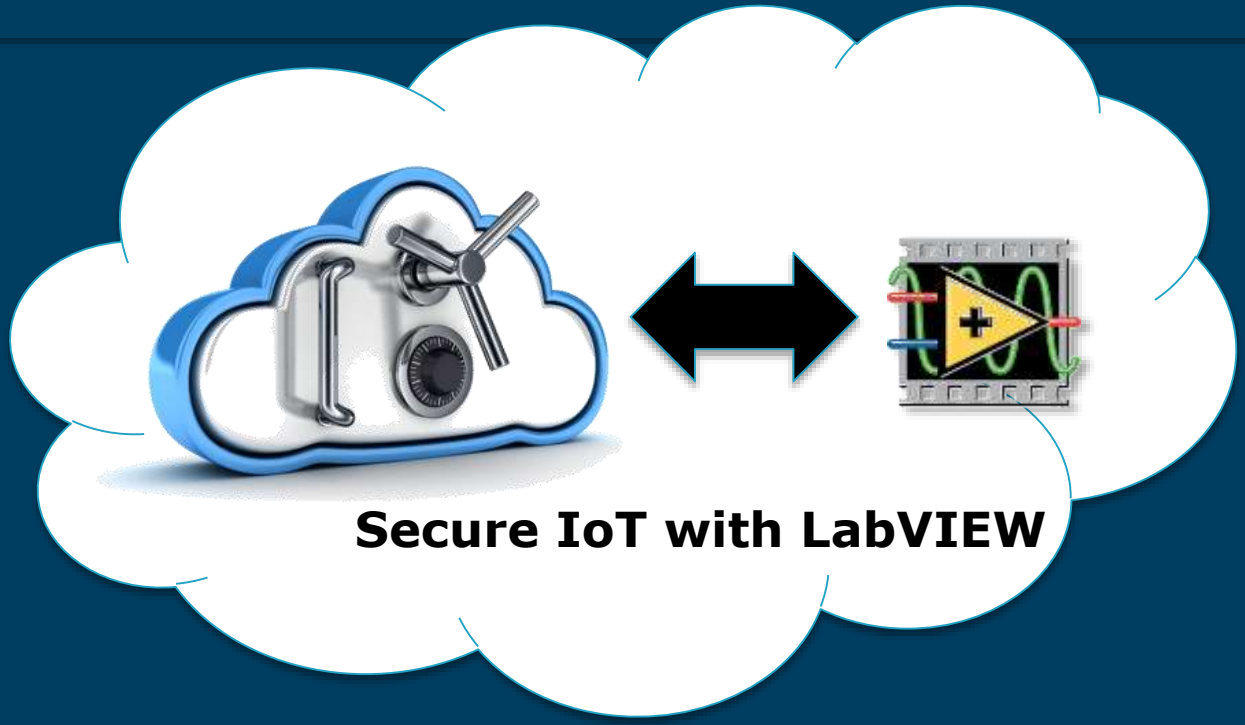


Secure IoT with LabVIEW

Jonas Mellroth, WireFlow AB

BACKGROUND
AND CHALLENGES

WireFlow experience

- Security dongles to prevent copying of LabVIEW applications
  - Using SHA 256 for message hashing
- User Access Toolkit
  - Using SHA 256 for password hashing and dongle access
  - Using AES 256 for data encryption
- WireQueue-MQTT toolkit
  - Using TLS for secure internet messaging from any LabVIEW target.
  - Using WireFlow dongles for Message integrity

Traditional data sharing

- Peer-to-peer communication
- Full control of endpoints
- Communication took place on a LAN behind firewalls, or on some other communication link (RS232?)

Not much concern for security breaches from outside of the LAN.

Internet risks

- Eavesdropping
- Stolen credentials
- Unauthorized access
- Replay of sensitive commands
- Unsupported commands inserted in the system
- ...

