Security with MQTT Protocol

E. Wegscheider, SEC Consult, e.wegscheider@sec-consult.com

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825050-5GCroCo

About me

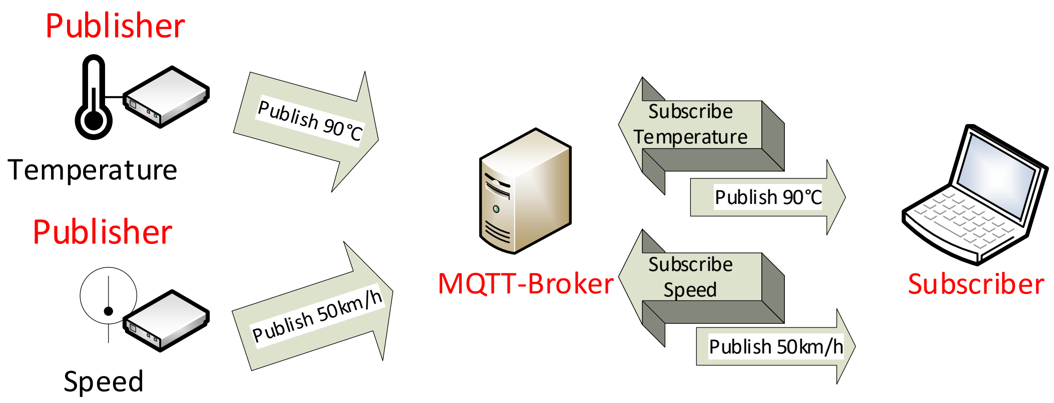
- Erik Wegscheider
    - Senior Security Consultant
        - Military Projects (DOD-STD-2167A)
        - Data Security (PII-data, Health Data)
        - General Data Protection Regulation (GDPR)
        - Data Centers
        - Network Architecture
        - Cyber Security
    - Living in Austria
    - Working in Austria and Germany

Agenda

- MQTT-Protocol version v3.1.1
  - Basics
  - In accordance with the OSI Layers
  - illustrated in Wireshark
  - shown in default configuration
  - Boosting Step by Step the MQTT security
  - Encryption Methods
  - Authentication Methods
  - Data Integrity

MQTT Protocol – Basics (1)

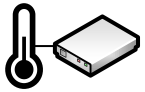
- MQTT is a Protocol for Machine-to-Machine-Communication
- Could use TCP/IP as transport media (could also use ZigBee, …)
- MQTT System is using 3 components:
  - Publisher (sensors, … for collecting data)
  - Subscriber (consumers of the data)
  - Broker

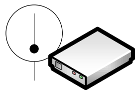


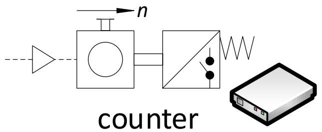
MQTT Protocol – Basics (2)

Component Publisher:

- is publishing messages → sending data / messages to the broker
- get the data from devices like sensors
- could also be a battery powered device
- could also stay most of the time in sleeping mode
- wake up a short time and send Data and then go back in sleeping mode
- can also exist behind NAT-Routers
- Publishers are only able to setup the connection from the their side


Temperature


Speed



counter

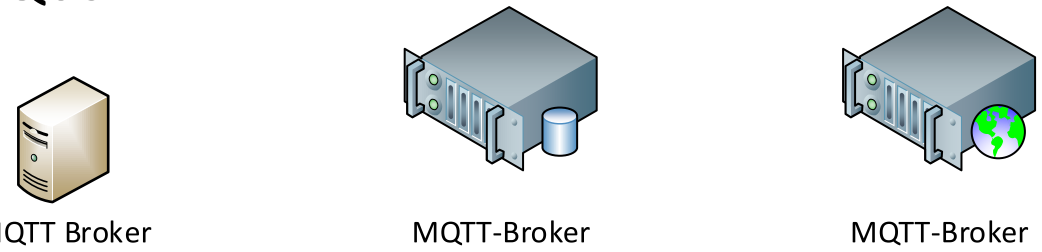

IoT Sensor

Both publishers and subscribers are MQTT clients.

MQTT Protocol – Basics (3)

Component Broker:

- the counterpart of the MQTT client is the MQTT broker
- the broker is a service that should be reachable twenty-four-seven
- communicates with both Publishers and Subscribers
- will receive and store the data and messages from the Publishers
- will answer data requests from Subscribers
- will send the data to the Subscribers
- Is able to handle QoS

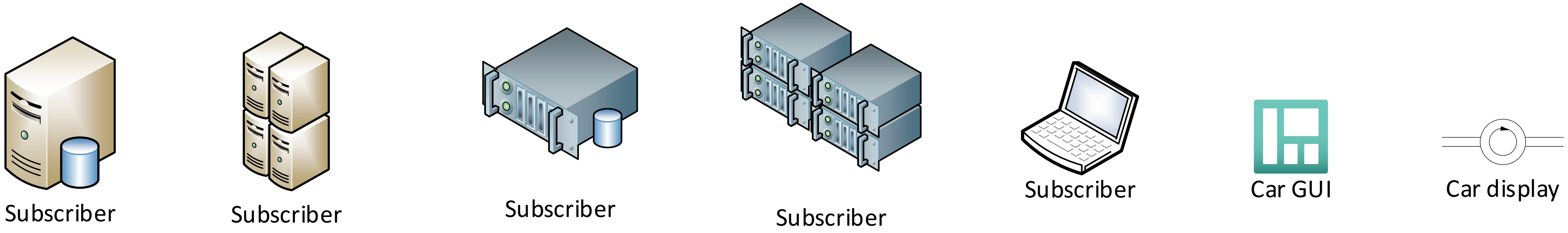


Both publishers and subscribers are MQTT clients.

MQTT Protocol – Basics (4)

Component Subscriber:

- will contact the Broker – never the Publishers
- will subscribe to one or more services
- will receive the data for all the subscribed services from the broker – the data that the broker has received from the Publishers



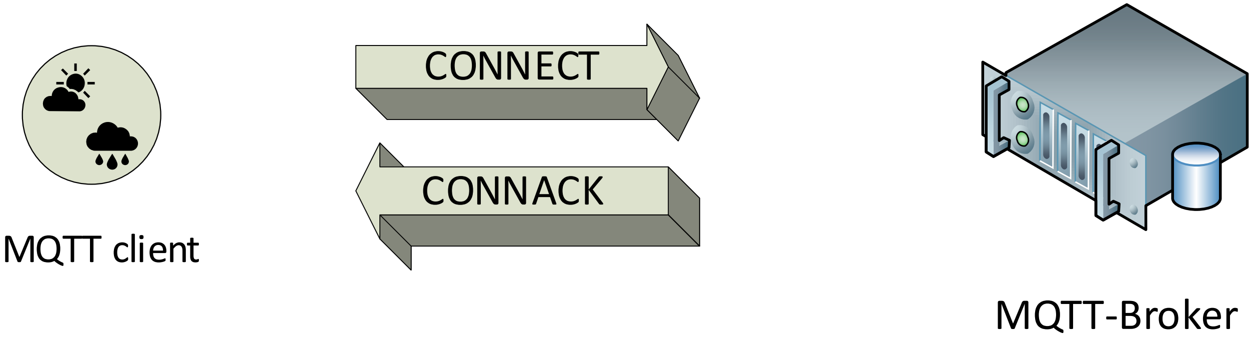
Both publishers and subscribers are MQTT clients.

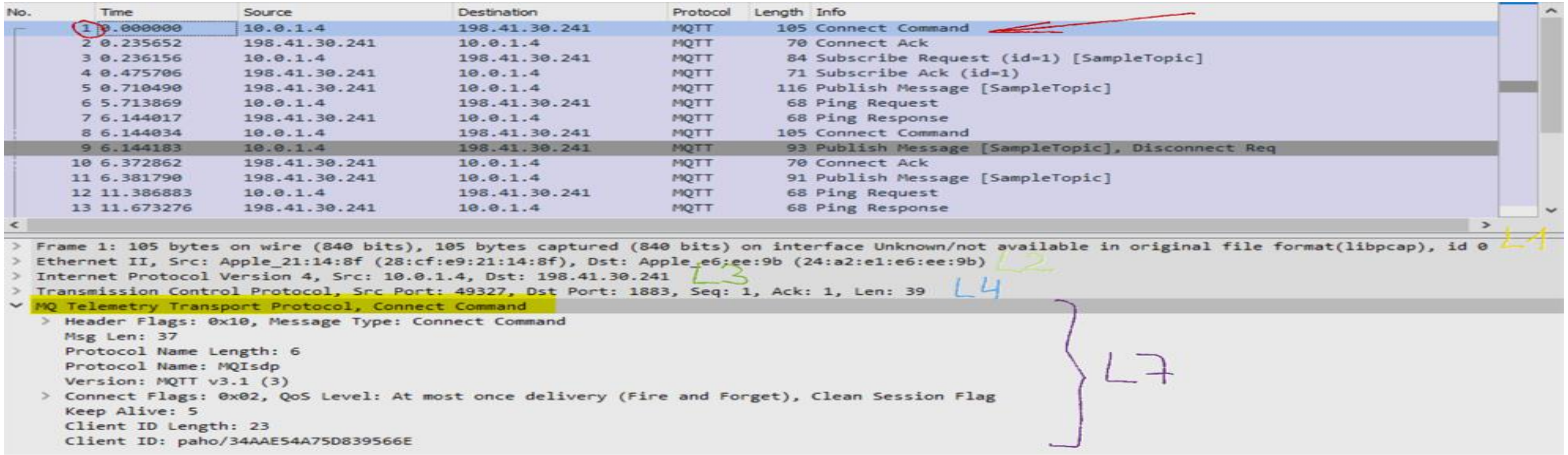
MQTT Protocol – Basics (5)

OSI Layer		MQTT Protocol		
7	APPLICATION	L7	APPLICATION	MQTT
6	PRESENTATION			
5	SESSION			
4	TRANSPORT	L4	TRANSPORT	TCP
3	NETWORK	L3	NETWORK	IP
2	DATALINK	L2	DATALINK	
1	PHYSICAL	L1	PHYSICAL	

MQTT Protocol – Basics (6)

MQTT Connection between client and broker: CONNECT





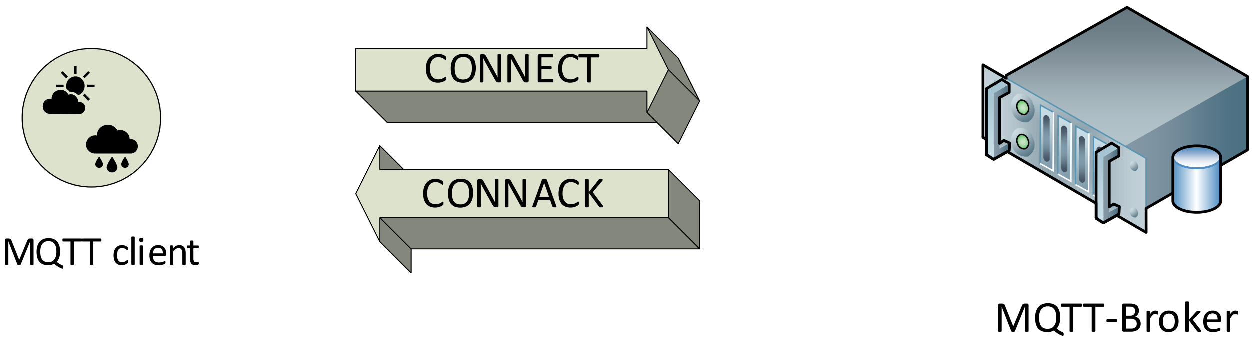
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
2	0.235652	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
3	0.236156	10.0.1.4	198.41.30.241	MQTT	84	Subscribe Request (id=1) [SampleTopic]
4	0.475706	198.41.30.241	10.0.1.4	MQTT	71	Subscribe Ack (id=1)
5	0.710490	198.41.30.241	10.0.1.4	MQTT	116	Publish Message [SampleTopic]
6	5.713869	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
7	6.144017	198.41.30.241	10.0.1.4	MQTT	68	Ping Response
8	6.144034	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
9	6.144183	10.0.1.4	198.41.30.241	MQTT	93	Publish Message [SampleTopic], Disconnect Req
10	6.372862	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
11	6.381790	198.41.30.241	10.0.1.4	MQTT	91	Publish Message [SampleTopic]
12	11.386883	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
13	11.673276	198.41.30.241	10.0.1.4	MQTT	68	Ping Response

```
> Frame 1: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface Unknown/not available in original file format(libpcap), id 0   L1
> Ethernet II, Src: Apple_21:14:8f (28:cf:e9:21:14:8f), Dst: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b)   L2
> Internet Protocol Version 4, Src: 10.0.1.4, Dst: 198.41.30.241   L3
> Transmission Control Protocol, Src Port: 49327, Dst Port: 1883, Seq: 1, Ack: 1, Len: 39   L4
v MQ Telemetry Transport Protocol, Connect Command
  > Header Flags: 0x10, Message Type: Connect Command
    Msg Len: 37
    Protocol Name Length: 6
    Protocol Name: MQIsdp
    Version: MQTT v3.1 (3)
  > Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget), Clean Session Flag
    Keep Alive: 5
    Client ID Length: 23
    Client ID: paho/34AAE54A75D839566E
```

L7

MQTT Protocol – Basics (7)

