at a network gateway
 - E.g. performs ARP poisoning in local network
- The attacker can intercept the sequence numbers and take over the connection
- Example tool:
 - shijack

shijack

- `./shijack eth0 10.0.0.2 53517 10.0.0.1 23`
 - interface you are going to hijack on
 - source IP and port of the connection
 - destination IP and port of the connection
 - [-r] Reset the connection rather than hijacking it
- Waiting for SEQ/ACK to arrive from the source to the destination
 - The tool runs and waits for another packet to get a working sequence number
 - As soon as it gets something, it will hijack the connection automatically
- `#Got packet! SEQ = 0xad6e5b8e ACK = 0x5ebaf20d`
`#Starting hijack session, Please use ^C to terminate`
`#Anything you enter from now on is sent to the hijacked`
`TCP connection`
 - Hijack of telnet session successful! Now we can send everything we want through the session to the server, like shell commands: `mkdir hello`

TCP adversary-in-the-middle example



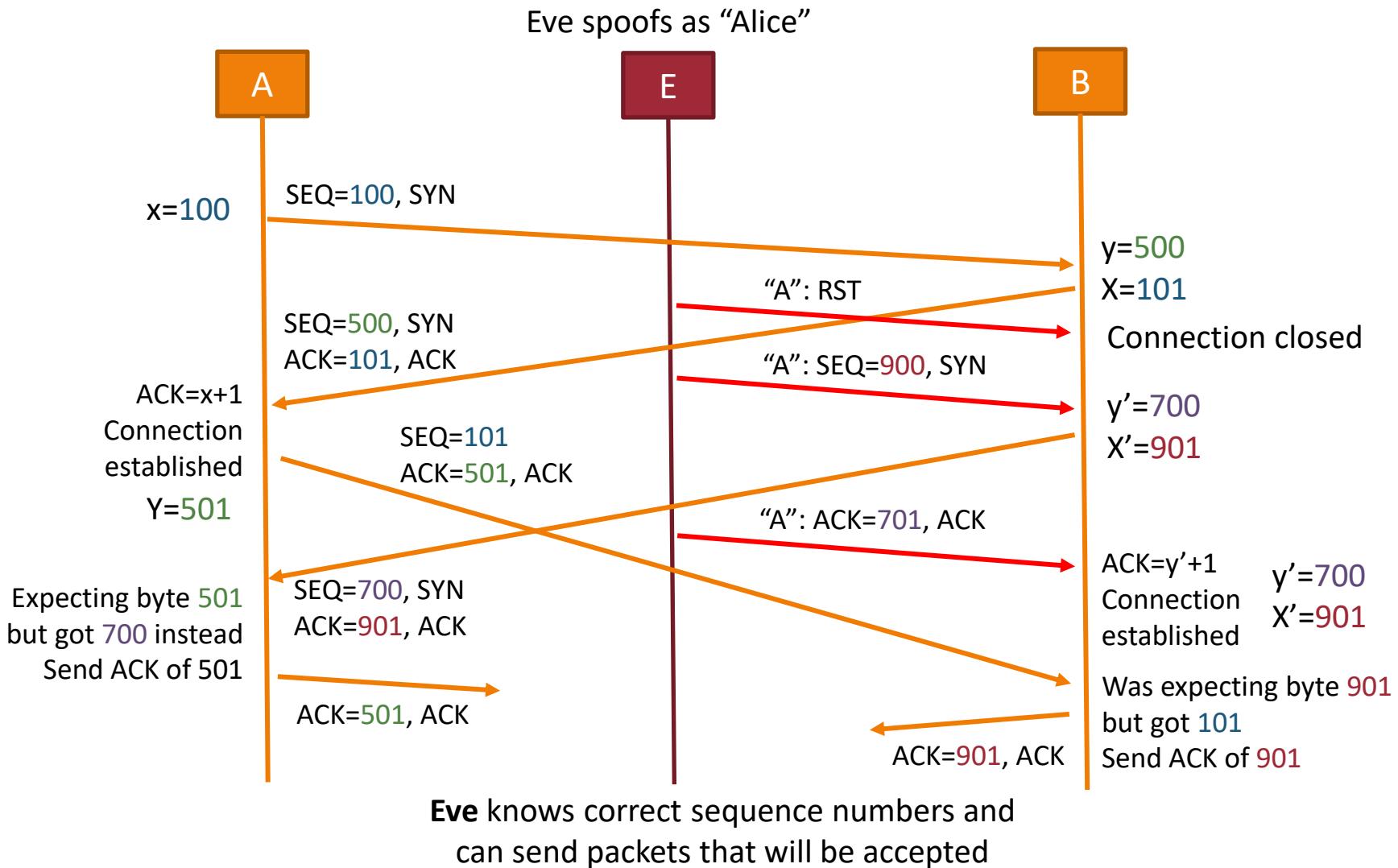
Eve can discard messages (some or all) and
can send some messages as if she were Alice or Bob

Weak adversary-in-the-middle

TCP hijack

- Attacker can only eavesdrop and spoof packets
 - The attacker is a man-in-the-middle that CANNOT drop packets
- Attacker must now exploit de-synchronization between hosts
 - Data sent out of the sliding window is discarded by the receiver
- Once the sender and the receiver are desynchronized, only the attacker can create data segments with correct numbers
 - The attacker packets are the ones that are NOT ignored
- How can we forge the de-synchronization?

TCP desynchronization example



Forging the de-synchronization

- The de-synchronization can be forged during the creation of a TCP/IP connection
 - With a reset and with false acknowledgements
- It can also be done for an already established connection
 - Send blank data to displace sliding windows
e.g. space chars are usually ignored in a telnet session
- Side-effects: receivers generate many ACK packets trying to acknowledge
 - This “TCP ACK storm” can be used to detect the de-synchronization
 - Meanwhile, the attacker is sending packets that are accepted...

Blind TCP hijack

- The attacker cannot capture return traffic from the host connection
 - The attacker is NOT an adversary-in-the-middle
- The attacker “blindly” sends malicious or manipulated packets
 - Spoofed source IP
 - Guessed sequence number
- The attacker does not receive any confirmation of the desired effect through a packet capture
- For the attack to be successful, the attacker must guess the sequence numbers of the TCP packets
 - Brute force attack on a 32-bit value
 - Unless the initial sequence numbers (ISN) is predictable...
 - *Some older Unix OSes also incremented the ISN with a time dependent algorithm*
 - October 1999 - Microsoft Security Bulletin MS99-046 – Critical
“Microsoft has released a patch that significantly improves the randomness of the TCP initial sequence numbers (ISNs) generated by the TCP/IP stack in Windows NT 4.0”

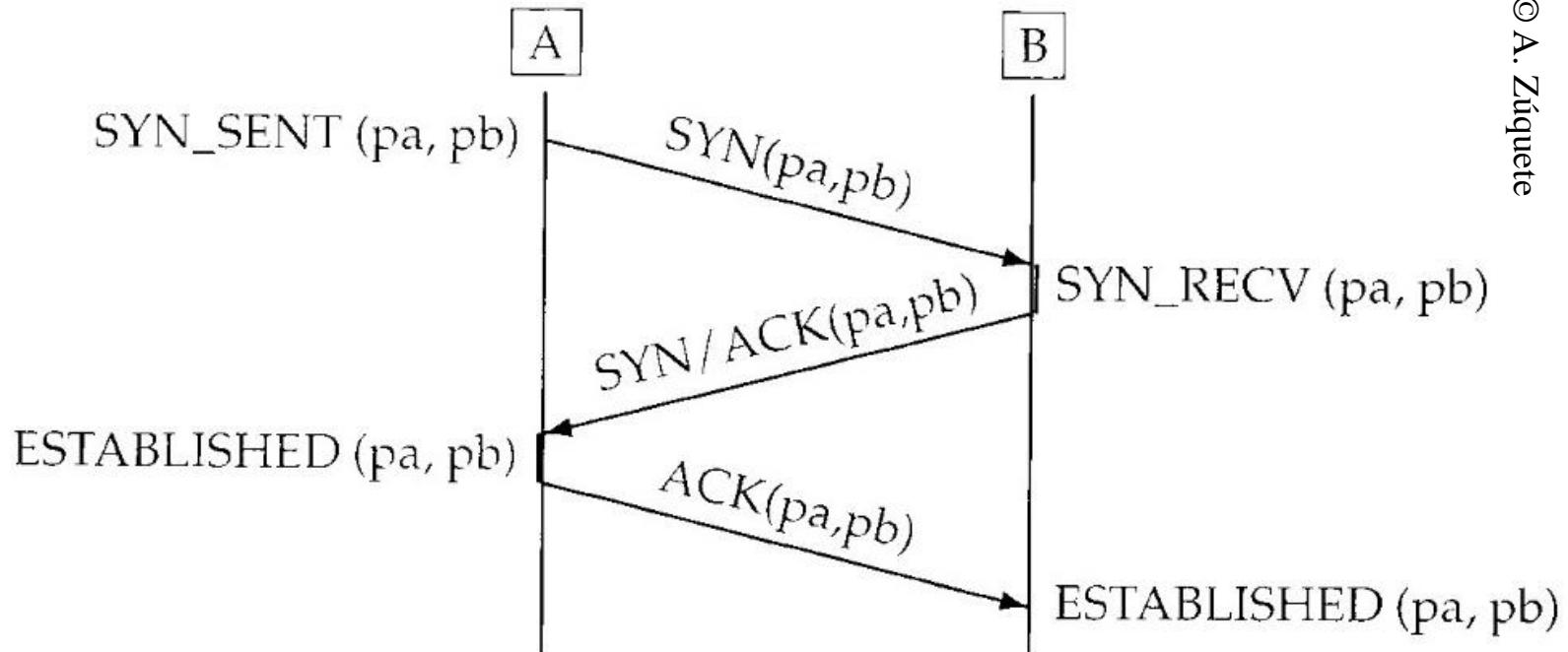
TCP connection hijacking protection

- Random generation of the ISN (initial sequence number)
 - Useful if attacker does not observe the packets
- Avoid any *host-based authentication* based on the IP address
- *Firewalls (we will see more later)*
 - Filter/discard data segments with *source-routing*
 - Use IP masquerading (NAT) for insecure connection nodes
- Protection at the IP level or higher
 - IPsec, TLS, SSH, etc.

TCP DoS attack: SYN flooding (1/2)

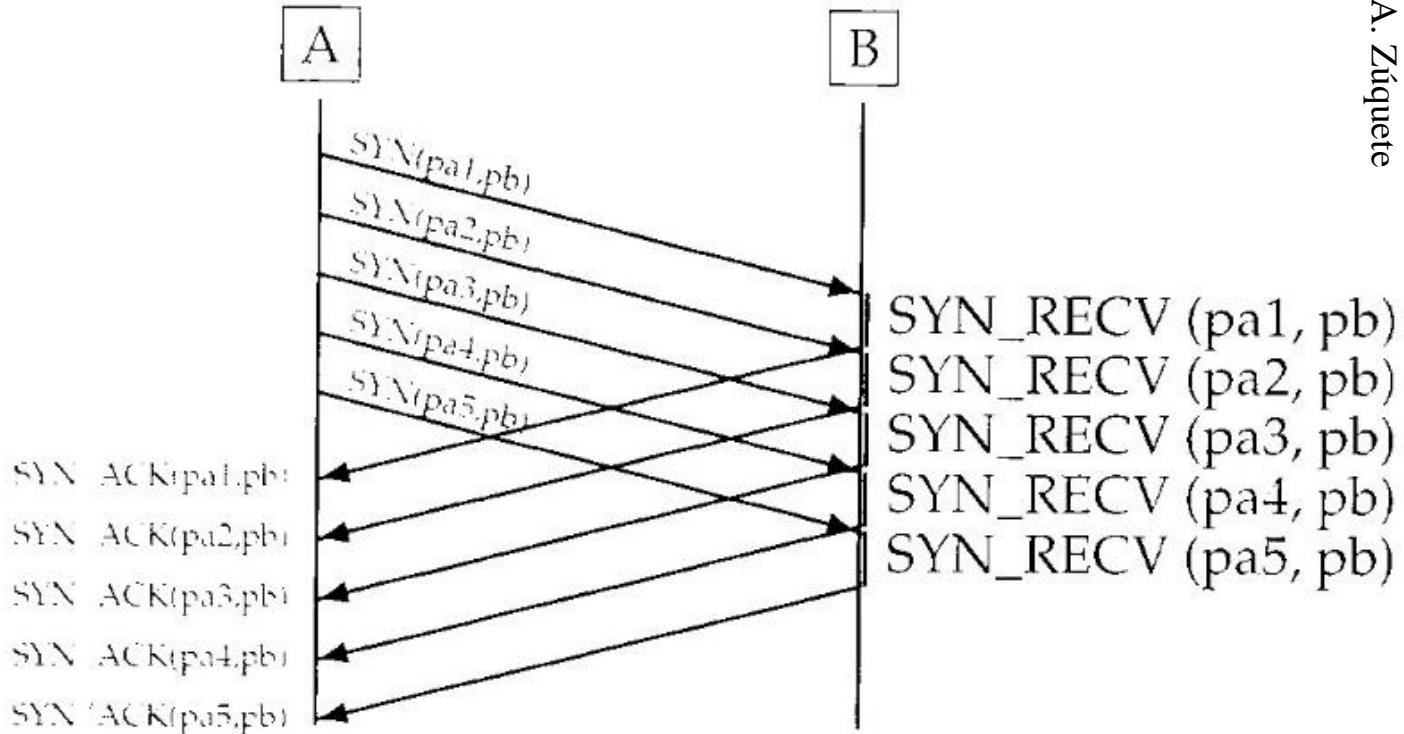
- Consists of overloading a host with incomplete TCP/IP connection requests
 - $X \rightarrow A$: SYN
 - $A \rightarrow X$: SYN+ACK
 - $X \rightarrow A$: ACK ----- missing
- Typically the attacker uses IP *spoofing*
 - Fake the sender IP
 - Often TCP is insensitive (when in the SYN_RECV state) to ICMP error messages: “host unreachable” or “port unreachable”
 - Forging one or more unused IP addresses
 - Easy to block temporarily
 - Forging random IP addresses
 - Harder to block

TCP connection (again)



pa / pb - ports

SYN flooding attack



pa1..5 and pb are port numbers

TCP DoS attack: SYN flooding (2/2)

- Explored vulnerabilities
 - No authentication in the SYN segments
 - The server needs to reserve more resources than the client/attacker
- Impact on the attacked machine
 - Storage of the connection requests until they are eliminated by timeout
 - TCP connection in the SYN_RECV state
 - The amount of connection requests per port are limited:
 - The subsequent requests are discarded
 - Correct requests may be discarded due to the existence of false connection requests

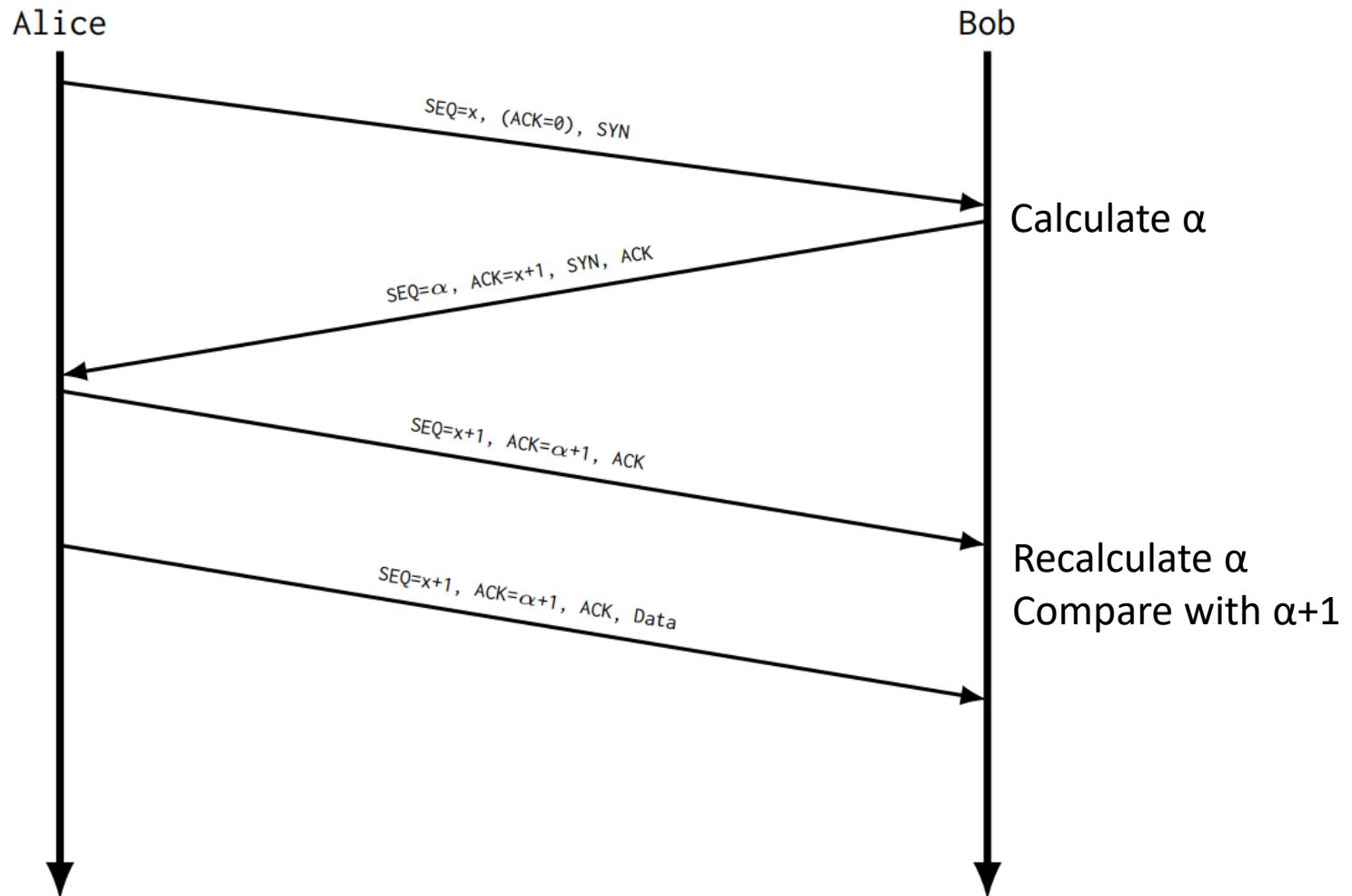
SYN flooding mitigation

- No definite solution for IPv4
- Modifying TCP for the servers
 - Bigger request queues, lower timeouts
 - Random Drop
 - **SYN cookies**
- Cooperation with firewall and attack detector

SYN flooding mitigation with SYN cookies

- SYN cookie: choice of the initial seq number by Bob
 - Bob generates the initial sequence number α such as:
 - $\alpha = h(K, SSYN)$
 - h is a one-way hash function
 - K : a secret key known only by the server
 - $SSYN$: source IP address of the SYN packet
 - At arrival of the ACK message, Bob calculates α again
 - If it knows K and received the source IP
 - Then, it verifies if the ACK number is correct
 - If yes, it assumes that the client has sent a SYN message recently and it is considered as normal behavior

Handshake with SYN cookie (RFC 4987)



SYN cookies tradeoffs

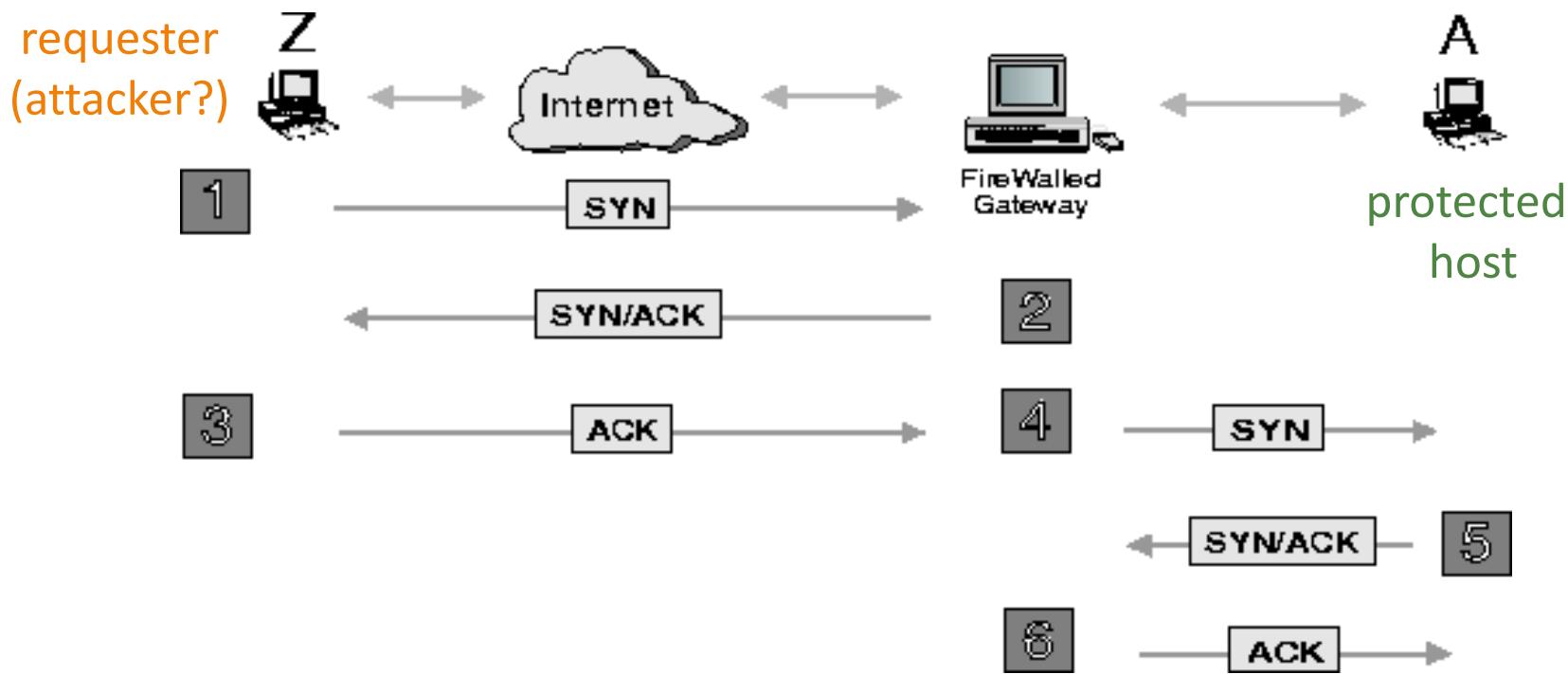
- Advantages:
 - Server does not need to allocate resources after first SYN packet
 - Client does not need to be aware that server is using SYN cookies
 - SYN cookies does not require changes in the specification of the TCP protocol
- Disadvantages:
 - Calculating α may be CPU consuming
 - Moved the vulnerability from memory overload to CPU overload
 - TCP options cannot be negotiated e.g. large window option
 - Use SYN cookies only when an attack is assumed
 - ACK/SEQ number are only 32 bit long
 - May be vulnerable to cryptoanalysis
 - The secret needs to be changed regularly

SYN flooding mitigation with firewall

- Cooperate with firewalls and attack detectors
 - Handshake relay
 - Firewall stands in front of server and protects it until the handshake is complete
 - Gateway
 - Firewall keeps the connection alive on server and terminates it if the client leaves the connection open but without traffic

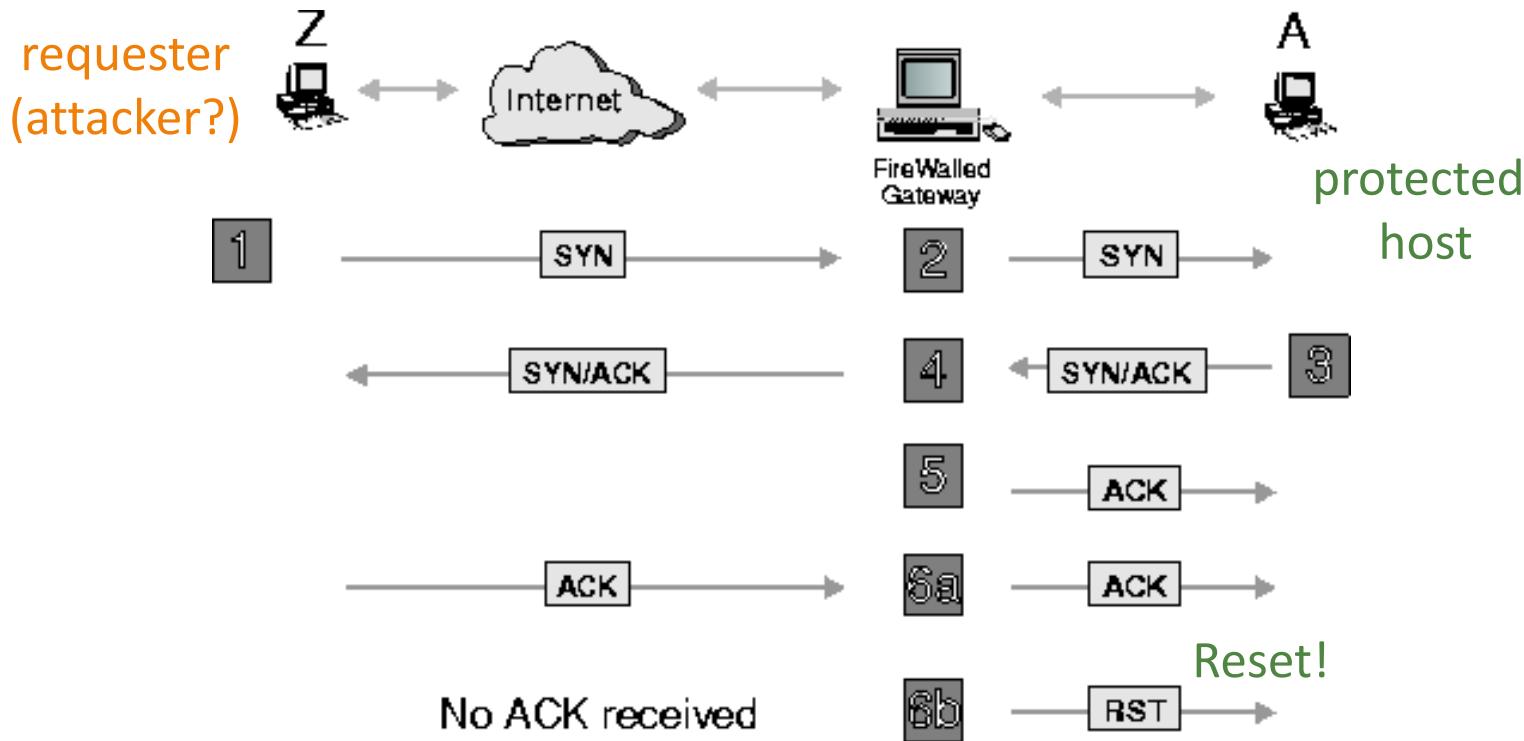
Firewall working as Handshake Relay

- Solution:
 - Firewall does handshake with requester
 - If handshake OK,
then firewall does it with protected host



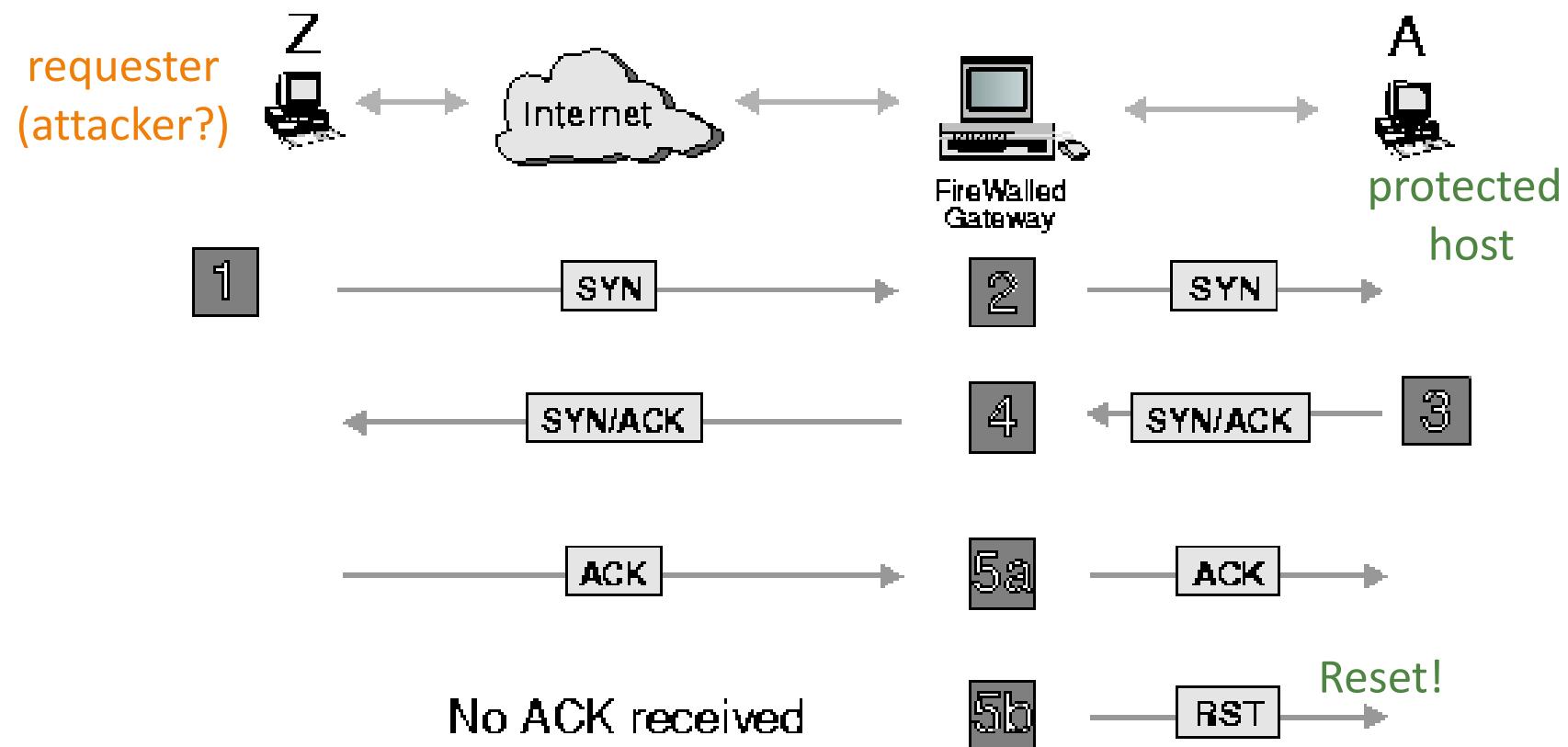
Firewall working as Gateway

- Solution:
 - Firewall does handshake with both and finishes handshake with the protected host
 - If handshake with requester not OK, resets the other



Firewall – Passive gateway

- Solution:
 - Similar but just forwards the requester's packets



Roadmap

- **Network vulnerabilities**
 - Physical layer
 - Data link layer
 - Network layer
 - Transport layer
 - **Application layer**
- Network security models

Application layer



(Layer 7) Application Layer

- Topics:
 - DNS – critical infrastructure service
 - Remote Code Execution
 - Dynamic Code Execution
 - Memory unsafety

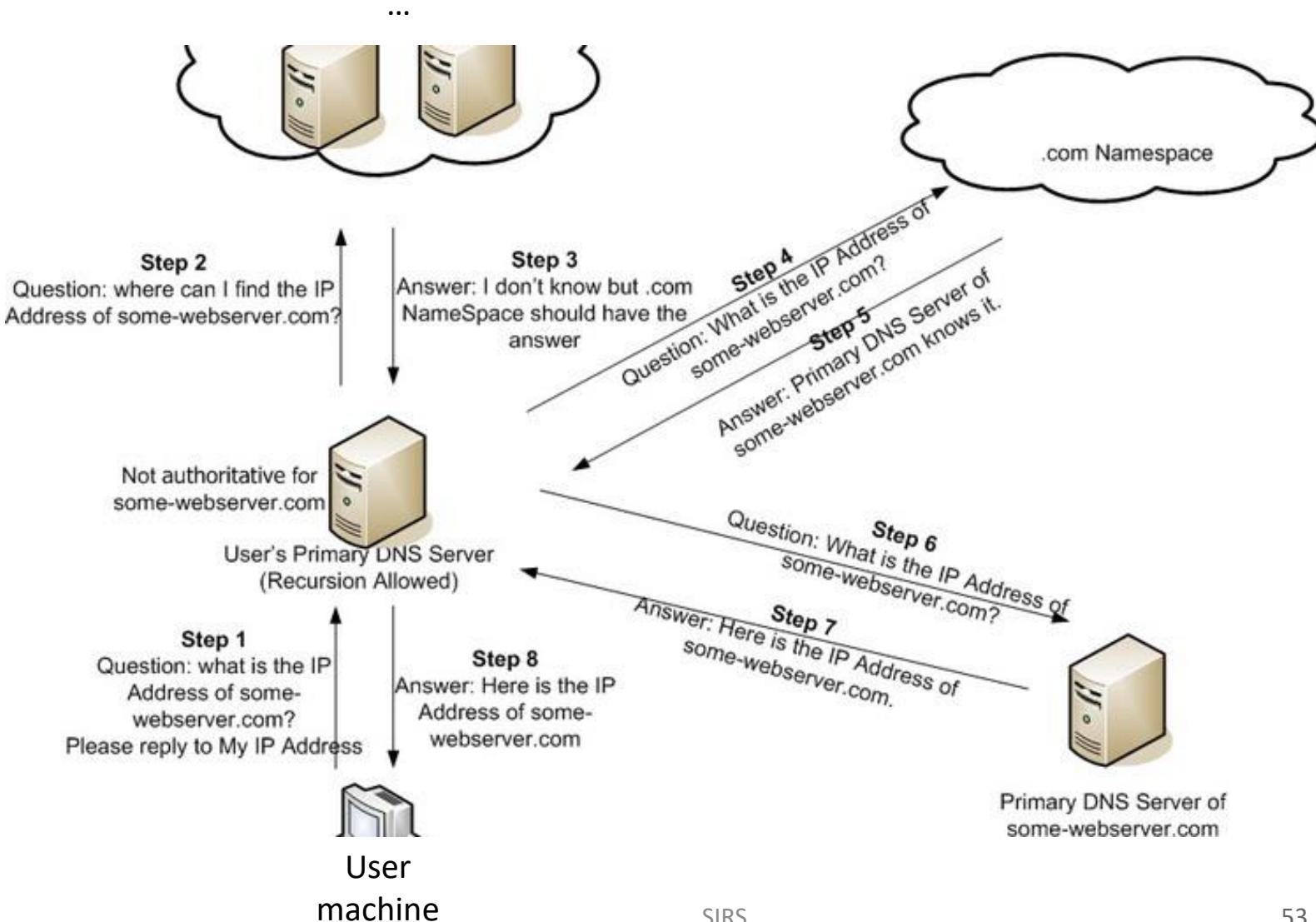
DNS (Domain Name System)

- Entities
- Resource records
- Threats
 - Kaminsky attack
- DNSSEC

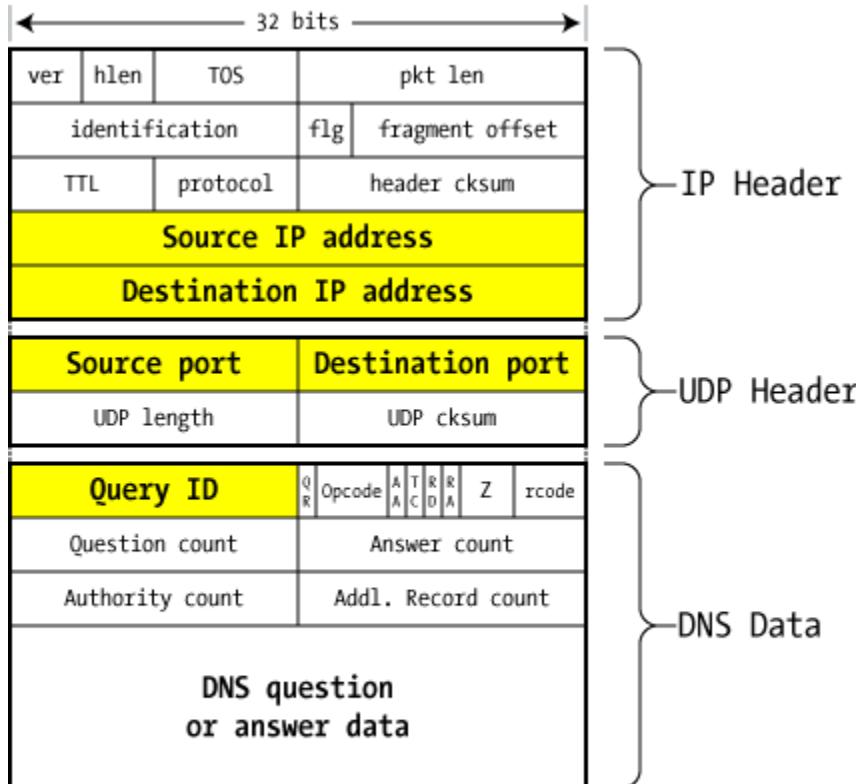
DNS in action

- Translate Domain Names to IP addresses
 - www.tecnico.ulisboa.pt  193.136.128.66
- Reverse Translation
 - 66.128.136.193.in-addr.arpa  www.tecnico.ulisboa.pt
- Mail Server Localization
 - Ricardo.Chaves@tecnico.ulisboa.pt  smtp.tecnico.ulisboa.pt
- Other name translations

DNS resolving steps

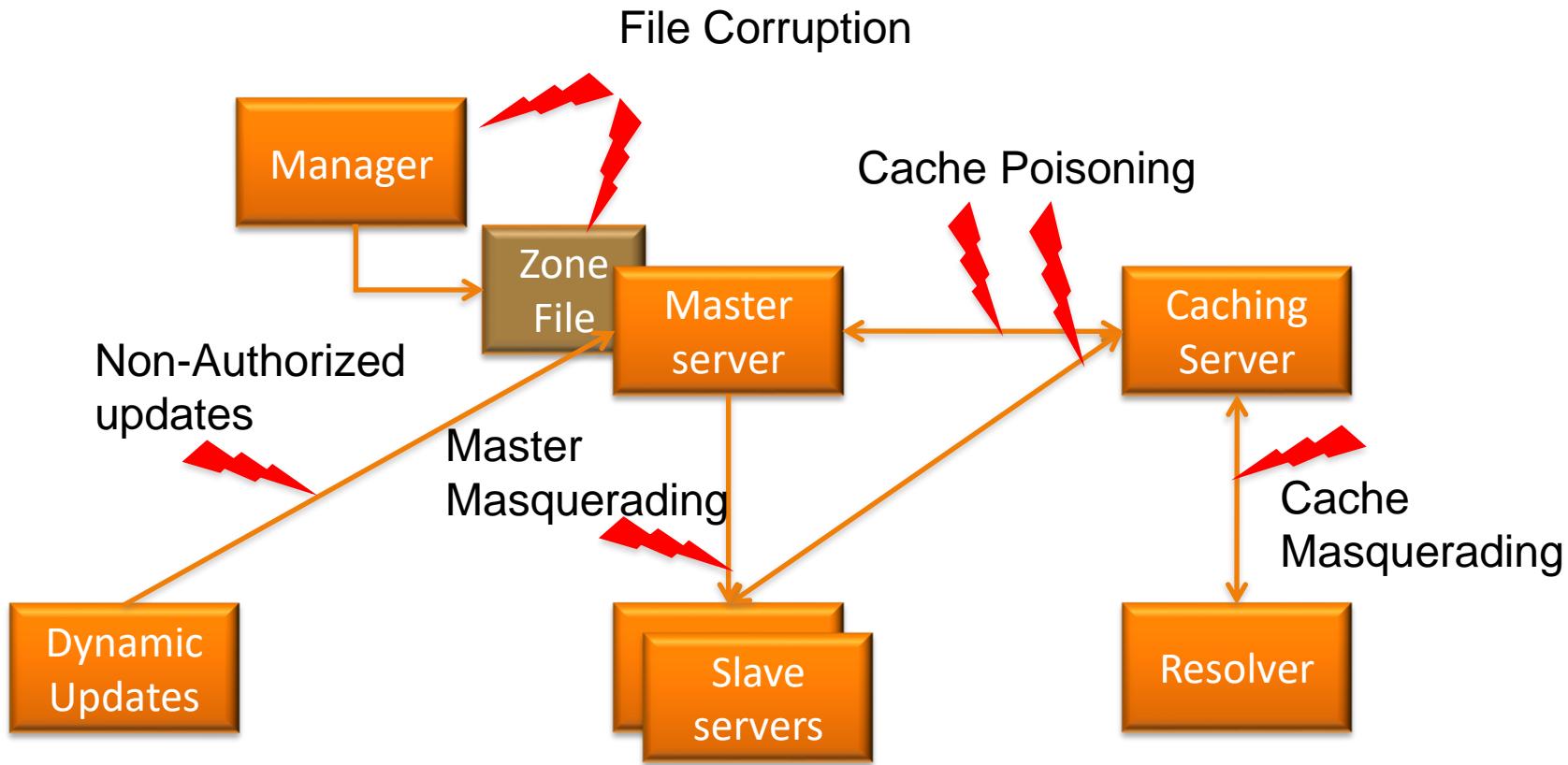


DNS Message



DNS packet on the wire

DNS Architecture Threats



Kaminsky Attack

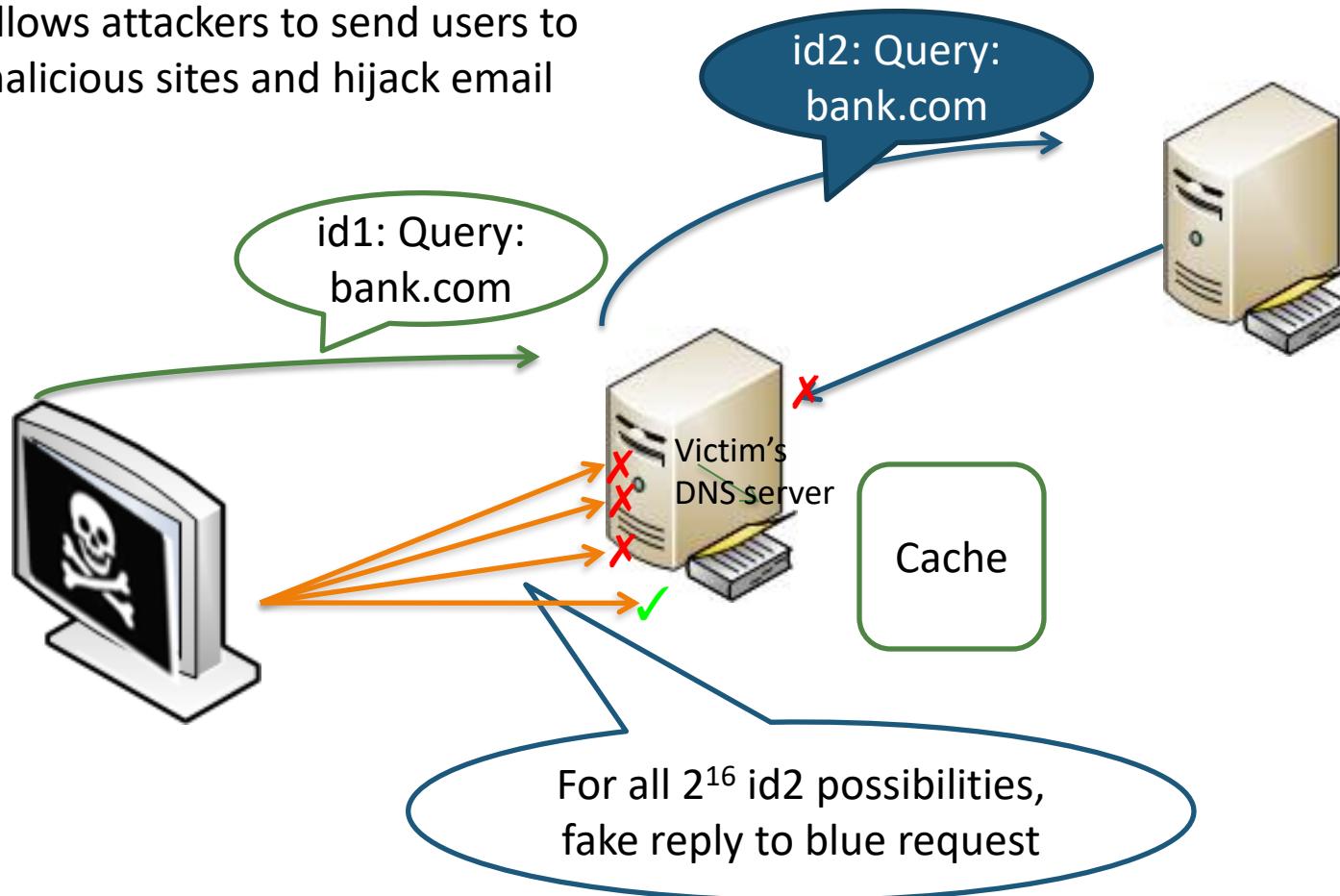
- Feb/2008 – Dan Kaminsky reports the problem
- 8/Jul/2008 – Patch for several systems
- 21/Jul/2008 – Public knowledge
- 8/Aug/2008 – Details on BlackHat
- 28/Aug/2008 – Memo for adoption of DNSSEC in .gov
- .pt – <https://www.dns.pt/pt/seguranca/dnssec/>



Dan Kaminsky
1979-2021

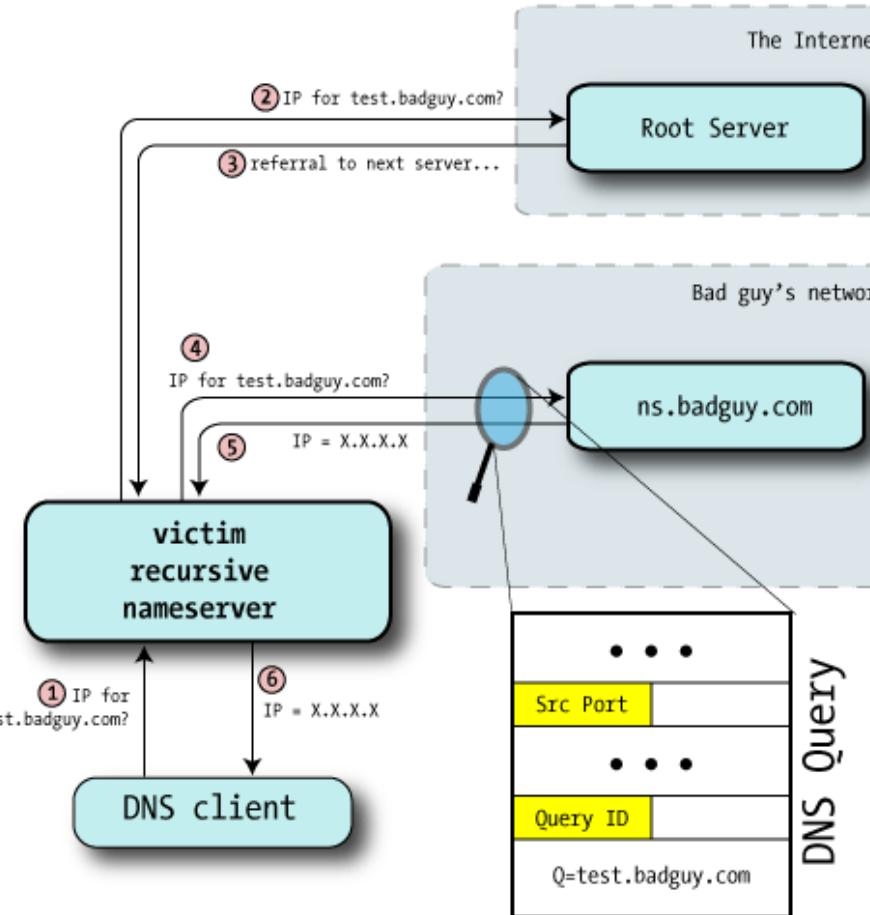
Kaminsky attack (cache poisoning)

Allows attackers to send users to malicious sites and hijack email



The attack is successful if it can guess the Query ID value

© unixwiz.net

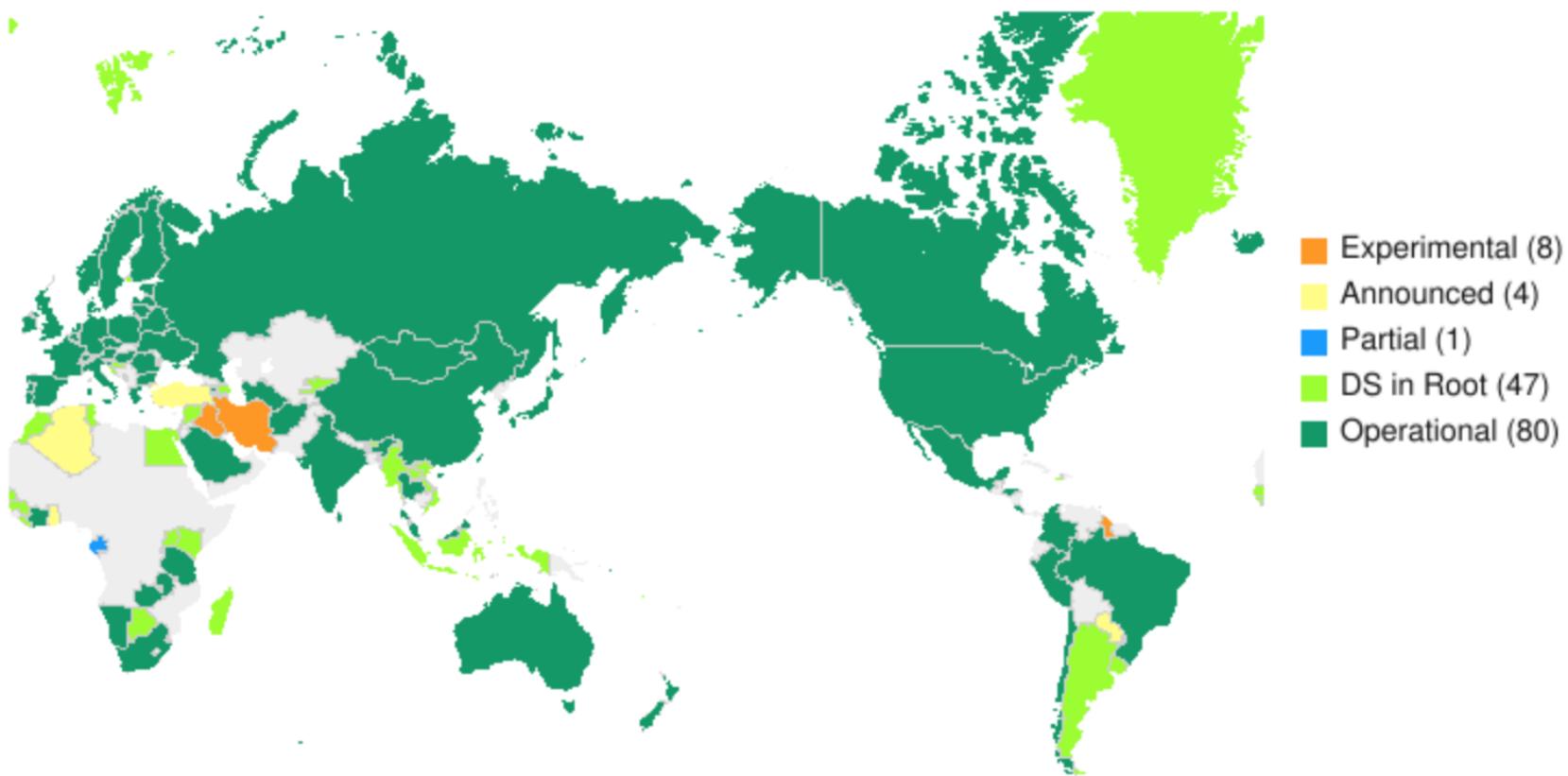


Current solution: request takes random source port and random query identifier

DNSSEC

- **DNSSEC** – DNS with digitally signed responses
 - Each zone has its own key-pair for signing
 - Responses can be validated using the respective public key
 - Public Keys are published in the DNS itself
 - As a DNSKEY Resource Record
 - One needs to get the public keys from a trusted source
 - Ideally only for the parent zones of the DNS hierarchy
 - DNSSEC provides integrity and authenticity for RRs of the signed zones
 - Does not provide more reliability, confidentiality or protection against DoS

ccTLD DNSSEC status jan. 2019



More about layer 7

- DNS is at the application layer, but it is an infrastructure service
- We also must be concerned with the application code exposed over the network

Remote Code Execution (RCE)

- RCE is a class of software security vulnerabilities
 - Much more about these in SSoft course
- RCE vulnerabilities allow a malicious actor to execute any code of their choice on a remote server machine
 - Arbitrary code execution
 - Over LAN, WAN, or **Internet**
- Exploits:
 - Dynamic code execution
 - Memory unsafety

Dynamic code execution

- Most programming languages have some way to generate code in run-time and execute it
 - E.g., parse a string as code and execute it
 - Powerful programming concept, can be very convenient
- However, a malicious actor can abuse it
 - Often, generated code is based on some user input
- If the user inputs are not vetted, then that code will be executed on the target machine
- Examples:
 - PHP code injection
 - SQL injection

Memory unsafety

- Software may have flaws when managing memory
 - Compiler, interpreter, operating system kernel or libraries
 - Virtual machines too
- Buffer overflow
 - Typically, program accepts input that is bigger than the allocated buffer
 - Memory following the buffer is overwritten
 - Program may “jump” to a different function
- An attacker can carefully craft the requests to a server to cause buffer overflow
 - Modify system memory on the affected machine
 - Cause execution of arbitrary code

Vulnerabilities inside application code

- *Dynamic code execution:*
 - using PHP (Code Injection)
 - using SQL (SQL Injection)
 - using JavaScript (XSS – Cross-site scripting)
- *Memory unsafety:*
 - using C (Overflows)

PHP – Eval Injection

vuln.php

```
<?php  
$var = "value";  
$v = $_GET['argument'];  
eval("\$var = $v;");  
?>
```

[http://victim.com/vuln.php?argument=1;phpinfo\(\)](http://victim.com/vuln.php?argument=1;phpinfo())

```
eval("value = 1; phpinfo();");
```

Attack effect: run the `phpinfo()` function

PHP – Local File Inclusion

vuln.php

```
<?php  
$page = $_GET[page];  
include($page.php);  
?>
```

```
http://victim.com/vuln.php?page=../../../../../../../../etc/passwd%00
```

Attack effect: get the content of file /etc/passwd

How to prevent code injection?

- Avoid using data as code as much as possible
- **Sanitize inputs**
 - Remove illegal characters
 - PHP now provides native filters that you can use to sanitize the data
 - Such as e-mail addresses, URLs, IP addresses, etc...



Problem goes beyond PHP

- These attacks are not exclusive to PHP programming
- They can be done whenever inputs are parsed and interpreted as code

SQL Injection

Java code

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" +  
strUsername + "' AND Password = '" + strPassword + "'"  
strAuthCheck = getQueryResult(SQLQuery)  
  
if (strAuthCheck.equals("")) bAuthenticated = false; else bAuthenticated = true;
```

Login: admin

Password: ' OR '1' = '1

SELECT Username FROM Users WHERE Username = 'admin'
AND Password = '' OR '1' = '1'

Attack effect: login as user admin without knowing the password

Login: ' OR '1'='1' ; DROP TABLE Users --

Password: does not matter!

SELECT Username FROM Users WHERE Username = ''
OR '1'='1' ; DROP TABLE Users -- ' AND Password = '???'
Attack effect: delete table Users

How to prevent SQL Injection

- Best solution is to use **prepared statements** with **parameters** that are always properly sanitized and treated as data

```
...
Set cmd = CreateObject("ADODB.Command")
cmd.CommandText = "select Username from Users where
Username=? and Password=?"

Set param1 = cmd.CreateParameter(...)
param1.Value = strUsername
cmd.Parameter.Append param1

Set param2 = cmd.CreateParameter(...)
param2.Value = strPassword
cmd.Parameter.Append param2

Set strAuthCheck = cmd.Execute
...
```

Problem goes beyond SQL

- Similar attacks can be made with other database languages, e.g.:
 - MongoDB (NoSQL)
 - Graph query language (Neo4J)
- Input values should be escaped and never used as statements

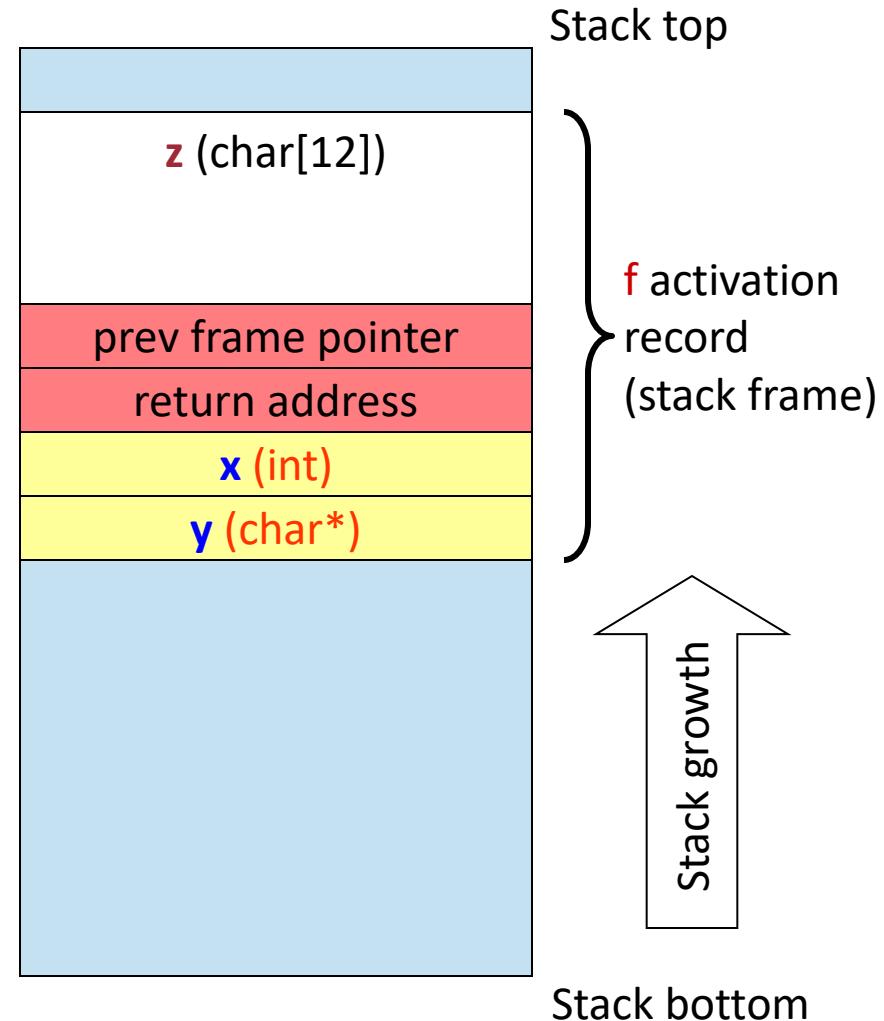
C language and overflows

- **Stack smashing**
- Heap overflow
- BSS overflow
- Print and format overflow

Overflows: Stack smashing

Standard usage:

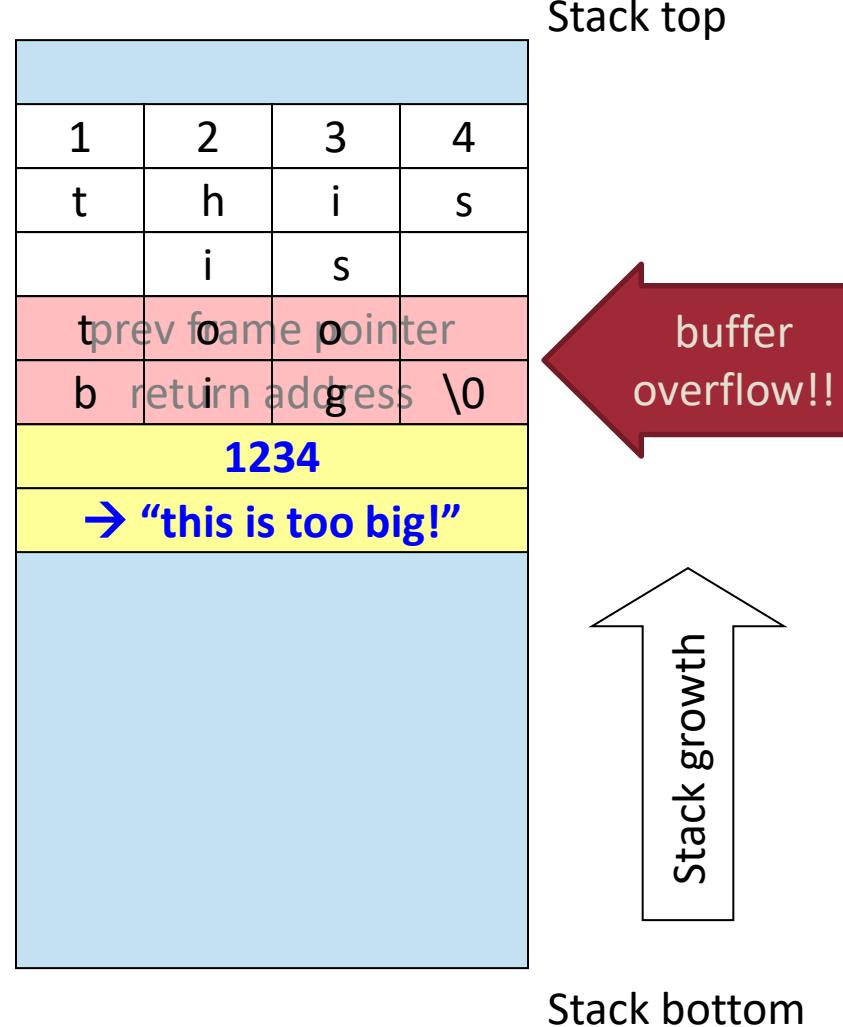
```
void f ( int x, char * y )
{
    char z[12];
    sprintf (z, "%d %s", x, y );
    write ( 2, z, strlen(z) );
}
```



Overflows: Stack smashing

```
void f ( int x, char * y )
{
    char z[12];
    sprintf (z, "%d %s", x, y );
    write ( 2, z, strlen(z) );
}