Terminology

- Authentication
  - Authentication verifies **who you are**.
    - login + password
    - MAC address

- Authorization
  - Authorization verifies **what you are authorized to do**.
  - Determine permission level

- Access Control
  - Limits access to resources

IoT challenges

More and more smart devices are communicating over internet.

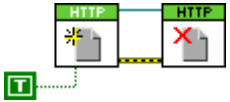
- Authentication
  - How to manage authentication when the number of devices can vary, or when new users are added on the fly?
- Authorization
  - How do we ensure that a user has elevated right regardless of which device he/she uses (e.g. security functions)
- Access control
  - How to limit or grant access to a device for new users. Maybe the device is in a remote location behind a firewall
- Communication type selection
  - Multiple devices, means multiple connections
  - Must be able to pass through gateways and firewalls
- Data protection
  - How do we hide the data from eavesdropping?
  - How do we know who we are communicating with?

LabVIEW challenges

- Multiple OS's might need to be supported
  - Windows, MacOSX, Linux
  - VxWorks, Pharlap, RT-Linux
- Deploying to different targets
  - Desktops, Laptops
  - PXI controllers
  - cRIO, SOM

LabVIEW challenges

- No direct access to low level sockets (raw) → hard to implement encrypted sockets (like TLS)
  - Https functionality exists, but http adds a lot of overhead
- Different platforms/targets expose different ways to access additional resources (like raw sockets)



- No built in encryption algorithms in LabVIEW
- No built in hashing algorithms in LabVIEW (except file MD5 ?)

PROTECTING THE SYSTEM

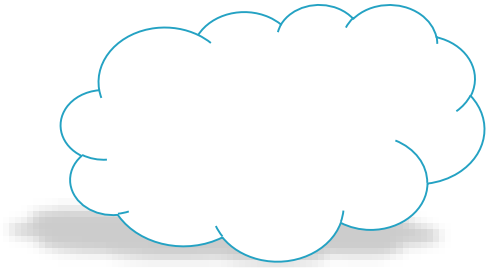
System requirements

- It must be easy to manage access control so that new devices can be added without much work
- All devices should authenticate against a central database
- No devices should communicate sensitive data without encryption
- Internet connections can be limited, so “sticky” messages should be supported, e.g. last known good state

Cloud based server - Message Broker

- Publish – subscribe messaging
- Central communication point
  - All devices can be configured with the same web address
  - All devices authenticate against the same database
- Broker handles all access control, e.g. based on clientID and/or user name
  - Not necessary to forward all messages to all devices, some devices are less secure

The message broker



Pros

- Each device only has to know about the address to the cloud based server
- No knowledge about other devices in the system
- Access controlled at a central point, e.g. configured through a web-interface
  - Banning a device is instant
- Firewall friendly
  - Less firewall configuration to allow incoming connections since all connections are initiated by the devices

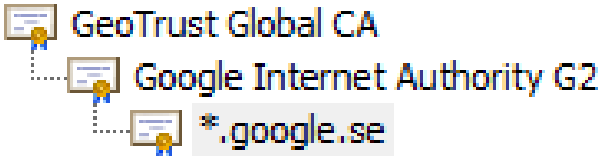
Cons

- Lower performance
- Requires extra service or server to manage the devices
- All communication go through one single server: if this is compromised all devices are open to attacks.

AUTHENTICATION
AUTHORIZATION
ACCESS CONTROL

Authentication

"who you are"



- Transport layer:
  - The server must be authenticated using a valid server certificate (not expired).
  - Each device must be able to verify the server certificate against locally stored trusted certificates, all the way back to the CA (including checking the server name against the certificate)
- Application layer
  - Each device identifies itself by an ID, an user id and a security token (password).
  - Server should check id + user + passwd against the central database

Authorization

”what you are authorized to do”

- The server Access Control List authorizes a device in terms of what messages the device can publish and/or subscribe to.