MQTT Connection between client and broker: CONNACK

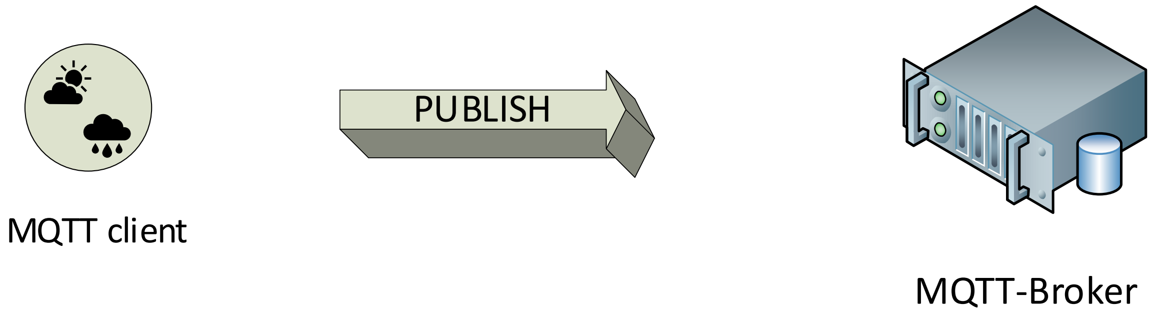


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
2	0.235652	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
3	0.236156	10.0.1.4	198.41.30.241	MQTT	84	Subscribe Request (id=1) [SampleTopic]
4	0.475706	198.41.30.241	10.0.1.4	MQTT	71	Subscribe Ack (id=1)
5	0.710490	198.41.30.241	10.0.1.4	MQTT	116	Publish Message [SampleTopic]
6	5.713869	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
7	6.144017	198.41.30.241	10.0.1.4	MQTT	68	Ping Response
8	6.144034	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
9	6.144183	10.0.1.4	198.41.30.241	MQTT	93	Publish Message [SampleTopic], Disconnect Req
10	6.372862	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
11	6.381790	198.41.30.241	10.0.1.4	MQTT	91	Publish Message [SampleTopic]
12	11.386883	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
13	11.673276	198.41.30.241	10.0.1.4	MQTT	68	Ping Response

```
> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface Unknown/not available in original file format(libpcap), id 0   L1
> Ethernet II, Src: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b), Dst: Apple_21:14:8f (28:cf:e9:21:14:8f)   L2
> Internet Protocol Version 4, Src: 198.41.30.241, Dst: 10.0.1.4   L3
> Transmission Control Protocol, Src Port: 1883, Dst Port: 49327, Seq: 1, Ack: 40, Len: 4   L4
v MQ Telemetry Transport Protocol, Connect Ack
   > Header Flags: 0x20, Message Type: Connect Ack
     Msg Len: 2                                             L7
     Reserved: Not set
     Return Code: Connection Accepted (0)
```

MQTT Protocol – Basics (8)

MQTT client sends a PUBLISH message to the broker:



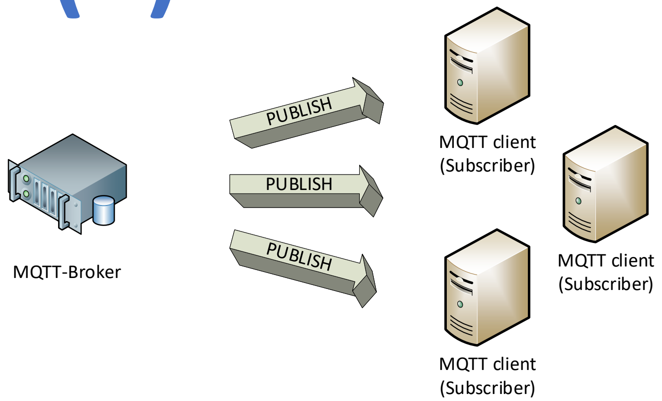
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
2	0.235652	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
3	0.236156	10.0.1.4	198.41.30.241	MQTT	84	Subscribe Request (id=1) [SampleTopic]
4	0.475706	198.41.30.241	10.0.1.4	MQTT	71	Subscribe Ack (id=1)
5	0.710490	198.41.30.241	10.0.1.4	MQTT	116	Publish Message [SampleTopic]
6	5.713869	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
7	6.144017	198.41.30.241	10.0.1.4	MQTT	68	Ping Response
8	6.144034	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
9	6.144183	10.0.1.4	198.41.30.241	MQTT	93	Publish Message [SampleTopic], Disconnect Req
10	6.372862	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
11	6.381790	198.41.30.241	10.0.1.4	MQTT	91	Publish Message [SampleTopic]
12	11.386883	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
13	11.673276	198.41.30.241	10.0.1.4	MQTT	68	Ping Response

```
> Frame 9: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface Unknown/not available in original file format(libpcap), id 0   L1
> Ethernet II, Src: Apple_21:14:8f (28:cf:e9:21:14:8f), Dst: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b)   L2
> Internet Protocol Version 4, Src: 10.0.1.4, Dst: 198.41.30.241   L3
> Transmission Control Protocol, Src Port: 49330, Dst Port: 1883, Seq: 40, Ack: 1, Len: 27   L4
v MQ Telemetry Transport Protocol, Publish Message
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    Msg Len: 23
    Topic Length: 11
    Topic: SampleTopic
    Message: 48656c6c6f204d515454
v MQ Telemetry Transport Protocol, Disconnect Req
  > Header Flags: 0xe0, Message Type: Disconnect Req
    Msg Len: 0
```

L7

MQTT Protocol – Basics (9)

Broker sends the published message to all subscribers that are interested (have subscribed) for that topic:

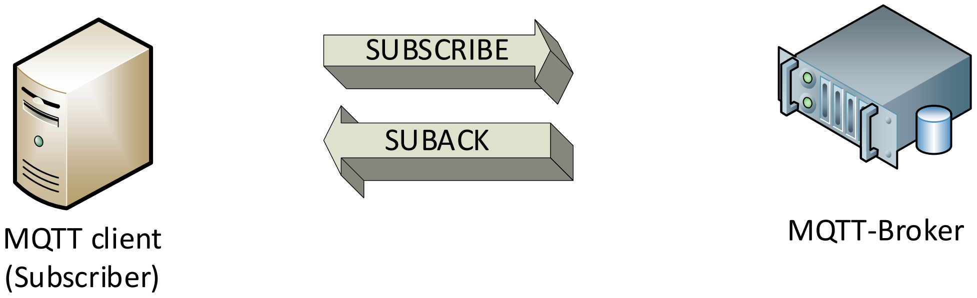


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
2	0.235652	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
3	0.236156	10.0.1.4	198.41.30.241	MQTT	84	Subscribe Request (id=1) [SampleTopic]
4	0.475706	198.41.30.241	10.0.1.4	MQTT	71	Subscribe Ack (id=1)
5	0.710490	198.41.30.241	10.0.1.4	MQTT	116	Publish Message [SampleTopic]
6	5.713869	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
7	6.144017	198.41.30.241	10.0.1.4	MQTT	68	Ping Response
8	6.144034	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
9	6.144183	10.0.1.4	198.41.30.241	MQTT	93	Publish Message [SampleTopic], Disconnect Req
10	6.372862	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
11	6.381790	198.41.30.241	10.0.1.4	MQTT	91	Publish Message [SampleTopic]
12	11.386883	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
13	11.673276	198.41.30.241	10.0.1.4	MQTT	68	Ping Response

```
> Frame 11: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface Unknown/not available in original file format(libpcap), id 0   L1
> Ethernet II, Src: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b), Dst: Apple_21:14:8f (28:cf:e9:21:14:8f)   L2
> Internet Protocol Version 4, Src: 198.41.30.241, Dst: 10.0.1.4   L3
> Transmission Control Protocol, Src Port: 1883, Dst Port: 49327, Seq: 62, Ack: 60, Len: 25   L4
v MQ Telemetry Transport Protocol, Publish Message
    > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
      Msg Len: 23
      Topic Length: 11
      Topic: SampleTopic
      Message: 48656c6c6f204d515454
                                                                                          L7
```

MQTT Protocol – Basics (10)

MQTT client sends a subscribe message for a topic to the broker: SUBSCRIBE

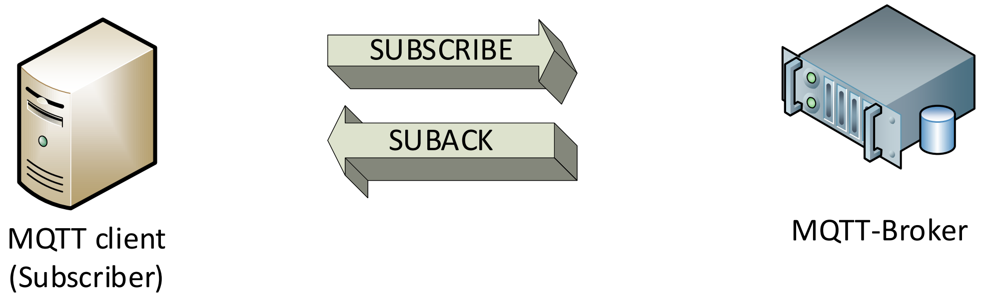


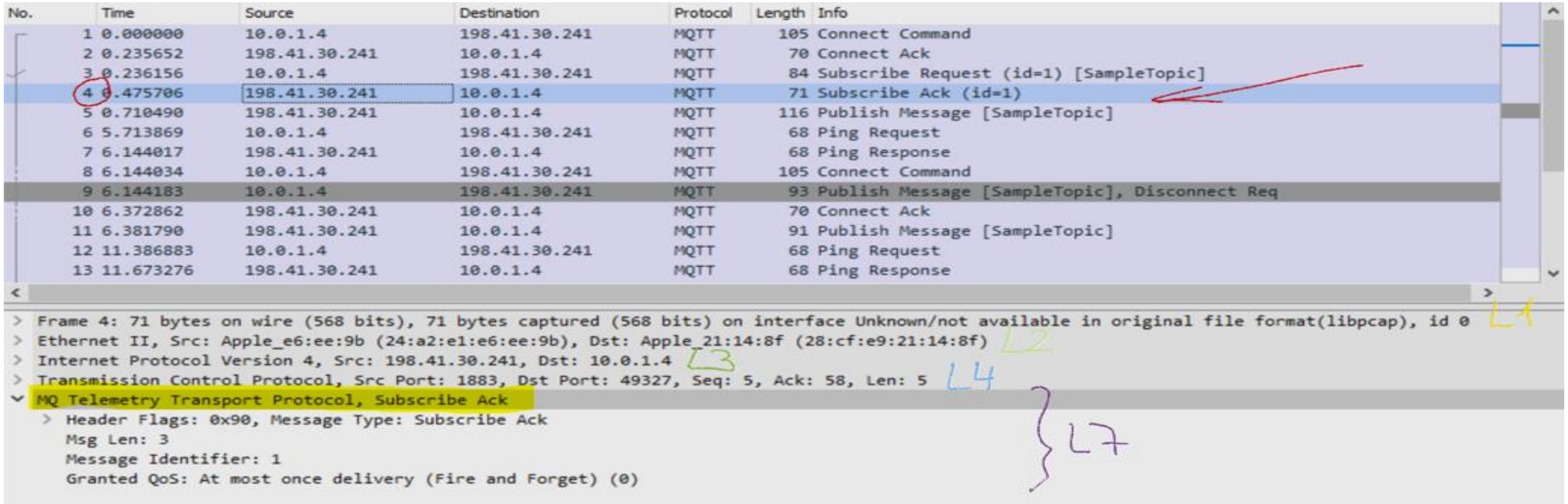
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
2	0.235652	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
3	0.236156	10.0.1.4	198.41.30.241	MQTT	84	Subscribe Request (id=1) [SampleTopic]
4	0.475706	198.41.30.241	10.0.1.4	MQTT	71	Subscribe Ack (id=1)
5	0.710490	198.41.30.241	10.0.1.4	MQTT	116	Publish Message [SampleTopic]
6	5.713869	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
7	6.144017	198.41.30.241	10.0.1.4	MQTT	68	Ping Response
8	6.144034	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
9	6.144183	10.0.1.4	198.41.30.241	MQTT	93	Publish Message [SampleTopic], Disconnect Req
10	6.372862	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
11	6.381790	198.41.30.241	10.0.1.4	MQTT	91	Publish Message [SampleTopic]
12	11.386883	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
13	11.673276	198.41.30.241	10.0.1.4	MQTT	68	Ping Response

```
> Frame 3: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface Unknown/not available in original file format(libpcap), id 0   L1
> Ethernet II, Src: Apple_21:14:8f (28:cf:e9:21:14:8f), Dst: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b)   L2
> Internet Protocol Version 4, Src: 10.0.1.4, Dst: 198.41.30.241   L3
> Transmission Control Protocol, Src Port: 49327, Dst Port: 1883, Seq: 40, Ack: 5, Len: 18   L4
v MQ Telemetry Transport Protocol, Subscribe Request
  > Header Flags: 0x82, Message Type: Subscribe Request, QoS Level: At least once delivery (Acknowledged deliver)
    Msg Len: 16
    Message Identifier: 1
    Topic Length: 11
    Topic: SampleTopic
    Requested QoS: At most once delivery (Fire and Forget) (0)
```
L7

MQTT Protocol – Basics (11)

MQTT Broker sends a SUBACK acknowledgment message as confirmation back to the client: SUBACK