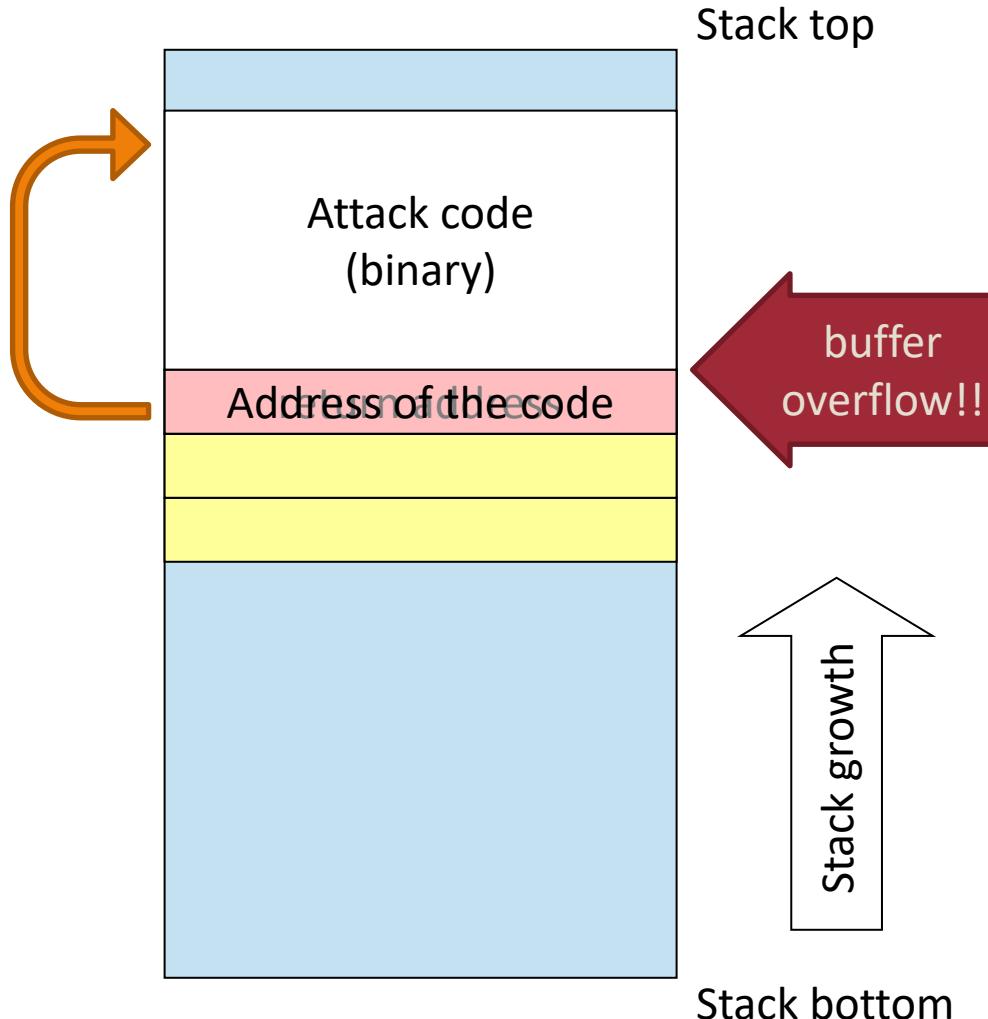
{
    ...
    f (0x1234,"this is too big");
    ...
}
```



Overflows: *Stack smashing*

Code injected by the attacker is executed!

This is worst that can happen...



C/C++ memory unsafety

- Most buffer overflow attacks target C or C++ code since these languages do not have built-in buffer size checks
- So, is this only a concern for C/C++ developers?
 - No, because most other languages end up using C/C++ libraries under the surface
 - Also makes them vulnerable to this kind of attack
- Examples
 - Python calling C libraries
 - Java Native Interface
 - Node.js engine and add-ons

How to prevent stack overflows?

- Non-executable stack
- Randomization of addresses
- Canaries for detecting tampering

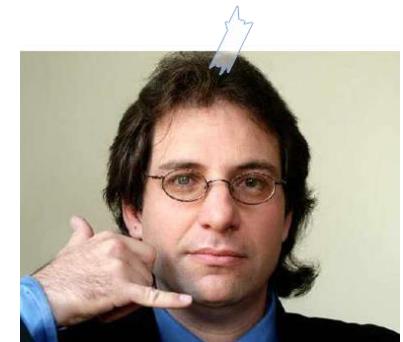
(Detailed in SSof course)

Is there a layer 8 ?

- Layer 8 *informally* refers to the “user”
 - Users are often the **weakest link** in security
- Considering layer 8 explicitly can allow IT administrators to define processes to:
 - Identify users
 - Control Internet activity of users in the network
 - Set user-based policies
 - Generate reports by user
- We can even add more layers:
 - Layer 9: The organization
 - Layer 10: Government or legal compliance

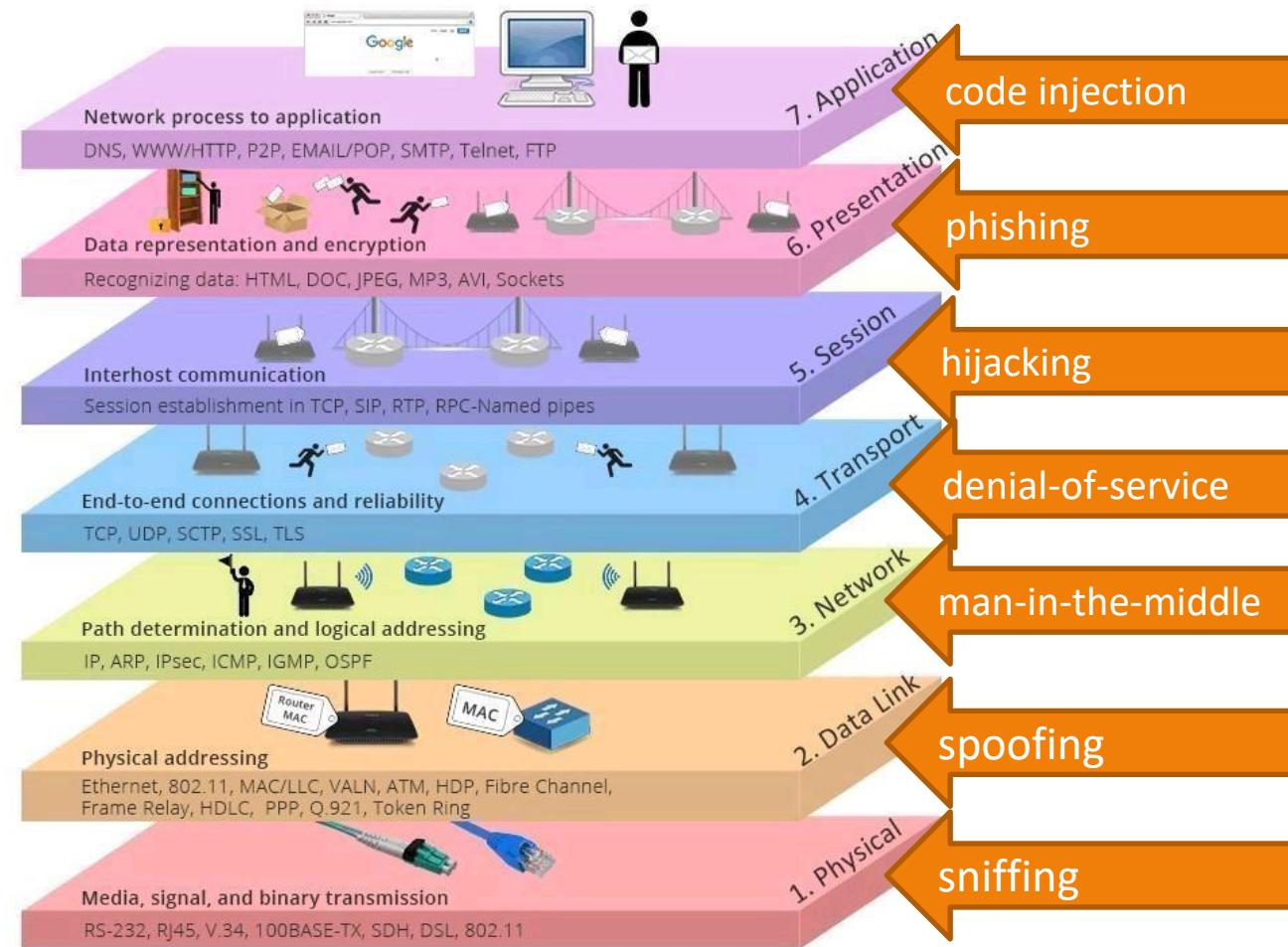
“Social Engineering”

- Psychological **manipulation** of people into performing actions or divulging confidential information
 - Trick a user to grant access to resource or reveal some secret
- Examples of social engineering attacks:
 - **Pretexting**
 - E.g., attacker claims to be part of the administrative team and asks for password to “repair” the system
 - **Baiting**
 - E.g., attacker leaves unattended USB drive with malware that the user inserts into the computer
 - **Phishing**
 - E.g., attacker sends email with malicious attachment or link to be clicked
 - **Deep fakes**
 - Voice and video



Kevin Mitnick
(1963-2023)

Example attacks on each layer

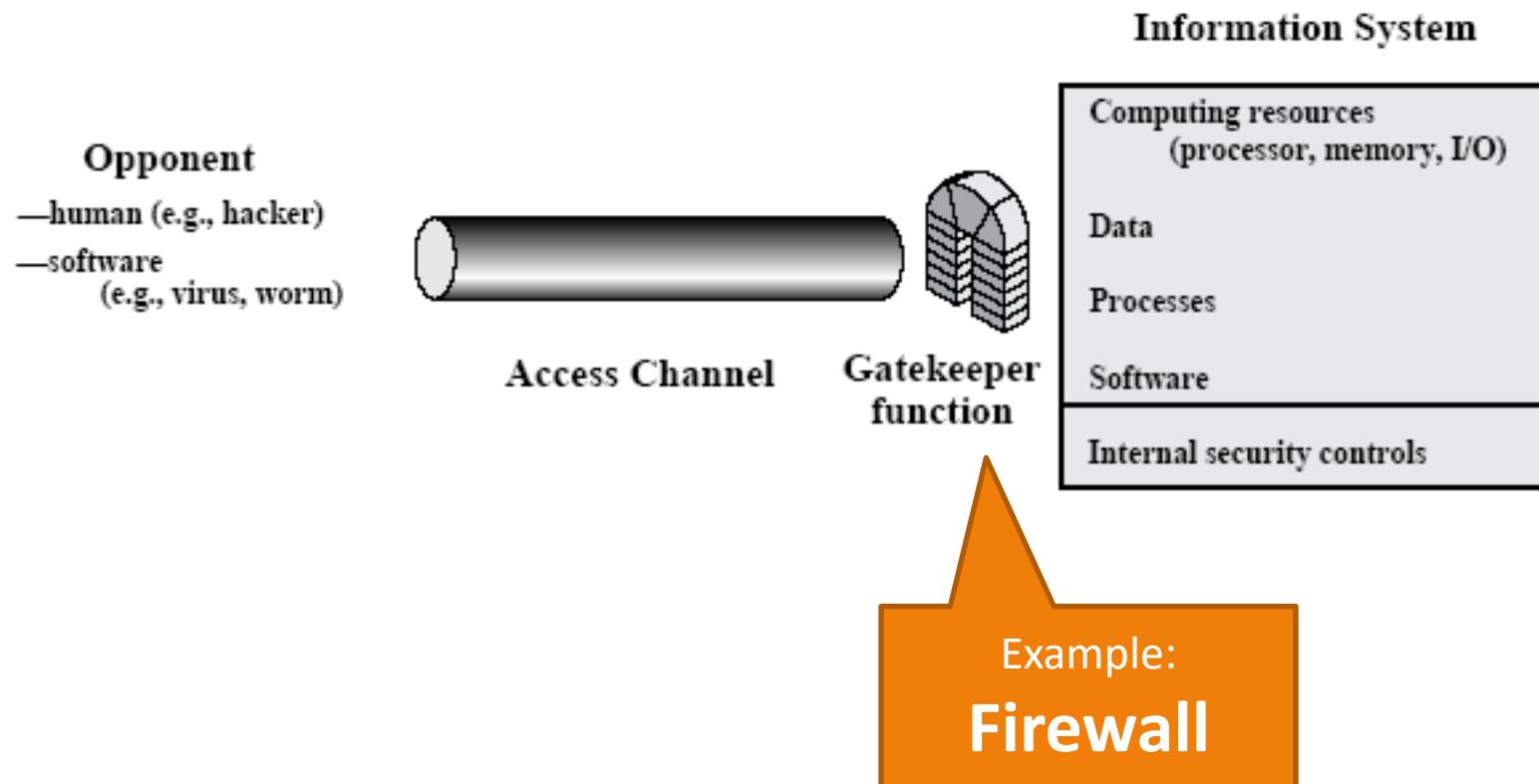


Roadmap

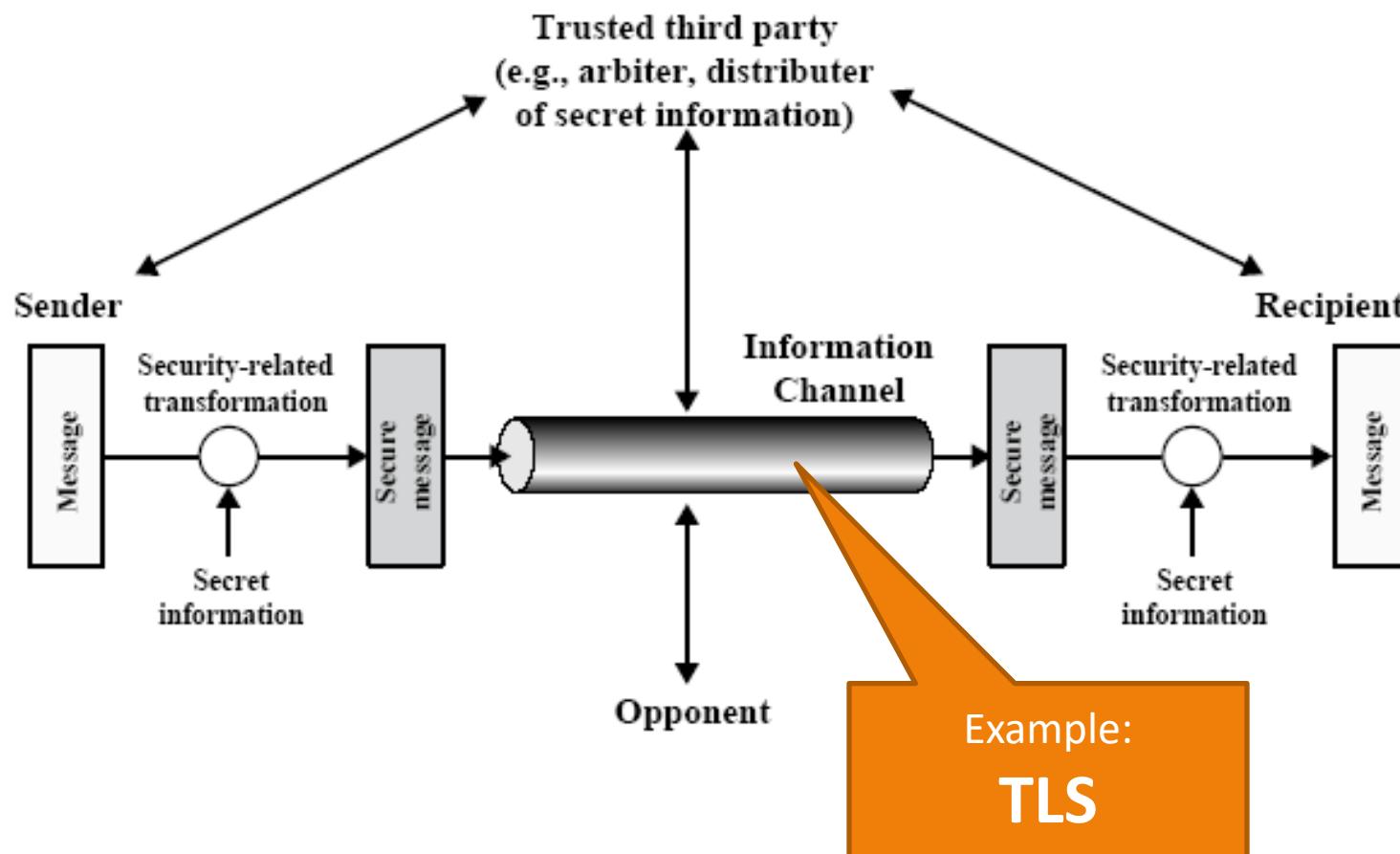
- Network vulnerabilities
 - Physical layer
 - Data link layer
 - Network layer
 - Transport layer
 - Application layer
- **Network security models**

NETWORK SECURITY MODELS

Network Security Model I: Gatekeeper for access control



Network Security Model II: Secure communication channel



Summary

- Network models
 - OSI and Internet
 - Address resolution
- Network attacks
- Network vulnerabilities
 - Physical layer
 - Data link layer
 - Network layer
 - Transport layer
 - Application layer
- Network security models

Firewalls and Intrusion Detection

Segurança Informática em Redes e Sistemas
2024/25

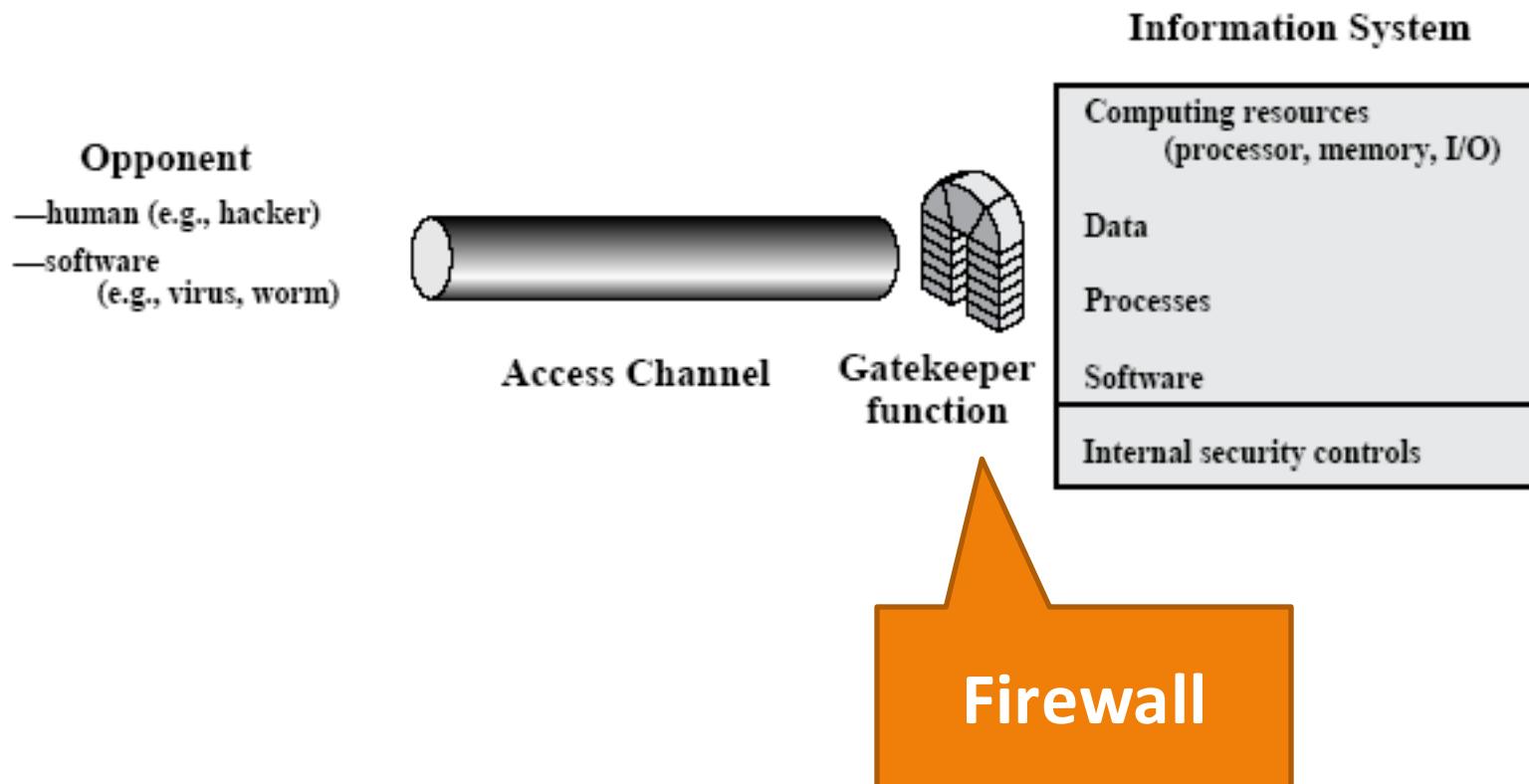
David R. Matos, Ricardo Chaves

Ack: Carlos Ribeiro, André Zúquete, Miguel P. Correia,
Miguel Pardal

Roadmap

- **Firewalls**
- Intrusion Detection Systems

Gatekeeper for access control

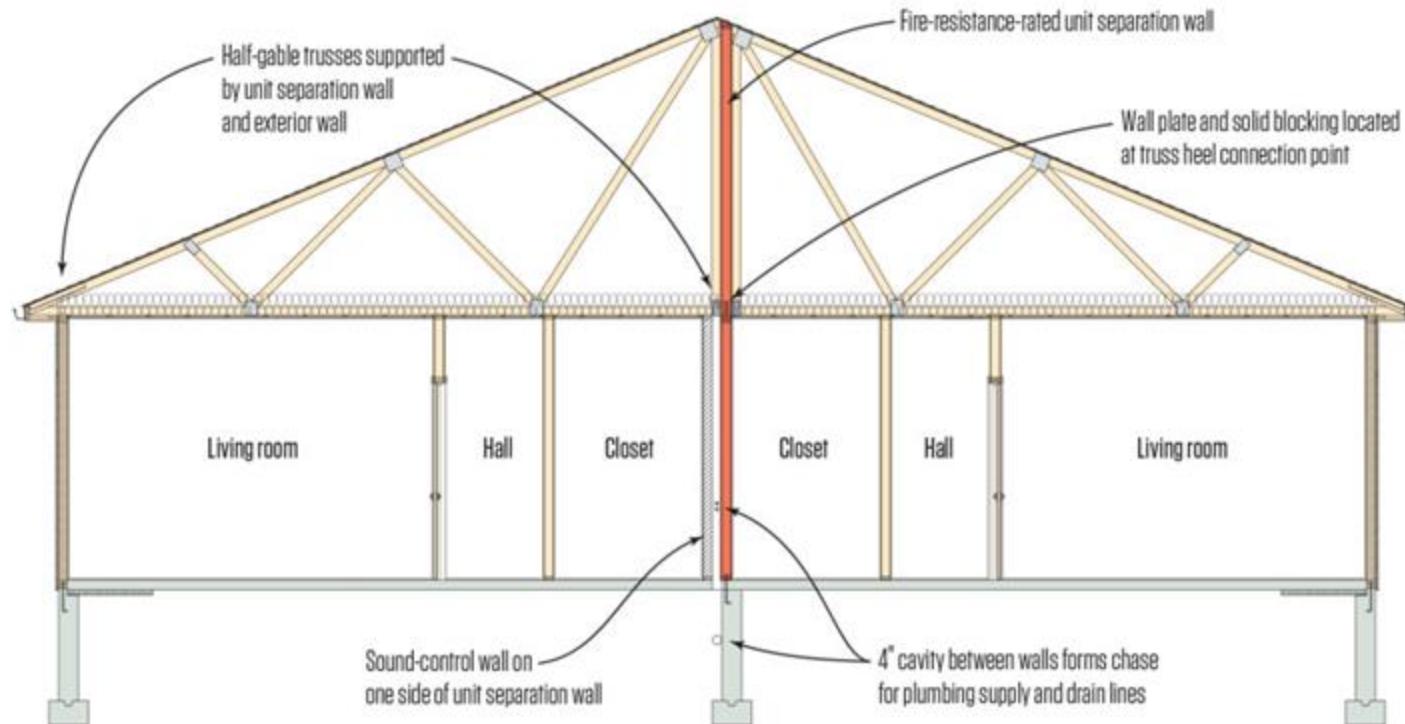


Roadmap

- **Firewalls**
 - Concept
- Intrusion Detection Systems

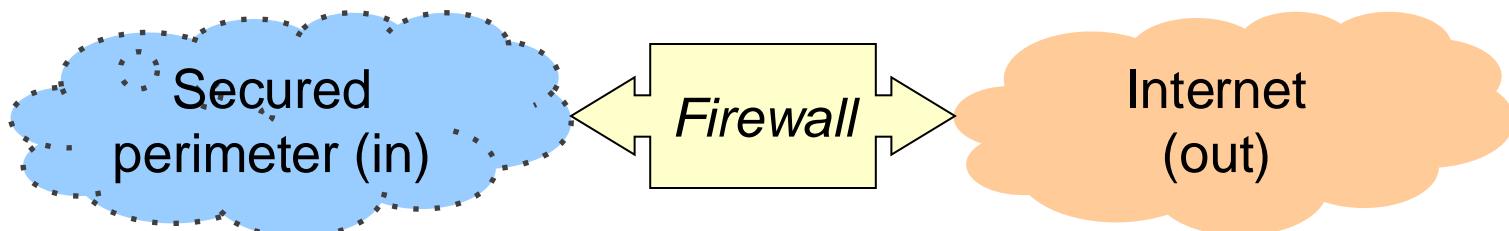
Firewall in building construction

- Wall that keeps a **fire** from spreading from one part of the building to another



Firewall

- A Firewall is a means of protecting a local system or network of systems from network threats
 - Creates a **perimeter of defense**
 - Allow only authorized access to inside network
 - Set of authenticated users/hosts
 - Prevents illegal modification/access of internal data
 - Mitigates denial-of-service attacks



Firewall analogy

- Better compared to a moat of a medieval castle
 - Prevents attackers from getting close to other defenses
 - Restricts people to enter at one carefully controlled point
 - Restricts people to leave at one carefully controlled point



Problems addressed by Firewall

- Connection of protected networks to the Internet
 - Each machine accessible from the Internet is a potential target of attacks
- Enforce access control policies
 - In a simple, scalable way
 - Authentication
 - Authorization
- Firewall allows the enforcement and implementation of security policies in a centralized manner

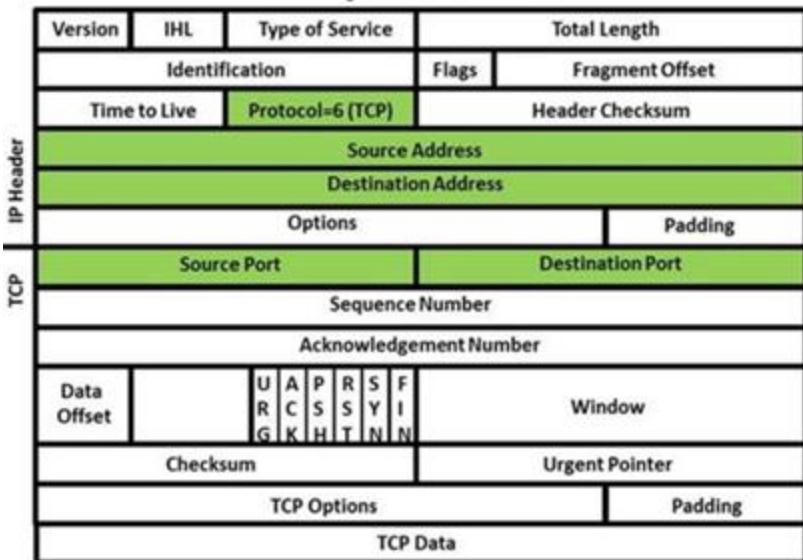
Roadmap

- **Firewalls**
 - Example: packet filter on local network
- Intrusion Detection Systems

Packet filter

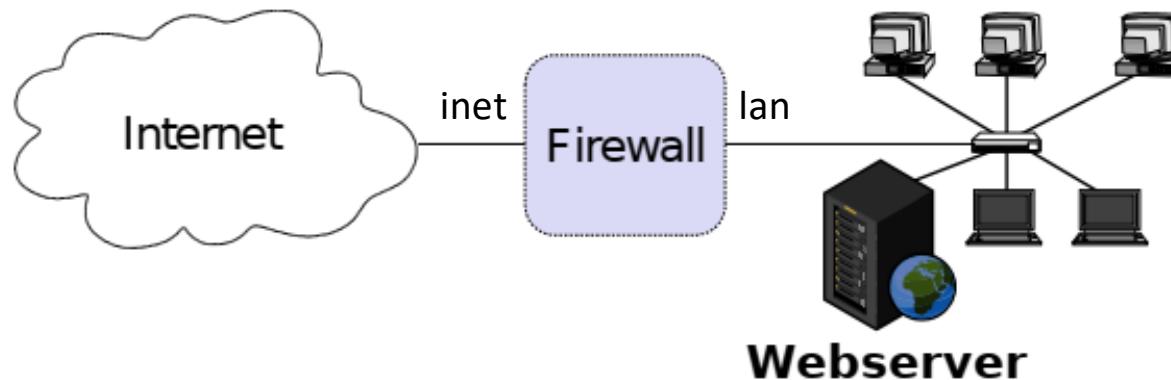
- Reject packets depending on the IP and transport layer headers:
 - IP addresses (origins and/or destinations)
 - Options in the IP header
 - Transport ports and protocols
 - Options in the headers of transport protocols
 - Direction in which virtual circuits are being created
 - Data sent via transport protocols
 - Type of the ICMP message
 - Datagram size
 - Network interface in which the data is sent/received

TCP/IP Packet



Example: small business LAN with Web Server

- Security policy
 - Allow HTTP traffic initiated by external hosts to webserver
 - Allow internal hosts to initiate HTTP and DNS
 - Do not allow other communication
 - No communication initiated by external hosts to the local hosts other than the webserver



Firewall rules table

- Example of creating rules for the firewall
- Rules are in an informal format:

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept, Drop

Rule number

Source interface:
LAN or Internet

Transport

Protocol:
UDP, TCP,
ICMP

Destination
IP address

Accept,
Drop

wildcard

Connection already established /
New connection

Example: LAN with Web Server (1)

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept

Rule 1: accept packets for already established connections

Example: LAN with Web Server (2)

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept
2	inet	external	> 1023	TCP	webserver	80	New	Accept

Rule 2: Allow HTTP traffic initiated by external hosts to web server

To fill-in this rule, we needed to know that HTTP uses TCP on port 80.

The rationale for requiring source port bigger than 1023 is because system ports should not be used by regular processes.

Example: LAN with Web Server (3)

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept
2	inet	external	> 1023	TCP	webserver	80	New	Accept
3	lan	internal	> 1023	TCP	external	80	New	Accept

Rule 3: Allow internal hosts to initiate HTTP requests to the outside

Example: LAN with Web Server (4)

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept
2	inet	external	> 1023	TCP	webserver	80	New	Accept
3	lan	internal	> 1023	TCP	external	80	New	Accept
4	lan	internal	> 1023	UDP	external	53	New	Accept

Rule 4: Allow internal hosts to initiate DNS queries to the outside

To fill-in this rule, we needed to know that DNS uses UDP on port 53

Example: LAN with Web Server (5)

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept
2	inet	external	> 1023	TCP	webserver	80	New	Accept
3	lan	internal	> 1023	TCP	external	80	New	Accept
4	lan	internal	> 1023	UDP	external	53	New	Accept
5	*	*	*	*	*	*	*	Drop

Final rule: drop all other packets

(rules are executed from top to bottom)

Complete example: LAN with Web Server

Rule	Itf.	Source IP	Source Port	Transp. Proto.	Dest. IP	Dest. Port	State	Action
1	*	*	*	*	*	*	Establ.	Accept
2	inet	external	> 1023	TCP	webserver	80	New	Accept
3	lan	internal	> 1023	TCP	external	80	New	Accept
4	lan	internal	> 1023	UDP	external	53	New	Accept
5	*	*	*	*	*	*	*	Drop

Common configuration errors 1/2

- How is your firewall management interface reachable?
 - From the Internet? From the complete internal network?
 - Via telnet? Via UPnP?
- What is allowed over the Internet?
 - NetBIOS? NFS? RPC? Telnet?
 - Other ICMP than Unreachable, Fragmentation Needed, TTL Exceeded, Ping?
 - IP header options?

Common configuration errors 2/2

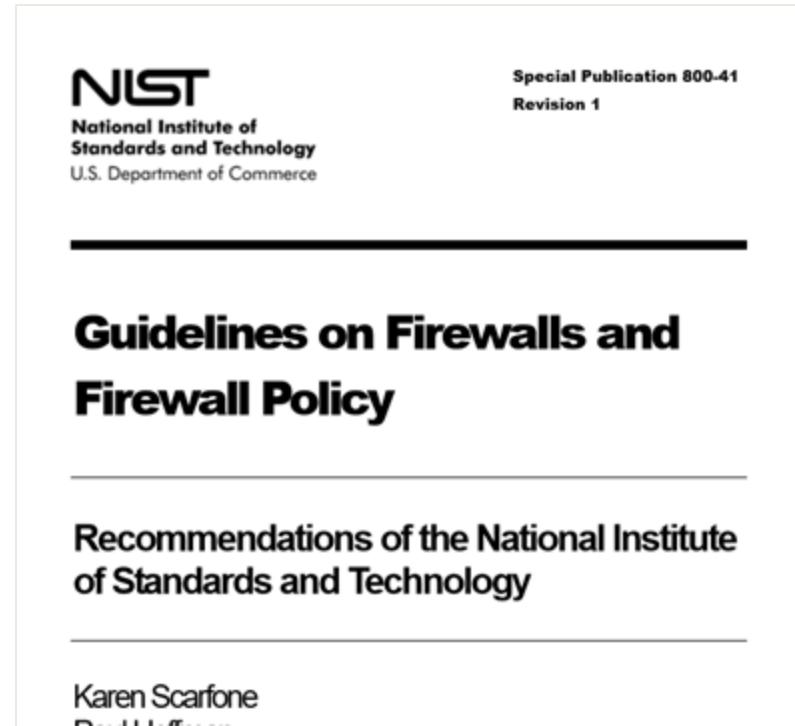
- IPv4 and IPv6?
 - Are the rule sets compliant?
- Outbound rule ANY?
 - Even private IP ranges or IP ranges that do not belong to you?
- Policy vs. Firewalls understanding of Inbound and Outbound?
 - If eth0 is your internal interface and the firewall says inbound on eth0, policy might say outbound.

Firewall rules for TCP and UDP

- Use deny by default policies for incoming TCP and UDP traffic
 - Use less stringent policies for outgoing TCP and UDP traffic
 - Most organizations permit their users to access a wide range of external applications
- Block/report malformed UDP and TCP traffic
 - Frequently used to scan for hosts
 - May also be used in certain types of attacks

Guidance for defining rules

- Defining firewall rules / policy
 - What rules should be included?
- NIST SP 800-41 document
 - Guidelines on Firewalls and Firewall Policy
 - General principles

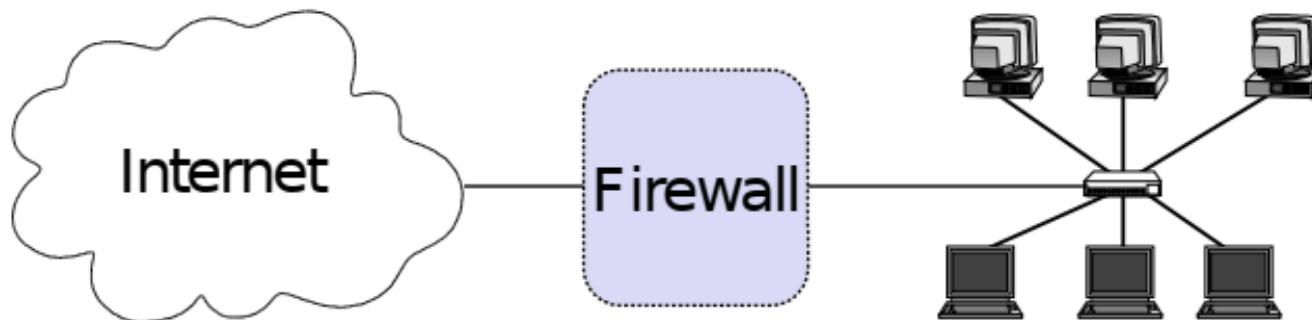


Roadmap

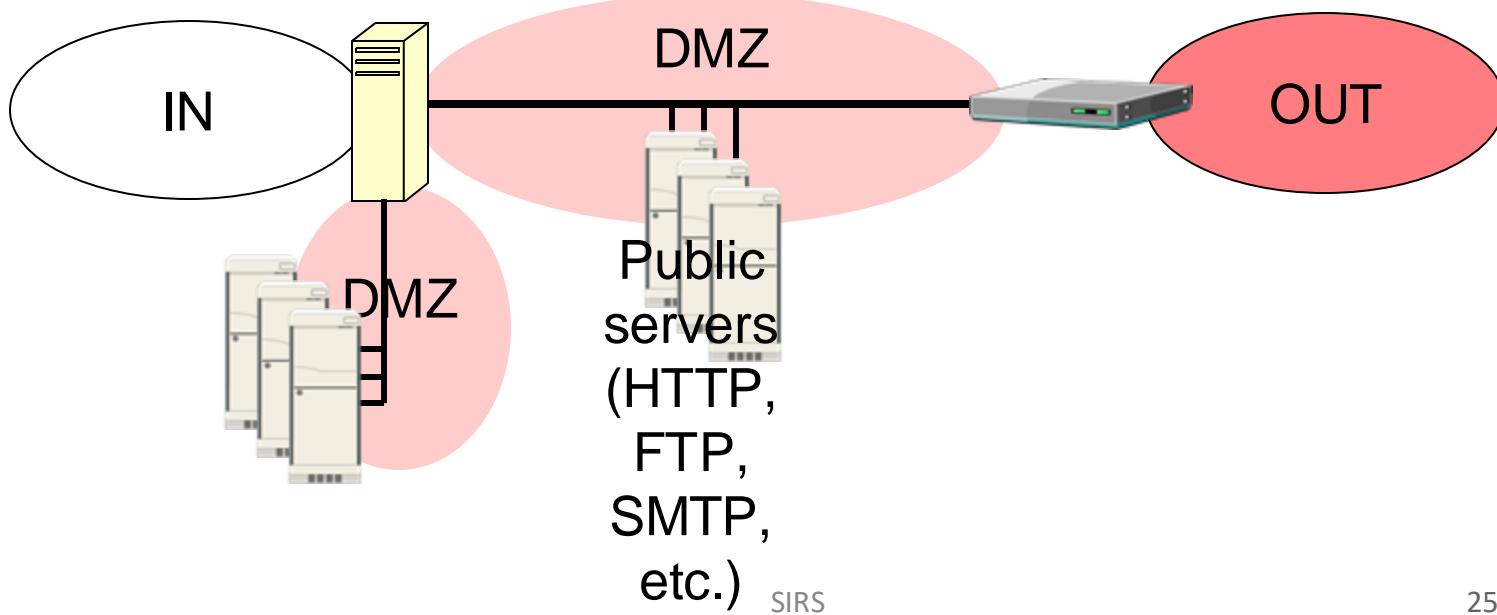
- **Firewalls**
 - Placement (topologies)
- Intrusion Detection Systems

Placement of Firewalls

- Install where a protected subnetwork is connected to a less trusted network
 - If not specified otherwise, we assume Firewall is placed between Internet and local network

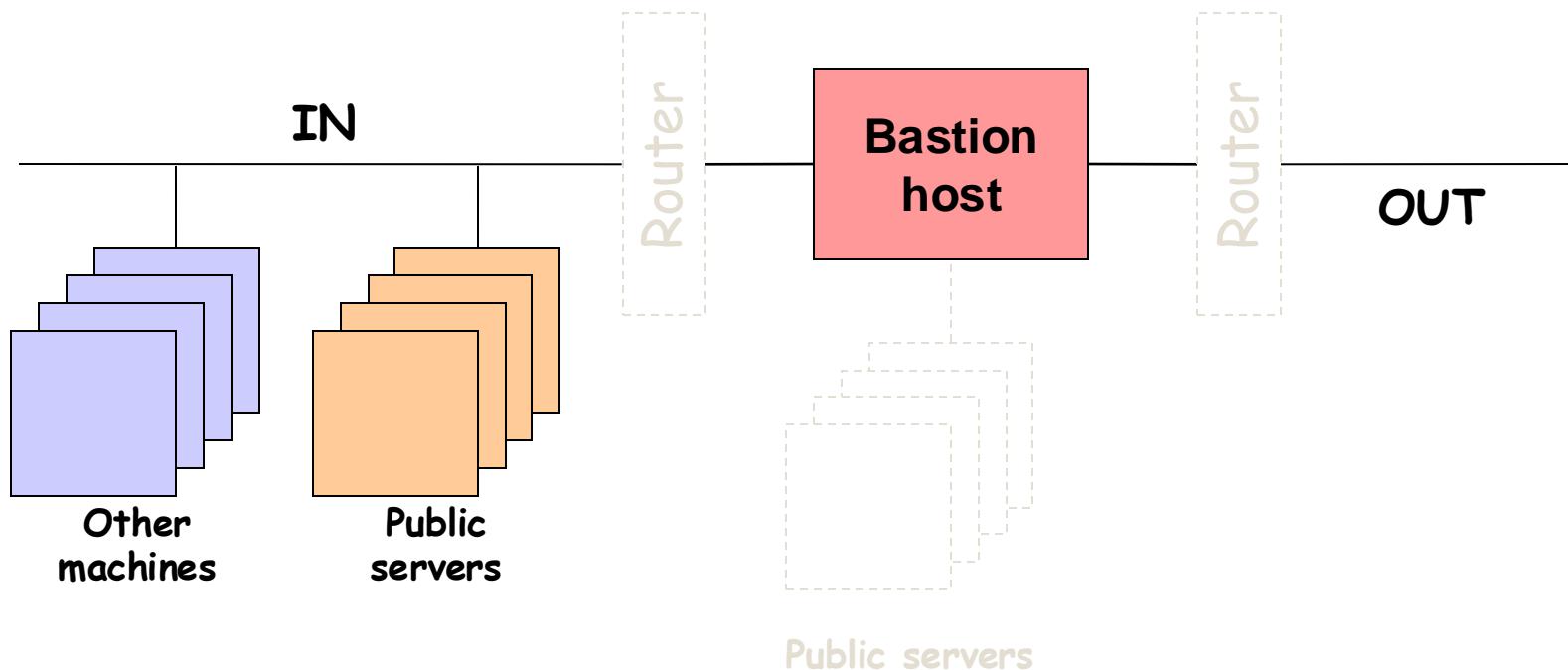


Firewalls: core topologies and variations



Bastion host: dual-homed host firewall

- Firewall / bastion has 2 home networks: IN, OUT



Core topologies: Dual-homed host

- Architecture
 - Single box (bastion host)
- Advantages
 - Simplicity and resource economy
- Disadvantages
 - Compromising the machine deactivates the firewall
 - All the processing load of the firewall in a single machine
 - Public servers are within the protected network

DMZ

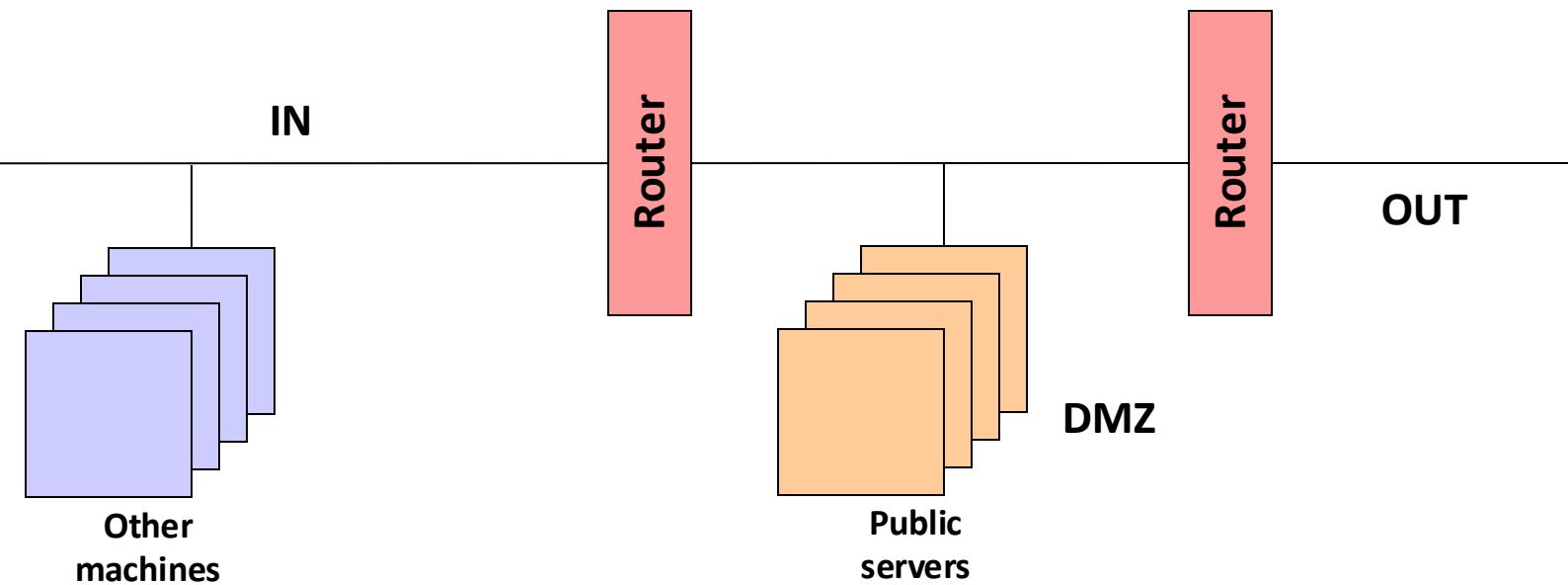
- DeMilitarized Zone



- Network Security use:
 - Not part of the protected perimeter
 - Not part of the outside because it is controlled

Screened subnet

- Two internal subnets: IN and DMZ
 - Routers are packet filters



Core topologies: Screened subnet

- Architecture
 - 2 routers, 1 DMZ
 - The public services are placed in the DMZ
- Advantages
 - Lower risk regarding the **public services**
 - Lower risk regarding a firewall being compromised
- Disadvantages
 - Lower control over the activities going on in the DMZ machines

Multiple levels of isolation

- If net isolation is “easy” – e.g. through virtualization
 - Multiple DMZ
 - One for each type of public server
 - Multiple levels
 - until airgapped

Roadmap

- **Firewalls**
 - **iptables**
- Intrusion Detection Systems

Firewall implementation on Linux



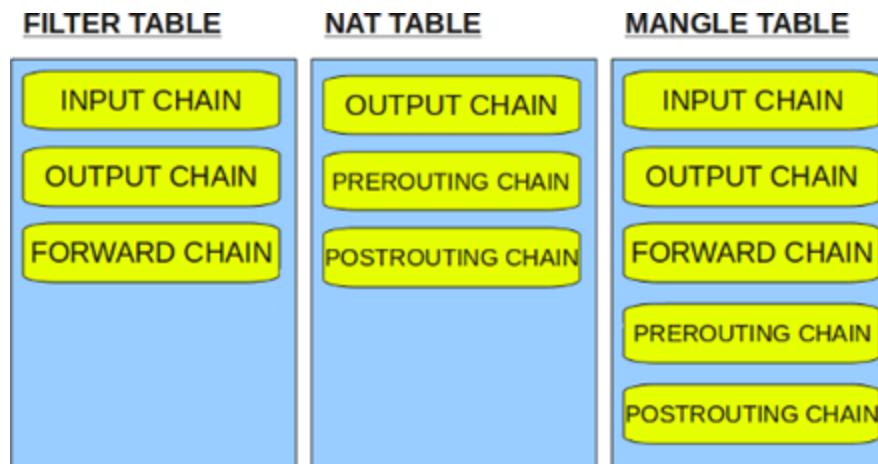
iptables / netfilter

- **netfilter** – a packet filtering framework for Linux
 - module of the Linux kernel that supports packet filtering, NAT,...
- **iptables** – program that configures netfilter's rules
- 3 important concepts:
 - Tables
 - Chains
 - Rules

Tables contain Chains
Chains contain Rules

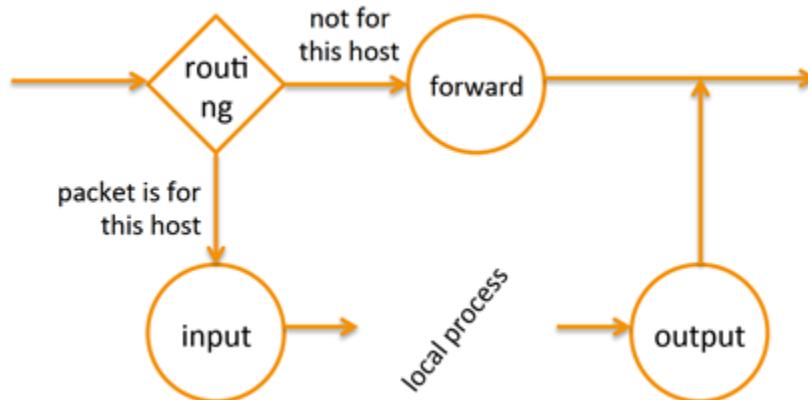
Iptables' tables and chains

- Tables:
 - **Filter** – (the only one) used for packet filtering
 - **NAT** – used to implement NAT rules
 - **Mangle** – used to modify packets arbitrarily

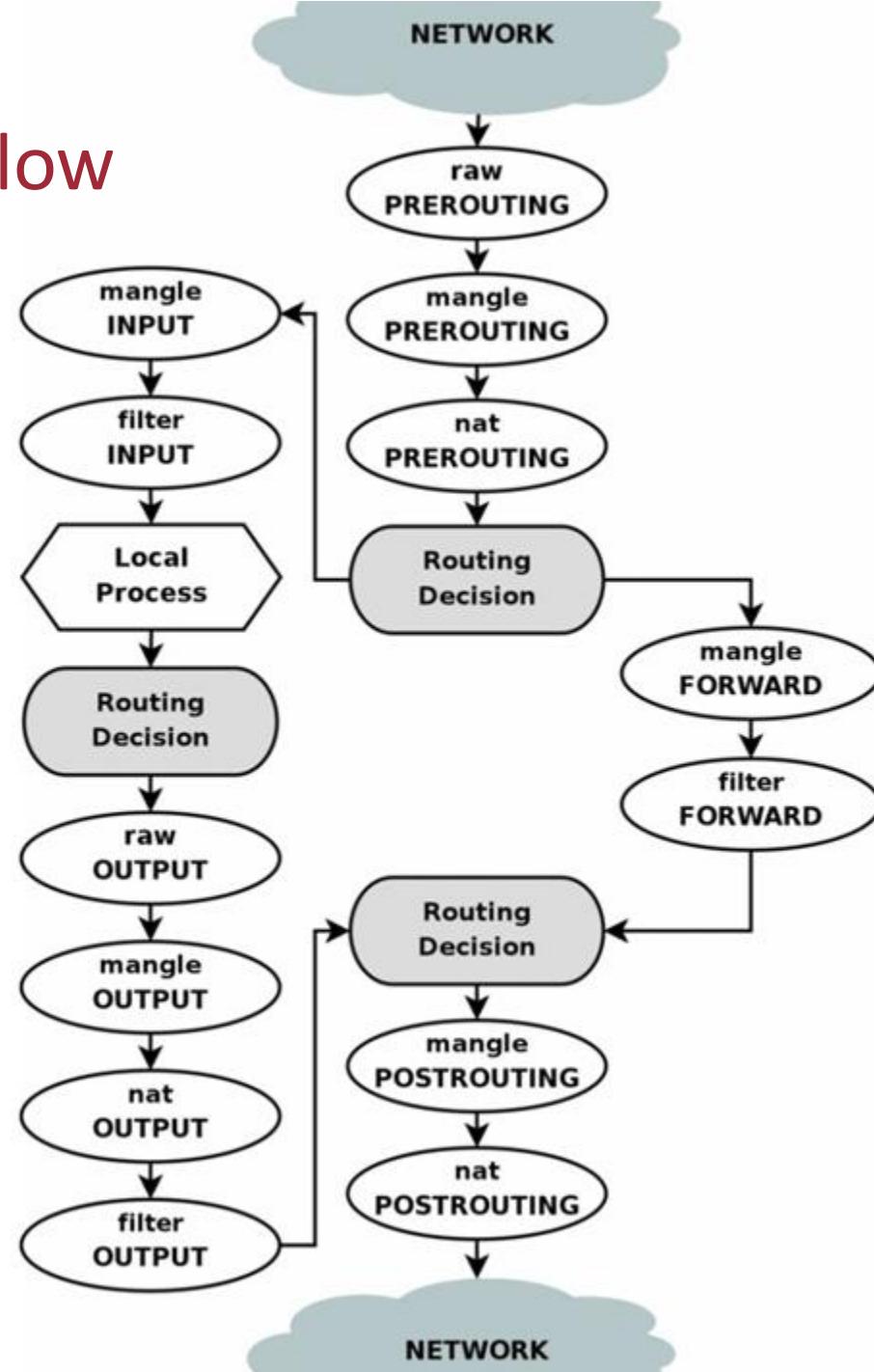


iptables

- **Filter table** has 3 chains (lists of rules):
input, output, forward
 - When a packet reaches a chain, the chain's rules decide the packet's fate:
drop (throw it away) or accept (continue)
 - Input – applied to incoming packet destined to the host
 - Output – applied to packet sent by a process at the host
 - Forward – applied to incoming packet to be forwarded



Overview of packet flow on iptables chains



Roadmap

- **Firewalls**
 - Types of firewall
- Intrusion Detection Systems

Firewall types

- **Packet Filter** – what we call **firewall by default**
- Circuit-Level Gateway
- Application-Level Gateway

Firewall type: Packet Filter

- Reject non-authorized interactions according to the content of the datagrams:
 - Network interface in which the data is sent/received
 - IP addresses (origins and/or destinations)
 - Options in the IP header
 - Transport ports and protocols
 - Options in the headers of transport protocols
 - Direction in which virtual circuits are being created
 - Data sent via transport protocols
 - Type of the UDP or ICMP message
 - Datagram size

Connection state

- TCP connections are explicit
 - Established with "SYN, SYN-ACK, ACK" and ended with a "FIN, FIN-ACK, ACK"
- UDP and ICMP are connectionless
 - But a "connection" can be approximately tracked through addresses and ports of packets source and destination

Stateless vs stateful packet filters

- Packet filters can be stateful or stateless
 - **Stateful:** stores state based on packets observed; packet verification depends on the state
 - States of a connection:
 - NEW – packet for a new connection
 - ESTABLISHED – packet for a previously seen connection
 - **Stateless:** each packet is verified independently
 - No connection state is kept in memory
 - Rely on packet contents and flags

Stateless vs stateful tradeoffs

- Keeping state needs fast memory
 - Expensive
- Performance depends on the number of rules
 - Less rules leads to better performance
 - With simple policies, only a few rules are necessary
 - Stateless is usually faster
 - With complex policies, many more rules are required
 - Stateful is more expressive, and requires less rules, so it can be faster
 - Best choice for performance will depend
 - Compare and evaluate!
- Stateless firewalls typically require more rules
 - Which makes configuration errors more likely...

Circuit-Level Gateway

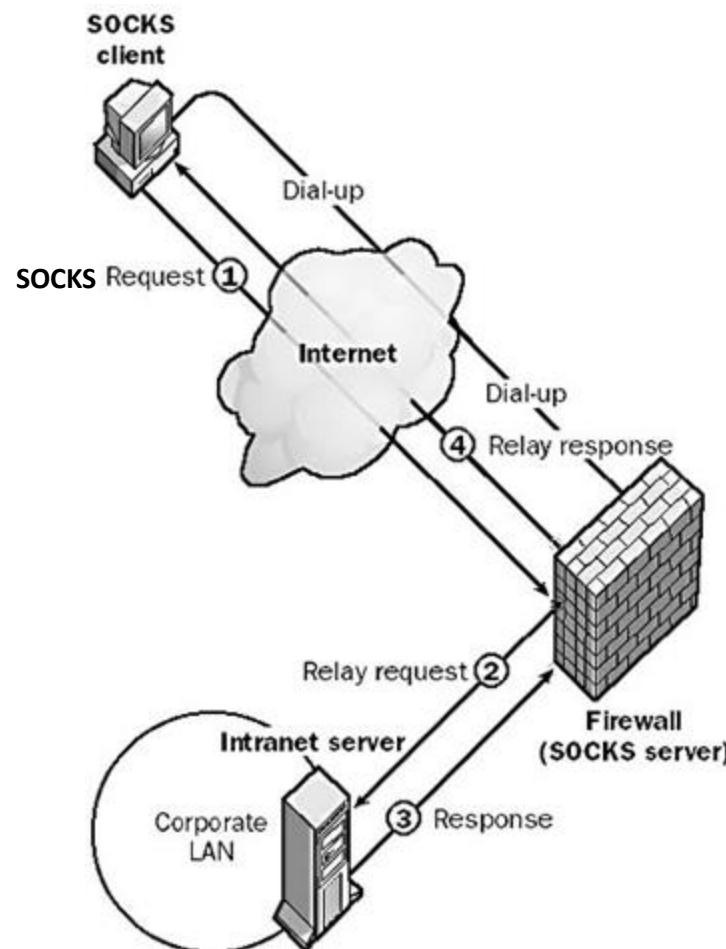
- Directly contacted by the clients at the gateway
 - Works at *transport layer* (typically TCP)
- Non-transparent interposition
 - Applications are aware of the gateway
 - Usually requires changes to the client application
 - Addition of extra redirections protocols:
 - ex. SOCKS

SOCKS

- **SOCKS (SOCKet Secure)**
 - *de facto* standard for circuit gateways
 - RFC 1928 defines version 5
- Components:
 - SOCKS server
 - Often runs on a Unix-based firewall
 - SOCKS client library
 - Runs on host protected by the firewall
 - SOCKS-ified versions of client programs
 - FTP, TELNET, ...

SOCKS operation

- Client opens a TCP connection to the appropriate SOCKS port on the SOCKS server
 - Typically TCP port 1080
- Client negotiates authentication method and authenticates
- Client sends a relay request
 - SOCKS server either establishes the connection or denies it



Firewall type: Application-Level Gateway

- Controls the iterations at the application layer
- Typically there is a specific proxy for each protocol
- Proxy operation characteristics:
 - User-oriented access control
 - Packet content analysis and modification
 - Detailed logging
 - Operations performed, at the application level
 - Proxying
 - Acts as an intermediary, representing other machines/services
 - Caching
 - Keep copies of frequently requested data

Summary of firewall types

- **Packet Filter** – what we call **firewall by default**
 - Reject packets depending on the IP and transport layer headers
 - Stateful vs Stateless
- **Circuit-Level Gateway**
 - Control iterations at the transport layer (typically TCP “circuit”)
 - Otherwise similar to Application-Level Gateways
- **Application-Level Gateway**
 - Control iterations at the application layer
 - Protocol-specific proxy

Comparison of firewall types

- **Packet Filters**
 - Faster but harder to configure
 - Unable to protect against “misbehaving” protocols
 - ex.: ftp, portmapper
 - Current/previous state is not always considered
- **Application-level gateways**
 - Slower but easier to configure
 - Individually for each protocol/application
 - Allow authentication mechanisms
 - Allow more fine-grained control
 - E.g. deny “put” in FTP, deny “delete” in HTTP
 - Less adaptable to new protocols

Roadmap

- **Firewalls**
 - Open issues
- Intrusion Detection Systems

Open issues with firewalls

- Attacks can come in through the IP/ports the firewall does not close
 - e.g., currently many attacks come through port 80 (HTTP) that is almost always open
 - IP spoofing: firewall cannot know if packet really comes from claimed source
 - If multiple applications need special treatment, each one must have its own gateway
 - Tradeoff: degree of communication with outside world, level of security

Open issues with firewalls (cont.)

- The firewall cannot protect against attacks that bypass it
 - Dial-up link, wireless, cellular router,...
- The firewall does not protect against insider threats / authorized users
 - A laptop, tablet, smartphone, or portable storage device may be used and infected outside the corporate network, and then attached and used internally
 - This is the BYOD (Bring Your Own Device) security problem

Roadmap

- Firewalls
- **Intrusion Detection Systems**

IDS definitions

- **Intrusion (or attack)**
 - Any set of actions with the intent of compromising the confidentiality, integrity, or availability (CIA) of a resource
- **Intrusion Detection System (IDS)**
 - Software that has the function to detect, identify, and respond to unauthorized or abnormal activities in the targeted system

Motivation for IDS

- All systems have vulnerabilities
 - Known or unknown
 - Can be used to carry out attacks
- Attacks can be detected by:
 - Becoming aware of / seeking unusual or suspicious actions
 - Searching for unusual or suspicious alterations in the information stored in the system
- What do we want to detect?
 - Intrusion preliminary phases (probes)
 - Attack accesses from the outside
 - Attacks from the inside

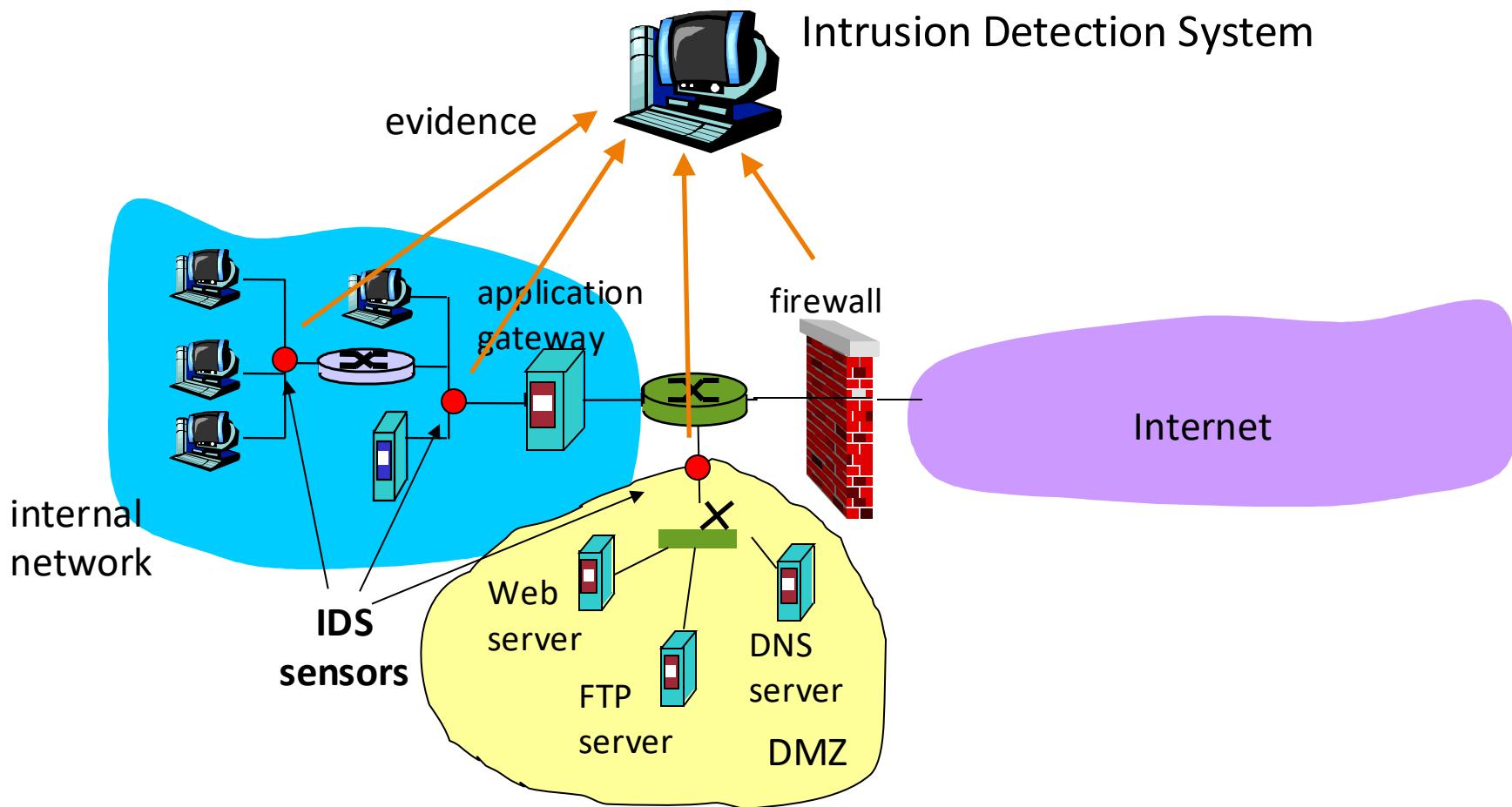
IDS complement firewalls

- *Firewalls (packet filters)*
 - Operate on TCP/UDP/IP/ICMP headers only +
 - Do not do correlation among packets or sessions
- *Intrusion Detection Systems (IDSS)*
 - **Deep packet inspection (DPI)**
 - Look at packet contents
 - e.g., check character strings in packet against database of known virus, attack strings
 - **Examine correlation among multiple packets**
 - Can detect attacks like port scanning, network mapping, DDoS
 - Not only network-based but also host-based

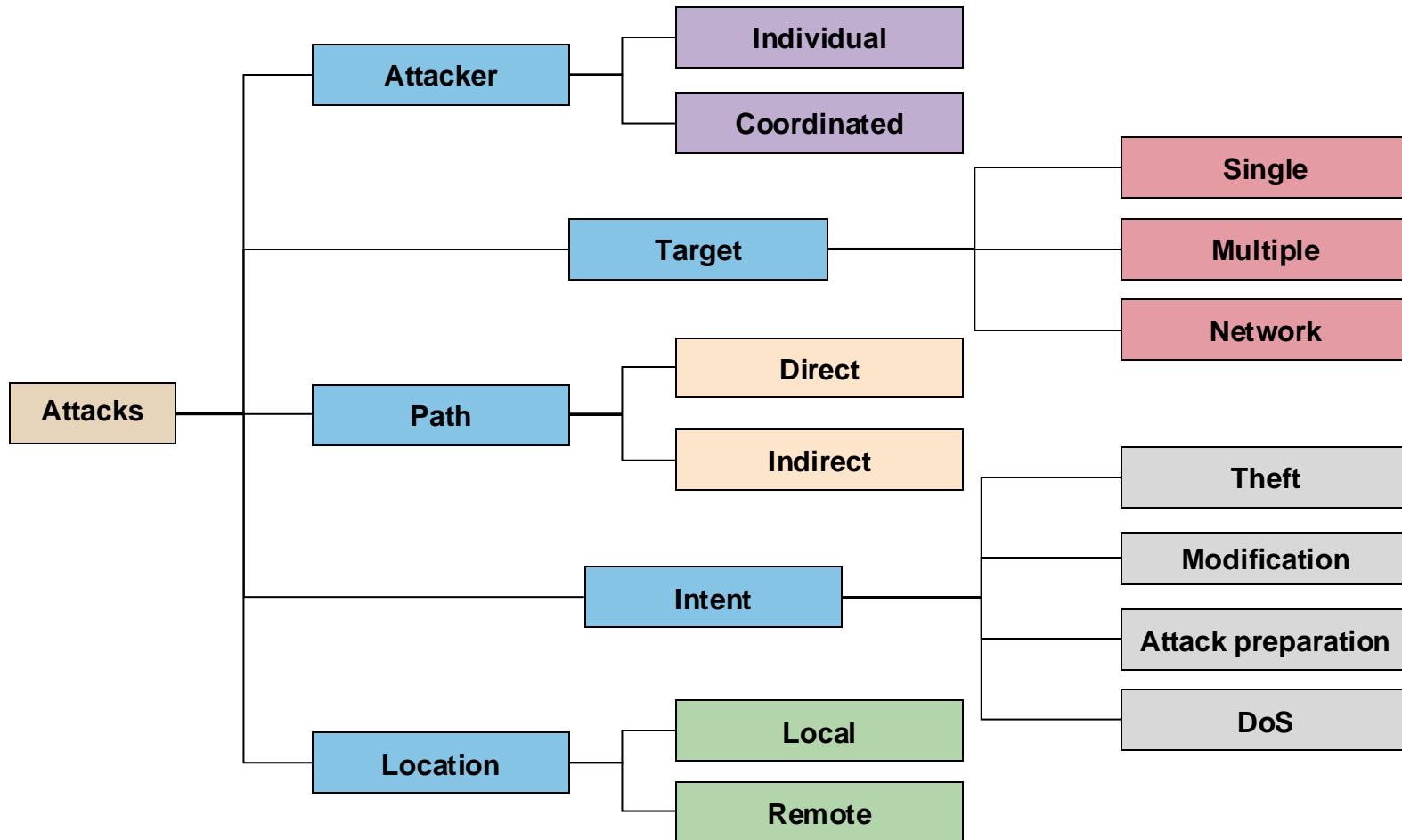
Intrusion detection – Why ?

- When prevention does not work
 - Detecting intrusion access (external ones)
 - Detecting intrusion preambles (probes)
 - Detecting abusive behaviors (internal ones)
 - Particularly effective in these cases
 - Given the more relaxed preventive measures for the internal users
 - e.g.: malicious staff, email, 802.11b, laptops and mobile devices,
 - ...
 - Gather intrusion information for later use
 - To document existing threats for the organization
 - Verify the effectiveness of the prevention mechanisms

Intrusion detection – How ?



Attack classification



Attack detection

- IDSs generate alarms
- There are detection errors
 - False positive (false alarm)
 - False negative (missing alarm)
 - Tradeoff: configuring the IDS for less false negatives usually leads to more false positives
 - False positives also very bad in practice
 - Can be used for subversion
 - E.g. change IDS behavior by forcing too many alarms
- Correlation
 - Detecting patterns in sets of atomic events
 - Requires temporary storage of state
- Data merging
 - Uniformly analyzing events from heterogeneous sources

True/False Positive/Negative

- True/false
 - First word indicates the **reality**
- Positive/negative
 - Second word indicates the **prediction**

(false positive)



(false negative)



Honeypots

- Vulnerable system created for **deception**
- Focus attacker's attention on an apparently weaker and valuable system:
 - Deflect the attack from the real system
 - Detect and learn about new attacks
 - Gather forensic information
- Example implementation:
 - <https://opencanary.org>
- Problem?
 - Can be used as an attack origin



IDS classification

Detection method	Misuse detection Anomaly detection
Data source	Network-based Host-based
Detection delay	Real-time <i>A posteriori</i>
Reaction	Passive Active
Analysis	Individual Cooperative

IDS classification

Detection method	Misuse detection
	Anomaly detection
Data source	Network-based
	Host-based
Detection delay	Real-time
	<i>A posteriori</i>
Reaction	Passive
	Active
Analysis	Individual
	Cooperative

Misuse Detection / Knowledge-based Detection

- System activity analysis in search of known attack patterns (**attack signatures**)
 - A match raises an alarm
- Advantages:
 - Efficient detection
 - Reduced amount of false positives
- Disadvantages:
 - Only detects known attacks
 - Useless with unknown attacks
 - May generate a large amount of false negatives

Anomaly Detection / Behavior-based Detection

- Matches observed behavior with a model of normal behavior
 - Using statistical heuristics (thresholds) or Machine Learning
 - No match raises an alarm
 - “*this is not normal...*”
- Advantages:
 - Able to detect new attacks
 - Can be used to collect data to define new attack signatures
- Disadvantages:
 - Needs a large amount of training data without attacks
 - If training set contains attacks, they are considered “normal”
 - Difficult to define adequate threshold values
 - Large amount of false positives

IDS classification

Detection method	Misuse detection Anomaly detection
Data source	Network-based Host-based
Detection delay	Real-time <i>A posteriori</i>
Reaction	Passive Active
Analysis	Individual Cooperative

NIDS: Network-Based IDS

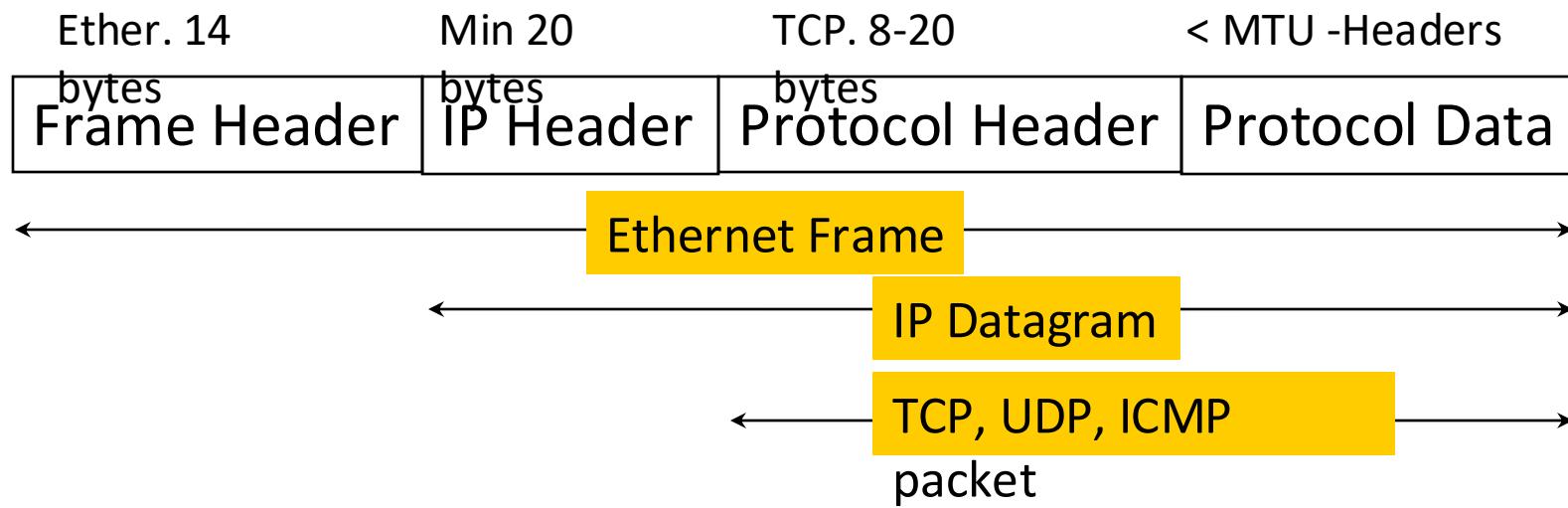
- Capture and do traffic analysis on network data (e.g., packets)
- Advantages:
 - Small amount of sensors can monitor a large network
 - Little to no impact in the network performance
 - Can be invisible to attackers
- Disadvantages:
 - Hard to process large amounts of data flowing through the network
 - Difficult to install in networks that are not shared
 - Cannot analyze ciphered data
 - Cannot assess with certainty if an attack was successful
 - Difficult to be aware of the connection state

Attacks targeted by NIDS

- Unauthorized access to the internal network
 - Base/bridge to other attacks
- Theft of information in the network
- Brute-force attacks, e.g., to get passwords
- Abuse of bandwidth resources
- Denial of Services (DoS)
 - Improperly formatted packets
 - Abnormally high data/packet flow
 - Distributed DoS

NIDS – Attack signature

- Packet content
 - More powerful
 - Ineffective in ciphered channels
- Packet header
 - Less powerful
 - Not affected by ciphered contents



NIDS example: SNORT

- Misuse detection (has signatures)
- SNORT can be used as a:
 - Packet sniffer - live analysis
 - Packet logger - *a posteriori* analysis
- Advantages
 - Open-source
 - Stable
 - Flexible - allows custom rules
 - There is an active community
 - Keeping attack signatures up-to-date
 - Also a company – SourceFire
 - provides rules immediately for subscribers, and free after 30 days



<https://www.snort.org/>

HIDS: Host-Based IDS

- Capture and do analysis on host data
 - Can look at processes running in the computer, their CPU and memory usage, I/O behavior, etc.; look at logs, registry, etc.
- Advantages:
 - Able to observe/detect attacks that cannot be seen by a NIDS
 - Able to function in environments with ciphered data
 - Not affected by network isolation (virtual channels)
- Disadvantages:
 - Hard to manage
 - Can be attacked and deactivated
 - Unable to detect scans
 - Degrades the performance of the systems

Attacks targeted by HIDS

- Abuse of privileges
 - Employees, administrators
 - Sub-contracted (external) staff
- User account usurpation:
 - Old employees
 - Created by misbehaving administrators
- Inadvertently assigned privileges
- Access and modification of critical information
 - Browsing critical information
 - Modification of configuration files
 - Modification of Web site
- Information leakage

HIDS example: OSSEC

- OSSEC has:
 - Correlation and analysis engine
 - Log analysis
 - File integrity checking
 - Centralized policy enforcement
 - Rootkit detection
 - Alerting
 - Active response – e.g. black list IP addresses
 - Optional web-based graphical monitoring interface
- Advantages
 - Open-source
 - Runs on most operating systems
 - Linux, OpenBSD, FreeBSD, MacOS, Solaris and Windows

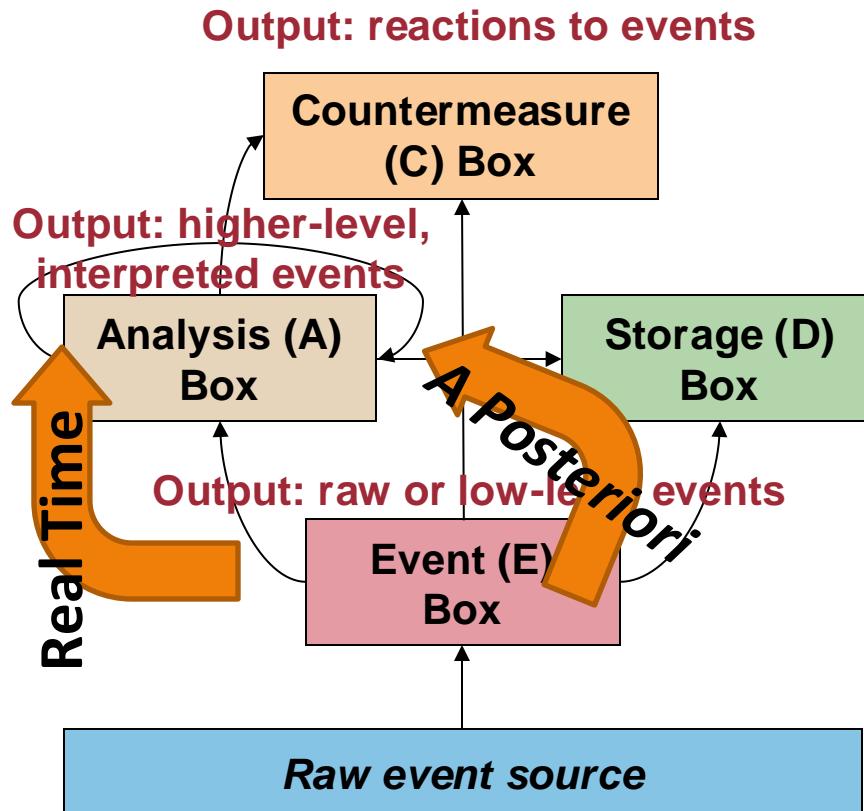


<https://ossec.github.io/>

IDS classification

Detection method	Misuse detection Anomaly detection
Data source	Network-based Host-based
Detection delay	Real-time <i>A posteriori</i>
Reaction	Passive Active
Analysis	Individual Cooperative

Functional architecture



- Event Box
 - Sensors that capture events
- Analysis Box
 - Performs analysis of low-level events and generates higher-level events
- Data Box
 - Information storage module
- Countermeasure Box
 - Reaction modules: execution of predetermined actions / methods as a response to events

Based on the **Common Intrusion Detection Framework (CIDF)**,
an old IETF proposal for IDS interoperability

IDS classification

Detection method	Misuse detection Anomaly detection
Data source	Network-based Host-based
Detection delay	Real-time <i>A posteriori</i>
Reaction	Passive Active
Analysis	Individual Cooperative

How to react to attacks?

- Passive
 - Only detect and report the detection results:
 - Alarms and notifications
 - Logging and report creation
- Active
 - Respond to the attacks:
 - Close connections: TCP RST
 - Perform system/operational modifications
 - Reconfiguration of routers/firewalls, etc.
 - Counterstrike
 - Typically, illegal
