Connecting Securely to the Cloud

Security Primer

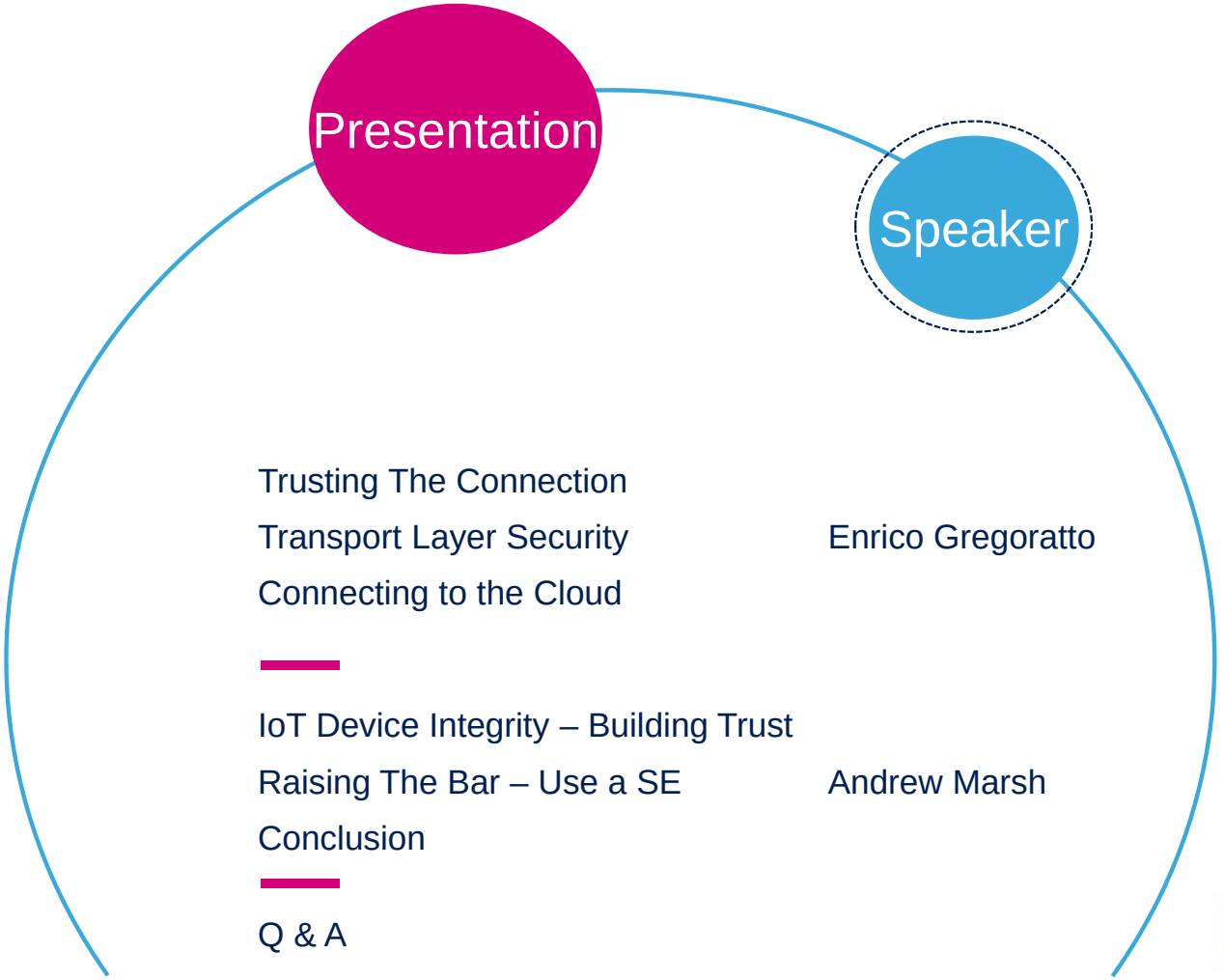
Presented by

Enrico Gregoratto
Andrew Marsh

life.augmented

Technology Tour 2017

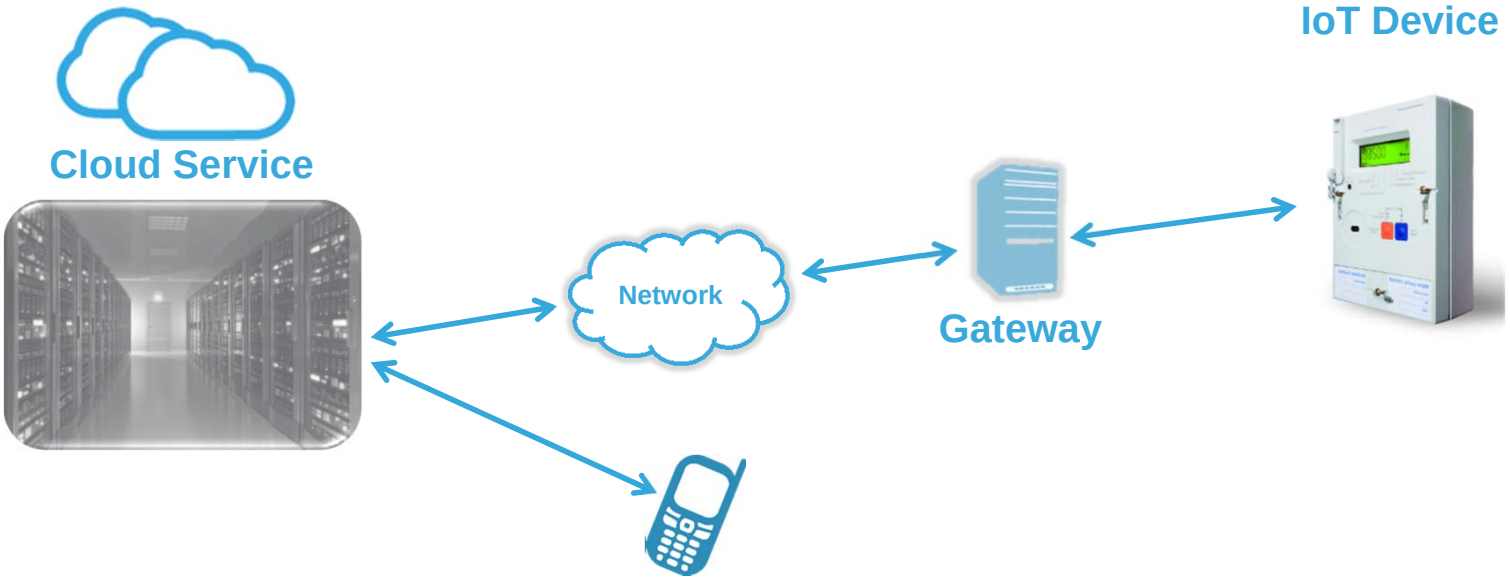
Agenda



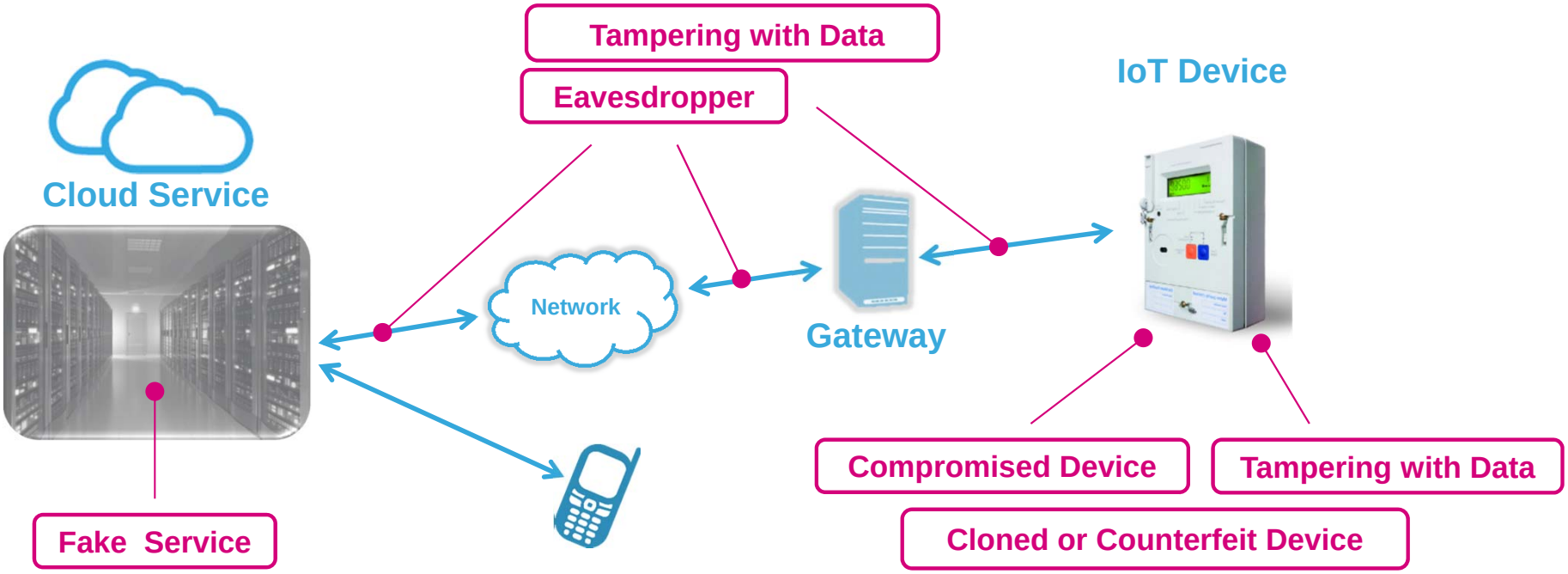
Trusting The Connection	
Transport Layer Security	Enrico Gregoratto
Connecting to the Cloud	
IoT Device Integrity – Building Trust	
Raising The Bar – Use a SE	Andrew Marsh
Conclusion	
Q & A	

Communications With A Cloud Service

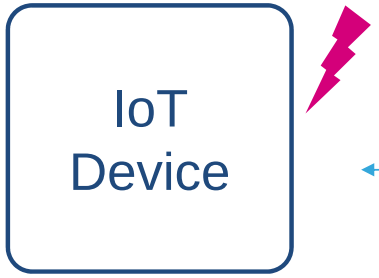
Overview



Identifying Threats



Classes of Attacks



Internet

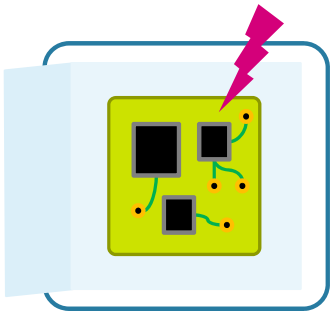
Cloud Service

ATTACKS

Software Attack

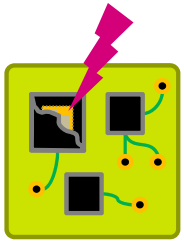
Misuse of network protocols
- Exploit communication protocol errors

Flaws in software design / implementation



Board Level Attack

With the case opened / removed
- Test / debug port access
- Inter device bus and IO probing
- Reset, clock attacks
- Power analysis
- Temperature / electrical attacks (glitch, overvoltage)



Silicon Level Attack

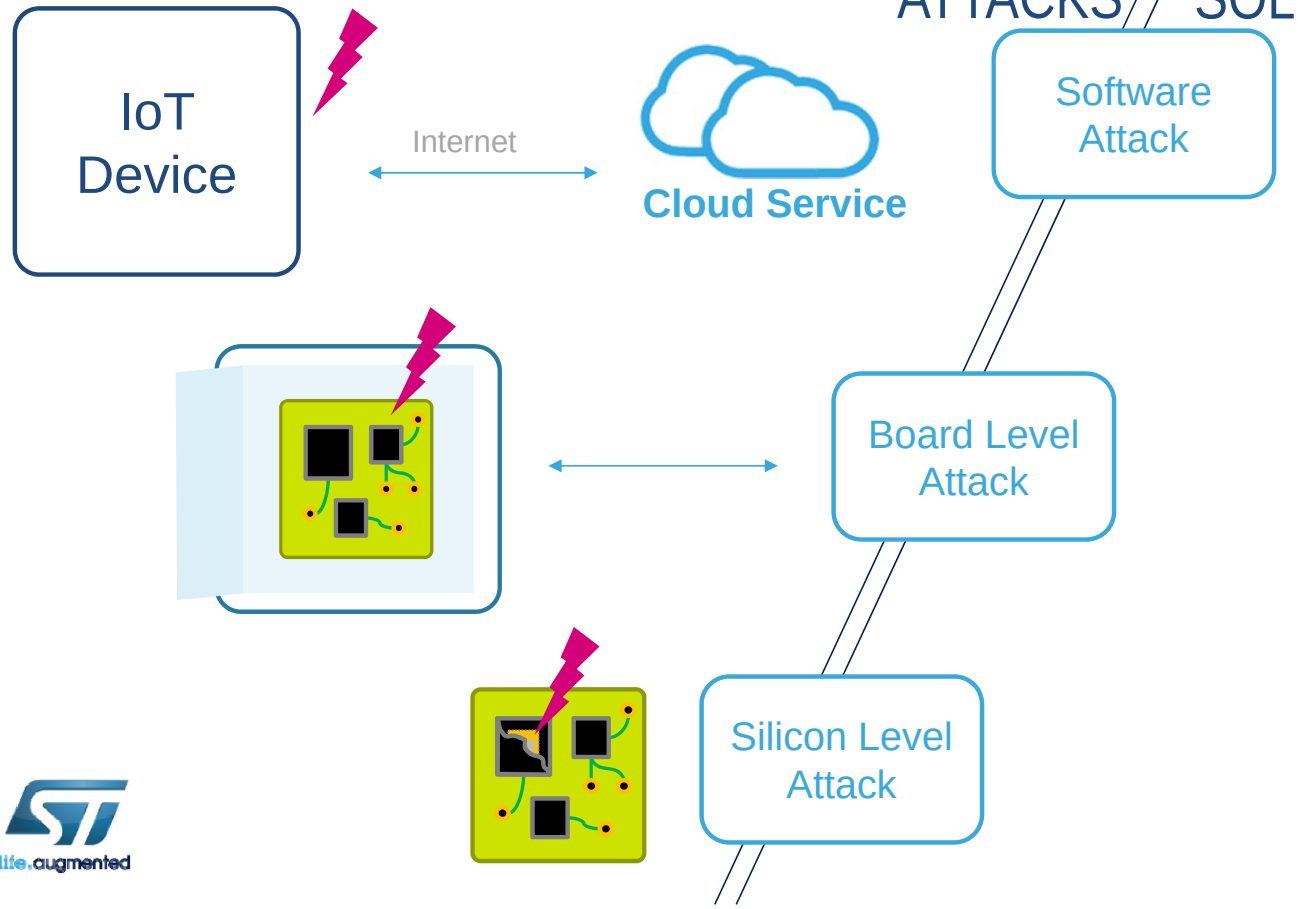
Device de-packaged
- Circuit analysis and probing