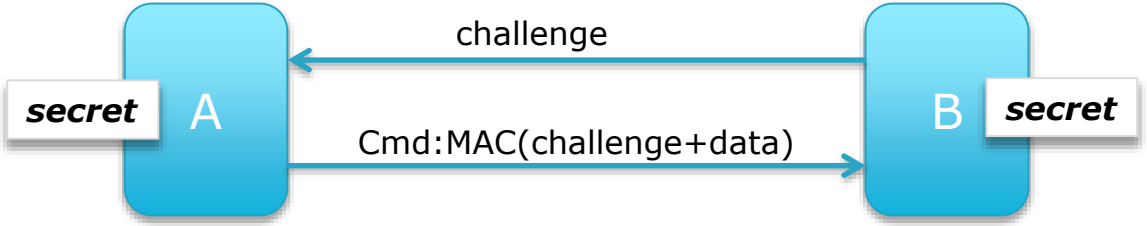
- The integrity of individual messages, e.g. security control, can be achieved by attaching an externally created MAC (Message Authentication Code). Message is only accepted if the MAC is correct.

Authorization

Message Integrity - Data Validation

Validate individual messages in the application layer by hash or Message Authentication Code (MAC)

- Sensitive messages can be validated by calculating a HASH code for the message. This can be used to verify that the message has not been tampered with.
- Message Authentication Code, MAC, is essentially a HASH but also includes an external secret, this means that a calculated MAC for a message both validates the message and authenticates the sender of the message.



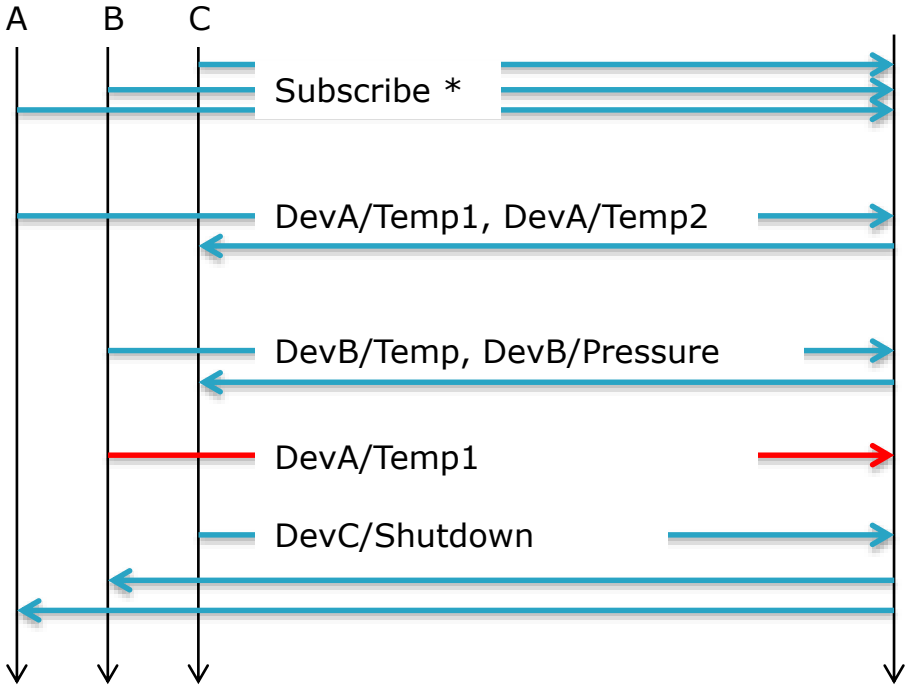
The application that can accept sensitive commands, sends out a challenge. The client side calculates a MAC based on the command content and the challenge.

Challenge changes between each command to prevent replay attacks.

Access Control

- Based on the credentials (DeviceID and User name) and an Access Control List (ACL), the cloud server are ready to accept and/or forward messages from the device.
  - No messages are forwarded until the device explicitly starts a subscription
- For security reasons the ACL should be setup so that new devices are not allowed to read or write any messages.