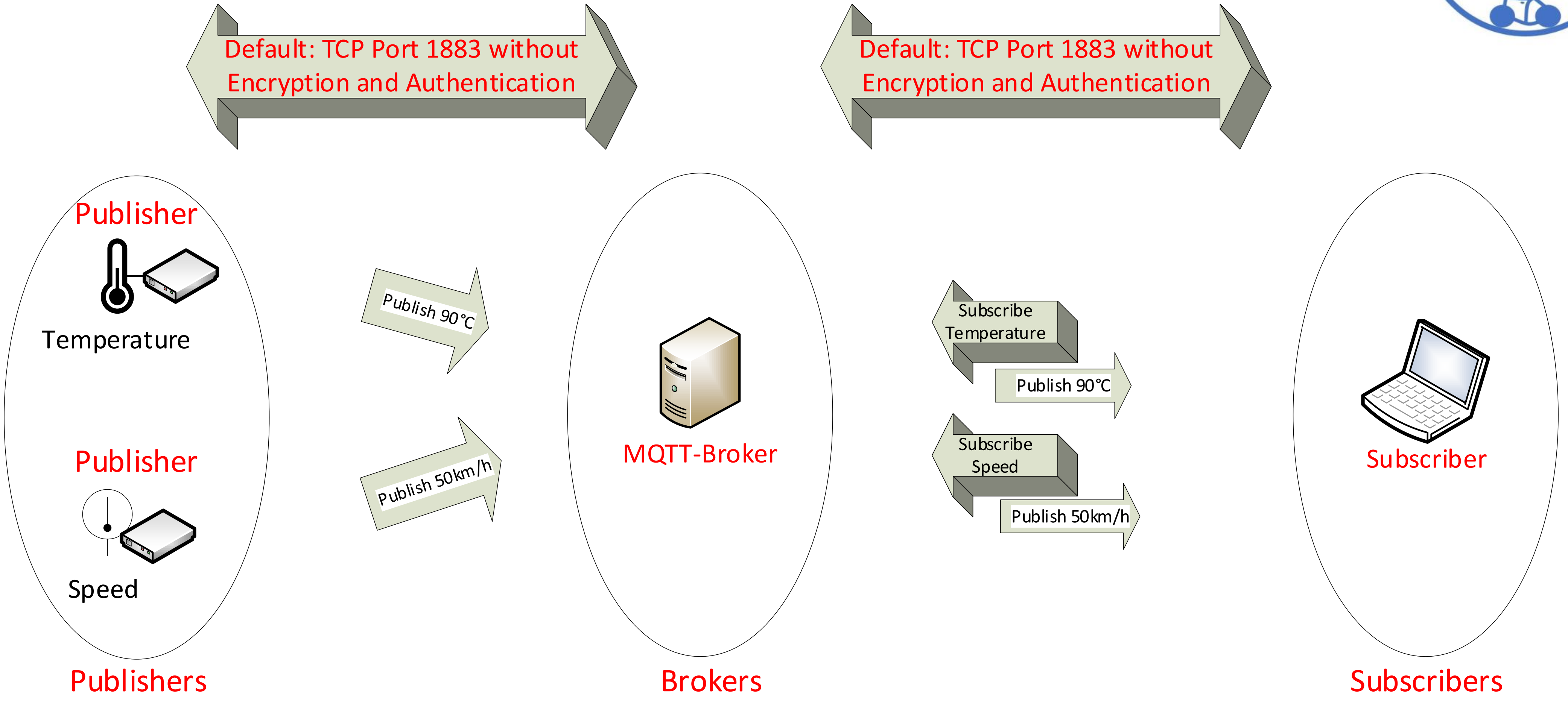




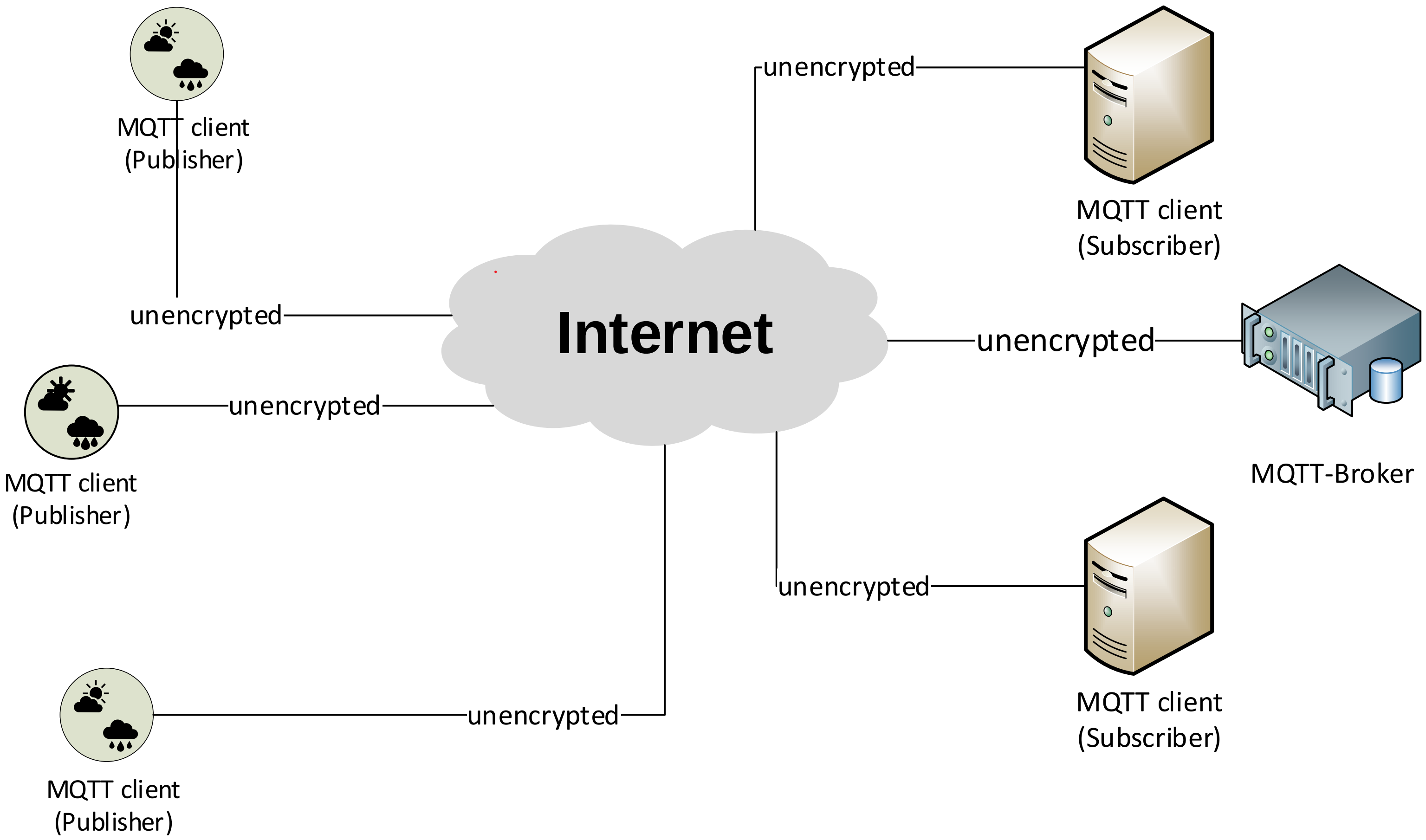
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
2	0.235652	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
3	0.236156	10.0.1.4	198.41.30.241	MQTT	84	Subscribe Request (id=1) [SampleTopic]
4	0.475706	198.41.30.241	10.0.1.4	MQTT	71	Subscribe Ack (id=1)
5	0.710490	198.41.30.241	10.0.1.4	MQTT	116	Publish Message [SampleTopic]
6	5.713869	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
7	6.144017	198.41.30.241	10.0.1.4	MQTT	68	Ping Response
8	6.144034	10.0.1.4	198.41.30.241	MQTT	105	Connect Command
9	6.144183	10.0.1.4	198.41.30.241	MQTT	93	Publish Message [SampleTopic], Disconnect Req
10	6.372862	198.41.30.241	10.0.1.4	MQTT	70	Connect Ack
11	6.381790	198.41.30.241	10.0.1.4	MQTT	91	Publish Message [SampleTopic]
12	11.386883	10.0.1.4	198.41.30.241	MQTT	68	Ping Request
13	11.673276	198.41.30.241	10.0.1.4	MQTT	68	Ping Response

```
> Frame 4: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface Unknown/not available in original file format(libpcap), id 0   L1
> Ethernet II, Src: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b), Dst: Apple_21:14:8f (28:cf:e9:21:14:8f)   L2
> Internet Protocol Version 4, Src: 198.41.30.241, Dst: 10.0.1.4   L3
> Transmission Control Protocol, Src Port: 1883, Dst Port: 49327, Seq: 5, Ack: 58, Len: 5   L4
v MQ Telemetry Transport Protocol, Subscribe Ack
    > Header Flags: 0x90, Message Type: Subscribe Ack
      Msg Len: 3
      Message Identifier: 1
      Granted QoS: At most once delivery (Fire and Forget) (0)
                                                                  L7
```

MQTT Protocol – Basics (12)

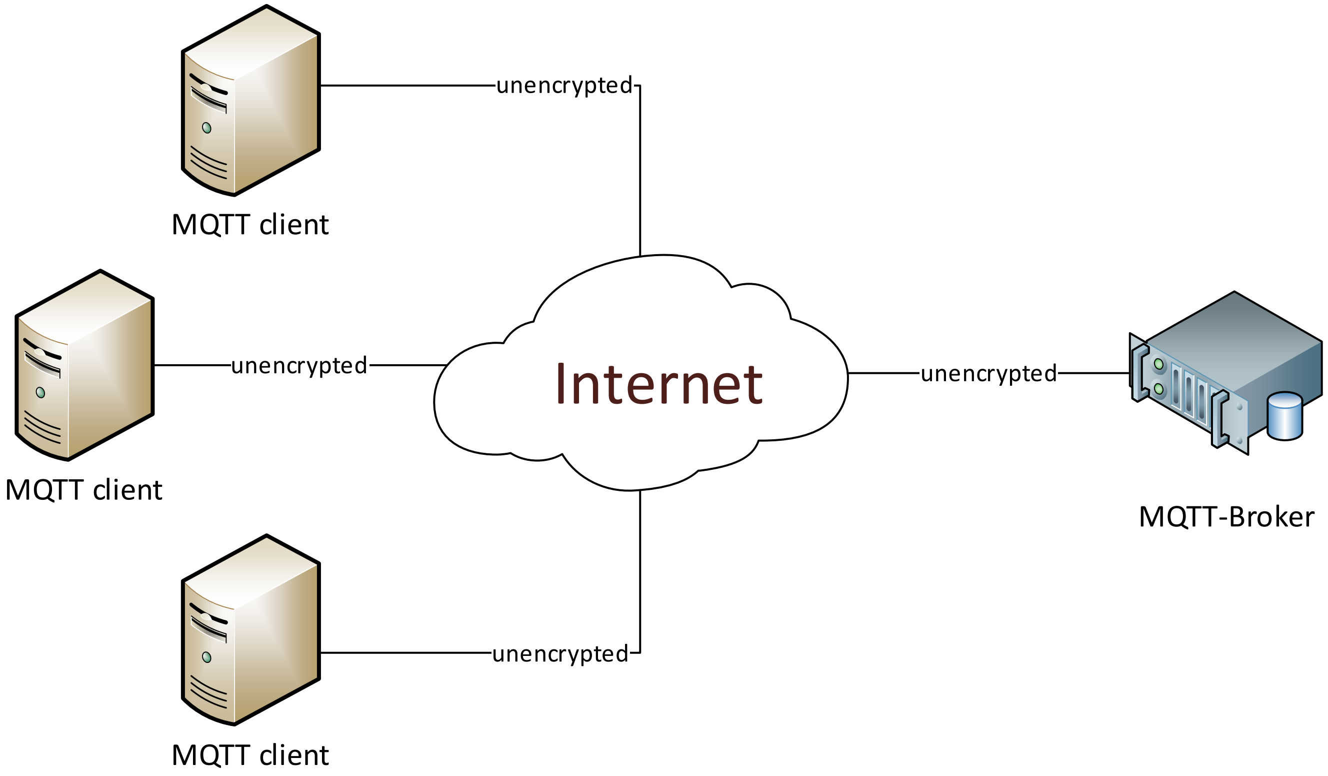


MQTT - using default configuration (no security implemented)



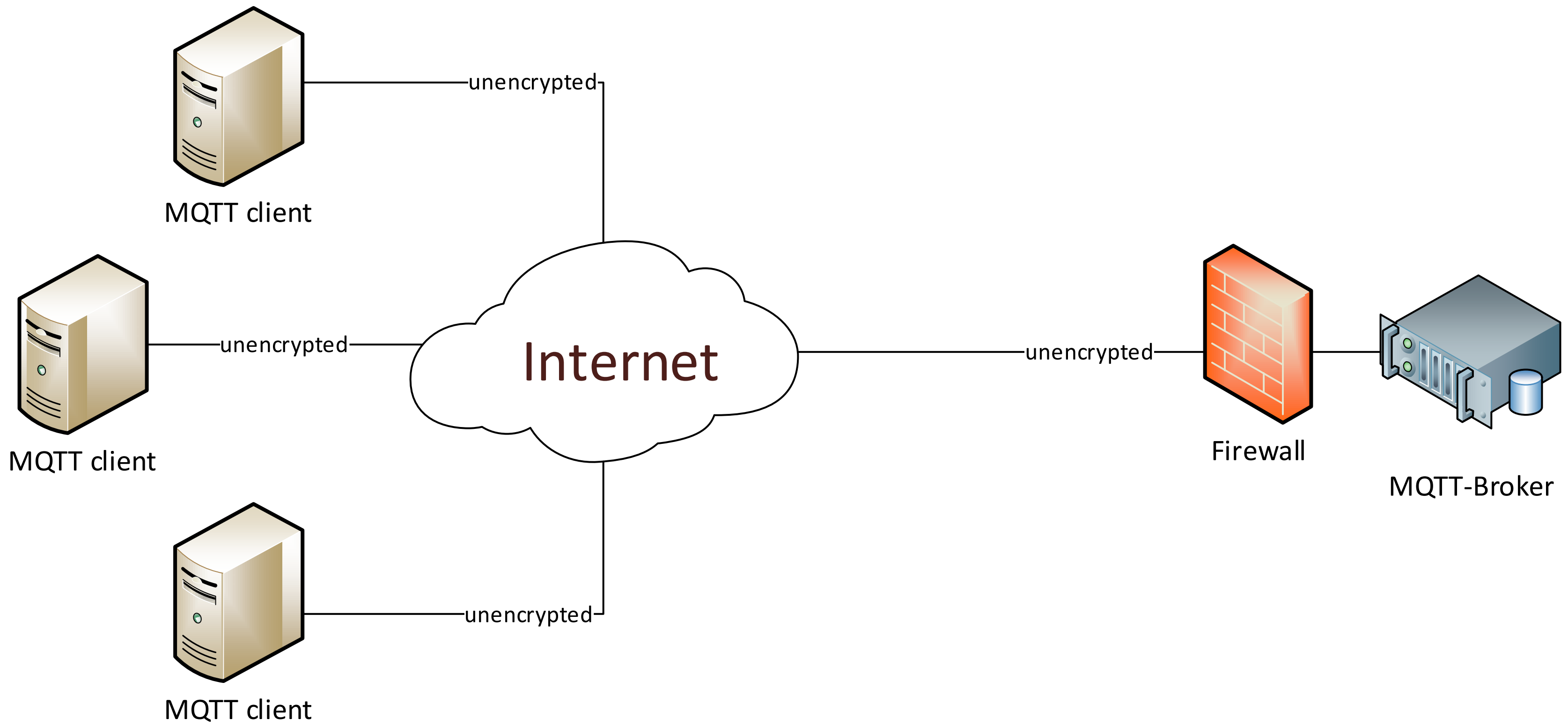
TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

MQTT – using default configuration
(no security implemented)



TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

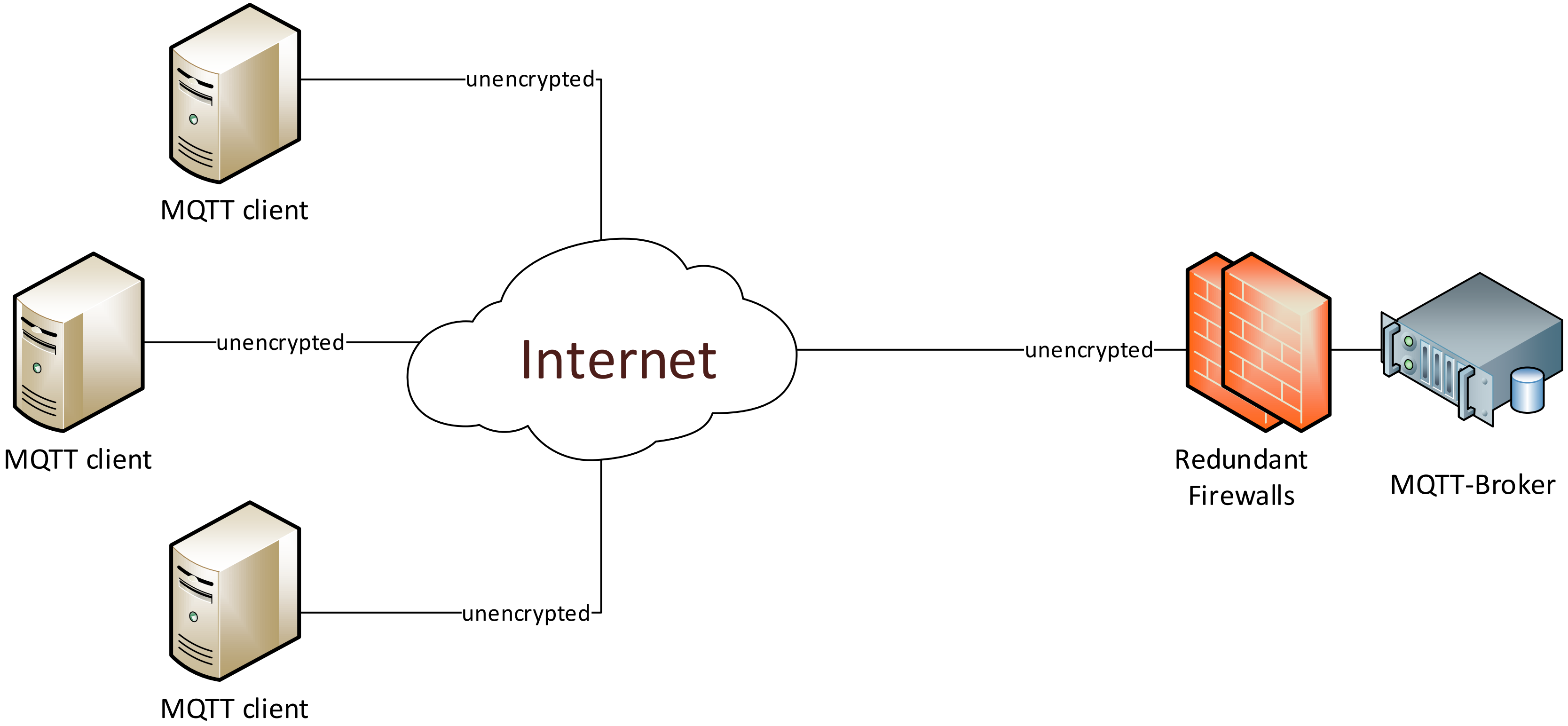
MQTT on the Network
(a Firewall protects the Broker)



Don’t block the following standard MQTT ports:

TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

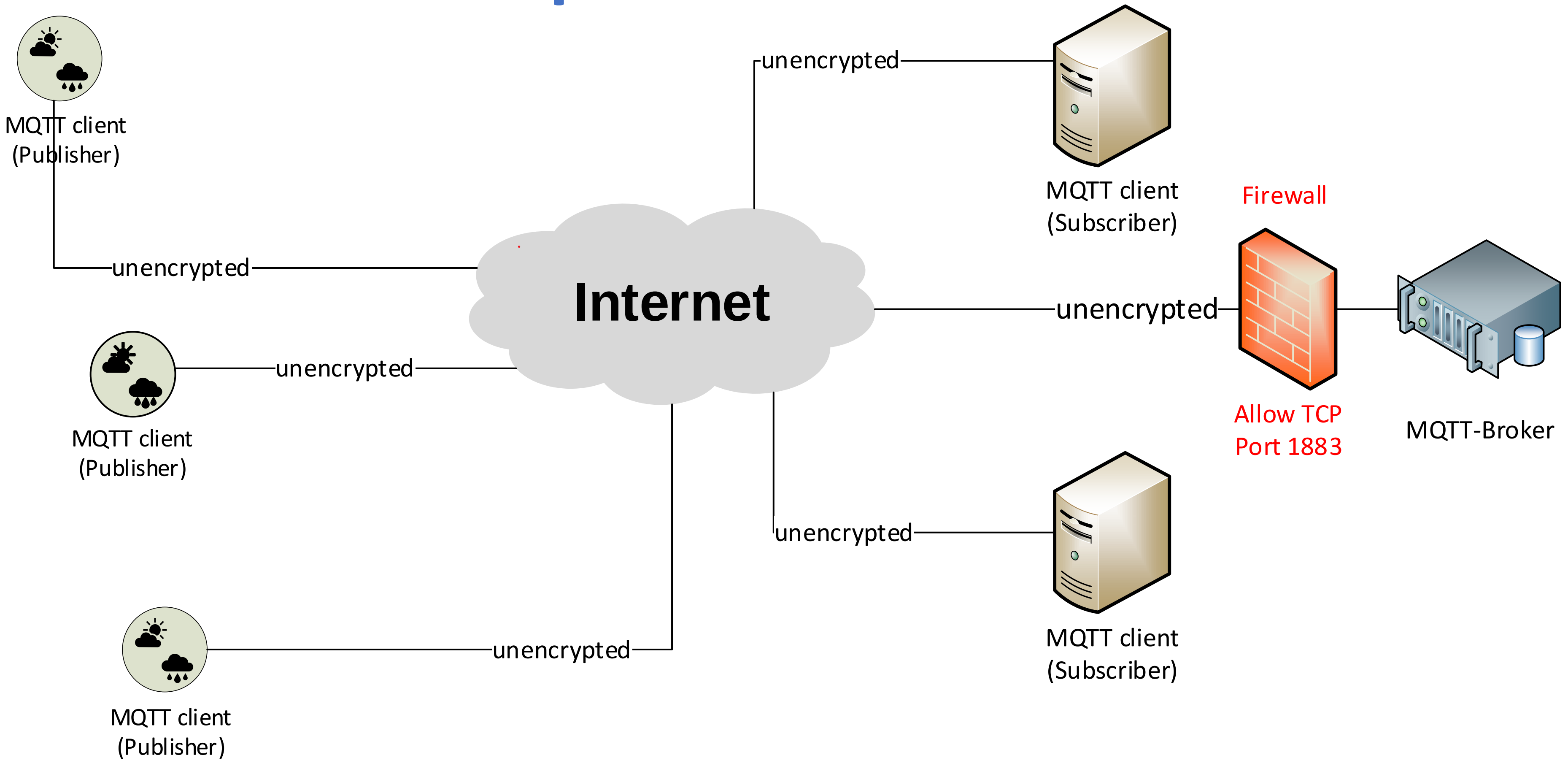
MQTT on the Network
(redundant Firewalls protect the Broker)



Don’t block the following standard MQTT ports:

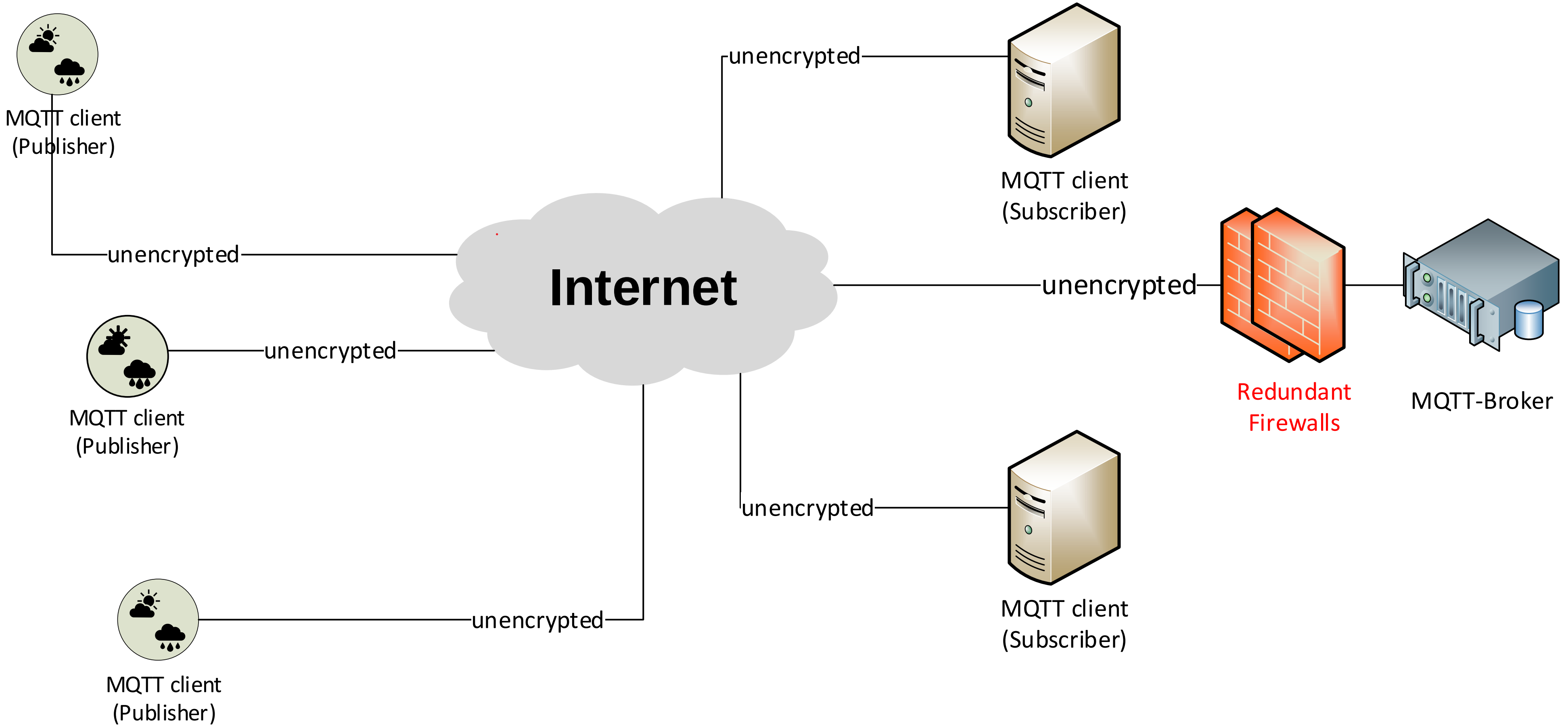
TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

MQTT on the Network
with Firewall to protect the broker



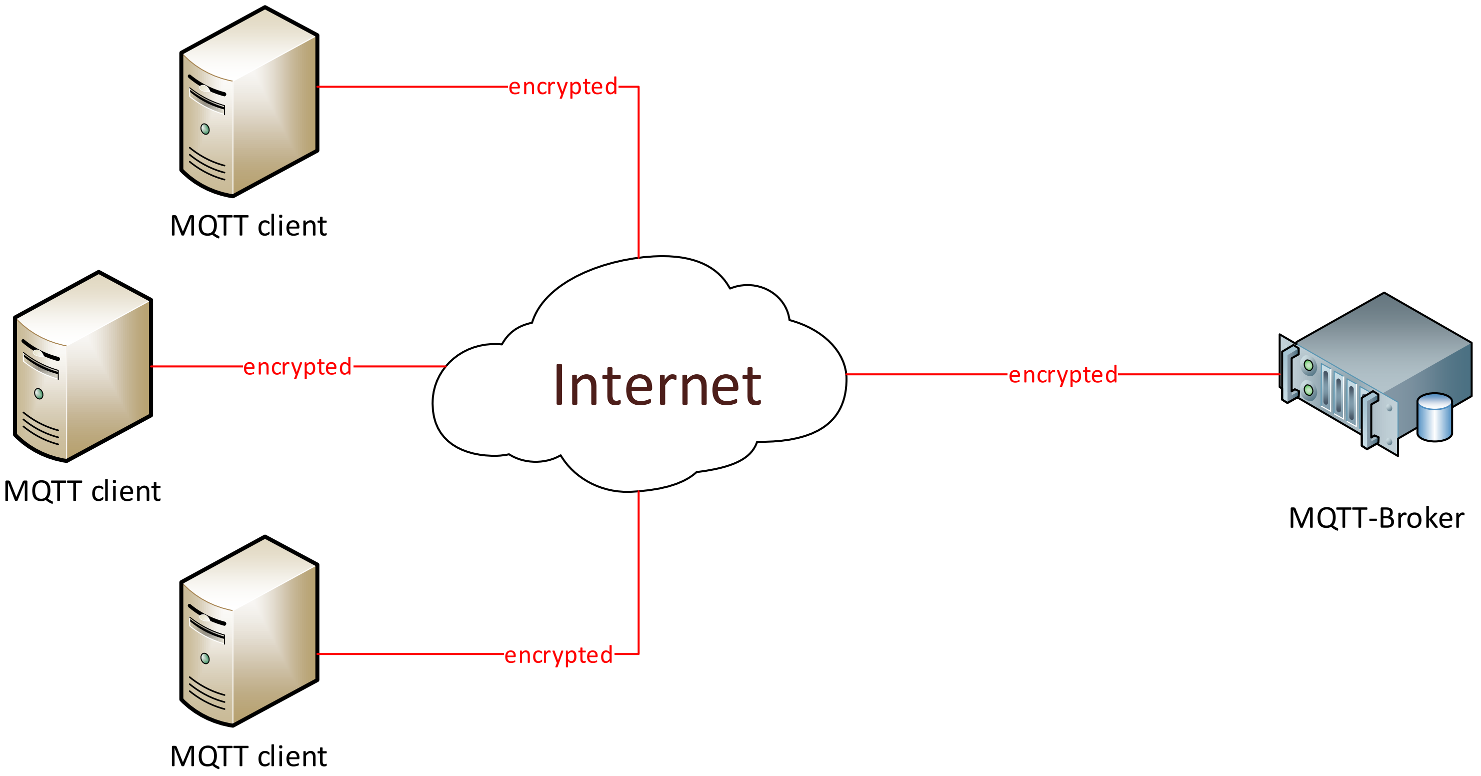
TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