Fault injection
- Laser beam

life.augmented

Technology Tour 2017

Classes of Attacks



ATTACKS // SOLUTION

Software Attack

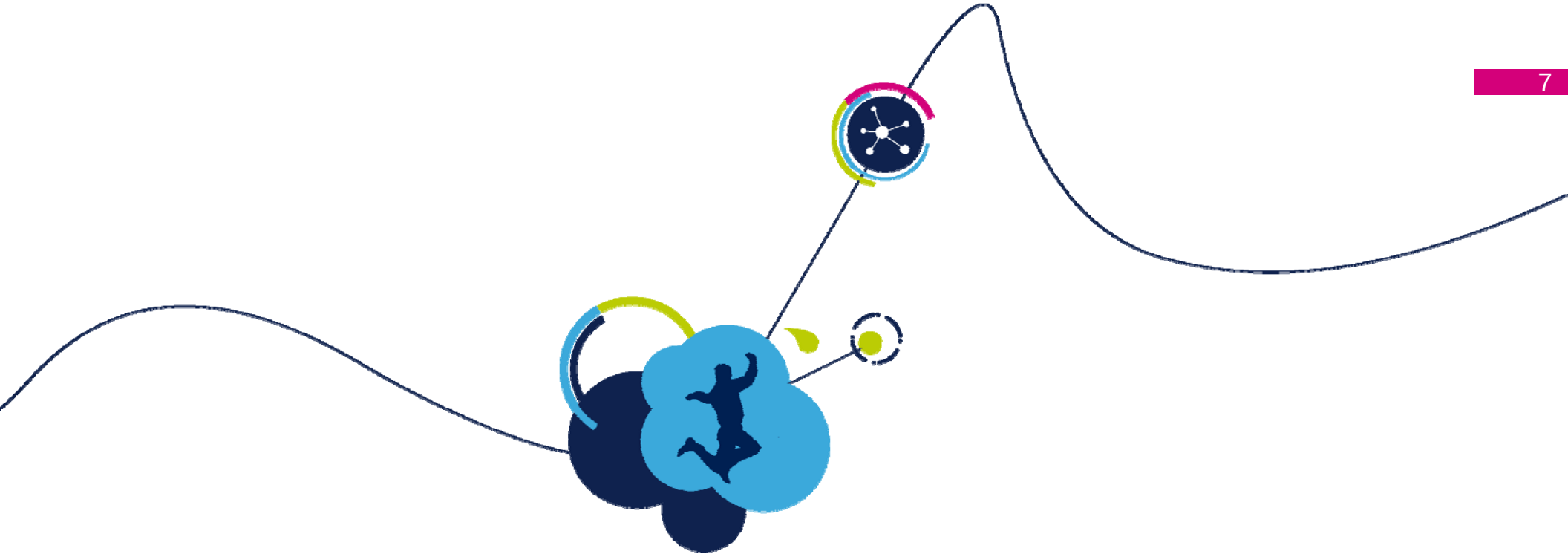
Board Level Attack

Silicon Level Attack

Confidentiality
- Data / Identity Protection
- Secure Communications
- Secure Storage

Authentication
- Device to Server
- Server to Device

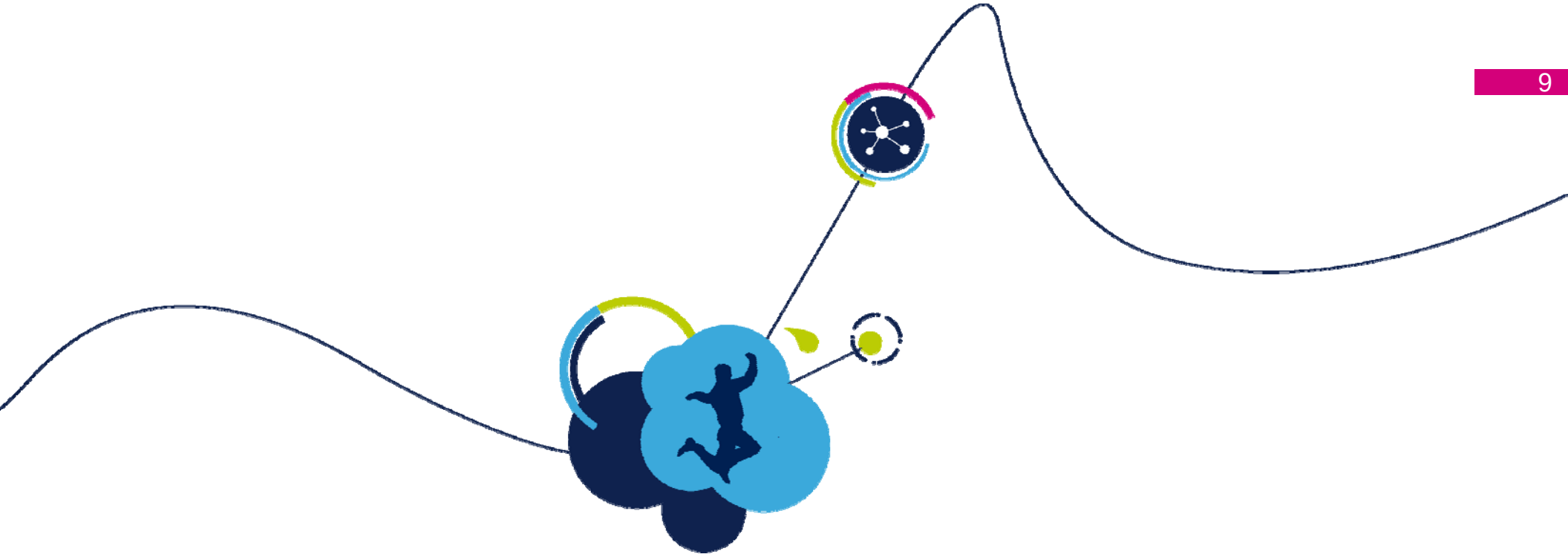
Integrity and Availability
- Secure Boot
- Secure Firmware Upgrade
- Trusted Processing



Securing the Connection

Secure Communications

- Cloud Services have selected to use a standards based secure communication protocols
- Cloud Services vendors use "lightweight" messaging protocols like
  - MQTT - Message Queue Telemetry Transport (ISO / IEC PRF 20922)
  - CoAP - Constrained Application Protocol (RFC7252)
- These messaging protocols were designed for connections with remote devices with limited resources (memory / CPU) or where network bandwidth is limited
- These messaging protocols use Transport Layer Security (TLS) to secure communications



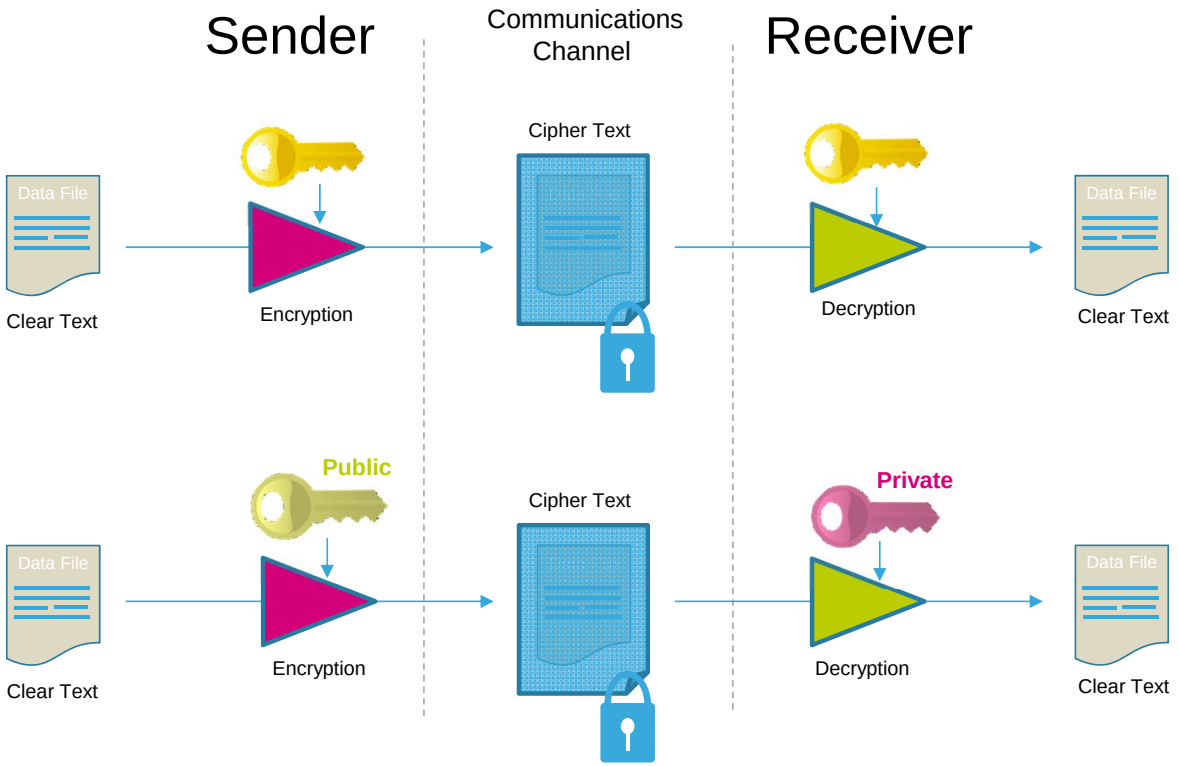
Crypto Basics

Cryptography

One Key or Two?

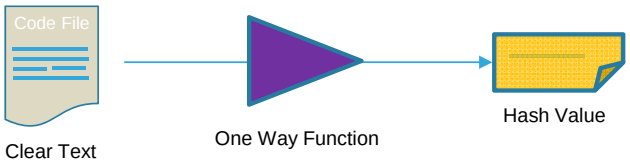
Secret Key Cryptography (Symmetric)

Public Key Cryptography (Asymmetric)



Cryptography
HASH Algorithms

- Cryptographic Hash
  - Easy to compute and quick
  - A one way function, non-reversible (computationally infeasible to obtain the original message from the hash), unique (practically impossible for two different messages to generate same result)
  - Used in many processes like authentication, secure boot, secure downloads
- NIST (Secure Hashing Algorithm) SHA-2, SHA-3

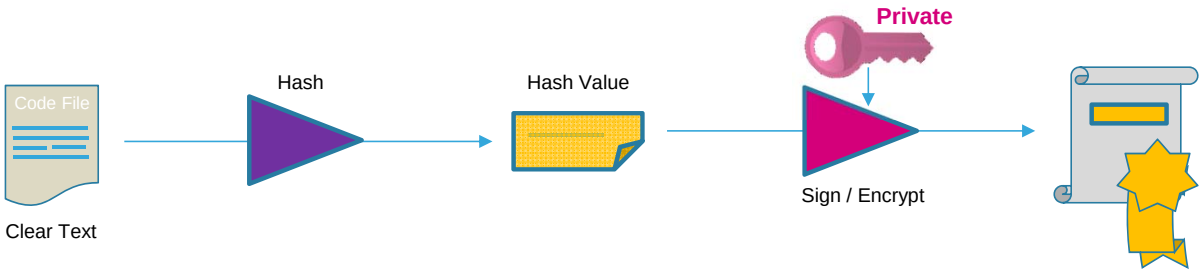


Hash function has no key, and the clear text can not be recovered from the hash value

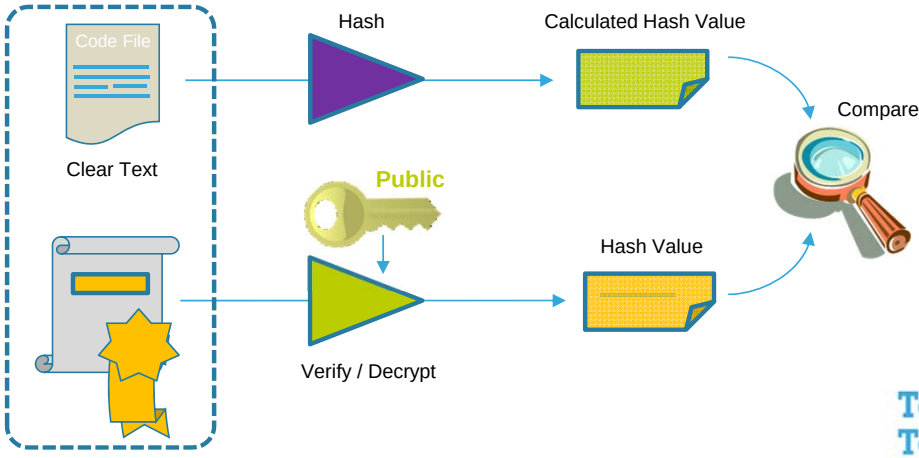
Cryptography
Digital Signatures

- Digital Signatures
  - Used to check the authenticity of information - code, files, messages, Public Keys

