UNIVERSITÀ DEGLI STUDI DI PADOVA

Master's degree ICT Internet Multimedia Engineering

Department of Information Engineering (DEI)
Master degree on ICT for Internet and Multimedia Engineering (MIME)

Internet of Things and Smart Cities
13 – Security

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

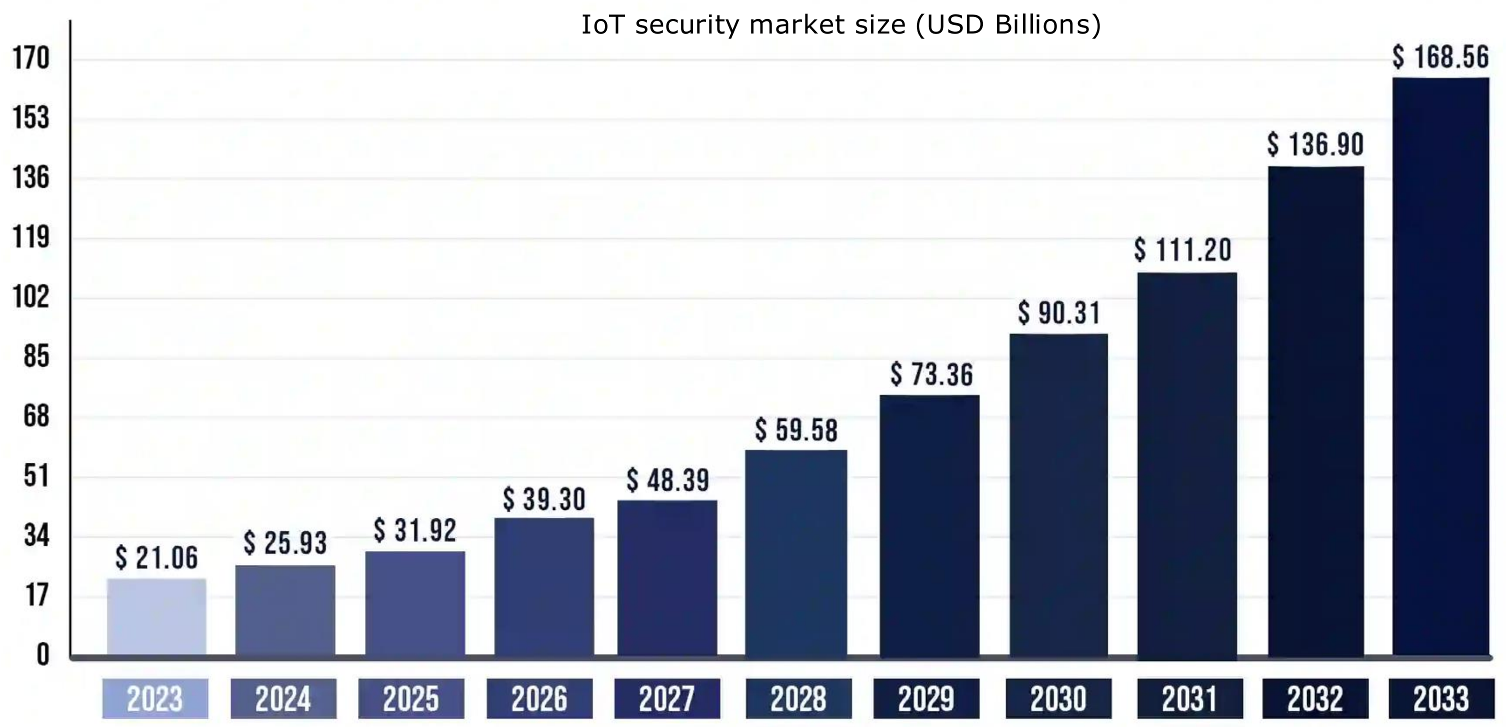
Special thanks to: Dr. Sara Baldoni

Overview

Some numbers…

- The investments in IoT security are following a rapid increase.

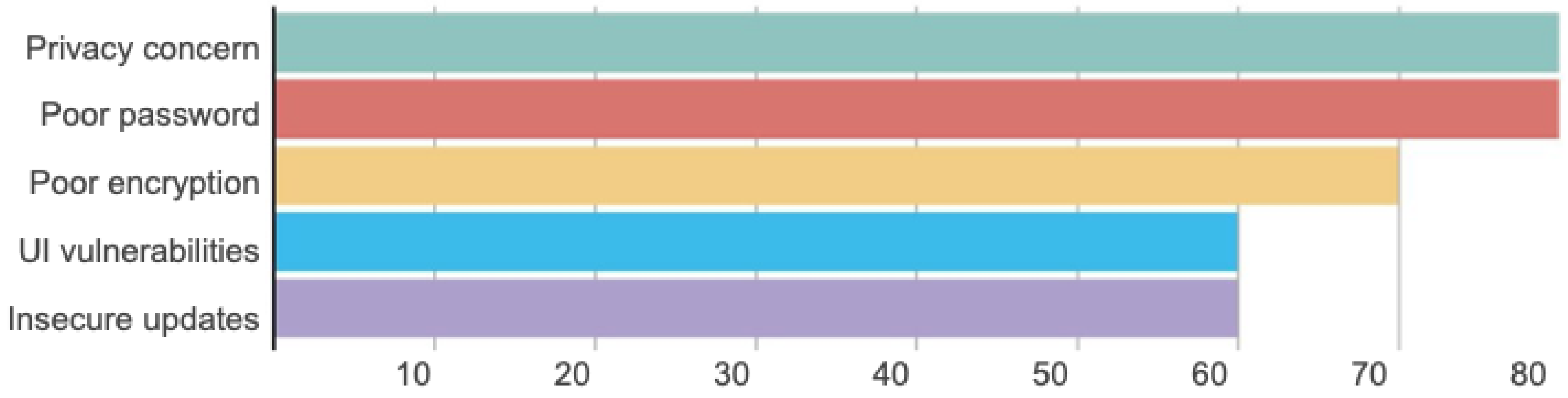
https://www.precedenceresearch.com/iot-security-market



Overview

Some numbers…

- More than 70% of IoT devices are highly vulnerable to attacks.
  - Additional efforts are required to guarantee that IoT systems are **safe** and **secure**.



Thanks to: Sara Baldoni (sara.baldoni@unipd.it)

Overview

Introduction

- IoT systems come with **inherent security concerns**:
  - IoT systems are **distributed** systems with many geographically dispersed heterogeneous nodes with different capabilities, connected via separate physical networks.
  - IoT systems can interact directly with the physical world and can thus **impact the health and safety** of people.
  - IoT systems are **constrained** devices (energy, capacity, etc.).
  - IoT systems are deployed in **unprotected** and unattended environments.
  - IoT systems often deal with confidential data, so **privacy** becomes a primary concern.

Overview

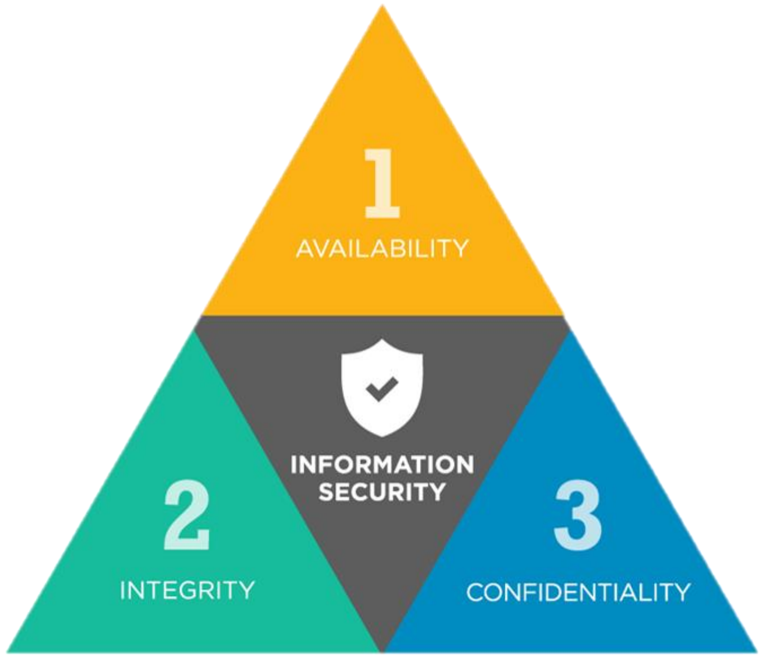
Introduction

- Still, there are some **good features**:
  - IoT Edge and many other components are **fixed** or narrow-function devices, not general-purpose computing systems.
  - IoT endpoints can be directed to communicate only with a **limited population** of known authenticated entities that may be identified and with whom trust can be established.
  - Additional **precautionary measures** may be taken to reduce exposure:
    - Not allowing the downloading of unsolicited or unauthenticated software.
    - Closing of ports not used for IoT communication.
    - Elimination of OS features that allow remote login, shell access, or support unsafe protocols.