 - Be careful not to start a cyberwar...
- Active IDSs also known as:
intrusion prevention systems (IPSs) or
intrusion detection and prevention systems (IDPSs)

Attack reaction

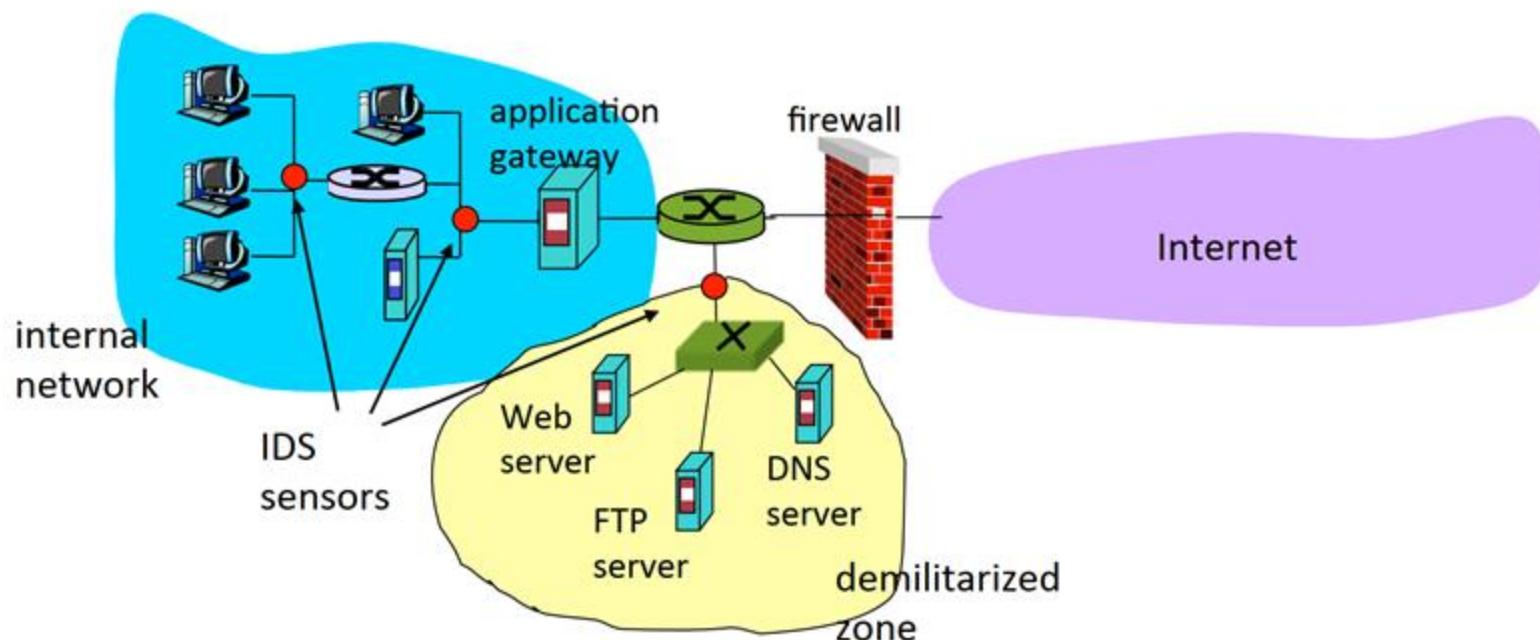
- Block malicious traffic
 - Closer to firewalls, but with deep packet inspection and correlation
 - Blocking mechanisms
 - Ending TCP connections
 - Dropping packets (like packet filters)
 - Throttling bandwidth usage
 - Sanitizing traffic (e.g., dropping malicious mail attachment)
 - Reconfiguring other network devices (firewalls, routers,...)
 - Running third-party program or script
- False positives are even more dangerous in this case
 - The reaction might be intentionally caused by the attacker
 - e.g., mislead you to block a range of IPs

IDS classification

Detection method	Misuse detection Anomaly detection
Data source	Network-based Host-based
Detection delay	Real-time <i>A posteriori</i>
Reaction	Passive Active
Analysis	Individual Cooperative

Cooperative IDS

- Not necessarily centralized as a firewall
 - Multiple IDSs: different types of checking at different locations



IDS classification

Detection method	Misuse detection Anomaly detection
Data source	Network-based Host-based
Detection delay	Real-time <i>A posteriori</i>
Reaction	Passive Active
Analysis	Individual Cooperative

Summary

- Firewalls
- Intrusion Detection Systems

Conclusion

- Firewalls and IDS are important security mechanisms that can enforce security policies
 - Firewalls control accesses
 - IDS monitor activities
 - Both can and should be combined
 - Several configurations and solutions exist

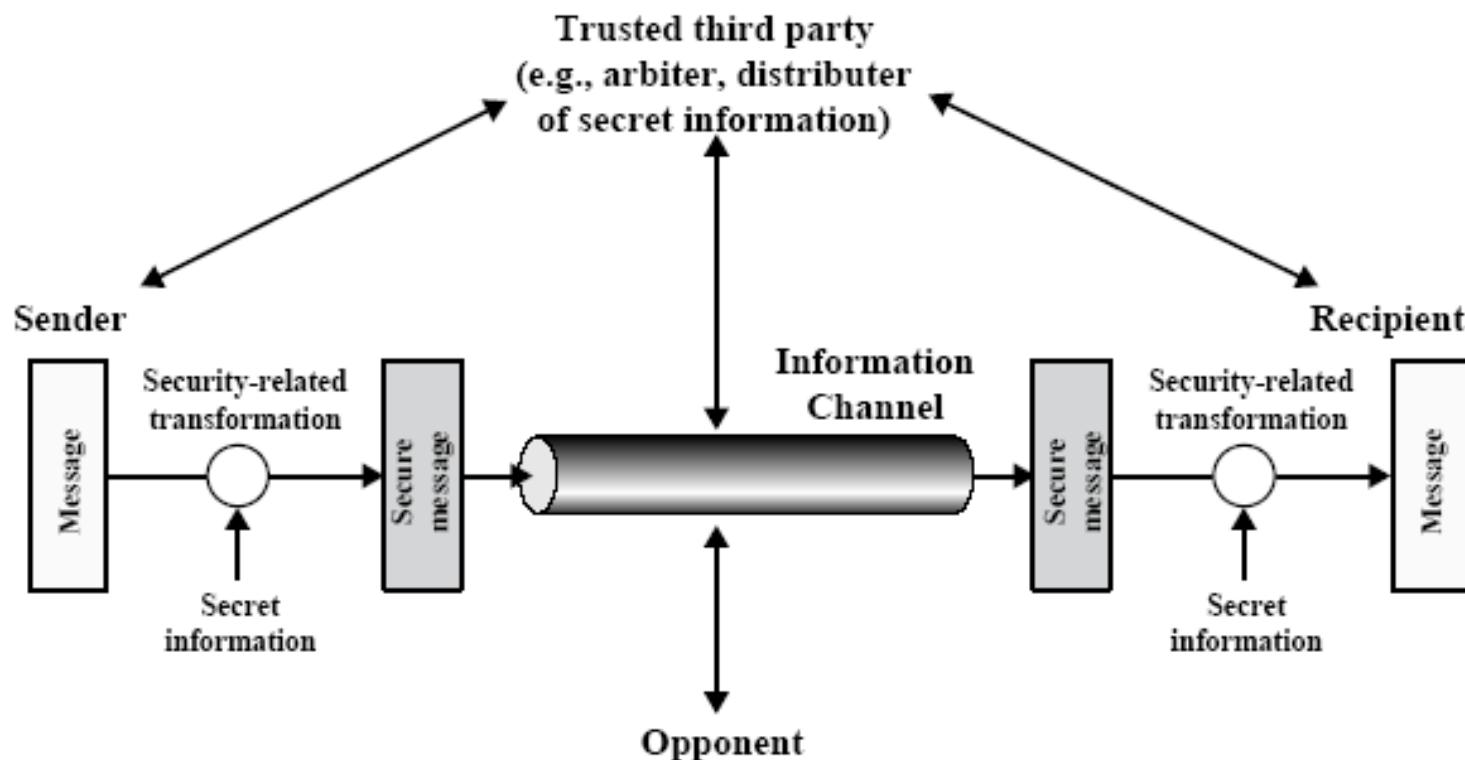
Secure channel: SSH and TLS

Segurança Informática em Redes e Sistemas
2024/25

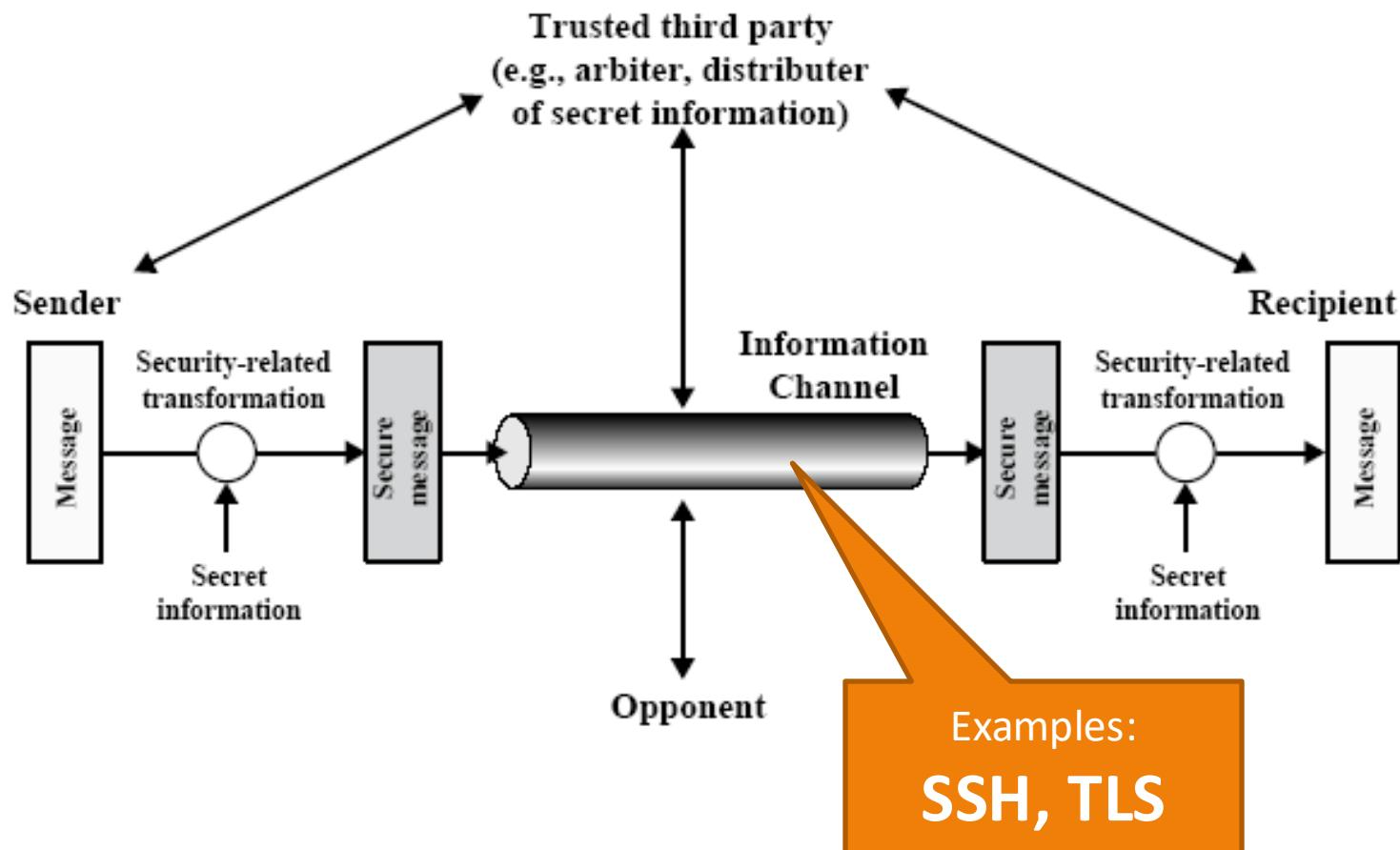
David R. Matos, Ricardo Chaves

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Secure communication channel



Secure communication channel



Secure communication: *Transport layer and above*

	Layers	Responsibility	Approach	Solutions
OSI Layers	Transaction	Local data manipulation applications		PGP, PEM, S/MIME
	Application	Applications for remote data exchange	End-to-end security	HTTPS, IMAPS SSH
	Presentation			
	Session			
	Transport	Operating Systems		TLS
	Network			IPsec
	Link	Devices	Link security	IEEE 802.11*
	Physical			

Attacker model

- Attacker who has full control over the network
 - but cannot break cryptographic primitives
- Attacker can intercept, send, and modify messages
 - but not decrypt them without the key
- This model is called the **Dolev-Yao** model
 - This is the default attacker model presumed in distributed systems
 - Dolev, D., & Yao, A. C. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 198–208. <https://doi.org/10.1109/TIT.1983.1056650>

Approach: Information Isolation through Cryptography

- Cryptography can render information unintelligible to unauthorized parties
 - Encrypted data is effectively isolated
 - It appears as noise to anyone without the key to decrypt it
- Encrypted information can be transmitted across communication networks or stored in computer systems and stay isolated

Roadmap

- **SSH – Secure Shell**
- **TLS – Transport Layer Security**

SSH

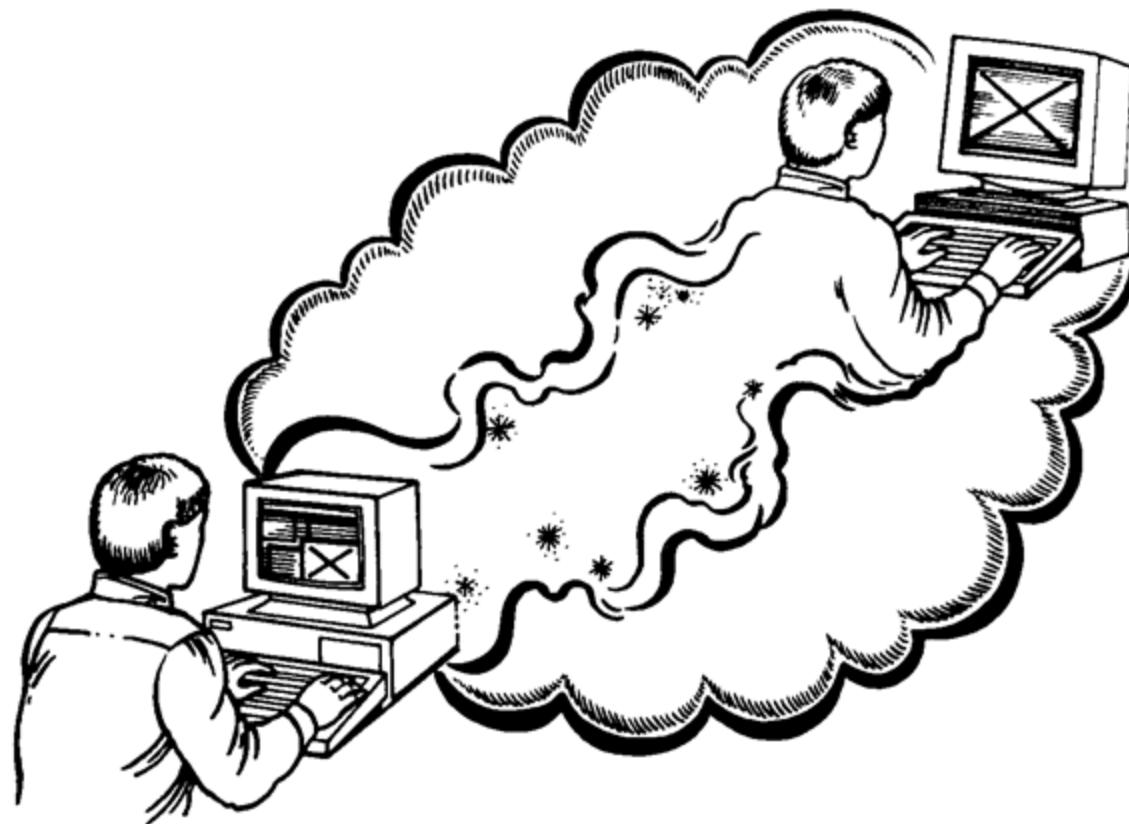


Figure 7.1 Remote login is a lot like astral projection.

SSH goals

- Secure communication application and protocol over TCP
 - Allows **secure remote sessions (~telnet)** and **file transfer (~ftp)**
 - Allows **tunneling** of TCP/IP traffic
- Security mechanisms
 - **Confidentiality** and **integrity** of the communication
 - **Distribution of keys**
 - Communicating parties' **authentication**
 - The server (or, usually, the server machine)
 - The user

SSH Keys

- SSH use **public-key cryptography**
- **Pair of keys**
 - One is **private**, personal, non-transmissible
 - One is **public**, can/should be widely known
- Allow for
 - **Confidentiality** with the exchange of secret keys
 - **Authentication and Integrity** with digital signatures

SSH protocols (1/2)

- Transport Layer Protocol
 - Server authentication
 - Signature of the exchanged DH ephemeral values
 - Server public key can be transferred at this time, if not already held by the client
 - → TOFU (Trust on First Use) model
 - Vulnerable to man-in-the-middle
 - No certificates or CA
 - Distribution of keys
 - Diffie-Hellman key exchange
 - The session keys are computed with ephemeral DH values
 - Creation and analysis of secure messages
 - Compression, encryption, integrity control

SSH protocols (2/2)

- User authentication towards the remote machine:
 - Password or
 - Signature with private key of client
 - Server knows the public key of client
- Connection Protocol
 - Information/data flow **multiplexing** over a secure session

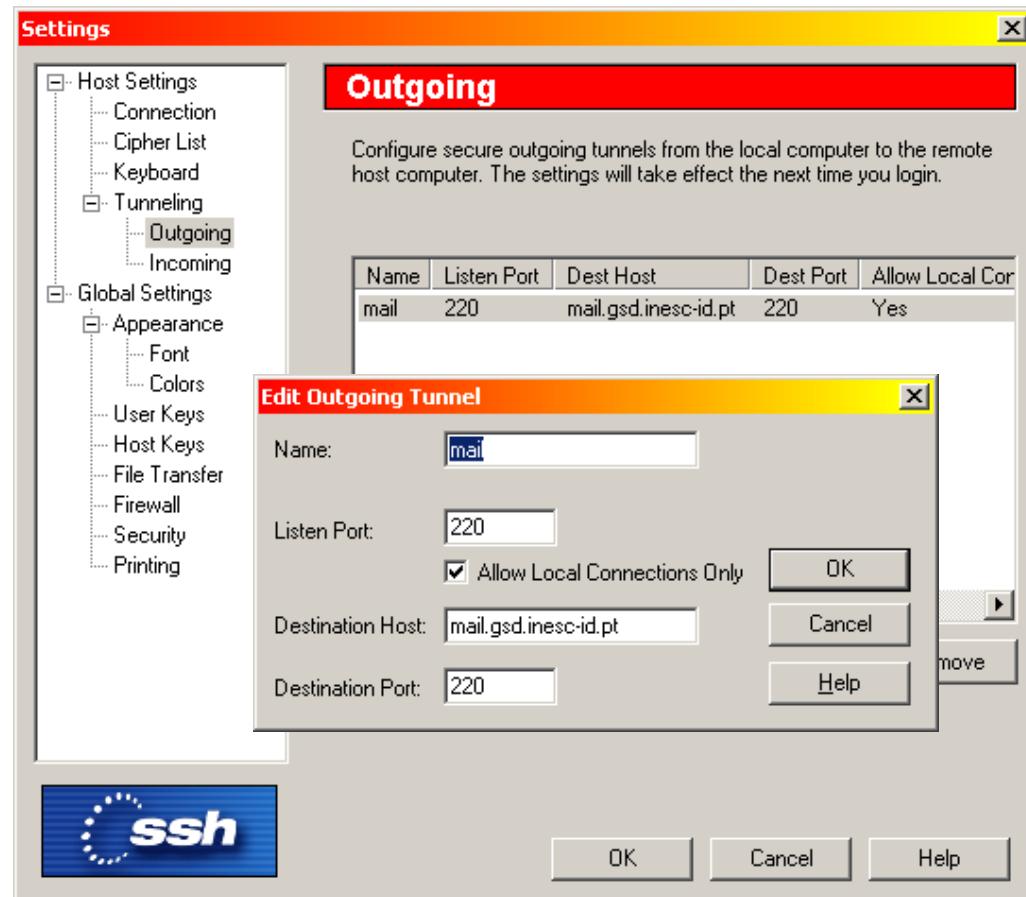
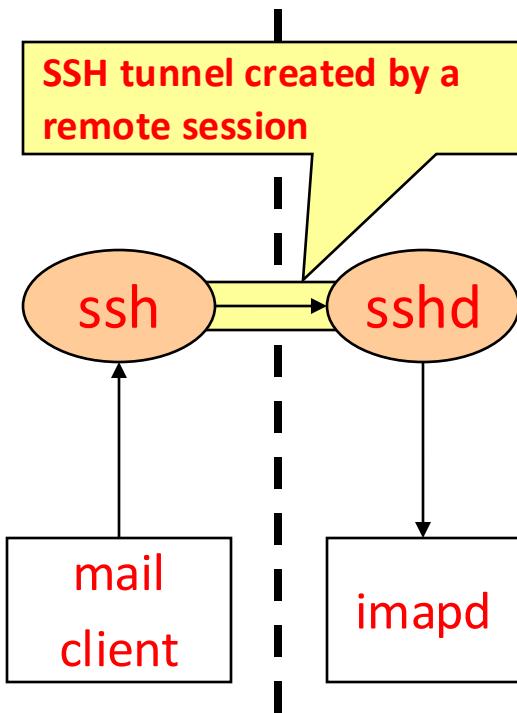
Roadmap

- SSH – Secure Shell
 - **SSH Tunnels**
- TLS – Transport Layer Security

Tunneling with SSH

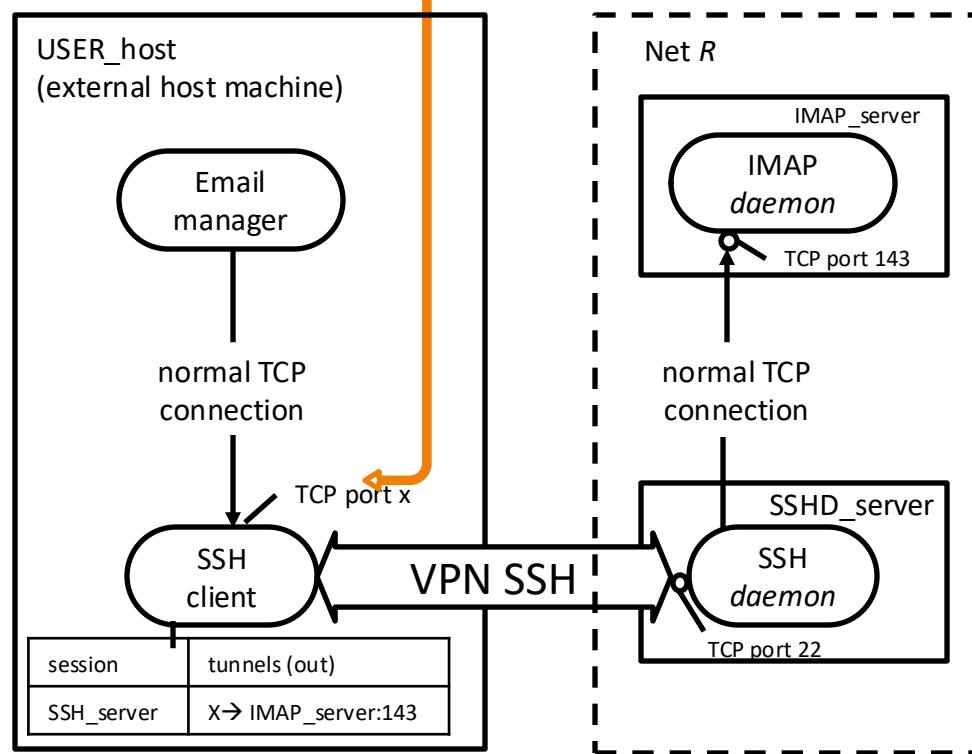
- In the client machine, a **mapping** is created between a **local TCP port** and a **port in the remote machine**
 - e.g., *localhost:IMAP* → *mail.myorg.pt:IMAP*
- **SSH session / tunnel** is created to *mail.myorg.pt*
- Client application is configured to use the **local port**
 - When using the port, it will securely interact via SSH with the remote server *mail.myorg.pt*
 - Client SSH and server SSHd operate as a secure relay mechanism

Configuration example of SSH tunnel for IMAP (port 220)



Output tunnel

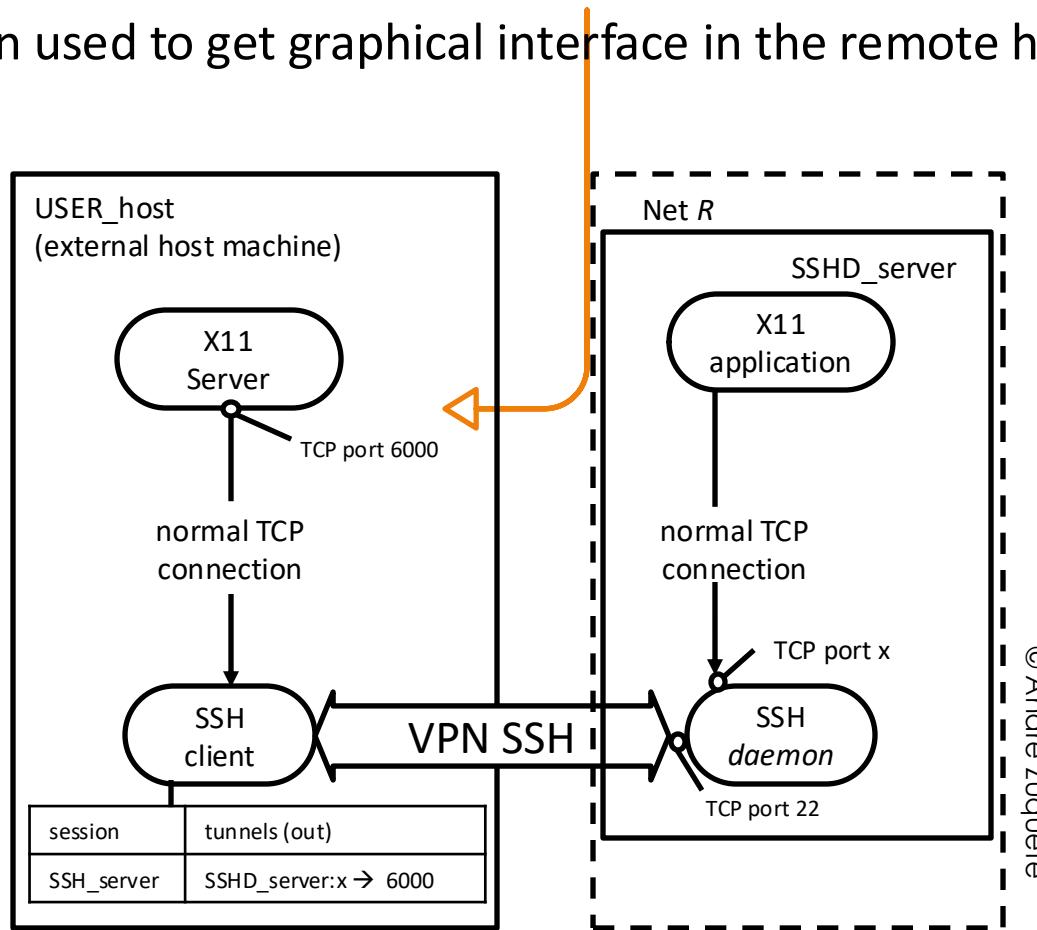
- TCP connections started at the client, so **outbound**
 - **SSH client** opens **port X** and waits for connections from client application
 - Similar situation to remote login



© André Zúquete

Input tunnel

- TCP connection started at the server, so **inbound**
 - **SSH client** connects to port in **local server**
 - Often used to get graphical interface in the remote host



Tunneling

- Transparent for apps
- But complex/difficult for admins
 - Port management
- Low flexibility for developers and users

Roadmap

- SSH – Secure Shell
- **TLS – Transport Layer Security**

http



https



TLS is the S in HTTPS

- **HTTPS = HTTP over SSL (now TLS)**
 - SSL - Secure Sockets Layer
 - TLS – Transport Layer Security
- SSL was originally designed to be used with HTTP
 - Netscape Navigator (1994)
 - Forefather of Mozilla Firefox
 - Evolved from SSL in 1994 to TLS 1.3 by 2018
 - SSL 1.0 1994
 - SSL 2.0 1995
 - SSL 3.0 1996
 - TLS 1.0 1999
 - TLS 1.1 2006
 - TLS 1.2 2008
 - TLS 1.3 2018

TLS overview

- TLS provides secure channels over TCP/IP, authentication, integrity, confidentiality, and key distribution
- Many implementations:
 - SSLref, OpenSSL, GnuTLS, SSLeay, Mbed TLS, JSSE (Java Secure Socket Extension)
- TLS is not bound to HTTP
 - Can be used with: SMTP, IMAP, POP3
- TLS is very significant, as it ensures **data in transit** security across many critical Internet services

TLS goals

- Secure communication channels over TCP/IP
 - Current version: **TLS 1.3** – august 2018
 - Standard based on the deprecated **SSL (Secure Sockets Layer)**
 - Sometimes called **SSL** for that reason
 - Manages secure sessions over **TCP/IP** per application
 - Initially designed for the HTTP protocol (HTTPS)
 - Currently used by other protocols, e.g., SMTP, IMAP, POP3
- Security mechanisms
 - **Authentication** of the communicating parties
 - **Confidentiality** and **integrity** of the communication
 - **Key distribution**

TLS utilization

- TLS is just a protocol; not a standard API
- Common APIs:
 - Reference API for SSL: SSLref (Netscape)
 - Public implementations: OpenSSL, GnuTLS, SSLeay, Mbed TLS, Java Secure Socket Extension (JSSE)
- Remote interfaces:
 - Conventional: protocol/port
 - e.g., TCP/443 for HTTPS
 - STARTTLS: allows upgrading text connection to TLS
 - Defined for SMTP, IMAP, POP, SMTP, FTP, IRC,...

TLS operation management

- Client-Server model as in TCP
- The applications (e.g., web, mail) define the strategy
- **Authentication**
 - If it is needed and how it is performed
- **Cryptographic algorithms**
 - The client presents the **cipher suites** it supports
 - The server selects one
- **Session key management**
 - Lifetime of the **master secret** = lifetime of the **TLS session**
 - Used whenever the client and the server decide to communicate
 - Maximum of 24 hours recommended
 - Lifetime of the **session keys**: at most lifetime of the TCP connection

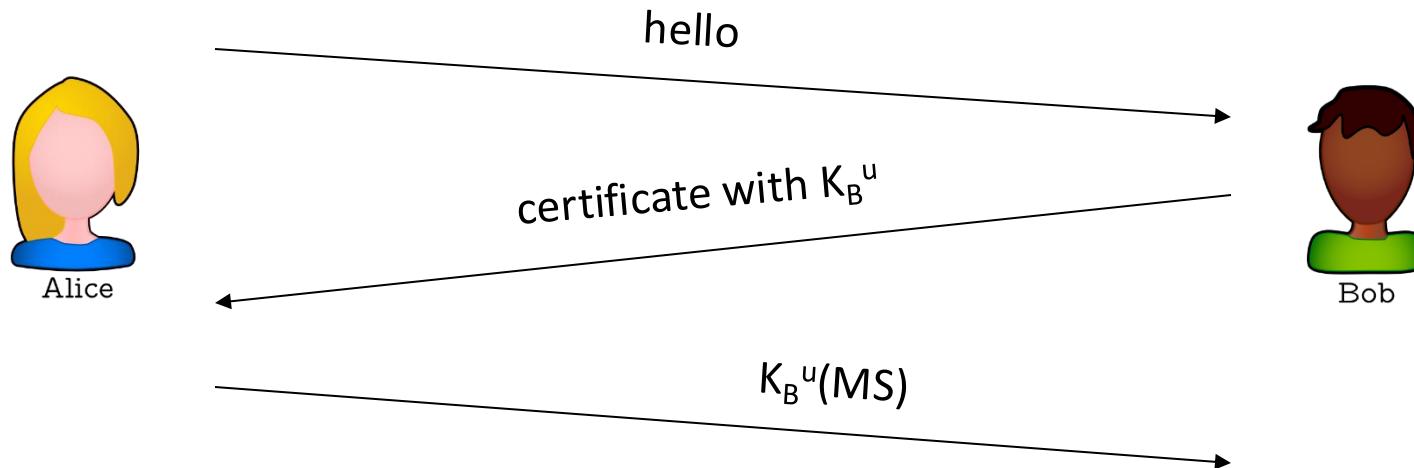
Roadmap

- SSH – Secure Shell
- TLS – Transport Layer Security
 - Toy TLS

Toy TLS: simplified secure channel

- **Handshake**
 - Alice and Bob use their certificates, private keys to authenticate each other and exchange a shared secret
- **Key derivation**
 - Alice and Bob use shared secret to derive set of keys
- **Data transfer**
 - Data to be transferred is broken up into series of records
- **Connection closure**
 - Special messages to securely close connection

Toy TLS: a simple handshake



- $K_B^u = \text{Bob's public key}$
- $MS = \text{master secret}$

Toy TLS: key derivation

- It is bad to use same the key for more than 1 cryptographic operation
 - So, use different keys for MACs and encryption
- Therefore **4 keys**:
 - K_c = encryption key for data sent from client to server
 - M_c = MAC key for data sent from client to server
 - K_s = encryption key for data sent from server to client
 - M_s = MAC key for data sent from server to client
- Keys derived using **key derivation function (KDF)**
 - Takes master secret (MS) and additional random data and creates the keys

Toy TLS: data records

- Why not encrypt data in stream as we write it to TCP?
 - Where would we put the MAC? If at end, no message integrity until the end of the connection!
- Instead, break stream in series of **records** (\simeq messages)
 - Each record carries a MAC
 - Receiver can act on each record as it arrives
- Issue: records have variable length and receiver needs to distinguish MAC from data
 - Solution:

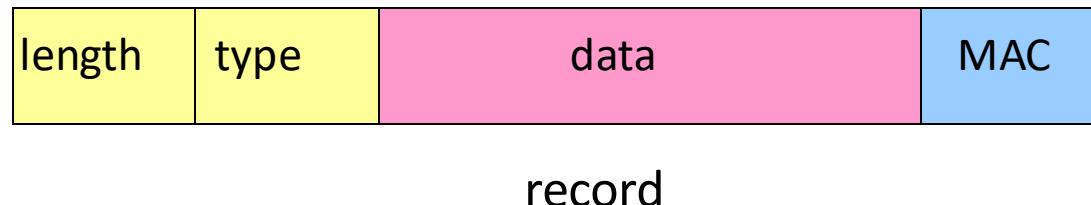


Toy TLS: sequence numbers

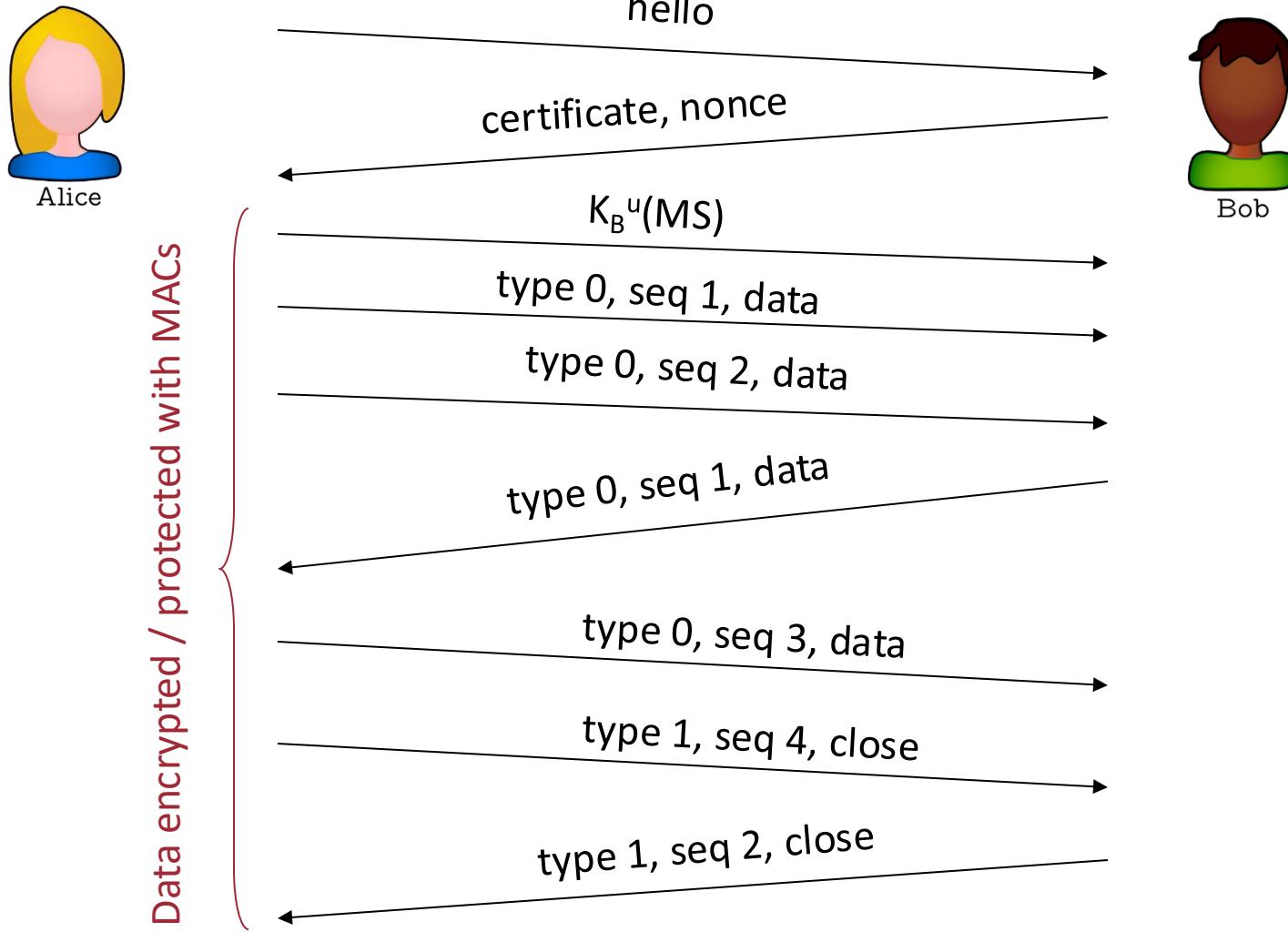
- Attacker can replay or re-order records
- Solution: put sequence number (**seq**) into MAC:
 - **seq** starts at 0; incremented for every record
 - $\text{MAC} = \text{MAC}(M_x, \text{seq} \parallel \text{data})$ key $M_x = M_c$ or M_s
 - Important: there is no sequence number field in the record, so **seq** is implicit, just like the **key** M_x
- Attacker could still replay all of the records
 - Use a **nonce** to prevent this attack

Toy TLS: control information

- Truncation attack
 - Attacker forges TCP connection close segment
 - One or both sides thinks there is less data than there actually is
- Solution: **record types**, with a type for closure
 - type 0 for data; type 1 for closure
- $\text{MAC} = \text{MAC}(M_x, \text{seq} \mid \mid \text{type} \mid \mid \text{data})$



Toy TLS: summary



Toy TLS is not complete

- How long are the fields?
- Which encryption protocols to use?
- Negotiation?
 - Allow client and server to support different encryption algorithms
 - Allow client and server to choose together specific algorithm before data transfer

Roadmap

- SSH – Secure Shell
- **TLS – Transport Layer Security**

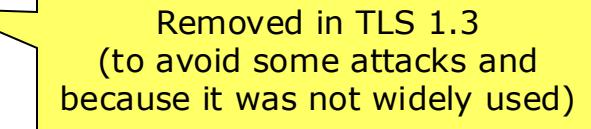
TLS protocols

- **Record Protocol**

- Creation and verification of secure messages
 - Cipher, integrity control, ~~compression~~

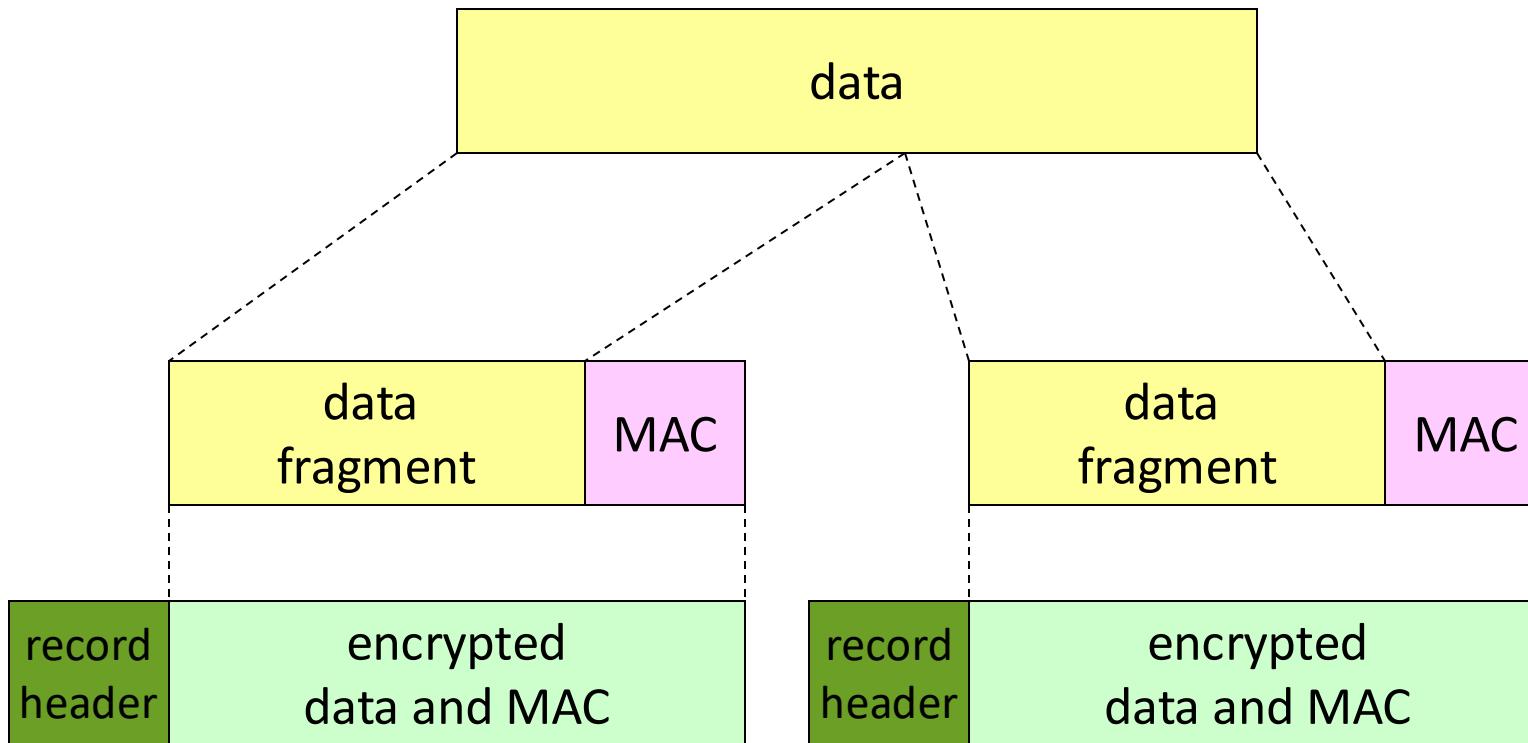
- **Handshake Protocol**

- Exchange of identity and supported cipher suites
- Authentication of the communicating parties
 - Client challenges the server, that proves its identity with certificate(s) *[Optional, but mandatory if the next exists]*
 - Server challenges the client, that proves its identity with certificates *[Optional]*
- Key distribution



Removed in TLS 1.3
(to avoid some attacks and
because it was not widely used)

TLS record protocol

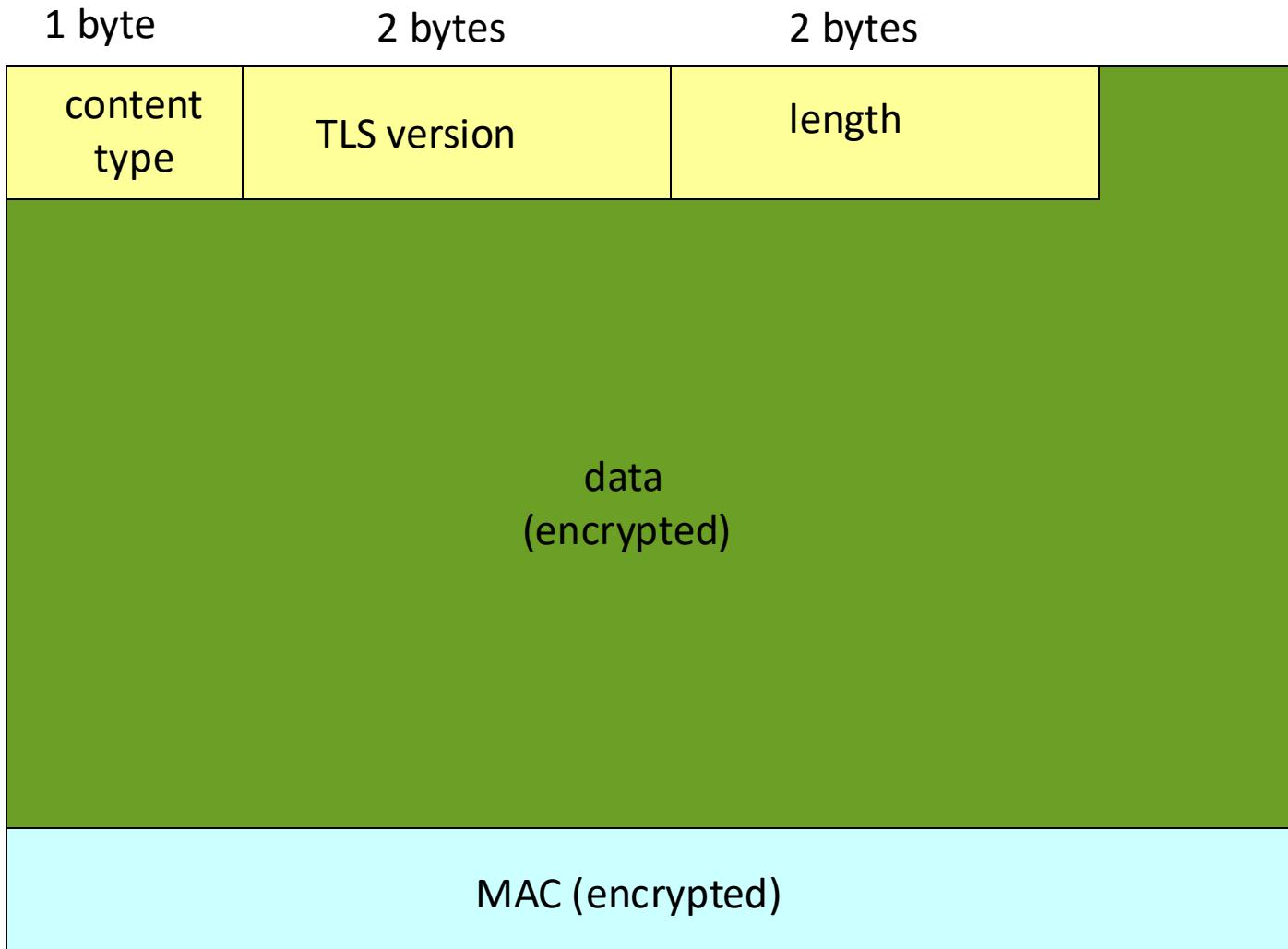


Record header: content type; TLS version; length

MAC: is function of the sequence number and the MAC key M_x

Fragment: each data fragment has up to 2^{14} bytes (~ 16 Kbytes)

TLS record format



TLS handshake (1)

Purpose

1. Server authentication
2. Negotiation: agree on crypto algorithms
3. Establish keys
4. Client authentication (optional)

TLS handshake (2)

1. **Client** sends list of algorithms it supports, along with client nonce
2. **Server** chooses algorithms from list; sends back: choice + certificate + server nonce
3. **Client** verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. **Client and server** independently compute encryption and MAC keys from pre_master_secret and nonces
5. **Client** sends a MAC of all the handshake messages
6. **Server** sends a MAC of all the handshake messages

TLS Cipher Suites

- **Cipher suite**
 - Public-key algorithm
 - Symmetric encryption algo.
 - MAC algorithm
- TLS supports several
- **Negotiation:**
 - Client offers choice
 - Server picks 1
 - e.g., the most secure
- **Common algorithms**
 - Public-key encryption
 - RSA, ECDSA
 - Symmetric ciphers
 - AES: block
 - ChaCha20: stream
 - Hash functions
 - SHA-2
 - SHA-3

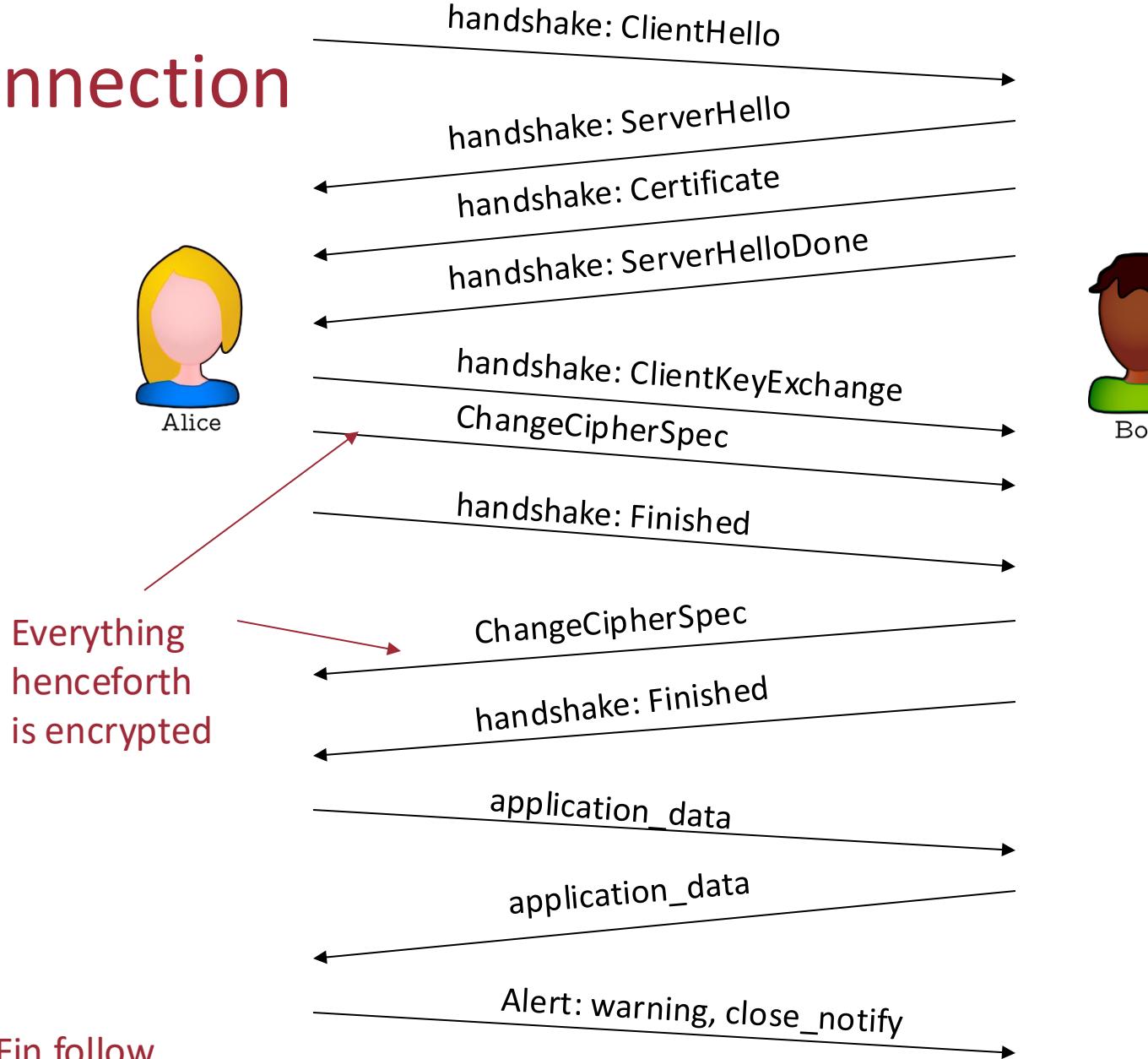
TLS handshake (3)

- Last two steps protect handshake from tampering:
 - Client typically offers range of algorithms, some strong, some weak
 - Man-in-the-middle could delete best algorithms from list
 - Last two message MACs allows detecting such an attack
- Why not simply sign or add MACs to the handshake messages?
 - Not possible: it is the handshake that sets up the keys!

TLS handshake (4)

- Why 2 nonces? (steps 1 and 2 of the handshake)
- Suppose Trudy sniffs all messages between Alice & Bob
- Next day, Trudy sets up TLS connection with Bob (server), sends exact same sequence of records
 - Bob (e.g., Amazon) thinks Alice made two separate orders for the same thing
 - Solution: Bob sends different nonce for each connection. This causes encryption keys to be different on the two days
 - Trudy's messages will fail Bob's integrity check

Real connection



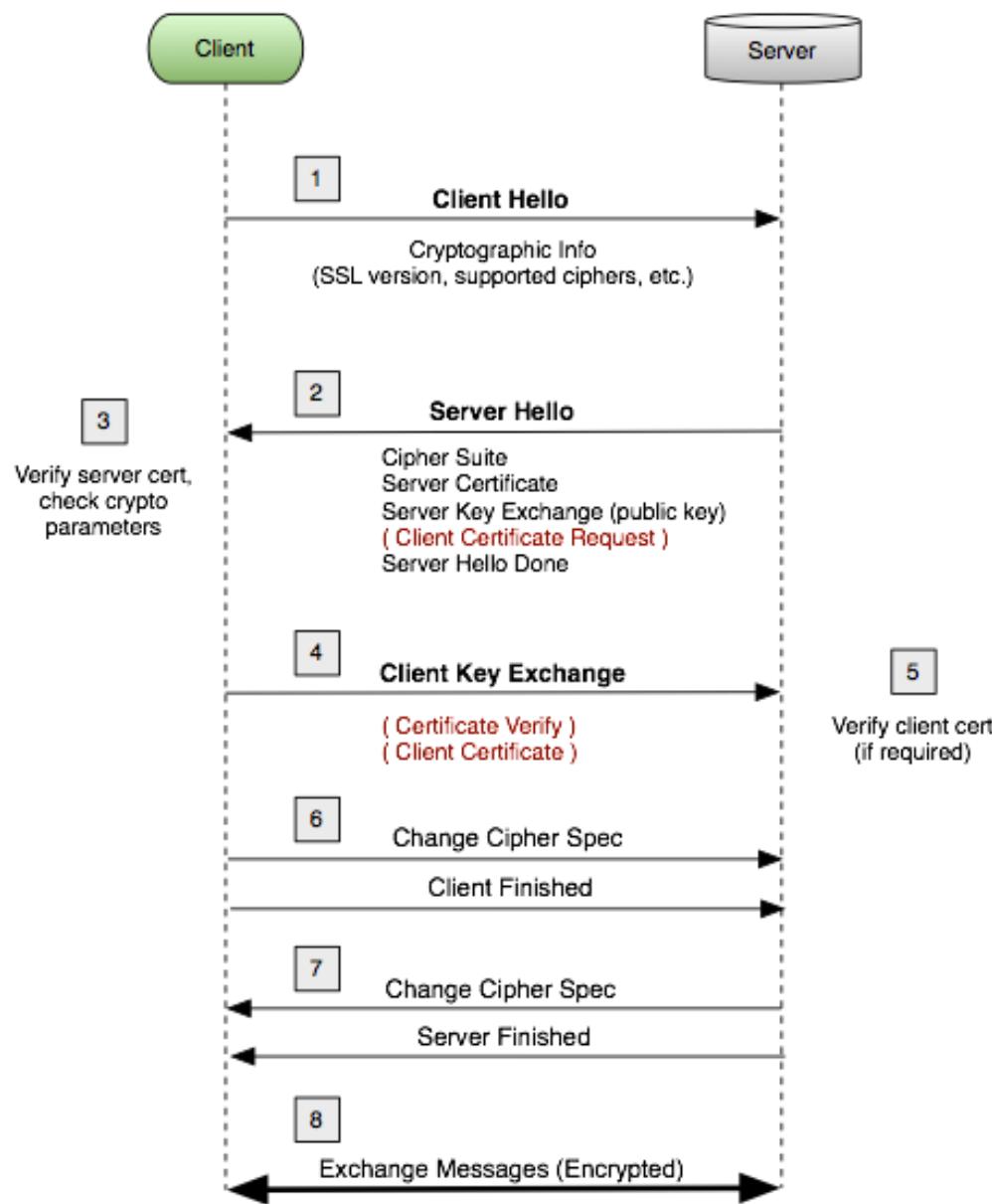
Key derivation

- Client nonce, server nonce, and pre-master secret input into **pseudo random-number generator**, produce **master secret** (MS)
 - **pseudo random-number generator** => same input at client and server produces the same result
- **Master secret** and nonces input into another **pseudo random-number generator**, produce **key block**, cut into:
 - Client MAC key
 - Server MAC key
 - Client encryption key
 - Server encryption key
 - Client initialization vector (IV)
 - Server initialization vector (IV)

TLS secrets and keys

- Master secrets (48 octets)
 - Pre-Master Secret (PMS)
 - Computed with DH; or
 - PMS is chosen by the client and sent to the server ciphered with its public key
 - Master secret (MS)
 - $MS = MD5(PMS \mid SHA('A' \mid PMS \mid R1 \mid R2)) \mid$
 $MD5(PMS \mid SHA('BB' \mid PMS \mid R1 \mid R2)) \mid$
 $MD5(PMS \mid SHA('CCC' \mid PMS \mid R1 \mid R2)) \mid \dots$
- Session keys
 - Computed from a master secret and the random values exchanged in the protocol
 - $ClientMacKey = MD5(MS \mid SHA('A' \mid MS \mid R1 \mid R2))$
 - $ServerMacKey = MD5(MS \mid SHA('BB' \mid MS \mid R1 \mid R2))$
 - $ClientKey = MD5(MS \mid SHA('CCC' \mid MS \mid R1 \mid R2))$
 - $ServerKey = MD5(MS \mid SHA('DDDD' \mid MS \mid R1 \mid R2))$

TLS



Example of client and server “talking”

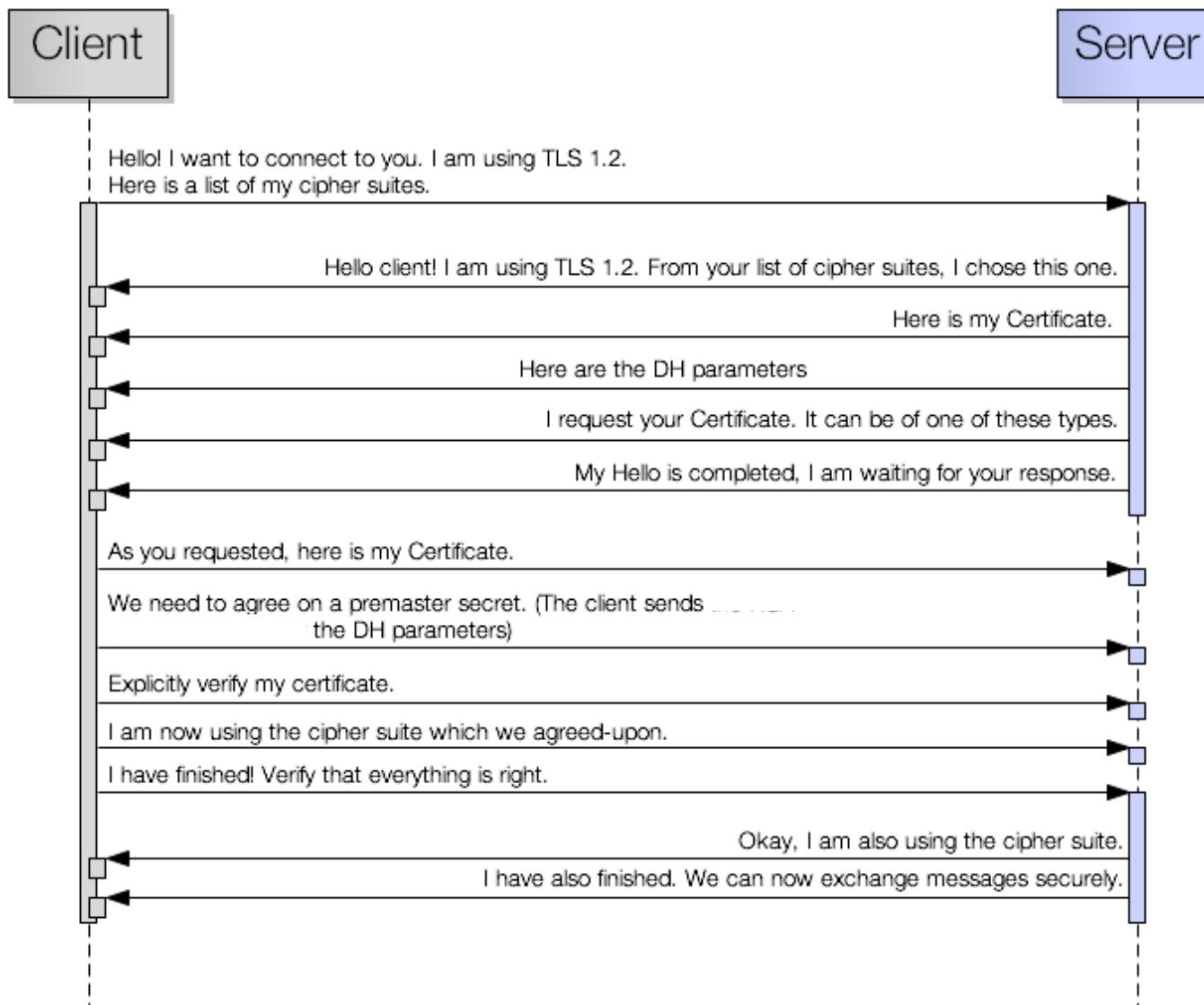


Figure credits: André Joaquim

TLS performance

- First implementations were **slow**
 - TLS was only used for some sensitive connections
- Currently, when properly configured, **TLS can be fast**
 - *"On our production frontend machines, TLS accounts for less than 1% of the CPU load, less than 10 KB of memory per connection and less than 2% of network overhead."*
 - Adam Langley, Google "Overclocking SSL"
<https://istlsfastyet.com/>
- Nowadays, TLS can be the **default** for all connections

Roadmap

- SSH – Secure Shell
- TLS – Transport Layer Security
 - Key management

Key management

- TLS and SSH use **public-key cryptography**
 - (We will study this cryptography later, in more detail)
- **Pair of keys**
 - One is **private**, personal, non-transmissible
 - One is **public**, can/should be widely known
- Allow for
 - **Confidentiality** with the exchange of secret keys
 - **Authentication and Integrity** with digital signatures

Private key

- The private key represents its **owner** so:
 - The probability of it being compromised must be minimized
 - Backup copies must be physically secure
- **Private key** must be protected
 - The access path to the private key must be **restricted**
 - **Password** protected, e.g., JKS, PGP
 - Security of applications using the private key must be **guaranteed**

Private key confinement

- Storage and use of the private key in an autonomous device, e.g., a smartcard
 - The device generate the key pairs
 - The device ciphers/deciphers the data with the key pair controlled by on-chip access mechanisms
 - e.g., access PIN
 - Allows for qualified signatures
 - EU eIDAS Regulation



Public key certificate

- Certificates are documents signed by a certification entity
 - Certification Authority (CA), public organization or company
 - Certificates are public documents
 - Certificates have a digital signature to assure authenticity
- Can distribute public key through unsecure channel
 - Receiver can validate the certificate signature using the CA public key
 - If it trusts the CA and the signature is valid, then it can trust the public key
- Certificate standard format: X.509 (RFC 3280)

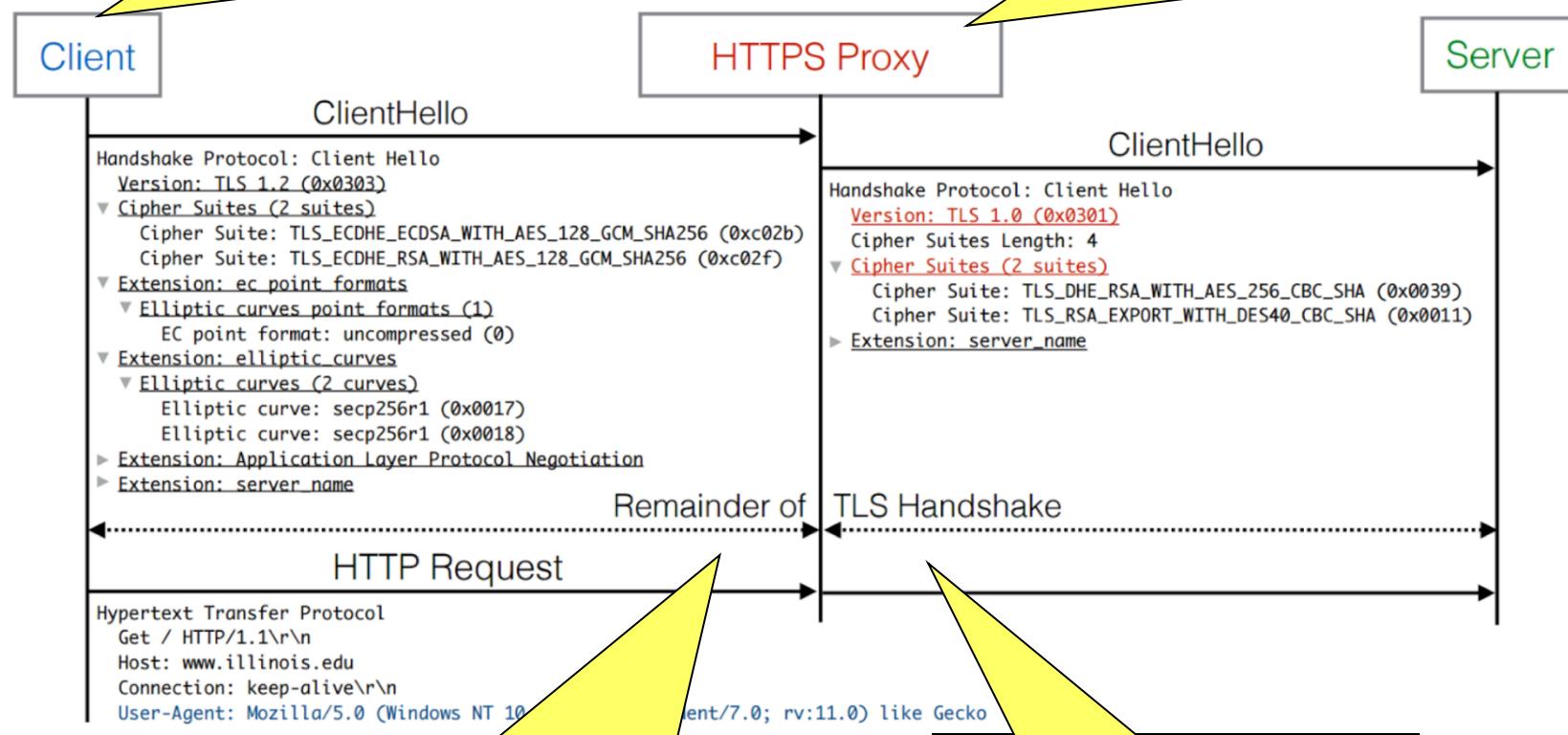
Roadmap

- SSH – Secure Shell
- TLS – Transport Layer Security
 - **Interception attacks**

TLS interception attack

Client is forced (or misled) to install a CA certificate controlled by the proxy

Proxy generates a certificate for the server “on the fly”



TLS session between client and proxy
(client does not see any difference
because it trusts the certificate
generated by the malicious CA)

TLS session between proxy and server

TLS Fingerprints

Fingerprints



Is your employer, school, or Internet provider

eavesdropping on your secure connections?

370 sets of fingerprints checked per day

2,370,158 sets of fingerprints checked for our visitors

Secure browser connections can be intercepted and decrypted by authorities who spoof the authentic site's certificate. But **the authentic site's fingerprint CANNOT be duplicated!**

Domain Name	Certificate Name	EV	Security Certificate's Authentic Fingerprint
www.grc.com	grc.com	—	7A:85:1C:F0:F6:9F:D0:CC:EA:EA:9A:88:01:96:BF:79:8C:E1:A8:33
www.facebook.com	*.facebook.com	—	43:29:AB:C9:18:BB:BB:D5:B1:FC:8E:97:26:59:14:EB:92:B1:12:0D
www.paypal.com	www.paypal.com	●	BD:DE:B3:95:6B:86:79:B8:27:86:DA:4D:75:3C:53:AA:04:1E:08:92
twitter.com	twitter.com	—	73:33:BB:96:1D:DB:9C:0C:4F:E5:1C:FF:68:26:CF:5E:3F:50:AB:96
www.blogger.com	*.blogger.com	—	4D:8C:20:14:4A:3D:BB:CC:1E:62:36:9B:19:1A:3C:CF:E0:B4:CB:F1
www.linkedin.com	www.linkedin.com	—	2C:5C:AD:EB:6F:F3:9F:15:48:61:84:43:29:9C:B5:74:5A:BA:6E:A4
www.yahoo.com	*.www.yahoo.com	—	AD:D7:73:36:32:68:AB:33:71:3E:6E:1D:1B:73:93:1A:F4:2B:DC:D7
wordpress.com	*.wordpress.com	—	7A:C1:B2:7E:09:FF:88:03:C3:E9:B7:4F:31:F4:AC:75:79:BA:66:E6
www.wordpress.com	*.wordpress.com	—	7A:C1:B2:7E:09:FF:88:03:C3:E9:B7:4F:31:F4:AC:75:79:BA:66:E6

Each site's authentic security certificate fingerprint (shown above) was just now obtained by GRC's servers from each target web server. If your web browser sees a different fingerprint for the same certificate (carefully verify the Certificate Name is identical) that forms strong evidence that something is intercepting your web browser's secure connections and is creating fraudulent site certificates.

<https://www.grc.com/fingerprints.htm>

Summary

- SSH – Secure Shell
 - SSH Tunnels
- TLS – Transport Layer Security
 - Toy TLS
 - TLS record and handshake protocols
 - Key management
 - Interception attacks

Management of Public Keys

Segurança Informática em Redes e Sistemas
2023/24

David R. Matos, Miguel Pardal

Ack: Ricardo Chaves, Miguel P. Correia,
André Zúquete, Carlos Ribeiro

Roadmap

- Introduction
- Distribution of public keys
- Public-key certificates
- Certificate issuance
- Certificate distribution
- Certificate revocation

Roadmap

- **Introduction**
- Distribution of public keys
- Public-key certificates
- Certificate issuance
- Certificate distribution
- Certificate revocation

Management problems

- Assure correct use
 - Private keys: assure their privacy/confidentiality
 - Public keys: assure their correct distribution
- Evolution of the mapping between entity \leftrightarrow key pair:
 - Handle common management operations
 - e.g. key renewal
 - Deal with catastrophic situations
 - e.g. loss of the private key
- Assure unpredictability
 - The generation of asymmetric key pairs must use good random number generators

Management goals

- Key management
 - How and when should the asymmetric keys be generated
- Usage of private keys
 - How is their privacy/confidentiality protected
- Distribution of public keys
 - How are public keys distributed in a correct way
- Key lifetime
 - For how long should the key pairs be used
 - How to check for obsolete keys

Guidelines for key pair generation

- Use good random values generators
 - Able to generate acceptable keys for the targeted ciphering algorithm
 - Unpredictability of all the key bits
 - Equiprobability of all the key bits
- Efficiency without sacrificing security
 - Allow the computation to be accelerated in one of the ciphering directions, without compromising the security
- The key pair – especially the private key – should be generated by its owner
 - To assure the maximum privacy of the private key

Correct usage of private keys

- The private key represents its **owner** so:
 - The probability of it being compromised must be minimized
 - Backup copies must be physically secure
- **Private keys must be protected**
 - The access path to the private key must be restricted
 - **Password** protected, e.g., JKS, PGP
 - Security of applications using the private key must be guaranteed

Private key confinement

- Storage and use of the private key in an autonomous device, e.g., a smartcard
 - The device generates the key pairs
 - The device ciphers/deciphers the data with the key pair controlled by on-chip access mechanisms
 - e.g. access PIN
 - Allows for qualified signatures
 - EU eIDAS Regulation



Roadmap

- Introduction
- **Distribution of public keys**
- Public-key certificates
- Certificate issuance
- Certificate distribution
- Certificate revocation

Distribution of public keys

- Techniques
 - Manual
 - Not practical
 - Using a shared secret
 - If a shared secret already exists
 - Public announcement
 - Public directory
 - **Public distribution using digital certificates
(next section)**
-
- The diagram illustrates the progression of public key distribution techniques over time. It features two main colored boxes: an orange box labeled 'PAST' containing 'If a shared secret already exists' and a green box labeled 'PRESENT' containing 'Public distribution using digital certificates (next section)'. A large orange 'X' is drawn through the 'Public announcement' and 'Public directory' items. A diagonal line connects the bottom-left corner of the 'PAST' box to the top-right corner of the 'PRESENT' box, indicating a transition from past methods to present ones.

Roadmap

- Introduction
- Distribution of public keys
- **Public-key certificates**
- Certificate issuance
- Certificate distribution
- Certificate revocation

Public key certificate (digital certificate)

- Certificates are documents signed by a certification entity
 - Certification Authority (CA), public organization or company
 - Certificates are public documents
 - Certificates have a digital signature (cryptographic protection)
- Used to distribute public keys through unsecure channels
 - Receiver can validate the certificate signature using the CA public key
 - If it trusts the CA and the signature is valid, then it can trust the public key
- Certificate structure
 - X.509 standard (RFC 3280)
 - PKCS #7 Cryptographic Message Syntax (CMS) standard (RFC 5652)
 - SPKI (Simple Public Key Infrastructure) – historical
 - KeyNote trust-management system – historical

X.509 v3 Digital Certificate (RFC3280)

- Contents
 - Version
 - Serial number (of the CA)
 - Issuer (CA)
 - Validity
 - Not Before
 - Not After
 - Subject
 - Subject Public Key info
 - Public Key Algorithm (ex. RSA)
 - Subject Public Key
 - Extensions (optional)
 - Certificate Signature Algorithm (ex.: RSA w/SHA-256)
 - Certificate Signature Value
- Extensions
 - Issuer Unique Identifier (v2)
 - Subject Unique Identifier (v2)
 - Authority Key Identifier
 - Subject Key Identifier
 - Key Usage
 - digitalSignature
 - nonRepudiation
 - keyEncipherment
 - dataEncipherment
 - keyAgreement
 - keyCertSign
 - CRLSign
 - encipherOnly
 - decipherOnly
 - Extended Key usage
 - CRL Distribution Points
 - Private Key usage period

See a certificate in the browser