MQTT on the Network
with Firewalls to protect the broker



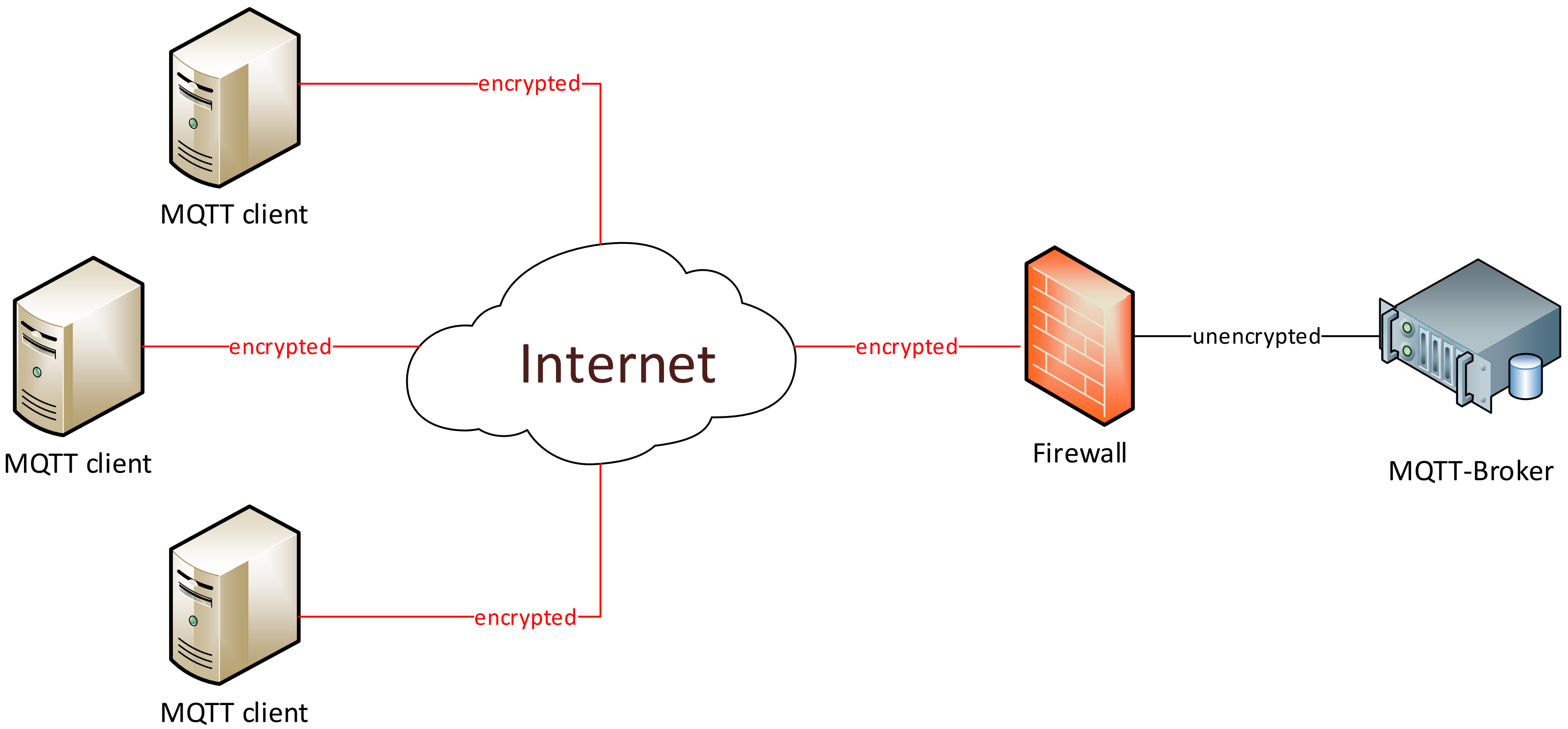
TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

MQTT on the Network
Security: use Encryption



TCP Port 8883: This is the default MQTT port for MQTT over TLS. It’s registered at IANA for Secure MQTT.

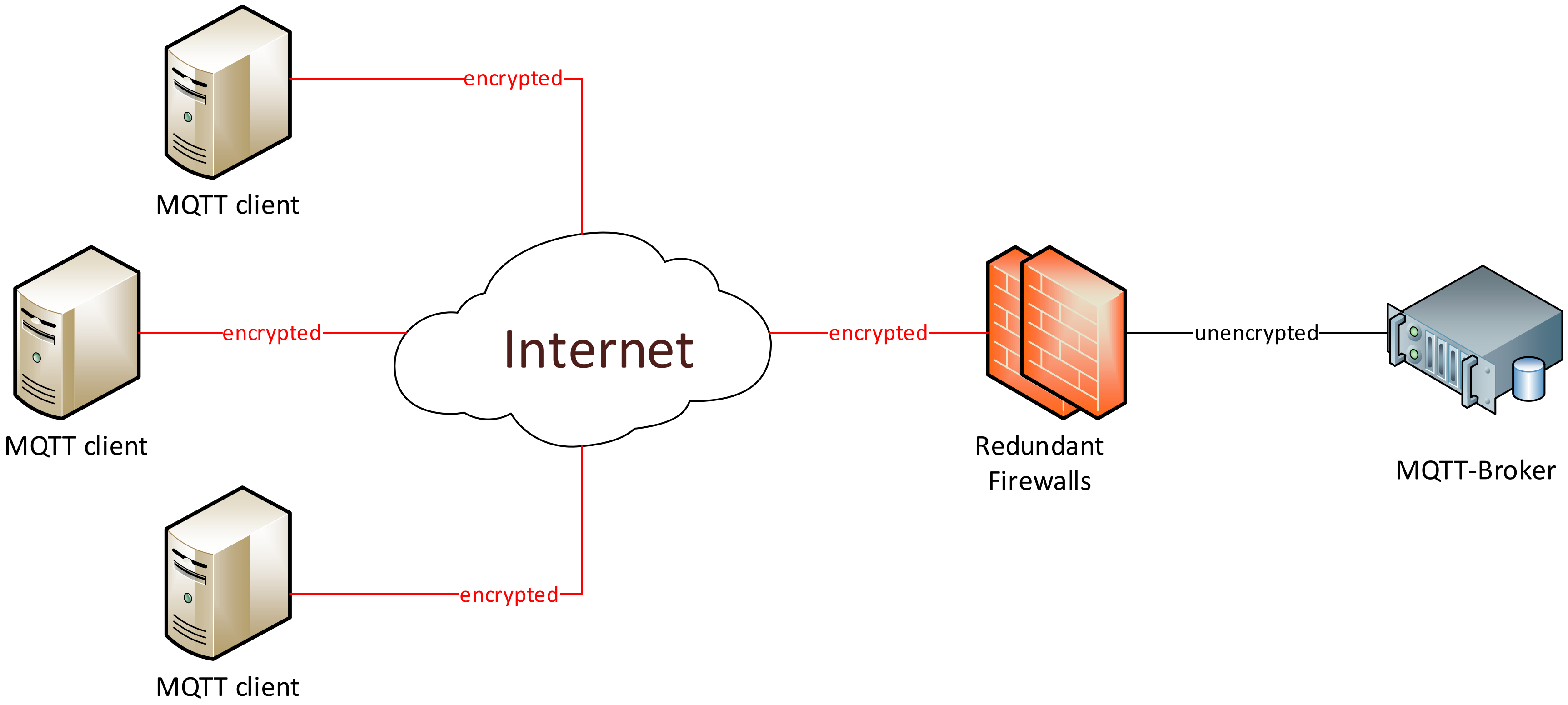
MQTT on the Network
Security: use Encryption and Firewall



TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

TCP Port 8883: This is the default MQTT port for MQTT over TLS. It's registered at IANA for Secure MQTT.

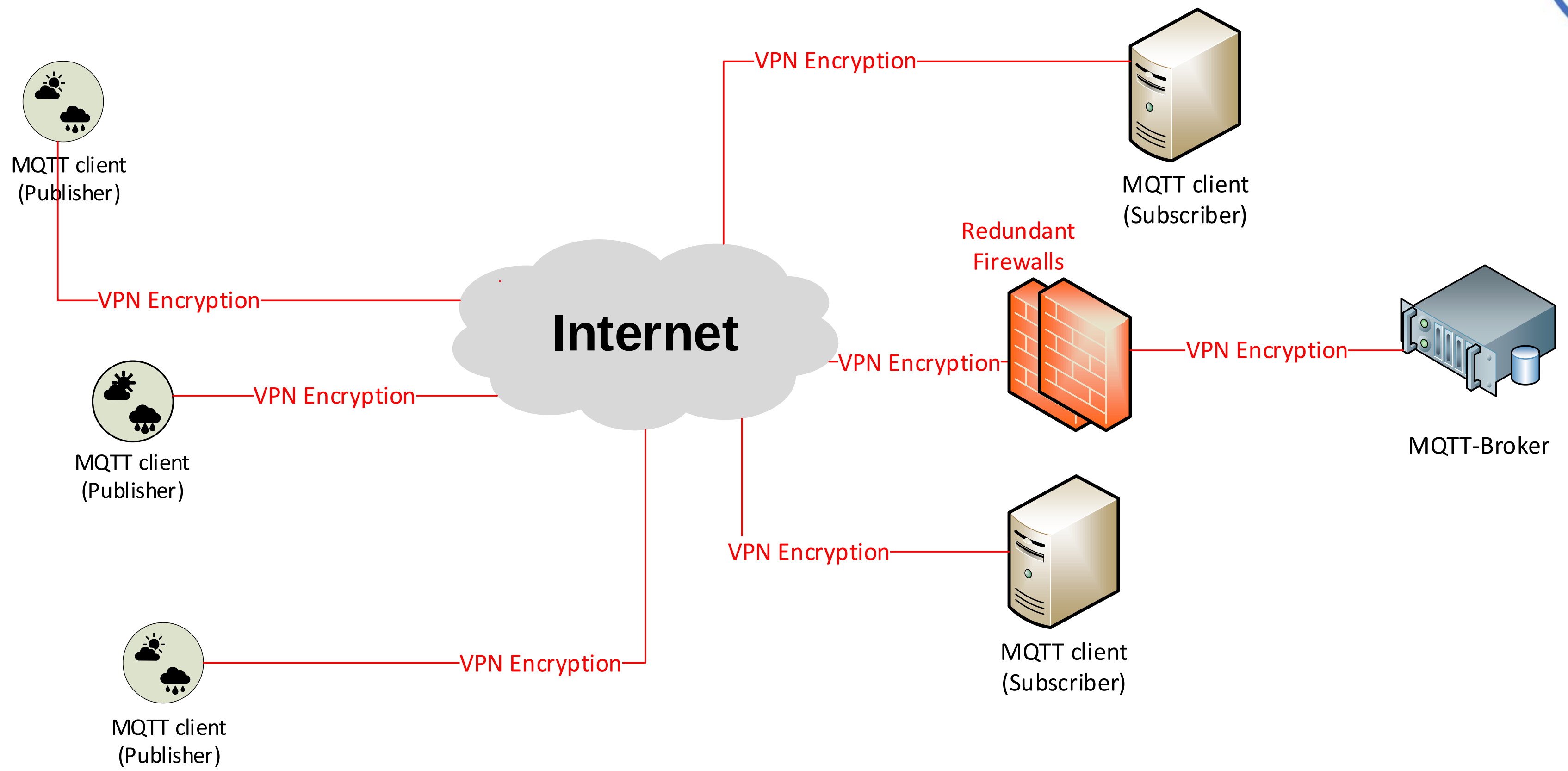
MQTT on the Network
Security: Encryption and Firewalls



TCP Port 1883: This is the default MQTT port. 1883 is defined at IANA as MQTT over TCP.

TCP Port 8883: This is the default MQTT port for MQTT over TLS. It's registered at IANA for Secure MQTT.

MQTT - Encryption with VPN

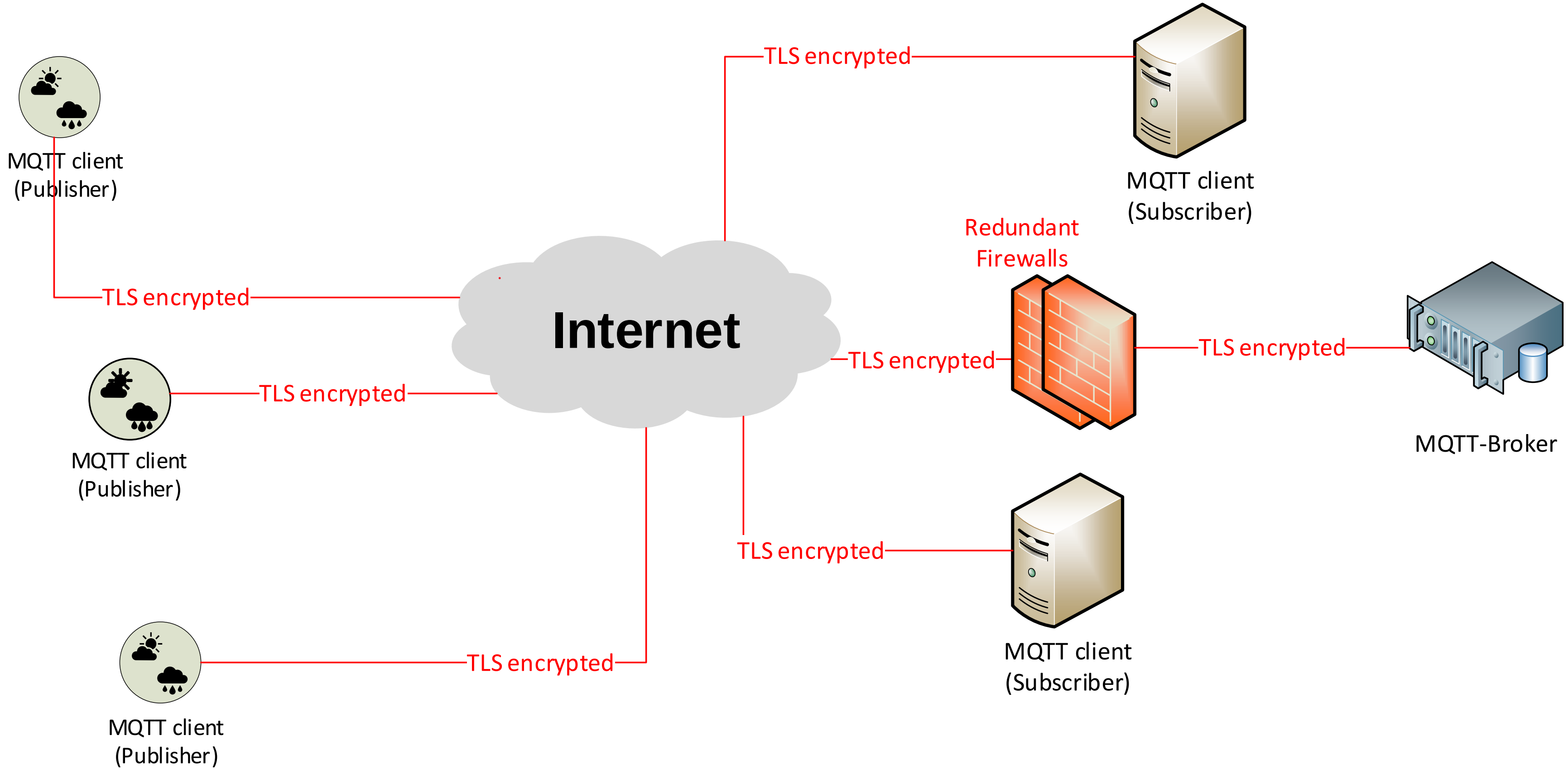


For using VPN, Gateway applications are needed.