Security threats

Introduction

- IoT security can be defined as "protecting it from unauthorized access or changes."
- Security objectives can be represented as a pyramid (**CIA**):
  - **Confidentiality**: ensure that information (data) is available only to authorized parties.
  - **Integrity**: ensure that information (data) is accurate and unchanged (untampered).
  - **Availability**: ensure that information is available for access whenever an authorized user requires it.



Security attacks

Physical attacks

- **Physical attacks** consist in manually attacking the target system:
  - Node substitution/cloning.
  - Physical damage.
  - Creation of faulty measurement (e.g., using a lighter for a thermal sensor).
  - Hinder a measurement (e.g., using a piece of paper in front of a security camera).

Security attacks

Software attacks

- **Software attacks** can cause different altered behaviors (e.g., modify the way an actuator interacts with the physical world):
  - Viruses and worms.
  - Trojans.
  - Code injection: commanding the actuator to perform unsafe actions.
  - Phishing emails.
  - Configuration of nodes with the default common user and password combinations.

Security attacks

Network attacks

- **Network attacks** occur on the communication channel:
  - Sniffing or eavesdropping (on wired or wireless channels).
  - Spoofing.
  - Man-in-the-middle.
  - Denial of Service (e.g., flooding a wired channel, jamming a wireless channel).
  - Sinkhole attack (routing): a node declares itself as having exceptional resources and power, thus causing its neighbors to choose it as a routing waypoint.
  - Wormhole attack (routing): two malevolent nodes claim that there is only a single hop between them and thus divert a lot of routed traffic to themselves.
- Wireless networks can provide additional exposure because they can be eavesdropped without requiring physical taps on the communication medium.

Security principles

Requirements

- The requirements for IoT systems are:
  - **Safety**: components need to be certified, and continuous monitoring is needed.
  - **Reliability**: the system needs to operate continuously (redundancy can help).
  - **Resilience**: the system has to absorb and limit the effects of small issues and resume operation rapidly after major accidents.

Security principles

Guidelines

- Saltzer and Schroeder outlined general design principles for protection:
  - **Least** privilege: Every user should use the least privileges necessary for a task.
  - **Separation** of privilege: satisfy more than one condition (e.g., 2 keys for a vault).
  - **Least common** mechanism: minimize the number of mechanisms shared among users.
  - **Economy** of mechanism: keep the design simple.
  - **Complete** mediation: every access should be checked for authorization.
  - **Fail-safe** default: the default condition should be lack of access.
  - **Open** design: the design should not be secret (do not rely on the ignorance of attackers).
  - **User acceptability**: the mechanism needs to be user-friendly.
  - **Work factor**: stronger security measures should make attackers work harder.
  - **Compromise recording**: system keeps attack records to discover unauthorized use.

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Master's degree ICT Internet Multimedia Engineering

13 – Security

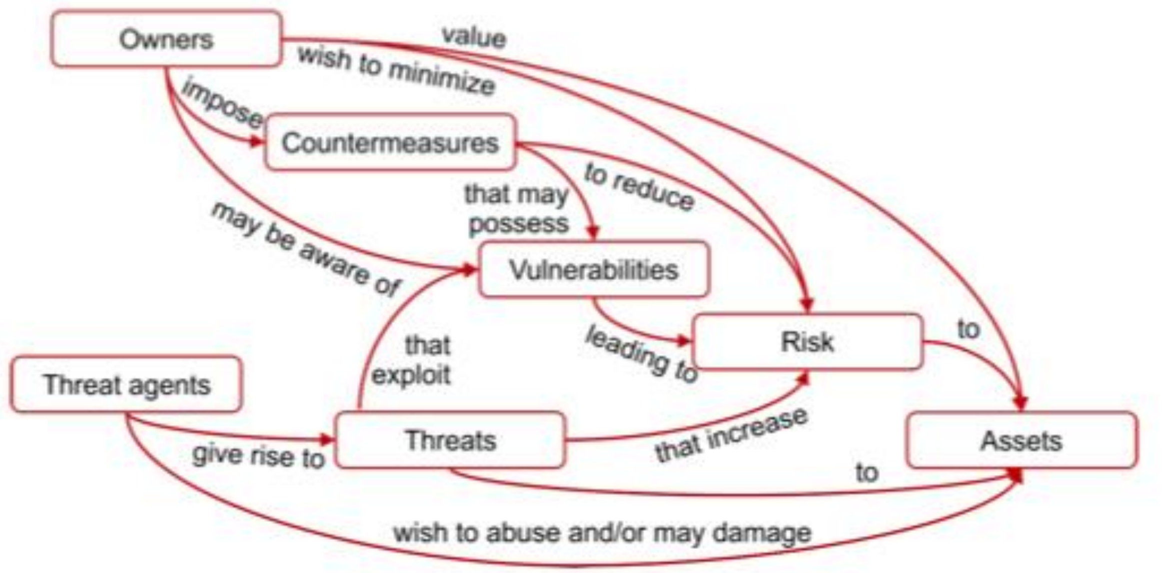
Security planning and analysis

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Risk analysis

Introduction

- Security should be considered in an IoT system **from the initial design** and it is a continuous process.



Thanks to: Sara Baldoni (sara.baldoni@unipd.it)

Risk analysis

Definitions

- **Risk** can be defined as *"the possibility that a threat agent will exploit a vulnerability to damage an asset."*
- A **vulnerability** can be defined as *"a software/hardware bug or misconfiguration that a malicious individual can exploit."*
  - Simply put, it is a weakness in a system (or countermeasures for protecting it).
  - The vulnerability meaning is related to the threat concept.
- A **threat** can be defined as *"any activity or event that can cause an unwanted outcome (e.g., damage, disruption or loss of an asset)."*
  - The relevance of a threat depends on its impact.

Risk analysis

Definitions

- The amount of risk the vulnerability presents depends on:
  - The **number** of system affected by the vulnerability.
  - The **criticality** of the affected systems. It represents the measure of how valuable the asset is to the organization, if compromised.
  - The **exposure** those systems present to the organization.

- Therefore, the risk can be computed as:

$$\text{Risk} = \text{Vulnerability} \times \text{Attacks} \times \text{Threat} \times \text{Exposure}$$

Risk analysis

Definitions

- Risk cannot be **eliminated**, but a proper analysis of threats can help in risk mitigation.
- Risk analysis allows to define the **proper level of security** for satisfying the operational objectives at a cost that provides an acceptable return on investment.
- Risk **avoidance** seeks to eliminate the risk by eliminating exposure to the specific threats (e.g., eliminate nonessential features).
- **Risk acceptance** consists in deciding not to invest in preventive measures if the attack has a very low probability, or its impact can be managed.
- Risk **transference** consists in transferring a risk to a third party (e.g., insurance company) upon payment. Valid for unanticipated (low-probability) incidents.

Risk analysis

Example

- Risk analysis is important not only to protect the assets which are clearly critical, but also all the other assets connected to them.
- Consider a castle containing a treasure that is being attacked by an army.
  - **Exposure** represents how exposed the castle is to the attack (e.g., walls, moat?).
  - **Periphery** represents the extent of the walls/moat, and the number of openings.
  - **Threat** is a measure of the enemy armies who are performing the attack.
  - **Atack** is represented by the actual arrows and breach attempts on the walls.
  - **Vulnerability** is a measure of how easy is for the treasure to be accessed.
  - **Criticality** represents a measure of the value of the treasure.

Security threat modeling

Introduction

- Threat modelling: identify how an attacker can attempt to compromise the system.
- The basic steps in the threat modeling process are:
  - **Model the system**, i.e., create an architecture diagram (components and flows).
  - **Enumerate threats**.
  - **Mitigate threats**.
  - **Validate mitigations**.

Security threat modeling

Model the system

- Usually performed by **analyzing the system model**.
- It should identify all key **components** (sensors and actuators), their **connections**, and **data** and control flows between specific components and in the overall system.
- The model is evolved by determining the **requirements** that specify what the system needs to so (e.g., monitor and control the operation of a process).

Security threat modeling

Enumerate threats

- Known threat types are summarized in the **STRIDE** model.
- Attacks may combine several of these modes.