Formats & Extensions for X.509

- **.PEM, .CRT, .KEY, etc.** – certificate in textual Base64 format
 - “-----BEGIN CERTIFICATE-----”
 - “-----END CERTIFICATE-----”
 - Most commonly used
- **.DER** – certificate in DER binary format
 - DER – ASN.1 Distinguished Encoding Rules (tag, length, value)
 - **.CER** – set of certificates in the DER format
 - Typically used in Java platforms
- **.P7B and .P7C** – certificate(s) in PKCS#7 textual Base64 format
 - Used in Microsoft Windows
- **.PFX and .P12** - PKCS#12 – binary format
 - Set of certificates and private keys, protected by password
 - Used in Microsoft Windows

Run locate *.pem then cat some files; same with *.der

Certification Authorities (CA)

- CAs: organizations that manage certificates
 - Define policies and mechanisms for the generation and distribution of certificates
 - Manage the certificate revocation lists
- Trust in the CAs
 - Manual distribution of their public keys
 - Centralized certification (single CA)
 - Ad-hoc certification (e.g. PGP)
 - Certification hierarchy
 - Public key certificates for the CAs
 - **Manual distribution of root CA public keys, e.g., in web browsers**

Roadmap

- Introduction
- Distribution of public keys
- Public-key certificates
- **Certificate issuance**
- Certificate distribution
- Certificate revocation

Asymmetric key pairs validity

- Keys to assure **confidentiality**
 - The public key of X is used by the sender to assure confidentiality of the data sent to X
 - And the private key of X is used to decipher the received information
 - These keys can be refreshed frequently
 - In the worst scenario, the data is re-sent
- Keys to assure **authentication**
 - The private key of X is used to sign the content
 - And the corresponding public key to validate the signature
 - These keys should not be renewed frequently
 - To simplify the signature validation process

Roadmap

- Introduction
- Distribution of public keys
- Public-key certificates
- Certificate issuance
- **Certificate distribution**
- Certificate revocation

PKI (Public Key Infrastructure)

- Infrastructure to manage certificates in a certain context
 - Example context: the Web
- Encompasses:
 - A set of CAs and similar entities
 - Policies and mechanisms
- Operations supported
 - Secure creation of asymmetric key pairs
 - Creation and distribution of public key certificates
 - Definition and usage of certification chains
 - Update, publication and query of certificate revocation lists

PKI entities

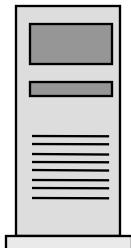
Certification Authority (CA)

Reliable entity that creates and publishes the certificates in the repository.



Certification Revocation List Authority (CRLA)

Trusted entity that creates and publishes the revocation certificates in the repository.



Repository

Subscriber

- Generates a key pair
- Requests a certificate for its public key
- Receives the certificate
- Uses its private key



Verifier

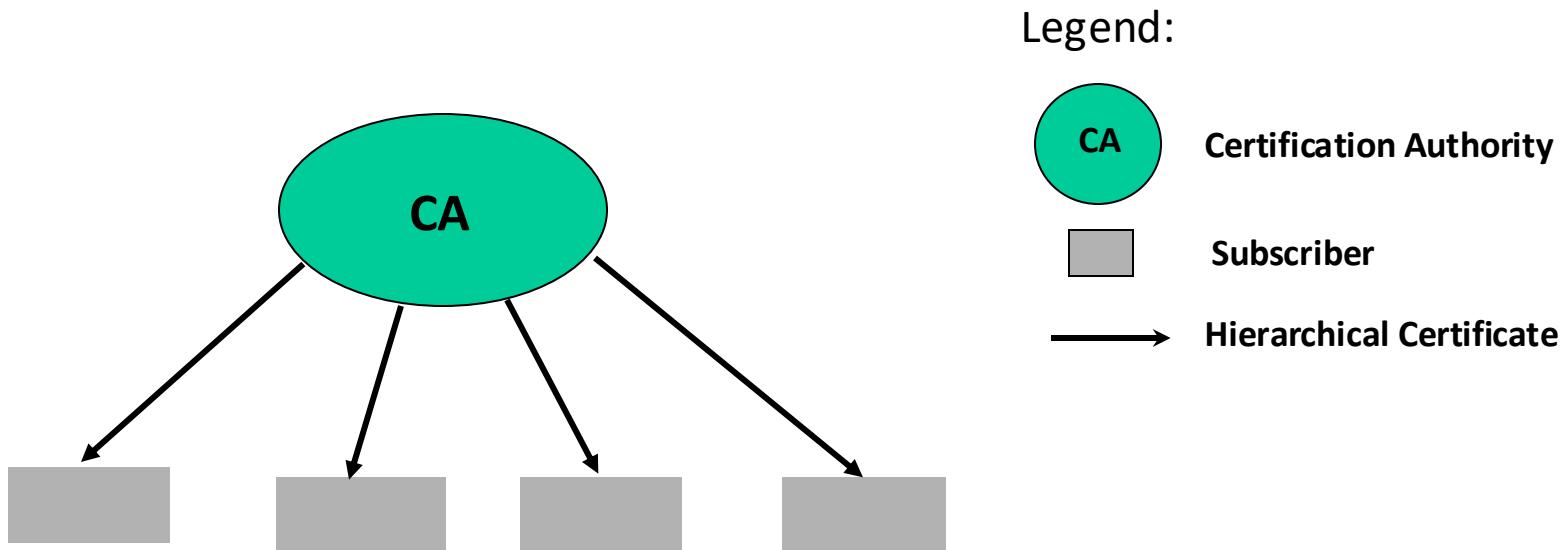
- Finds out certificates in the repository
- Validates certificates in order to validate a certification chain
- Uses the public key of the subscriber

PKI: Trust relations

- A CA establishes trust relations in two ways:
 - By issuing public key certificates of other CAs
 - Below in the hierarchy or hierarchically unrelated
 - By requiring certification of its public key to other CAs
 - Above in the hierarchy or hierarchically unrelated
- Typical trust relations
 1. Flat
 2. Hierarchical
 3. List of CAs

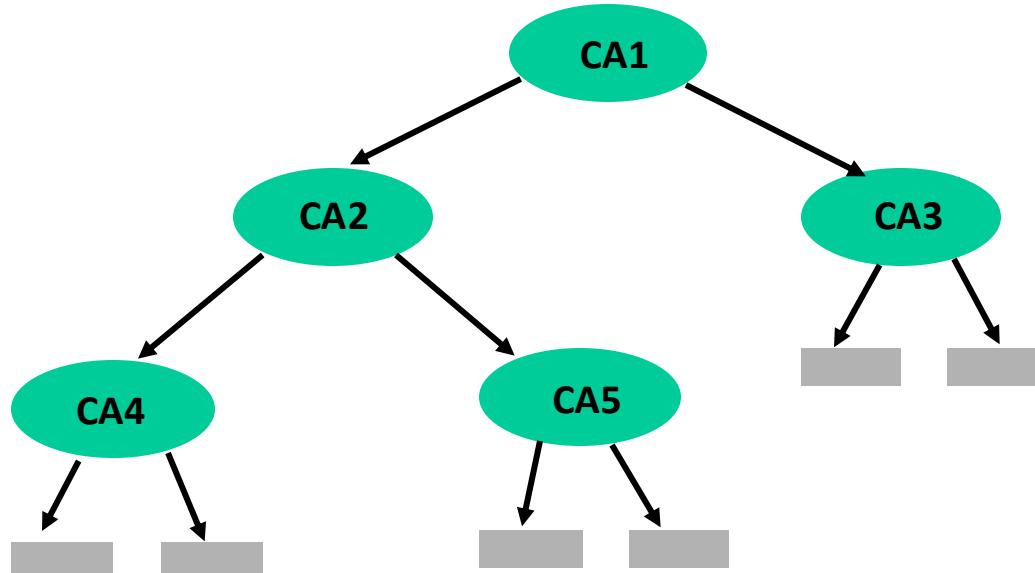
1. Flat

- Trusted single root CA
 - Verifying entities trust the public key of a single well-known CA
- Verifying entities check the certificates validity with the public key of the CA.



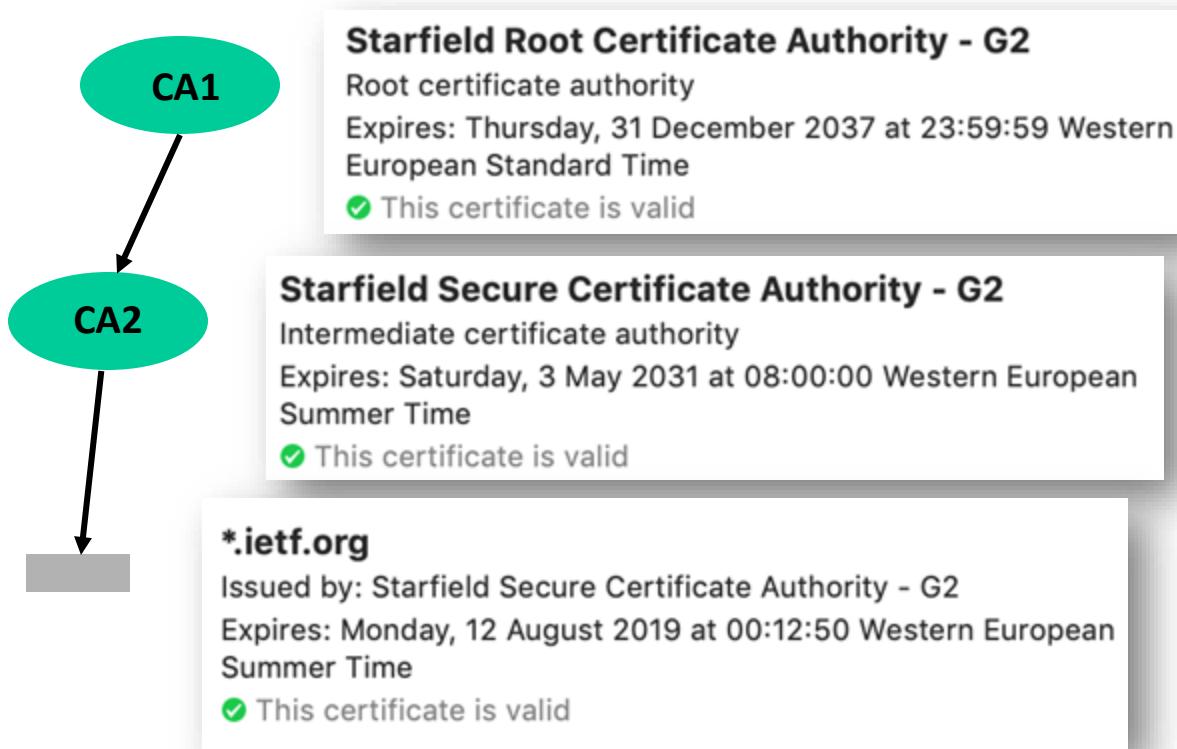
2. Hierarchical

- A tree of CAs
- The verifying entities trust the key of CA1
- CAs issue certificates to subscribers and other CAs
- *Verifying entities verify the certificates of the subscribers by sequentially checking the certificates up to the root certificate*



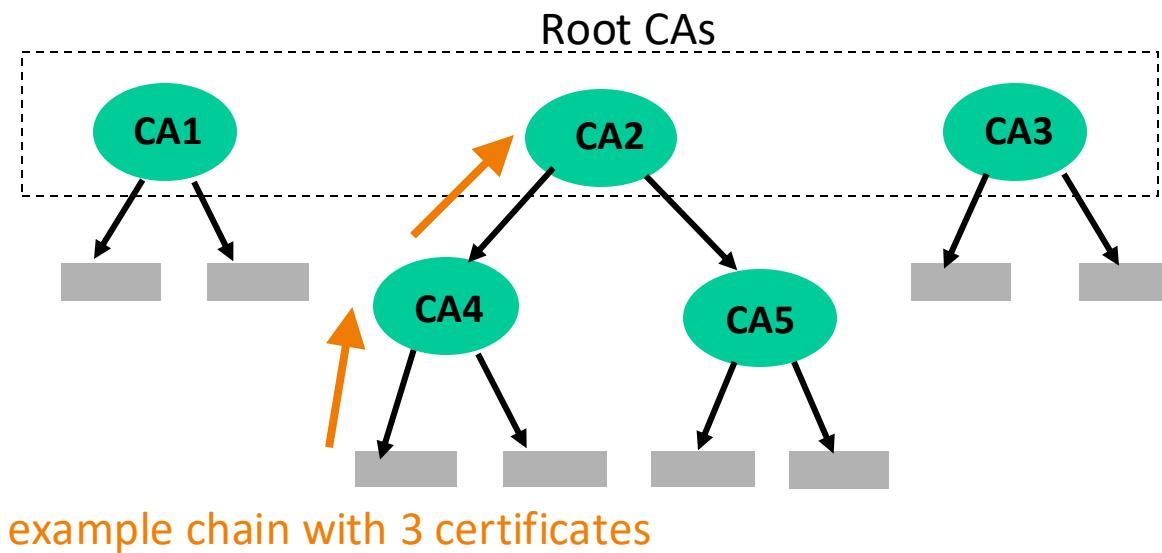
Intermediate certificates

- Two primary reasons to use intermediate certificates:
 - Protect the PKI root certificate
 - To delegate signing authority to another organization (sub-CA); needed for scalability reasons



3. List of certificates

- The verifying entities trust the keys of several root CAs
- The verifying entities validate the chain of certificates that lead to any of the CAs



OS/browser root stores

- **Root store:** list of certificates of trusted CAs
 - CAs trusted to issue certificates to the correct entities
 - Applications that use X.509 need to have a root store
 - Operating systems have root stores
 - Windows, OS X (Keychain), Linux
 - Browsers use root stores: Mozilla ships its own, Edge uses Windows' root store, Chrome can pick OS or its own, etc.

See OS root store

Windows Trusted Root Certificate Authorities

The screenshot shows the MMC console window titled "Console1 - [Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities\Certificates]". The left pane displays a tree view of certificate categories, with "Certificates (Local Computer)" expanded to show "Trusted Root Certification Authorities" and its sub-node "Certificates" selected. The right pane is a grid table listing various certificates, with columns for "Issued To", "Issued By", "Expiration Date", "Intended Purposes", and "Friend". The table lists numerous entries, including well-known root CAs like DigiCert, Comodo, and GlobalSign, along with Microsoft's own root certificates.

Issued To	Issued By	Expiration Date	Intended Purposes	Friend
AAA Certificate Services	AAA Certificate Services	2028-12-31	Client Authentication...	Sectigo
Actalis Authentication Root CA	Actalis Authentication Root CA	2030-09-22	Client Authentication...	Actalis
AddTrust External CA Root	AddTrust External CA Root	2020-05-30	Client Authentication...	Sectigo
AffirmTrust Commercial	AffirmTrust Commercial	2030-12-31	Client Authentication...	Affirm
Baltimore CyberTrust Root	Baltimore CyberTrust Root	2025-05-12	Client Authentication...	DigiCert
Buypass Class 2 Root CA	Buypass Class 2 Root CA	2040-10-26	Client Authentication...	Buypass
Certum CA	Certum CA	2027-06-11	Client Authentication...	Certur
Certum Trusted Network CA	Certum Trusted Network CA	2029-12-31	Client Authentication...	Certur
Chambers of Commerce Root - 2...	Chambers of Commerce Root - 2...	2038-07-31	Client Authentication...	Cham
Class 3 Public Primary Certificat...	Class 3 Public Primary Certificatio...	2028-08-01	Client Authentication...	VeriSig
COMODO ECC Certification Auth...	COMODO ECC Certification Auth...	2038-01-18	Client Authentication...	Sectigo
COMODO RSA Certification Auth...	COMODO RSA Certification Auth...	2038-01-18	Client Authentication...	Sectigo
Copyright (c) 1997 Microsoft C...	Copyright (c) 1997 Microsoft Corp.	1999-12-30	Time Stamping	Micros
Deutsche Telekom Root CA 2	Deutsche Telekom Root CA 2	2019-07-09	Client Authentication...	Deutsc
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	2031-11-10	Client Authentication...	DigiCe
DigiCert Global Root CA	DigiCert Global Root CA	2031-11-10	Client Authentication...	DigiCe
DigiCert Global Root G2	DigiCert Global Root G2	2038-01-15	Client Authentication...	DigiCe
DigiCert Global Root G3	DigiCert Global Root G3	2038-01-15	Client Authentication...	DigiCe
DigiCert High Assurance EV Ro...	DigiCert High Assurance EV Root ...	2031-11-10	Client Authentication...	DigiCe
DST Root CA X3	DST Root CA X3	2021-09-30	Client Authentication...	DST Re
D-TRUST Root Class 3 CA 2 2009	D-TRUST Root Class 3 CA 2 2009	2029-11-05	Client Authentication...	D-TRU
Entrust Root Certification Auth...	Entrust Root Certification Authority	2026-11-27	Client Authentication...	Entrus
Entrust Root Certification Auth...	Entrust Root Certification Authori...	2037-12-18	Client Authentication...	Entrus
Entrust Root Certification Auth...	Entrust Root Certification Authori...	2030-12-07	Client Authentication...	Entrus
Entrust.net Certification Author...	Entrust.net Certification Authority	2029-07-24	Client Authentication...	Entrus
Equifax Secure Certificate Auth...	Equifax Secure Certificate Authority	2018-08-22	Code Signing, Secu...	GeoTr
Federal Common Policy CA	Federal Common Policy CA	2030-12-01	Client Authentication...	U.S Go
GeoTrust Global CA	GeoTrust Global CA	2022-05-21	Client Authentication...	GeoTr
GeoTrust Primary Certification ...	GeoTrust Primary Certification Au...	2038-01-18	Client Authentication...	GeoTr
GeoTrust Primary Certification ...	GeoTrust Primary Certification Au...	2037-12-01	Client Authentication...	GeoTr
Global Chambersign Root - 2008	Global Chambersign Root - 2008	2038-07-31	Client Authentication...	Global

MMC –
Microsoft
Management
Console

Basic Solutions: advantages & disadvantages

- Flat
 - + Simpler
 - Limited to a single organization
 - Scales poorly
- Hierarchical
 - + Simple to find a certification path
 - Clients trust a single global entity
- List of Certificates
 - + Solves problems of the previous two
 - Client does not know the practices of each root CA
 - Client does not know which CA was used to verify a given certificate
 - Revocation is difficult

Roadmap

- Introduction
- Distribution of public keys
- Public-key certificates
- Certificate issuance
- Certificate distribution
- **Certificate revocation**

Certificate withdrawal

- There are several cases when a certificate must be withdrawn:
 - Corresponding private key compromised
 - Certificate owner does not operate service any longer
 - Key ownership has changed
 - Certificates issued to entity that not the one indicated

<https://www.zdnet.com/article/microsoft-warns-fraudulent-digital-certificates-issued-for-high-value-websites/>

Certificate revocation

- Revocation is crucial — yet often neglected
 - No certificate should be considered valid without a revocation check
 - Because we need confirmation that a certificate is valid **at the moment** of interest, not sometime in the past
- In these cases, there are two options: CRLs and OCSP

CRL (Certificate Revocation Lists)

- CRLs are lists of revoked certificates
 - Should be regularly checked by the certificate holders
- Maintenance and dissemination of CRLs
 - Institutional certification
 - Each CAs maintains and allows reading access to the list it keeps/knows
 - Example: <http://crl.multicert.com/>
 - The CAs exchange CRLs themselves in order to facilitate the knowledge of all revoked certificates
 - Ad-hoc certification
 - The entity that holds the revoked key pair must create and publicize the revocation certificate the best it can

Problems with CRLs

- Intermediate certificates should be checked too
 - Induces load and network activity
- There is a time interval between two updates which is a window for attack
- CRLs can become large
 - Solution: delta CRLs that contain only latest updates
 - Requires server-side support—very rarely used
- Downloads of CRLs can be blocked by a man-in-the-middle
- For these reasons: most browsers have never activated CRLs checks by default 😞

OCSP (Online Certificate Status Protocol)

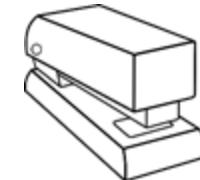
- OCSP allows live revocation checks over the network
- Request-response model
 - Request: lookup of certificate in server-side CRL data structure
 - Certificates contain the URL of their issuer's OCSP server
 - Query by several hash values and certificate's serial number
 - Protection from replay attacks with nonces
 - Query may be signed
 - Does not require encryption
 - Response:
 - Contains certificate status: **good**, **revoked**, unknown
 - Must be signed
 - Does not require encryption

Problems with OCSP

- Lookups go **over the network**
 - Induces latency
- OCSP information must be **fresh**
- OCSP servers must have **high availability**
- OCSP **can be blocked** by a man-in-the-middle
 - Many browsers will ‘soft-fail’ = show no error
 - Browsers ‘**accept as good**’ if no OCSP response received
- **Privacy exposure:** OCSP servers know which sites the users are accessing

OCSP stapling

- Idea: the web server obtains a fresh OCSP response and “staples” it to the certificate given to the web browser
 - The browser checks the signature of the certificate and of the recent OCSP validity assertion
- More efficient
 - One call to OCSP gets a response that can be served to multiple clients for a period
 - Browser receives certificate and OCSP from server in the same response
- More secure
 - CA will deny OCSP response for a revoked certificate
- More private
 - It is the server that calls OCSP, not the client
- Support for OCSP stapling is increasing, but still not universal
 - Servers: Windows, Apache and Nginx
 - Clients: Chrome and Firefox



New approaches to revocation

- In-browser revocation lists
 - Browsers preload a list of revoked certificates for the most common and important domains
 - Limited number, not scalable
 - Updates are distributed via the browser's update mechanism
 - E.g. Google Chrome
- Short-lived certificates
 - Give certificates a very short validity period
 - 1 hour–1 day
 - Replace certificates fast; do not attempt any other revocation
 - Works well and gives clearly-defined window of attack
 - Problem: certification becomes a frequent and ‘live’ operation
 - Not applied in the Web so far

Revocation conclusion

- Revocation is crucial—but not fully solved so far
 - CRLs are of limited use
 - OCSP checks are expensive (latency, load) and not enough against an attacker who can drop traffic to the CA
 - Other approaches have not gained wide adoption

Summary

- Introduction
- Distribution of public keys
- Public-key certificates
- Certificate issuance
- Certificate distribution
- Certificate revocation

Management of Secret Keys

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chave
Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Roadmap

- Introduction
- Generation and manual distribution
- Distribution with shared values
- Distribution without shared values
- Distribution with third parties
- Key renewal

Roadmap

- **Introduction**
- Generation and manual distribution
- Distribution with shared values
- Distribution without shared values
- Distribution with third parties
- Key renewal

Secret key management: problems

- Ciphered data is confidential, but only if the ciphering key is secret
 - Distribution/storage of the keys must assure its confidentiality
- The more unpredictable the generated keys are, the harder it is to “guess” their value
 - Key values should be as random as possible
- Computers are not good random generators
 - We need to discover and use random data and random behaviors in the system
- The excessive use of the keys eases their discovery
 - We need to quantify and impose limits to the use of a key

Secret key management: aspects

- Key generation
 - How and when should the secret key be created?
- Key distribution
 - How are the keys distributed to a limited number (typically, 2) of communicating parties?
- Key lifetime
 - For how long should a key be used?

Secret key management: Renewal of keys

- Goal
 - Minimize the cryptanalysis risk
 - Applicable to session keys and long-term keys
- Criteria
 - After a predetermined time interval
 - To avoid its discovery during its usage lifetime
 - That might allow to deterministically modify cryptograms and observe the plaintext
 - After a given amount of exchanged cipher data
 - To avoid the excessive use of the key

Roadmap

- Introduction
- **Generation and manual distribution**
- Distribution with shared values
- Distribution without shared values
- Distribution with third parties
- Key renewal

Generation of secret keys: principles

- Use good random values generators
 - Should be able to generate any of the key values acceptable for the ciphering algorithm
 - Equiprobability of all the key bits
 - Typically generated by pseudo-random generators
 - Validated by randomness test functions
 - Unpredictability of all the key bits
 - Should not be predictable even if the algorithm and all the generation history is known
 - Symmetric ciphers usually have a few weak keys
 - Must be discarded when returned by the random key generator

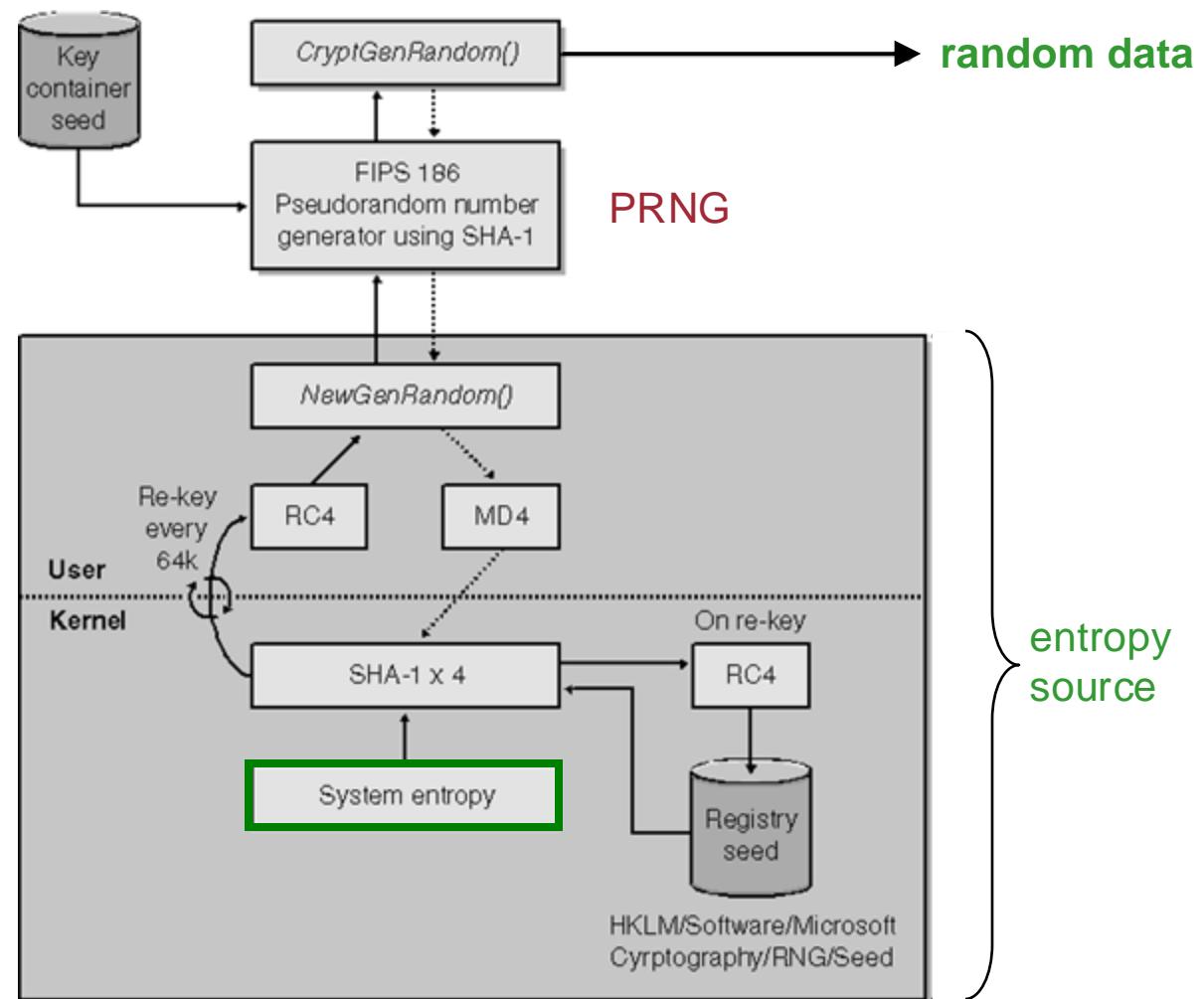
Random number generation: hardware

- Entropy: measure of randomness in a signal/random event
 - High entropy means high randomness (what we need)
 - Concept from Information Theory
- Hardware random number generator (HRNG)
 - A.k.a. true random number generator (TRNG)
 - Hardware device that gets entropy from a physical source
- Example physical sources:
 - Atmospheric noise – read by a radio receiver, for example
 - Thermal or quantum-mechanical noise
 - Amplified to provide a random electrical signal
 - e.g. thermal noise from a resistor
 - Nuclear decay radiation
 - e.g. some commercial smoke alarms, detected by a Geiger counter



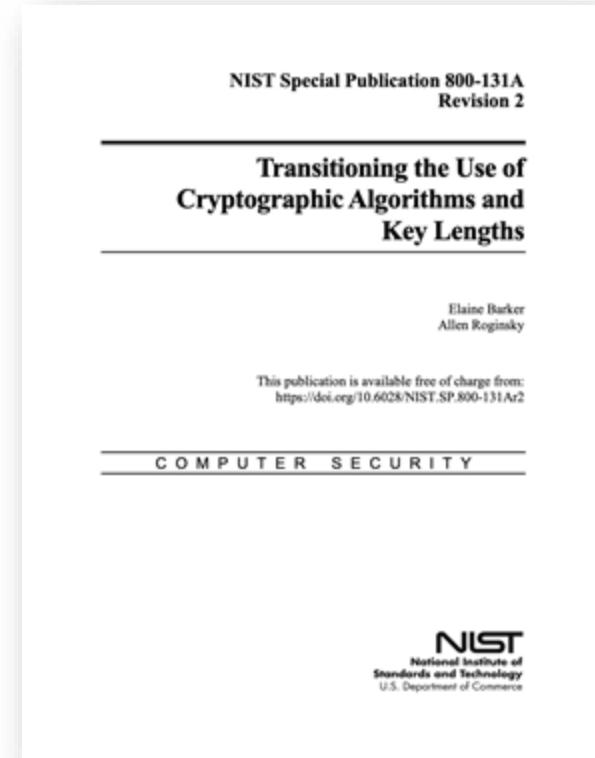
Random number generation: Windows CryptGenRandom / BCryptGenRandom

- Software random number generator
- Sources of entropy:
 - Ticks since boot
 - Current time
 - Several high-precision performance counters
 - Low-level system info (idle processing time, I/O read and write transfer counts, ...)



Generation of secret keys: size

- What should the size of a secret key be?
Depends on:
 - Algorithm strength
 - Lifetime of the key
 - Usage of the algorithm + key
 - Attacker's power
- Follow recommendations
 - ENISA, NIST,...



Manual distribution (1/2)

- Usefulness:
 - Personal keys – that authenticate a person (e.g. password)
 - Large sets of keys – to be used for long periods of time
- Common requirements:
 - Confidentiality: keys cannot be revealed during generation and distribution
 - Authenticity and integrity: the receiver must be able to check the key's authenticity and integrity
 - All entities that may have access to the key should be considered
 - System administrators, key distributors, etc.

Manual distribution (2/2)

- Physical support
 - On volatile media
 - e.g. screen showing the new password to the users
 - On paper
 - Typically used to transmit personal keys
 - ATM (Multibanco) or VISA PINs
 - Writable media
 - USB drives, magnetic cards, smartcards
- Distribution
 - On-site
 - Hand-to-hand

Roadmap

- Introduction
- Generation and manual distribution
- **Distribution with shared values**
- Distribution without shared values
- Distribution with third parties
- Key renewal

Distribution with long-term shared secrets (1/3)

- Usefulness
 - Allow exchanging temporary secrets between entities that already share some secret information (long-term secrets)
- Nomenclature
 - Long-term shared secrets
 - Key Encrypting Keys, KEK
 - Temporary secrets to be shared
 - Sessions keys, Ks

Perfect Forward Secrecy (PFS)

- Is a desirable characteristic of a key agreement protocol
- Gives assurance that **session keys will not be compromised even if the private key of the server is compromised**
- Protects *past* sessions against *future* compromises of keys
 - By generating a unique session key for every session a user initiates, the compromise of a single session key will not affect any data other than that exchanged in the specific session protected by that particular key

Distribution with long-term shared secrets (2/3)

- Distribution

$$A \rightarrow B: \{K_s\}_{KEK}$$

- Encrypted using a symmetric cipher
- Guarantees authenticity under a set of assumptions:
 - Only A and B know KEK
 - B verifies the message **freshness**
 - Avoid **replay attacks** (see later)
 - B verifies the actual content of the message is $\{K_s\}_{KEK}$

Distribution with long-term shared secrets (3/3)

- Practical aspects to consider
 - KEKs should only be used to cipher session keys
 - In order to prevent cryptanalysis
 - The more session data is ciphered, the more the KEK is exposed
 - Perfect Forward Secrecy (PFS) is not assured
 - The disclosure of the KEK reveals all session keys that have been exchanged between the communicating parties
 - A session key should not be used as a KEK
 - Because, by definition, it is or will be extensively exposed by its repeated use

Distribution with shared public keys

- Similar to distribution of keys with **shared secrets (keys)**
 - No KEKs, but the **public key** of the receiver
 - Typically designated **hybrid ciphers** or **hybrid encryption**
 - Example: PGP (using RSA asymmetric keys)
$$A \rightarrow B: \{K_s\}_{KB-Pub}$$
- Does not assume authentication
 - The receiver's **public key** is used to send the secret
 - Anyone can know the receiver's **public key**
- Practical aspects to be considered
 - Perfect Forward Secrecy (PFS) not assured: disclosure of the receiver's **private key (!)** reveals all session keys exchanged

Distribution of secret keys: key size

Table 5: Approval Status for the RSA-based Key Agreement and Key Transport Schemes

Scheme	Implementation Details	Status
SP 800-56B Key Agreement and Key Transport schemes	$\text{len}(n) < 2048$	Disallowed
	$\text{len}(n) \geq 2048$	Acceptable
Non-SP 800-56B-compliant Key Agreement and Key Transport schemes	$\text{len}(n) < 2048$	Disallowed
	PKCS1-v1_5 padding	Deprecated through 2023 Disallowed after 2023
	Other non-compliance with SP 800-56B	Deprecated through 2020 Disallowed after 2020

NIST Special Publication 800-131A
Revision 2

Transitioning the Use of
Cryptographic Algorithms and
Key Lengths

Elaine Barker
Allen Roginsky

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-131A2>

COMPUTER SECURITY



Roadmap

- Introduction
- Generation and manual distribution
- Distribution with shared values
- **Distribution without shared values**
- Distribution with third parties
- Key renewal

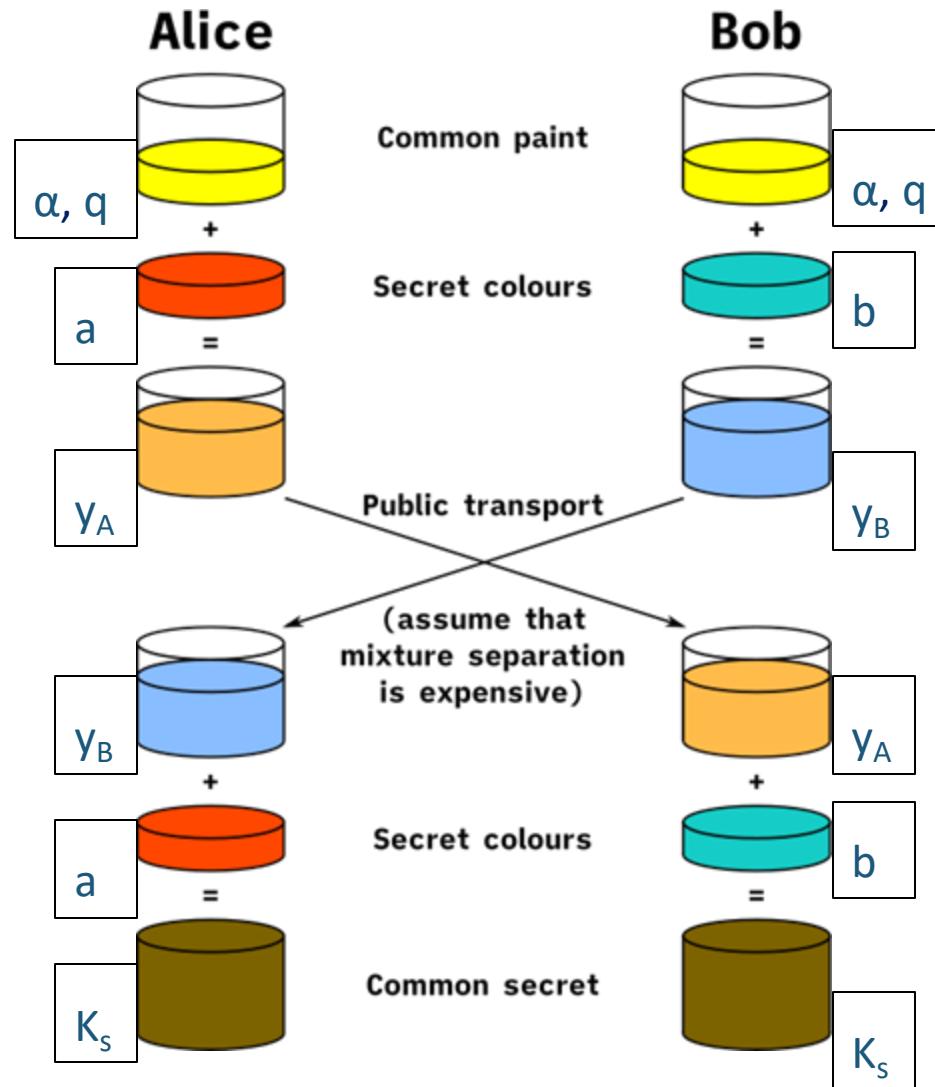
Distribution without sharing values

- Diffie-Hellman (DH) algorithm
 - DH is seminal asymmetric cryptography algorithm, published in 1976
 - Allows generating a shared key
 - But it is **not** an encryption algorithm
 - In practice, DH values have to be shared:
 - The shared values are **not secret**, they are public
 - The sharing is **ephemeral** (temporary), not long-term
 - Participants start with two public parameters: q and α
 - q is a large prime; operations are done **modulo q**
 - α is the exponentiation base
 - α is a primitive root modulo q , i.e., for every integer a coprime to q , there is an integer k such that $\alpha^k \equiv a \pmod{q}$

Diffie-Hellman algorithm (1/3)

- Algorithm (α and q are public)
 - A and B generate random and secret values a and b
 - A computes $y_A = \alpha^a \text{ mod } q$
B computes $y_B = \alpha^b \text{ mod } q$
 - A and B exchange y_A and y_B (public values of DH)
 - A computes $K_s = y_B^a \text{ mod } q = (\alpha^b \text{ mod } q)^a \text{ mod } q = \alpha^{ba} \text{ mod } q = \alpha^{ab} \text{ mod } q$
 - B computes $K_s = y_A^b \text{ mod } q = (\alpha^a \text{ mod } q)^b \text{ mod } q = \alpha^{ab} \text{ mod } q$
- The security of the scheme is based on the complexity of the discrete logarithm problem
 - Knowing α , q , y_A and y_B it is unfeasible to obtain a , b and K_s
 - Specifically, it is unfeasible to compute $a = \log_\alpha(y_A)$ (same for b)
- Elliptic curve version exists: ECDH

Diffie-Hellman: color analogy



Diffie-Hellman algorithm (2/3)

- Distribution does not provide authentication
 - Vulnerable to adversary-in-the-middle attacks
- To authenticate y_A and y_B with digital signatures:
 - A and B must know the other's public key
 - A needs to send y_A with a **signature** obtained with its private key
 - And B needs to sign its value with its private key
 - Example: PGP (with DH/DSS asymmetrical keys)

Diffie-Hellman algorithm (3/3)

- Practical aspects to be considered
 - If both secret values a and b are ephemeral (e.g., used only once),
then there is Perfect Forward Secrecy
 - If a or b are long-term secret values and one of them is disclosed, then all keys they helped generate are revealed

Perfect Forward Secrecy (PFS) with DH

- PFS means that the **session keys** from **past sessions** are not compromised
 - Even if all secrets of the system are compromised **in the present**
- PFS means that the **messages** exchanged between the parties **in the past** will remain protected
- Way to achieve this:
 - Use DH with ephemeral a and b random values

Roadmap

- Introduction
- Generation and manual distribution
- Distribution with shared values
- Distribution without shared values
- **Distribution with third parties**
- Key renewal

Distribution with a trusted third party

(1/3)

- Trusted third parties (Key Distribution Centers)
 - Act as mediators between the communicating parties
 - Distribute credentials for a secure interaction
 - Simplifies the management of long-term shared secrets
 - Avoids need of sharing a secret between any two communicating parties
 - Allows the authentication to be centralized
 - Central point of knowledge of shared secrets
- Assumptions
 - Third party always acts correctly (“trusted”)
 - Do not disclose nor incorrectly use the secrets they know
 - Generate unpredictable/random session keys
 - Are secure, i.e., manage to protect the secrets they store

Distribution with a trusted third party (2/3)

- Distribution