VPN connection should include the MQTT devices and MQTT sensors on the client side and the MQTT broker on the other side

VPN needs a big effort: Gateway applications and Plug-Ins are needed for a VPN solution

MQTT - Encryption with TLS



When confidentiality is a goal, TLS is commonly used for transport encryption.

TLS can also provide client-certificate authentication to verify the identity of both sides.

MQTT - Encryption with TLS

OSI Layer	
7	APPLICATION
6	PRESENTATION
5	SESSION
4	TRANSPORT
3	NETWORK
2	DATALINK
1	PHYSICAL

MQTT Protocol		
L7	APPLICATION	MQTT
L5	SESSION	TLS
L4	TRANSPORT	TCP
L3	NETWORK	IP
L2	DATALINK	
L1	PHYSICAL	

MQTT - Encryption with TLS (1)

- Transport Layer Security (TLS) provides a secure communication channel between a client and a server.
- In order to use TLS, the server needs a Public / Private Key Pair.
- TLS use a handshake mechanism to negotiate various parameters to create a secure connection.
- Servers provide a X.509 certificate that clients must use to verify the identity of the server.
- After the handshake is complete, an encrypted communication between client and server is established.
- MQTT clients must always validate the X.509 certificate from the MQTT broker to prevent Man-in-the-Middle attacks.
- Only certificates from trusted CAs should be used.

If CPU intensive Public Key Operations should be avoided, TLS-PSK cipher suites may be an option.

MQTT - Encryption with TLS (2)

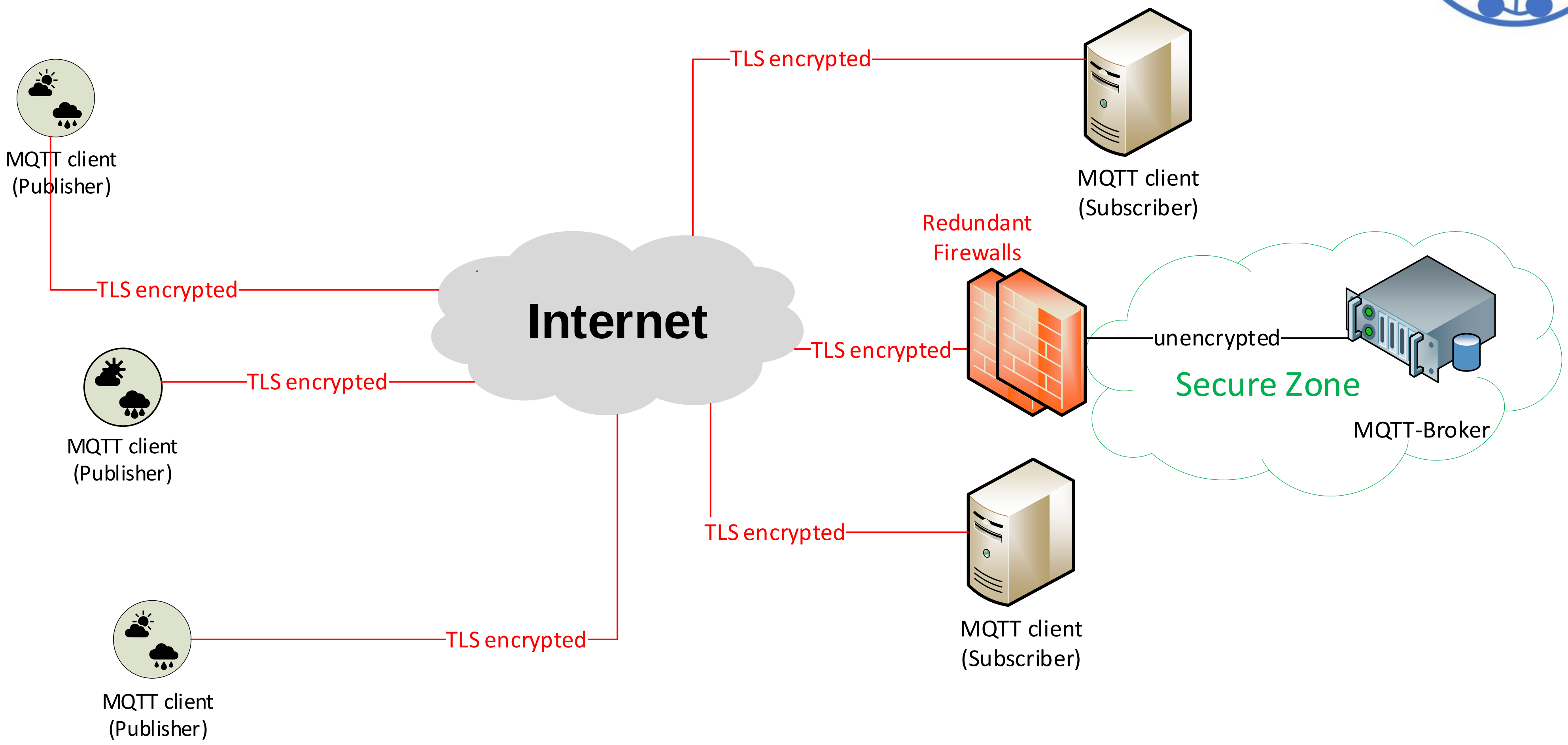
- For using Transport Layer Security (TLS) the server has a Public / Private Key Pair.
- The client can also have a unique Public / Private Key Pair for the TLS handshake.
- The client send its certificate as part of the TLS handshake after it has validated the server certificate.
- The Server must then verify the identity of the client.
- The Server must abort the handshake if the verification of the client certificate fails.
- This process authenticates the client before a secure connection will be established.

Using also client certificates has the following advantages:

- Verification of the identity of the MQTT clients
- Authentication of the MQTT clients at the transport level
- Lock out invalid MQTT clients before MQTT CONNECT messages are sent

To prevent malicious clients from using leaked certificates, a certification revocation mechanism is needed: Certificate Revocation List (CRL) is a list of invalid certificates

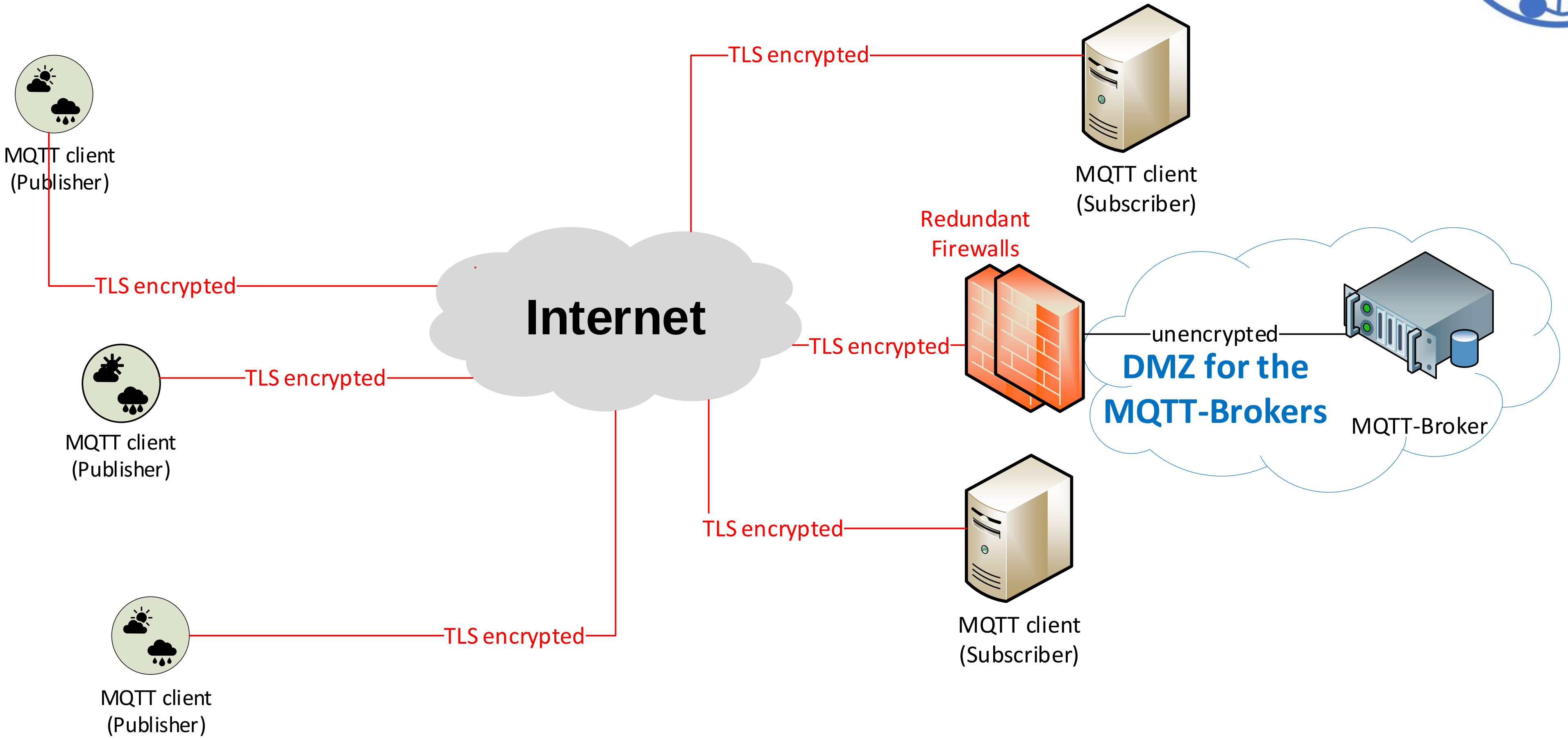
MQTT on the Network
Security: TLS Encryption and Firewalls



The firewall must be able to encrypt / decrypt the traffic to examine the data.

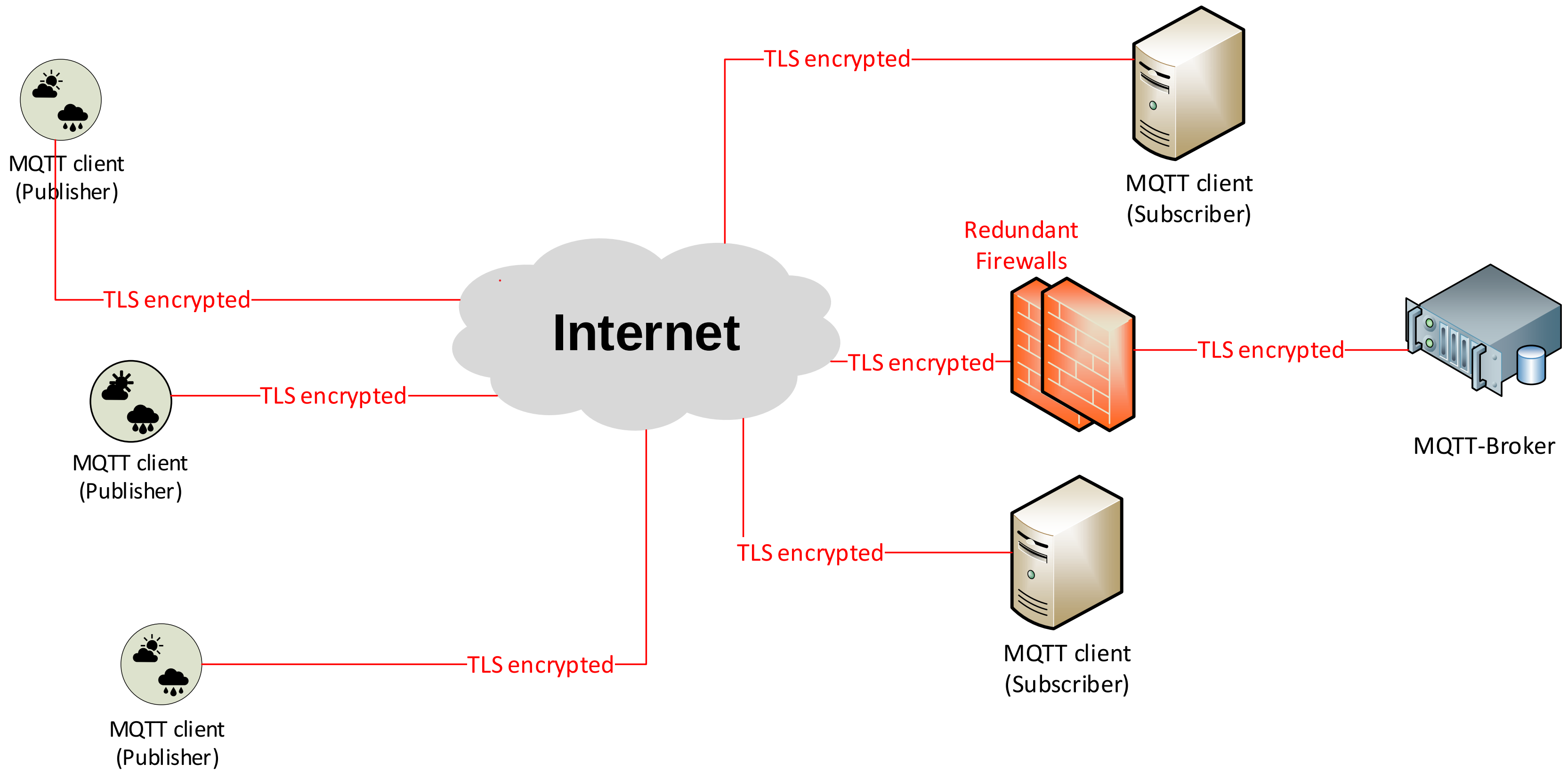
MQTT on the Network

Security: DMZ, TLS Encryption and Firewalls



The firewall must be able to encrypt / decrypt the traffic to examine the data.