Threat	**Violated property**	**Description**
Spoofing	Authentication	Impersonating someone/something else
Tampering	Integrity	Altering hardware, data, code...
Repudiation	Non repudiation	Denial of being involved
Information disclosure	Confidentiality	Exposing the asset to unauthorized parties
Denial of Service	Availability	Denial of offered services
Elevation of privilege	Authorization	Gain capabilities without authorization

Security threat modeling

Enumerate threats

- The **DREAD** model has been developed to rating the threats.
  - **Damage potential**: the potential damage that an exploit can cause
  - **Reproducibility**: how easy it is to reproduce an exploit
  - **Exploitability**: what is required to make the exploit successful
  - **Affected users**: the number of users affected by the exploit
  - **Discoverability**: the ease with which the threat can be uncovered

Security threat modeling

Mitigate and validate threats

- Designing the appropriate **defence mechanisms**.
- **Validate** that the chosen mitigations address all identified vulnerabilities.
- When analyzing threats, the general advice is to **think like an attacker** and to examine ways in which they may attempt to penetrate the system.
  - Reconnaissance.
  - Scanning.
  - Access.
  - Maintain access.
  - Cover the tracks.

UNIVERSITÀ DEGLI STUDI DI PADOVA

Master's degree ICT Internet Multimedia Engineering

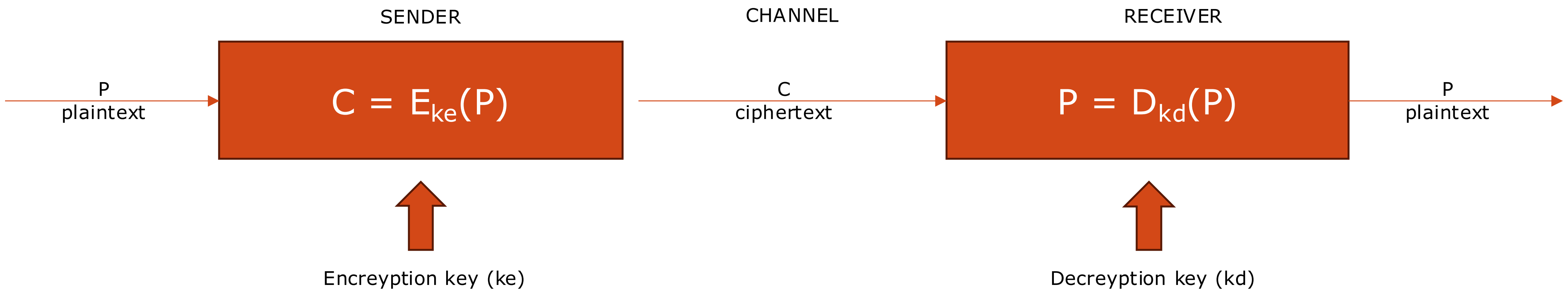
13 – Security
Cryptography

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Cryptography

Overview

- Cryptography is the **enciphering** and **deciphering** of messages in secret code.
- It is the principal solution for providing confidentiality, integrity and authentication.
- It allows to convert a message (**plaintext**) to a secret code (**ciphertext**).
- The transformation is performed using a **key**, so that without the correct key it is not possible to perform the inverse conversion.



Cryptography

Threats

- Attacks targeting cryptographic systems can be:
  - **Ciphertext only**: the attacker sniffs the traffic and has to analyze the encrypted data through cryptanalysis techniques.
  - **Known plaintext**: the attacker has access to the plaintext and the ciphertext.
  - **Chosen plaintext**: the attacker is able to get the source system to insert into the system a message chosen by him/her.
- Cryptographic systems are subject to **brute-force attacks** where the ciphertext can be computer-analyzed by rapidly trying many, and in some cases all, different combinations or keys.
  - The process can be aided by exploiting known weaknesses (e.g., language patterns).

Cryptography

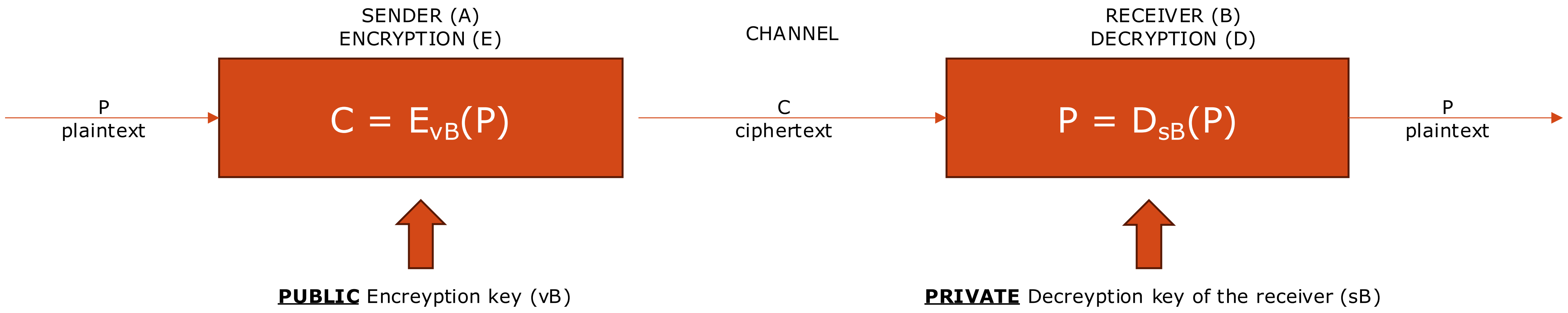
Types

- **Symmetric**: Use the same key for both encryption and decryption.
  - The (private) key should be **secret**.
  - An additional mechanism for secure key distribution is required.
  - This may be accomplished by using separate channels or key exchange mechanisms.
- **Asymmetric** (or **public**): Different keys are used for encryption and decryption.
  - Encryption and decryption algorithms are easy to compute.
  - It is computationally easy to generate a key pair.
  - It is computationally infeasible to derive a private key from the corresponding public one.
  - Given the ciphertext and the public key, it is infeasible to reconstruct the plaintext.

Asymmetric cryptography

Public-key cryptography

- Asymmetric system.
- **Use different keys for encryption and decryption** (publicly visible and private).
  - Public keys may be visible to all.
  - Private keys are kept secret by each node and stored in hardware-secured registers.
- Each entity uses a method to compute a matching pair of public and private keys.



Asymmetric cryptography

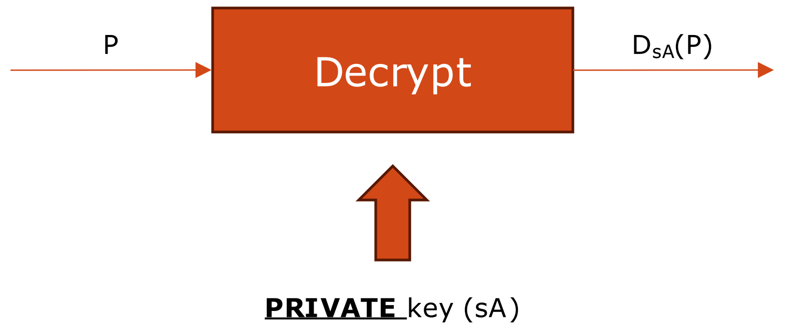
Public-key cryptography

- The ciphertext is sent to the recipient over insecure communication links.
- **Confidentiality:** ensured, since **only the recipient B knows its private key** with which the message can be decrypted.
  - An eavesdropper can obtain the ciphertext and access the public key, but finds it computationally infeasible to reconstruct the original message.
- What about **integrity** and **authentication**?

Asymmetric cryptography

Public-key cryptography: double transformation

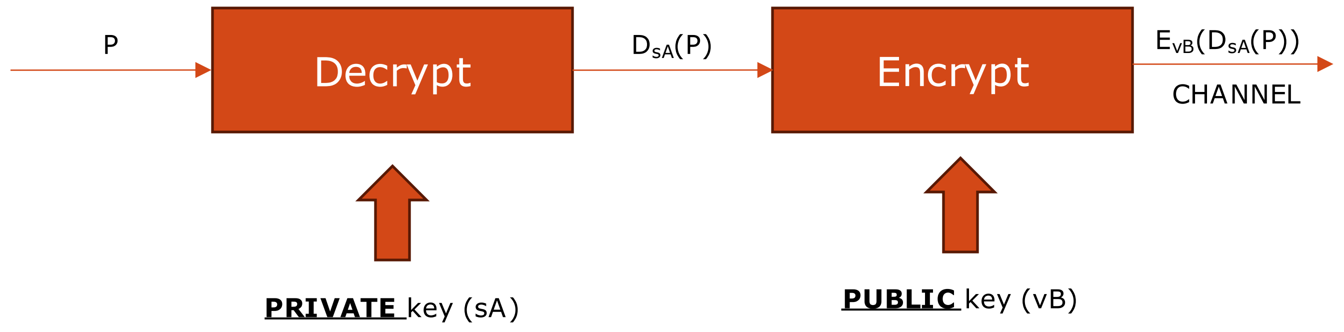
- Used to simultaneously achieve confidentiality, integrity, and authentication.
- The sender (A) performs a **decrypt transformation** on the message using its private (sequestered) key (sA).