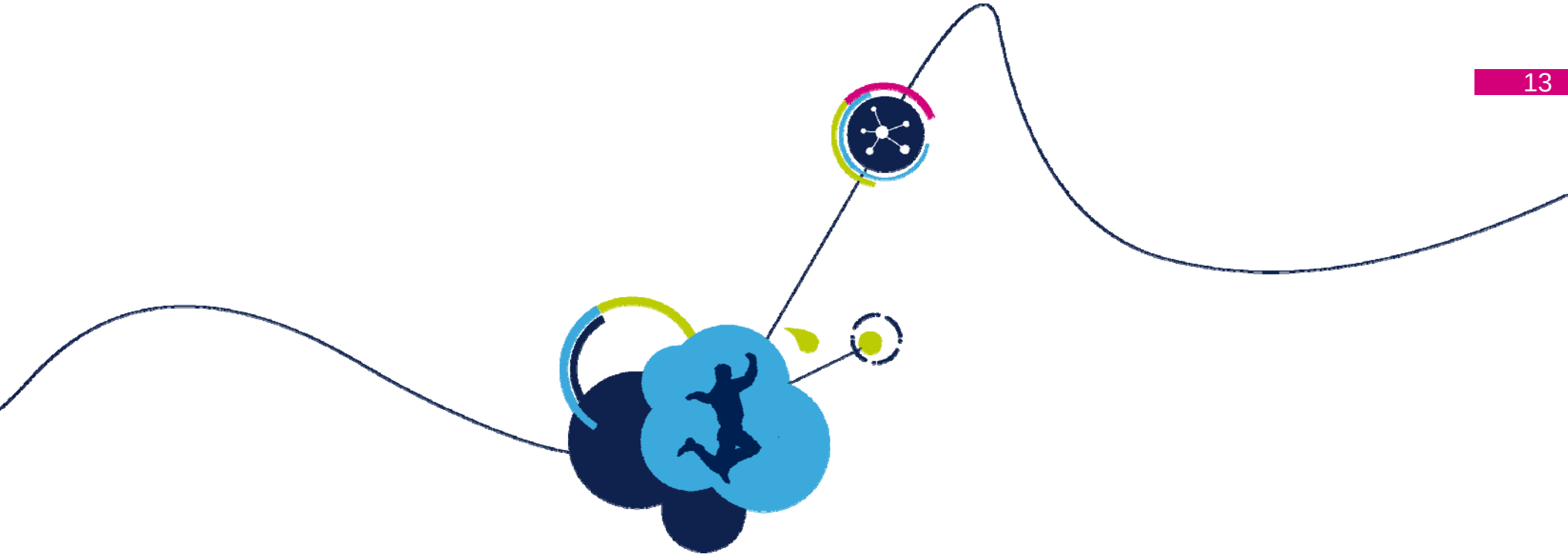
Signing
Using RSA



Authentication
Using RSA



If the Hash values are equal the file is authentic

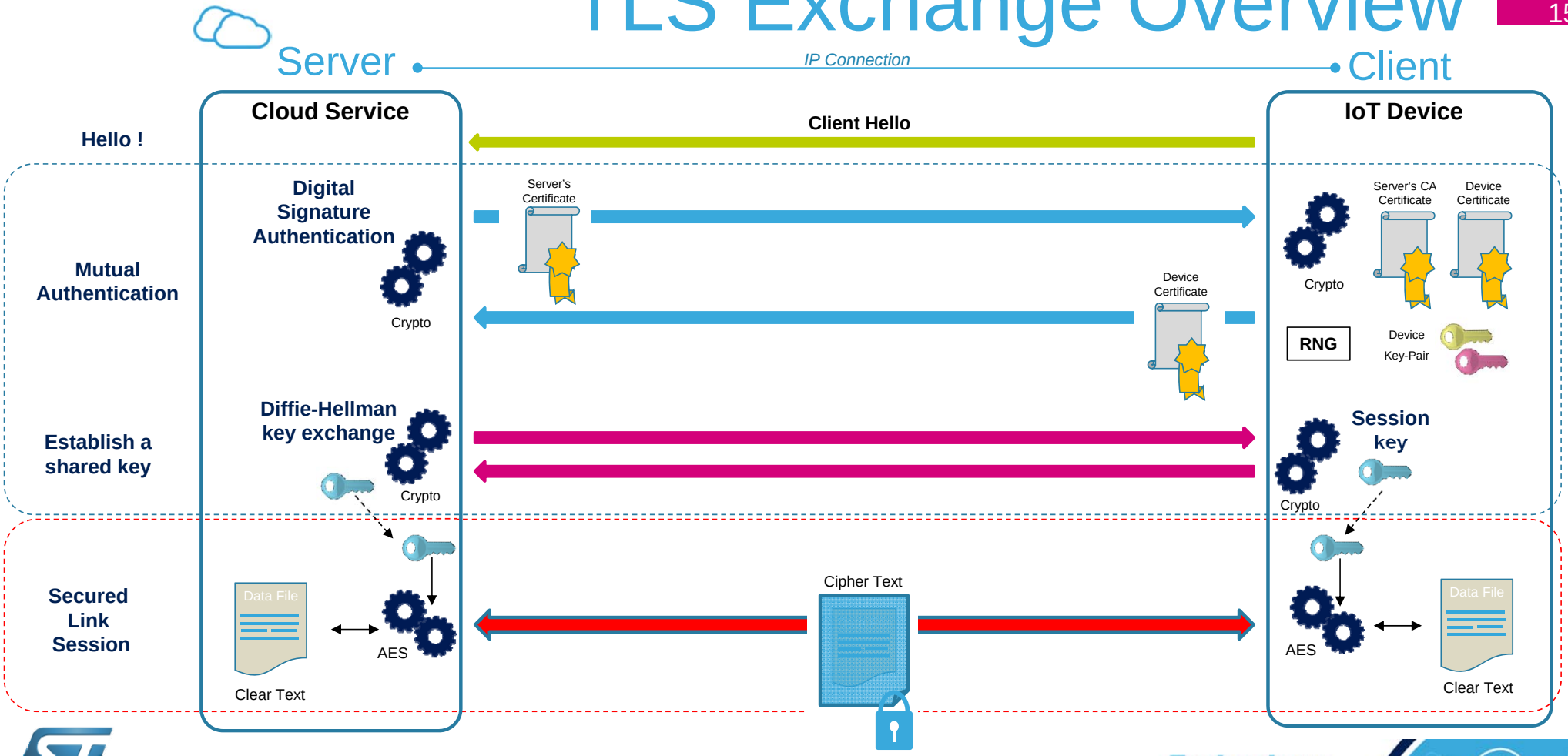


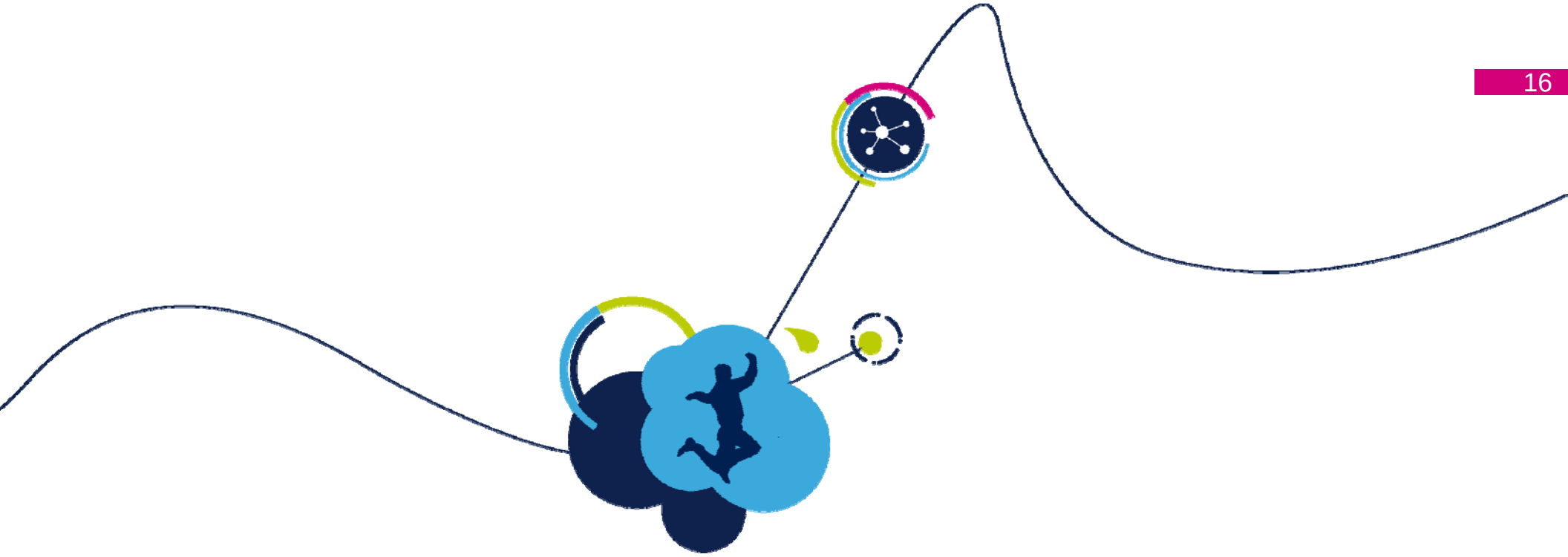
Transport Layer Security - TLS

Transport Layer Security

- Transport Security Layer - TLS (RFC 5446) protocol provides -
  - Device / Server Authentication (optional)
    - making use of asymmetrical crypto like :- RSA and ECC
  - Information Confidentiality / Data Protection
    - key exchange crypto like:- Diffie-Hellman
    - encryption using symmetric ciphers like :- AES, Triple DES
  - Message Integrity
- Makes extensive use of X.509 certificates and a Public Key Infrastructure
- Device authentication may be carried out at the Application Layer
- A IoT Device due to its limited capabilities (memory/processing) does not need to support a full crypto suite

TLS Exchange Overview



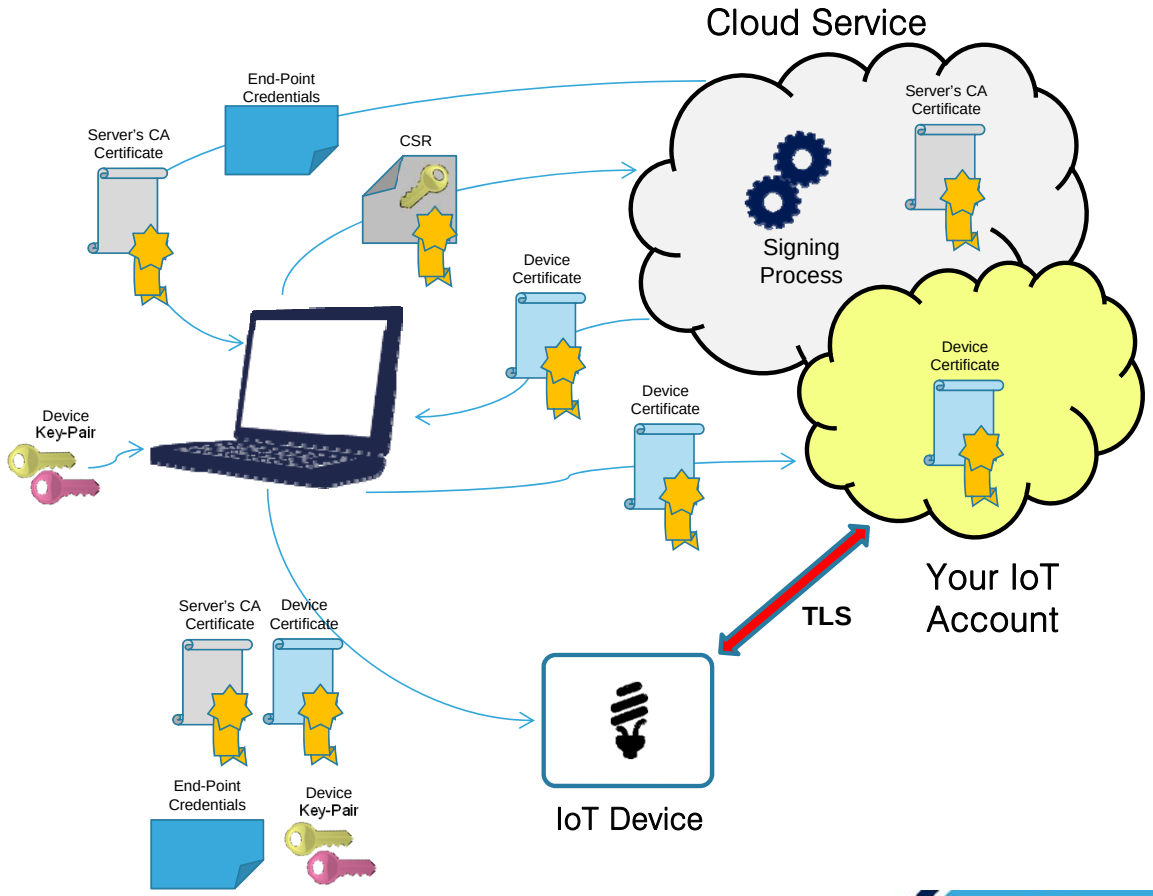


Connecting to The Cloud

Example of Registering to Cloud Service

1) Gather the End-Point Credentials and Server CA Certificate
2) Generate a Public/Private key pair
3) Create a Certificate Signing Request (CSR)
4) Request a Device Certificate
5) Register the Device Certificate into Your IoT Account
6) Program Certificates, End-Point Credentials and keys into the IoT Device