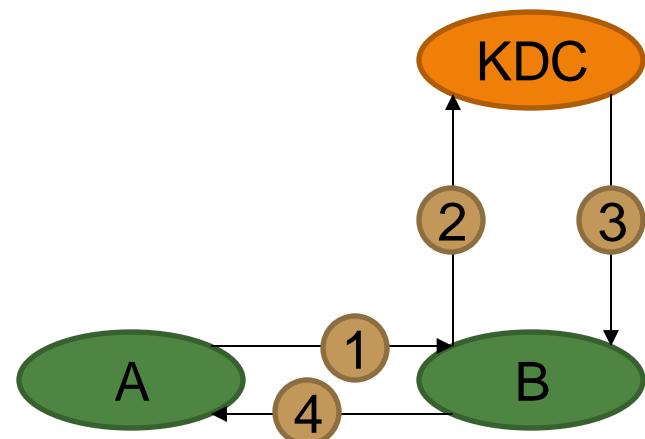
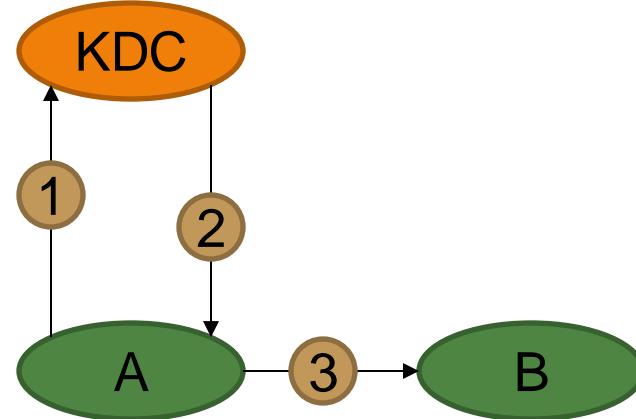
- Pull model

- 1: A -> KDC: A, B
- 2: KDC -> A: $\{K_s\}_{K_A}, \{A, K_s\}_{K_B}$
- 3: A -> B: $A, \{A, K_s\}_{K_B}$
 $A \Leftrightarrow B: \{M\}_{K_S}$

- Push model

- 1: A -> B: A
- 2: B -> KDC: A, B
- 3: KDC -> B: $\{K_s\}_{K_B}, \{B, K_s\}_{K_A}$
- 4: B -> A: $\{B, K_s\}_{K_A}$
 $A \Leftrightarrow B: \{M\}_{K_S}$

K_A shared between A and KDC; same for K_B



Distribution with a trusted third party (3/3)

- Distribution assumes authentication
 - Only those who share a key with the KDC can obtain session key
 - When B receives $\{A, K_s\}_{K_B}$ it is assured that it is receiving a key K_s to communicate with A
- Problems to be solved
 - Message authentication
 - Origin, content, freshness
 - Cooperation between different KDCs
 - Facilitate the key exchange between entities known by different KDCs
- Practical aspects to be considered
 - Perfect Forward Secrecy (PFS) is not assured

Replay attacks

- Messages copied and later resent
- Avoided by guaranteeing message freshness
 - Sequence numbers
 - Timestamps
 - Challenge/Response

Replay attacks (1/3)

- Sequence numbers
 - Sender adds counter value to message content and increments it
 - Receiver checks if counter value received is ok
- Problems
 - Participants need to keep synchronized counters
 - Difficult when message loss or duplication occurs

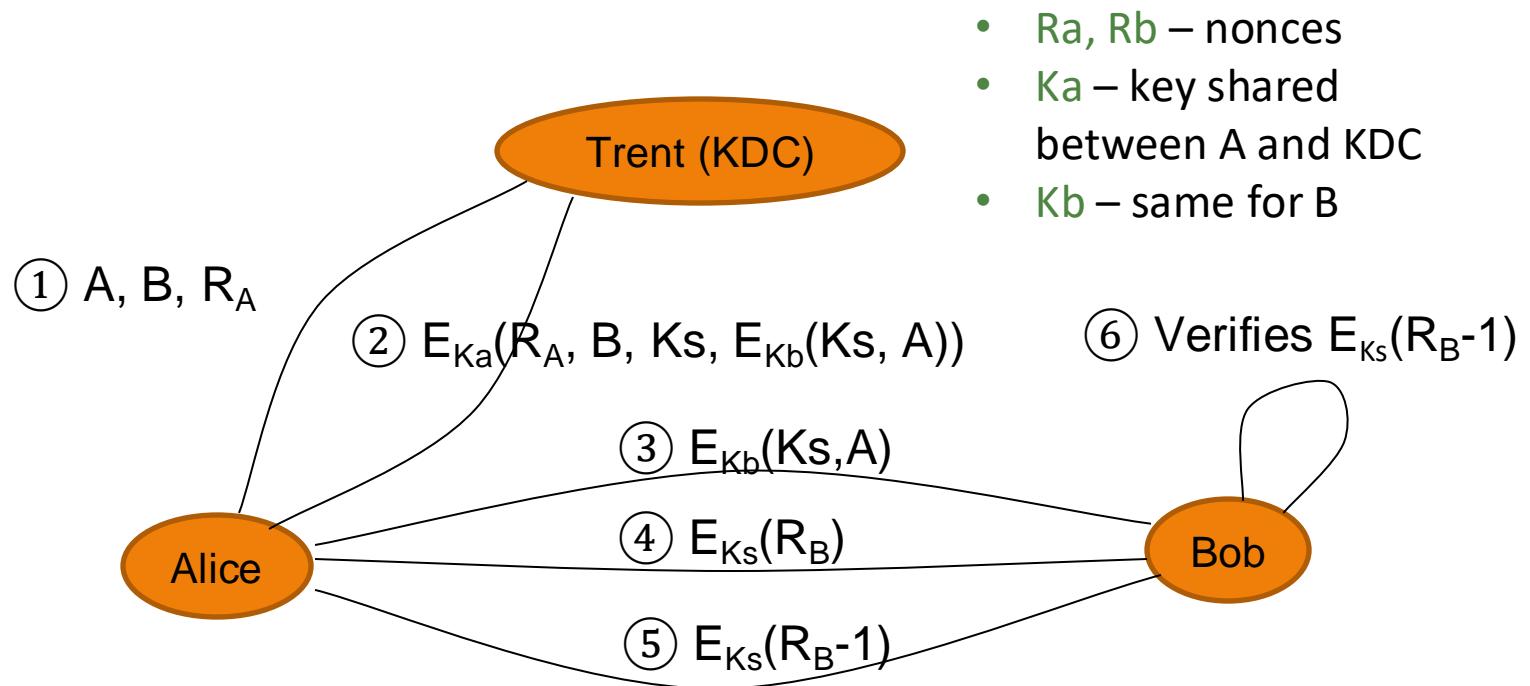
Replay attacks (2/3)

- Timestamps
 - Messages contain a timestamp
 - Messages are only accepted if their timestamps are within a given timeframe
- Frequently used (e.g., in Kerberos), however, problems exist:
 - Clock must be synchronized
 - Tolerance to network delay

Replay attacks (3/3)

- Challenge/Response
 - The communication initiator sends a nonce (number used only once)
 - and waits for that nonce (or its transformation) to come in the reply
- Easy to implement but:
 - More messages are required
 - Needs for both parties to be active
 - Not applicable to communications without a connection

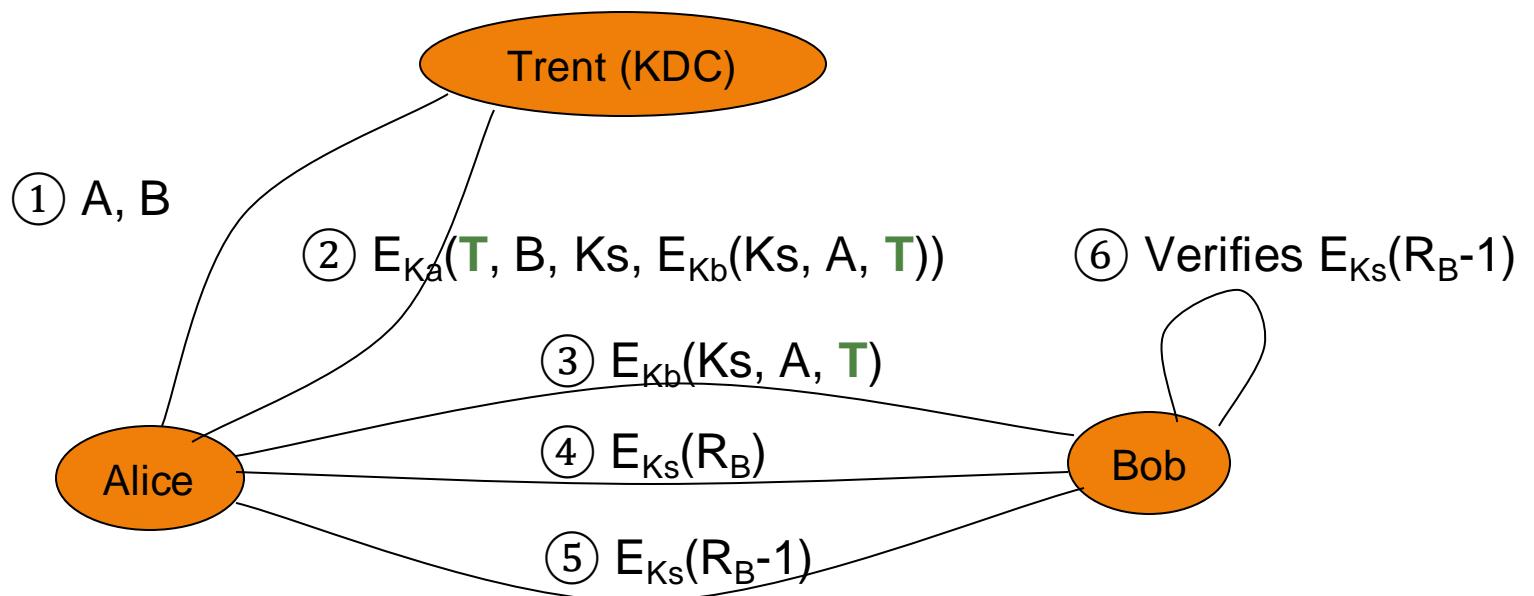
Needham-Schroeder (NS)



- Can message ③ be sent directly by Trent to Bob?
- What are the messages ④ and ⑤ used for?
- What happens if someone can obtain/discover a session key?

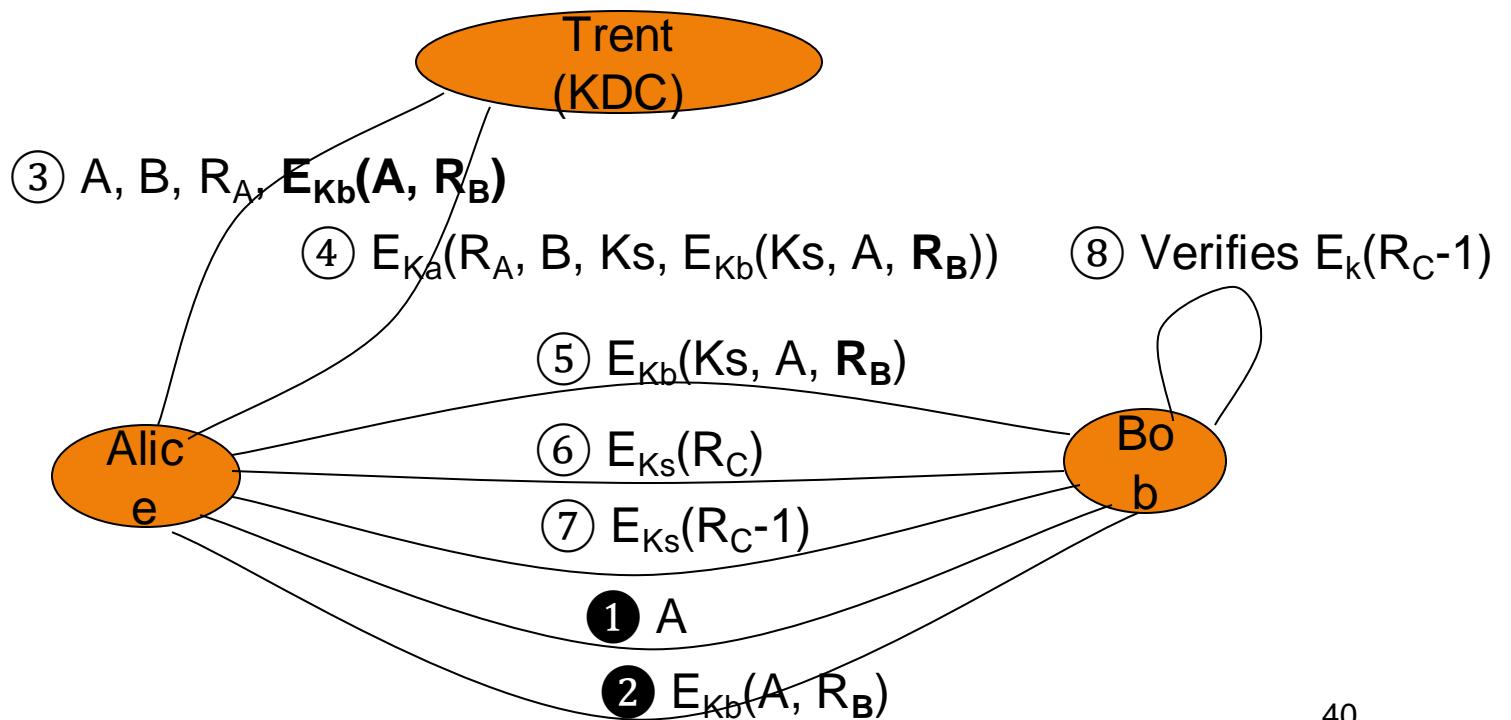
NS with timestamps

- Modification proposed by Dorothy Denning
- Bob accepts message ③ only if it comes within the timeframe
- The time interval to share the session key is limited; no nonces
- Needs clock synchronization



NS revisited

- Modification proposed by Needham & Schroeder
 - Uses nonces to validate the **freshness** of connection request from A
- Does not need clock synchronization

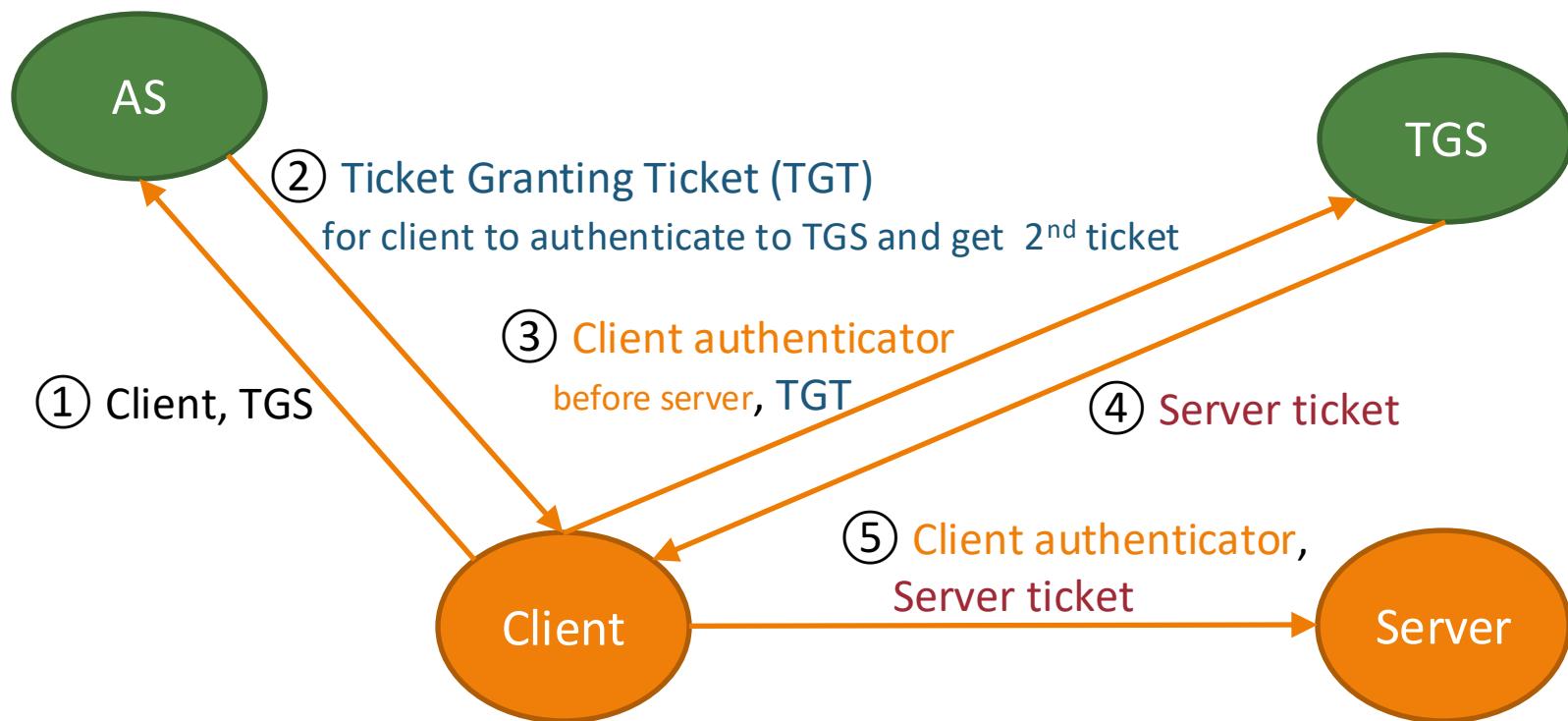


Kerberos

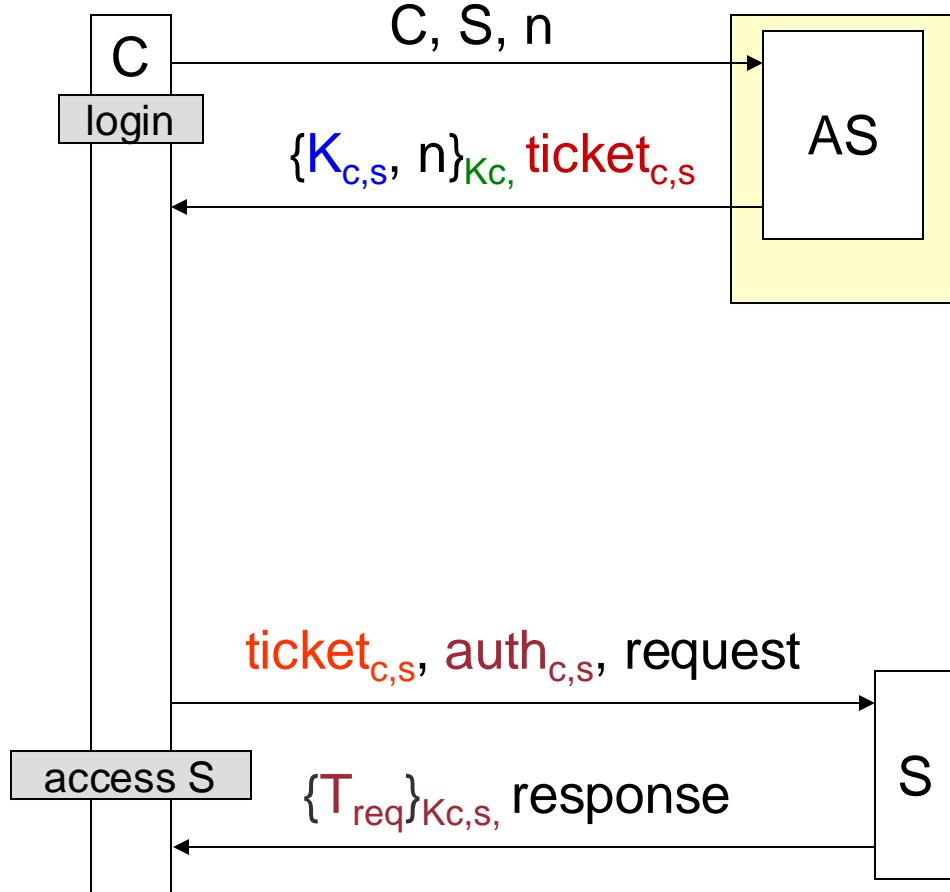


- Based on Needham-Schroeder with timestamps
 - Solves **problem of NS** requiring using K_a for every contact with KDC
 - ... and K_a typically obtained from a password provided by the user
- Ticket Granting Service (TGS)
 - Provides time-limited credentials (tickets) for several services/servers
- Authentication Service (AS)
 - Allows client to login in Kerberos
 - Each client has a shared key with AS derived from password
- Kerberos operates in **organizational realms / security domains**
 - Multi-realms possible if realms cooperate
- Communication over TCP/IP

Kerberos overview



Kerberos (AS only)



	C	S	AS
C			
S		$K_{c,s}$	
AS	K_c	K_s	

$$\text{ticket}_{x,y} = \{x, y, T_1, T_2, K_{x,y}\}_{K_y}$$

$$\text{auth}_{x,y} = \{x, T_{req}\}_{K_{x,y}}$$

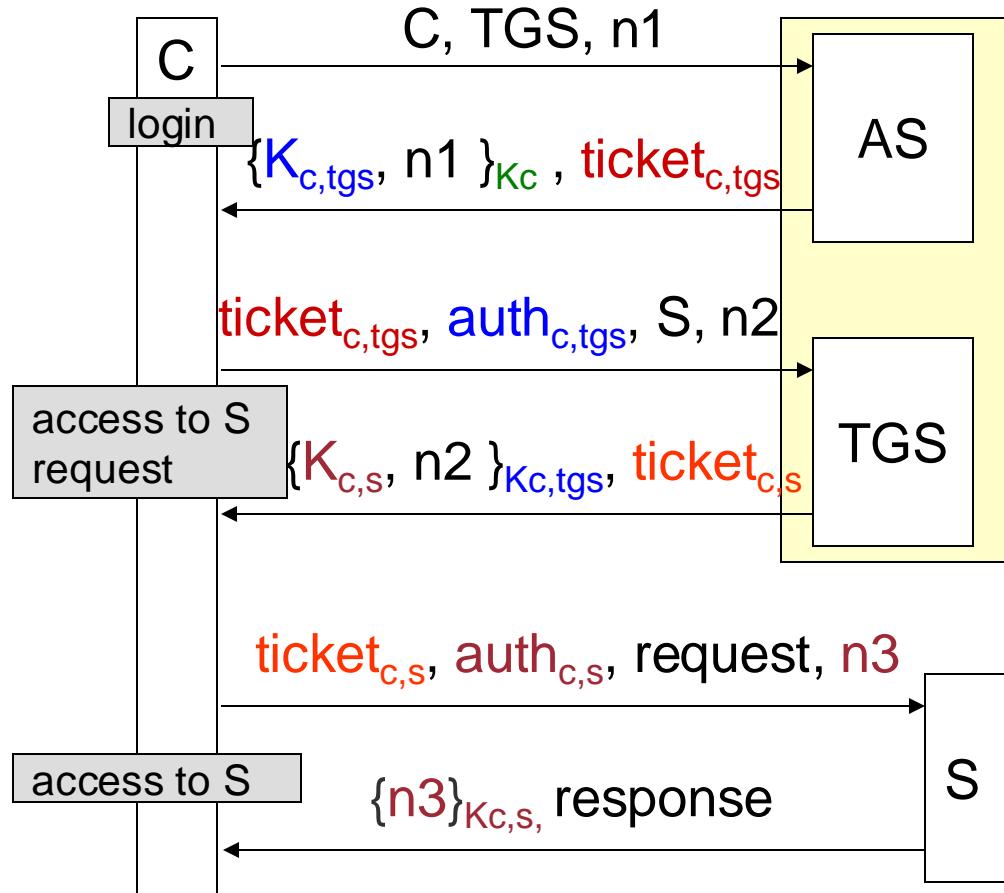
What is inside a ticket?

- $\text{ticket}_{x,y} = \{ x, y, T_1, T_2, K_{x,y} \}_{K_y}$
 - X – client identifier
 - Y – server identifier
 - Timestamps
 - T_1 – beginning of validity period
 - T_2 – end of validity period
 - To avoid reuse of old tickets
(implies clock synchronization)
 - $K_{x,y}$ – session key value
 - Information ciphered with server key

What is inside an authenticator?

- $\text{auth}_{x,y} = \{ x, T_{\text{req}} \}_{Kx,y}$
 - X – client identifier
 - T_{req} – timestamp of request
 - To avoid resending of old request
(also implies clock synchronization)
 - Information ciphered with session key

Kerberos V5



	C	S	TGS	AS
C				
S		$K_{c,s}$		
TGS	$K_{c,tgs}$	K_s		
AS	K_c		K_{tgs}	

$$\text{ticket}_{x,y} = \{x, y, T_1, T_2, K_{x,y}\}_{K_y}$$

$$\text{auth}_{x,y} = \{x, T_{\text{req}}\}_{K_{x,y}}$$

Why separate AS and TGS?

- Separate keys of users from keys of services
- Separate authentication function (AS) from authorization function (TGS)
 - Distribute load between servers
- Minimize use of Kc
 - Used only on login
- Allow composition of TGS servers
 - To access other realms

Roadmap

- Introduction
- Generation and manual distribution
- Distribution with shared values
- Distribution without shared values
- Distribution with third parties
- **Key renewal**

Renewal of keys

- Renewal methods
 - Using KEK keys to distribute new session keys
 - Using trusted third parties
 - Example: distribution of keys in Kerberos
- Perfect Forward Secrecy
 - Key renewal *per se* does not assure Perfect Forward Secrecy
 - Although it can, if Diffie-Hellman is used with *ephemeral* private values

Summary

- Introduction
- Generation and manual distribution
- Distribution with shared values
- Distribution without shared values
- Distribution with third parties
- Key renewal

Authentication

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves
Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication

Roadmap

- **Authentication**
- Authentication with passwords
- Biometric authentication

Authentication: Problems

- Services must only be used by authorized users
 - To do **authorization**, first services need to **authenticate the users**
- Authentication must be trustworthy
 - It must be proved that the connecting party is **who it claims to be**
- Authentication must be secure
 - The authentication protocol must **not** be used by a third party to **impersonate** legitimate users
- Authentication should be simple to use
 - So that people are able to deal with it without great **cost**
- Users can be careless
 - And choose poor passwords, for example

Authentication protocols: Goals

- Entity authentication
 - People, services, servers, machines, etc.
- Facilitate protocol usage
 - Provide mechanisms to simplify use of personal secrets (keys, passwords)
 - Interconnection with key distribution protocols
- Assure protocol correctness/accuracy in hostile environments
 - Accuracy in the proof of authentication
 - Confidentiality of the used secrets as proof of authentication
- Prevent impersonation from attackers
 - Prevent attacks to the messages used in the on-line protocols
 - Prevent off-line attacks with dictionaries

Machine Authentication

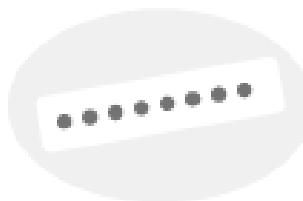
- **Bad option: by name or address**
 - Name, DNS, IP, MAC, or other
 - Unreliable, but used by several old services
 - e.g., NFS, TCP wrappers
 - Some validations are possible
 - e.g., translation DNS → IP → DNS
- **With secrets (keys)**
 - Shared key between machine pairs
 - Asymmetric key pair per machine
 - Methods used in:
 - Secure network protocols, e.g., IPSEC
 - Secure communication protocol that interact with a daemon representing a machine, for tunneling applications' communication, e.g., SSH

Services Authentication

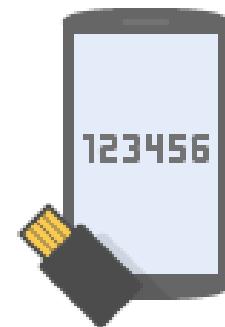
- **Machine authentication**
 - Allows to univocally authenticate a set of services running in that machine
- **Service authentication** uses similar approaches to machine authentication but only for a specific service running on the machine
 - e.g., Web Server
- Shared secret key
 - Sharing a long-term secret with the user
 - e.g., remote login with password
- Asymmetric key pair per service
 - Allows to differentiate services in a more defined way
 - e.g., HTTPS servers with private key corresponding to public key in digital certificate

Personal Authentication

Using something that
you know



Using something that
you have



Using something that
you are

Personal authentication methods

- Using something that **you know**
 - Typically, a shared secret
 - Password, PIN, etc.
- Using something that **you have**
 - Magnetic card, Smart Card, key generators, etc.
- Using a **physical characteristic**
 - Evaluation of a static unique personal feature:
 - Fingerprint, iris, retina, facial features, voice, etc.
- Using a **behavioral characteristic**
 - Evaluating unique dynamic features of a person
 - Writing patterns on a keyboard
 - Patterns of a signature

Biometric features

Multi-Factor Authentication

- Use of several authentication methods combined
 - Designated **Double Factor Authentication** or **2FA** when two methods are used
- Can assign different methods to different tasks
 - As users perform more and more sensitive tasks, must authenticate in more and more ways (presumably, more stringently)
 - Example in home banking: password, 3 values from matrix, code sent by SMS

Resolução do Conselho de Ministros n.º 41/2018

Diário da República, 1.ª série—N.º 62—28 de março de 2018

Recomenda-se que para novos sistemas seja sempre usado como padrão de autenticação o 2FA.

Roadmap

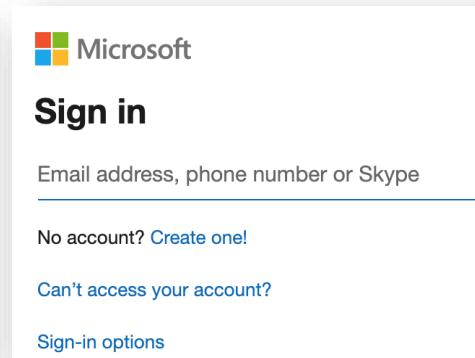
- Authentication
- **Authentication with passwords**
- Biometric authentication

Usernames/passwords

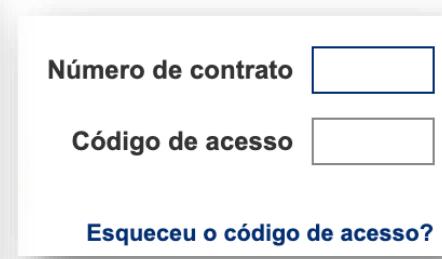
- Usernames/passwords can take different forms today
 - Email address
 - Phone number



The Google sign-in form features the Google logo at the top left. Below it is the text "Iniciar sessão". A "Continuar para o Gmail" link is present. A large input field is labeled "Email ou telemóvel" and contains a placeholder "Email ou telemóvel". At the bottom, a blue link says "Esqueceu-se do email?".



The Microsoft sign-in form starts with the Microsoft logo. It has a prominent "Sign in" button. Below it is a text input field labeled "Email address, phone number or Skype". Underneath the input field are links for "No account? Create one!" and "Can't access your account?". At the bottom, there is a "Sign-in options" link.



This is a custom sign-in form with fields for "Número de contrato" (with a blue outline) and "Código de acesso" (with a grey outline). At the bottom, a blue link reads "Esqueceu o código de acesso?".

Authentication of people: with memorized passwords

- Operation
 - The user memorizes a **password** and can provide it when prompted
 - Password **validation** relies on a **secret** stored on behalf of the user
 - The password value should never be stored
 - Instead, the personal secret is protected by being transformed by a **one-way function**, for example, using a **cryptographic hash function** or a **key derivation function** (*next*)
- Advantages of passwords
 - Simplicity
 - Can be used in any device with text input
- Problems of passwords
 - Poor selection by users
 - Remote transmission through unsecure channels

Key derivation function (KDF)

- KDF transform a **variable-length text password** into a **fixed-length binary cryptographic key**
 - String -> byte[]
 - KDFs often use **non-secret parameter** – called **salt** – in addition to the **secret password** to produce diverse hash values across machines (*more on this later*)
 - KDF may ensure that derived keys have other desirable properties
 - e.g., avoiding “weak keys” in some specific encryption systems
- Common uses of KDFs are:
 - **Password hashing** – the one we are interested here
 - **Generating secret keys from passwords**
e.g., to cipher documents

Password attacks

- Mislead a user to reveal password
 - Pretexting, baiting, phishing and other “social engineering” attacks
- Try to guess the password
 - Brute-force – systematically test all possible combinations
 - Dictionary attack – test likely values first
 - Values that are easy for humans to memorize, like words in dictionary, dates, etc.



Top 50 breached passwords

#	Windows Password	Number of Times Detected	#	Windows Password	Number of Times Detected
1	123456	23,174,662	26	myspace1	735,980
2	123456789	7,671,364	27	121212	732,832
3	qwerty	3,810,555	28	homelesspa	727,480
4	password	3,645,804	29	123qwe	711,669
5	111111	3,093,220	30	a123456	679,353
6	12345678	2,889,079	31	123abc	637,906
7	abc123	2,834,058	32	1q2w3e4r	631,071
8	1234567	2,484,157	33	qwe123	630,653
9	password1	2,401,761	34	7777777	623,994
10	12345	2,333,232	35	qwerty123	592,110
11	1234567890	2,224,432	36	target123	587,949
12	123123	2,194,818	37	tinkle	585,933
13	000000	1,942,768	38	987654321	585,426
14	iloveyou	1,593,388	39	qwerty1	581,151
15	1234	1,256,907	40	222222	579,444
16	1q2w3e4r5t	1,141,300	41	zxcvbnm	575,310
17	qwertyuiop	1,081,655	42	1g2w3e4r	573,735
18	123	1,023,001	43	gwerty	573,292
19	monkey	980,209	44	zag12wsx	572,800
20	dragon	968,625	45	gwerty123	572,625
21	123456a	968,369	46	555555	551,013
22	654321	932,752	47	fu█you	549,663
23	123321	911,514	48	112233	534,650
24	666666	876,983	49	asdfghjkl	527,764
25	1qaz2wsx	756,613	50	1q2w3e	498,741

Source: <https://haveibeenpwned.com/Passwords>

Mitigations

- Against “social engineering”
 - Provide user education and training
- Against brute-force
 - Increase number of combinations
 - Ask for longer passwords, e.g., minimum 10 characters
 - Force use of diverse characters: letters, capital letters, numbers, symbols
 - Refresh passwords periodically
 - E.g., every 6 months
 - Make sure new password is very different from old password
- Against dictionary attacks
 - Reject popular/trivial passwords
 - And variations with prefixes, suffixes, replacements of letters by numbers, and other ways that users try to use weak passwords

Online password attacks

- The attacker tries to guess the password, **one attempt at a time**, through the user interface
- Mitigation: limit the number of attempts
 - **Throttling**, e.g., limit to N attempts (per minute)
 - “*Human verification*” with challenges like CAPTCHA

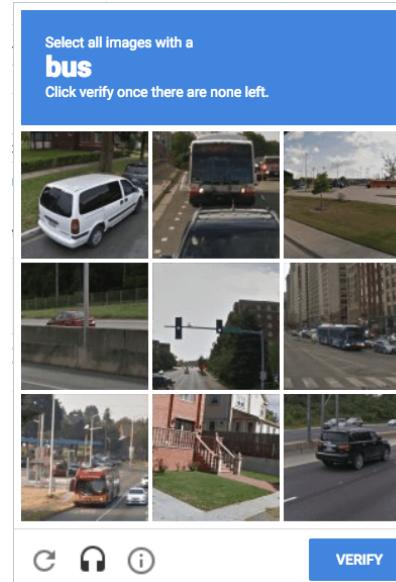
Match the characters in the picture Help

To continue, type the characters you see in the picture. [Why?](#)

The picture contains 8 characters.

Characters:

Continue



Offline password attacks

- The attacker exfiltrated the **password file** and can try **many guesses** rapidly
 - Throttling and CAPTCHA are circumvented
- Mitigation: store a **transformation** of the password, not its actual value
 - Use of cryptographic hash
 - Use of salt
 - Key stretching / strengthening



Salt



- **Salt** – random data used as an additional input to a one-way function that hashes a password
 - Useful if attacker gains access to the saved password hash because they help defending from attacks that use:
- **Rainbow tables** (precomputed hash databases)
 - Tables that containing passwords and their hashes
 - Checking the hashes is fast: compare stolen hashes with the table
 - Without the need of calculating the hashes
 - Rainbow tables are expensive to compute
 - Without salt, attacker can build table once, then use it always
 - MD5 of "password" is always
5f4dcc3b5aa765d61d8327deb882cf99
 - SHA1 of "password" is always
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
 - With a salt, attacker has to compute a rainbow table for each salt value

Key stretching / strengthening

- **Derived key = KDF(Key, Salt, Iterations)**
- Parameters:
 - **Key** – original key or **password** (password in the present case)
 - **Salt** – random non-secret number
 - **Iterations** – number of executions of a sub-function
 - The difficulty of a brute force attack increases with the number of iterations
 - Practical limit is the unwillingness of users to tolerate perceptible delay
- Modern password-based KDF: **PBKDF2** (RFC 2898)
 - Uses cryptographic hash function (e.g., SHA-2), salt (e.g., 64 bits or more) and a high iteration count (often tens or hundreds of thousands)
 - NIST requires at least 128 bits of random salt and a NIST-approved cryptographic function, such as the SHA or AES
- 2013: Password Hashing Competition
 - Choose a new standard algorithm for password hashing: Argon2

Historical examples: Unix Crypt

- **Passwords** with 8 7-bit ASCII characters
 - Key = $8 \times 7 = 56$ bits (password)
- 25 **iterations** of modified DES algorithm: **DES_{Key(0)}**
 - Modified DES to avoid hardware attacks that existed at the time
- **Salt** of 12 bits
 - Additional permutation in DES' interactions depending on the salt
 - Stored in cleartext next to the password
- **Encoding** as printable string to be stored in the /etc/passwd file
 - 64 bits resulting from cipher of “0” + 2 bits = 66 bits (2 bits for padding)
 - Encoded as 11 base64 digits

too small today

Historical examples: Windows NT

- Two hashes for compatibility reasons: NT and LanMan
- NT hash **NT-Hash** = MD4(pwd) (MD4 is an old hash function; insecure)
 - Weak: no salt, no iterations
- LanMan (LM) hash
 - PWD=CutOrPad(Uppercase(password), 14) – 14 characters
 - L = Left(PWD); R = Right(PWD) – 7 characters each
 - **LM-Hash** = DES_L("KGS!@#\$%") | DES_R("KGS!@#\$%") – 16-byte hash
 - Optionally **Hash** = DES_{ID}(Hash)
- NTv2 and LMv2 (since Windows NT 4.0 SP4, 1998)
 - SC = 8-byte server challenge, random
 - CC = 8-byte client challenge, random
 - CC* = (X, time, CC, domain name) (X is a constant)
 - v2-Hash = HMAC-MD5(NT-Hash, user name, domain name)
 - **LMv2** = HMAC-MD5(v2-Hash, SC, CC)
 - **NTv2** = HMAC-MD5(v2-Hash, SC, CC*)

LanMan Security Weaknesses

- Passwords are limited to a maximum of only 14 characters
 - Theoretical maximum keyspace of $95^{14} \approx 2^{92}$
 - But passwords longer than 7 characters are divided into two pieces and each piece is hashed separately: $95^7 \approx 2^{46}$
 - Password lowercase letters are changed into uppercase, which further reduces the key space for each half: $69^7 \approx 2^{43}$
 - In addition, any password that is shorter than 8 characters will result in the hashing of 7 null bytes ($L = 0$), yielding a constant value
- LM hash does not use salt
- Ophcrack tool (2003) had rainbow table that targets the weaknesses of LM
 - Can crack all alphanumeric LM hashes in a few seconds

Roadmap

- Authentication
- Authentication with passwords
 - One-time passwords
- Biometric authentication

Authentication of people: one-time passwords (OTP)

- Operation
 - Validation of a **one-time password** (i.e., single use) regarding a secret stored for the user
 - A one-time password can only be used once
 - Therefore, it is pointless for an attacker to capture it
 - Which one-time password to use can be known by several means
 - Challenge-response
 - Synchronization
 - Generation every 30 to 60 seconds
 - Requiring a synchronization server
 - Typically, also has a secret

Assessment of one-time passwords

- **Advantages**
 - Security in remote authentications over insecure networks
 - Security if used in insecure terminal (e.g., with a key logger running)
- **Problems**
 - In some implementations the users need to use some device/application to generate the one-time passwords
 - Dedicated equipment (smartcard, etc.)
 - Computation algorithms and functions
 - Not much appreciated by the users
 - Memorization of one-time passwords by the attackers
 - Off-line attacks with dictionaries using collected keys
 - Dictionary attacks can be prevented if non-textual seed-passwords are used

Roadmap

- Authentication
- Authentication with passwords
 - Challenge-response
- Biometric authentication

Authentication of people: challenge-response

- Operation
 - **Authenticator** provides a value: **challenge**
 - **User** being authenticated transforms the challenge using the **secret** that he shares with the authenticator
 - **User** sends the result to the authenticator: **response**
 - **Authenticator** validates the response / transformation
- Advantages
 - Security for authentication over unsecure networks
- Problems
 - The authenticator needs to have access to shared secrets with all users
 - Attackers may store the challenge-response pairs exchanged
 - Offline attacks with dictionaries to the collected pairs

Roadmap

- Authentication
- Authentication with passwords
- **Biometric authentication**

Authentication of people: biometric authentication

- Operation
 - User validated through biometrical data: fingerprint, physiognomy,...
 - This info is compared with records for each user recognized by the system
- Advantages
 - Solves the problem of choosing a good password
 - Eliminates the need for the users' memory
- Problems
 - Sometimes can be deceived
 - Do not allow the transference of authentication between subjects
 - Do not support change of authentication data (it is fixed)
 - Complicates secure remote authentication

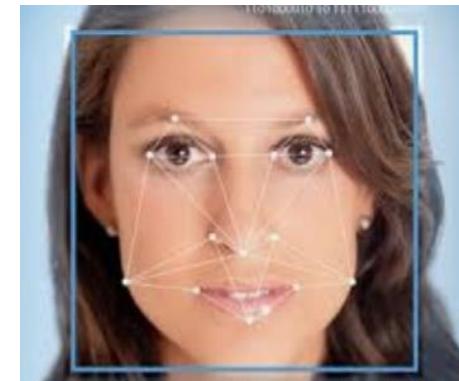
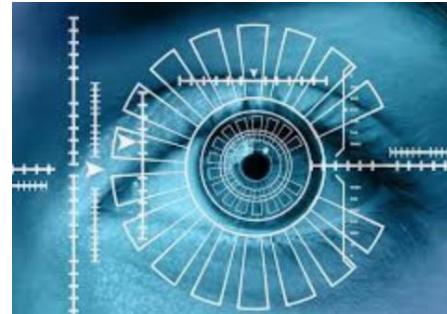
Identification vs. Authentication

- **Identification**
 - Search for an entity of a community given an identifying feature
 - Example: searching for person in a criminal record given his fingerprint
 - *Open search*
- **Authentication** (or verification)
 - Verifying that an entity is who it claims to be (the id is given)
 - Example: verify in an access control device if the fingerprint presented by a person matches the one registered for that same person
 - *Closed search*



Examples of biometric technologies

- Fingerprints
- Iris recognition
- Retina scan
- Facial recognition
- Palm vein biometrics
- Voice recognition



Working principle of biometric authentication

- **Registration**
 - Acquisition of information about the person
 - Biometric data and other information
 - Construction of the validation information: the **template**
 - Typically from the template it is not possible to reconstruct the biometric data
- **Authentication**
 - By providing the biometric information along with the claimed identity
 - Checking the obtained biometric data with the stored template
- Authentication in this context is unidirectional
 - A person proves to be who he claims
 - The authentication system does not prove its authenticity
 - It either accepts the individual authentication or not

Biometric authentication: desirable properties

- Universality
 - Ability to be applied to all individuals
- Unicity
 - Ability to distinguish all the individuals
- Stability
 - Ability to operate continuously without problems during lifetime of the individuals
- Correctness
 - Ability to acquire and use validation data capable of distinguishing all individuals
 - related to Unicity, but Correctness is “practical”, Unicity is “theoretical”
- Convenience
 - Ability not to cause discomfort or repulsion
- Acceptance
 - Ability not to cause rejection due to loss of privacy or ethical-social issues

Exercise: how do fingerprinting and face recognition stand in each property?

Biometric authentication: Advantages & disadvantages (1/2)

- Biometric characteristics cannot be lost, forgotten, or lost
 - But also cannot be modified if needed
 - Biometric authentication may abuse **privacy** principles
- Biometric authentication requires physical presence
 - This complicates certain actions:
 - Delegation of responsibilities
 - Remote authentication
 - May cause discomfort or rejection
 - e.g. exposing the retina to lasers
- Dictionary attacks are not possible
 - But typically a shared secret key cannot be derived from the authenticator – validation is **probabilistic**

Biometric authentication: Advantages & disadvantages (2/2)

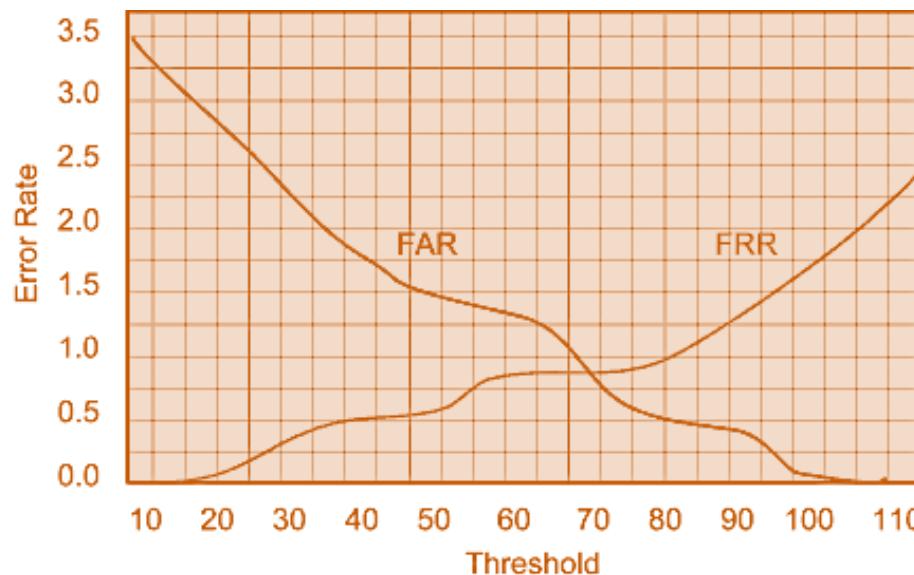
- The likelihood of 2 individuals with the same biometric features existing is low
 - But the biometric **acquisition mechanisms** and the **template matching** processes may not be able to distinguish the two
- Allow for maintenance cost reduction in password-based systems
 - However, if they fail, alternative authentication mechanisms are required
 - Typically passwords
 - Which is even worst:
 - The initial goal of eliminating the need for password is not achieved!
 - The system becomes even more complex
 - People must remember passwords that are rarely used

Technical aspects: Correction

- Making no errors; 2 types of errors:
 - **False Acceptance / False Positive (FP)**
 - Accepting incorrect biometric data
 - **False Rejection / False Negative (FN)**
 - Not accepting true biometric data
- **Accuracy:** $(TP + TN) / (TP + TN + FP + FN)$
 - The greater the accuracy the greater the distinction between individuals
 - Can be critical for identification
 - Typically, not so critical for authentication
- Error rate / accuracy adjustment
 - Allows to select acceptable error rates considering the operational environment

Adjustment of error rates

- The reduction of one error rate increases the other
 - The reduction of the **False Rejection Rate (FRR)** is associated to an increase of the **False Acceptance Rate (FAR)** and vice-versa
- The error rate depends on the stored validation information (template)



Authentication protocols

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos
Ribeiro

Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication
- **Authentication protocols**

Examples of authentication protocols

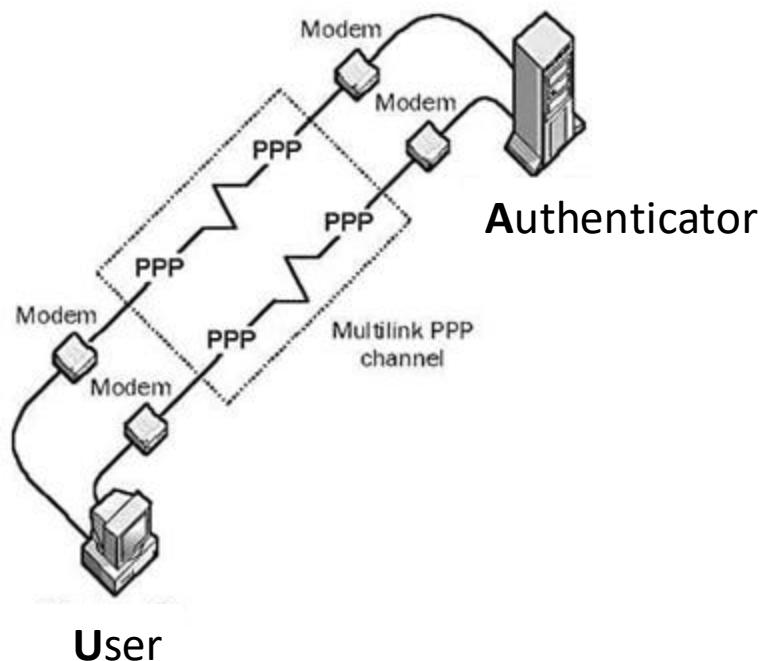
- With **passwords**