Asymmetric cryptography

Public-key cryptography: double transformation

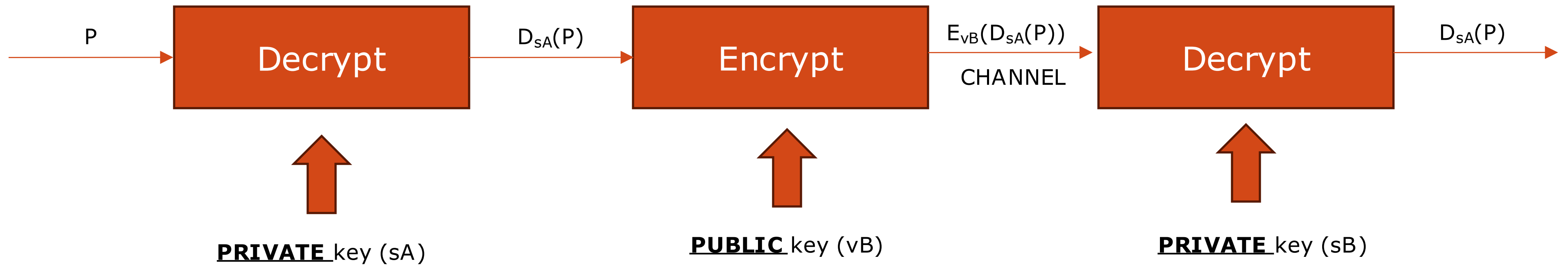
- The sender (A) then encrypts the result using the recipient’s public key (vB), and sends the message to the receiver (B).



Asymmetric cryptography

Public-key cryptography: double transformation

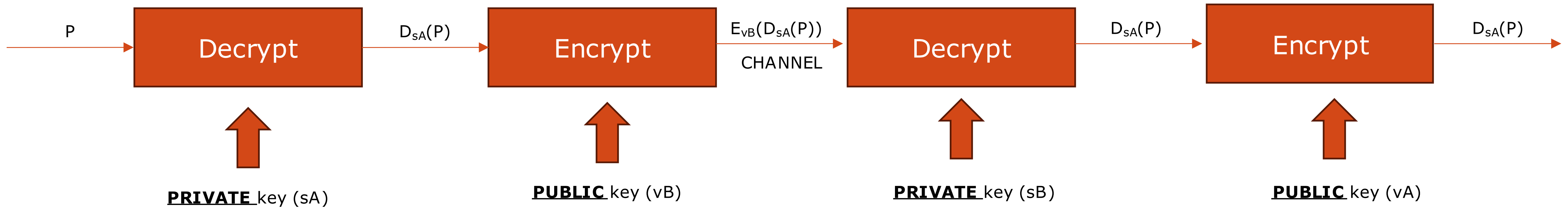
- The receiver (B) also performs a double transformation, starting by decrypting the received ciphertext to obtain D_{sA} (P).



Asymmetric cryptography

Public-key cryptography: double transformation

- This ciphertext is encrypted using A’s public key (vA) to obtain original message.



Asymmetric cryptography

Digital certificates

- Public keys need to me somehow distributed, and **the link between the key owner and the key must be certified.**
- **Digital certificates** can be employed for this purpose.
- They are managed by a trusted third party: **Certificate Authority (CA).**
- To obtain a digital certificate, an entity provides its identity and public key to the CA which validates the identity and issues the certificate.
- The certificate includes:
  - Name of the entity that owns it and its public key.
  - Certificate validity/expiration.
  - The CA which issued it (and many more data).
  - It is signed with the CA's private key (anyone with its public key can verify the certificate).

Asymmetric cryptography

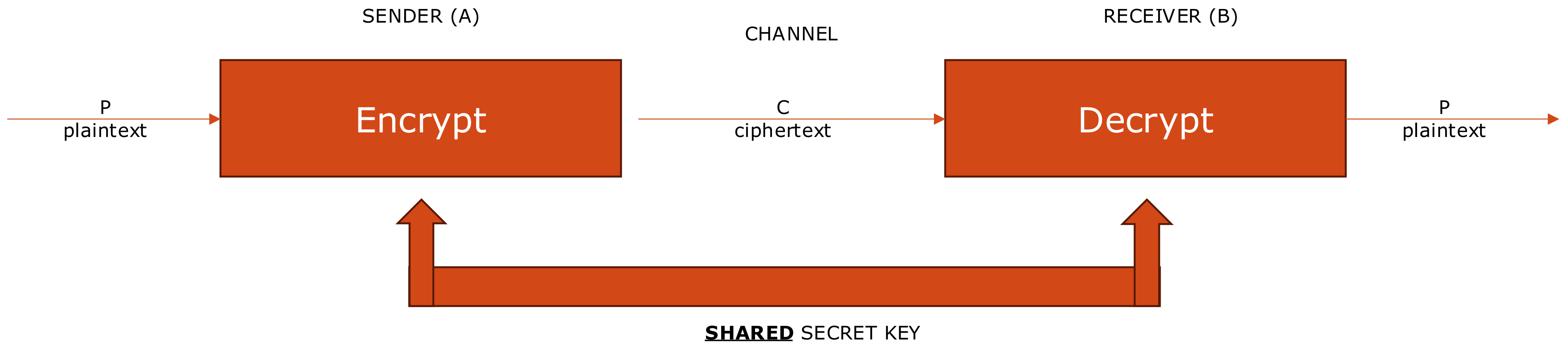
Public-key cryptography

- The standard cryptographic algorithms **may not be directly applied to IoT systems** since they are constrained devices.
  - To increase security, use of longer keys is recommended, but it also increases the computational burden on node.
- Some recommendations for IoT systems:
  - Use **Lightweight cryptography (LWC)** (which does not mean "weak").
  - Coupled with the addition of **dedicated hardware assists**.
  - Use symmetric cryptography due to the smaller computational cost.
  - Use asymmetric cryptography only to exchange the symmetric keys.

Symmetric cryptography

Introduction

- Use the same key for encryption and decryption.
- Used to ensure **confidentiality**, **integrity**, and **authentication** of messages.
- Communicating nodes must share a **secret key**.



Symmetric cryptography

Introduction

- Symmetric-key algorithms are **public** and thus follow the principle of open design.
  - The sending node (A) uses a mutually agreed-upon encryption algorithm to encrypt the message using its copy of the shared key.
  - The receiving node (B) decrypts the message using the matching algorithm and its copy of the shared key.
  - The cryptographic algorithm defines encryption and decryption as inverses of each other.
  - Any other recipient (or interceptor) does not have the shared key and cannot recover the original plaintext message sent by A.

Symmetric cryptography

Key exchange

- Nodes using symmetric cryptography need to **securely** establish secret keys.
- This can be done through a message exchange using **public-key cryptography**.
  - However, this approach imposes the burden of implementing a public-key crypto system which may be too much overhead if it is only needed for the key exchange.
- Another option: jointly construct the key (Diffie and Helman).
  - Execute incrementally a specified algorithm with the exchange of intermediate results and parameters over an insecure channel.
  - By knowing each other's intermediate outputs, both nodes compute the same secret key.
  - It is infeasible for a third party to compute the same secret key by knowing the chosen algorithm, parameters, and partial outputs exchanged by the two nodes.
  - The identical secret key is computed without having to actually exchange it.
  - Susceptible to a man in the middle attack.

Cryptography

Comparison

Aspect	Symmetric cryptography	Asymmetric cryptography
Key usage	Single shared key for encryption and decryption.	Uses a pair of public and private keys.
Speed	Faster because of simpler algorithms and shorter key lengths.	Slower due to more complex algorithms and longer key lengths.
Key management	Requires secure distribution of the shared key to all parties.	No need to share private key; public key can be freely distributed.
Security	Less secure if the shared key is intercepted or leaked.	More secure as private key is never shared.
Computational overhead	Low computational overhead, efficient for large-scale encryption.	High computational overhead, more suited for initial secure handshakes.