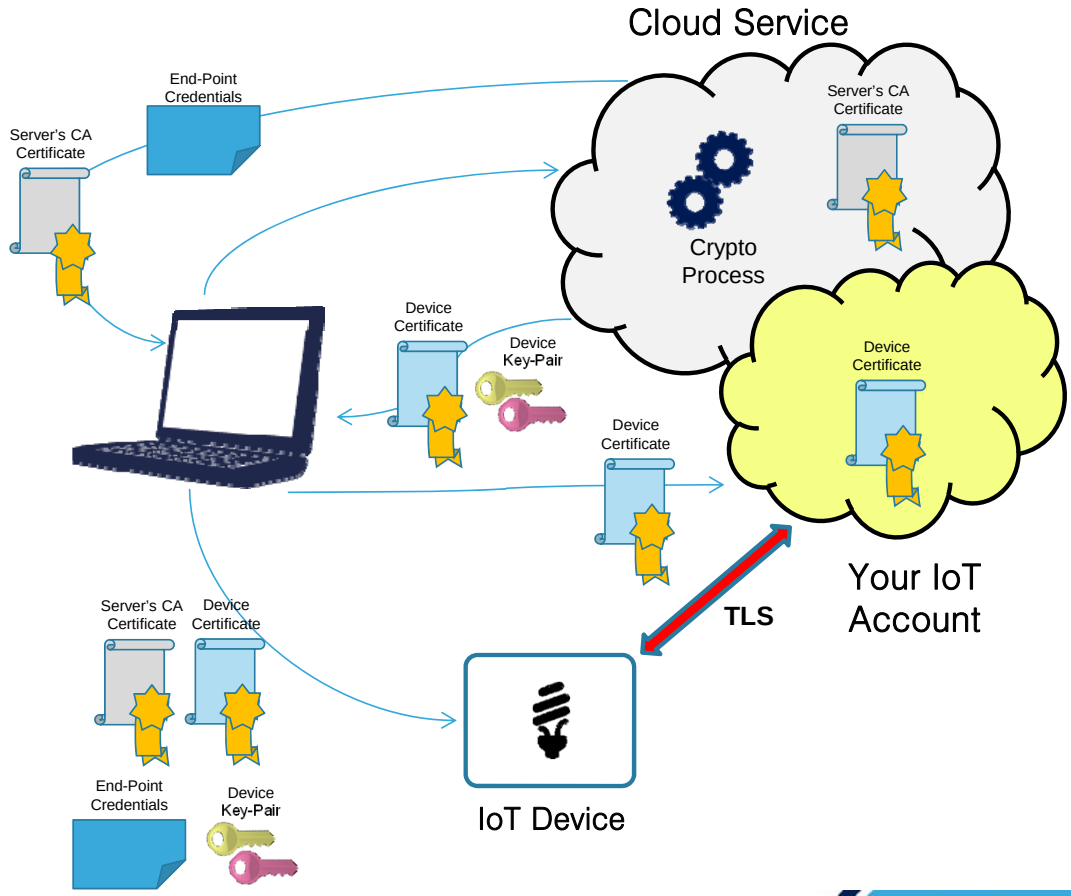
Now the IoT Device can establish a secure communications channel with the Cloud Service



Registering Demo to Cloud Service

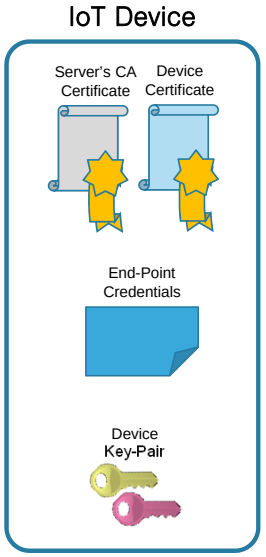
1) Gather the End-Point Credentials and Server CA Certificate from the Cloud Services Provider
2) Request a Device Certificate and Public/Private key pair
3) Register the Device Certificate into Your IoT Account with the Cloud Services Provider
4) Program Certificates, End-Point Credentials and keys into the IoT Device

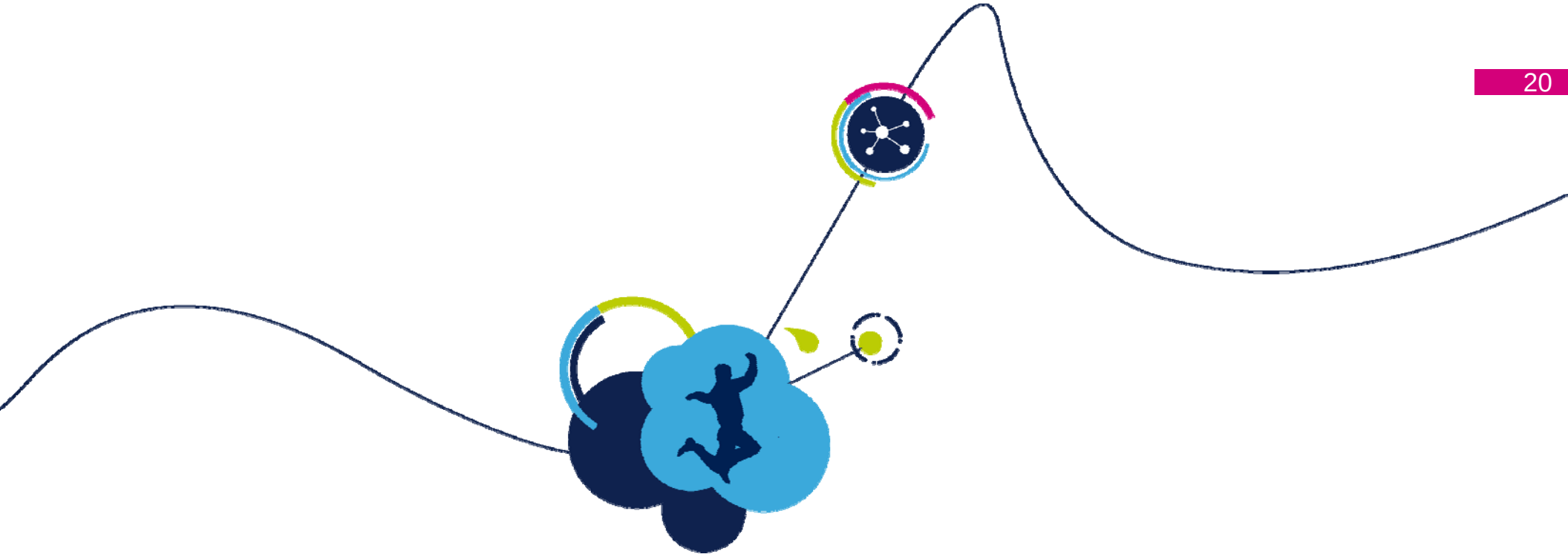
Now the IoT Device can establish a secure communicate channel with the Cloud Service



Handling TLS Credentials

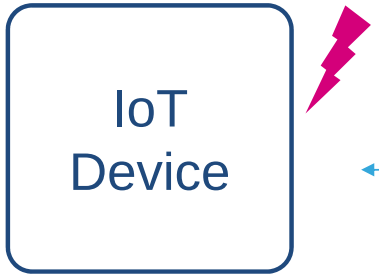
- Protect the keys (especially the Private Key)
- Store certificates and other credentials such that they cannot be easily replaced
- 95% attacks today are software

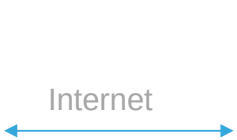




IoT Device Integrity – Building Trust

Classes of Attacks



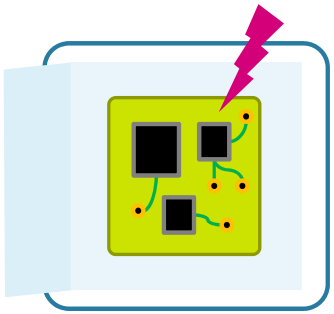


ATTACKS

Software Attack

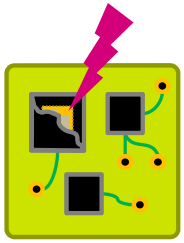
Misuse of network protocols
- Exploit communication protocol errors

Flaws in software design / implementation



Board Level Attack

With the case opened / removed
- Test / debug port access
- Inter device bus and IO probing
- Reset, clock attacks
- Power analysis
- Temperature / electrical attacks (glitch, overvoltage)



Silicon Level Attack

Device de-packaged
- Circuit analysis and probing

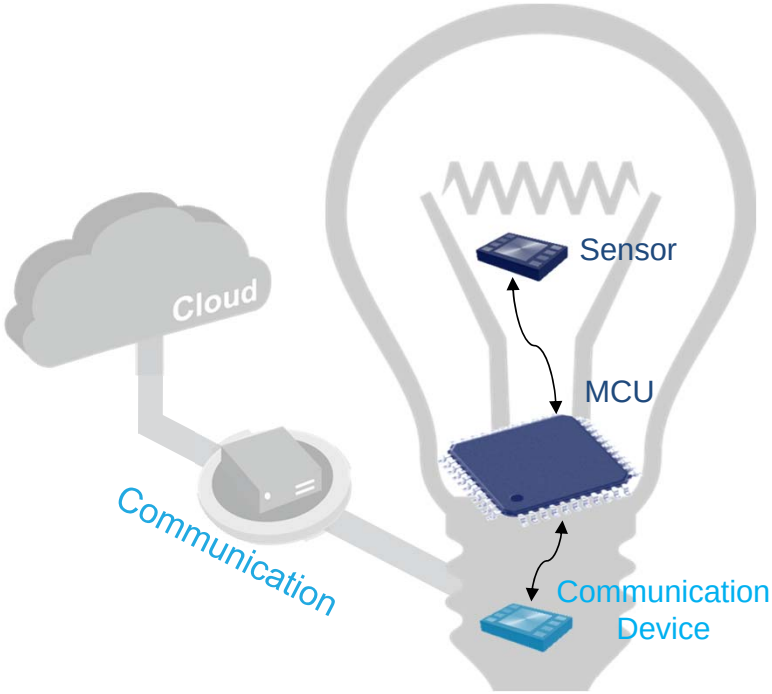
Fault injection
- Laser beam

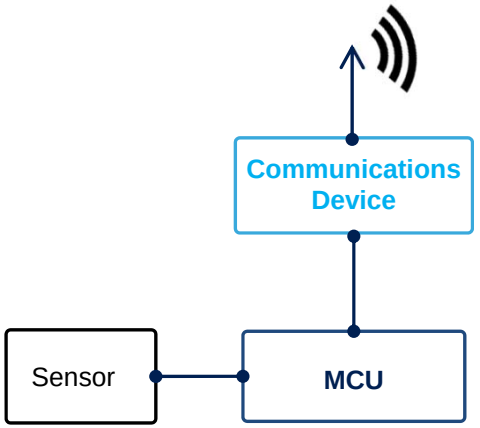
Security Needs for an IoT Device

Cloud Service providers wish to protect their services, their network and their brand

- ✓ Information/data confidentiality and integrity through the use of cryptography
- ✓ Trust-worthiness through cryptographic authentication between communicating devices
- ✓ Trust-worthiness of the platform through device integrity

Example of a Simple IoT Device

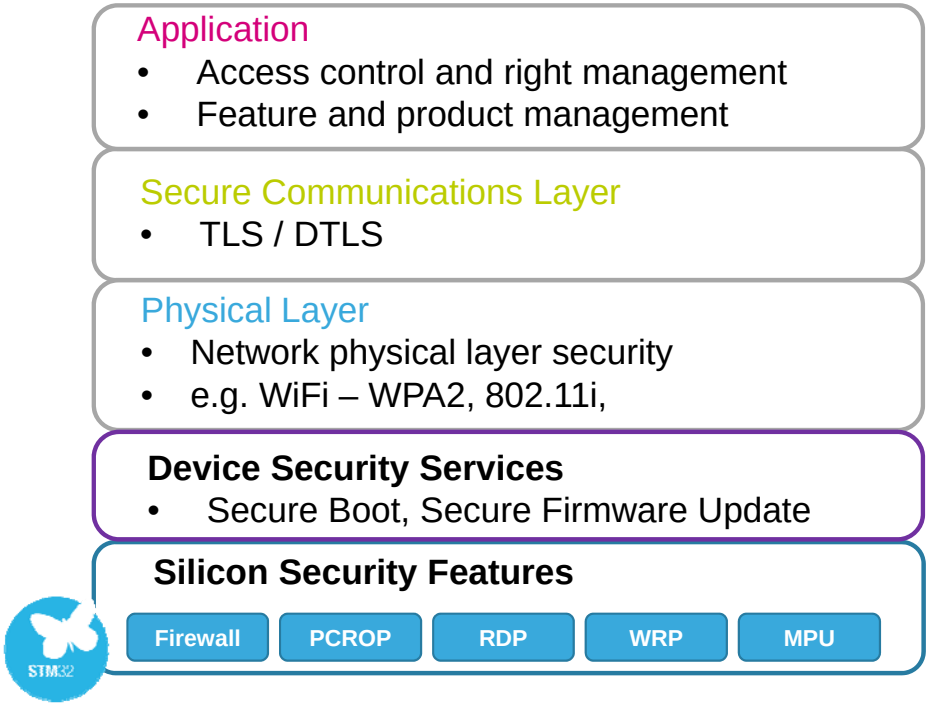




Security Layering

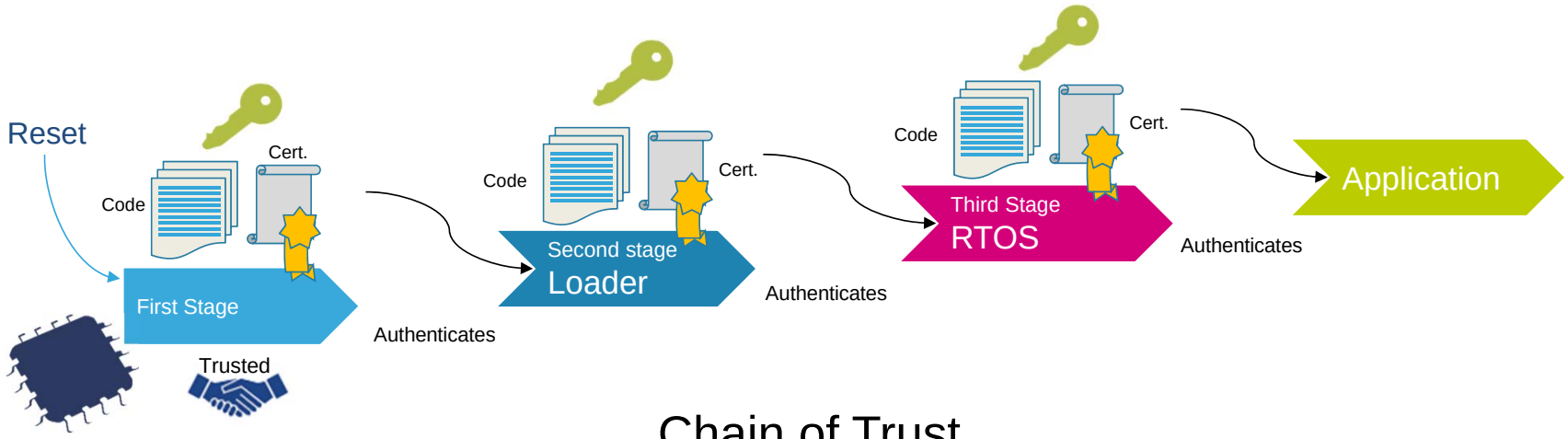
- Silicon Security Features
  - Used to establish a robust platform on which trusted processes and associated cryptography can be performed
- Secure Boot-Loader (SB)
  - Establishing a Root-of-Trust
  - Verifying firmware image on every boot