 - PAP, CHAP, MS-CHAP, Kerberos
- With **one-time passwords**
 - S/Key, SecurID
- With **asymmetric keys**
 - SSL, SSH, PGP

Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication
- **Authentication protocols**
 - With passwords

PAP and CHAP

- Both protocols are used by PPP (Point-to-Point Protocol)
 - Secret shared between **authenticator (A)** and **user (U)** authenticated
 - Authentication is unidirectional (A authenticates U)



PAP

- PPP Authentication Protocol
 - Simple exchange of a pair UID/password
 - U ? A: username, password
 - A ? : OK/not OK
 - Insecure: the password is sent as is, in plaintext

CHAP

- Challenge Handshake Authentication Protocol

U ↗ A: username

A ↗ U: authID, challenge (authID: identifier for this attempt)

U ↗ A: Hash(authID, password, challenge)

A ↗ : OK/not OK

- Authenticator may request authentication of the user at any moment

- *Problem with CHAP: A stores the passwords;
solution: next slide*

MS-CHAP (Microsoft CHAP)

- **MS-CHAPv1** (RFC 2433)

U ↗ A: username

A ↗ U: authID, C

U ↗ A: R

A ↗ U: OK/not OK

- C – challenge

- $R = DES_{PH}(C)$ – cipher C with password hash PH

- PH = NT-Hash or LM-Hash
 - NT-Hash = MD4(password)
 - LM-Hash based on DES
 - See “Windows Password Authentication” before

- **MS-CHAPv2** (RFC 2759)

U ↗ A: username

A ↗ U: authID, C_A

U ↗ A: $C_U, R1$

A ↗ U: OK/not OK, $R2$

- $C = SHA(C_U, C_A, \text{username})$

- $R1 = DES_{PH}(C)$

- PH = MD4(password)

- $R2 = SHA(SHA(MD4(PH), m1), C, m2)$

- **Mutual authentication**

- Authenticator shows knowledge of PH (in R2)

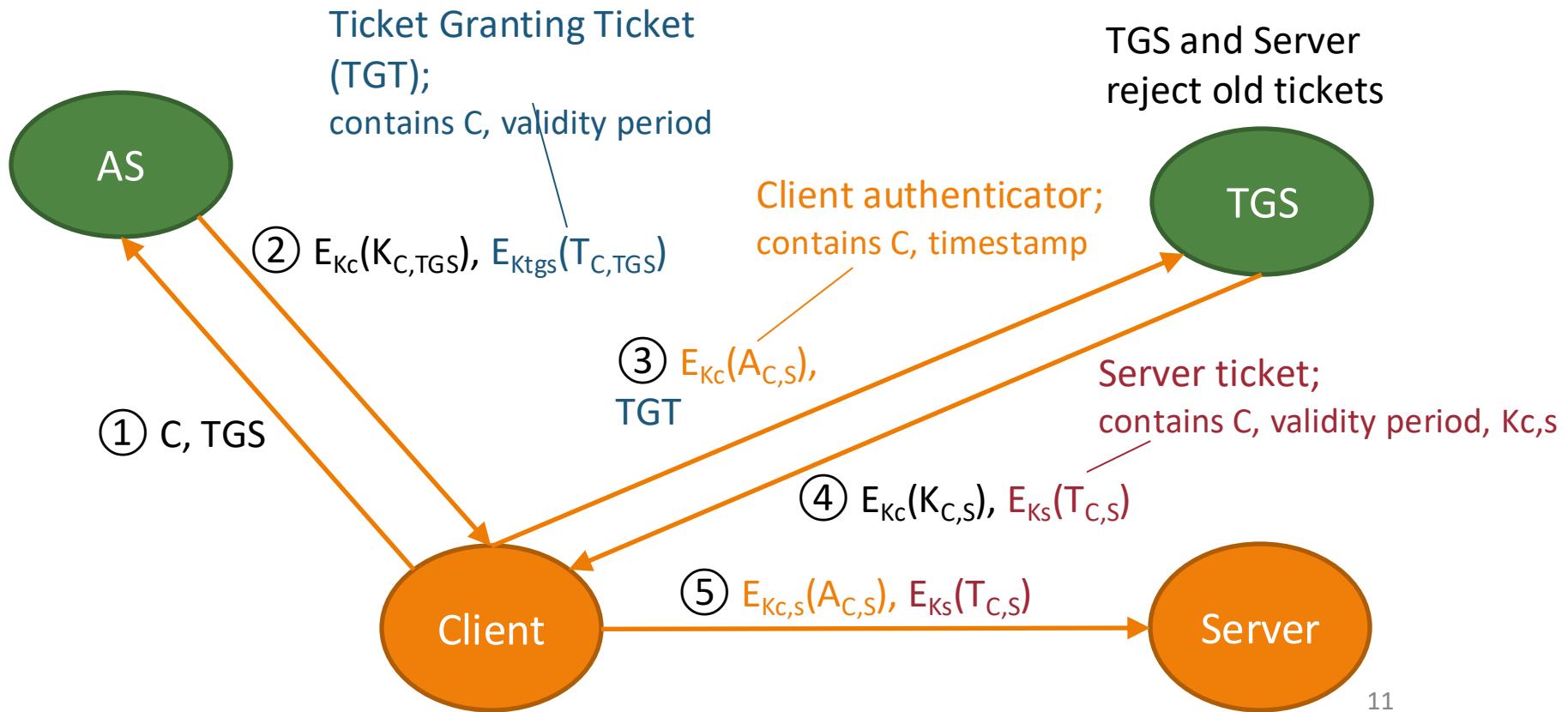
- Insecure LM-Hash is no longer allowed

A stores only hash of the password

Kerberos 5



- C, S, TGS – client, server, TGS identifiers
- Ktgs – TGS secret key, shared with AS
- **Kc – client secret key, shared with Kerberos**
- **Ks – server secret key, shared with Kerberos**
- Kc,s – session key, distributed to client and server by the protocol



Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication
- **Authentication protocols**
 - With one-time passwords

S/Key

- Authentication protocol with **one-time passwords**
 - Password used only once, so sent in plaintext
- **Authenticator** gives **user** a sequence of one-time passwords
$$\text{OTP}_1 = \text{Hash}(\text{seed}, \text{password}) \dots \text{OTP}_n = \text{Hash}(\text{OTP}_{n-1})$$
 - For each user, the authenticator **stores** only: **seed** of the one-time password sequence; current **index**; $\text{OPT}_{\text{index}}$ value
- **Authentication process:**
 - Authenticator sends **index** to the user
 - User sends $P = \text{OPT}_{\text{index}-1}$ to the authenticator (the one-time password)
 - Authenticator computes $\text{Hash}(P)$ and compares with stored $\text{OPT}_{\text{index}}$
 - If values are equal, then success, server stores **index-1** and $\text{OPT}_{\text{index-1}}$
 - The 1st time the authenticator sends **index = n**, next **index = n-1**, etc.

S/Key interaction

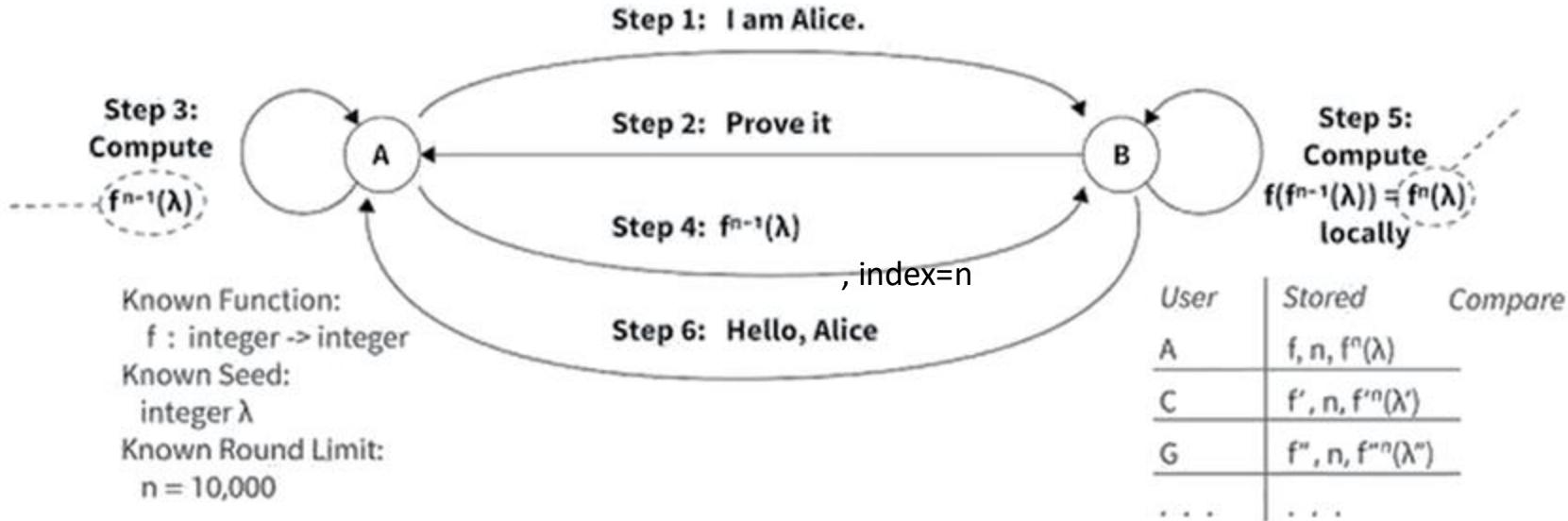


Image credits: Ed Amoroso, Tag Cyber

- Security:
 - Alice is the User, Bob is the Authenticator
 - All communication is in plaintext
 - Given $\text{OPT}_{\text{index-1}}$ it is not possible to get $\text{OPT}_{\text{index-2}}$ due to hash function
 - Easy to go in one direction but impossible in the opposite

RSA SecurID

- Personal authentication device from RSA Security (now Dell)
 - Also, in software for handheld devices and smartphones
- Generates a **unique number every minute**
 - They are essentially one-time passwords
 - They are computed using:
 - A **64-bit key** stored inside the card (shared with the RSA server)
 - The current date
 - A hash algorithm - **SecurID Hash**
 - Some versions allow inserting a PIN
- Authentication with a one-time password
 - The user generates a one-time password, combining a PIN with the card number
 - The RSA server performs the same operation and verifies if the values are equal
 - Server is synchronized with real time (RSA Security Time Synchronization)



Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication
- **Authentication protocols**
 - With asymmetric keys

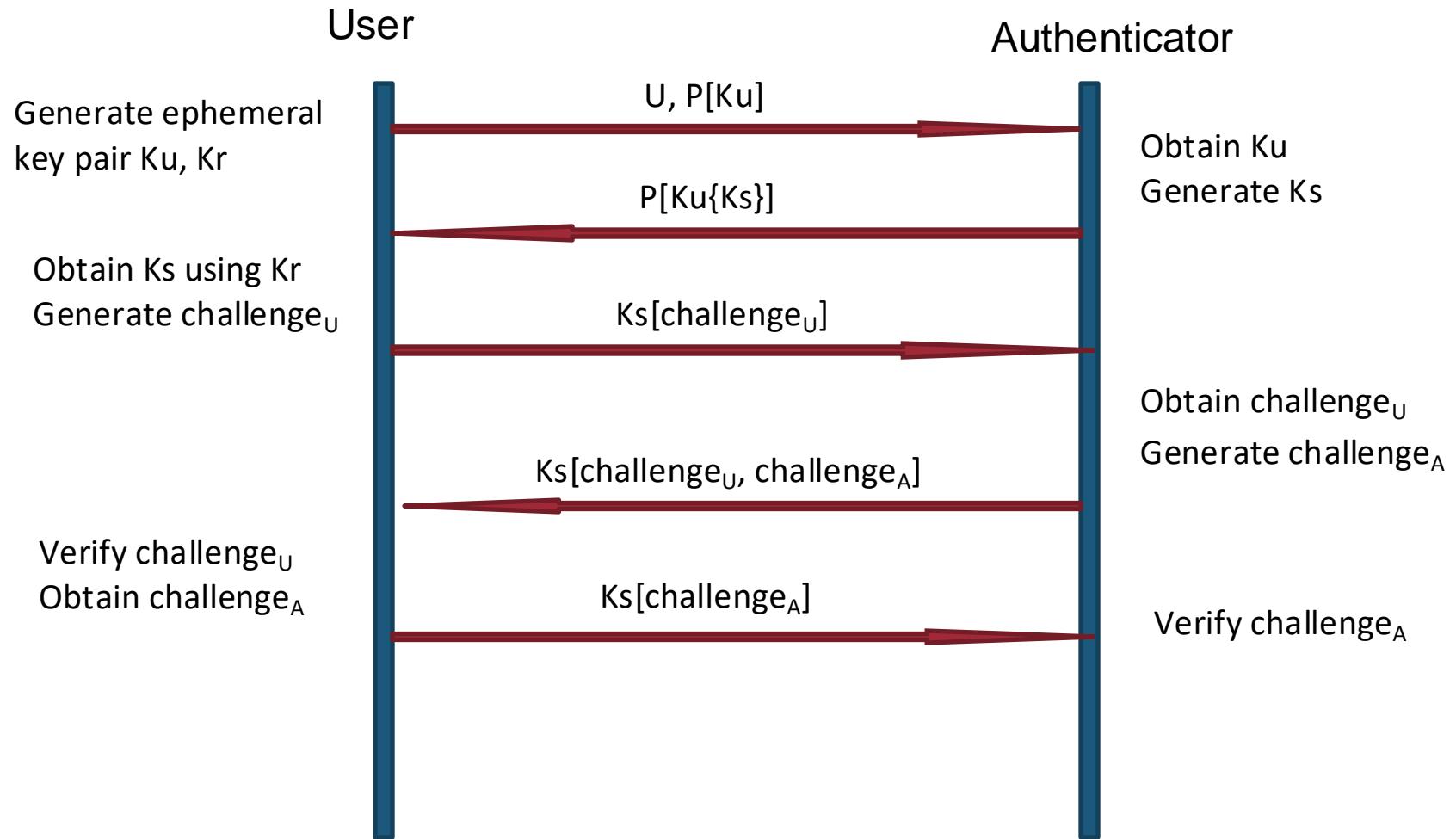
EKE (Encrypted Key Exchange)

- **Key agreement** –establishing a **session key (Ks)**
- **EKE** is a protocol that does
password-authenticated key agreement
 - Uses combination of asymmetric and symmetric ciphers
 - It is one of the few key password-authenticated agreement protocols that is **resistant to dictionary attacks**
- Can be used with several asymmetric techniques:
 - RSA, ECC, Diffie-Hellman
- Uses **ephemeral keys** – keys used only once

EKE notation

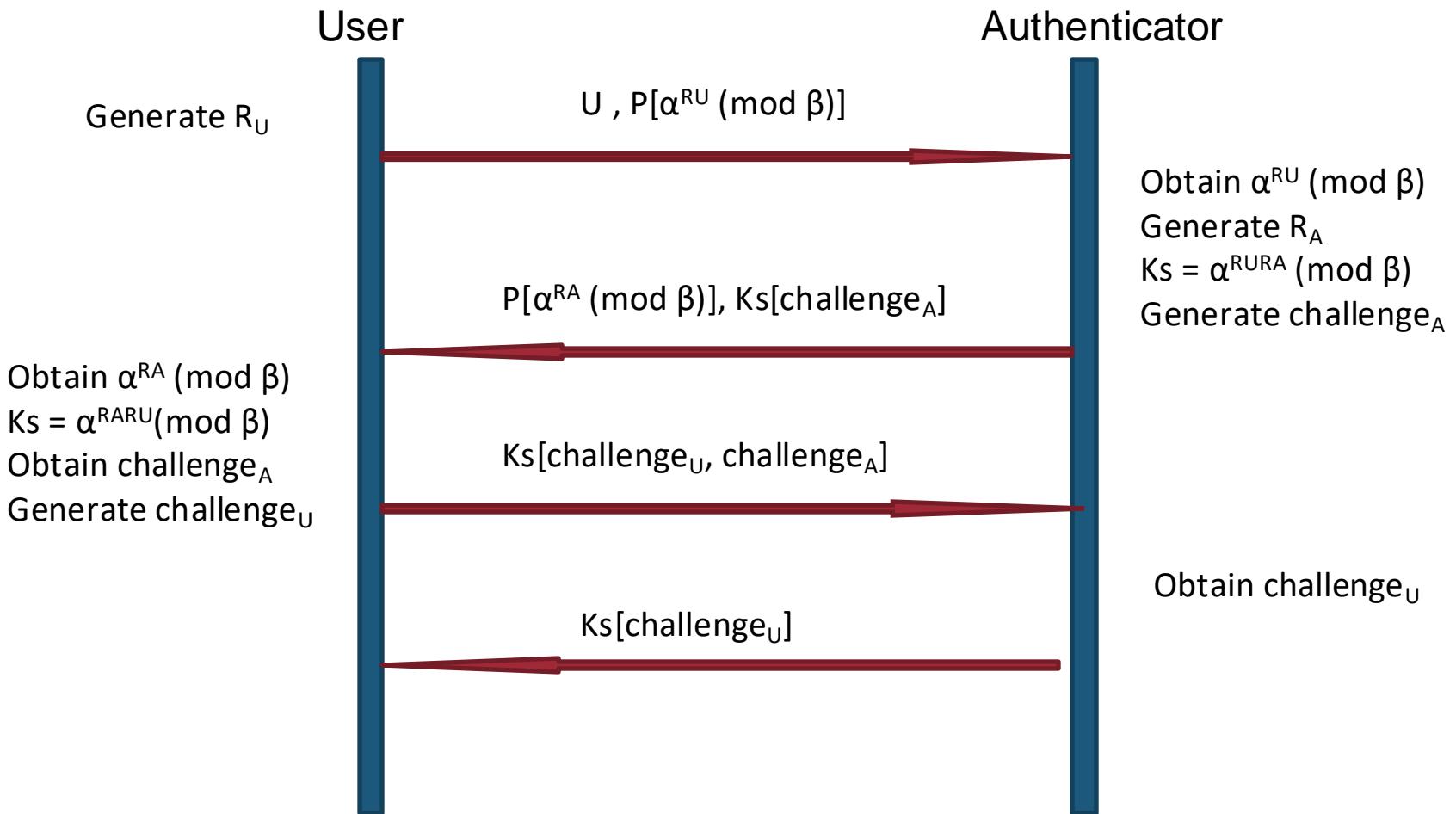
- General
 - U, A – User and Authenticator identifiers
 - P – Password, known by U and by A (!); used as a secret key
 - K_s – Session key established by the protocol; the result of the protocol
 - $RU, RA, challengeU, challengeA$ – values generated by the protocol
- Symmetric cipher
 - K – some secret key
 - $K[m]$ – Cipher m with the secret key K
 - $K^{-1}[m]$ – Decipher m with the secret key K
- Asymmetric cipher
 - K_u, K_r – public and private keys, respectively
 - $K_u\{m\}$ – Cipher m with the public key K_u
 - $K_r\{m\}$ – Decipher m with the private key K_r

EKE with RSA



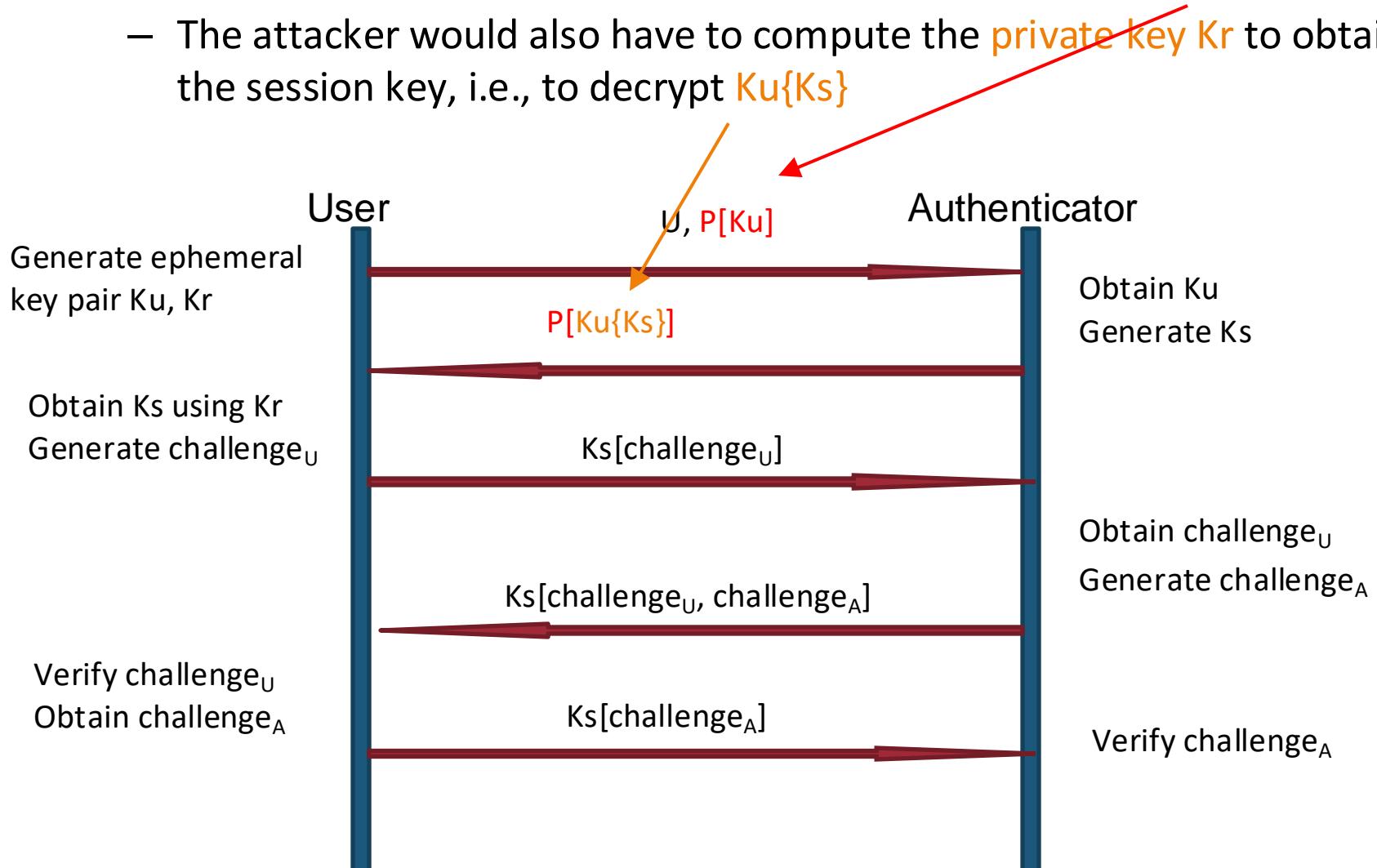
EKE with Diffie Hellman

DH-EKE is similar to EKE but with DH



EKE protection against dictionary attacks

- Even if successful decrypting the public key: $P'^{-1}[P[Ku]] = Ku'$
 - The attacker would also have to compute the **private key Kr** to obtain the session key, i.e., to decrypt $Ku\{Ks\}$

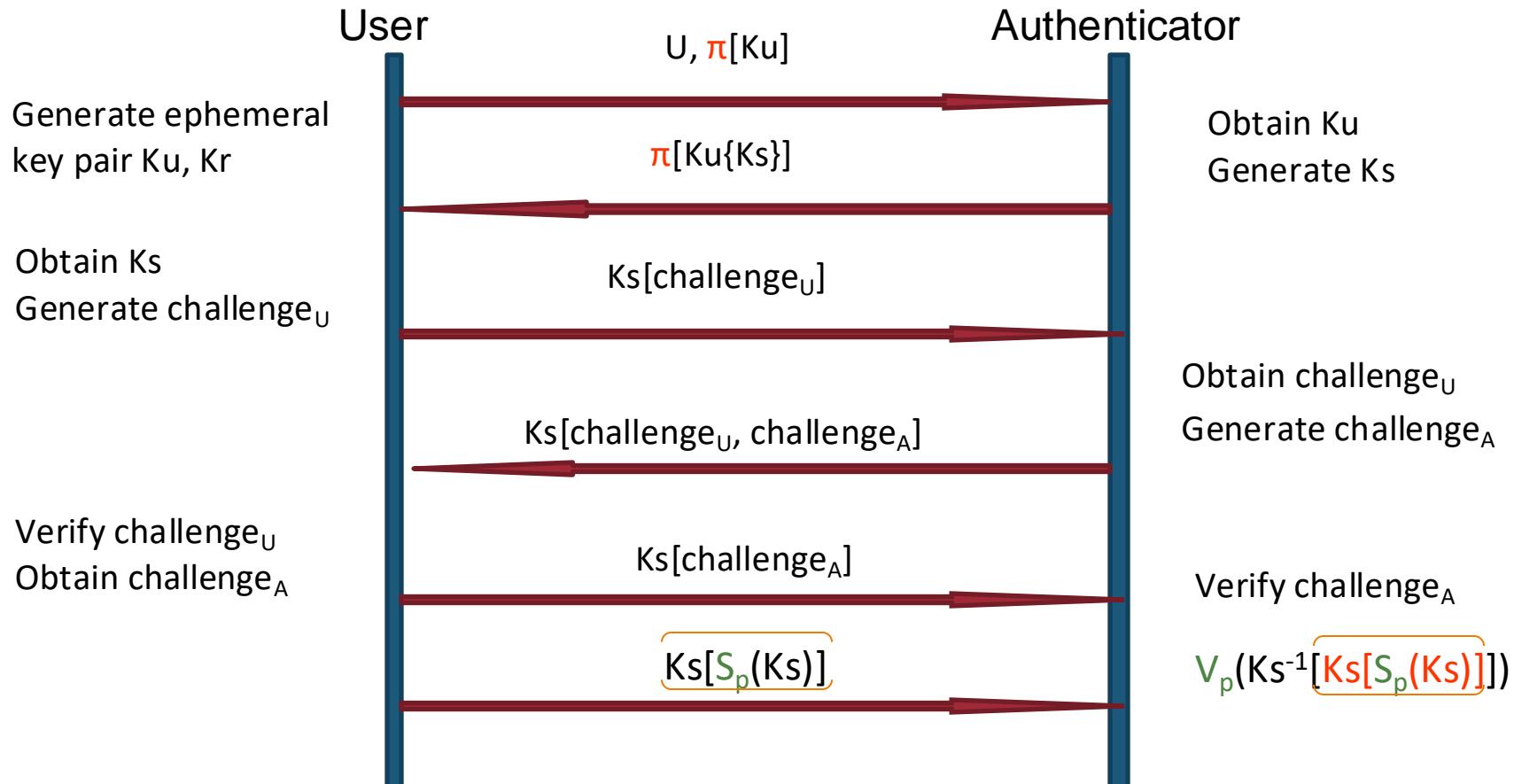


EKE security

- Man-in-the-middle protection
 - The session key K_s is only known by who generated it (Authenticator) and by who has the private key K_r (User)
- Replay protection
 - The last three messages validate the freshness of the key K_s
- Attack to the password database in the Authenticator would give access to P
 - Solution: **Augmented EKE** that only stores $\text{Hash}(P) = \pi$

Augmented EKE (A-EKE)

- Prevents impersonation of U with password P stolen from A
 - $\pi = \text{hash}(P)$ – user knows P; authenticator has only this hash
 - $S_p()$: one-way function configured with P (e.g., MAC); verified with V_p



Examples of Vp and Sp

- **Sp Function - Hashing with Salt:**
 - A cryptographic hash function combined with a unique salt for each user.
 - **Example:** $\text{Sp}(\text{password}, \text{salt}) = \text{hash}(\text{salt} \parallel \text{password})$
 - where a secure hash function like SHA-256 is used, and \parallel denotes concatenation
 - method prevents pre-computation attacks, like rainbow tables, by ensuring each password hash is unique, even if the same password is used by different users
- **Vp Function - Secure Password Verification:**
 - Time-constant comparison to mitigate timing attacks.
 - **Example:** $\text{Vp}(\text{stored_hash}, \text{input_password}, \text{salt}) = \text{constant_time_compare}(\text{hash}(\text{salt} \parallel \text{input_password}), \text{stored_hash})$
 - ensures that the verification time is constant regardless of the number of characters that match, preventing attackers from gaining information through timing analysis

Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication
- **Authentication protocols**
 - Web authentication

HTTP Basic/Digest Authentication

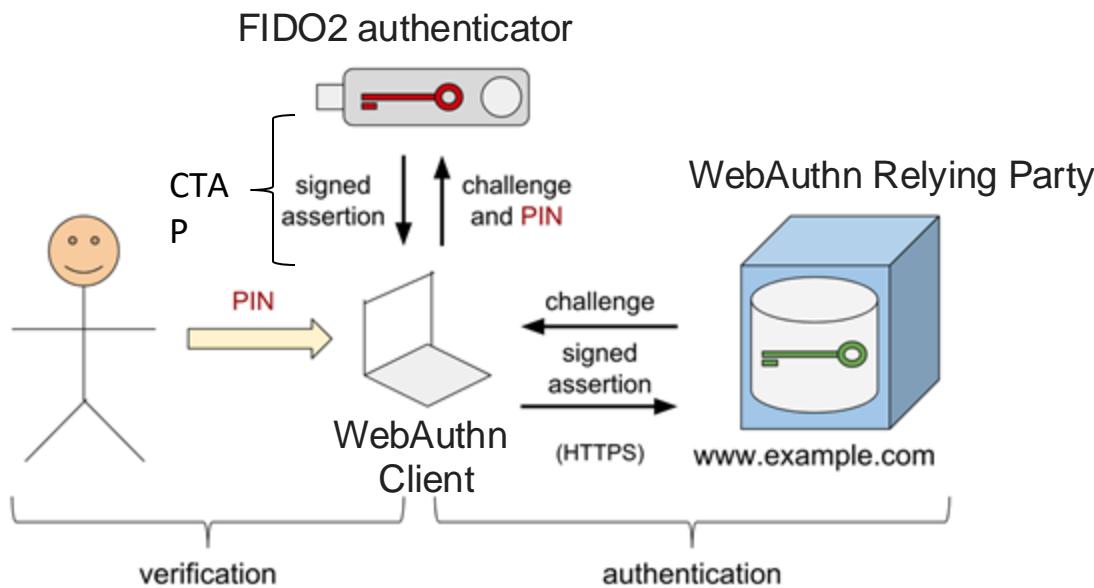
- note: never use on insecure connections, only over HTTPS
- Basic Authentication: when user asks for protected page:
 - server sends reply with status code 401 (unauthorized) and WWW-Authenticate header, e.g.:
 - *WWW-Authenticate: Basic realm="Authorized Personnel Only"*
 - browser asks user for username and password for that page and realm
 - browser sends username and password (**bad**) to the server encoded in base 64 (trivial to decode), e.g.:
 - *Authorization: Basic YWJlbfyZG86YmFzaWM=*
- Digest Authentication:
 - server sends something similar but: *Digest*, a nonce, a hash algorithm
 - browser sends **hash** instead of credentials

Authentication in HTTPS

- **Server authentication**
 - Authentication provided by SSL/TLS (later)
 - Browser authenticates the server using the server's public key stored in a certificate
 - Browser has certificates with public keys of CAs
- **User/client authentication**
 - Typically, server does not authenticate the browser (**client**)
 - Instead, the **user** authenticates himself using, e.g., username/password
 - Ok if HTTPS connection already established (credentials are sent encrypted over the network)

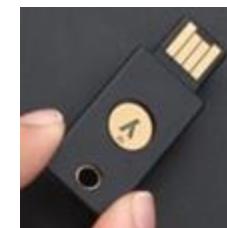
FIDO2 user authentication

- **Strong user authentication for the web**
 - Based on user-controlled cryptographic authenticators: smartphone, hardware security key
- Two components:
 - W3C Web Authentication (WebAuthn) standard
 - FIDO Client to Authenticator Protocol (CTAP)



Yubico YubiKey 5

NFC



Yubico Yubikey
5C



CryptoTrust
OnlyKey



Session-based authentication

- Server sends **cookie** with session identified to the client
 - Set-cookie: string in the header
- Whenever browser sends request to the server, cookie is inserted automatically
 - cookie: string in the header
- With **HTTPS**: cookies ciphered; OK
- Without **HTTPS**: cookies sent in plain text; can be stolen and used by attacker

Roadmap

- Authentication
- Authentication with passwords
- Biometric authentication
- **Authentication protocols**
 - Web single-sign on

Single-Sign On

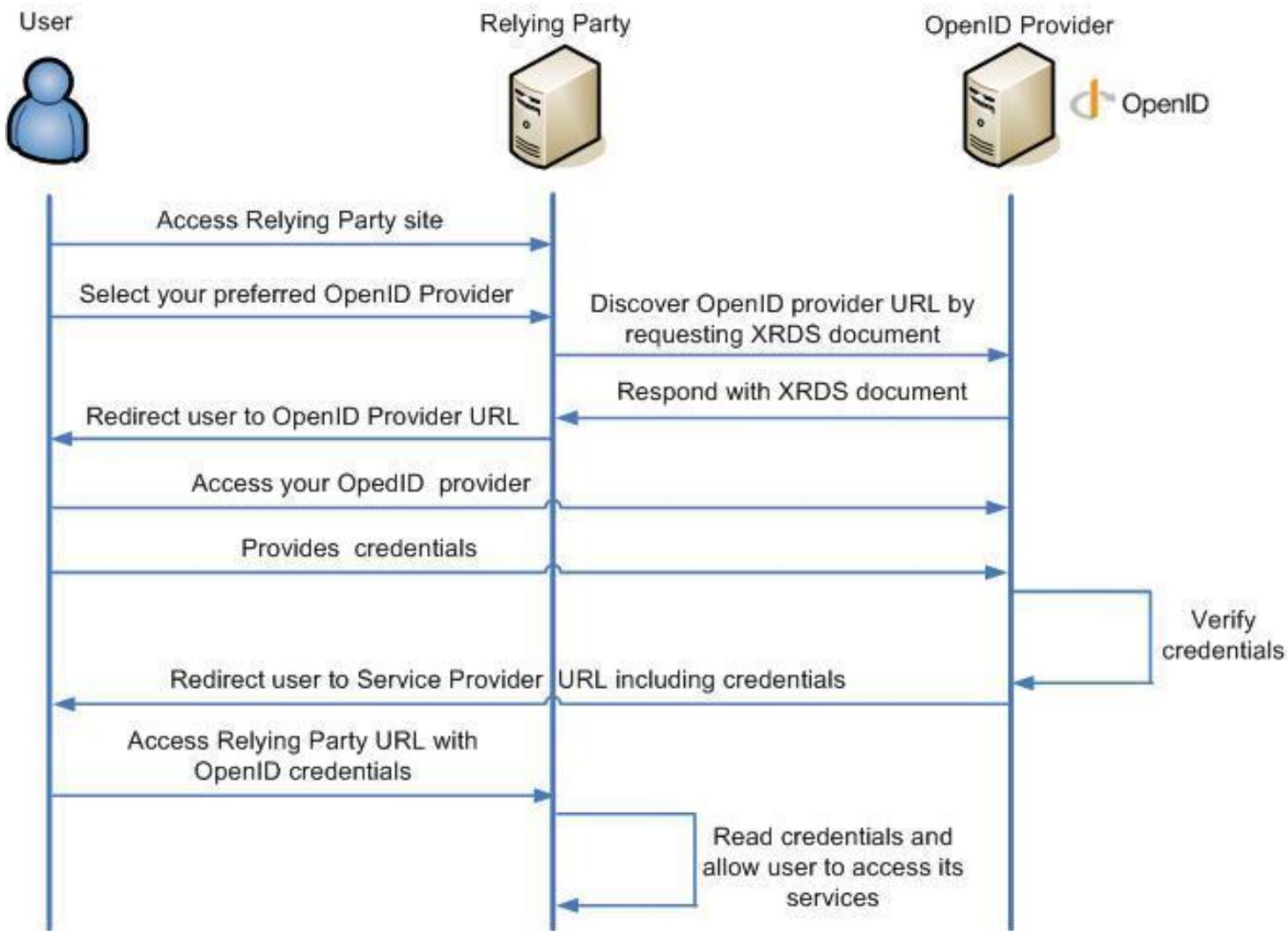
- Problem: users need to authenticate in many services, keep many passwords
 - Reusing passwords is bad, if one is stolen...
- Solution: **Single-Sign On**
 - Authenticate once, use several times, or
 - Have a single means of authentication, use several times
- For the web there are few SSO solutions:
 - **OpenID 2.0**
 - **OpenID Connect / OAuth**
 - Have a single means of authentication, use several time

OpenID 2.0



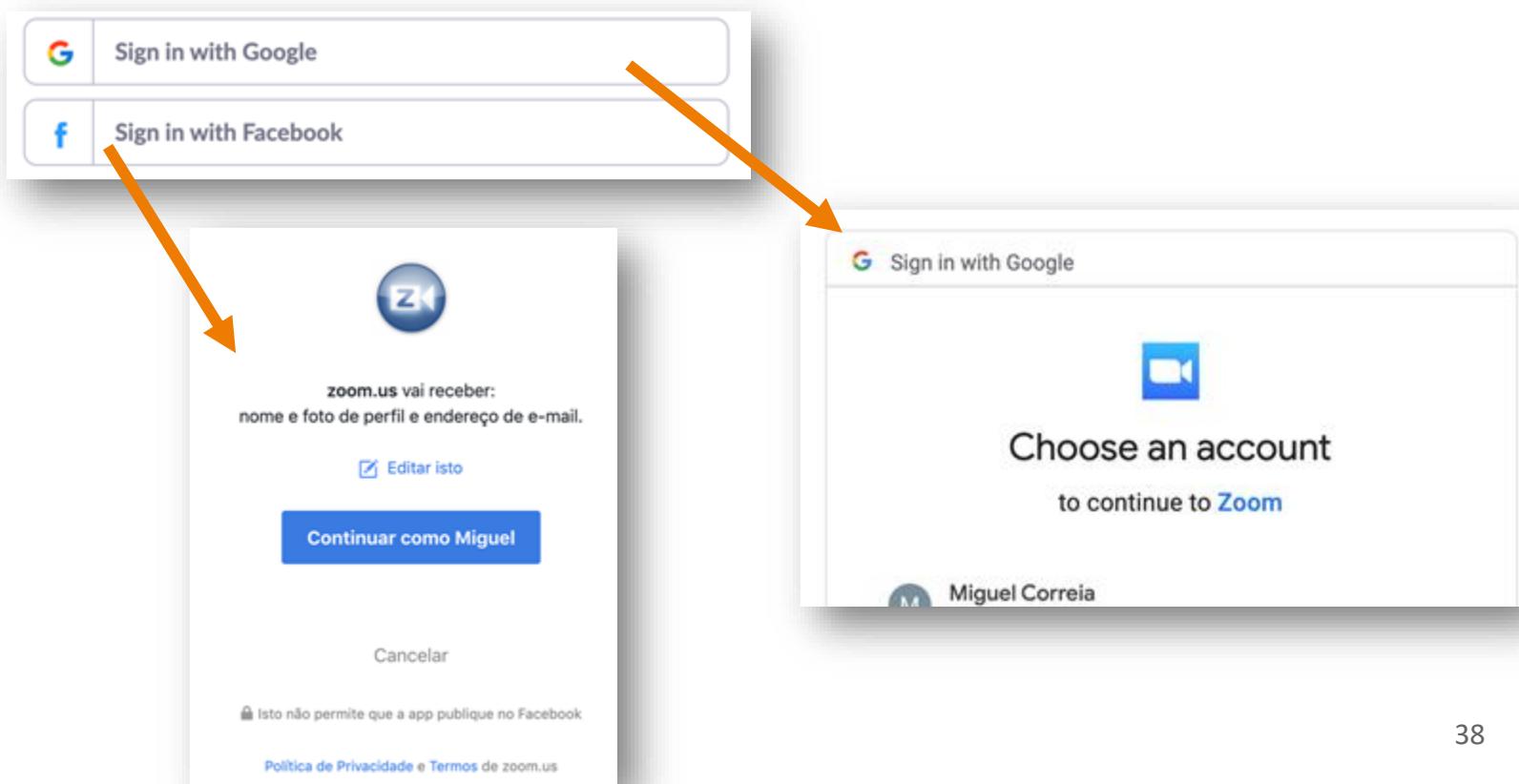
- Service provided by an **OpenID provider**
 - Google, Yahoo!, Wordpress...
- **User** wants to login in a **relying party**, i.e., a web site that supports OpenID (displays the logo)
 - User provides his **OpenID**
 - Relying party transforms the OpenID in a URL and redirects the user's browser to the **OpenID provider**
 - OpenID provider typically asks for password and the user to allow the relying party to use the account
 - User is redirected to the relying party website; OpenID provider sends authentication data to the relying party
- **Important:** all the communication links use HTTPS
 - Secure channels with authenticated servers

OpenID 2.0

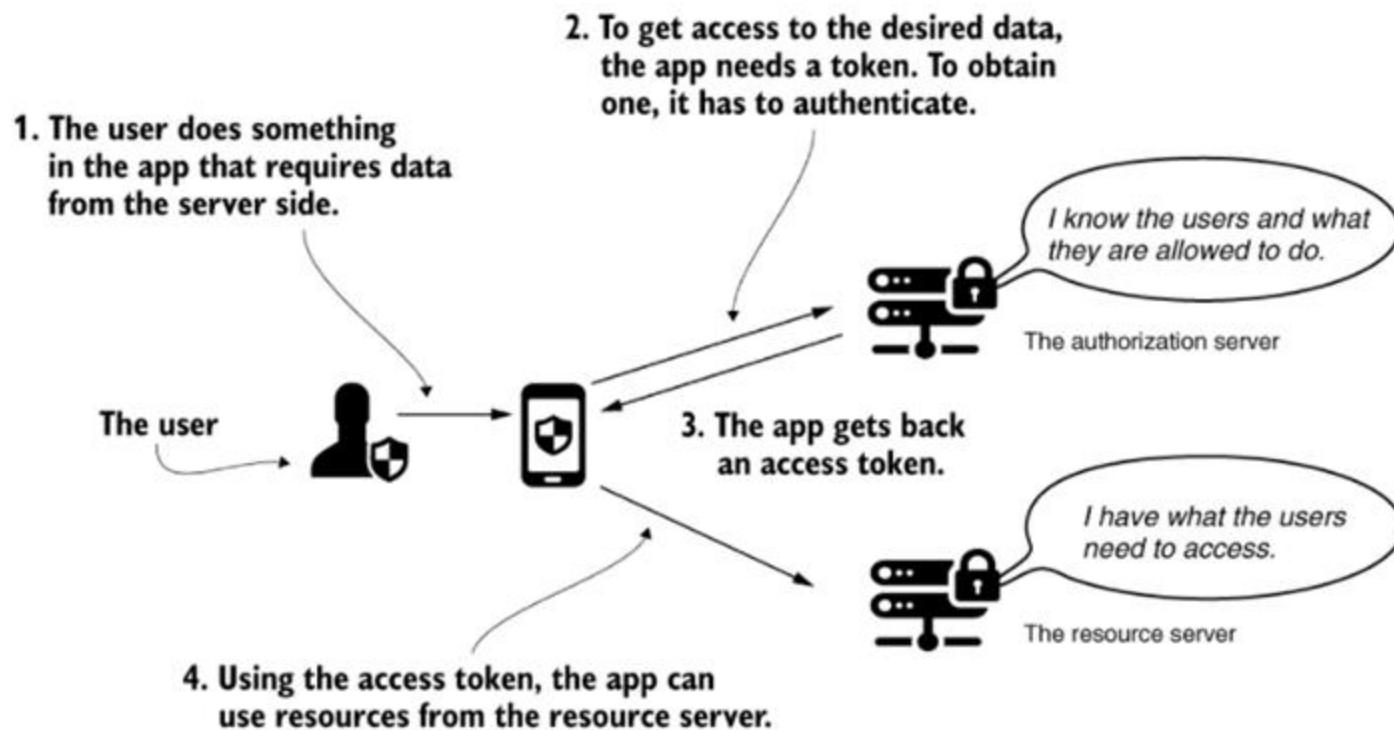


OpenID Connect / OAuth

- **OpenID Connect** (OIDC) works similarly to OpenID
 - Works on top of **OAuth 2.0**, a distributed authorization framework
 - Example from Zoom's Help Center:



OAuth2 authorization flow



Summary

- Authentication
- Authentication with passwords
- Biometric authentication
- Authentication protocols

Authorization

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Carlos Ribeiro, Miguel P. Correia

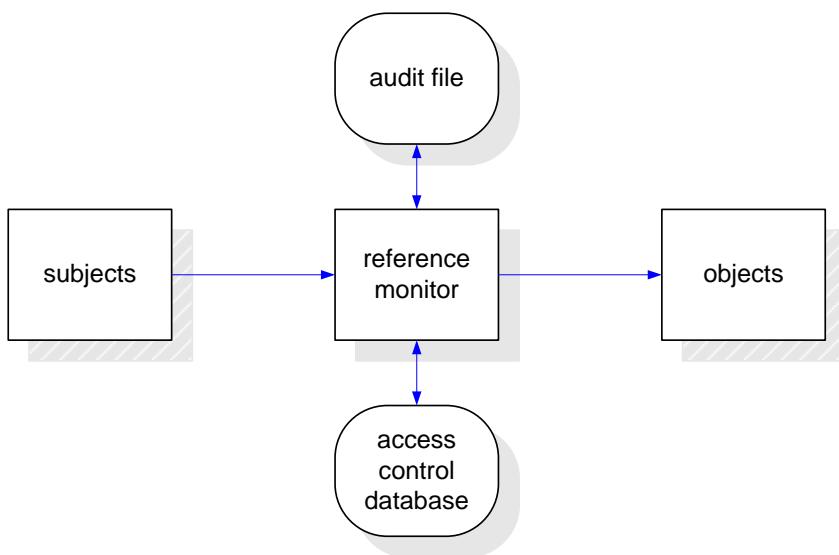
Roadmap

- Authorization / access control
- Access control models

Roadmap

- **Authorization / access control**
- Access control models

Access control



- Subjects
 - People, processes
 - Active entities
- Objects
 - Files, processes
 - Passive entities
- Accesses
 - Operations on objects
- **Reference Monitor**
 - Small and verifiable
 - Total mediation

Access actions

- There are several options; they can be:
 - **Specific**
 - Each object has a set of specific operations
 - Large policy size 😞
 - **Generic**
 - Only write and read (change or not the object's state)
 - **Mixed**
 - (next slide)

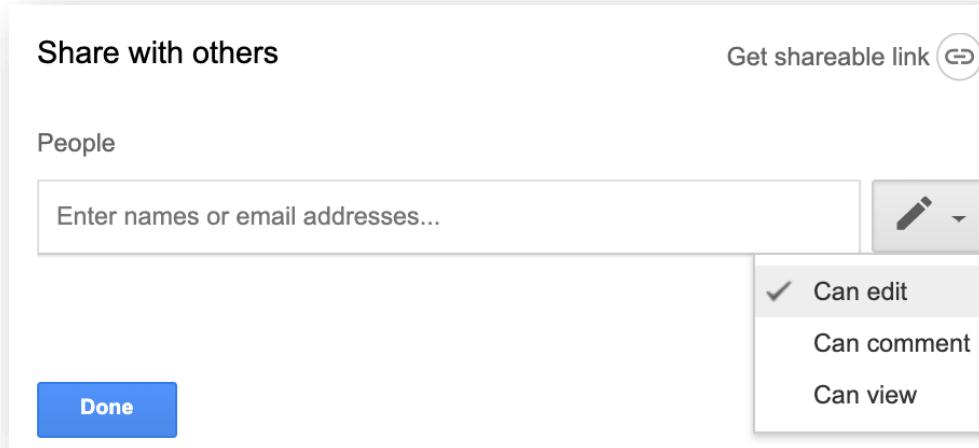
Mixed access actions

- **Mixed**
 - Bell LaPadula (BLP): execute, append, read, write
 - Different objects, different meanings;
example from Unix:

Operation	Meaning for directories	Meaning for files
<u>Read</u>	List content	Read content
<u>Write</u>	Create / rename	Create / rename
<u>Execute</u>	Enter (cd) and access files/dirs	Run program in the file

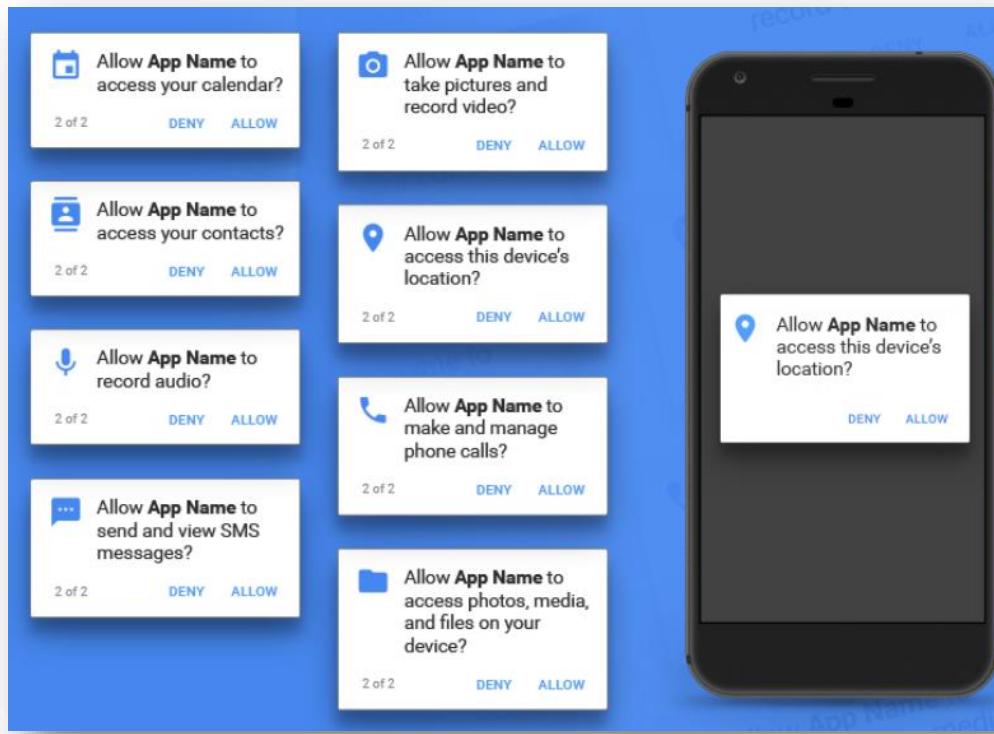
Specific access actions example: Google Drive

- Actions: edit, comment, view



Specific access actions example: Android

- Actions: access, record, send/view, ...
- Many objects: calendar, contacts, audio, SMSs, location,...



Management and Ownership

- Owner
 - Usually the creator of the object
- **Discretionary access control systems (DAC)**
 - Management made by the owner
- **Mandatory access control systems (MAC)**
 - Management made by a global policy
 - Less susceptible to malware



Caution: unrelated to Message Authentication Codes, Medium Access Control

Roadmap

- Authorization / access control
- **Access control models**

Model goals

Trustworthiness

Expressivity



Performance

Administration

Discretionary Access Control

Access matrix

- **Access control matrix**
 - Theoretical model, very sparse

- **Access Control List (ACL)**

- Matrix columns
- Matrix rows

	Obj 1	Obj 2	Obj 3
Suj 1	{read}		{read, write}
Suj 2		{read, write}	
Suj 3	{read, write}		

Access Control List

- One ACL for each object
 - ACL – non-empty element of each column in the matrix
 - ACE (Access Control Entry) – a cell of the matrix
- Can be assigned to groups of subjects
 - Minimize policy
 - Negative permissions may be required
- Hard to manage individual subject permissions

Capabilities

- **One list of capabilities for each subject**
 - Each capability is a non-empty element in a matrix row
- It is hard to:
 - Know who can access an object
 - Revoke a capability
- Use:
 - Traditionally less used than ACLs
 - In distributed systems
 - Example: access document with link (URL)
 - Whoever knows the link, can access the object

Roadmap

- Authorization / access control
- **Access control models**
 - Role-Based Access Control

Role-Based Access Control (RBAC)

- Performs access control based on **roles**
- Allows the description of complex policies
 - Segregation of duties (static and dynamic)
 - Static: allows role memberships that are mutually exclusive
 - Dynamic: allows same subject having 2 roles but not using both in same operation
 - Least privilege:
 - possible to assign the least privileges the subject needs to the role
 - Delegation:
 - possible to transfer privileges
 - Restrictions based on: time, context, history

RBAC mechanism

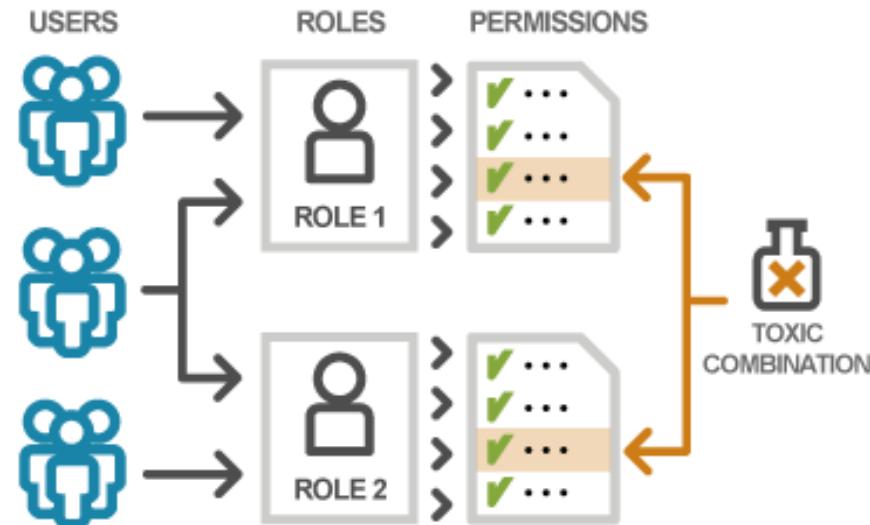
- Users are associated with **roles**
- Roles are associated with **permissions**
 - A user has a permission if he has an association with a role that has an association with a permission
- Reduces size of policy
 - Better scalability
 - Reduces error probability
 - Simplifies administration

RBAC administration benefits

- Associations between roles and permissions are more stable than associations between users and permissions
 - So easier to administrate: just associate users to roles
- May be associated with different types of concept
 - Position
 - Authority
 - Skill
 - Responsibility
- Allows propagating rights across hierarchies
 - Similar to inheritance in object oriented programming

RBAC overall assessment

- RBAC models categorize users based on similar needs and groups them into roles
 - The role concept uses approximations for the sake of simplicity
 - There is a never-ending struggle to refine the definition of a role
 - and to maintain a sound segregation of duties

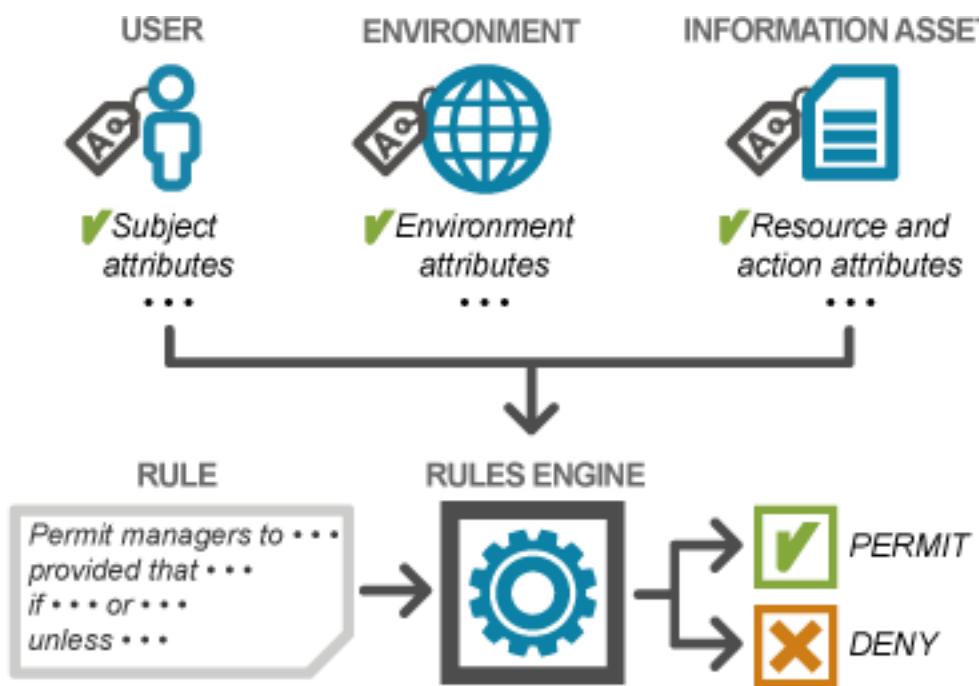


Roadmap

- Authorization / access control
- **Access control models**
 - Attribute Based Access Control

Attribute-Based Access Control (ABAC)

- Permissions are granted or denied depending on the values of named attributes



ABAC assessment

- **Dynamic permissions**
 - Current business rules
 - Risk mitigating precautions
 - Context-related security measures
- **Fine-grained authorization**
- **Major disadvantages:**
 - Complex model – big attack surfaces
 - Lower performance (higher delay)

Roadmap

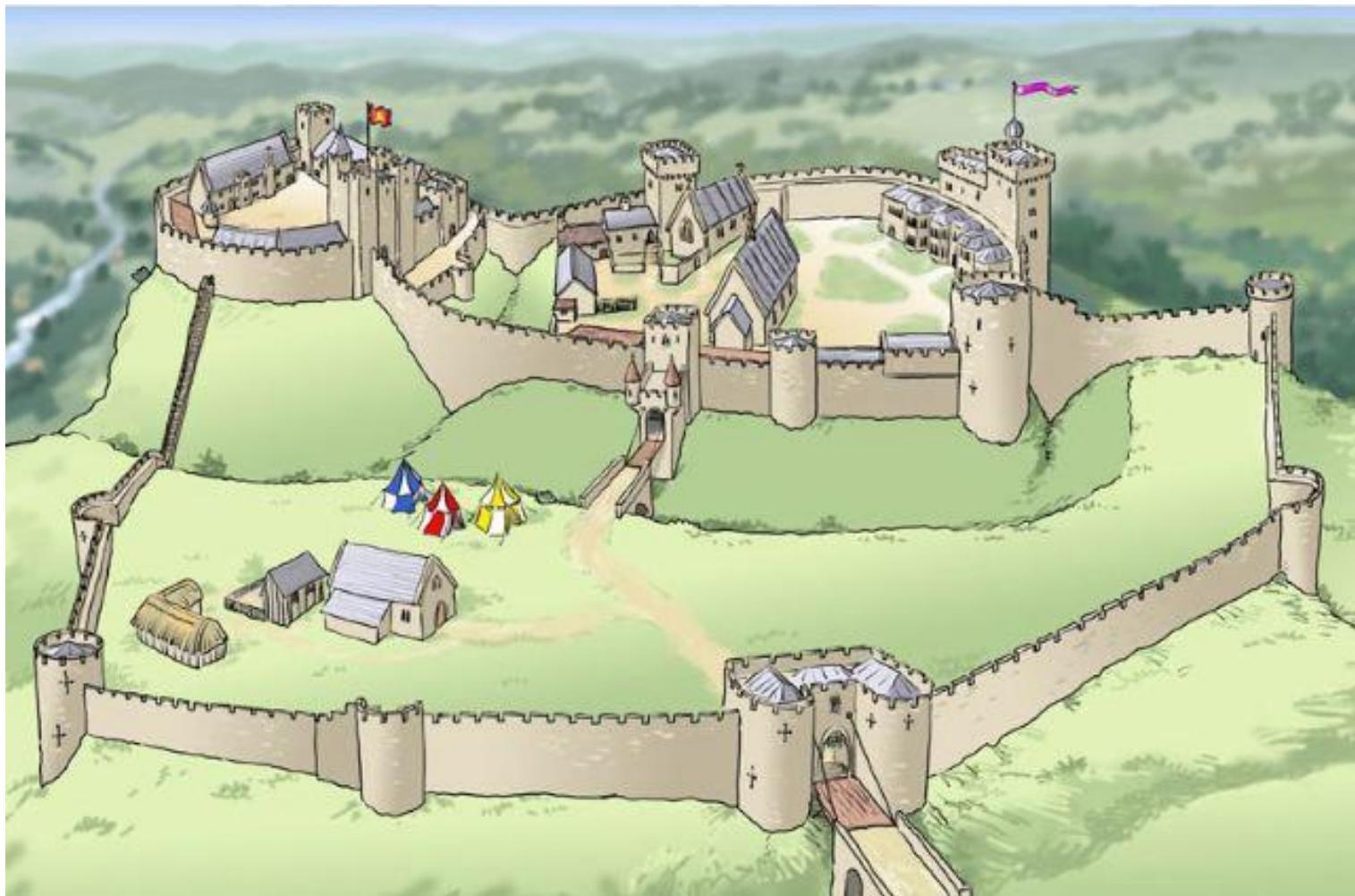
- Authorization / access control
- **Access control models**
 - Bell-LaPadula model

Bell-LaPadula (BLP) model

- Mandatory Access Control (MAC)
- State machine
 - ls – clearance of subject s*
 - lo – classification of object o*
- Simple Security Property
 - **No read up:** access granted iff $lo \leq ls$
- Confinement Property (\star -Property)
 - **No write down:** access granted iff $ls \leq lo$



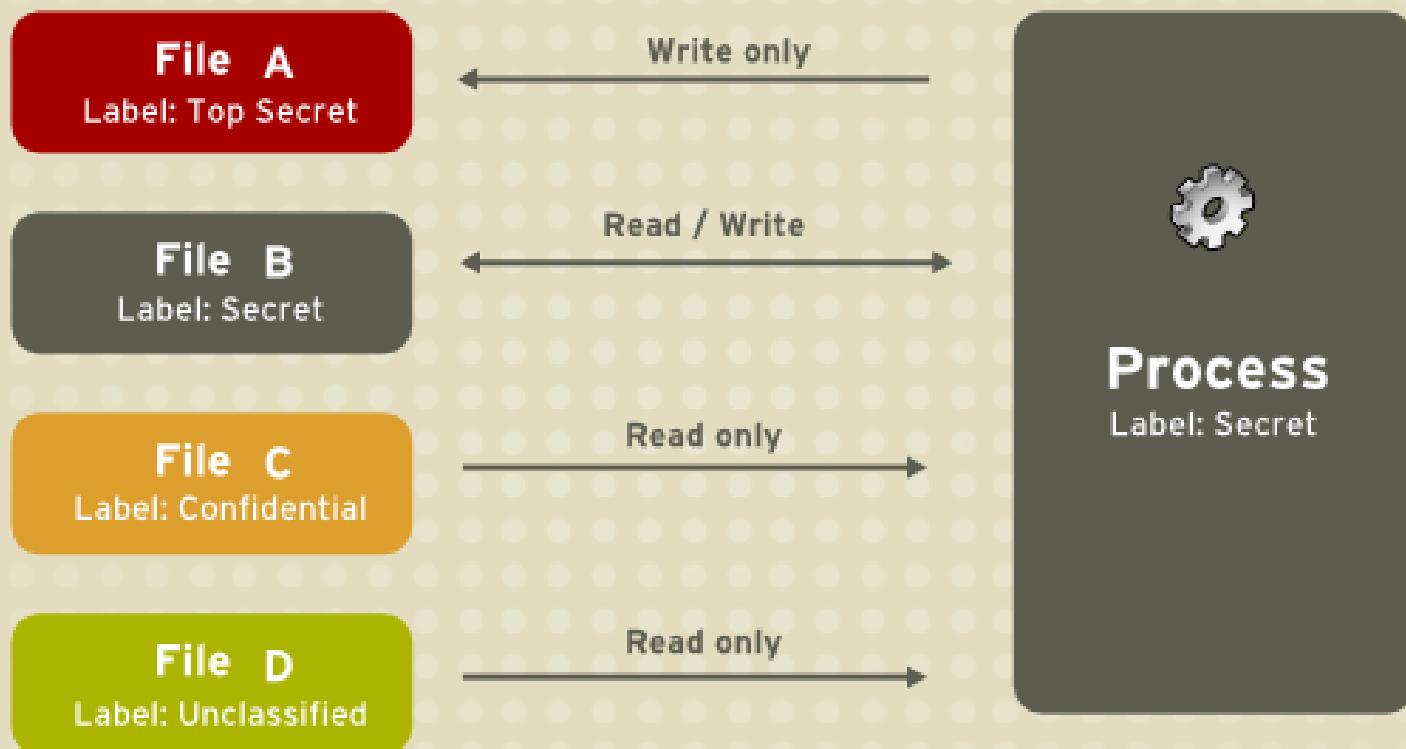
Bell-LaPadula (BLP)



© selfless security

BLP: file access example

Available data flows using an MLS system.



Processes can read the same or lower security levels but can only write to their own or higher security level.

Bell-LaPadula (BLP) overall assessment

- Confidentiality only
 - No integrity
- Data changes only through specific programs
- There are covert channels
 - e.g. file names
- Does not allow management (rights are fixed)
 - Nor delegation

Summary

- Authorization / access control
- Access control models

Secure Communication Overview

Segurança Informática em Redes e Sistemas
2024/25

Ricardo Chaves, David R. Matos

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Roadmap

- Goals
- Basic approximations
 - Link security *versus* end-to-end security
- Acting layers

Goals of secure communication

- Provide security mechanisms and protocols for communication assuring:
 - **Confidentiality** of the exchanged data
 - **Integrity** verification of the exchanged data
 - **Authenticity** of the communicating parties
 - Machines, services, users

Secure channel functionalities

- Trustworthy **authentication** between parties
- Negotiation/computation of **session keys**
 - Session key: secret shared between the communicating parties
 - Support secure communication
 - Session key establishment often integrated with the authentication (above)
- Secure **data exchange**
 - Leveraging secure channels using the session keys

Basic approximations

- **Link security**
 - Security is assured between a pair of machines
 - Is not scalable / not adaptable to the Internet
 - We would have to trust each “hop” in the network path
- **End-to-end security**
 - Security is assured by end-point software exchanging the messages
 - Scalable and adaptable to the Internet

End-to-end principle

- Formulated by Saltzer, Reed and Clark in 1981, applied to **reliability**
- IP assumes this principle for correcting errors
 - It is easier to obtain reliability beyond a certain margin by mechanisms in the end hosts of a network rather than in the intermediary nodes
 - Especially when the intermediary nodes are beyond the control of, and not accountable to the end hosts
- End-to-end principle can be applied to **security**
 - By implementing security in the lower levels, all channels must bear its cost
 - Even if not necessary
 - Hard to verify
 - By implementing security in the higher levels, the channel endpoints can implement the functionalities tailored to the application's needs
 - Avoid redundancies

Acting layers: Actions and solutions

Layers		Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS	
	Presentation			SSH	
	Session				
	Transport	Operating Systems		TLS	
	Network			IPsec	
	Link	Devices	Link security	IEEE 802.11*	
	Physical				

Choosing the correct layer (1/2)

- **Transaction layer solution**
 - Allows the secure exchange of objects
 - e.g., mail messages, documents, files
 - Allows to assure **non-repudiation** of object authorship
- **Application and transport layer solution**
 - Allows security to adapt to the needs of distributed applications
 - Requires the applications to be modified to use these protocols
 - It is not easy to design a solution that is simultaneously:
 - **Generic** – to be used by many applications and
 - **Powerful** – to address the requirements of all applications

Choosing the correct layer (2/2)

- **Network and link layer solution**
 - Requires modification of the operating system or device
 - Tend to be simpler and more generic
 - Independent of the applications
 - Allow the applications to remain unchanged
 - but may not cover all their security requirements
 - Requires extra-application control
 - Definition of minimum security requirements
 - Selecting policy of security mechanisms

Acting layers

- Advantages of acting at **upper layers**
 - Better fit to the application's requirements
 - Possible to enforce non-repudiation
 - Awareness of the entities involved: users, services
 - Awareness of session and connection that allows
 - Different key management protocols
 - Different ways of exploring security mechanisms
- Advantages of acting at **lower layers**
 - Simplicity of use, coverage
 - No need to modify applications and higher layer infra-structures

Secure communication: Data Link layer and below

Layers		Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS, SSH	
	Presentation				
	Session				
	Transport	Operating Systems		TLS	
	Network			IPsec	
Link		Devices	Link security	IEEE 802.11*	
Physical					

Secure communication: Network layer

Layers		Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS SSH	
	Presentation				
	Session				
	Transport	Operating Systems		TLS	
	Network	Devices	Link security	IPsec	
	Link			IEEE 802.11*	
	Physical				

Secure communication: Transport layer and above

	Layers	Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS, SSH	
	Presentation				
	Session				
	Transport	Operating Systems	Link security	TLS	
	Network			IPsec	
	Link	Devices		IEEE 802.11*	
	Physical				

Wi-Fi security

Segurança Informática em Redes e Sistemas
2024/25

Ricardo Chaves, David R. Matos

Ack: Carlos Ribeiro, André Zúquete,
Miguel Pardal, Miguel P. Correia

Secure communication: Data Link layer and below

Layers		Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS	
	Presentation			SSH	
	Session				
	Transport	Operating Systems		TLS	
	Network			IPsec	
Link		Devices	Link security	IEEE 802.11*	
Physical					

Roadmap

- Wireless networks
- Wi-Fi / WLANs
 - WEP
 - WPA
 - 802.1X and EAP

Roadmap

- **Wireless networks**
- Wi-Fi / WLANs
 - WEP
 - WPA
 - 802.1X and EAP

Wireless network challenges

- Inexistence of a controlled physical connection
- Worsens the problems of:
 - **Eavesdropping** of data exchanged
 - Breaking *confidentiality*
 - **Impersonation** of machines
 - Breaking *authenticity*

Wireless communication protocols

- Mobile phones
 - GSM
 - GPRS
 - UMTS
- Wireless home phones
 - DECT
- **Data networks**
 - Bluetooth (IEEE 802.15)
 - **Wi-Fi (IEEE 802.11*)**
 - where * = a, b, g, n, ...
 - standard for WLAN (Wireless Local Area Networks)

Roadmap

- Wireless networks
- **Wi-Fi / WLANs**
 - WEP
 - WPA
 - 802.1X and EAP

IEEE 802.11* Architecture

- **Station (STA)** 
 - Device capable of connecting itself to a wireless network
 - Each station has an identifier
 - Media Access Control (MAC) address
- **Access Point (AP)** 
 - Device that allows the interconnection between a wireless network and other equipment or networks
- **Wireless network**
 - Network composed of stations (STA) and access points (APs) that communicate through **radio signals**

New Wi-Fi version names



- New versioning scheme, with sequential numbers
 - To replace the old, confusing standard names with letters, like “802.11ac”
- Wi-Fi versions
 - Wi-Fi 1 would have been 802.11b, released in 1999
 - Wi-Fi 2 would have been 802.11a, also released in 1999
 - Wi-Fi 3 would have been 802.11g, released in 2003
 - Wi-Fi 4 is 802.11n, released in 2009
 - Wi-Fi 5 is 802.11ac, released in 2014
 - Wi-Fi 6 is the new version, also known as 802.11ax.
It was released in 2019
 - Wi-Fi 7 expected in 2025...

Wi-Fi security



- 1999
 - WEP – Wired-Equivalent Privacy
- 2003
 - WPA – Wi-Fi Protected Access
- 2004
 - **WPA2 (802.11i)**
- 2018
 - WPA3

IEEE 802.11* Security

- Initial very basic mechanisms and protocols
 - *Service Set Identifier* (SSID)
 - *MAC Address Filtering*
 - *Wired Equivalent Privacy* (WEP)
- Enterprise/campus authentication: 802.1X
 - *Enhanced Authentication Protocol* (EAP)
 - *Protected EAP* (PEAP)

IEEE 802.11*

SSID (Service Set Identifier)

- **SSID** = identifier/name of a wireless LAN
 - Used by the AP to restrict access of stations
 - Stations have to know and use the SSID of the AP that they are connected to
- Works like a *weak* password
 - Everyone knows it
 - It is exchanged in plaintext in each message
 - The AP announces it

CARLOS_5G_EXT
CARLOS_EXT
DIRECT-90-HP ENVY 5640
DIRECT-tFE0443180msDY
MEO
MEO-10F49A
MEO-39E593 5GHz
MEO-A10B05
MEO-B8A8F0
MEO-B8A8F1-5G
MEO-WiFi
NOS-3610
NOS-5B53
NOS_Wi-Fi_Hotspots
SMC
Thomson724D8F
Vodafone-D36C04
ZON Repeater
ZON-5010
ZON-E490

IEEE 802.11*

MAC Address Filtering

- Each Station has a distinct **MAC**
 - The idea was to be fixed but today can be changed
- Each AP is able to restrict the access of a Station according to their MAC, but:
 - MAC is transmitted in clear text and can be eavesdropped
 - MAC can be spoofed by an attacker



Minko Gechev @mgechev · 1 d
A bash function I use constantly on airports:

```
function changeMac() {  
    local mac=$(openssl rand -hex 6 | sed  
's/\(..\)/\1:/g; s/.$/"/')  
    sudo ifconfig en0 ether $mac  
    sudo ifconfig en0 down  
    sudo ifconfig en0 up  
    echo "Your new physical address is  
$mac"  
}
```

Unlimited WiFi ✨

46

667

3 226



Roadmap

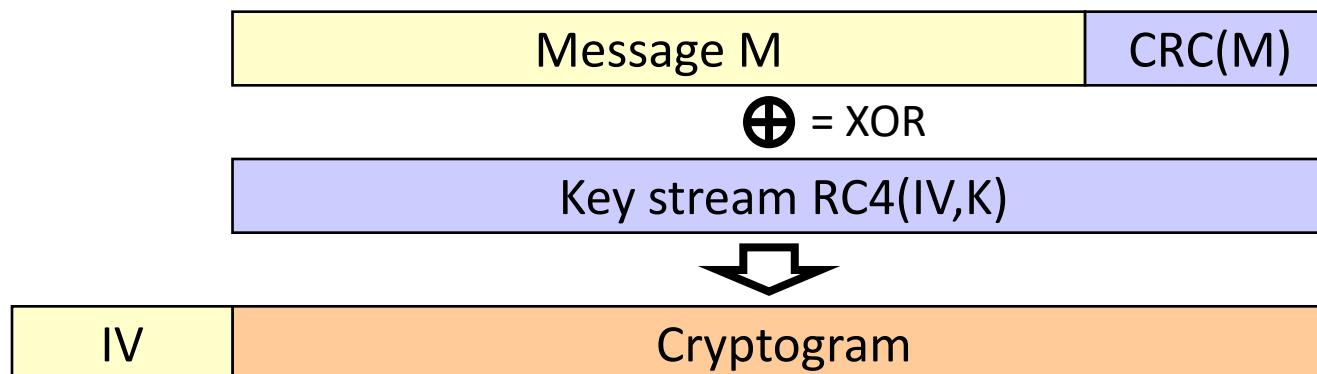
- Secure channels *versus* communication layers
- Wireless networks
- Wi-Fi / WLANs
 - **WEP**
 - WPA
 - 802.1X and EAP

IEEE 802.11*: WEP (*Wired Equivalent Privacy*)

- Goal
 - Protection of radio communications between stations and APs
 - Confidentiality and integrity control
- Usage
 - Uses **shared symmetric keys** of 40 or 104 bits
 - Defined by administrator and shared between stations
 - Manual distribution of keys
 - Uses a **stream cipher**: the RC4 algorithm

WEP message security

- **Integrity and Confidentiality**
 - Every message takes a CRC (Cyclic Redundant Check) value
 - and is encrypted using RC4 (stream cipher)



WEP problems (1/3)

- The **AP** is not authenticated
- Excessive use of the **shared key**
 - No key redistribution
- No control over the variation of the **IV**
 - Which allows ad-hoc repetition of ciphered messages previously sent, modified, or new messages
- It is possible to repeat the same **keystream**

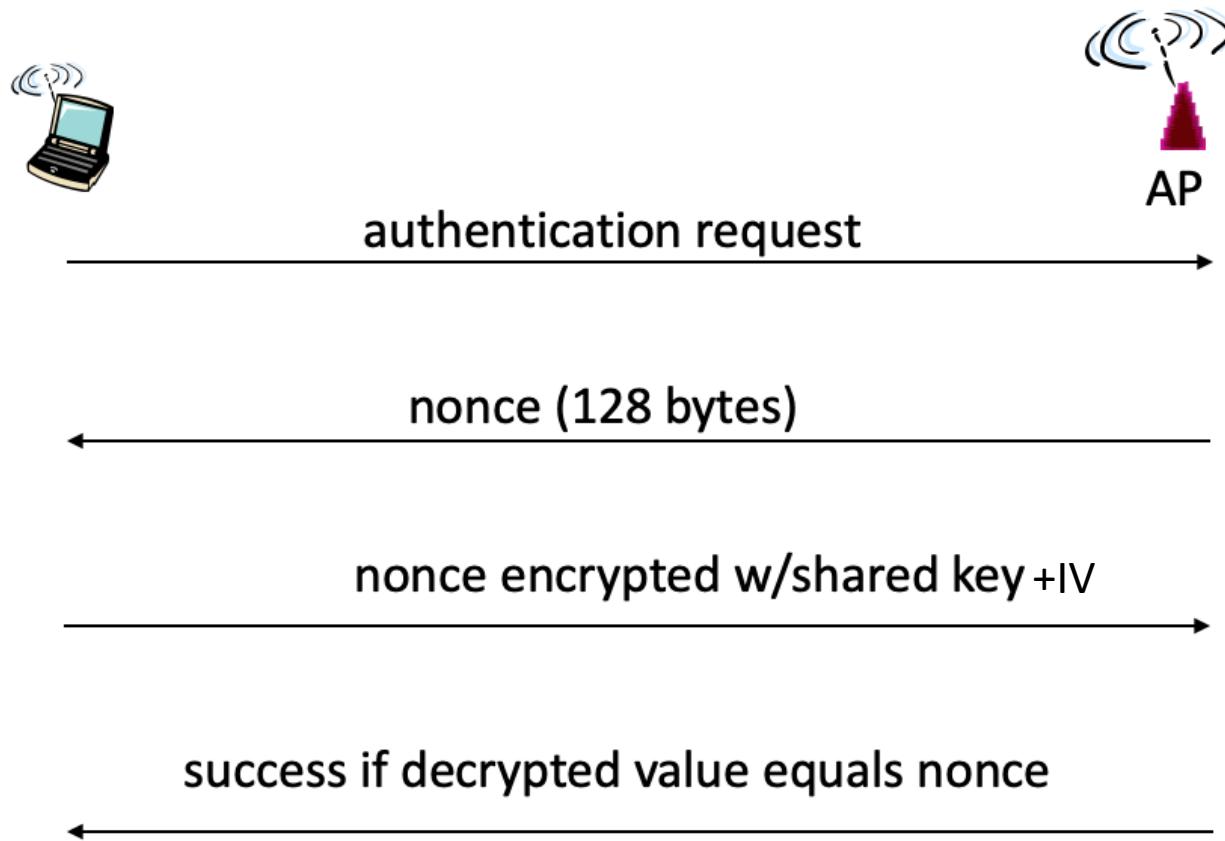
WEP problems (2/3)

- Repetition of **IVs** for the same SSID and the same Key
 - Same IV and Key \Rightarrow same keystream
 - XORing 2 cryptograms obtained with the same keystream, one obtains the XOR of the two messages and their CRC
 - $C_1 = M_1 \text{ xor } \text{keystream}(IV, K)$
 - $C_2 = M_2 \text{ xor } \text{keystream}(IV, K)$
 - $C_1 \text{ xor } C_2 = (M_1 \text{ xor } \text{keystream}(IV, K)) \text{ xor } (M_2 \text{ xor } \text{keystream}(IV, K)) = M_1 \text{ xor } M_2$
 - IV has only 24 bits and sometimes is poorly managed
 - Constant (IEEE 802.11 standard states that IV update is optional)
 - 0 on reset (in some equipments)

WEP problems (3/3)

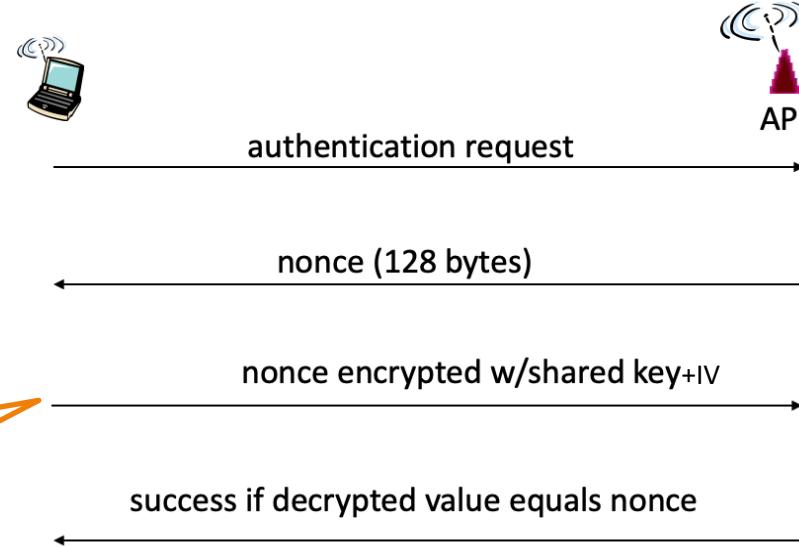
- **Integrity control** is weak
 - CRC-32 (Cyclic Redundancy Check) is a linear function
 - Changing n^{th} bit of cryptogram,
changes same n^{th} bit in message,
does a deterministic change of some bits of the CRC
 - Thus, the deciphered CRC can be tampered with by
inverting the corresponding CRC bits in the cryptogram

WEP authentication



WEP authentication attacks

- **Authenticating** with the same keystream
 - AP sends message with **nonce (M)** in plaintext to good station
 - Good station ciphers **nonce** and sends it back in cryptogram **C**
 - **Attacker** observes **C** and **M**, so it can obtain 128 bytes of the keystream for a given **IV**: $C = M \text{ xor keystream}(IV, K)$
 - **Attacker** sends authentication request and uses the keystream 128 bytes (without knowing **K**) to encrypt the **nonce**



Roadmap

- Secure channels vs communication layers
- Wireless networks
- Wi-Fi / WLANs
 - WEP
 - **WPA**
 - 802.1X and EAP

WPA (*Wi-Fi Protected Access*)

Improvements over WEP:

- **Master key** has 128 bits and is never used to cipher data; **temporary keys** are derived from this master key (TKIP protocol)
- The size of the **IV** is increased to 48 bits
- Each packet is protected with a different key
- **IV** is used as a packet counter: **TSC** (TKIP Sequence Counter)
 - For each new (temporary) integrity key, TSC is reset
 - Out of order TSCs are discarded to prevent replay attacks
- CRC-32 (linear) is ‘replaced’ by **MIChael**, a **Message Integrity Code**
 - Computed over the entire unencrypted data in the frame and the source and destination MAC addresses
 - If two wrong MICs are sent within 60s, the key is renewed, to prevent trial-and-error attacks

TKIP (*Temporal Key Integrity Protocol*)

- RC4 stream cipher algorithm
 - Master key is subject to an initial key mixing with the IV of 48 bits
- MIC (Message Integrity Control) in every message
 - 64-bit message integrity check value
- Improved management of dynamic keys
 - **PMK** – Pairwise Master Key (generated by 802.1X)
 - **PTK** – Pairwise Transient Key
 - **PTK** = PRF-512(PMK, “Pairwise key expansion”, ST_MAC, AP_MAC, SNonce, ANonce)
 - ST_MAC, AP_MAC – station and AP MAC addresses
 - Nonce = PRF-256(random, “Init Counter”, MAC, Time)
 - Ensures that every data packet is sent with a unique encryption key

WPA problems

- MIC does not protect the full packet
- In some cases, the same keystream is reused
- WPA was just a *draft* of the IEEE 802.11i standard, known as WPA2
 - Compatible with the same hardware devices as WEP

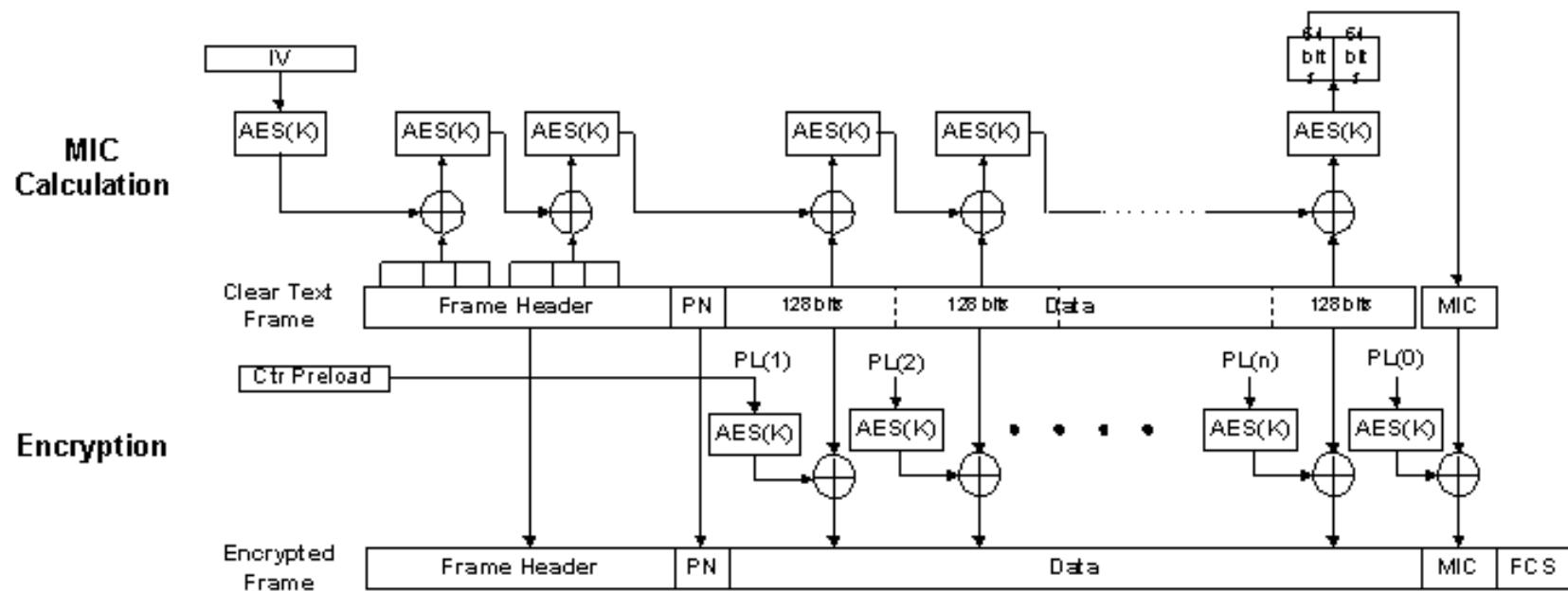
IEEE 802.11i a.k.a. WPA2



- $802.11i = \text{WPA2} = \text{WPA} + \text{AES-CCMP}$
- **WPA – Wi-Fi Protected Access**
 - TKIP – Temporal Key Integrity Protocol
 - WPA-Personal / WPA-PSK – uses a pre-shared key
 - WPA-Enterprise / WPA-802.1X - uses **802.1X** with all authentication methods seen above
- **AES-CCMP – AES, CTR, CBC-MAC**
 - Advanced Encryption Standard (**AES**) in Counter mode (**CTR**)
 - **CBC-MAC** – MAC function based on block cipher (AES) in Cipher Block Chaining (CBC) mode
 - **CCM** = **CTR** + **CBC-MAC**
 - **CCMP** = **CCM** + Padding

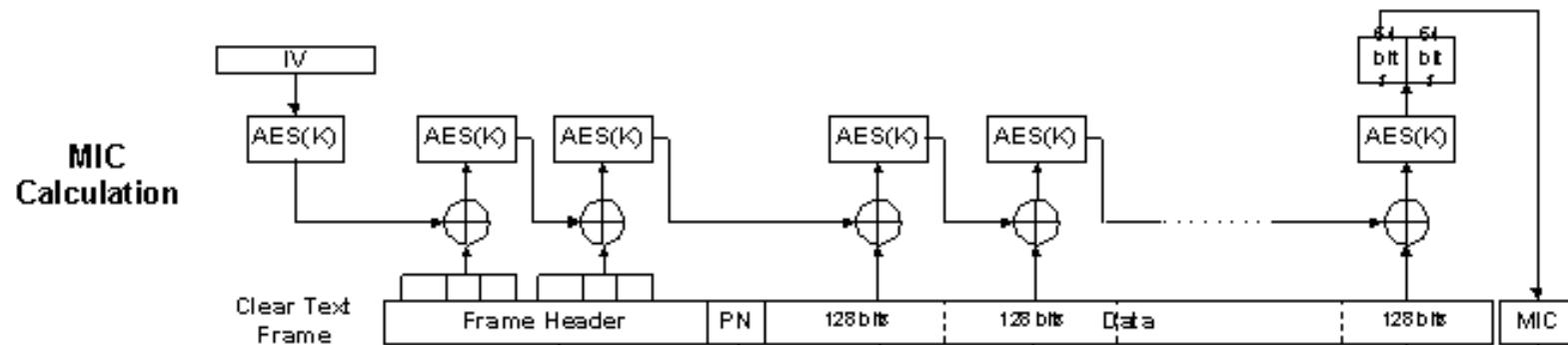
AES CCMP

(Counter Mode with CBC-MAC)

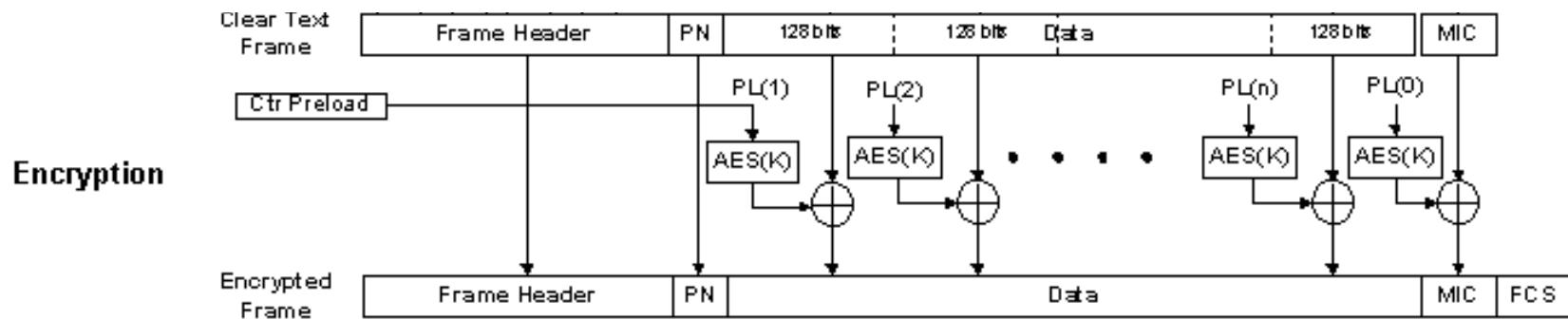


PN – packet number, similar to the TSC
(TKIP Sequence Counter)

Integrity protection: AES CBC-MAC



Confidentiality protection: AES Counter Mode



Comparison WEP vs WPA vs WPA2

	WEP	WPA (TKIP)	WPA2 (AES-CCMP)
Cipher	RC4	RC4	AES CTR
Key size	40 or 104 bits	128 bits encryption, 64 bits authentication	128 bits
Key lifetime	24-bit IV, wrap	48-bit IV	48-bit IV
Frame data integrity	CRC-32	MIChael	CBC-MAC
Frame header integrity	None	MIChael	CBC-MAC
Replay detection	None	IV sequencing	IV sequencing
Key management	None	EAP / 802.1X	EAP / 802.1X

WPA3

- Defined 2018
 - Required since July 2020
- Uses an equivalent 192-bit cryptographic strength in WPA3-Enterprise mode (AES-256 in GCM mode with SHA-384 as HMAC)
 - Still mandates the use of CCMP-128 (AES-128 in CCM mode) as the minimum encryption algorithm in WPA3-Personal mode
- The WPA3 standard also replaces the Pre-Shared Key exchange with Simultaneous Authentication of Equals as defined in IEEE 802.11-2016
 - More secure initial key exchange in personal mode
 - Forward secrecy
 - Mitigate security issues posed by weak passwords
 - Simplify the process of setting up devices with no display interface
- Protection of management frames as specified in the IEEE 802.11w amendment

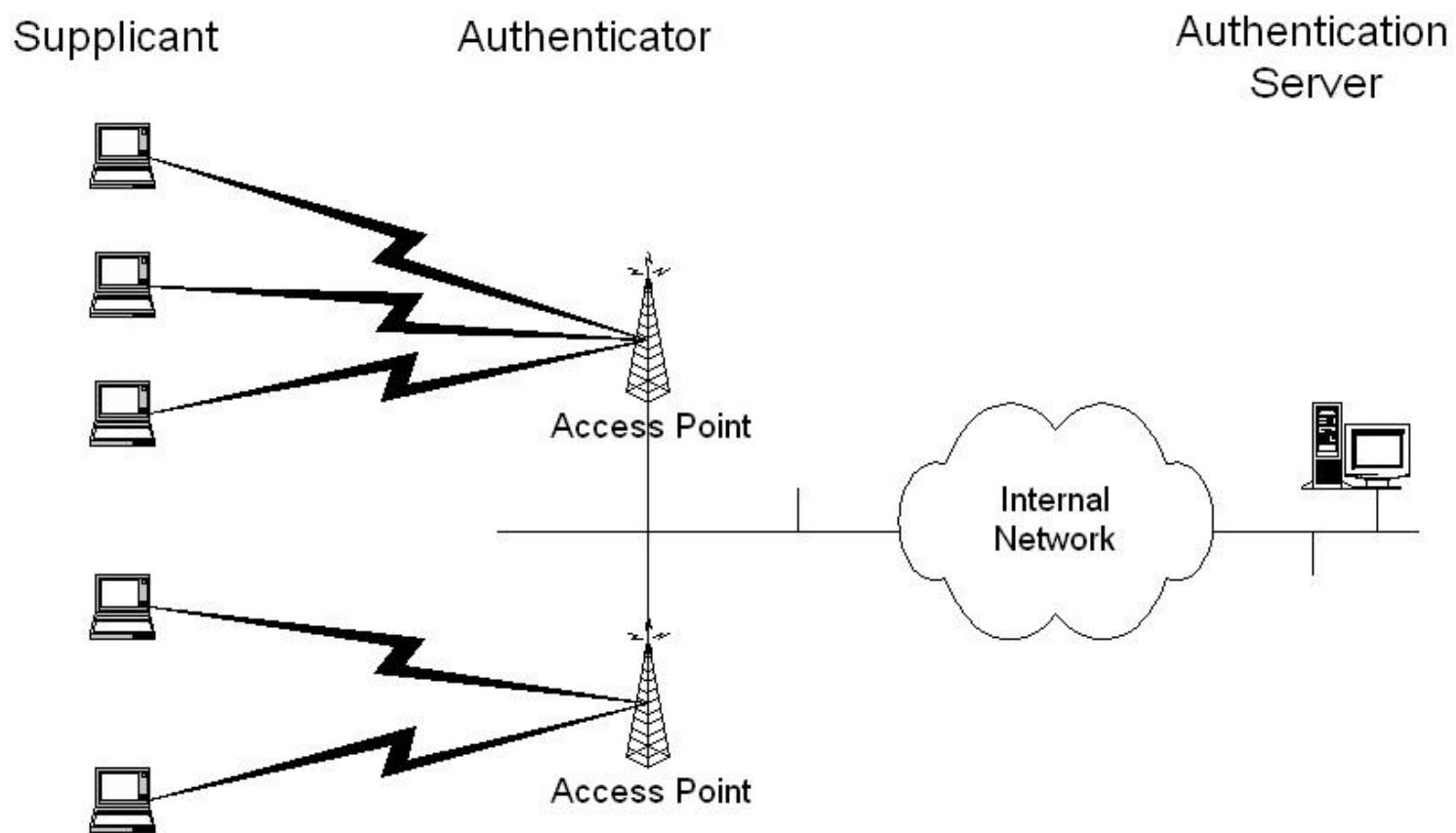
Roadmap

- Secure channels *versus* communication layers
- Wireless networks
- Wi-Fi / WLANs
 - WEP
 - WPA
 - **802.1X and EAP**

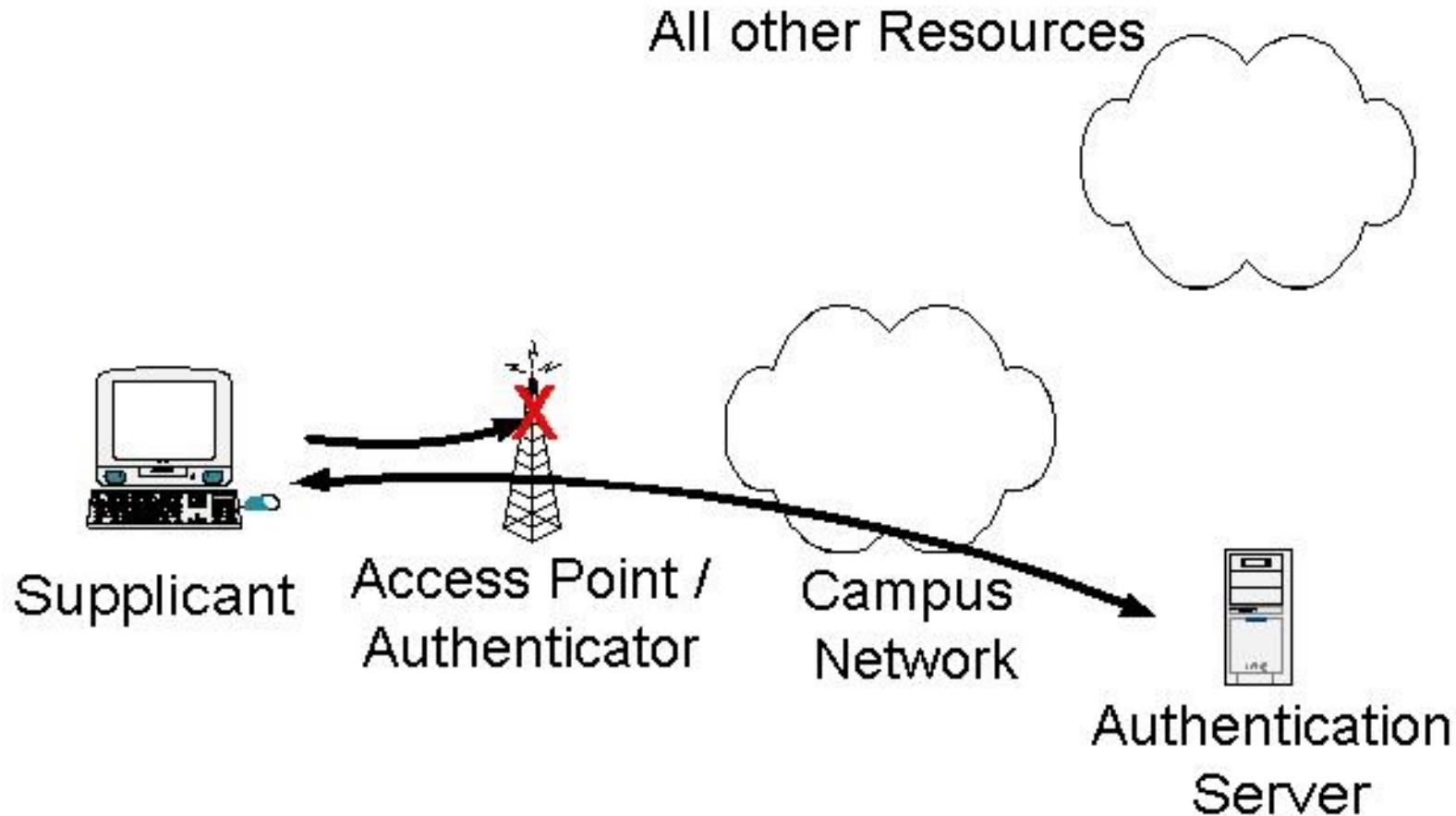
IEEE 802.1X

- Authentication model for IEEE 802 networks
 - Wired and wireless
 - Data link layer mutual authentication
- Standard for **port-based network access control (NAC)**
 - Permits or not an entity to **logically** connect to a port/LAN
 - Logically because physically it already connected somehow
- Originally designed for larger scale networks
 - College campus, etc.
 - Extended model for wireless networks
- Does **authentication + key distribution**

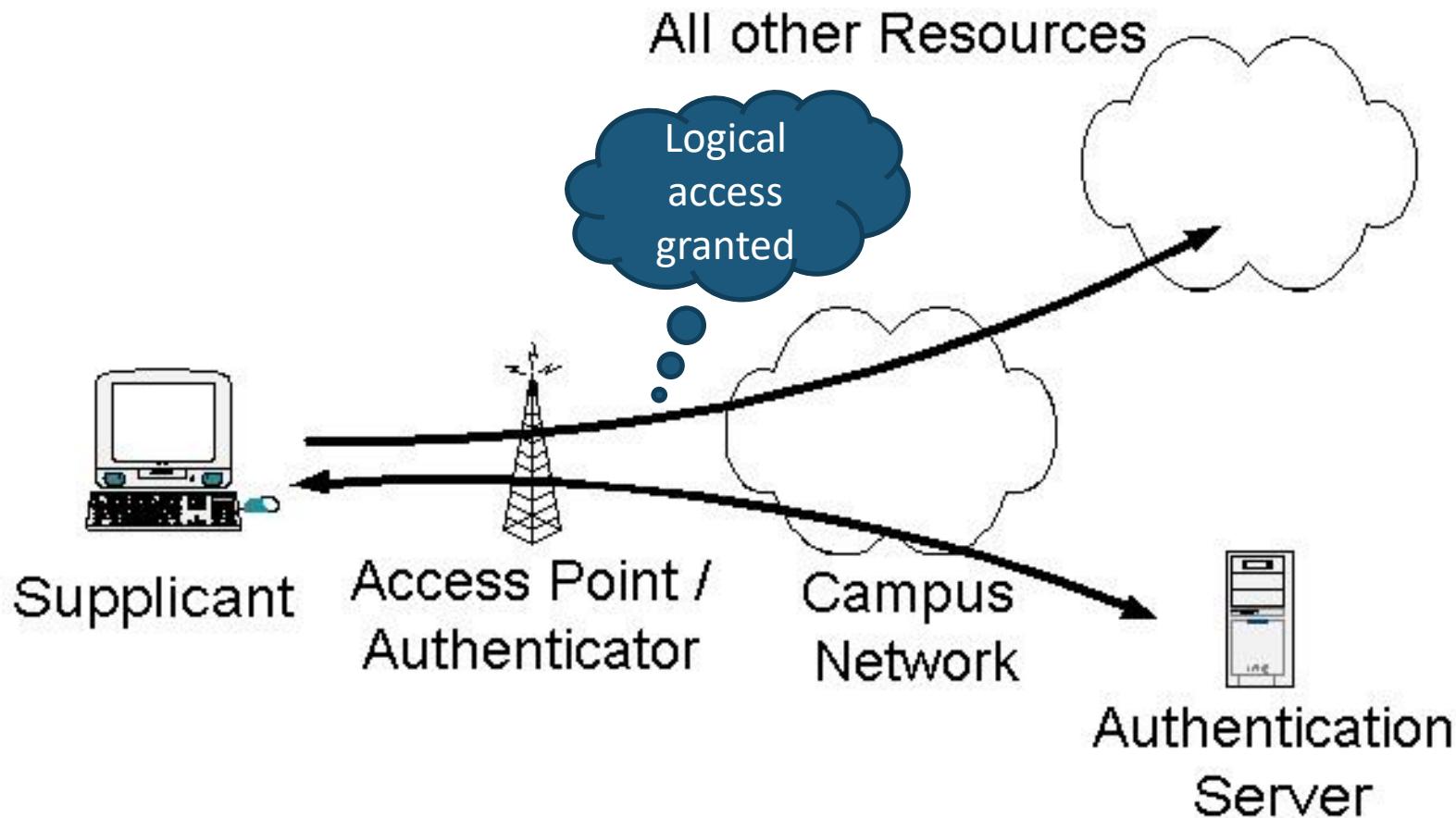
802.1X: Participants



802.1X: Pre-authentication state

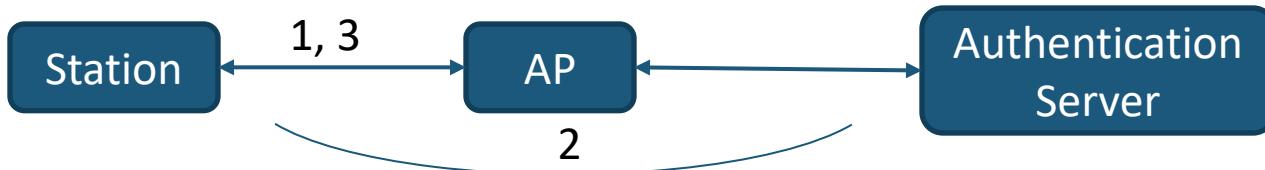


802.1X: Post-authentication state



802.1X stages for wireless networks

1. WEP association between Station and AP
 - The protocol we have seen before: authentication request, etc.
 2. **EAP** (Extensible Authentication Protocol)
 - **Authentication and key establishment** between Station and Authentication Server; produces **MSK (Master Session Key)**
 - It is a meta-protocol: there are several variations
 3. Four-way handshake
 - Mutual authentication of Station and AP using **nonces** and **MSK**
 - Derive **Temporary Key (TK)** to be used for secure communication
 - Validation of stage 1 requests and responses
- 802.1X
encapsulates
and extends
EAP for IEEE
802 networks



EAP

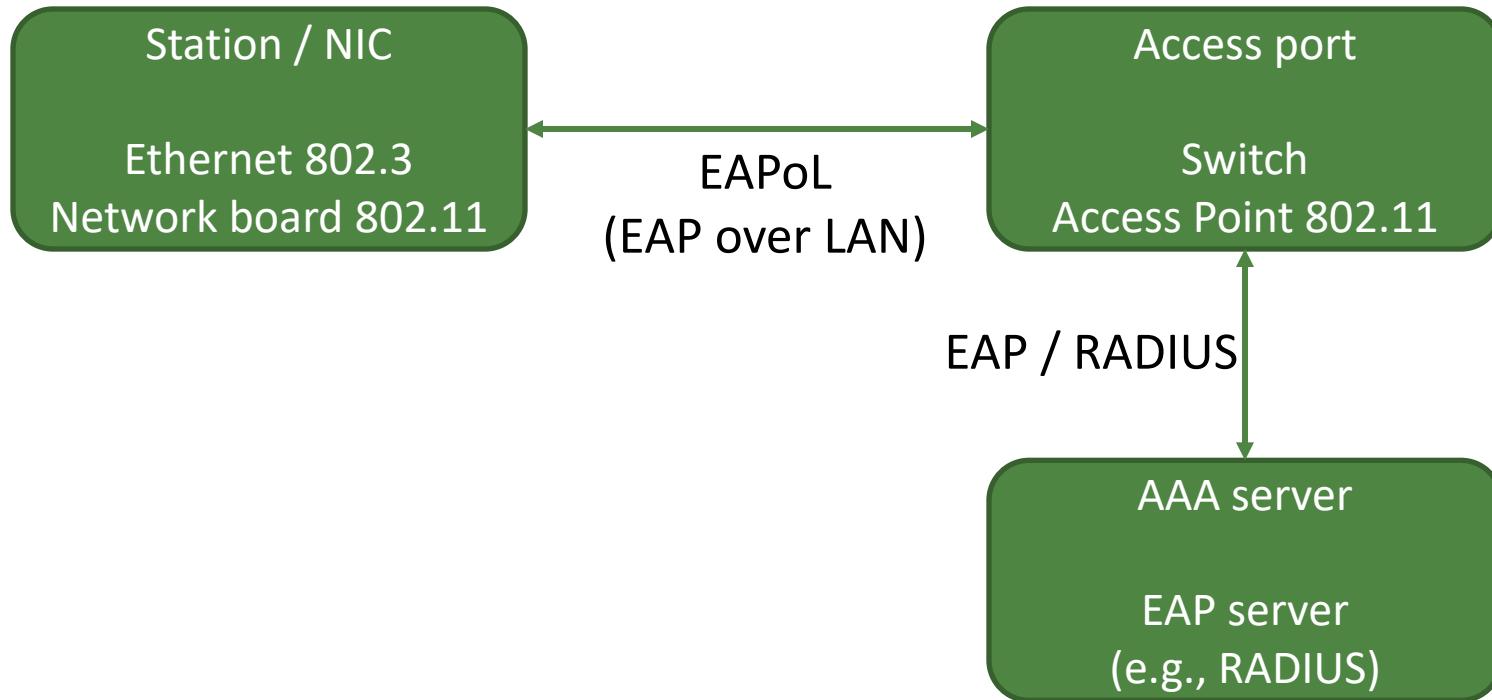
(Extensible Authentication Protocol)

- Initially designed for PPP (Point-to-Point Protocol)
 - Targeted at 802.1 (for wired networks)
- AP is not involved
 - Only allows the passage of EAP messages
 - The use of different authentication protocols does not imply modification to APs
- EAP was not designed for wireless networks
 - The communication between Stations and APs must be protected during EAP with WEP
 - Mutual authentication may not exist
 - A Station can be tricked by a more powerful AP

EAP – Types of requests

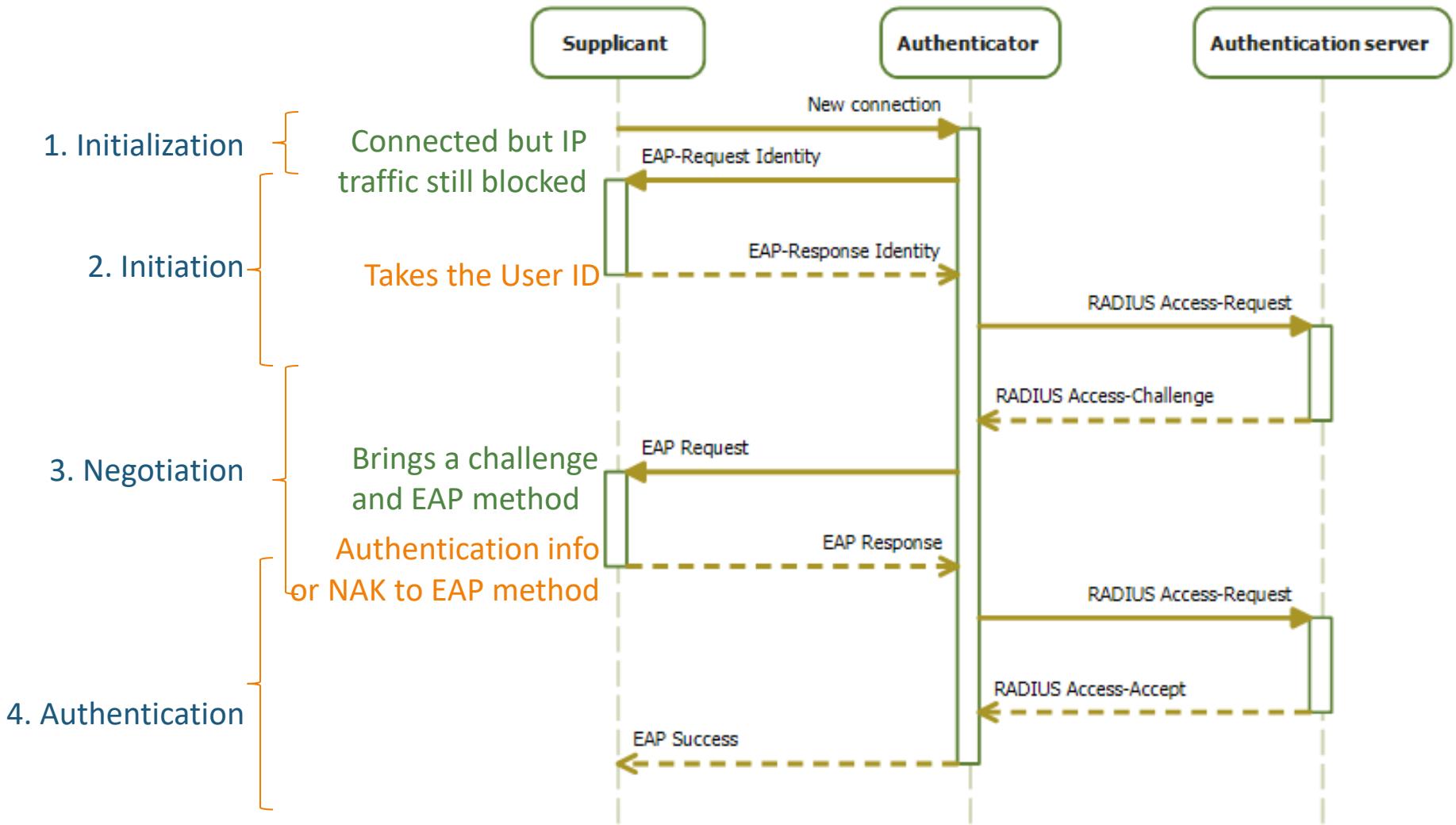
- Type 1: User identity
- Type 2: Message for the user
 - ACK
 - E.g. password about to expire
- Type 3: NAK
- Type 4: MD5 Challenge
- Type 5: One-time password
- Type 6: Cards
 - SecurID, etc.
- Type 13: TLS

802.1X architecture



RADIUS: Remote Authentication Dial-In User Service
SIRS

Distribution of keys with 802.1X



EAP Protocols

- PAP, CHAP
 - Used by PPP; no mutual authentication, only user authentication
 - We have seen them when we talked about Authentication
- EAP-TLS
 - Requires certificates for both parties
- PEAP, EAP-TTLS
 - Both use TLS tunnels
- LEAP
 - Proprietary (CISCO)

Summary

- Wireless networks
- Wi-Fi / WLANs
 - WEP
 - WPA
 - 802.1X and EAP

IPsec and VPNs

Segurança Informática em Redes e Sistemas
2024/25

Ricardo Chaves, David R. Matos

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Roadmap

- VPNs and tunneling
- IPsec
 - Main mechanisms
 - Key distribution

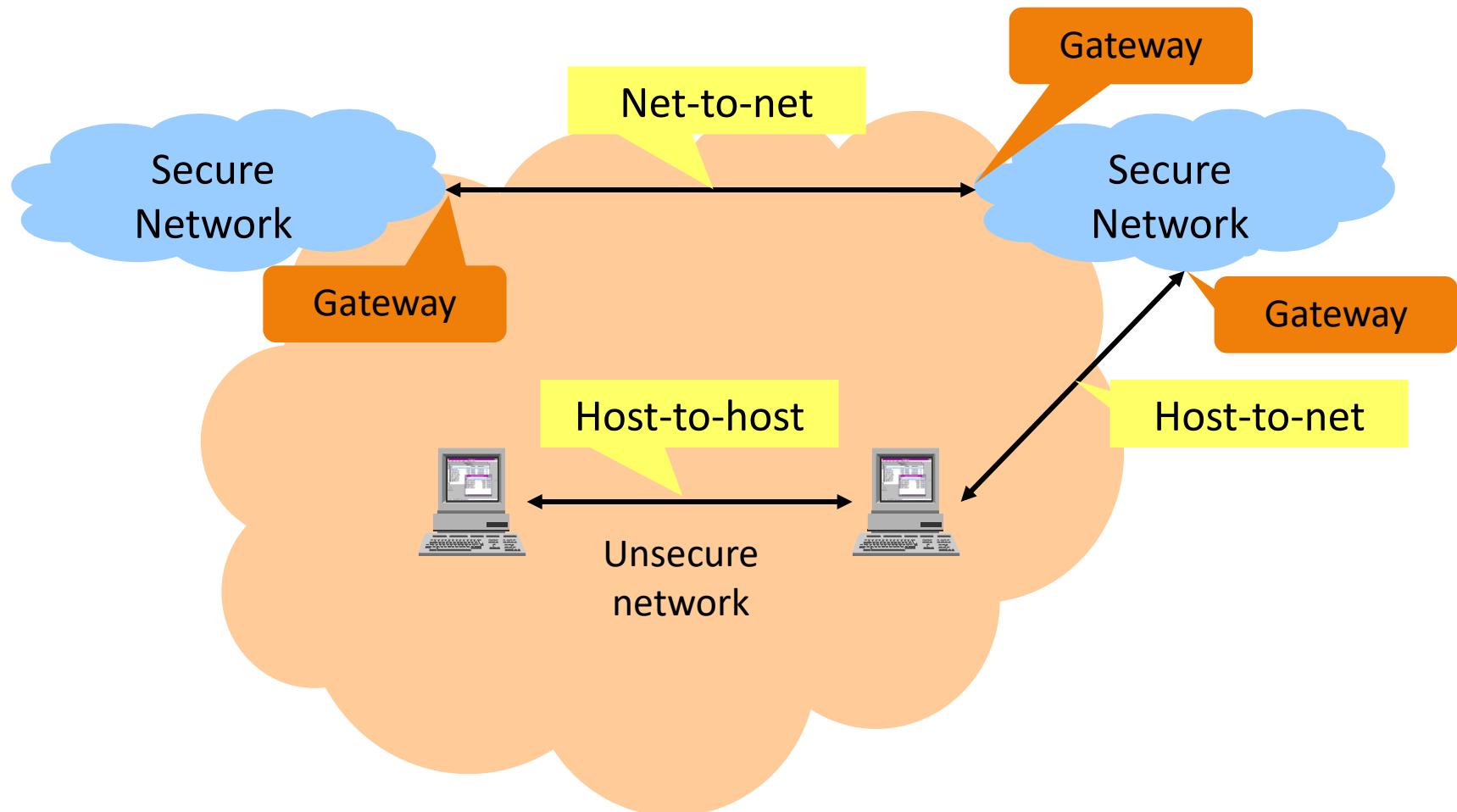
Roadmap

- **VPNs and tunneling**
- IPsec
 - Main mechanisms
 - Key distribution

VPN (Virtual Private Network)

- A **VPN** extends a private network across a public network
 - Enables users to send and receive data across shared or public networks
 - **As if** their computing devices were **directly connected** to the private network
- Applications running across a VPN may benefit from the functionality, security, and management of the private network

VPN usage modes



Tunneling

- Objective
 - Encapsulating a protocol inside another protocol

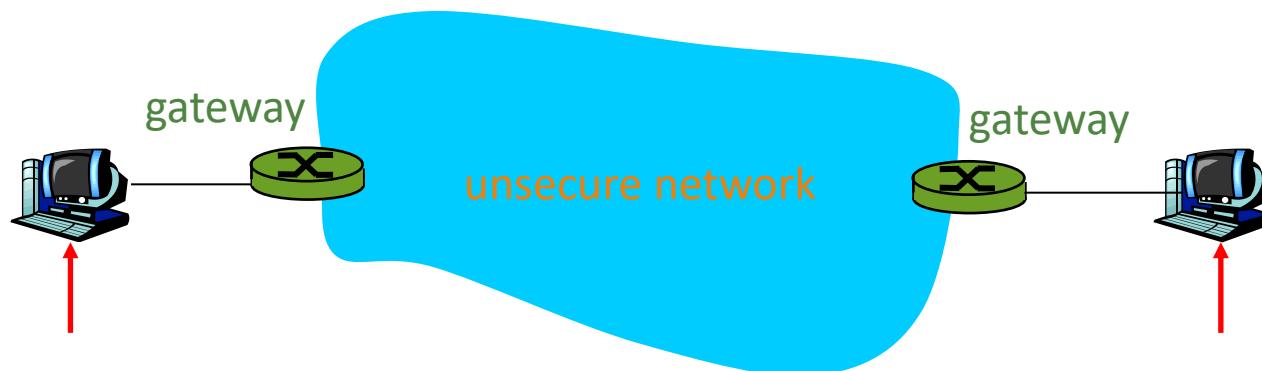


- Security benefits
 - Allows to associate security to the tunneling operation
 - Without interfering with the management infrastructure of the core protocol



VPN Gateway

- Communication unprotected in **internal networks**
 - Where it is assumed that no risk exists
- When data flows into an **unsecure network**, a **VPN gateway** transforms it to assure security
- Additional benefits
 - Intermediate nodes, e.g., Internet routers, can be unaware of secure channel
 - Firewall between gateway and node has access to data in cleartext



A VPN can hide traffic routes

- The traffic can be redirected by a gateway to a final destination
- A VPN is a type of **overlay network**
 - Virtual network built on top of existing network infrastructure
 - Enables custom routing, independent of the physical network layout
 - The VPN tunnel obscures the data's original routing information
 - Providing more security and privacy

Roadmap

- VPNs and tunneling
- IPsec
 - **Main mechanisms**
 - Key distribution

Secure communication: Network layer

Layers		Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS SSH	
	Presentation				
	Session				
	Transport	Operating Systems	Link security	TLS	
	Network	Devices		IPsec	
	Link			IEEE 802.11*	
	Physical				

IP security

- Goals
- Operational scenarios
- Gateway
- Establishment of SA (Security Associations)

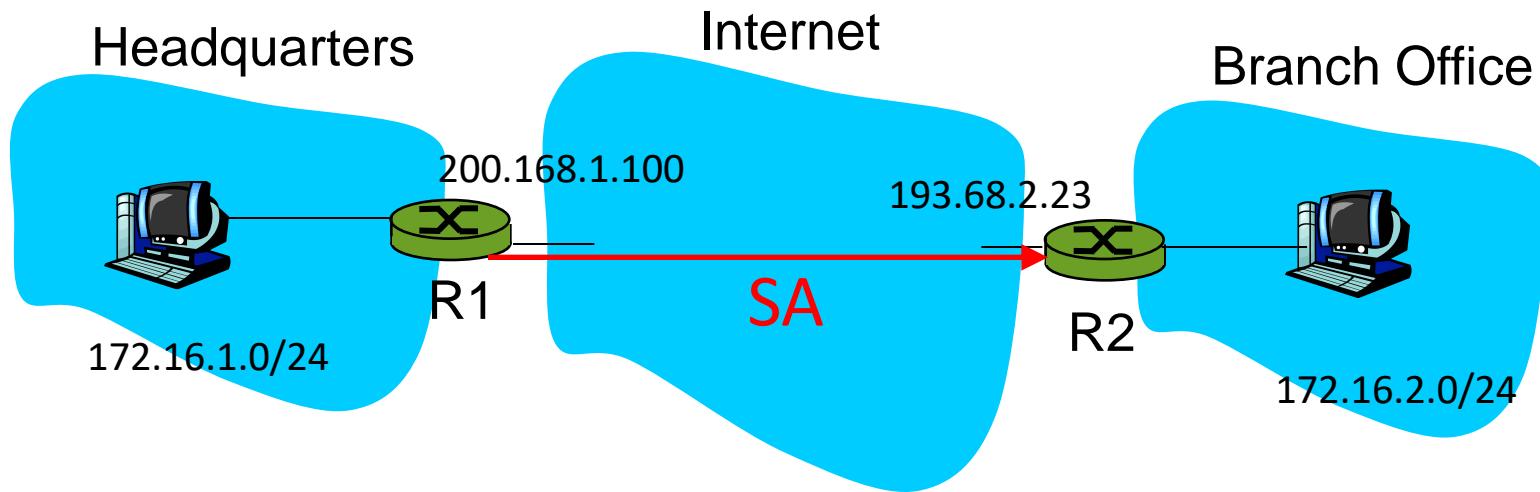
IPsec goals

- Protect all IP traffic between nodes:
confidentiality, integrity, freshness, authenticity
- Two modes of operations:
 - **Transport Mode**: the original IP header is used for routing
 - **Tunnel Mode**: the original IP header is encapsulated
- Benefits
 - Provides secure channels
 - Only affects the network layer; transparent to higher-layer protocols
 - Avoids the update/modification of existing distributed applications
 - Uses the existing IP routing infrastructure; IP header is in plaintext
 - Security between endpoints is independent of the ISPs

SAs (Security Associations)

- Before sending data, **security associations (SAs)** are established at sending and receiving entity
- SAs are **simplex**, i.e., one for each direction
- Gateways maintain *state information* about SAs
 - IP is connectionless; **IPsec is connection-oriented!**

Example SA from R1 to R2



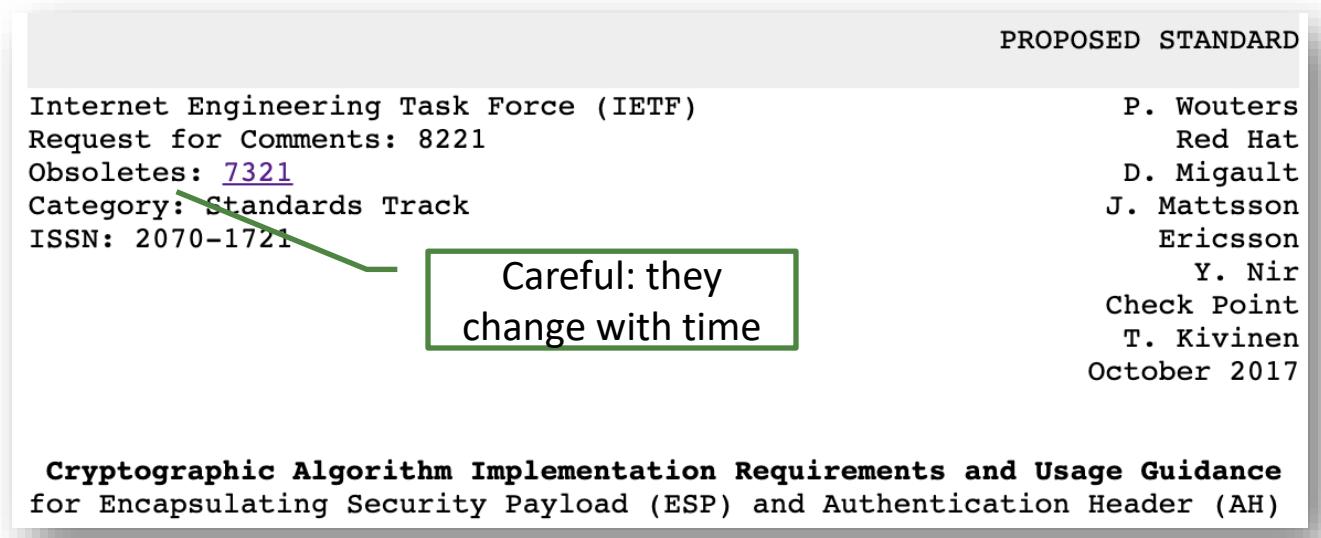
Router 1 stores for an SA:

- 32-bit SA identifier called **SPI (Security Parameter Index)**
- Origin SA interface IP address (200.168.1.100)
- Destination SA interface IP address (193.68.2.23)
- Type of encryption used (e.g., AES with CBC)
- Encryption key
- Type of integrity check used (e.g., HMAC with SHA-2)
- Authentication key (for obtaining HMACs)

SAD (Security Association Database)

- Endpoint holds SA state in the **SAD database**, where it can locate them during processing
 - The SAD is indexed using the **SPI (Security Parameter Index)**
- When **sending** IPsec datagram:
 - R1 accesses SAD to determine how to process the datagram
- When IPsec datagram **arrives** to R2:
 - R2 fetches the **SPI** from the IPsec datagram
 - indexes SAD with the **SPI**
 - and processes datagram accordingly

IPsec cryptographic techniques



- RFC 8221 (Oct 2017) essentially mandates:
 - ESP Encryption Algorithms: AES, ChaCha20
 - ESP and AH Authentication Algorithms: HMAC w/SHA-2
- In fact, it is more complicated:
 - It defines specific configurations of the algorithms
 - It makes “should”, “must”, “must not” statements

IPsec sub-protocols

- IPsec has two sub-protocols: AH and ESP
- **Authentication Header (AH)**
 - Can provide source authentication, data integrity, freshness, but *not* confidentiality
- **Encapsulating Security Payload (ESP)**
 - Can provide source authentication, data integrity, freshness, *and* confidentiality

AH vs ESP

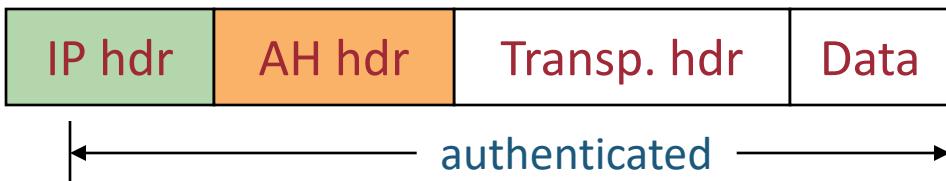
- **Authentication Header**
 - Can prevent IP spoofing (packet tampering) but **not** eavesdropping
 - Simpler and lower overhead than ESP
- **Encapsulating Security Payload**
 - Can prevent IP spoofing (packet tampering) and eavesdropping
 - ESP provides encryption and optional authentication

IPsec AH (Authentication Header)

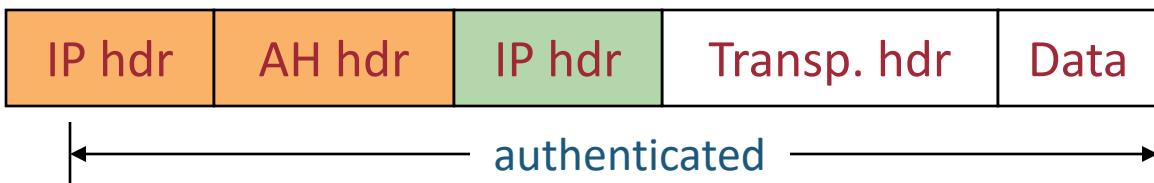
Original packet



Transport mode



Tunnel mode



AH header

Next Hdr	Payload Length	
SPI		
Sequence Number		
Authentication Data		

AH provides source authentication, data integrity, freshness, but not confidentiality

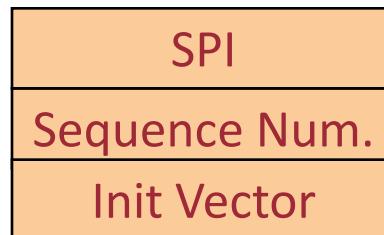
Mostly protected but not totally:
TTL and checksum are not protected

IPsec ESP (Encapsulating Security Payload)

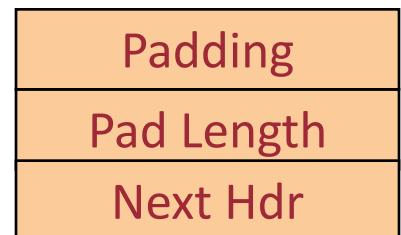
Original packet



ESP header



ESP trail

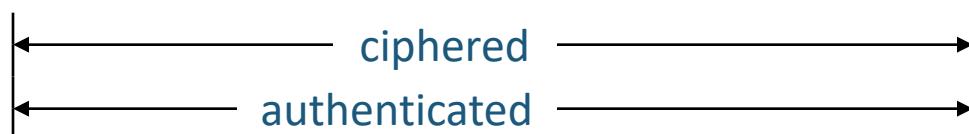


Transport mode

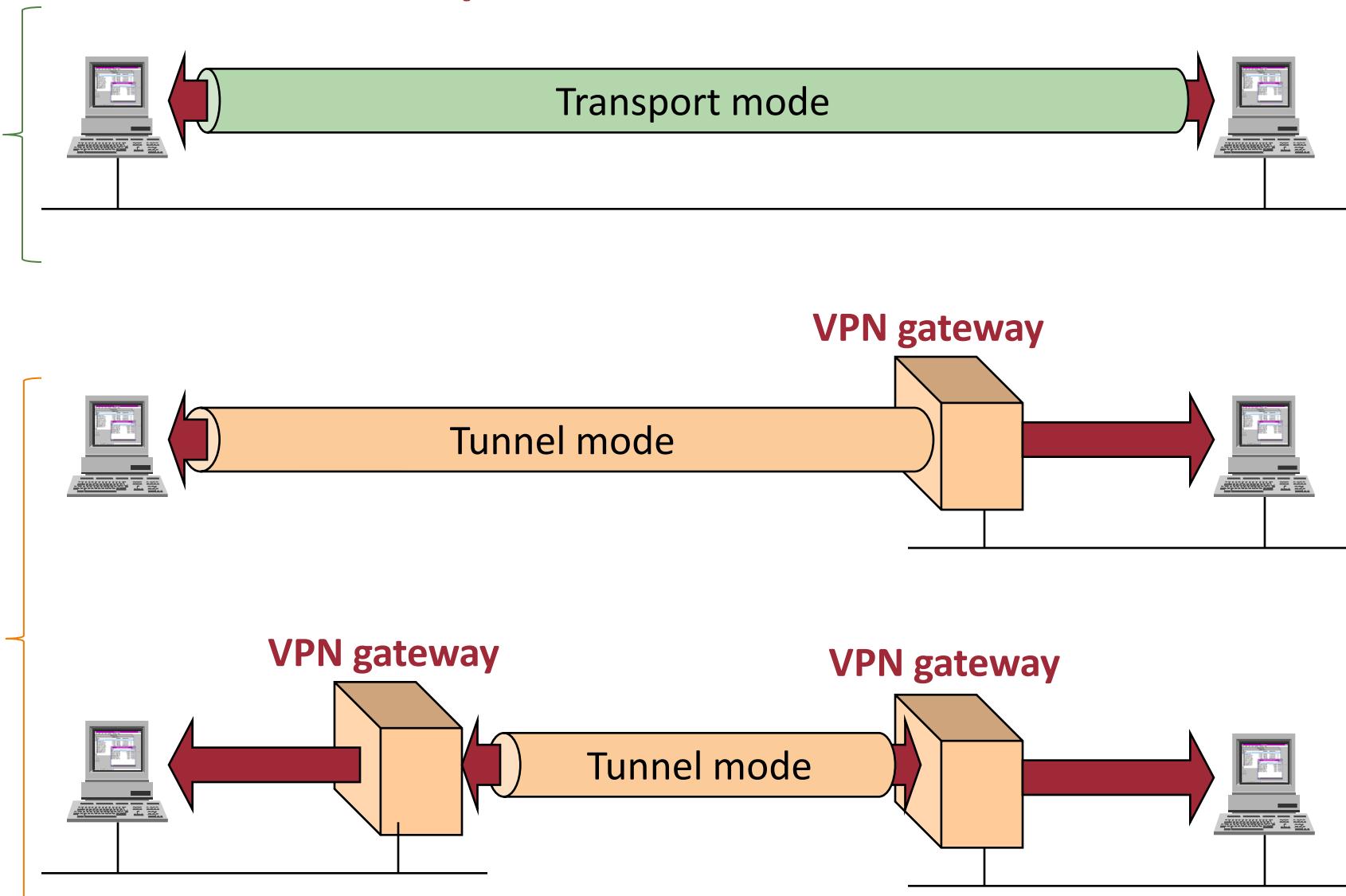


ESP provides source authentication, data integrity, freshness, and confidentiality

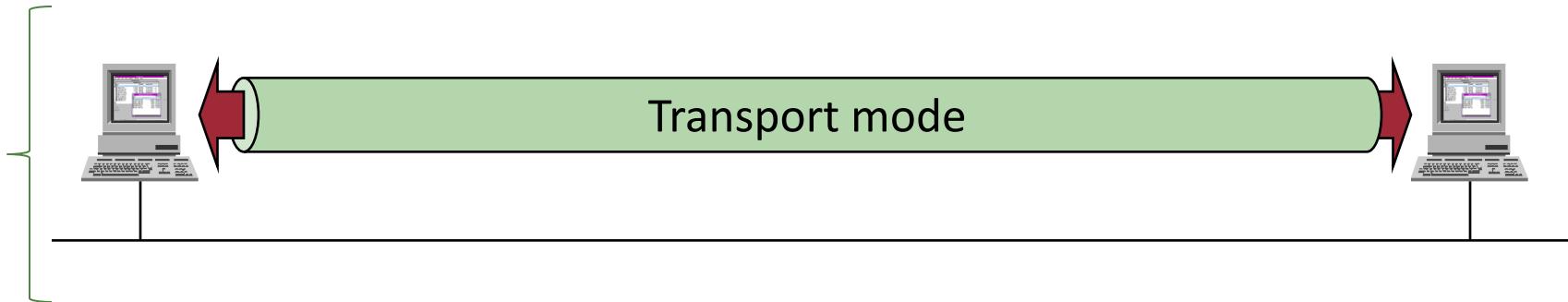
Tunnel mode



IPsec operational scenarios



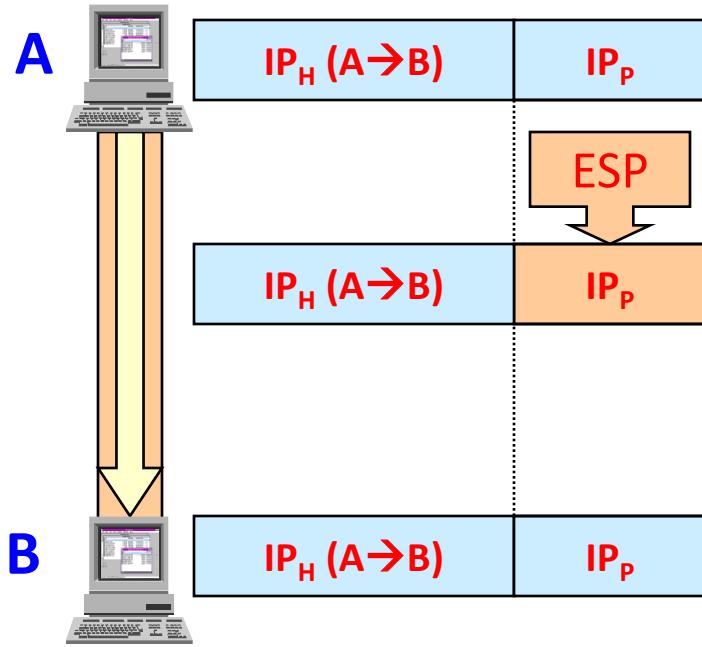
IPsec operational scenarios: host-to-host



ESP Transport Mode

- **Transport Mode:** the original IP header is used for routing
- ESP is deciphered by the destination host
 - Deciphers the ESP and reconstructs the original datagram
- ESP auth (MAC) is verified by the destination host
- SA in use is indexed by the SPI of the datagram

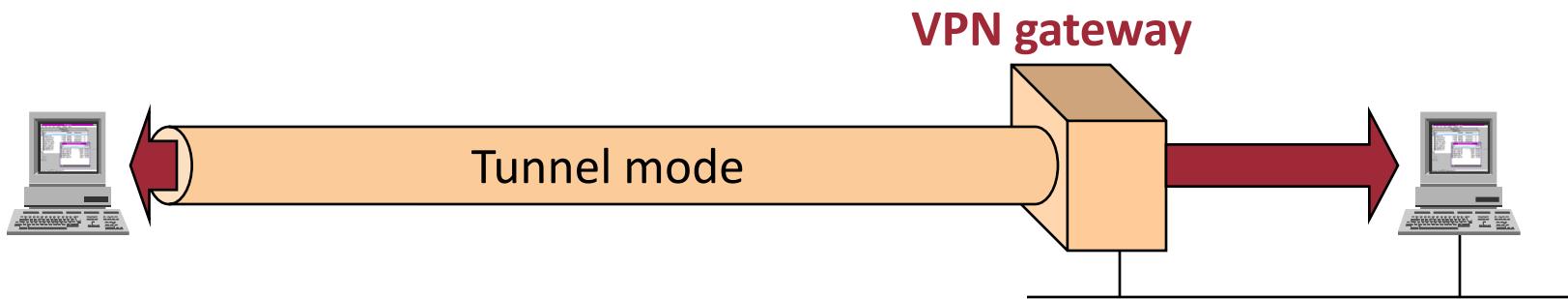
Host-to-host using ESP Transport Mode



IP_H – IP header
IP_P – IP payload

- **Host-to-host**
 - IPsec security exists between A and B
 - Both machines need to know how to use IPsec
- Machine A and B take the initiative of using IPsec between them
 - Two SAs must exist between A and B

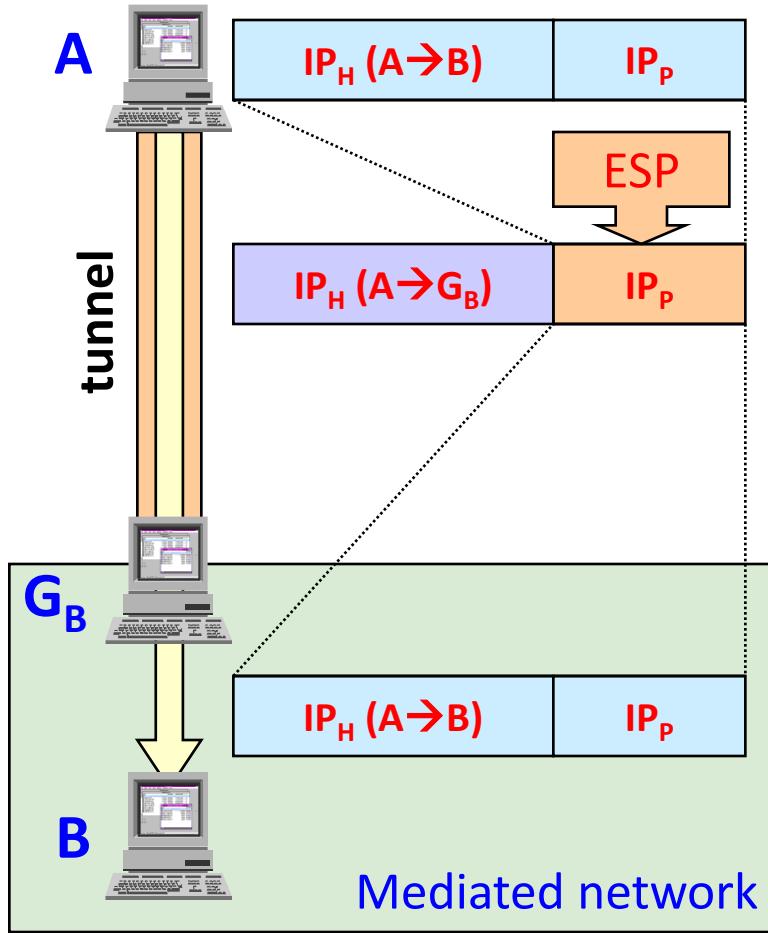
IPsec operational scenarios: host-to-net



ESP Tunnel Mode with gateway

- **Encapsulating Security Payload (ESP)** created with a **full IP datagram**
- Operation
 - Sender or gateway generates ESP with datagram addressed to machine B
 - The final datagram is sent to the gateway of B
 - Host IP addresses may be private
 - The ESP is deciphered; integrity is checked by the gateway of B, then sent to B
- Additional advantages
 - Conceals the real IP addresses from the intermediate nodes
- Disadvantages
 - Increases the IP packet size

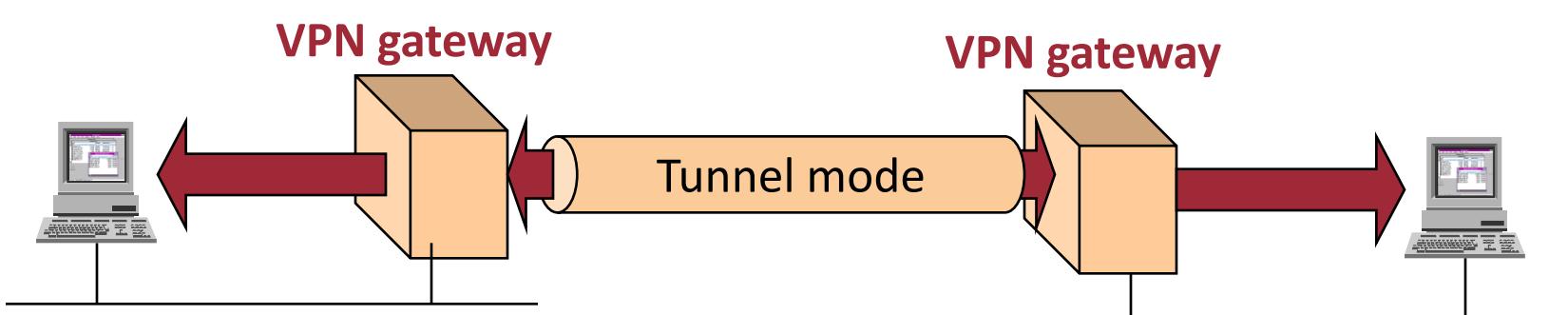
Host-to-net VPN using ESP Tunnel Mode



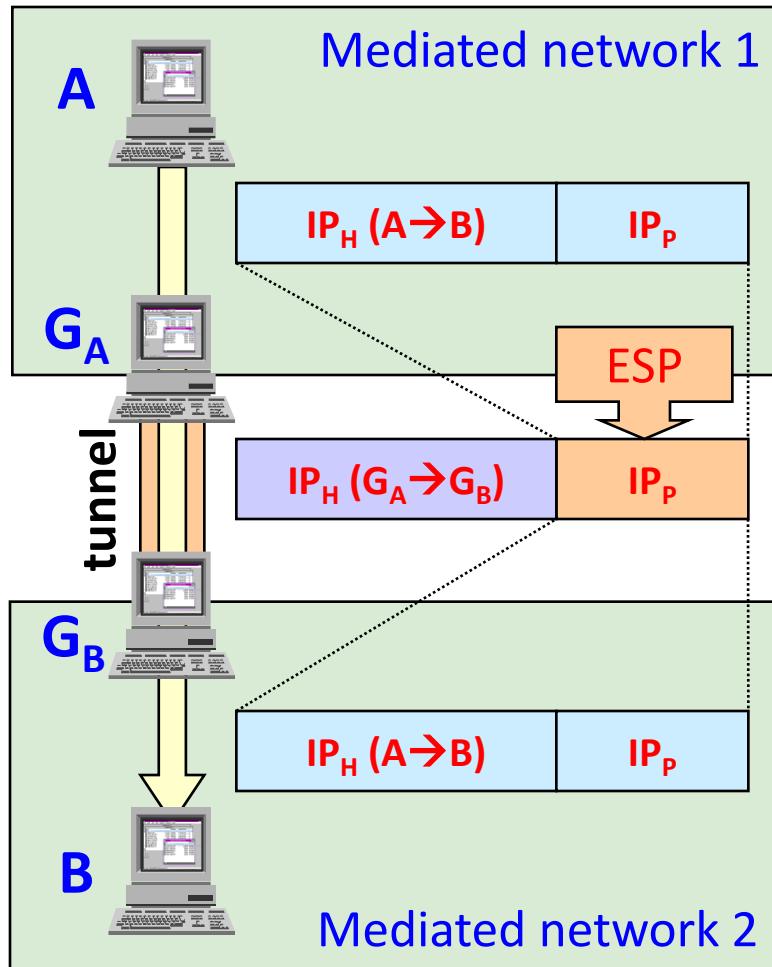
IP_H – IP header
 IP_P – IP payload

- **Host-to-net VPN**
 - IPsec security exists between A and gateway G_B
 - Machine B does not need to know how to use IPsec
 - The address of B does not have to be public
- Host A and gateway G_B take the initiative of using IPsec tunneling between them
 - Two SAs must exist between A and G_B

IPsec operational scenarios: net-to-net



Net-to-net VPN using ESP Tunnel Mode



IP_H – IP header
IP_P – IP payload

- **Net-to-net VPN**
 - IPsec security exists between gateways G_A and G_B
 - Machines A and B do not need to know how to use IPsec
 - The IP addresses of A and B do not have to be public
- Gateways G_A and G_B take the initiative of using an IPsec tunnel between them
 - Two SAs must exist between them

IPsec limitations

- **Performance**
 - All the dataflow between two nodes is protected
 - Even if not needed
 - Tunneling size overhead
 - More headers, more processing
- **Security quality**
 - Some messages are more critical than others
 - Depending on the applications and the users
 - There may be need of assuring security at higher layers
 - Leading to a duplication of effort
- **Key management**
 - There is no unique key management protocol
 - The most common solution is manual distribution

Roadmap

- VPNs and tunneling
- IPsec
 - Main mechanisms
 - Key distribution

IPsec SA establishment (1/2)

- **Security Association (SA)**
 - Security policy, cryptographic mechanisms, and parameters used in the secure communication between a pair of machines
 - Security Parameter Index (SPI)
- Problem
 - How to create common SAs between a pair of machines?
 - Which is the SPI of those SAs?

IPsec SA establishment (2/2)

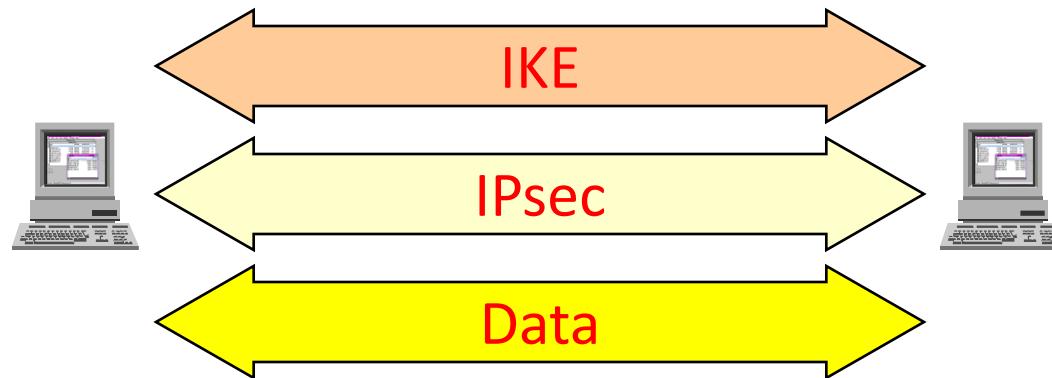
- Manual establishment
 - Manual configuration of the SAs
 - Manual attribution of SAs to IPs
- SA establishment with protocols
 - **ISAKMP** – framework
 - **IKE** – the standard protocol (IKEv2)

ISAKMP

- **ISAKMP** – Internet Security Association and Key Management Protocol
- Generic protocol (framework)
 - Allows the negotiation of keys and entity authentication
 - Does not specify any techniques
 - Only defines 5 types of information exchange
- It is an application layer protocol
 - Supports negotiations for several OSI layers
- There is only one protocol that uses it:
 - **IKE: Internet Key Exchange**, the standard for IPSEC
 - but can also be used with TLS

IKE operation

- IKE – Internet Key Exchange
- 1st phase: establishment of a bidirectional **IKE SA**
- 2nd phase: establishment of unidirectional **IPsec SAs**
 - Negotiation protected by the IKE SA
 - Several IPsec SAs can be established using the same IKE SA
- Data transmission secured by IPsec
 - Protected by unidirectional IPsec SAs



IKE negotiation modes

- Main mode (1st phase)
 - Establishment of a bidirectional IKE SA
 - Instance of the identity exchange of the ISAKMP
 - Ciphered identities (machine names)
 - The IKE SA can be negotiated by a third party rather than the machines using them
- Quick mode (2nd phase)
 - Establishment of two IPsec SAs
 - One for the outgoing data,
 - Another one for the incoming data (with its SPI)
 - Protected by an IKE SA
 - Generates a new key with DH (Diffie-Hellman) or refreshes the previous key

IKE: host authentication

- 1st Phase: two alternatives:
 - With **asymmetric cryptography**
 - DSA/RSA signatures and X.509 digital certificates
 - With a **shared secret**
 - Pre-shared key (PSK) – secret key, established manually
- 2nd Phase
 - With **shared secret only**
 - Uses secret established in the first phase

Roadmap

- IPsec
 - Main mechanisms
 - Key distribution
- **VPNs and tunneling**

Summary

- IPsec
 - Main mechanisms
 - Key distribution
- VPNs and tunneling

TLS in depth

Segurança Informática em Redes e Sistemas
2024/25

Ricardo Chaves, David R. Matos

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Secure communication: *Transport layer and above*

	Layers	Responsibility	Approach	Solutions
OSI Layers	Transaction	Local data manipulation applications		PGP, PEM, S/MIME
	Application	Applications for remote data exchange	End-to-end security	HTTPS, IMAPS, SSH
	Presentation			
	Session			
	Transport	Operating Systems		TLS
	Network			IPsec
	Link	Devices	Link security	IEEE 802.11*
	Physical			

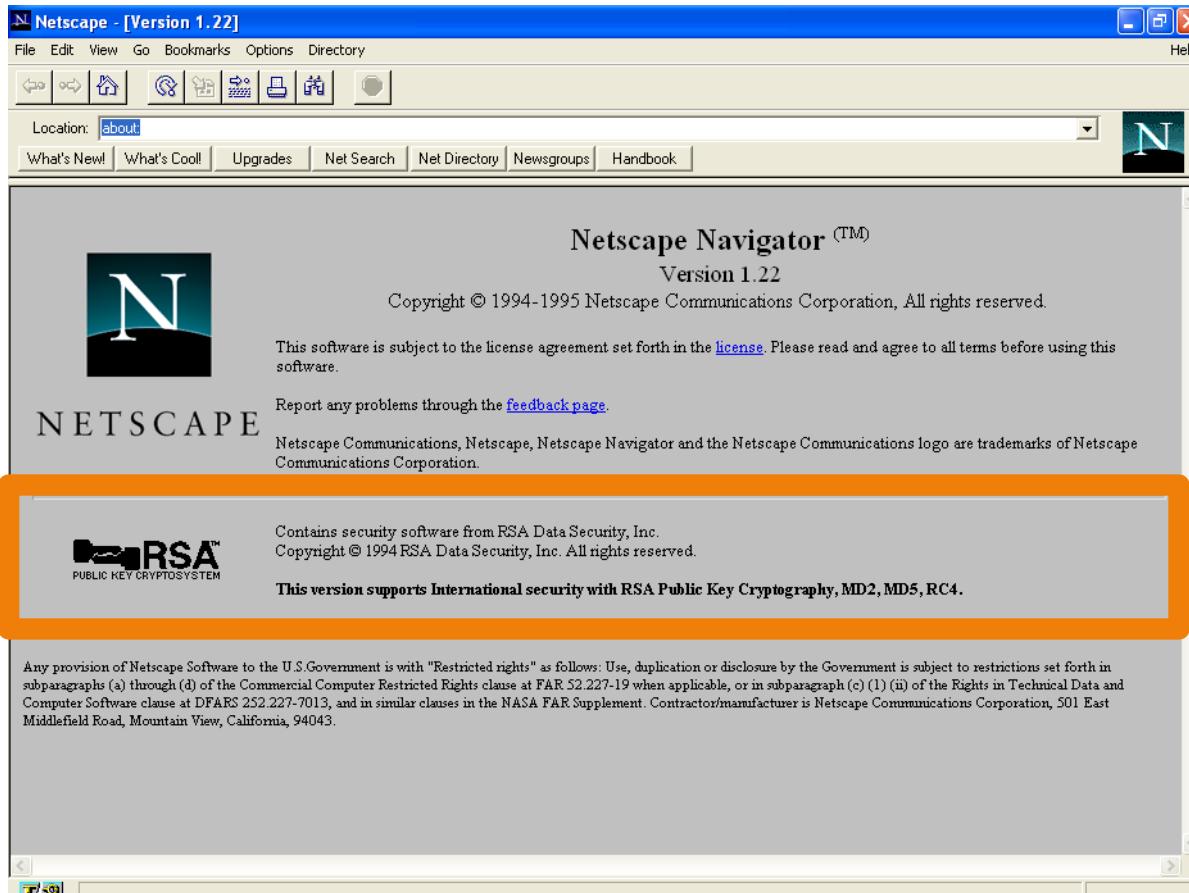
Roadmap

- TLS – Transport Layer Security
 - HTTPS

HTTPS

- **HTTPS = HTTP over SSL (now TLS)**
 - In fact, SSL was originally designed to be used with HTTP
 - Netscape Navigator (1994)
 - Forefather of Mozilla Firefox
 - Version history
 - SSL 1.0 1994
 - SSL 2.0 1995
 - SSL 3.0 1996
 - TLS 1.0 1999
 - TLS 1.1 2006
 - TLS 1.2 2008
 - TLS 1.3 2018

Netscape Navigator



- A few months later, in July 1995, a computer programmer ordered the first book ever sold by an online bookstore named after a river in South America...

Secure communication: *Transport layer*

Layers		Responsibility	Approach	Solutions	
OSI Layers	Transaction	Local data manipulation applications	End-to-end security	PGP, PEM, S/MIME	
	Application	Applications for remote data exchange		HTTPS, IMAPS, SSH	
	Presentation				
	Session				
	Transport	Operating Systems		TLS	
	Network	Devices		IPsec	
	Link	Link security	IEEE 802.11*		
	Physical				

TLS goals

- Secure communication channels over TCP/IP
 - Current version: **TLS 1.3** – august 2018
 - Standard based on the deprecated **SSL (Secure Sockets Layer)**
 - Sometimes called **SSL** for that reason
 - Manages secure sessions over **TCP/IP** per application
 - Initially designed for the HTTP protocol (HTTPS)
 - Currently used by other protocols, e.g., SMTP, IMAP, POP3
- Security mechanisms
 - **Authentication** of the communicating parties
 - **Confidentiality** and **integrity** of the communication
 - **Key distribution**

TLS utilization

- TLS is just a protocol; not a standard API
- Common APIs:
 - Reference API for SSL: SSLref (Netscape)
 - Public implementations: OpenSSL, GnuTLS, SSLeay, Mbed TLS, Java Secure Socket Extension (JSSE)
- Remote interfaces:
 - Conventional: protocol/port
 - e.g., TCP/443 for HTTPS
 - STARTTLS: allows upgrading text connection to TLS
 - Defined for SMTP, IMAP, POP, SMTP, FTP, IRC,...

TLS operation management

- Client-Server model as in TCP
- The applications (e.g., web, mail) define the strategy
- **Authentication**
 - If it is needed and how it is performed
- **Cryptographic algorithms**
 - The client presents the **cipher suites** it supports
 - The server selects one
- **Session key management**
 - Lifetime of the **master secret** = lifetime of the **TLS session**
 - Used whenever the client and the server decide to communicate
 - Maximum of 24 hours recommended
 - Lifetime of the **session keys**: at most lifetime of the TCP connection

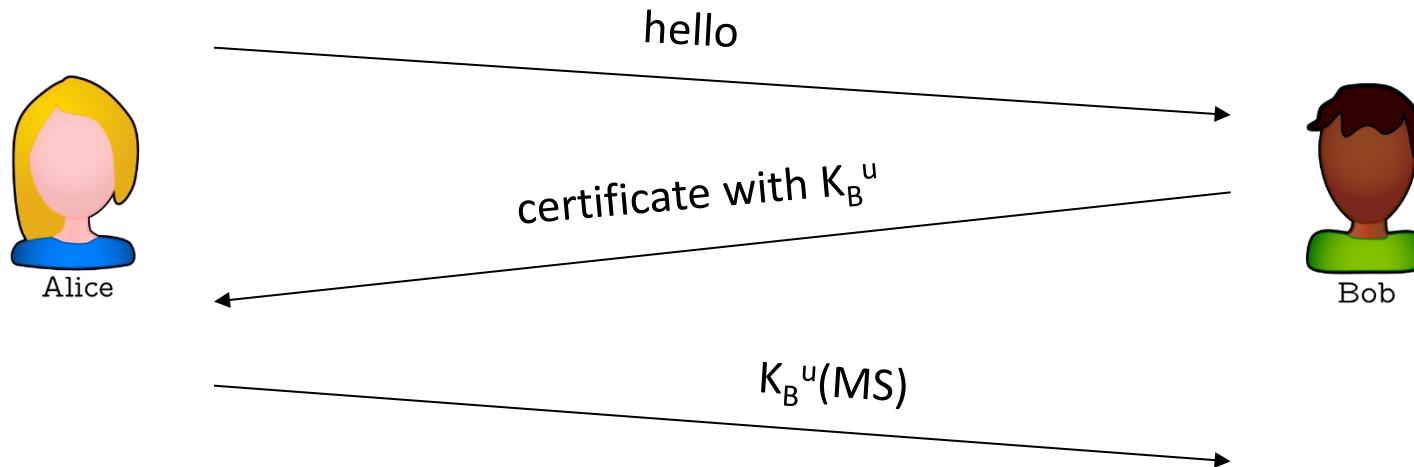
Roadmap

- TLS – Transport Layer Security
 - Toy TLS

Toy TLS: simplified secure channel

- **Handshake**
 - Alice and Bob use their certificates, private keys to authenticate each other and exchange a shared secret
- **Key derivation**
 - Alice and Bob use shared secret to derive set of keys
- **Data transfer**
 - Data to be transferred is broken up into series of records
- **Connection closure**
 - Special messages to securely close connection

Toy TLS: a simple handshake



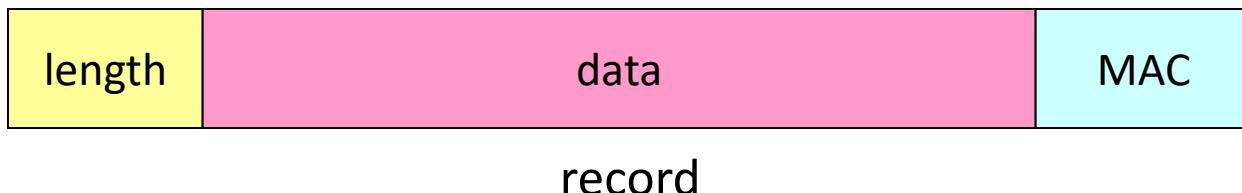
- $K_B^u = \text{Bob's public key}$
- $MS = \text{master secret}$

Toy TLS: key derivation

- It is bad to use same the key for more than 1 cryptographic operation
 - So, use different keys for MACs and encryption
- Therefore **4 keys**:
 - K_c = encryption key for data sent from client to server
 - M_c = MAC key for data sent from client to server
 - K_s = encryption key for data sent from server to client
 - M_s = MAC key for data sent from server to client