Message authentication

Introduction

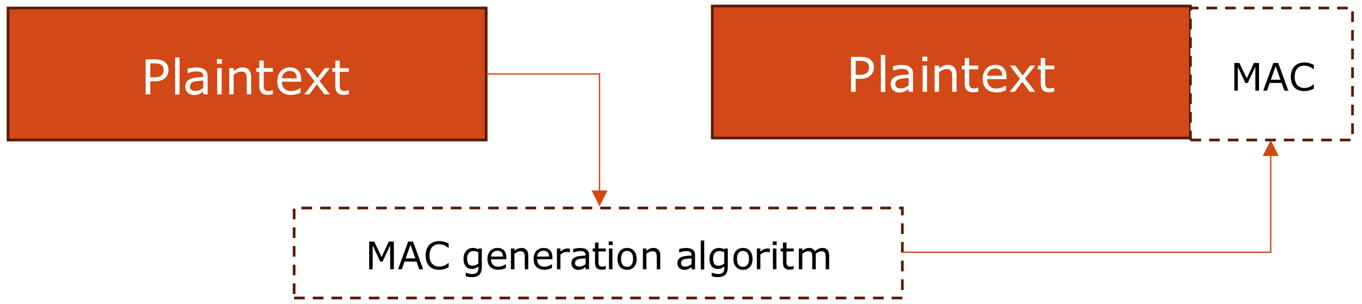
- Encryption schemes provide various combinations of message confidentiality, integrity, authentication, and nonrepudiation.
- Assurance that the message has not been altered (intentionally or by chance).

Encryption scheme	Confidentiality	Integrity	Authentication	Digital signature
Asymmetric $E_{vB}(P)$	YES	YES	NO	NO
Asymmetric $E_{sA}(P)$	NO	YES	Partial (YES with MAC)	YES
Double transformation	YES	YES	YES	YES
Symmetric, shared key	YES	YES	Partial (YES with MAC)	NO

Message authentication

Message Authentication Code (MAC)

- Realized by computing a **Message Authentication Code (MAC)**.
- It implies the usage of a **shared secret key** (possibly different from encryption key).
- Computed by the sender node using a known function and the secret key to produce a relatively short fixed-length value that serves as the authenticator (aka **tag**).
- Once computed, the MAC is appended to the message.
- MAC-generation algorithms have the property that it is not possible to alter the message without affecting the authentication tag.



Message authentication

Message Authentication Code (MAC)

- Upon receipt of the message, B uses the shared secret authentication key to compute the MAC on the payload part of the messagem and compares it with the MAC that was received.
- If the two **match**, the receiver knows that the message was not tampered with, and A is the only other party that has the secret key used for MAC generation.
- For security of authentication, it should be computationally infeasible to compute a valid authenticator (tag) of the given message without knowledge of the key.
- To be of practical value, computation of the MAC itself should be relatively simple when the encryption key is known.

Message authentication

Message Authentication Code (MAC)

- **Hashing** is a commonly used way of computing MACs.
  - Hashing functions produce a fixed length signature that may be used for file identification.
  - With the addition of a secret key to compute the hash, hashing can be made secure and computable only by the communicating parties who have the secret key.
- The MAC allows to authenticate the message but it does not provide confidentiality.
  - Okay for some types of IoT applications where data integrity may be more of a concern than confidentiality, such as reports of the non-sensitive sensor data.
- If confidentiality is needed, the message should be encrypted (a/symmetric).
  - Higher security means higher overhead with increased resource requirements and latency.

UNIVERSITÀ DEGLI STUDI DI PADOVA

Master's degree ICT Internet Multimedia Engineering

13 – Security
Endpoint security

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Endpoint security

Introduction

- Protection in all components, including **endpoints**.
- Basic objectives of IoT endpoint security include:
  - Node identification and authentication.
  - Secure state (bringing up and maintaining).
  - Secure communications.
  - Security monitoring and attestation.
- There is a wide range of techniques for implementation of endpoint security:
  - **Hardware** endpoint security.
  - **Software** endpoint security.

Hardware endpoint security

Hardware Security Modules (HSM)

- Hardware-based dedicated security component (**a physical device** with its own separate processor and storage) designed to provide high levels of security.
- They typically provide a **safe isolated execution environment** for implementation of security functions, usually coupled with the capability to generate and safeguard cryptographic keys that can be used for authentication, platform integrity, and secure communication.
- HSM can assist in bringing up of a node in a known good starting state.
- Several modalities:
  - Trusted Platform Module (TPM).
  - Secure Element (SE).
  - Trusted Execution Environments (TEEs).

Hardware endpoint security

Trusted Platform Module (TPM)

- A chip embedded in devices to provide a **secure processor** for storing cryptographic keys, performing cryptographic operations, and ensuring the integrity.
- Some of the key components and capabilities of a TPM include:
    - Secure storage: cryptographic keys, passwords, certificates.
    - Cryptographic functions: key generation, encryption, decryption, hashing, digital signing.
    - Integrity measurement: stores the integrity of the system's software (e.g., BIOS, firmware, bootloader) to detect unauthorized modifications.
    - Attestation: provides evidence to a remote or local entity that the system's configuration has not been tampered with.

Hardware endpoint security

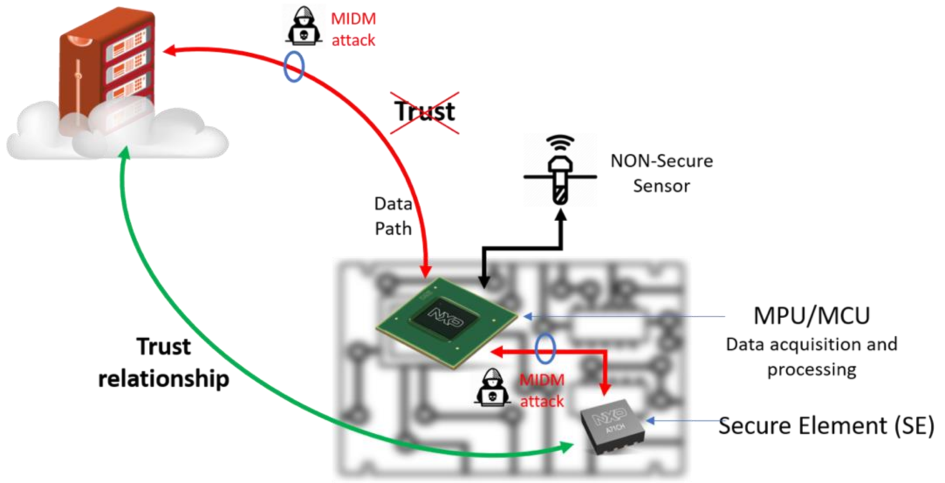
Trusted Platform Module (TPM)

- Several TPM forms (in decreasing order of security):
  - **Discrete** TPM: a separate dedicated hardware component.
  - **Integrated** TPM: into other processing elements.
  - **Firmware** TPM: in a separate protected execution environment.
  - **Software** TM: an emulator.

Hardware endpoint security

Secure Element (SE)

- Tamper-resistant hardware component capable of securely hosting code and confidential data (coupled with a dedicated processor for execution of secure code).
  - Examples: authentication, identification, signatures, and PIN management.
- It acts as a **vault**, protecting what is inside the SE (applications and data) from typical malware attacks in the host (i.e., the device operating system).



Thanks to: Adrian Buzescu
https://www.linkedin.com/pulse/role-secure-element-securing-iotembedded-designs-adrian-buzescu/

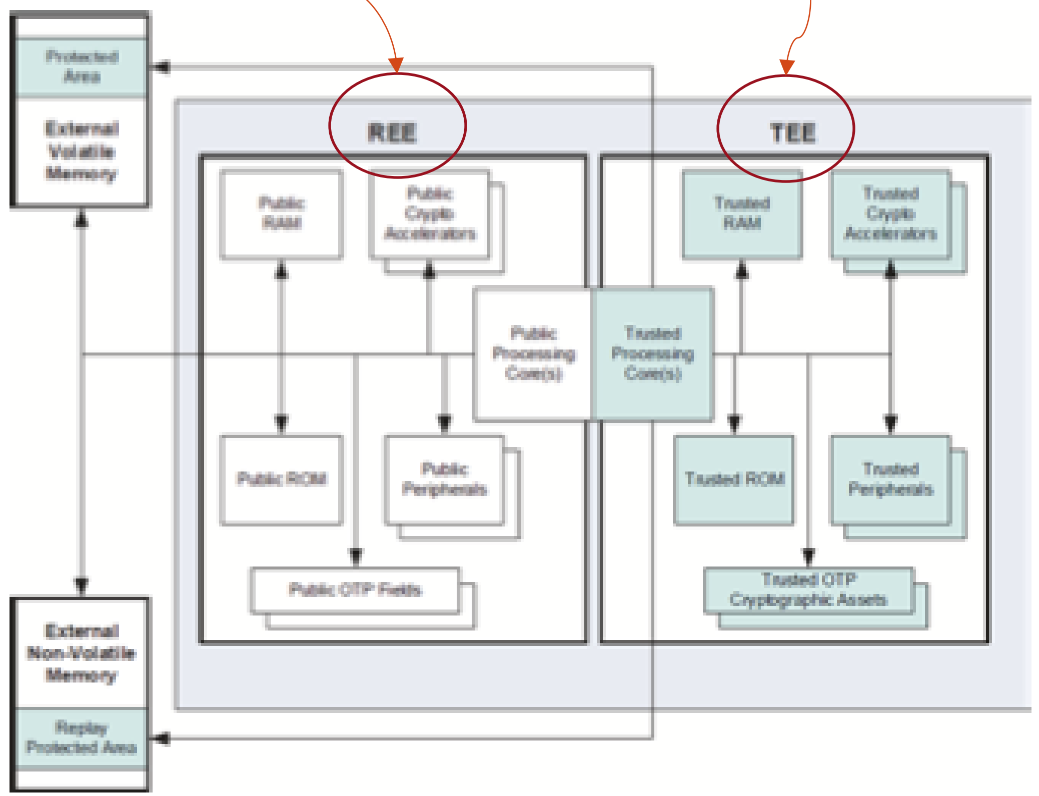
Hardware endpoint security

Trusted Execution Environment (TEE)