- Secure Firmware Update (SFU)
  - Building a system that can evolve to counter new threats, add new functionality, fix bugs in a controlled and secure way once device is in the field



Secure Boot

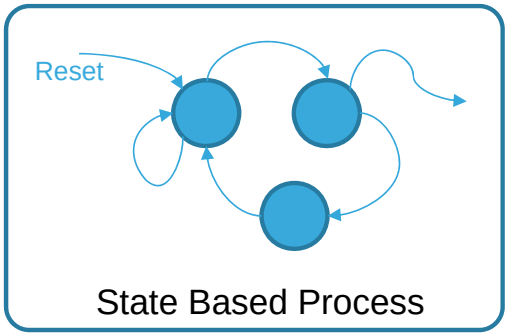
- Secure Boot uses cryptographic functions to confirm the authenticity of firmware before allowing it to run
- A multi-stage boot process consists of each stage authenticating the next

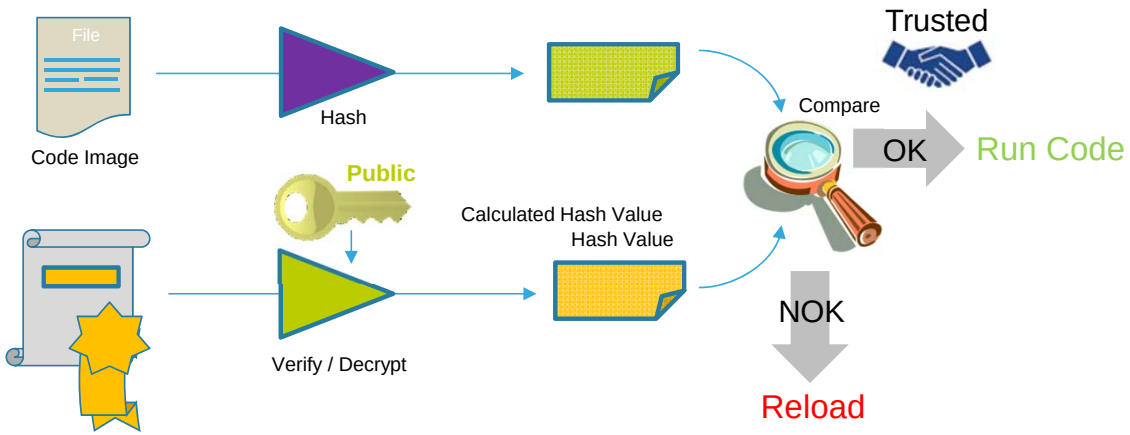


Chain of Trust

Secure Boot Process

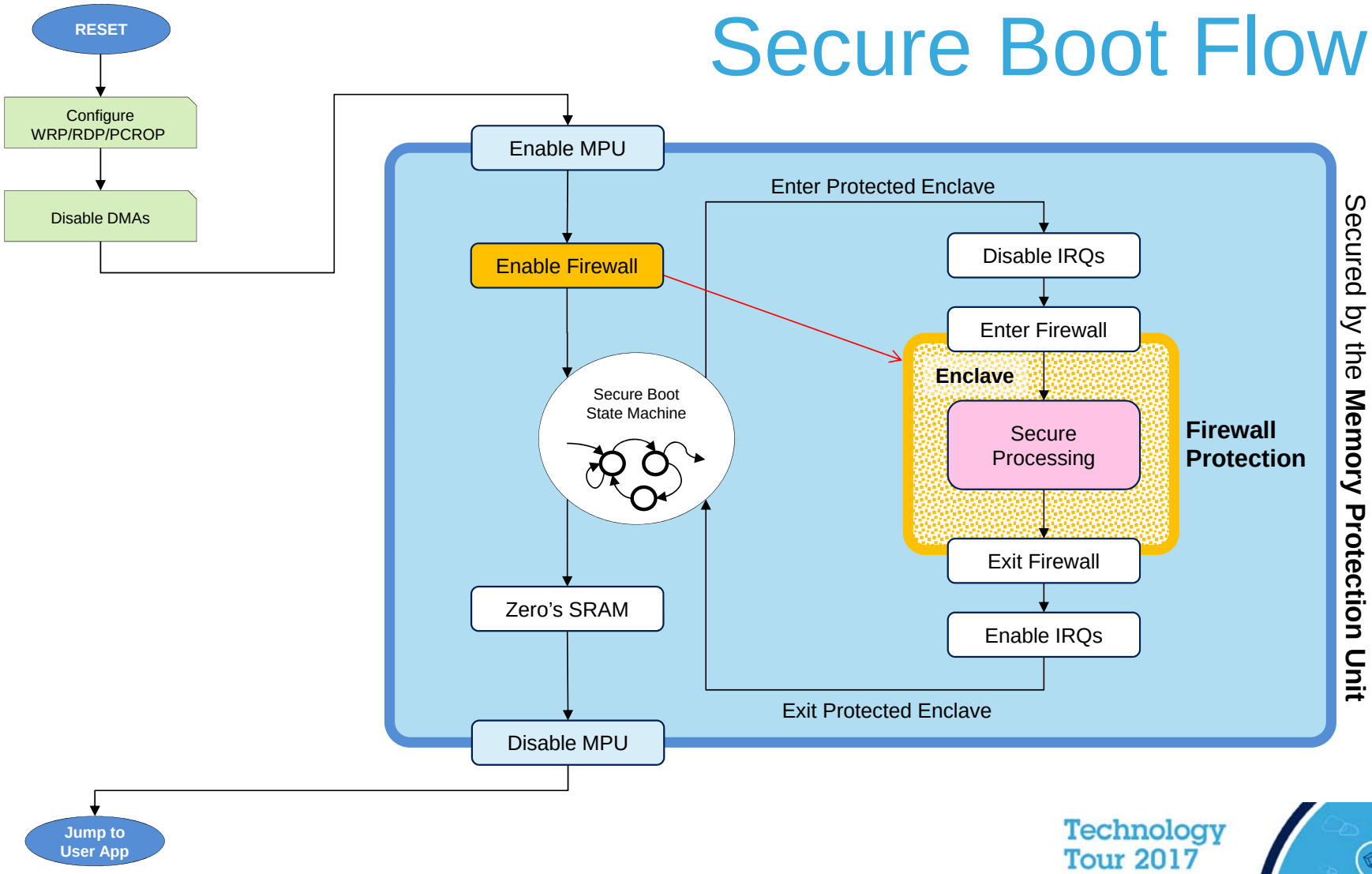
- Performed after a RESET, using a Public Key stored in the device
- It is a stateful process for predictable behavior