Access Control List (ACL) example



- All devices subscribe to everything (*)
- ACL defines
  - Only DevX can only post messages DevX/*
  - DevA and DevB only sees DevC messages
  - DevC sees all messages
- DevC will see any new device

ENCRYPTION

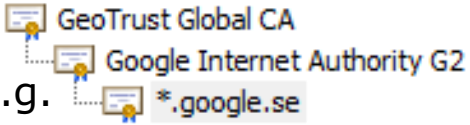
NOT OBFUSCATION

Encryption

- Transport layer
  - “Cloud based” means constant internet access, and this means we should base our transport encryption on a standard like TLS.
  - Always use an existing implementation, e.g. OpenSSL
  - Should stop unauthorized access to your system
- Application layer
  - Encrypting the actual messages prevents someone from seeing the data, but is more open to Man in the middle attacks
  - If we add Application layer encryption together with transport layer we get better protection at the cost of CPU and convenience
  - Helps authorizing messages between endpoint devices

Encryption - TLS protection

- TLS = Transport Layer Security
- Protects the transport layer with highest encryption supported by both server and device
- Certificates used for secure identification.
  - Certificate should be issued to the requested web-address, e.g. *.google.se
  - Each certificate is validated in a “chain of trust” that should end in a trusted Root Certificate Authority
- Data is expanded into packets
  - Bigger overhead for small packages, implementations use buffering to combine packages in order to reduce the overhead.
  - Expansion is part of the security, i.e. no information should be leaked by looking at the size of the encrypted data.

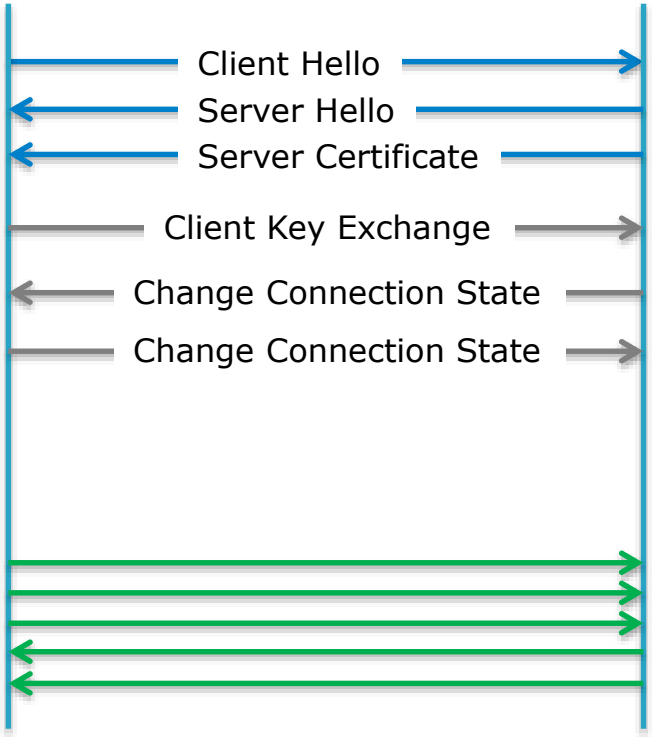


TLS - Certificate exchange

Client

Server

Client and Server negotiates a cipher to use for the Key-exchange



Server and Client exchange data using negotiated public key cipher (e.g. RSA)

Client verifies server certificate and exits if not trusted. If certificate is good, session continues with application data

If Client accepts the certificate, the rest of the communication is done with a cipher determined in the handshake (e.g AES256)

Encryption in LabVIEW

- TLS support not possible with native LabVIEW
- Encryption of application layer depends on how sensitive messages we have
- Listening on messages is maybe not a serious threat, but stealing credentials and replaying sensitive commands can be.
- When using AES or any other public encryption method, make sure to validate the encryption method against known test vectors.
- **Don’t develop your own encryption**

A number of encryption algorithms (e.g. AES) for LabVIEW is available at lavag.org
https://lavag.org/topic/14889-cr-encryption/