- Is a **secure area of a main processor** that isolates sensitive data and computations from the rest of the system. It may be used to:
  - Storage of cryptographic keys.
  - Dedicated storage and execution of security code.
  - Allow complete separation of trusted and untrusted portions of the system.
- The main processor is divided into the Normal World (**REE**) and the Secure World (**TEE**).

Rich Execution Environment (Normal World)
Runs the regular operating system and applications

Trusted Execution Environment (Secure World)
Handles sensitive tasks


Protected Area
External Volatile Memory
REE
TEE
Public RAM
Public Crypto Accelerators
Trusted RAM
Trusted Crypto Accelerators
Public Processing Core(s)
Trusted Processing Core(s)
Public ROM
Public Peripherals
Trusted ROM
Trusted Peripherals
Public OTP Fields
Trusted OTP Cryptographic Assets
External Non-Volatile Memory
Replay Protected Area

Hardware endpoint security

Comparison

Feature	TPM	SE	TEE
Definition	A hardware chip for cryptographic operations and platform integrity.	A dedicated hardware component for secure data storage and processing.	A secure area in the main processor for running trusted code.
Primary purpose	Platform security and integrity (e.g., attestation, key storage).	Store sensitive data (e.g., payment info, credentials) securely.	Isolate sensitive operations and data from the main OS.
Location	Separate hardware chip or integrated module.	A standalone hardware module (sometimes embedded in the SoC).	A hardware-isolated region within the main processor.
Key use cases	Secure boot, remote attestation, encryption, and signing.	Payment systems (e.g., EMV), SIM cards, and NFC transactions.	Mobile app security, DRM, and secure key handling.
Performance	Dedicated, so highly optimized for security tasks.	Typically slower but highly secure due to dedicated hardware.	Shares resources with the main processor, so faster than SE but less secure.
Hardware isolation	Full isolation as a discrete module.	Full isolation due to dedicated hardware.	Partial isolation (rely on main processor architecture).

UNIVERSITÀ DEGLI STUDI DI PADOVA

Master's degree ICT Internet Multimedia Engineering

13 – Security
Network security

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Network security

Introduction

- IoT networks are implemented using a layered design. Therefore, communications can be secured at different layers.
- It is believed that **IP** will be the base common network protocol for the IoT.
- However, there are (and will be) also devices organized in networks implementing application-specific communication protocols.

	Internet	IoT
Application	HTTP	CoAP
Transport	TCP	UDP
Network	IP	IPv6/6LowPAN
Link	MAC	MAC
Physical	PHY	PHY

Network security

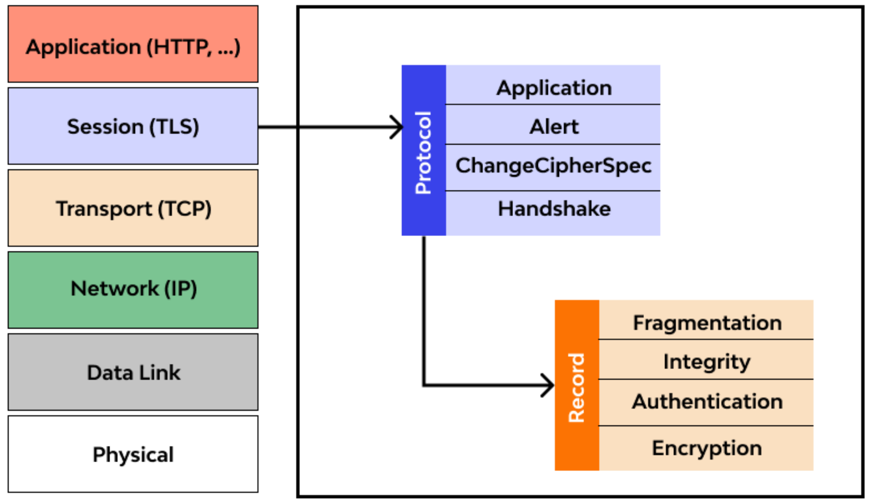
Introduction

- For example, at the app level, CoAP does not provide security functionalities.
- Security functionalities implemented by the **application** (protecting the data before the CoAP encapsulation) or at one of the **underlying layers**.
- **Application-level security**:
  - End-to-end protection can be guaranteed.
  - It simplifies the requirements for underlying layers and reduces the cost in terms of packet size and data processing: per data and not per-packet overhead is introduced.
- **Transport/network-level security**:
  - The same security mechanism can be shared by multiple applications.

Transport-level security

Transport Layer Security (TLS)

- Designed to work with **TCP**.
- The main functionalities are:
  - **Authenticating** the **endpoints** and define the set of cryptographic keys.
  - **Exchanging confidential data** through symmetric encryption.
  - **Authenticating messages** through secure hashing.



Transport-level security

Transport Layer Security (TLS)

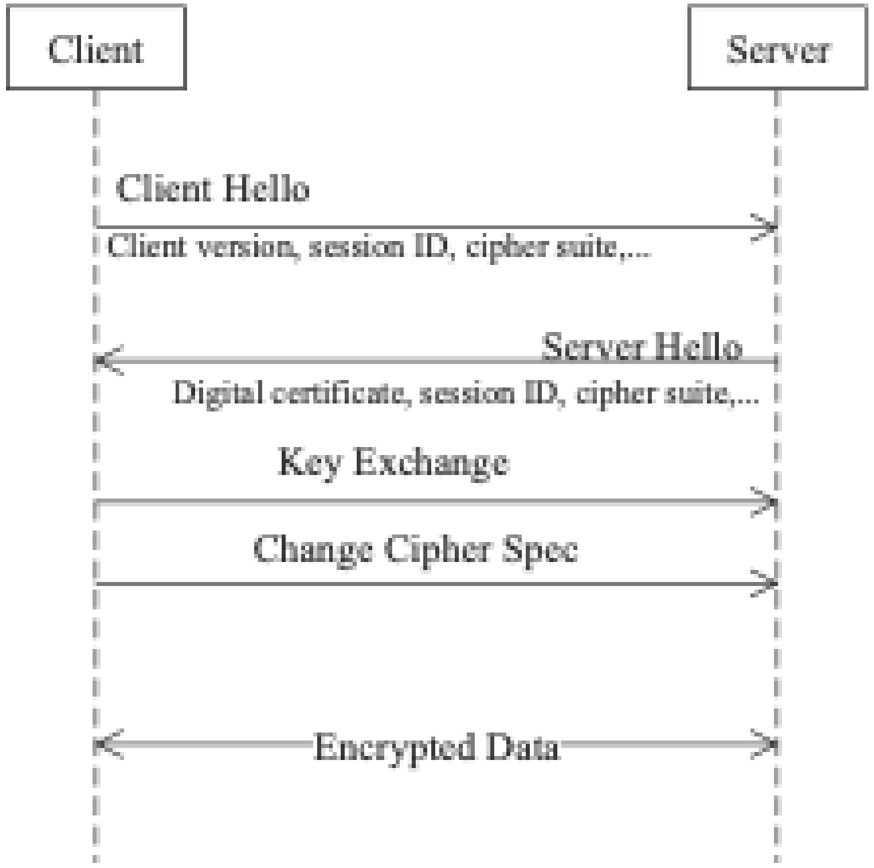
- Two primary components:
  - **Handshake** protocol: used to negotiate cryptographic modes and parameters, authenticate communicating parties, and establish the shared key material.
  - **Record** protocol: uses parameters established by the handshake to protect traffic between endpoints and divides traffic into a series of records, each of which is independently protected using traffic keys.