MQTT on the Network
Security: TLS Encryption and Firewalls

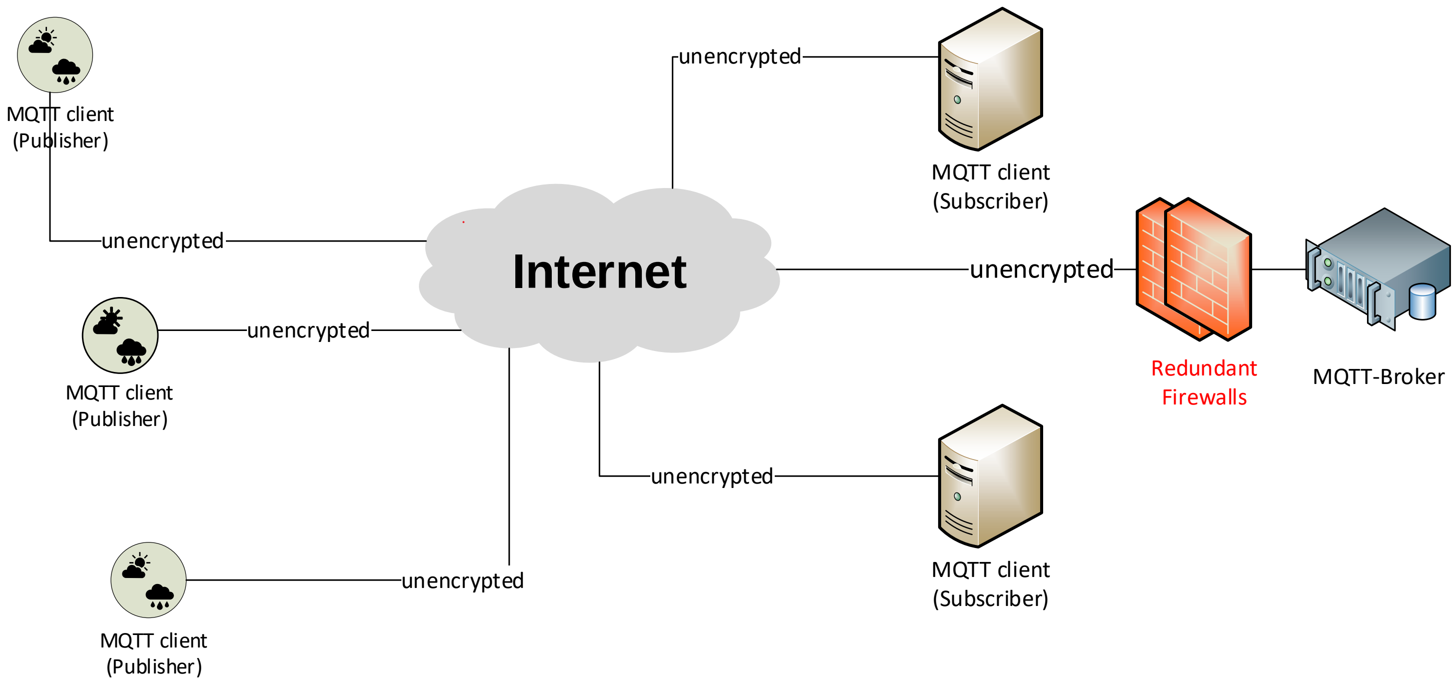


If the MQTT brokers are not in a secure zone, the firewall must create a new encrypted channel to the MQTT broker.

The firewall must be able to encrypt / decrypt the traffic to examine the data.

MQTT - Authentication Methods (1)

- MQTT protocol provides Authentication with Username and Password

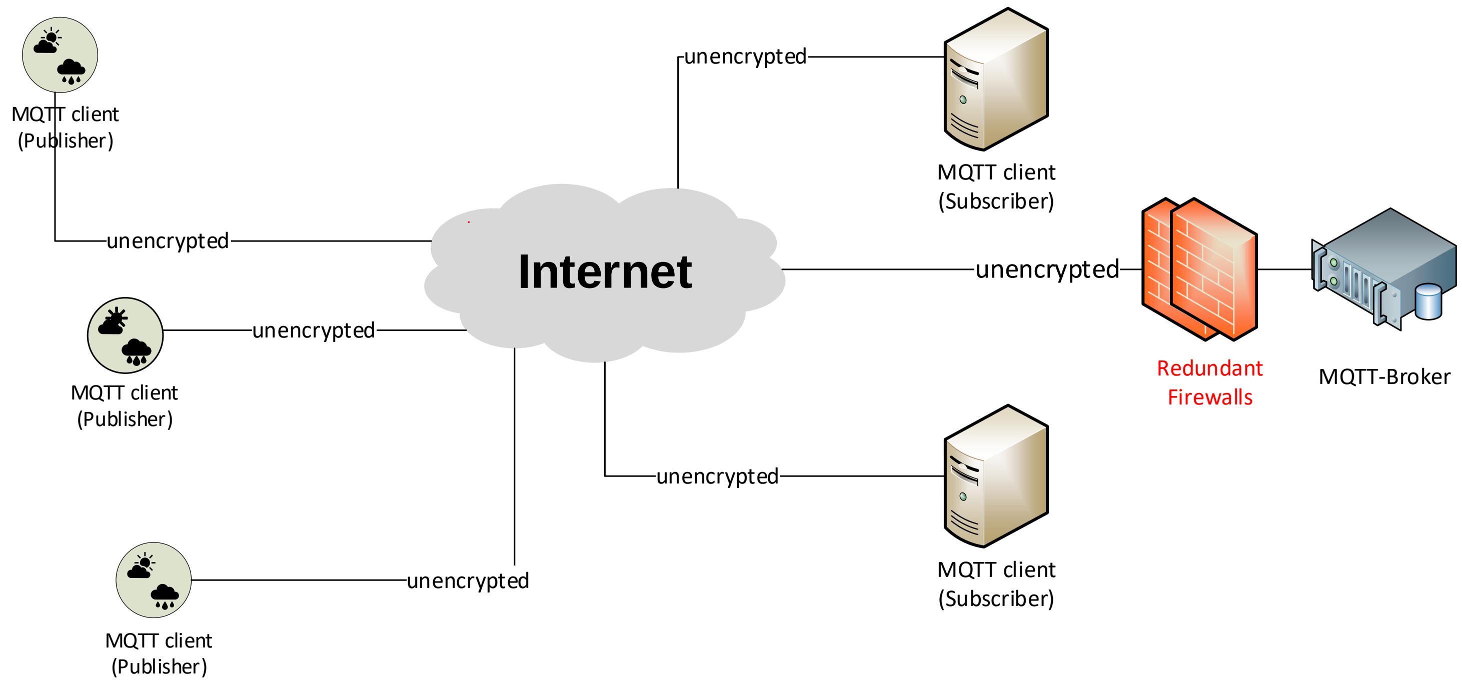


Authentication on the Application Level.

If no encryption is provided from the network, then the username / password credentials are transmitted in clear text.

MQTT - Authentication Methods (2)

- MQTT protocol provides Authentication with Client Identifier

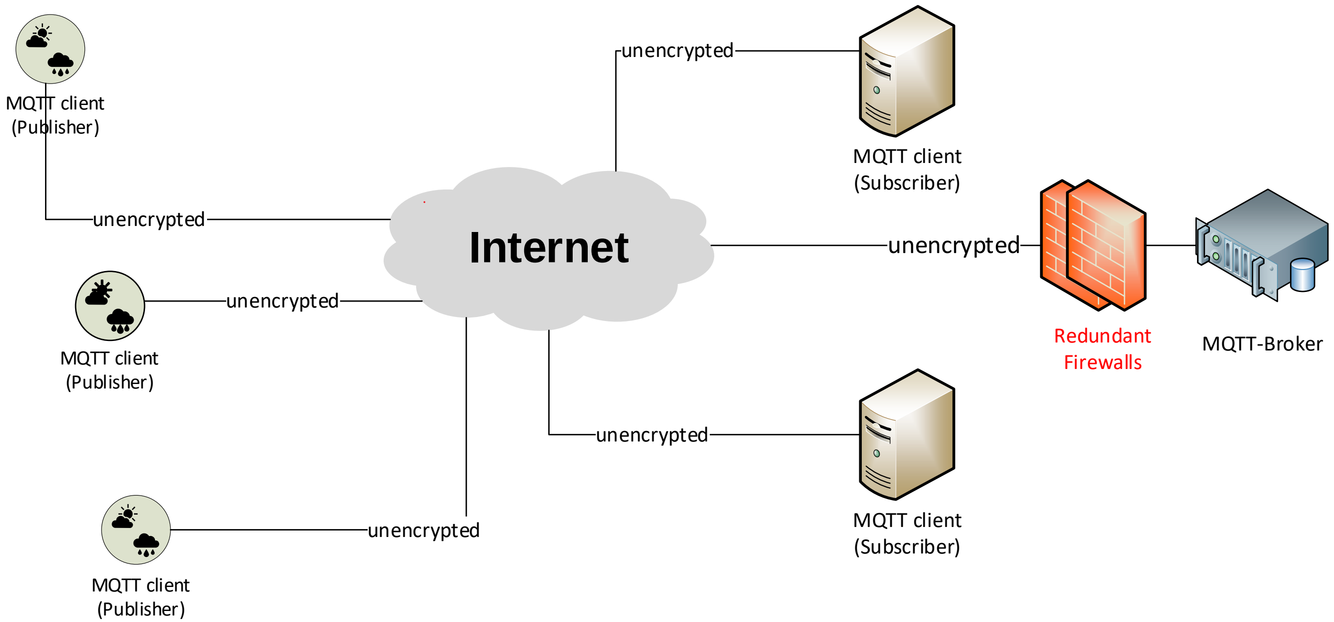


Authentication on the Application Level.

If no encryption is provided from the network, then the Client Identifier is transmitted in clear text.

MQTT - Authentication Methods (3)

- MQTT Authentication with Username / Password and Client Identifier



Authentication on the Application Level.

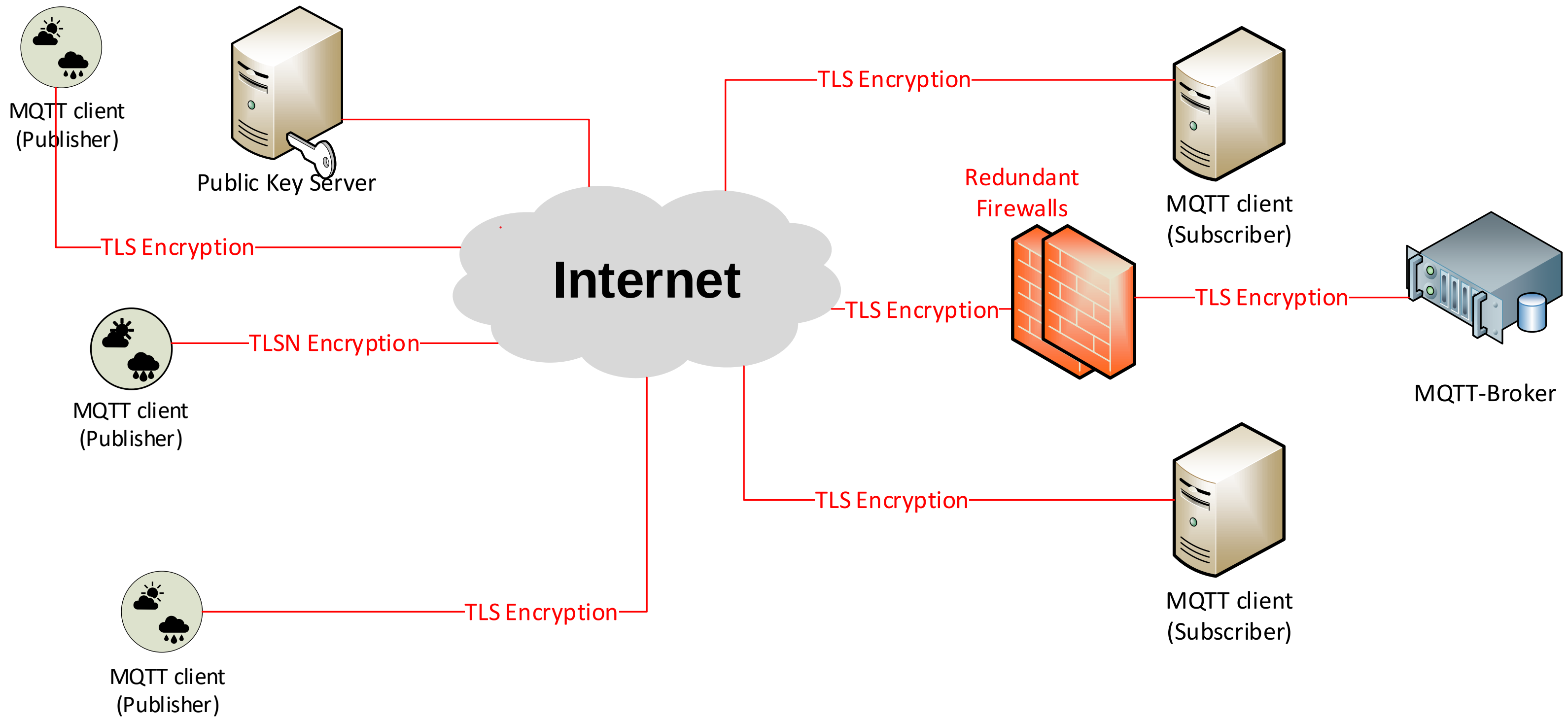
If in the authentication process the Client IDs is used in combination with the username and password, the MQTT broker is able to validate if the username/password and the client ID credential combination is correct.

If no encryption is provided from the network, then the username / password credentials and Client Identifier are transmitted in clear text.