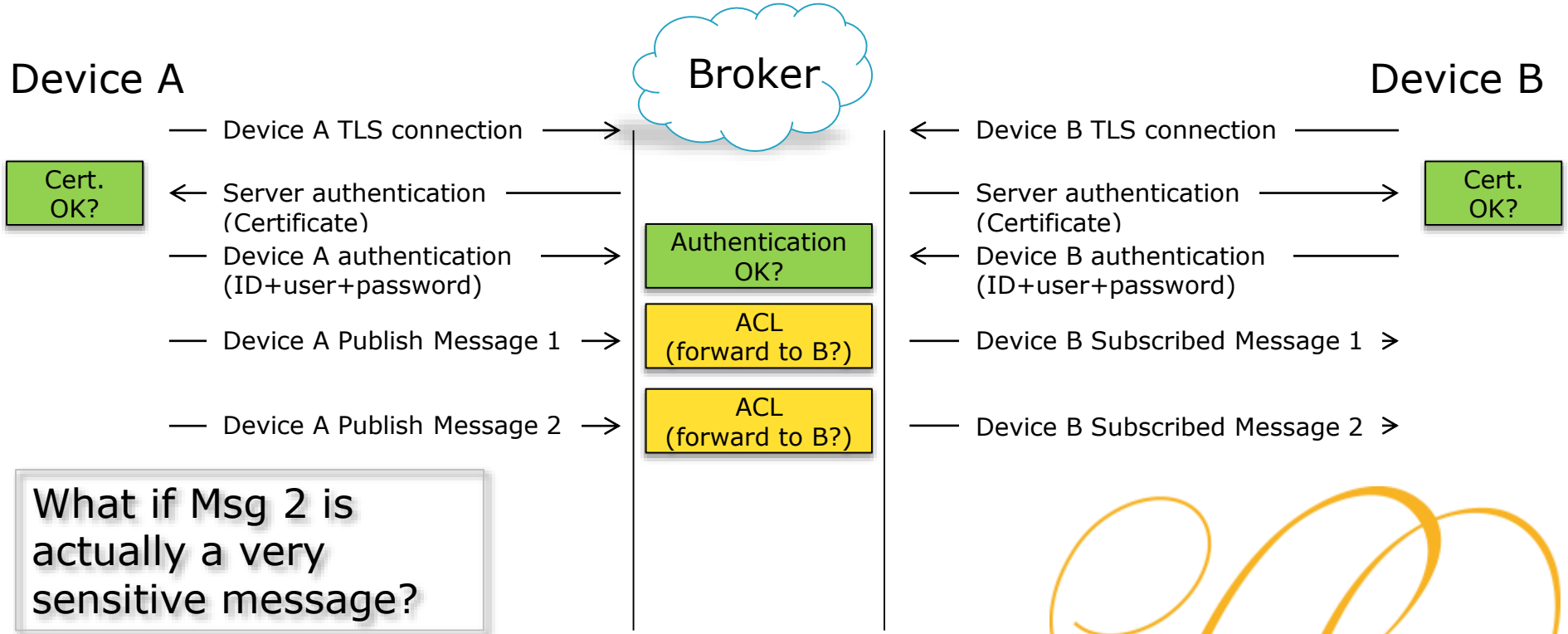
OpenSSL

- OpenSSL is free and can be compiled for just about any target, including Pharlap, VxWorks, Linux and so on.
- OpenSSL can handle certificates, securing sockets as well as encrypt or decrypt in a number of ciphers
  - Blowfish
  - DES
  - AES
  - ...