Transport-level security

Transport Layer Security (TLS): handshake

- **Server Hello** includes:
  - The level of security protocol;
  - The cipher suite to use from the client list;
  - **Random number** combined with datetime, session ID, and compression methods to use;
  - The digital certificate (containing its identity and public encryption key).

The random numbers are used to initiate and complete the key exchange using the agreed-upon protocol.


Client
Server
Client Hello
Client version, session ID, cipher suite,...
Server Hello
Digital certificate, session ID, cipher suite,...
Key Exchange
Change Cipher Spec
Encrypted Data

Transport-level security

Datagram Transport Layer Security (DTLS)

- Designed to work with **UDP**.
- Problem: overhead caused by DTLS (the underlying protocols, such as IEEE 802.15.4, have limited packet size).
  - Packet optimization and compression mechanisms.
- DTLS creates a point-to-point secure association (through a handshake process similar to TLS handshake): not compatible with multicast IP communications.
- DTLS provides three security modes:
  - **PreSharedKey**: the devices store symmetric pre-shared keys
  - **RawPublicKey**: the devices own a private-public key pair without certificate.
  - **Certificate**: the devices store an X.509 certificate.

Network-level security

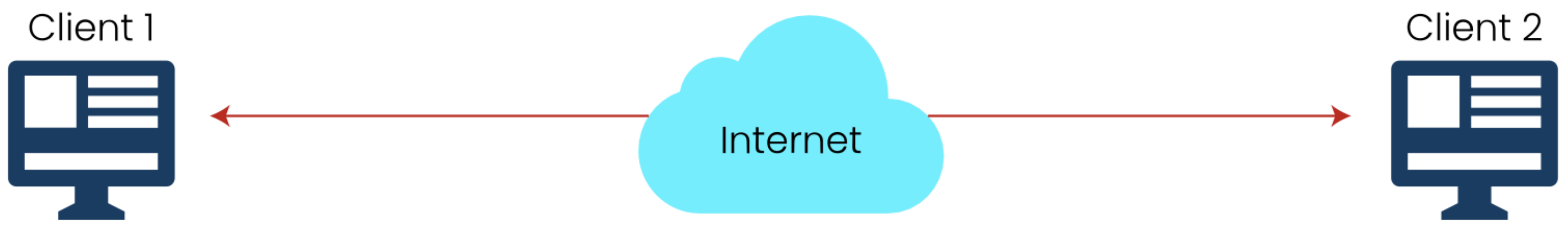
Internet Protocol Security (IPsec)

- It provides confidentiality, integrity, data-origin authentication and protection against replay attacks.
- The IPsec protocol suite includes two principal protocols:
  - **Authentication Header (AH)**: it provides source authentication and data integrity.
  - **Encapsulation Security Payload (ESP)**: it also provides confidentiality.
- IPsec has two different packet forms:
  - Tunnel mode.
  - Transport mode.

Network-level security

Internet Protocol Security (IPsec): Transport mode

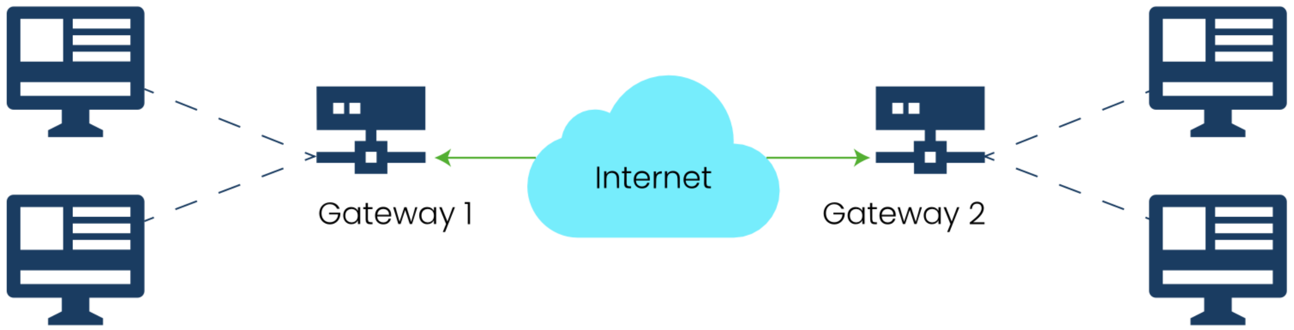
- It is employed for end-to-end communication between two hosts.
- In this case, the AH or ESP payload is represented by the portion of the original IP packet following the IP header.
- Drawback: the IP header is not protected, thus allowing traffic analysis.



Network-level security

Internet Protocol Security (IPsec): Tunnel mode

- It protects the whole IP packet.
- The AH or ESP payload is represented by the **complete original IP packet**.
- The packet is sent with a new IP header, and the intermediate routers cannot see anything about the original packet.
- Used when one or both ends of the SA are security gateways (e.g., firewalls).
- It simplifies the key exchange procedure.



Network-level security

Internet Protocol Security (IPsec)

- Before sending IPsec datagrams, the source and destination must create a logical connection called **Security Association (SA)**.
- It is a one-way connection.
- If IPsec traffic has to flow in both directions, so two SAs are needed.
- The state information of a SA include:
  - A Security Parameter Index (SPI) which identifies the SA.
  - The origin and destination interface of the SA (the IP addresses).
  - The security protocol identifier which indicates if it is an AH or ESP security association.
  - The type of encryption to be used.
  - The type of integrity check.

Network-level security

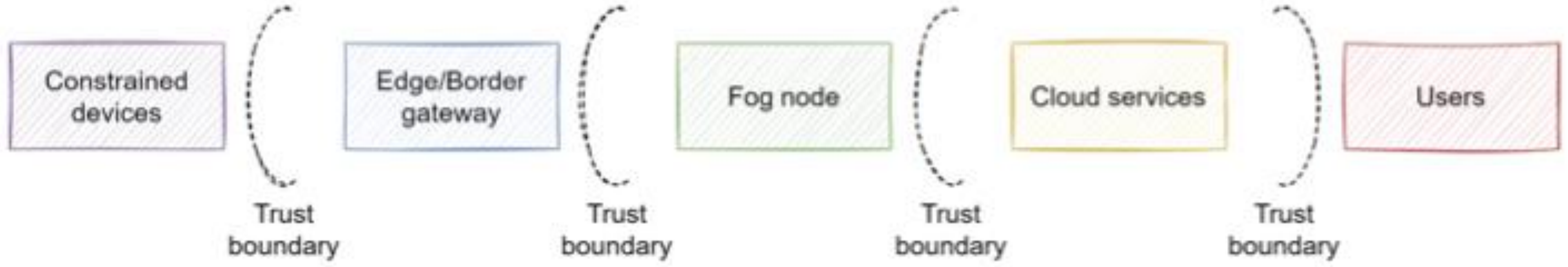
Internet Protocol Security (IPsec)

- The keys can be dynamically exchanged using the **IPsec INternet Key Exchange (IKE) protocol**.
- Problem: it uses asymmetric cryptography and is computationally heavy for constrained devices.
  - IKE extensions using lighter algorithms should be considered.
- Problem: strong data overhead due to the additional headers.
  - Header compression techniques are needed.
- Problem: no end-to-end security guarantees if data flows through proxies and application-level gateways.
  - A security gateway is needed.

Network security

Network segmentation

- Different portions of the IoT network may have **different security requirements**.
- A first classification can be made between the **data plane and the control plane**.
  - Since the actuation commands have direct impact on the physical world, they may be subject to stronger protection.
- It is a good design practice to divide networks into segments (**trust zones**).
- Each zone has access control, authentication rules, security policies and protocols.



Thanks to: Sara Baldoni (sara.baldoni@unipd.it)

Network security

Denial of Service (DoS)

- Example for IoT: **denial of sleep**.
- It consists in keeping IoT devices always active (receiving or processing data) thus preventing the switch to energy-saving sleep mode.
  - The DTLS handshake poses a relevant issue for DoS attacks towards the server.
  - An adversary may continuously transmit ClientHello messages to the DTLS server and start a large amount of DTLS handshakes.
  - When the handshake starts, the server allocates resources for the connection: waste.
  - If the attacker spoofs the IP of a victim node and starts a DTLS handshake with a server, the server will send reply messages to the innocent node: amplification effect.
- Possible countermeasures: frequency hopping, randomize the wake-up time.

Network security

Secure data aggregation

- Data aggregation requires to perform some processing in intermediate nodes to minimize the traffic so that only relevant information is passed in the network.
- **Concatenating** the payloads: inefficient
- **Semantic** concatenation: process data being aware of the final goal.
- Security issues if aggregators are not special nodes:
  - Aggregators perform decryption and encryption procedures in addition to aggregation processing: computational and energy impact.
  - Aggregators must keep secure associations (i.e., symmetric keys) with every node to which it communicates: memory issue.
  - Aggregators access the data they receive: privacy concerns.
- Solution: **homomorphic encryption** techniques.