- Keys derived using **key derivation function (KDF)**
 - Takes master secret (MS) and additional random data and creates the keys

Toy TLS: data records

- Why not encrypt data in stream as we write it to TCP?
 - Where would we put the MAC? If at end, no message integrity until the end of the connection!
- Instead, break stream in series of **records** (\simeq messages)
 - Each record carries a MAC
 - Receiver can act on each record as it arrives
- Issue: records have variable length and receiver needs to distinguish MAC from data
 - Solution:

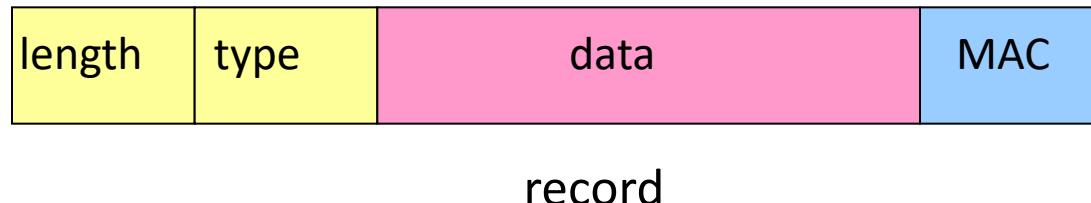


Toy TLS: sequence numbers

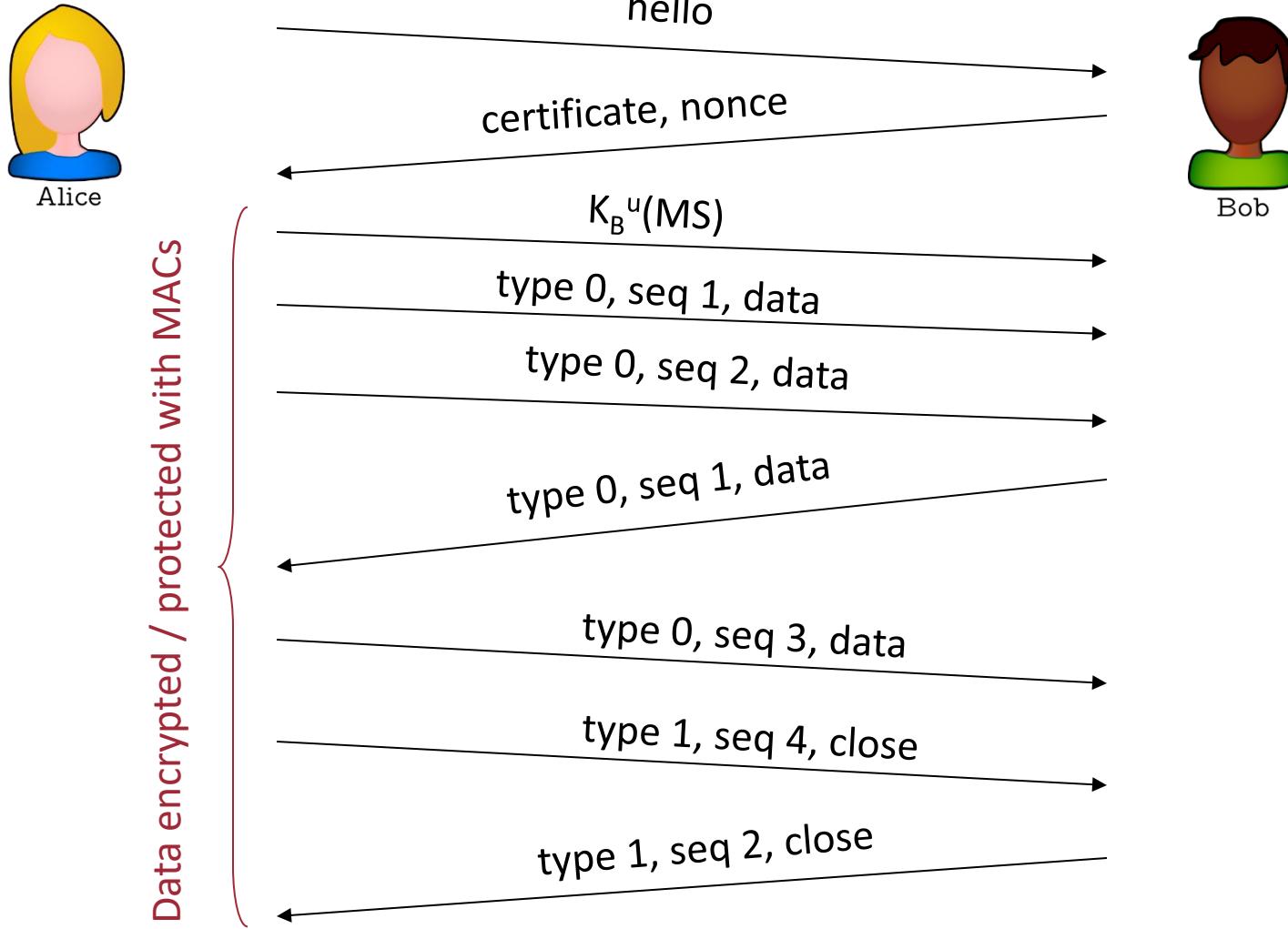
- Attacker can replay or re-order records
- Solution: put sequence number (**seq**) into MAC:
 - **seq** starts at 0; incremented for every record
 - $\text{MAC} = \text{MAC}(M_x, \text{seq} \parallel \text{data})$ key $M_x = M_c$ or M_s
 - Important: there is no sequence number field in the record, so **seq** is implicit, just like the **key** M_x
- Attacker could still replay all of the records
 - Use a **nonce** to prevent this attack

Toy TLS: control information

- Truncation attack
 - Attacker forges TCP connection close segment
 - One or both sides thinks there is less data than there actually is
- Solution: **record types**, with a type for closure
 - type 0 for data; type 1 for closure
- $\text{MAC} = \text{MAC}(M_x, \text{seq} \parallel \text{type} \parallel \text{data})$



Toy TLS: summary



Toy TLS is not complete

- How long are fields?
- Which encryption protocols?
- Negotiation?
 - Allow client and server to support different encryption algorithms
 - Allow client and server to choose together specific algorithm before data transfer

Roadmap

- **TLS – Transport Layer Security**

TLS Cipher Suites

- **Cipher suite**
 - Public-key algorithm
 - Symmetric encryption algo.
 - MAC algorithm
- TLS supports several
- **Negotiation:**
 - Client offers choice
 - Server picks 1
 - e.g., the most secure
- **Common algorithms**
 - Public-key encryption
 - RSA, ECDSA
 - No PQC yet!
 - Symmetric ciphers
 - AES: block
 - ChaCha20: stream
 - Hash functions
 - SHA-2
 - SHA-3
 - Poly1305

TLS protocols

- **Handshake Protocol**
 - Exchange of identity and supported cipher suites
 - Authentication of the communicating parties
 - Client challenges the server, that proves its identity with certificate(s) *[Optional, but mandatory if the next exists]*
 - Server challenges the client, that proves its identity with certificates *[Optional]*
 - Key distribution
- **Record Protocol**
 - Creation and verification of secure messages
 - Compression, cipher, integrity control

Removed in TLS 1.3

TLS handshake (1)

Purpose

1. Server authentication
2. Negotiation: agree on crypto algorithms
3. Establish keys
4. Client authentication (optional)

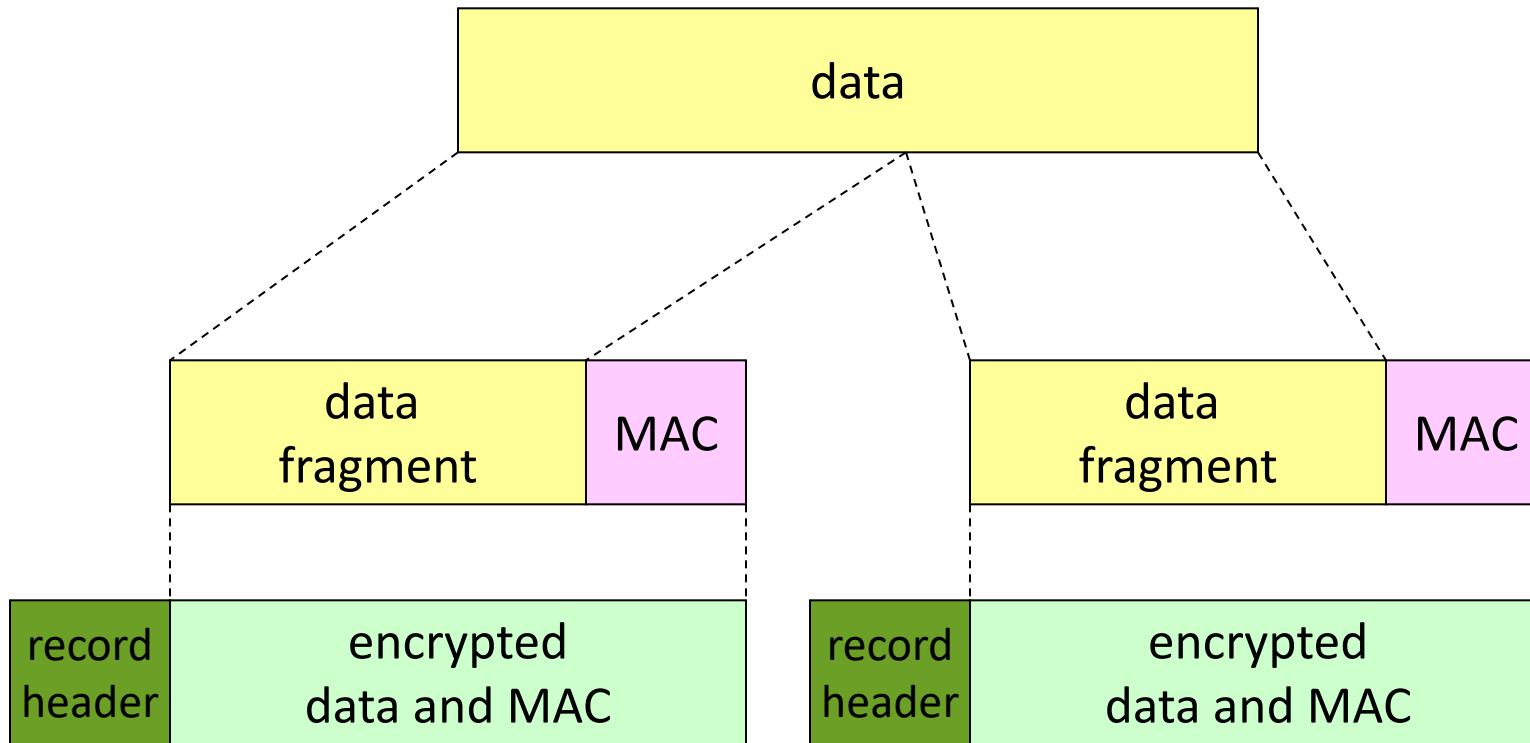
TLS handshake (2)

1. **Client** sends list of algorithms it supports, along with client nonce
2. **Server** chooses algorithms from list; sends back: choice + certificate + server nonce
3. **Client** verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. **Client and server** independently compute encryption and MAC keys from pre_master_secret and nonces
5. **Client** sends a MAC of all the handshake messages
6. **Server** sends a MAC of all the handshake messages

TLS handshake (3)

- Last two steps protect handshake from tampering:
 - Client typically offers range of algorithms, some strong, some weak
 - Man-in-the-middle could delete best algorithms from list
 - Last two message MACs allows detecting such an attack
- Why not simply sign or add MACs to the handshake messages?
 - Not possible: it is the handshake that sets up the keys!

TLS record protocol

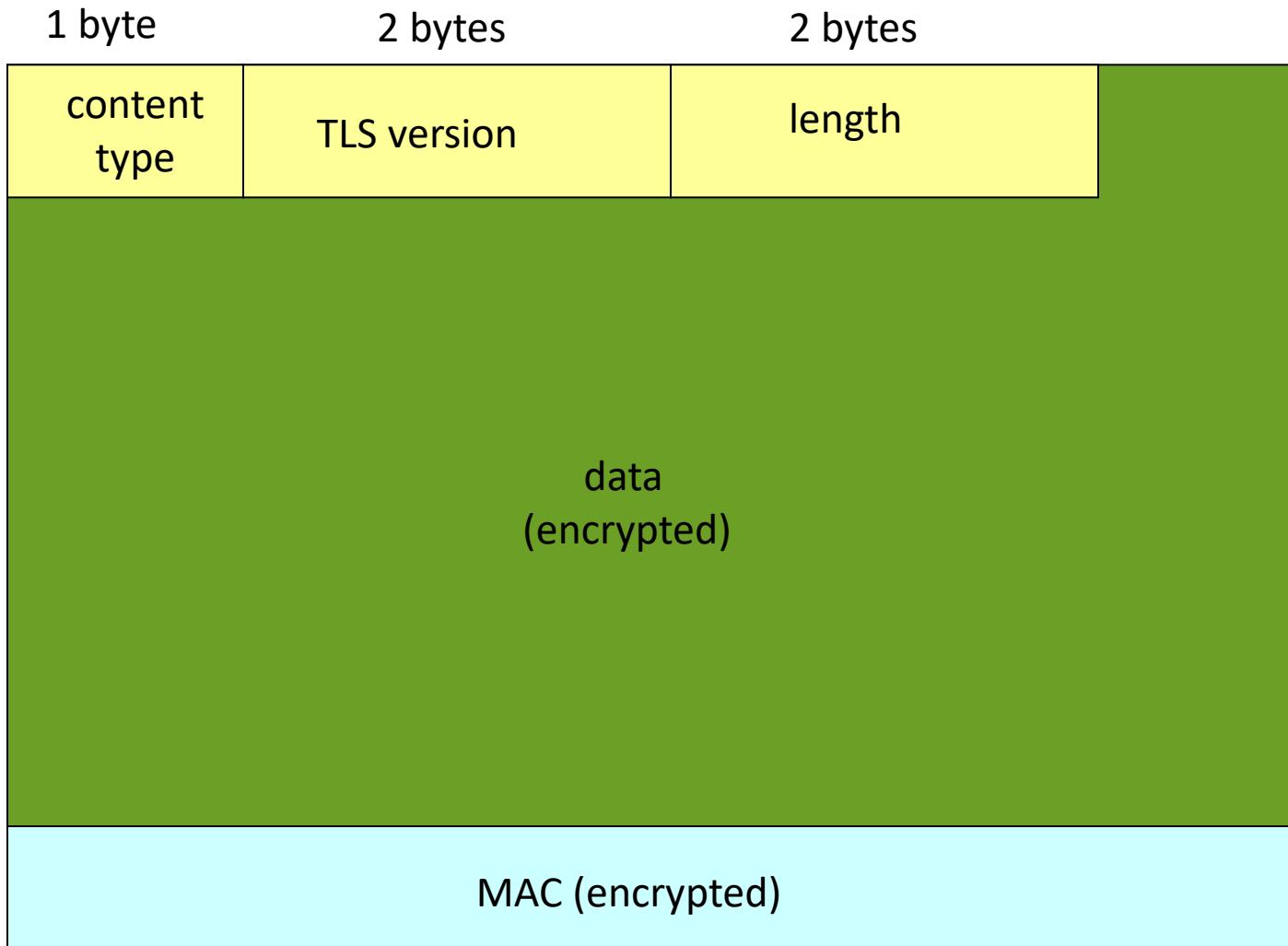


Record header: content type; TLS version; length

MAC: is function of the sequence number and the MAC key M_x

Fragment: each data fragment has up to 2^{14} bytes (~ 16 Kbytes)

TLS record format



TLS key distribution

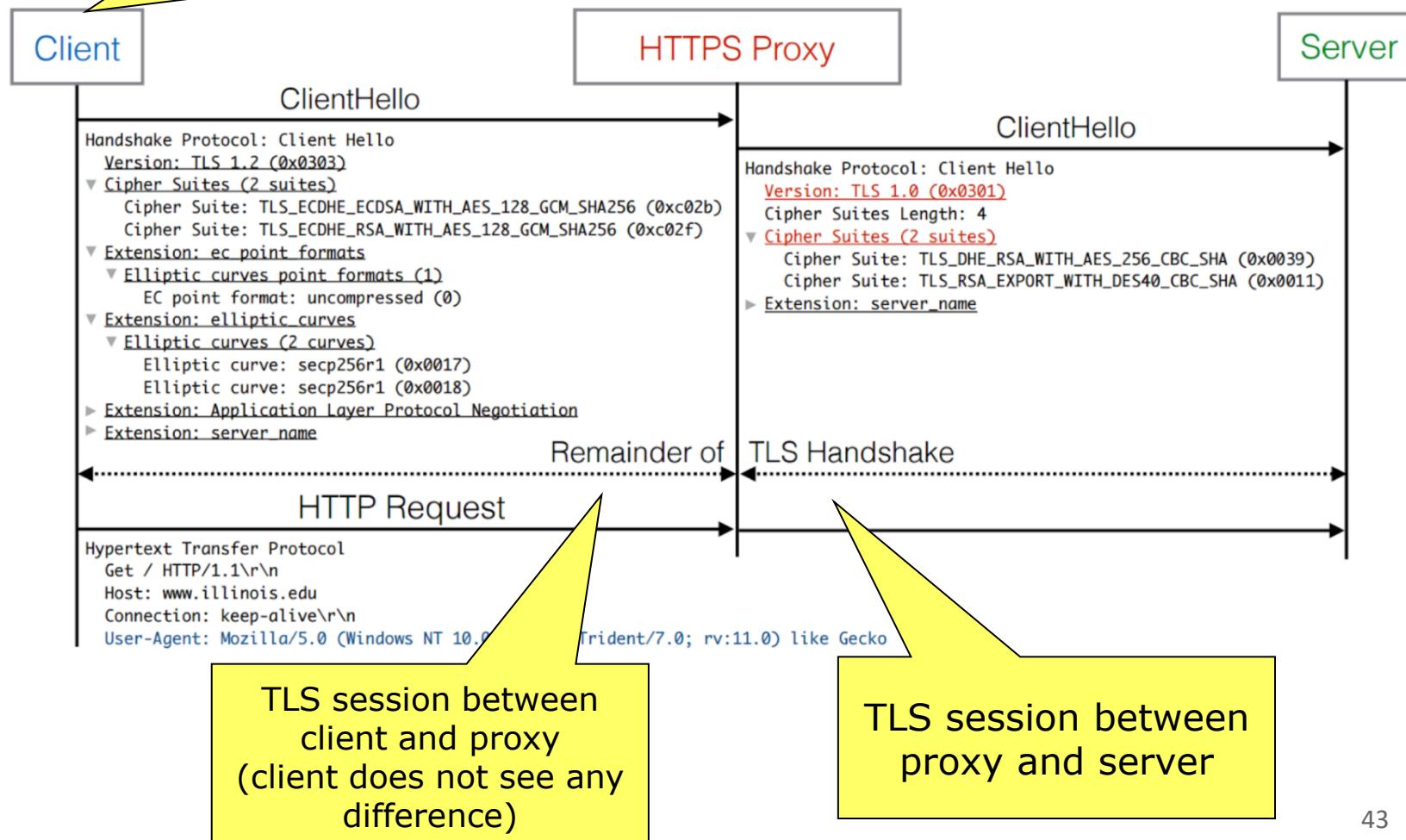
- Master secrets (48 octets)
 - Pre-Master Secret (PMS)
 - Computed with DH; or
 - PMS is chosen by the client and sent to the server ciphered with its public key
 - Master secret (MS)
 - $MS = MD5(PMS \mid SHA('A' \mid PMS \mid R1 \mid R2)) \mid$
 - $MD5(PMS \mid SHA('BB' \mid PMS \mid R1 \mid R2)) \mid$
 - $MD5(PMS \mid SHA('CCC' \mid PMS \mid R1 \mid R2)) \mid \dots$
- Session keys
 - Computed from a master secret and the random values exchanged in the protocol (**not updated example**)
 - ClientMacKey = $MD5(MS \mid SHA('A' \mid MS \mid R1 \mid R2))$
 - ServerMacKey = $MD5(MS \mid SHA('BB' \mid MS \mid R1 \mid R2))$
 - ClientKey = $MD5(MS \mid SHA('CCC' \mid MS \mid R1 \mid R2))$
 - ServerKey = $MD5(MS \mid SHA('DDDD' \mid MS \mid R1 \mid R2))$

TLS performance

- First implementations were slow
- Currently, when properly configured, can be fast
- “On our production frontend machines,
TLS accounts for
less than 1% of the CPU load,
less than 10 KB of memory per connection and
less than 2% of network overhead.”
 - Adam Langley, Google "Overclocking SSL"
 - <https://istlsfastyet.com/>

TLS interception attack

Client is forced (or misled) to install a CA certificate controlled by the proxy



TLS tool: Fingerprints

Fingerprints



Is your employer, school, or Internet provider

eavesdropping on your secure connections?

370 sets of fingerprints checked per day

2,370,158 sets of fingerprints checked for our visitors

Secure browser connections can be intercepted and decrypted by authorities who spoof the authentic site's certificate. But **the authentic site's fingerprint CANNOT be duplicated!**

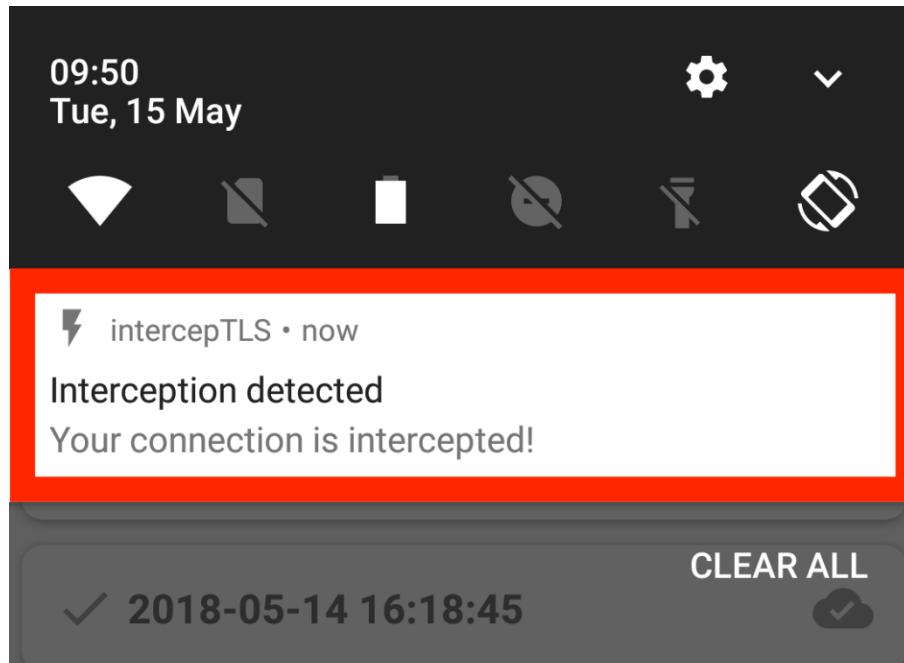
Domain Name	Certificate Name	EV	Security Certificate's Authentic Fingerprint
www.grc.com	grc.com	—	7A:85:1C:F0:F6:9F:D0:CC:EA:EA:9A:88:01:96:BF:79:8C:E1:A8:33
www.facebook.com	*.facebook.com	—	43:29:AB:C9:18:BB:BB:D5:B1:FC:8E:97:26:59:14:EB:92:B1:12:0D
www.paypal.com	www.paypal.com	●	BD:DE:B3:95:6B:86:79:B8:27:86:DA:4D:75:3C:53:AA:04:1E:08:92
twitter.com	twitter.com	—	73:33:BB:96:1D:DB:9C:0C:4F:E5:1C:FF:68:26:CF:5E:3F:50:AB:96
www.blogger.com	*.blogger.com	—	4D:8C:20:14:4A:3D:BB:CC:1E:62:36:9B:19:1A:3C:CF:E0:B4:CB:F1
www.linkedin.com	www.linkedin.com	—	2C:5C:AD:EB:6F:F3:9F:15:48:61:84:43:29:9C:B5:74:5A:BA:6E:A4
www.yahoo.com	*.www.yahoo.com	—	AD:D7:73:36:32:68:AB:33:71:3E:6E:1D:1B:73:93:1A:F4:2B:DC:D7
wordpress.com	*.wordpress.com	—	7A:C1:B2:7E:09:FF:88:03:C3:E9:B7:4F:31:F4:AC:75:79:BA:66:E6
www.wordpress.com	*.wordpress.com	—	7A:C1:B2:7E:09:FF:88:03:C3:E9:B7:4F:31:F4:AC:75:79:BA:66:E6

Each site's authentic security certificate fingerprint (shown above) was just now obtained by GRC's servers from each target web server. If your web browser sees a different fingerprint for the same certificate (carefully verify the Certificate Name is identical) that forms strong evidence that something is intercepting your web browser's secure connections and is creating fraudulent site certificates.

<https://www.grc.com/fingerprints.htm>

TLS tool: interceptTLS

- Android app to detect TLS interception
 - <https://tinyurl.com/interceptls>
- Research project (TU Munich)
 - Goal: detect networks that intercept TLS connections



TLS 1.3

- Stronger protection against downgrade attack
- Cryptographic improvements:
 - Full deprecation of MD5
 - New functions:
 - ChaCha20, Poly1305, Ed25519, x25519, x448
- Faster and more private handshake
- Session continuation improvements
- Perfect Forward Secrecy options

Summary

- HTTPS: Secure HTTP using SSL/TLS for encrypted communication.
 - Evolved from SSL in 1994 to TLS 1.3 by 2018
- TLS provides secure channels over TCP/IP, authentication, integrity, confidentiality, and key distribution
- Used beyond HTTP: e.g. SMTP, IMAP, POP3 protocols
- Many implementations: SSLref, OpenSSL, GnuTLS, SSLeay, Mbed TLS, Java Secure Socket Extension (JSSE)
- TLS is very significant, as it ensures data security across many critical Internet services

Trust and assurance

Segurança Informática em Redes e Sistemas
2024/25

Ricardo Chaves, David Matos

Ack: Miguel Pardal,
Miguel P. Correia, Carlos Ribeiro

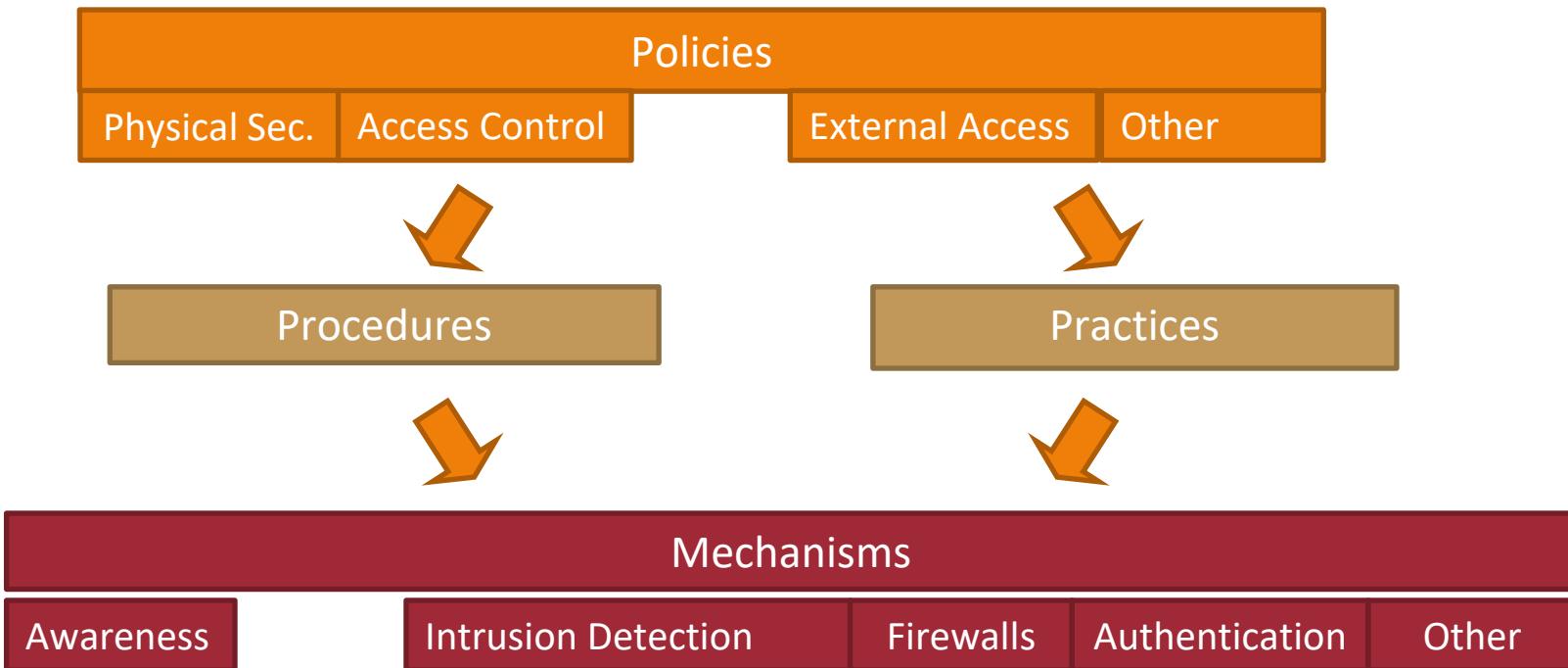
Roadmap

- Security Architecture
- Development cycle
- Recommendations
- Certification of applications and systems

Roadmap

- **Security Architecture**
- Development cycle
- Recommendations
- Certification of applications and systems

Security Architecture



Security Specifications

- Policies
 - Define acceptable behavior
- Procedures
 - Plans on how to do/enforce policies
- Practices
 - Rules to facilitate communication
- Analogy with legal world:
 - policies are the constitution,
 - procedures are the laws, and
 - practices are the regulations

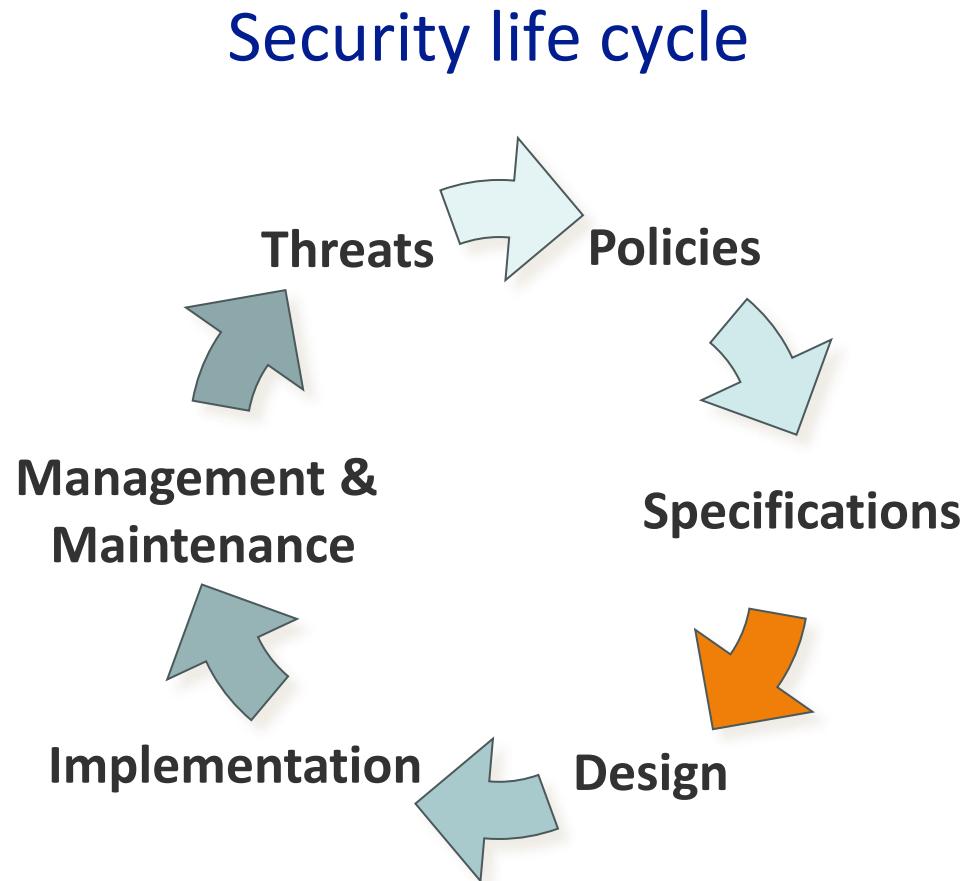
Mechanisms

- **Mechanisms** implement the specifications made in **policies**, **procedures**, **practices**
- Example generic security **mechanisms**:
 - Confinement (e.g., sandboxing)
 - Privileged execution
 - Authentication
 - Access Control
 - Filtering
 - Auditing
 - Cryptographic algorithms and protocols

Roadmap

- Security Architecture
- **Development cycle**
- Recommendations
- Certification of applications and systems

Development of secure applications

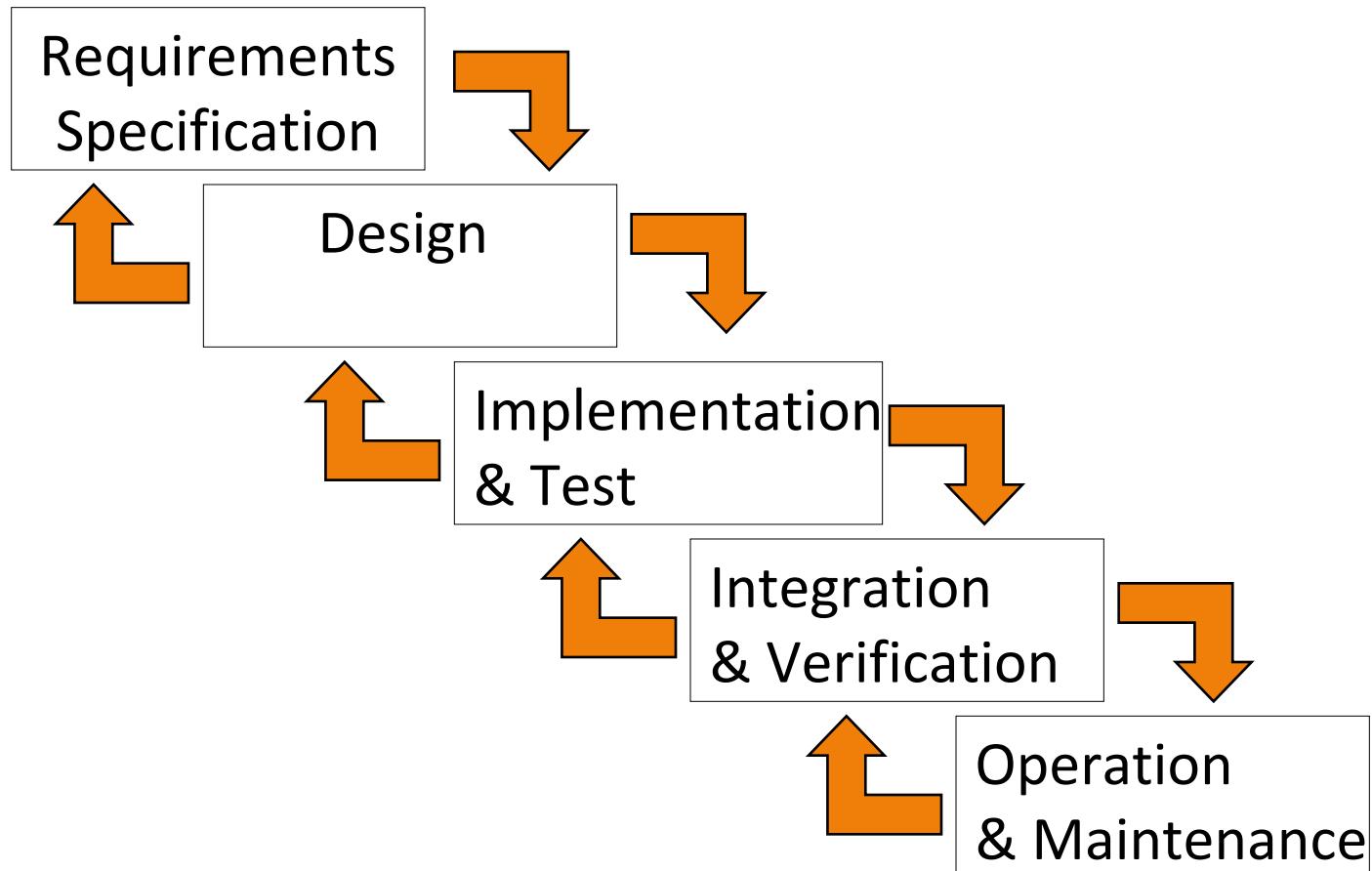


Development of secure applications

- Secure applications do not exist!
- Trust in the application:
 - A system is said to be trustworthy if there is enough evidence that it satisfies the security requirements
- Trust is obtained through assurance techniques:
 - Development methodologies
 - Formal methods
- Certification is the acceptance by assurance experts and the assignment of an assurance level

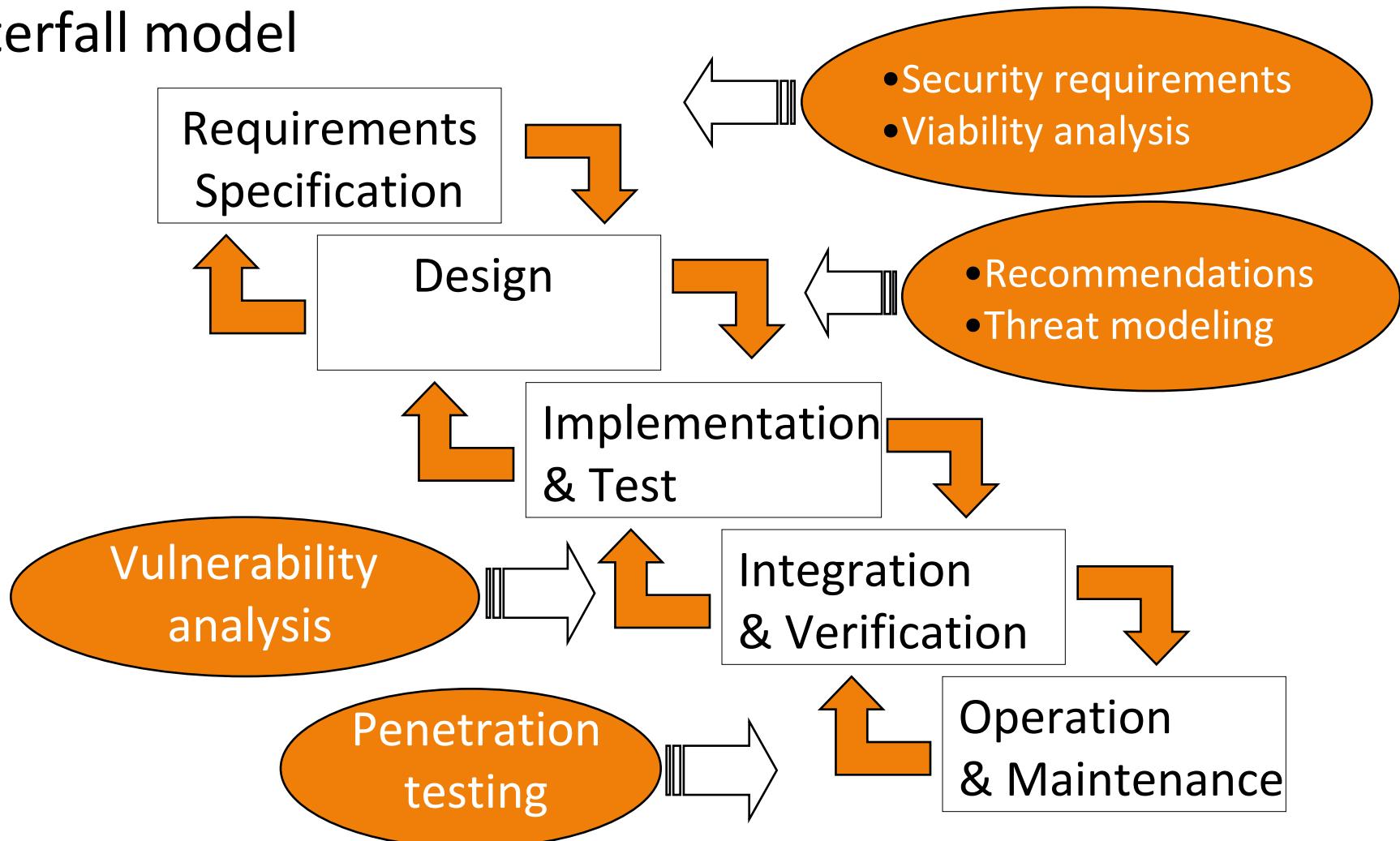
Development cycle

Waterfall model

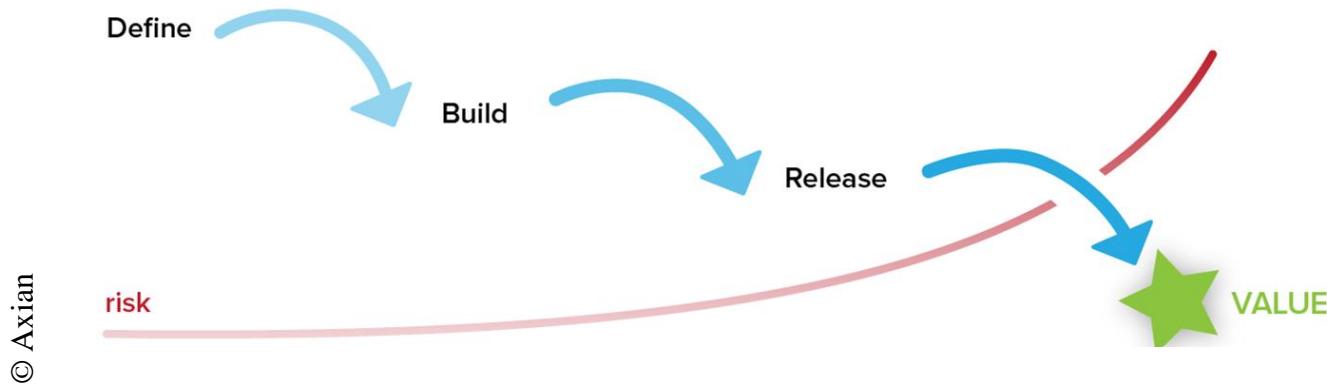


Security in the development cycle

Waterfall model

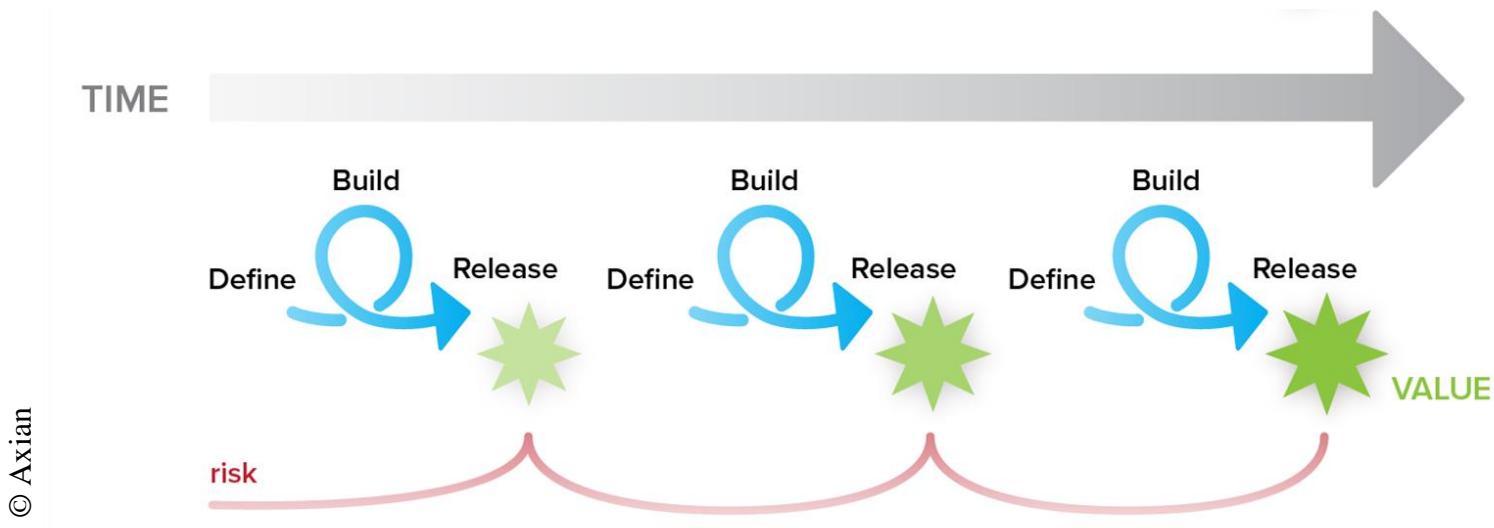


Classic development model: the “waterfall”



© Axian

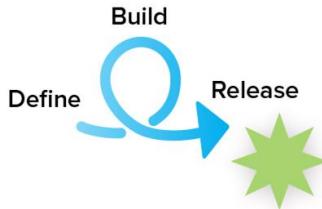
Agile development: “sprints” that add value and lower risk



© Axian

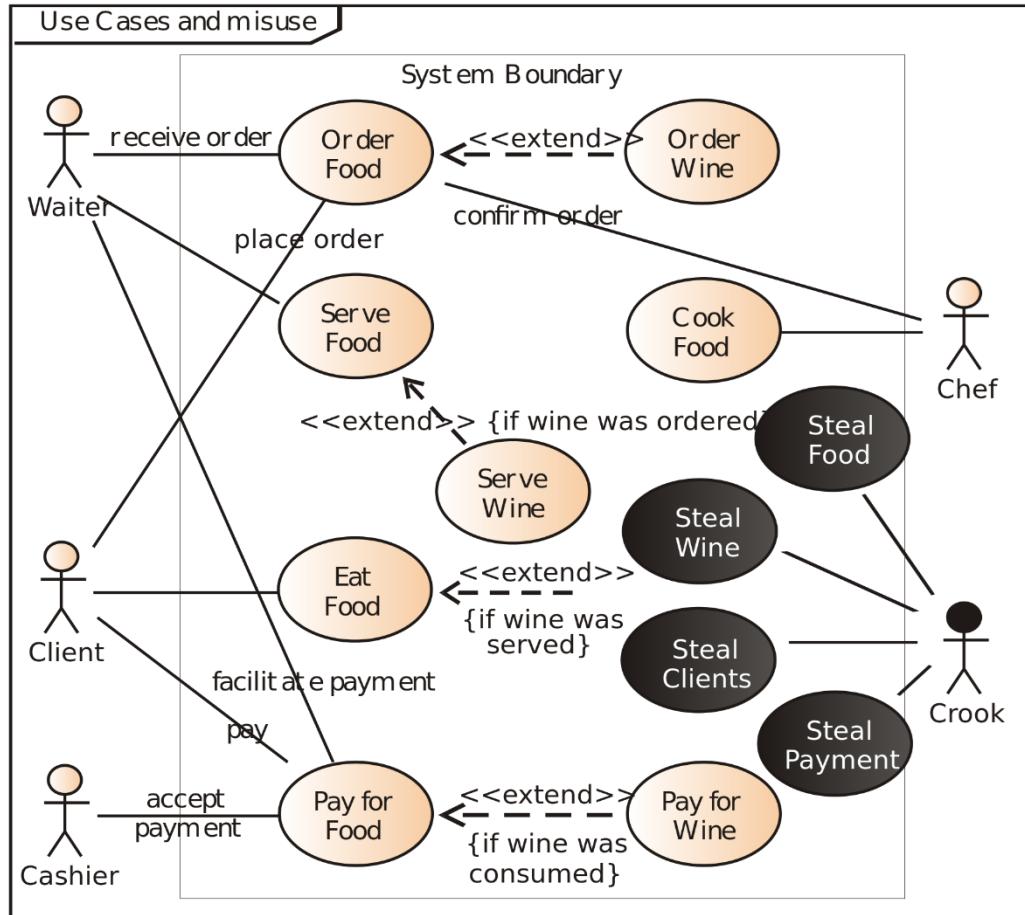
Agile Security Practices

- **Inception** practices: at the start of the Agile project
 - Risk Assessment
 - Requirements Definition
 - Incident Response
- **Iteration** practices: should be performed in every release
 - Threat Assessment
 - Code Review
 - Design Review
- **Regular** practices: on multiple sprints during the project
 - Dynamic Security Testing
 - Fuzz Testing (misuse)



Misuse case

(extension of UML use cases with explicit attackers)



Attacker model

- Attacker model is a formalization of the attacker capabilities
 - Allows us to justify the existence of defense mechanisms
 - What is each mechanism protecting against?

Attacker model example

- We consider the following capabilities to model different types of attackers:
 - A1 - Record any number of IP packets between a given source and destination and replay them from own IP address
 - A2 - Modify and suppress any number of IP packets from a given source and destination
 - A3 - Send IP packets from any IP address and receive packets

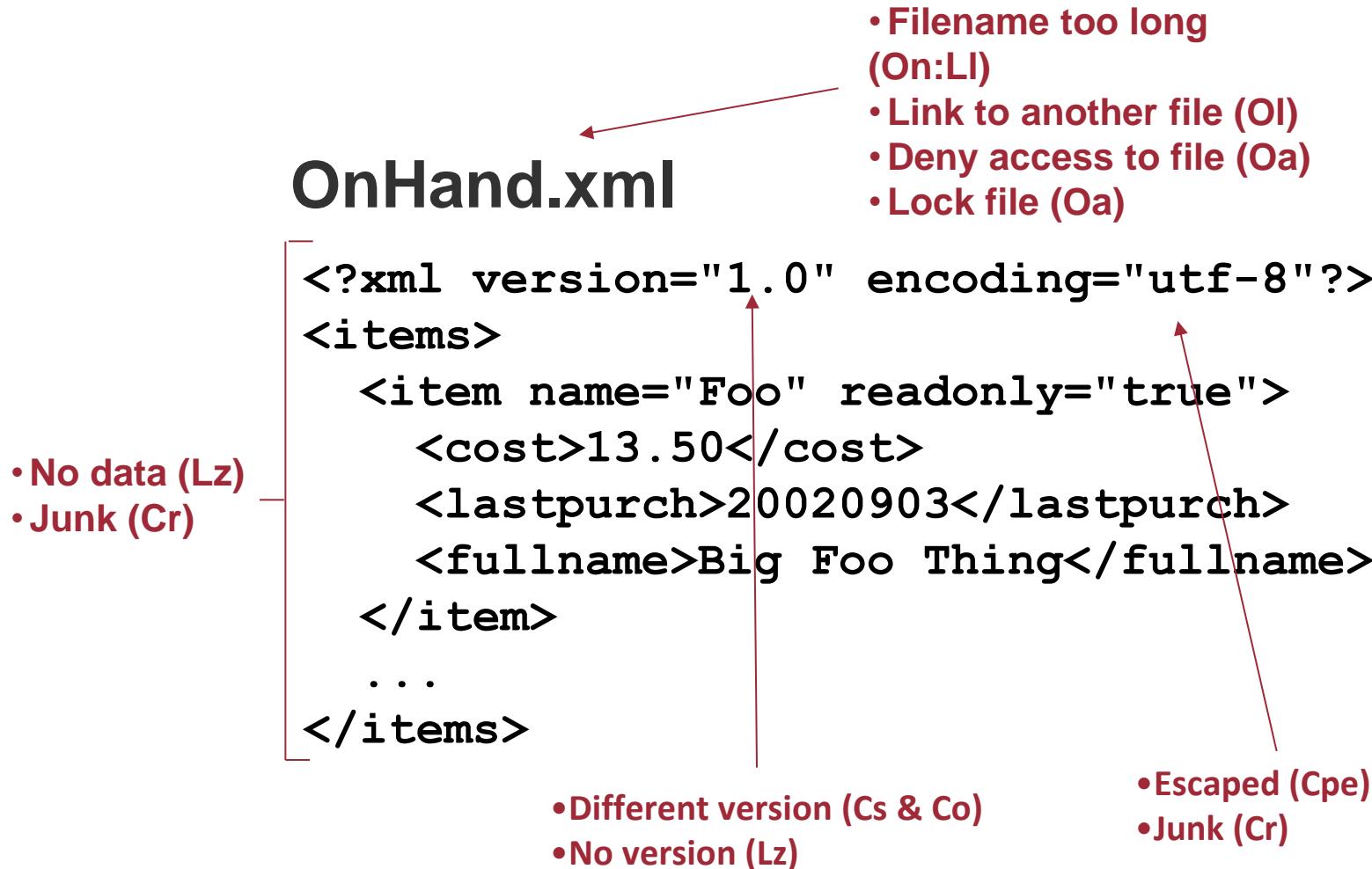
Fuzzing / security testing

- Identification of inputs
- Controlled input mutation
- Input injection
- Result analysis

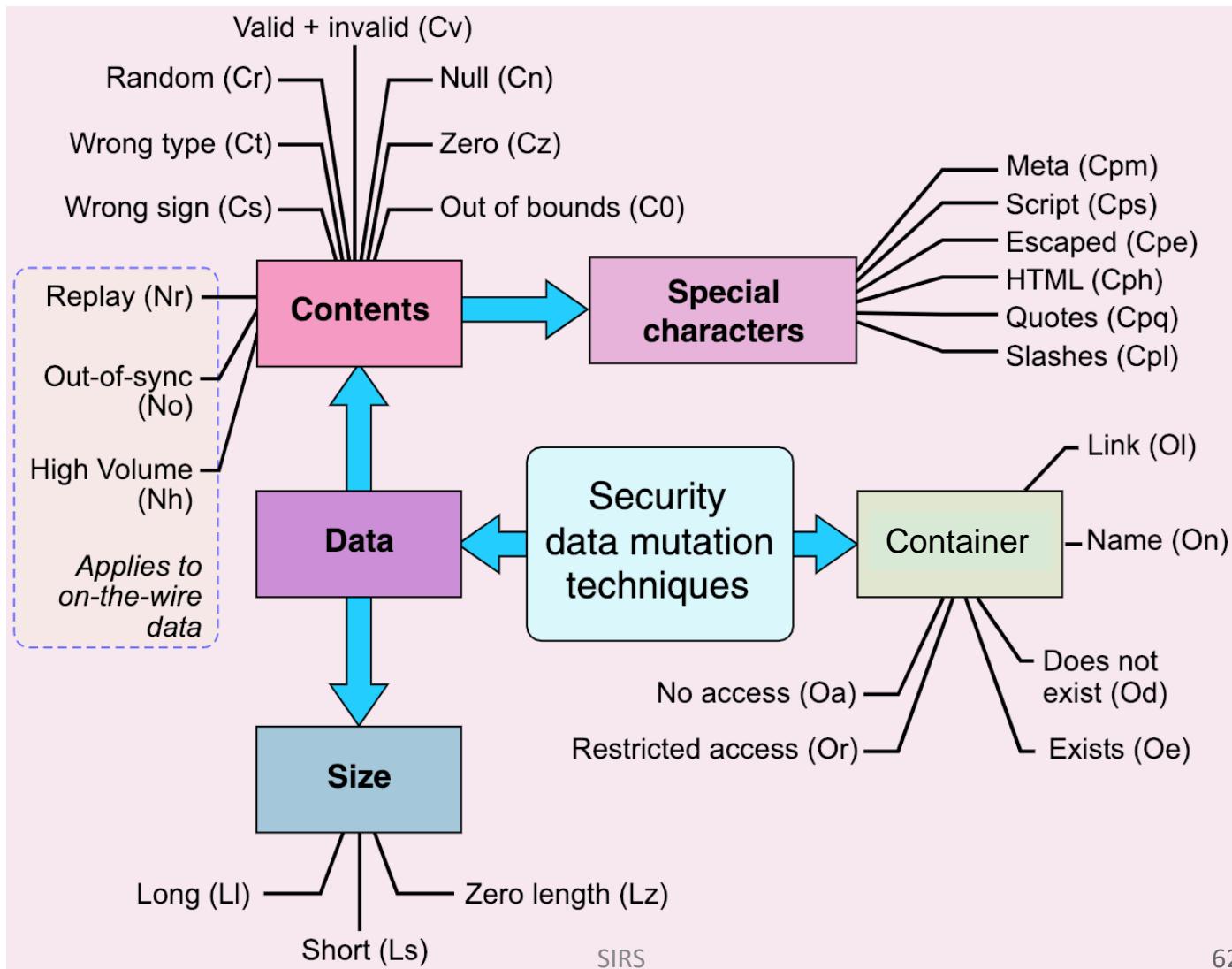
Identification of inputs

- Application decomposition
- Identification of the interfaces
- Enumeration of data inputs:
 - Sockets
 - Pipes
 - Registry
 - Files
 - RPC (etc.)
 - Input parameters
 - Etc.
- Enumeration of the data structures
 - C/C++ Data structures
 - HTTP headers
 - HTTP body
 - Search strings
 - Flags
 - Etc.
- Establish the valid constructs

Fuzzing example



Controlled input mutation



Incentives for ethical hackers: Bug bounty programs

The screenshot shows the homepage of the Bugcrowd Public Bug Bounty List. At the top, there's a navigation bar with links for Researcher Portal, Customer Portal, Products, Solutions, Customers, Researchers, Programs, Resources, About, and a prominent orange "Hack With Us" button. The main title "PUBLIC BUG BOUNTY LIST" is centered in large, bold, black capital letters. Below it, a descriptive subtitle reads: "The most comprehensive, up to date crowdsourced list of bug bounty and security disclosure programs from across the web curated by the hacker community." A note below states: "This list is maintained as part of the [Disclose.io](#) Safe Harbor project." Another note encourages suggestions: "Have a suggestion for an addition, removal, or change? Open a Pull Request to [disclose](#) on Github. Special thanks to all [contributors](#)." At the bottom, there's a search bar with placeholder text "Type here", a blue "Filters" button with a dropdown arrow, and a footer row with links for Program Name, New, Bug Bounty, Swag, Hall of Fame, Submission URL, and Safeharbor.

bugcrowd
#1 Crowdsourced Security Company

Researcher Portal | Customer Portal

Products Solutions Customers Researchers Programs Resources About Hack With Us

PUBLIC BUG BOUNTY LIST

The most comprehensive, up to date crowdsourced list of bug bounty and security disclosure programs from across the web curated by the hacker community.

This list is maintained as part of the [Disclose.io](#) Safe Harbor project.

Have a suggestion for an addition, removal, or change? Open a Pull Request to [disclose](#) on Github. Special thanks to all [contributors](#).

Type here

Filters ▾

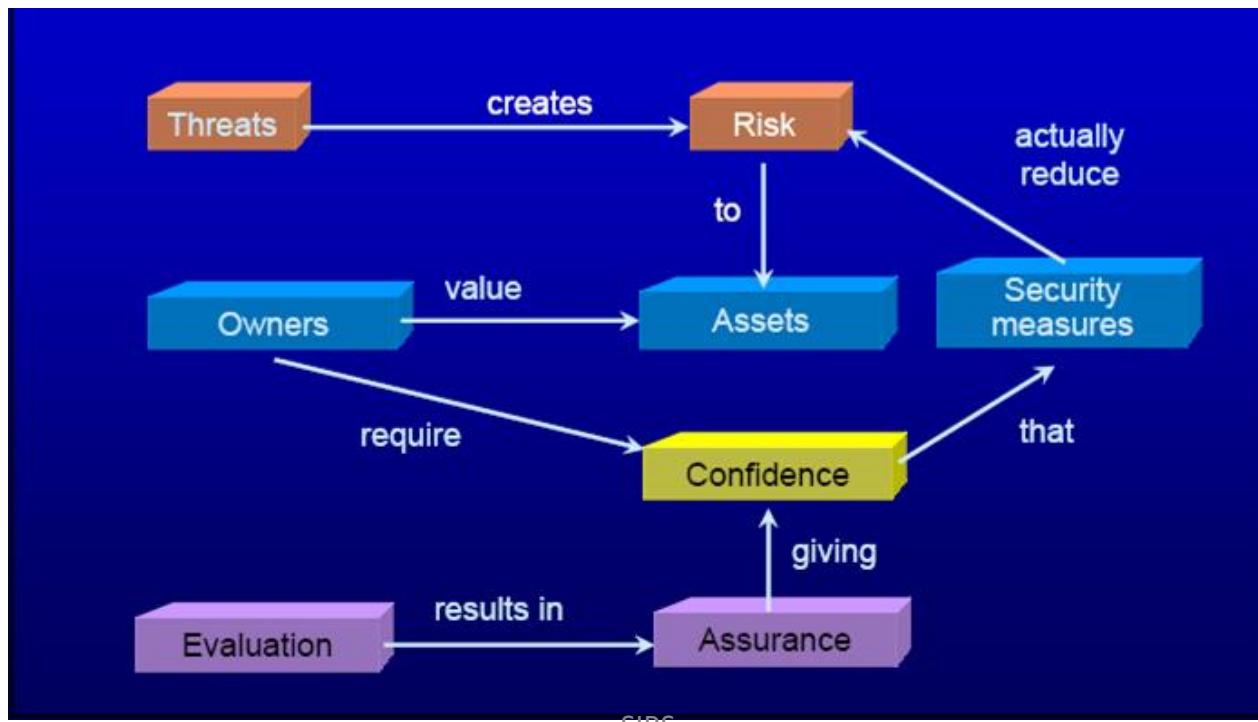
Program Name New Bug Bounty Swag Hall of Fame Submission URL Safeharbor

Roadmap

- Security Architecture
- Development cycle
- Recommendations
- **Certification of applications and systems**

Trust and Certification

- Assurance: ways of convincing others that a model, design or implementation is correct (trustworthy)

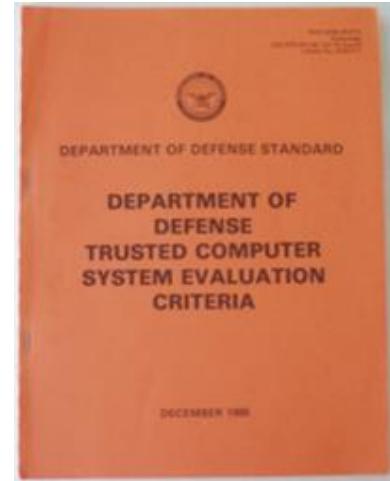


Certification

- For operating systems
 - TCSEC (Orange Book) (US)
 - ITSEC (UK, France, Germany, Nederland's)
- For computer networks
 - TNI - Trusted Network Interpretation (US)
- For cryptography
 - **FIPS 140** (US) ← Most relevant today
- For applications
 - ITSEC
 - **Common Criteria** (international) ↗

TCSEC (US standard)

- Evaluation:
 - Design analysis
 - Test analysis
 - Final review
- Evaluation done by independent evaluators
- Assigns a class: C1, C2, B1, B2, B3, A1
- Problems:
 - Focused heavily on confidentiality, disregarding other security properties
 - Narrow scope: mainly military operating systems
 - Based on the documentation, no access to the source code
 - Long evaluation time
 - Evaluation mixes assurance with functionality



ITSEC

- European standard
- Functionality and assurance are evaluated separately
- Applicable to systems and applications
 - TOE – *Target of Evaluation*
- Problems:
 - No validation that security requirements make sense
 - Inconsistency in evaluations (not as well defined as in TCSEC)

TCSEC	ITSEC
C1	F1+E2
C2	F2+E2
B1	F3+E3
B2	F4+E4
B3	F5+E5
A1	F5+E6

better

Common Criteria

- International standard ISO/IEC 15408
 - Parts:
 - CC documentation
 - Evaluation methodology of CC (CEM)
 - National schemas (Country specific): evaluators selection; certification attributions; interaction between evaluators and vendors, etc.
 - e.g. in the USA NIST accredits commercial organizations
 - CC Methodology (CEM)
 - Functional requirements
 - Assurance requirements
 - Evaluation Assurance Levels (EALs)
 - e.g. Java Smart Card has an EAL=5+ (maximum is 7)
 - Types of evaluation
 - Protection Profile (PP)
 - Security Target (ST)
- } next slide

CC – PPs and STs

- Protection Profiles – product independent
 - for categories of products:
 - Operating systems
 - Firewalls (packet filter, application-level gateway)
 - SmartCards
- Security Targets – specific for each product
 - Hitachi Universal Storage Platform V – EAL2
 - Cisco PIX Firewall – EAL4+
 - GemXplore Xpresso V3 Java Card Platform – EAL5+
- List of PPs <http://www.commoncriteriaportal.org/pps/>
- List of STs (certified products) <http://www.commoncriteriaportal.org/products/>

CC – PP and ST

Protection Profile (generic)

- Introduction
- **Description of** the class/family of target products, a.k.a. Target of Evaluation (**TOE**)
- Description of the execution environment
 - Assumptions regarding the system operation
 - Threatened resources
 - Security policy of the target organization
- Security objectives
 - Product/system objectives
 - Environment objectives
- Security requirements
 - Functional
 - Assurance
- Rational
 - Interconnects the previous points

Security Target (specific)

- Introduction
- TOE description
- Description of the execution environment
 - Assumptions regarding the system operation
 - Threatened resources
 - Security policy of the target organization
- Security objectives
 - Product/system objectives
 - Environment objectives
- Security requirements
 - Functional
 - Assurance
- TOE specification
 - Security mechanisms
 - Description on how to assure security
- **PP claims**
 - How the PP objectives /requirements are fulfilled
- Rational

CC security requirements

Functional Requirements

- Product/system behavior definition regarding security
- 11 classes divided in families that contain components
- Components have:
 - Requirements definition
 - Dependencies from other requirements
 - Requirements hierarchy
- Predefined classes:
 - Audit (FAU)
 - Cryptography Support (FCS)
 - Communications (FCO)
 - User Data Protection (FDP)
 - Identification and Authentication (FIA)
 - Security Management (FMT)
 - Privacy (FPR)
 - Protection of the TOE Security Functions (FPT)
 - Resource Utilization (FRU)
 - TOE Access (FTA)
 - Trusted Path/Channels (FTP)

Assurance Requirements

- Establish confidence in the security features
- Correction of the implementation
- Fulfillment of the security objectives
- 10 classes
 - 1 – Evaluation of PPs
 - 1 – Evaluation of STs
 - 1 – Maintenance of Assurance
 - 7 – Product assurances
- Assurance classes:
 - Development
 - TOE design, Functional specifications, ...
 - Delivery and Operation
 - Configuration
 - Product Documentation
 - Life cycle
 - Delivery, Flaw remediation, ...
 - Testing
 - Depth, coverage, ...
 - Vulnerability analysis

Evaluation Assurance Levels

- Derived from the assurance requisites

EAL1	Functionally Tested
EAL2	Structurally Tested
EAL3	Methodically Tested & Checked
EAL4	Methodically Designed, Tested & Reviewed
EAL5	Semiformally Designed & Tested
EAL6	Semiformally Verified Design & Tested
EAL7	Formally Verified Design & Tested

better
↓

Roadmap

- Security Architecture
- Development cycle
- **Recommendations**
- Certification of applications and systems

IEEE Center for Secure Design recommendations



Interested in keeping up with Center for Secure Design activities? Follow @ieeecsdesign on Twitter, catch up with us via cybersecurity.ieee.org, or contact Kathy Clark-Fisher, Manager, New Initiative Development (kclark-fisher@computer.org).

<http://cybersecurity.ieee.org/blog/2015/11/13/avoiding-the-top-10-security-flaws/>

Secure design

- Prevent security problems in **design** stage
 - Design flaws
 - Different from implementation bugs or defects
 - Avoiding flaws can significantly reduce the number and impact of security breaches
 - The goal of a secure design is to enable a system that **supports and enforces** the necessary authentication, authorization, confidentiality, data integrity, accountability, availability, and non-repudiation requirements, **even when the system is under attack**

Top 10 recommendations

1. Earn or give, but never assume trust
2. User authentication that cannot be bypassed or tampered
3. Authorize after you authenticate
4. Strictly separate data and control instructions
5. All data must be explicitly validated
6. Use cryptography correctly
7. Identify sensitive data and how to handle it
8. Always consider the users
9. Understand how external components affect attack surface
10. Be flexible when considering future objects and actors

1/10 Earn or give, but never assume **trust**

- Software systems rely on composition and cooperation of two or more software tiers or components
- Offloading security functions from server to client exposes those functions to a much less trustworthy environment
- When untrusted clients send data to your system or perform a computation on its behalf, the data sent must be assumed to be compromised until proven otherwise

2/10 Use authentication mechanism that cannot be bypassed or tampered with

- Authentication is the act of validating an entity's identity
- A securely designed system should also prevent that user from changing identity without re-authentication
- Authentication techniques should require one or more factors for more sensitive operations
 - Factors:
 - something you know,
 - something you are, or
 - something you have



3/10 Authorize after you authenticate

- Authorization should be conducted as an explicit check
 - Necessary even after an initial authentication has been completed
- Authorization depends not only on the **privileges** associated with an authenticated user, but also on the **context** of the request
 - Time, location, etc.
 - Handle revocation

4/10 Strictly separate data and control instructions

- Combining data and control instructions in a single entity, especially a **string**, can lead to injection vulnerabilities
 - Often leads to untrusted data controlling the execution flow of a software system
 - Concern at all levels: machine instructions, high-level instructions, domain specific languages

5/10 All **data** must be **explicitly validated**

- It is important to explicitly ensure that assumptions on data hold
 - Vulnerabilities frequently arise from implicit assumptions about data
- Design software systems to ensure that comprehensive data validation actually takes place and that all assumptions about data have been validated when they are used

6/10 Use cryptography correctly

- Through the proper use of cryptography, one can ensure the confidentiality of data, protect data from unauthorized modification, and authenticate the source of data
 - and more
- Common cryptography pitfalls:
 - Creating your own cryptographic algorithms or implementations
 - Misuse of libraries and algorithms
 - Poor key management
 - Randomness that is not random
 - Failure to allow for algorithm adaptation and evolution

7/10 Identify **sensitive** data and how it should be handled

- Data sensitivity is context-sensitive
 - Depends on regulation, company policy, contractual obligations, user expectation, etc.
 - Examples: user-input, data computed from scratch, data coming from external sensors, cryptographic material, and Personally Identifiable Information (PII)
- First step: create a policy that explicitly identifies different levels of classification
- Define most important property:
 - Confidentiality
 - Integrity
 - Availability

8/10 Always consider the **users**

- The security of a software system is inextricably linked to what its users do with it
- Always consider the users, and any other stakeholders, in the design and evaluation of systems
 - Factors
 - Trade-offs
- Make the most common usage scenario also secure
 - “secure by default”
 - Make relevant settings as easy to find

9/10 Understand how external components affect attack surface

- The attack surface are the different points where you can try to enter or extract data from the system
- You must assume that incoming external components are not to be trusted until appropriate security controls have been applied
- Align the component's attack surface and security policy with the overall system's

10/10 Be flexible when considering future changes to Objects and Actors

- Software security must be designed for change
 - Environments, threats and attacks
 - Rather than being fragile, brittle, and static
- Consider the security implications of future changes
 - Design for security updates
 - Design for security properties changing over time
 - Design for changes in components beyond your control
 - Design with the ability to isolate or toggle functionality
 - Design for changes to objects intended to be kept secret (keys)
 - Design for changes in entitlements (dynamic permissions)

Summary

- Security Architecture
- Development cycle
- Recommendations
- Certification of applications and systems