OpenSSL
Cryptography and SSL/TLS Toolkit

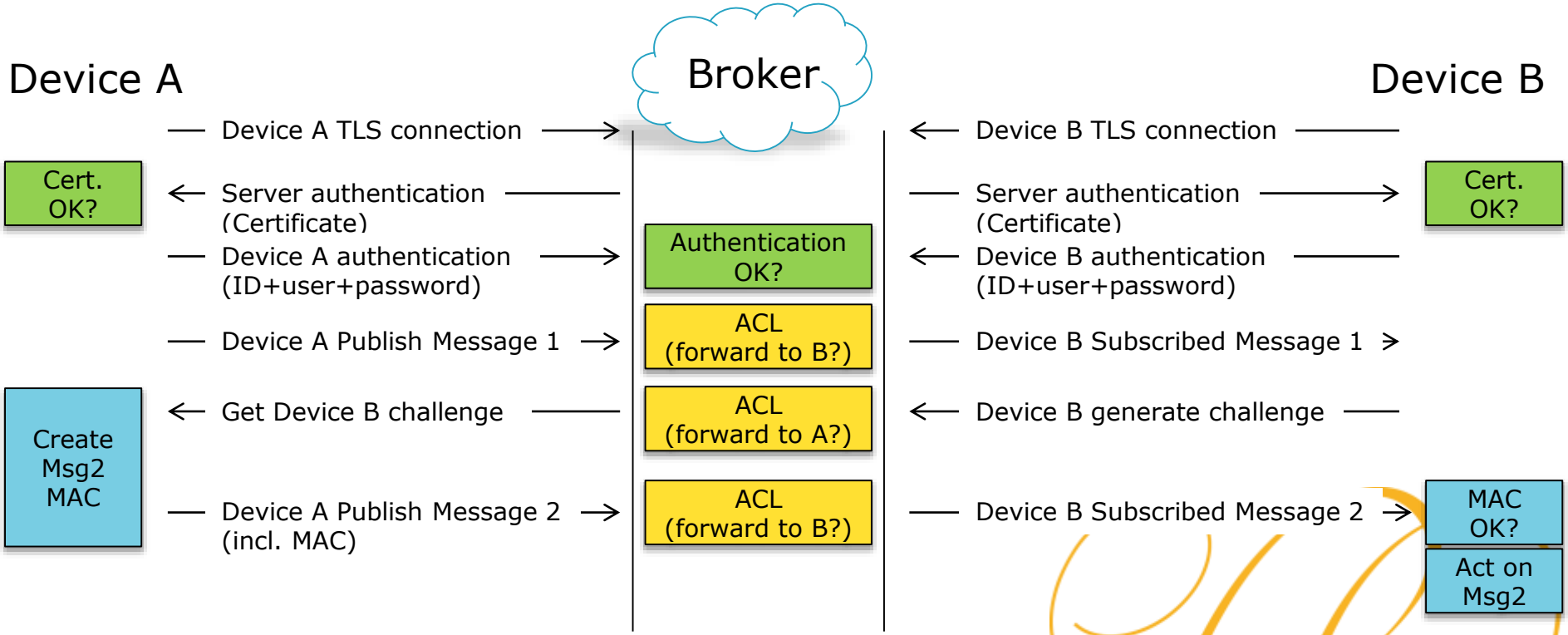
EXAMPLE

IoT Broker example (wo. MAC)



What if Msg 2 is actually a very sensitive message?

IoT Broker example (w. MAC)



WHAT IF?

TLS is not an option?

- VPN = Virtual Private Network, basically creates a secure tunnel for all traffic between two endpoints.
  - Data transfer is encrypted, e.g. using TLS
- The use of VPN can be added through usage of an external device, e.g. routers.
  - This means that unsecured devices can get good security without modification in code

- Warnings
  - A TLS/TCP session can be initialized from the device side only, but an open VPN tunnel allows the server to try to initiate new session to FTP, SSH etc.
  - If a VPN connection is compromised all local network resources on the tunnel are open, compared to a single TLS/TCP session where only the actual connection is compromised (e.g. FTP ports on a cRIO).
  - Depending on the configuration, VPN can require more Firewall configuration (more ports that needs to be opened bidirectional etc.).
  - Minimize other network traffic through the VPN tunnel, especially over cellular networks.

A Self signed certificate is used?

- A self signed certificate offers the same level of encryption as a signed one
- Each client must have an entry in the list of trusted certificates for this certificate
- Self Signed Certificates can be generated with OpenSSL libraries

- Warnings
  - With self signed certificates there is no guarantee that the website in the certificate actually owns the web domain in the certificate. A publically signed certificate can only be issued to the owner of a domain.