Network security

Authorization

- The process of granting approval to a client to access a resource.
- It allows to answer to the following questions:
  - Which users can access some given data?
  - How should the information be presented to a given user?
  - Which operations is a user allowed to perform?
- There are three main authorization approaches:
  - **Discretionary access control (DAC)**: restrict access based on the subject identity.
  - **Role-based access control (RBAC)**: map permissions to roles assigned to subjects.
  - **Attribute-based access control (ABAC)**: map permissions to attributes for the subjects.

Network security

Authorization

- The Open Authorization (OAuth) protocol has been introduced for allowing authorized third parties to access personal user data.
- It has to be adapted to the IoT scenario to be compatible with constrained devices.
- **Header compression** minimizing the amount of sent information.
- It defines different roles:
  - **Resource owner**: the entity that grants access to a protected resource.
  - **Resource server** (Service Provider – SP): a server hosting user-related information.
  - **Client** (Service Consumer – SC): third parting which needs access to user data.
  - **Authorization Server** (AS): server which issues access tokens to the client after obtaining authorization from the resource owner. The tokens are authentication credentials containing information about the SC identity and the user identity.

Network security

Authorization: IoT-OAS

- An example of an architecture based on OAuth for IoT is **IoT-OAS**.
- It defines an external service which can be invoked so that:
  - The smart object keeps its simplicity, since it only needs to invoke an external service
  - The access control policies at the SP can be dynamically and remotely configured, whereas it may be difficult to act directly on the smart object.

Network security

Security monitoring and management

- Security monitoring can follow different strategies:
  - **Real-time**: keep track of current system state, raise alarm when an anomaly is detected.
  - **Predictive**: identify patterns identifying when an attacker has breached a portion of the system, or indicating a future attack.
  - **Forensic**: record data concerning past attacks to gain insights and, if possible, enhance the system security in the future.
- Security analytics can be:
  - **Rule/signature-based**: rely on pre-defined rules or signatures to detect anomalies.
  - **Behavioral analysis**: nominal behavior patterns are learnt and an anomaly is declared when the current system behavior deviates from the modeled one.

Network security

Security monitoring and management

- After a security breach, an **incident response** has to be applied:
  - Identify the affected parts.
  - Isolate the affected parts (e.g., disconnection).
  - Start the recovery action to restore normal operation conditions.
  - Perform forensic analysis to determine the root cause and enhance system for the future.

13 – Security

Privacy

Marco Giordani (marco.giordani@unipd.it)
Department of Information Engineering (DEI) – SIGNET Research Group
University of Padova – Via Gradenigo 6/B, 35131, Padova (Italy)

Privacy

Introduction

- Manufacturing or production-centric organizations will be more focused on the risk scenarios and their impact on business.
- Organizations whose IoT systems are customer-centric will be more concerned with privacy and legal issues.
- **Privacy** is one of the main issues with IoT systems, due to the use of IoT devices:
  - Example: Home devices; wearable devices; in-vehicle devices.
- In addition, data is recorded regularly and in a continuous fashion.
  - This provides information about habits and trends, leading to **profiling** issues.

Privacy

Introduction

- User’s data should be collected only with the user **explicit consent**.
- Moreover, the user should be aware of the following information:
  - Which personal data are collected.
  - Who has access to the data.
  - If data are secured during transfer and storage.
  - How long the data are kept.
  - What are the notification procedures when data breaches are detected.

Privacy

Privacy in IoT

- Issue with IoT: difficulty of **interacting** with the customers.
  - Many IoT devices do not have screens.
  - Those who have screens may have small ones.
  - Some IoT products are not consumer facing (e.g., a sensor for electricity monitoring).

Privacy

Privacy in IoT

- IoT developers should distribute their products in a way that guarantees that the user receives all the relevant information concerning the data collection.
  - Providing consumers with opt-in choices at the time of purchase.
  - Offering video tutorials.
  - Affixing QR codes on devices.
  - Incorporating notices and providing choice as part of set-up wizards.
  - Providing a management portal for privacy settings that consumers can configure and revisit, possibly from a separate device such as a smartphone.
  - Use of icons
  - Communicating with the customer outside the device, for example, through email

Privacy

Data minimization

- A key principle which needs to be followed is **data minimization**.
- It consists in limiting the amount of data collected, and delete them
- when they are no longer needed.
- There are many data policies such as:
  - Not to collect data.
  - To collect only the type of data necessary or the functionality of the product being offered.
  - To collect less sensitive data.
  - To deidentify the data collected.
  - To seek consumers’ consent for collecting additional, unexpected categories of data.

Privacy

Deidentification

- **Deidentification** allows to reduce the risk concerning the users' privacy while preserving the value of the collected information.
- It reduces the risk of a privacy breach if the data is lost, stolen or accessed by unauthorized people.
- However, deidentification is <u>not</u> flawless, in some cases it has been demonstrated that deidentified data can be re-identified.

Privacy

Cybersecurity regulations in the EU

- **Cybersecurity** is regarded as a highly complex issue which requires the active involvement of a range of stakeholders, including the legislator.
- This is a significant change since initially it was perceived as a **purely technical matter** related to measures ensuring the availability, integrity and confidentiality of information and information systems.
- The first European Union Cybersecurity Strategy (2013) identifies cybersecurity as a new EU policy area:

> *"an excellent example of an area in which the different policy fields need to be combined (a requirement for horizontal consistency), and where measures need to be taken at the level of both the EU and Member States (calling for vertical consistency) "*

Privacy

Cybersecurity regulations in the EU

- **Five strategic EU cybersecurity priorities have been identified:**
  - **Achieving cyber resilience** by establishing minimum requirements for the functioning, cooperation and coordination of national competent authorities for network systems.

Privacy

Cybersecurity regulations in the EU

- **Five strategic EU cybersecurity priorities have been identified:**
  - **Reducing cybercrime** by:
    - Ensuring a swift transposition of the cybercrime related EU Directives;
    - Encouraging ratification of the Council of Europe’s Budapest Convention on Cybercrime;
    - Funding programmes for the deployment of operational tools.

Cybersecurity regulations in the EU

- **Five strategic EU cybersecurity priorities have been identified:**
  - **Developing cyberdefence policy and capabilities related to the Common Security and Defence Policy (CSDP)** by:
    - Assessing operational EU cyberdefence requirements;
    - Developing the EU cyberdefence policy framework;
    - Promoting dialogue and coordination between civilian and military actors in the EU;
    - Facilitating a dialogue with international partners.

Privacy

Cybersecurity regulations in the EU

- **Five strategic EU cybersecurity priorities have been identified:**
  - **Developing the industrial and technological resources for cybersecurity** by:
    - Establishing a public-private platform on Network and Information Security (NIS) solutions;
    - Providing technical guidelines for the adoption of NIS standards and good practices;
    - Encouraging the development of security standards for technology “with stronger, embedded and user-friendly security features.”

Cybersecurity regulations in the EU

- **Five strategic EU cybersecurity priorities have been identified:**
  - **Establishing a coherent international cyberspace policy for the EU and promoting core EU values** by mainstreaming cyberspace issues into EU external relations and Common Foreign and Security Policy (CFSP), and by supporting capacity building on cybersecurity and resilient information infrastructures in third countries.

Privacy

Cybersecurity regulations in the EU

- The main EU legal acts guiding the maintenance of information systems are:
  - **General Data Protection Regulation (GDPR)**: it aims at strengthening and clarifying rights concerning personal data of natural persons. It affects organizations globally if they manage data belonging to users in the European Union.
  - **Directive on Security of Network and Information Systems (NIS)**: it will likely be adopted differently by member states, although it defines what can be considered a minimum level of security responsibilities for information systems.
    - **Preventing risks**: use measures that are appropriate and proportionate to the risk.
    - **Ensuring IT security**: ensure a level of security of the information appropriate to the risks.
    - **Handling incidents**: minimize the impact of incidents on the systems used for the services.

Privacy

Cybersecurity regulations in the EU in IoT

- In IoT:
  - Data quality from data generators, such as sensors, must, in light of the GDPR, be accurate and reliable.
  - Meta information concerning for instance data source and access rights, should be recorded along with the data when it becomes associated with a natural person.
  - During feature engineering, efforts must be made such that descriptive statistics do not introduce new features that can be considered sensitive.
  - The need for transparent processing and transparent data transfers / manipulations / removals may be best met by an immutable data storage solution, such as a ledger-based technology like blockchain.