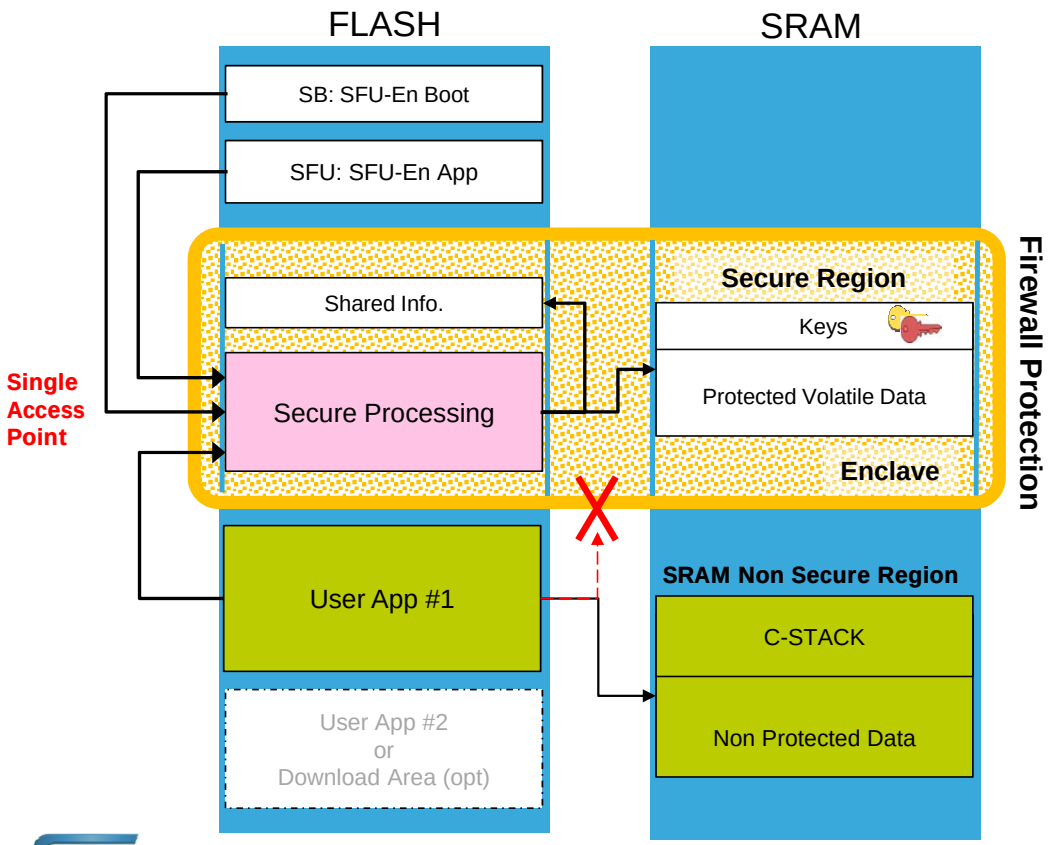


Code Authentication

Secure Boot Flow

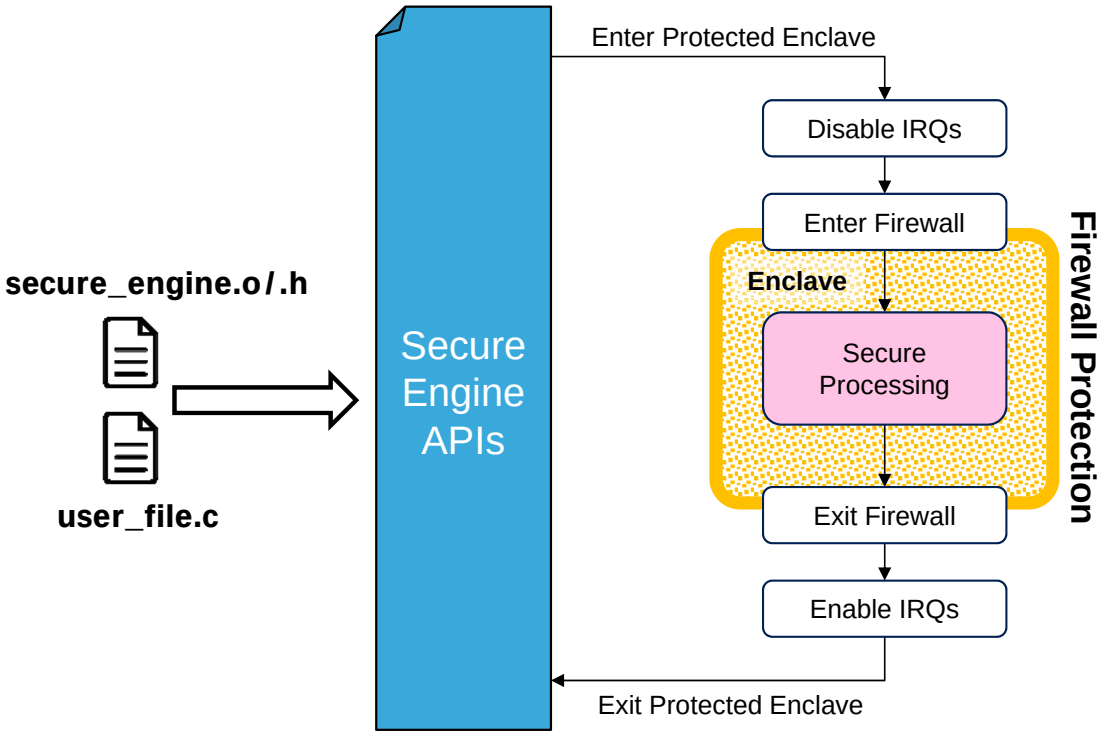


Secure Processing

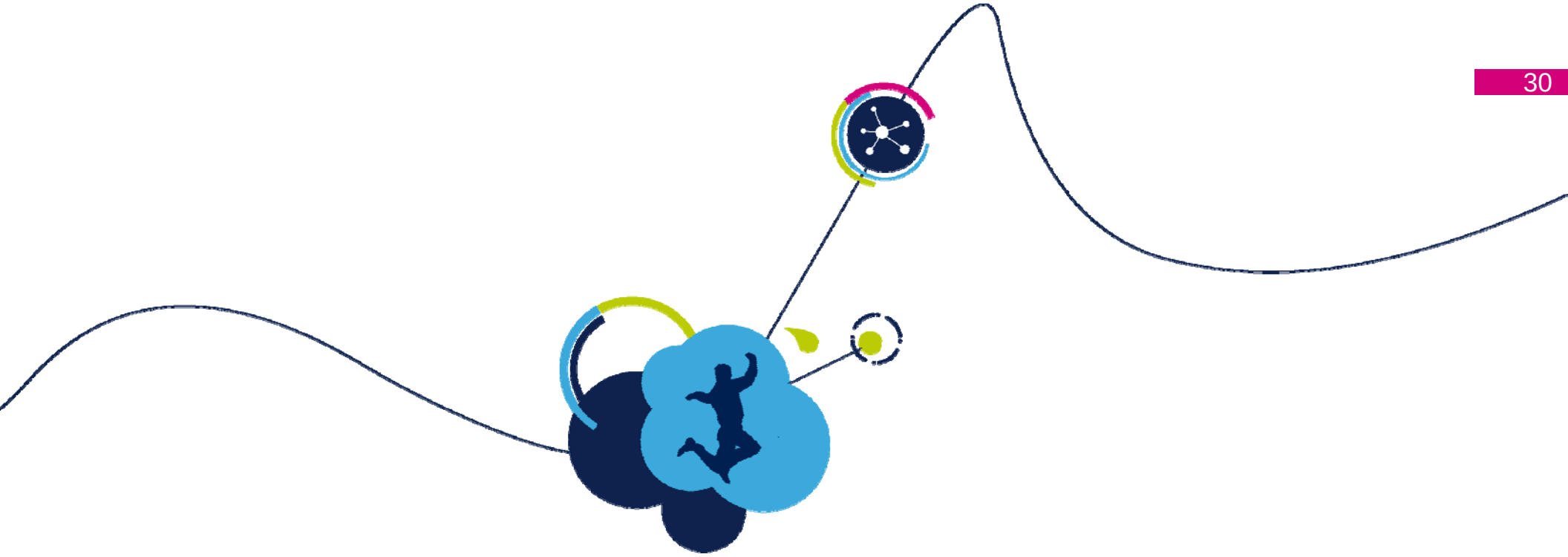


- Secure Processing Environment
  - Secure cryptographic functions
  - Protected access to ***Shared Info***. data-space
- Single access point – Call Gate
  - Code segment cannot be executed without passing through it
  - Volatile and Non-Volatile Data cannot be accessed from outside
- Environment Configured to protect -
  - Volatile data (SRAM)
  - Non-volatile data (FLASH)
  - Code - no jump calls

Secure Engine APIs

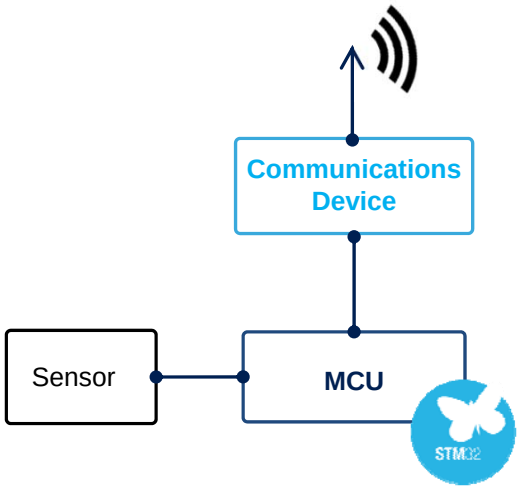


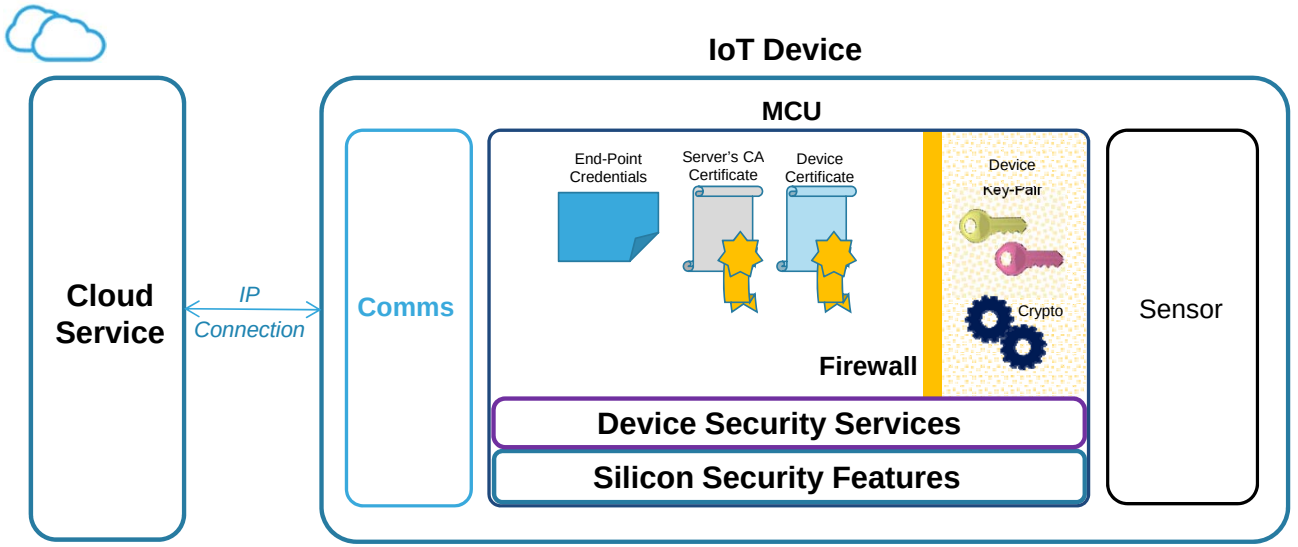
- Wrapper of internal protected functions
  - Encryption / Decryption
  - Shared Info. - Read / Write
- User-level APIs handling
  - Parameter parsing for the single Secure Engine
  - Entering and exiting of the secure environment



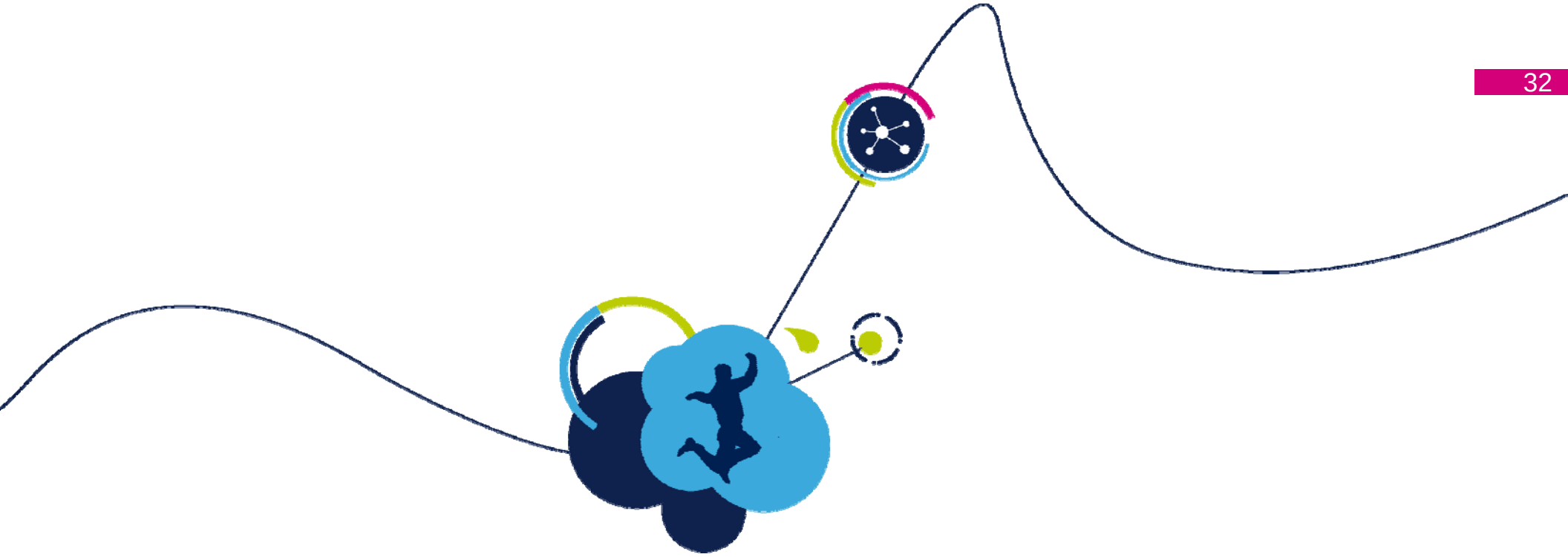
Handling TLS Assets

Handling TLS Assets



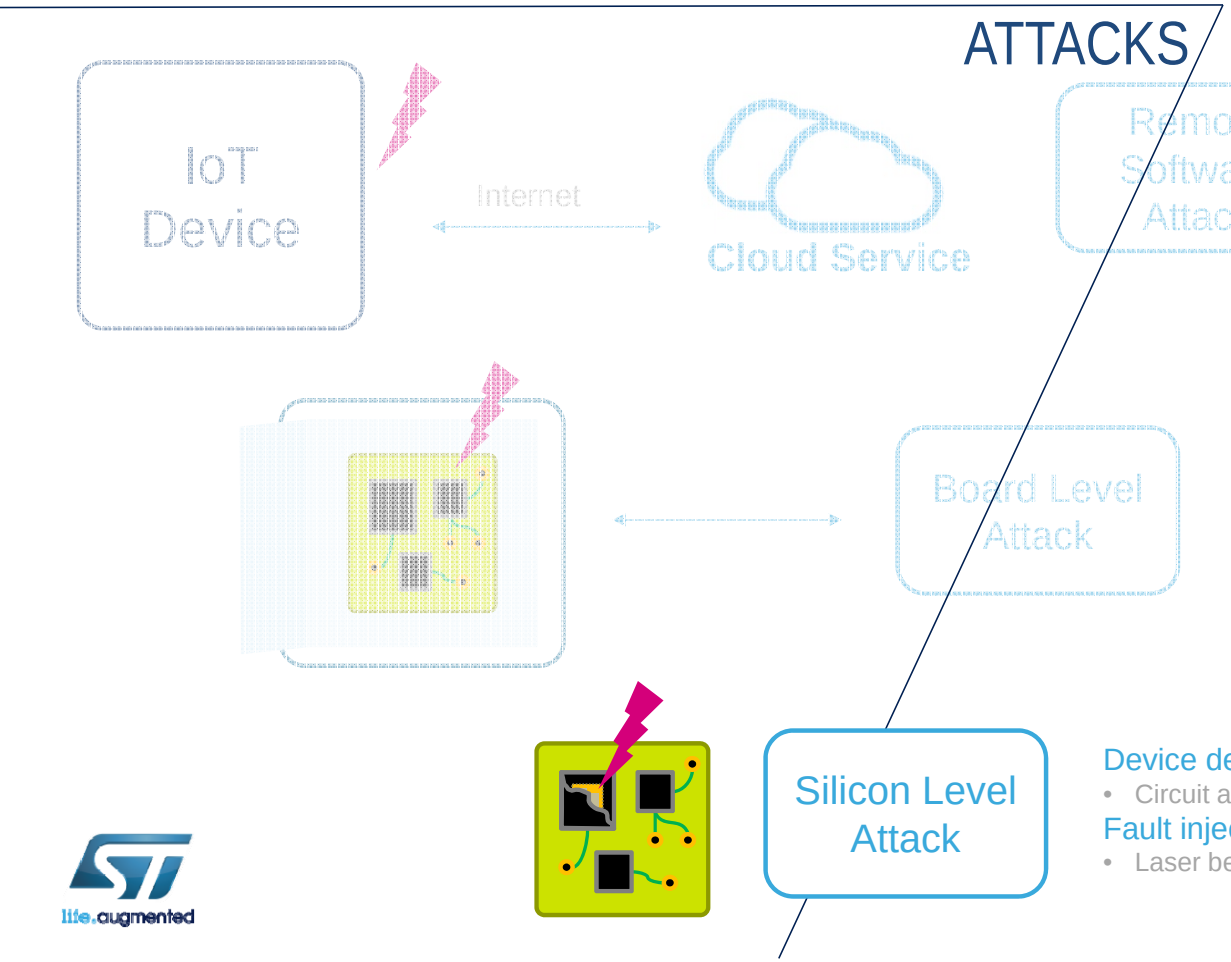


- Make use of the security mechanisms provided by the MCU to protect code, cryptographic functions and protect the keys (especially the Private Key)
- Store certificates and credentials in such that they cannot be easily replaced



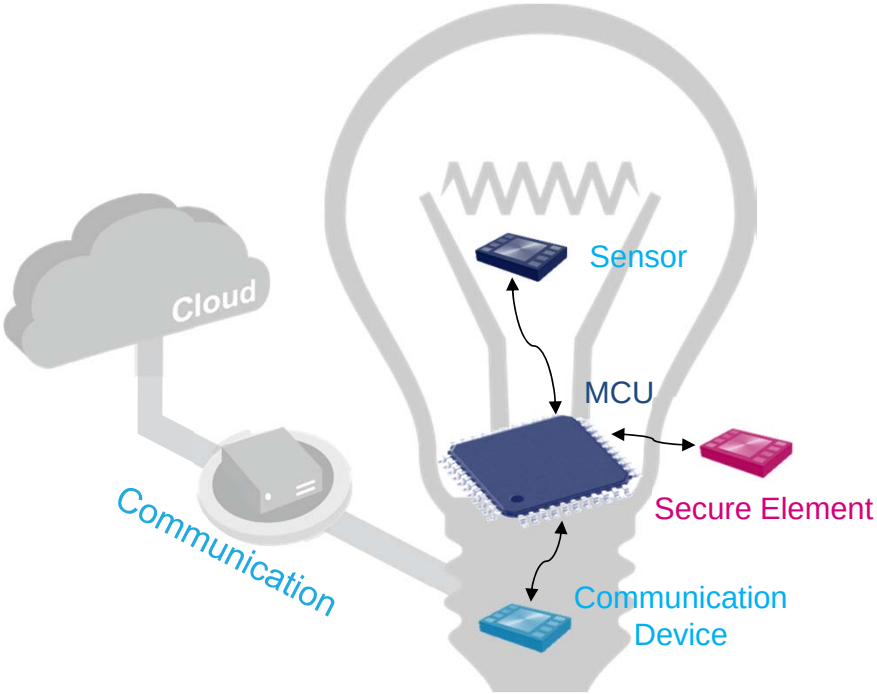
Enhanced Integrity
by adding a Secure Element