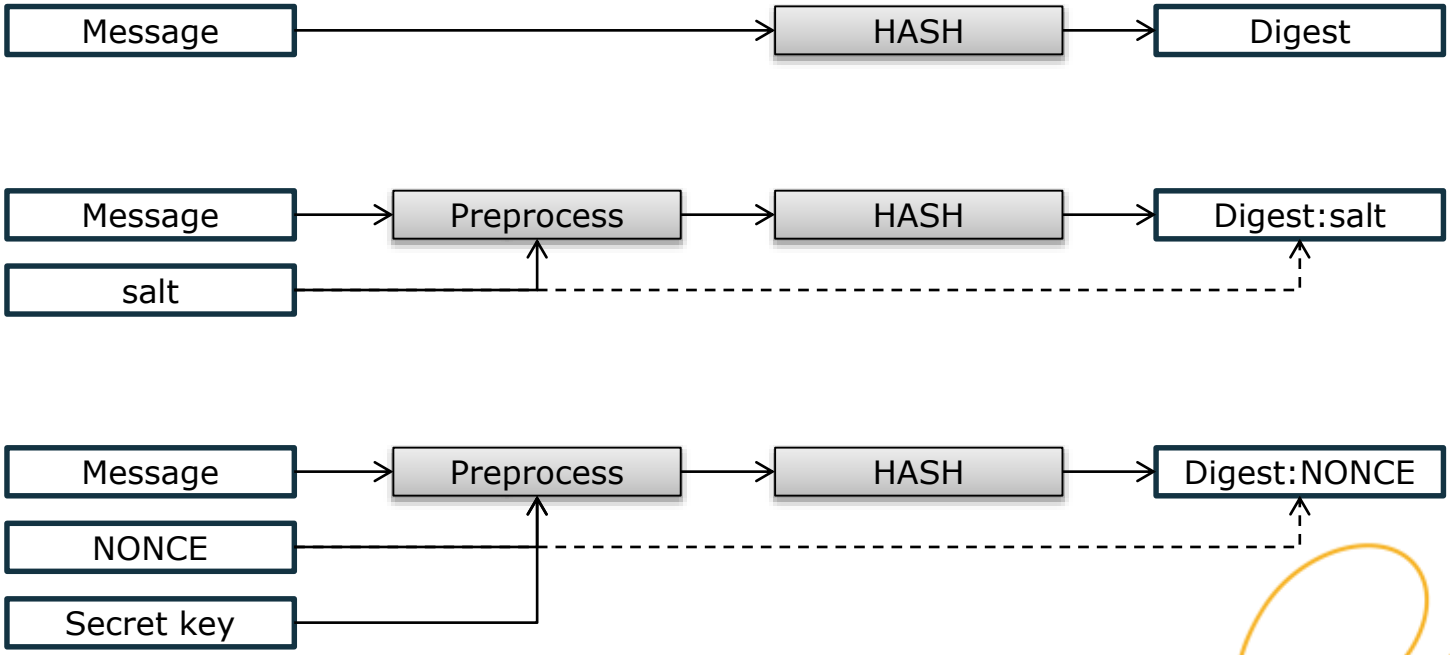
jonas.mellroth@wireflow.se

Cryptographic HASH algorithms

"A hash function which is considered practically impossible to invert, that is, to recreate the input data from its hash value alone."

- MD5 – broken and regarded unsecure
- SHA-0 – depracted and collisions can be found in ~2^33 hashes
- SHA-1 – still no collisions found, but is replaced by the SHA-2 family
- SHA-2 – the most widely used hash algorithm today. (224, 256, 384 or 512 bits), no attacks reported

HASH vs MAC



A hash is repeatable unless we add salt and preprocess before hashing.

By adding NONCE and a secret key, the hash becomes a MAC. NONCE = Number used ONCE

jonas.mellroth@wireflow.se