Without encryption, Authentication with Username / Password and Client Identifier is not recommended because of privacy reasons.

MQTT - Authentication Methods (4)

- MQTT protocol provides Authentication with X.509 Certificate



X.509 is a digital certificate that uses a public key infrastructure to verify that a public key belongs to the client.

In order for the client to utilize X.509 authentication, TLS (Transport Layer Security) must be implemented as encryption method.

MQTT - Encryption Methods (1)

- Symmetric Encryption

  - In conventional symmetric encryption only a single key is used.
  - With this single key, the sender can encrypt a message.
  - With this single key, the recipient can decrypt the message.

Best practice is to assign each MQTT Topic its own key (password) so that if a password for a topic is lost, messages on other topics can’t get decrypted by attackers.

MQTT - Encryption Methods (2)

- Asymmetric Encryption

  - In asymmetric encryption, the encryption key and the decryption key are different.
  - One key is the public key which is used from the sender to encrypt the message.
  - The other key is the private key by which a recipient can decrypt the message.
  - Once data is encrypted with the public key, it is not possible to retrieve the original message with the public key. Only the private key can decrypt the data.

The best practice for that approach is that each MQTT Topic gets its own public / private key pair.

MQTT - Encryption Methods (3)

- TLS Encryption

TLS provides security on the Network Layer.

This security is part of the TCP/IP Protocol and not part of MQTT.

TLS security will provide an encrypted pipe down which the MQTT messages can flow. This will protect all parts of the MQTT message, and not just the message payload.

Only the new versions TLS 1.2 or TLS 1.3 should be used.

Older versions of TLS (or SSL) are not secure and should not be used any more.

MQTT - Encryption Mechanisms (1)

- MQTT Payload Encryption

- Usually used to encrypt MQTT PUBLISH packet payload or the CONNECT LWT payload
- provides security on the Application Layer
- is not defined in the MQTT specification, it is completely application specific
- all MQTT PUBLISH metadata stay intact
- No custom mechanism is needed on the broker to decrypt the data
- only the payload of the message is encrypted
- To encrypt CONNECT and SUBSCRIBE packets, a custom broker Plug-In for decryption must be impemented

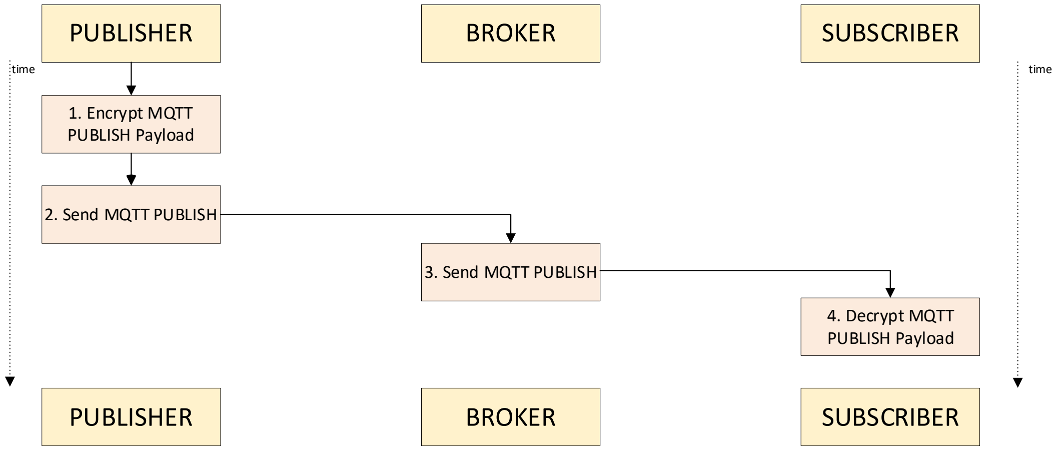
Disadvantage:

an attacker can replay the message or modify parts of the message (the topic, …).

MQTT - Encryption Mechanisms (2)

- MQTT Payload Encryption End to End (E2E)

- provides security on the Application Layer
- works without having to configure the MQTT Broker
- can protect the MQTT Payload (Application Data)
- MQTT broker can use the unencrypted packet metadata for routing and quality of service handling
- only trusted clients have access to the decryption key and are able to read the data.



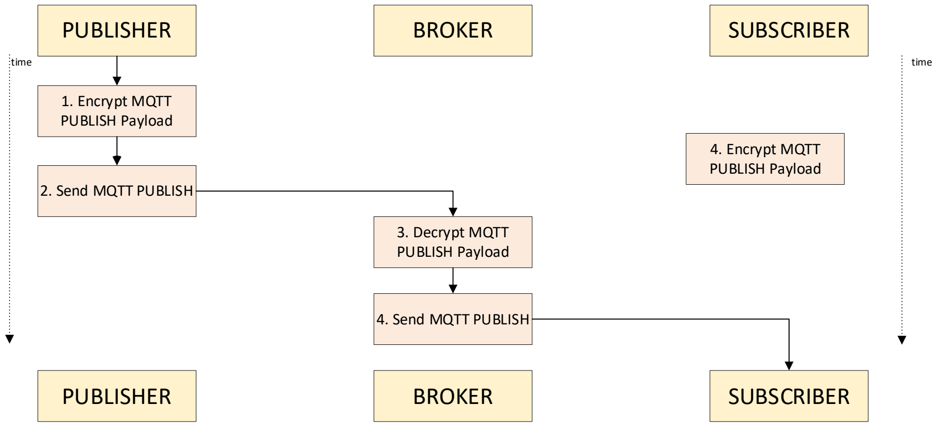
Disadvantage:

this type of encryption doesn’t protect usernames and passwords on the connection itself

MQTT - Encryption Mechanisms (3)

- MQTT Payload Encryption Client to Broker

- encrypt the MQTT Payload (Application Data) in the communication between the MQTT Publisher and the MQTT Broker
- MQTT Broker will decrypt the message received from the MQTT Publisher before distributing them to the MQTT clients
- all MQTT subscribers will receive the unencrypted message
- environment between MQTT Broker and MQTT Subscribers must be inside a secured area (because data are not encrypted)



<u>Disadvantage:</u>

this type of encryption doesn’t protect usernames and passwords on the connection itself

MQTT - Data Integrity (1)

- Data Integrity Check:

- It's important to check integrity of messages from untrusted MQTT clients
- It's important to check integrity of messages from MQTT clients that are not under your control
- Data Integrity Checks let us ensure that nobody did modify any content or the MQTT messages

MQTT PUBLISH packets can contain for Integrity Check:

- Digital Signature
- Message Authentication Code (MAC)
- Checksum

MQTT - Data Integrity (2)

Data Integrity Check verify the contents of the MQTT PUBLISH packet.

- A stamp is calculated of the contents of the packet.
- The calculated stamp is typically added to the payload.
- The receiver of the packet can verify the integrity of the data by recalculating the stamp
- If the recalculated stamp is the same like the stamp inside the packet then the message was not tampered.

Typically the stamp is calculated with the following data:

- The MQTT PUBLISH payload
- The topic of the MQTT PUBLISH packet

MQTT - Data Integrity (3)

Mechanisms for creating and validating a stamp:

- Checksum / Hash Algorithms (MD5, CRC, SHA-2, SHA-3…)
- Message Authentication Code (HMAC, CBC-MAC,)
- Digital signatures

	Checksum	MAC	Digital Signature
Data Integrity	☑	☑	☑
Authentication	⊘	☑	☑
Non-Repudation	⊘	⊘	☑
Key	none	symmetric	asymmetric

MQTT - Data Integrity (4)

- Using Checksums for MQTT Messages

- Checksum algorithms are typically very fast, even for large MQTT packets
- Don’t use Checksums without TLS because Checksums can get altered with the MQTT packet.

Disadvantage:

this type of encryption doesn’t protect usernames and passwords on the connection itself

Best Practice:

use Checksums only with TLS or use a MAC algorithm or digital signatures instead.

MQTT - Data Integrity (5)

- Using MACs for MQTT Messages

- MAC can be used for Data Integrity and for Authentication check
- Message Authentication Code Algorithms (for example, HMAC) are typically very fast compared to digital signatures
- The shared secret key must be exchanged securely prior to the MQTT communication
- HMACs can be used securely even if TLS is not deployed

Disadvantage:

all clients who are aware of the secret key can sign and verify because the same key is involved in both processes

MAC … Message Authentication Code

MQTT - Data Integrity (6)

- Using Digital Signatures for MQTT Messages

- Digital Signature can be used for Data Integrity, Authentication check and non-repudiation
- Digital signatures use public / private key cryptography
- Provisioning and Revocation of public / private keys adds complexity to the system
- In a MQTT-Publish / MQTT-Subscribe system, the receiver of a message is usually not aware of the identity of the sender
- <u>Only if</u> there is a guarantee that only <u>a specific client can publish to a specific topic</u> (i.e. through authorization mechanisms) then digital signatures can be a good fit

MQTT - Data Integrity (7)

- Using Data Integrity Check and Encryption for MQTT Messages

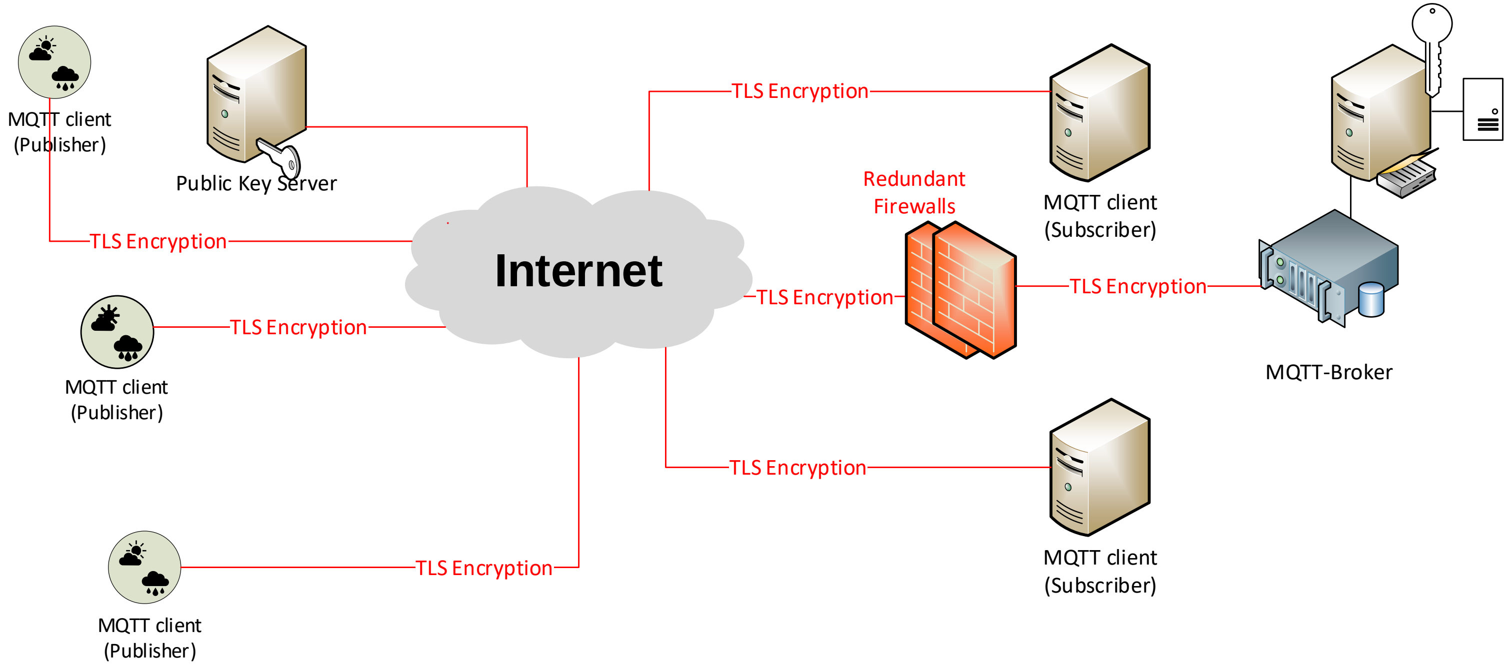
- Data integrity checks with stamps are a good addition to message encryption
- Even if the message is decrypted and encrypted it after modification - the integrity check still fails if the message was altered
- If MQTT is used without TLS, a data integrity check will enhance the security

MQTT - Data Integrity (8)

- Using Data Integrity Check and TLS for MQTT Messages

- If TLS is deployed and proper authentication and authorization mechanisms are in place, data integrity checks will not add much additional security, only the Non-Repudiation attribute may add additional security
- Make sense if dealing with sensitive data and command PUBLISH packets

MQTT - Identity Access Management
with X.509 certs for both Clients and Servers



After mutual authentication, the broker will authorize the MQTT clients only after it get green light for that option (Topic, …) from the Identity Access Management Server – the administrators have to configure the allowed actions for the clients in the IAM system.

X.509 is a digital certificate that uses a public key infrastructure to verify that a public key belongs to the client.

In order for the client to utilize X.509 authentication, TLS (Transport Layer Security) must be implemented as encryption method.

Delay from TLS Handshake Protocol

- Transport Layer Security (TLS) version 1.2

- RFC 5264

- Transport Layer Security (TLS) version 1.3

- RFC 8466
- TLS 1.3 is faster then TLS 1.2 (initial handshake has been shortened)
- All the at-risk crypto algorthms from TLS 1.2 have been removed, including SHA-1, RC4, DES, 3DES, AES-CBC, MD5, …

Delay from TLS Handshake Protocol

- Transport Layer Security (TLS) version 1.2

- 0ms: Client Connection Request
- 34ms: Server Connection Acknowledged
- 68ms: Client Hello
- 102ms: Server Hello – Certificate – Server Hello Done
- 136ms: Client Key Exchange – Change Cipher Spec – Client Finished
- 170ms: Server Change Cipher Spec – Server Finished
- 204ms: Client sends Application Data
- 238ms: Server receives Application Data

Delay from TLS Handshake Protocol

- Transport Layer Security (TLS) version 3

- 0ms: Client Connection Request
- 34ms: Server Connection Acknowledged
- 68ms: Client Hello – Client Key Share
- 102ms: Server Hello – Server key Share - Certificate – Server Finished
- 136ms: Client Finished – Client sends Application Data
- 170ms: Server receives Application Data

Thanks!

E. Wegscheider
SEC Consult
e.wegscheider@sec-consult.com

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825050-5GCroCo