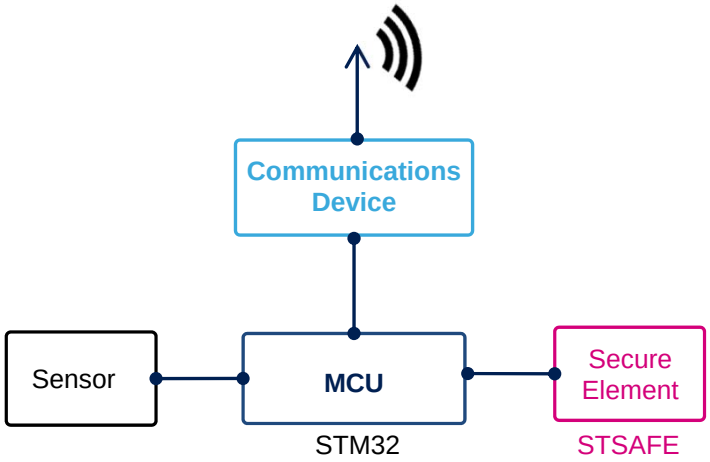
Classes of Attacks



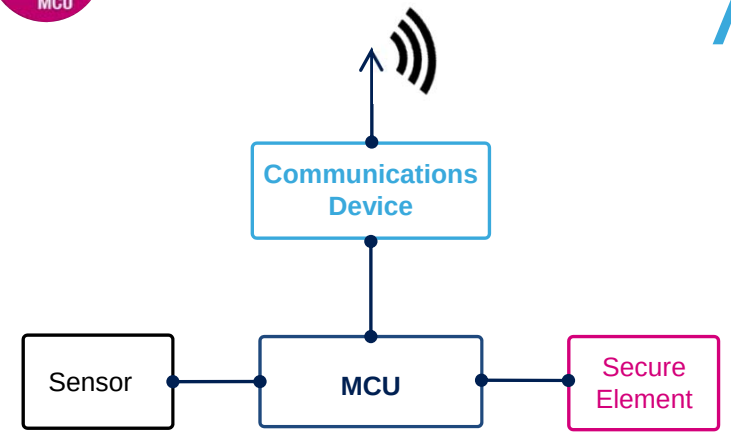
Add a Secure Element

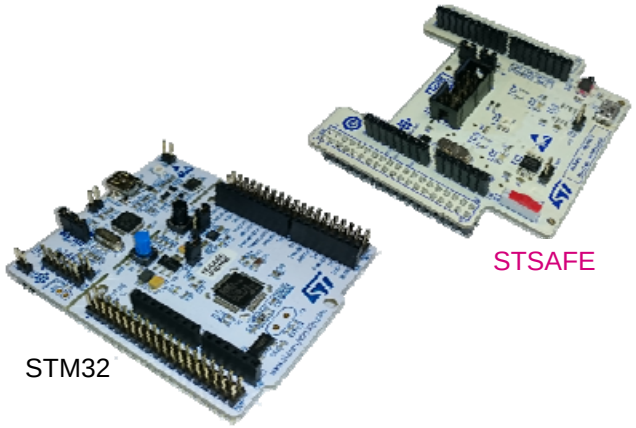
To Improve Integrity





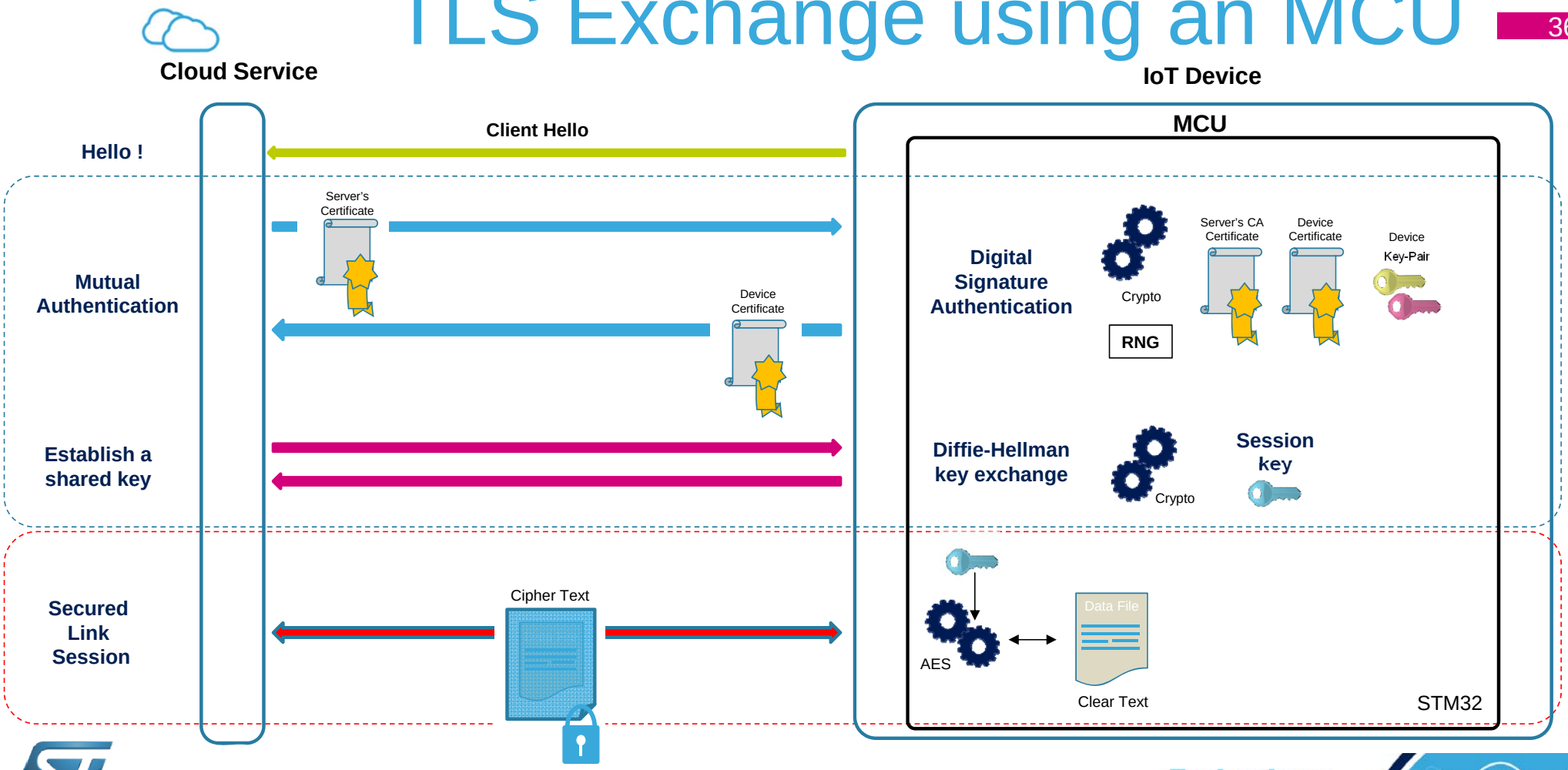
Adding a Secure Element



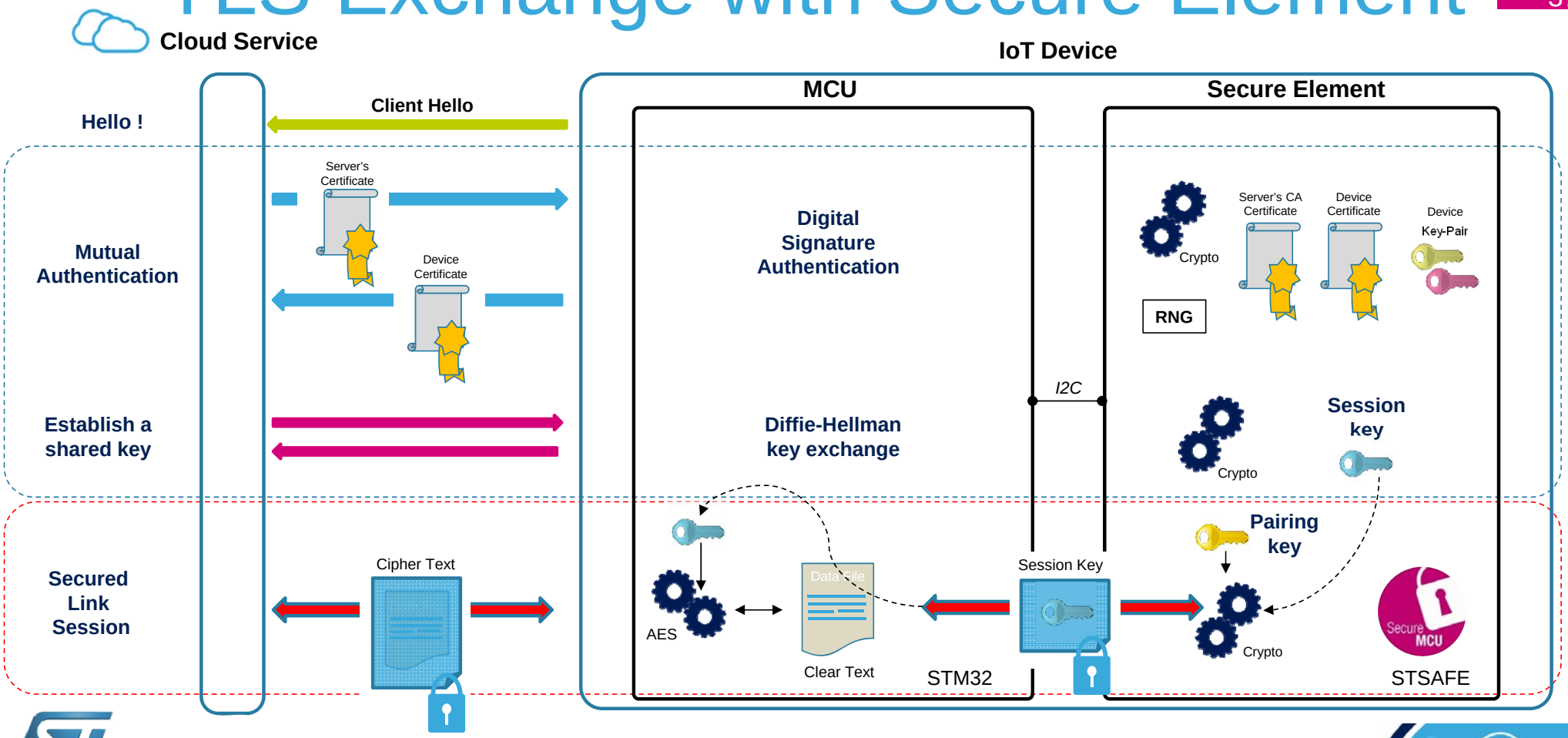


- A Secure Element is designed to thwart silicon invasive attacks
- Independently assessed, achieving very high standards like EAL5+ Common Criteria Certified
- Protects keys and performs cryptographic functions (ECC, AES)
  - For Secure Communications, Boot and Firmware Updates
- Provides a Secure Data Store
- Secure keys and certificates are provisioned during the manufacturing process

TLS Exchange using an MCU

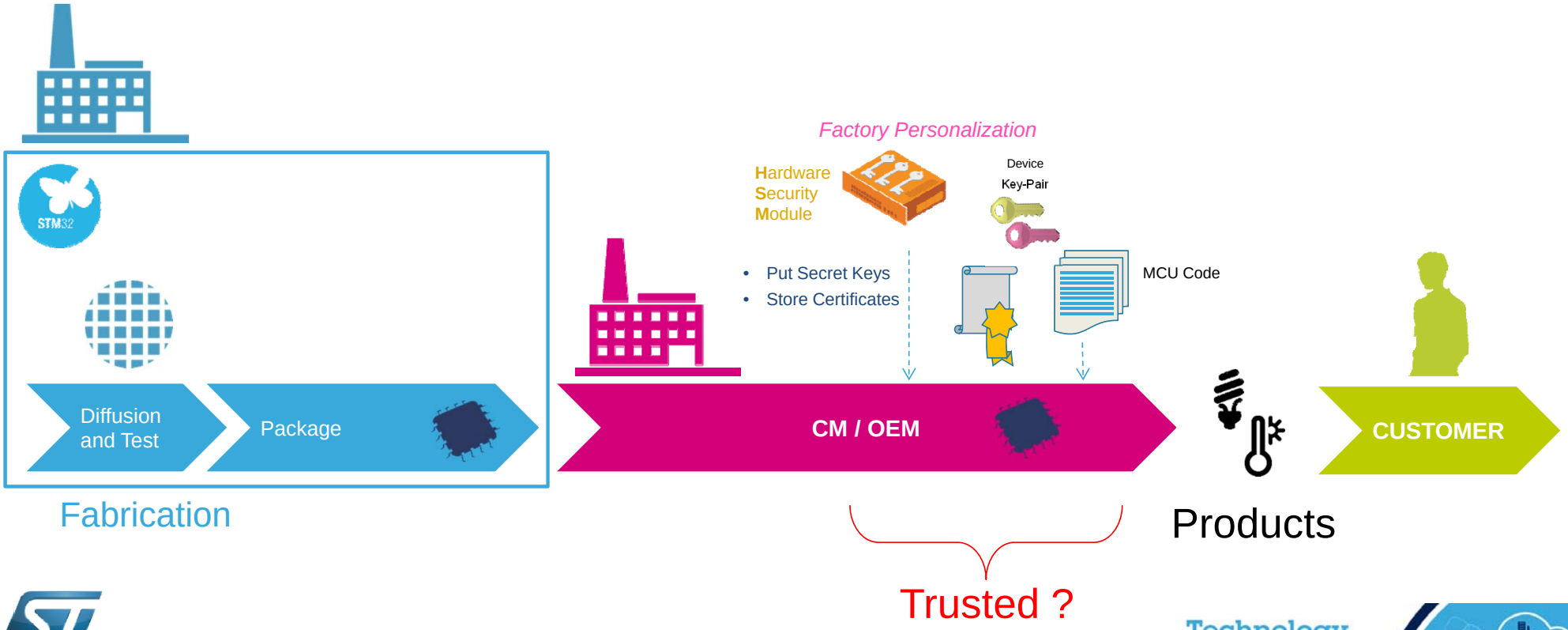


TLS Exchange with Secure Element



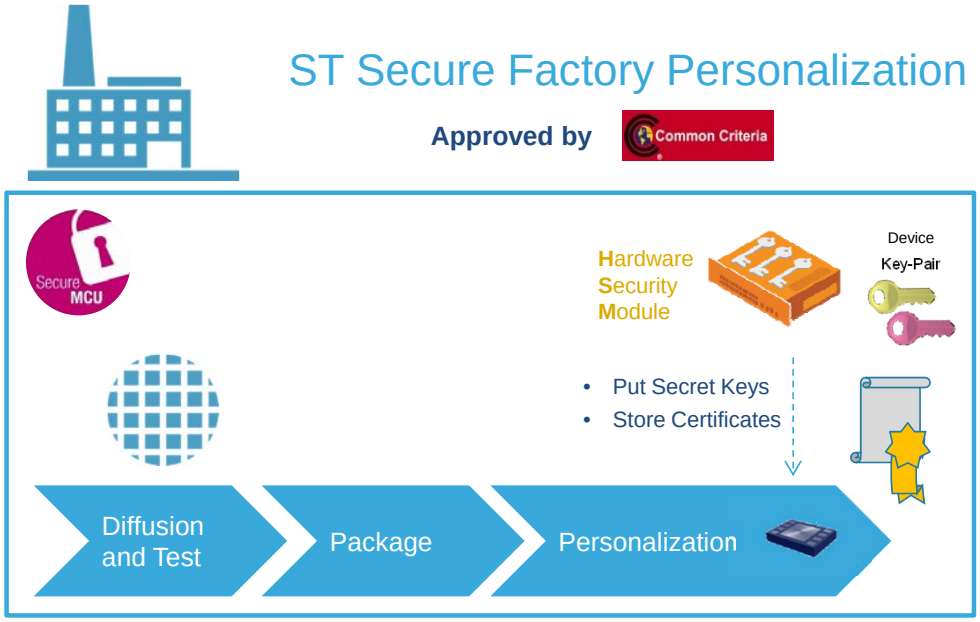
Production Flow with Personalization

Standard MCU

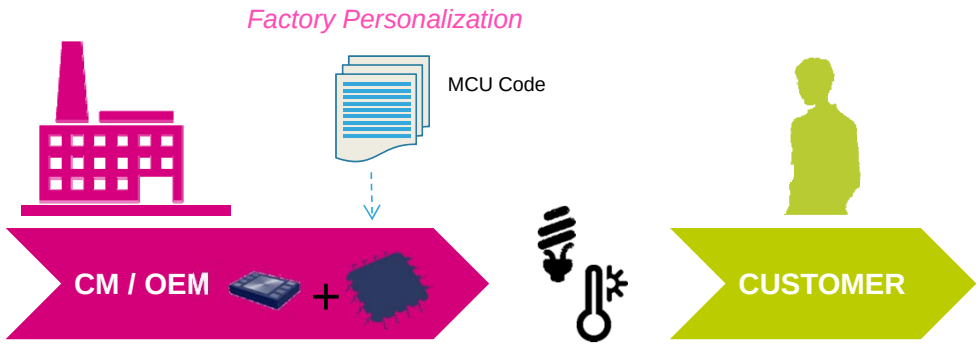


Production Flow with Personalization

Secure Element



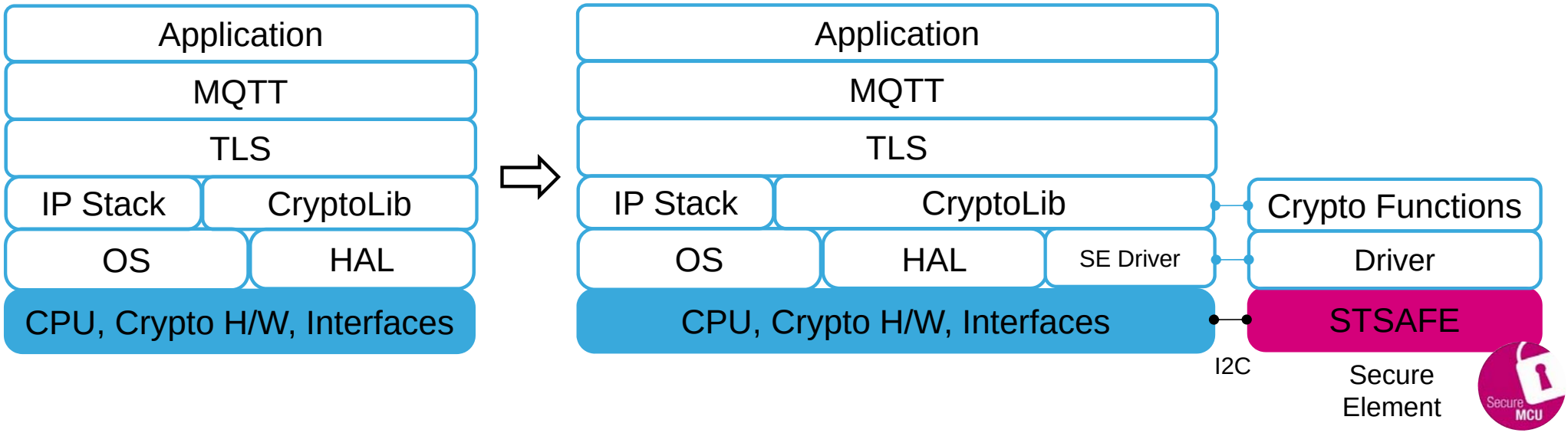
Fabrication and Personalization Phases

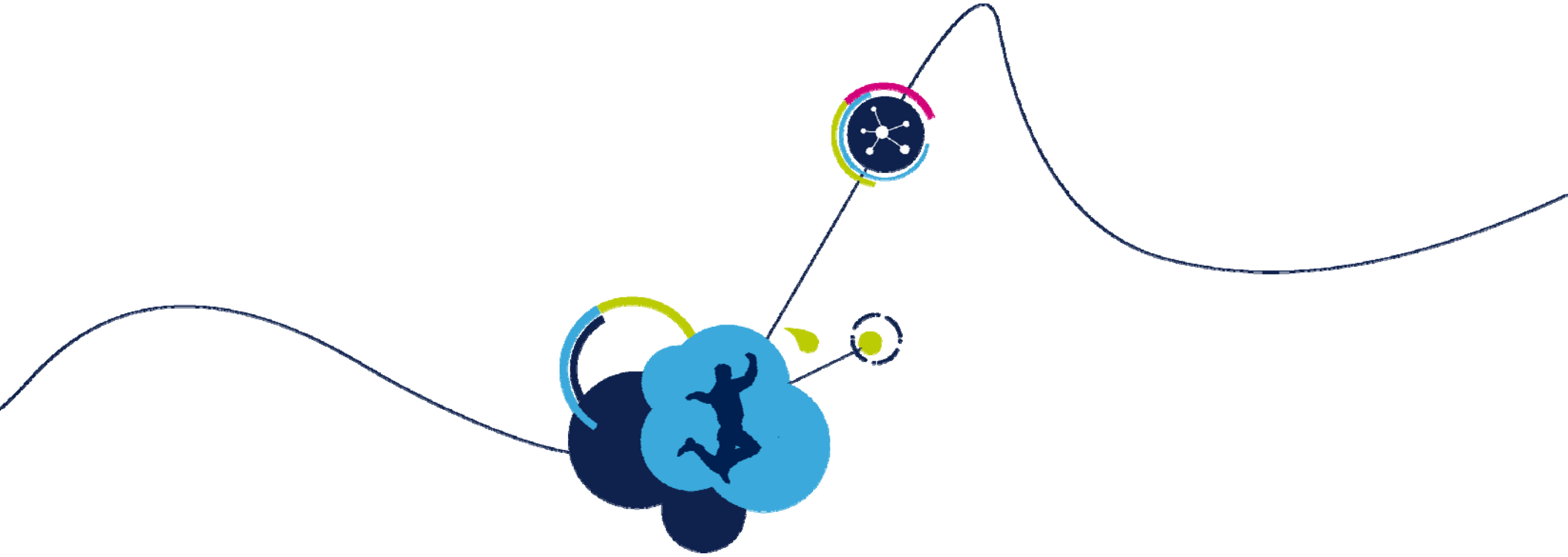


Products

Impact on Software Stack

- The Secure Elements crypto functions replace software implementations in the Crypto Library

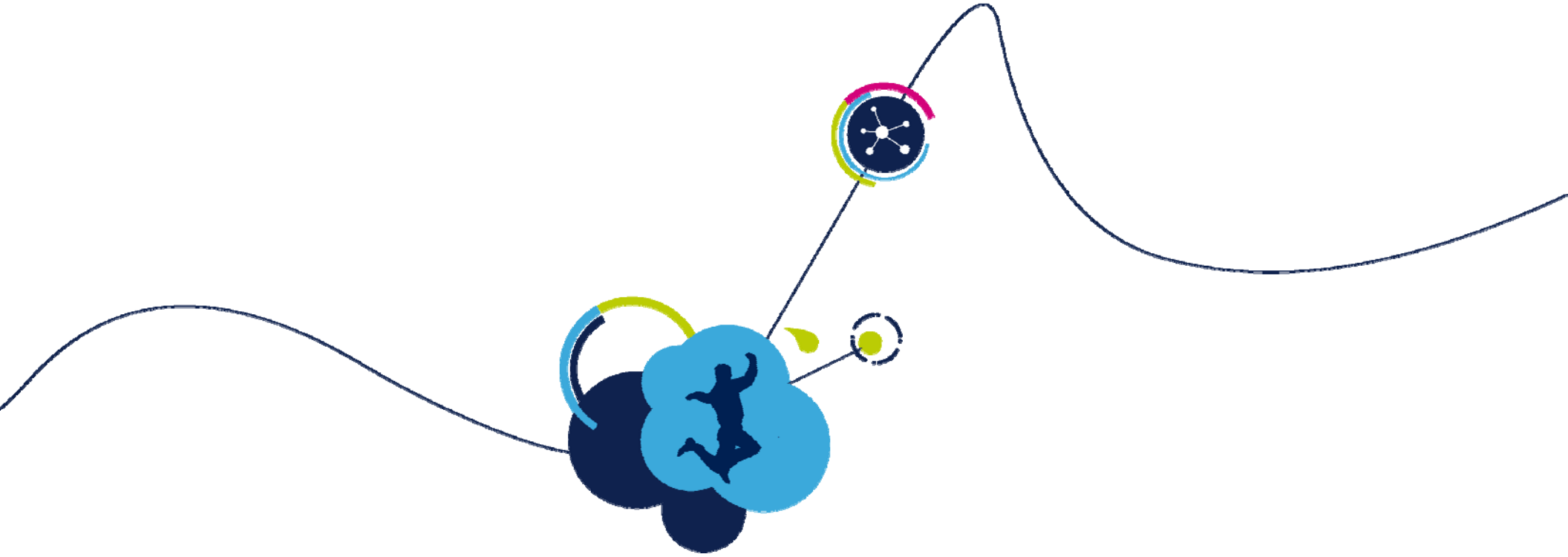




Conclusion

Conclusion

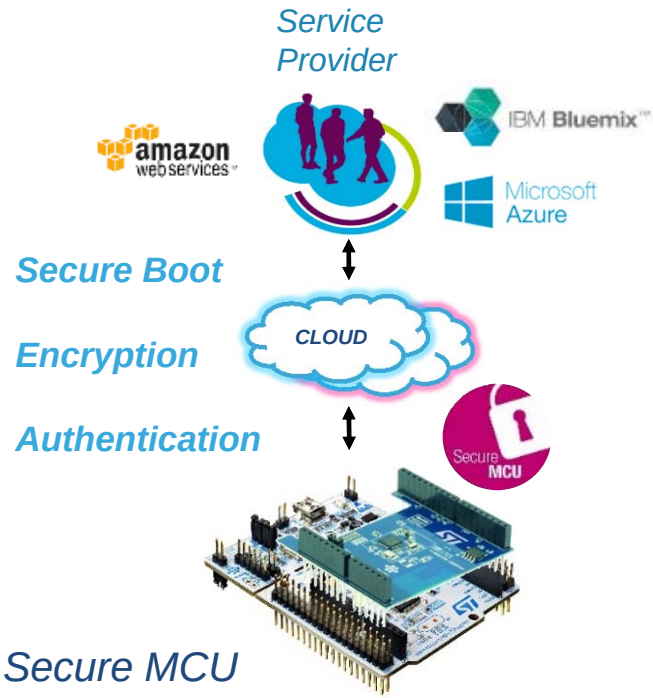
- Internet of Things presents a wealth of opportunities, a growth for commerce but an increased of risk of theft, mischief and damage or loss of life
  - Secure communications is essential for success
- Secure and trustworthy IoT devices, well fortified against threats, are essential
  - Design products resistant against threats throughout their life-cycle
  - Most of the software related attacks today can be thwarted by good firmware development practices
- Consider enhancing your IoT Devices integrity by using a Secure Element



Demos

ST Solution for Security in IoT

Solution
for IoT Device



Secure MCU

With STM32 Nucleo
and sensors Expansion Board


Q&A

life.augmented

Technology Tour 2017

Helping to Fortify Your IoT Solutions.
ST the Strong Element.

STMicroelectronics

2017

life.augmented