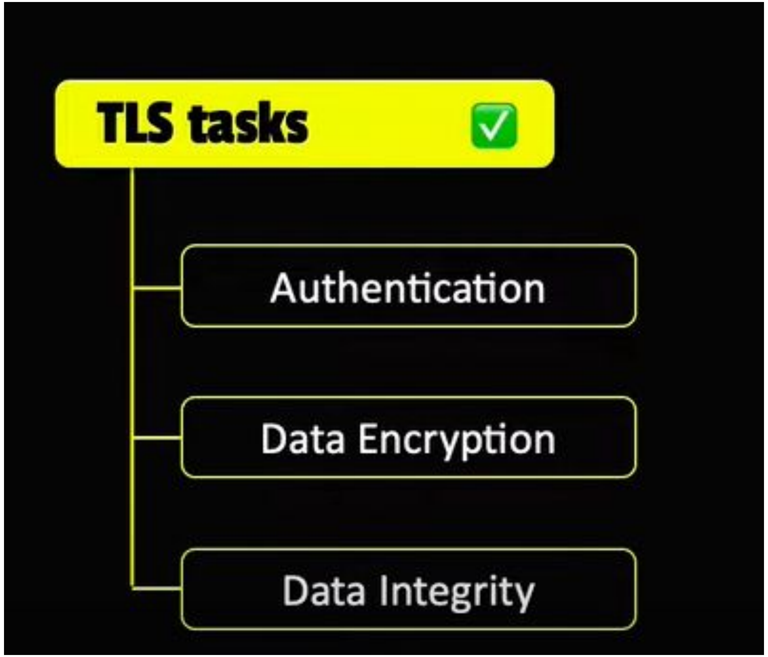
TLS

TLS (Transport Layer Security) is a cryptographic protocol that is designed for secure communication over a computer network.



TLS tasks ✅

- Authentication
- Data Encryption
- Data Integrity

TLS Authentication

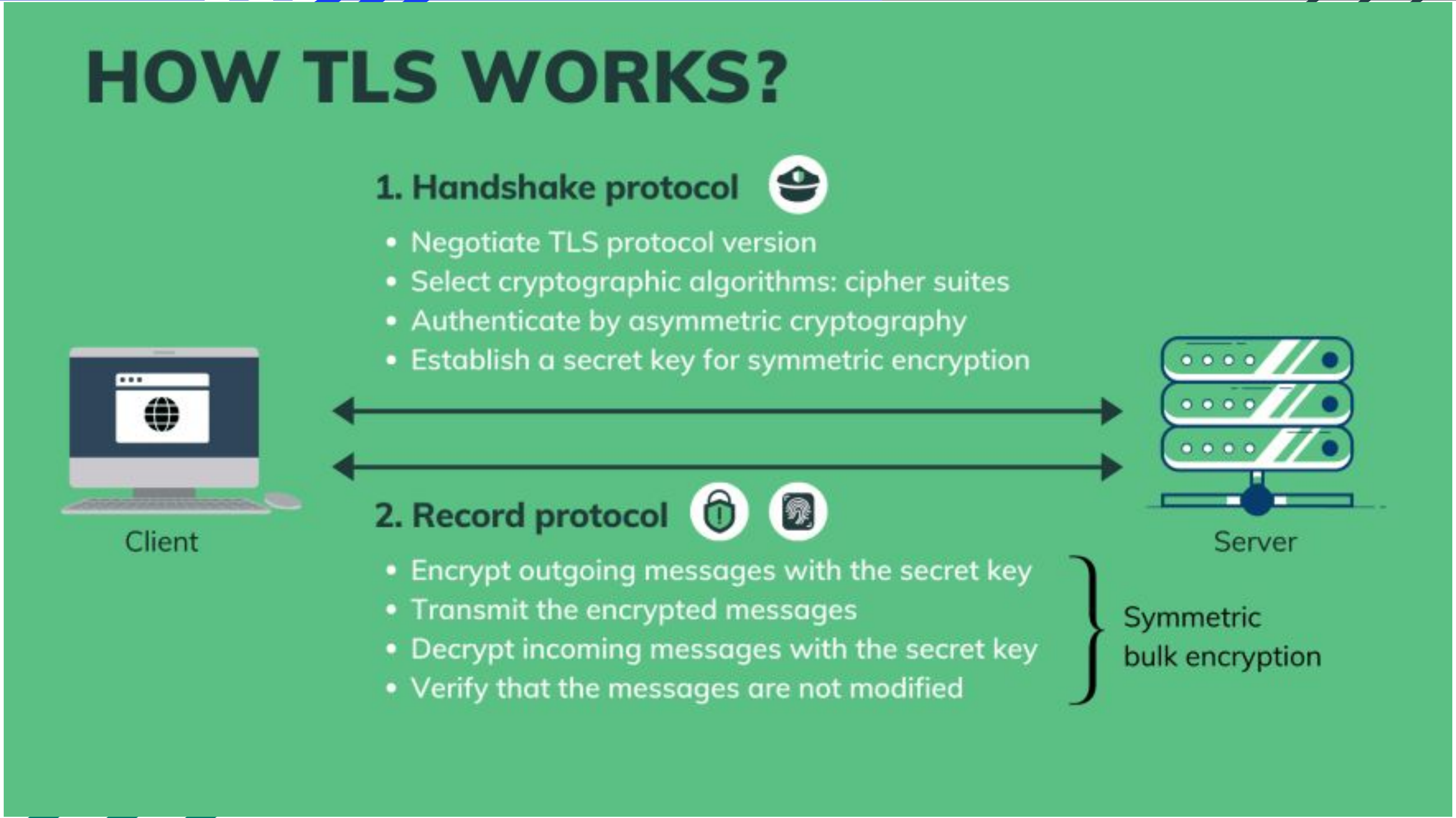
- The client must confirm the server's identity before connecting. The client verifies that the server's certificate and public key are valid and were issued by a certificate authority (CA) listed in the client's list of trusted CAs
- The client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.

TLS Encryption

1. The client contacts the server using a secure URL (HTTPS…).
2. The server sends the client its certificate and public key.
3. The client verifies this with a Trusted Root Certification Authority to ensure the certificate is legitimate.
4. The client and server negotiate the strongest type of encryption that each can support.
5. The client encrypts a session (secret) key with the server’s public key, and sends it back to the server.
6. The server decrypts the client communication with its private key, and the session is established.
7. The session key (symmetric encryption) is now used to encrypt and decrypt data transmitted between the client and server.

Integrity

- Ensures that the server receives the same message that the client has sent, and vice versa. It detects any *loss*, *deliberate modification*, or *tampering* of messages during data exchange, and thereby supports message integrity and authenticity.
- TLS recognizes any alteration of data during transmission by checking the message authentication code



HOW TLS WORKS?

1. Handshake protocol

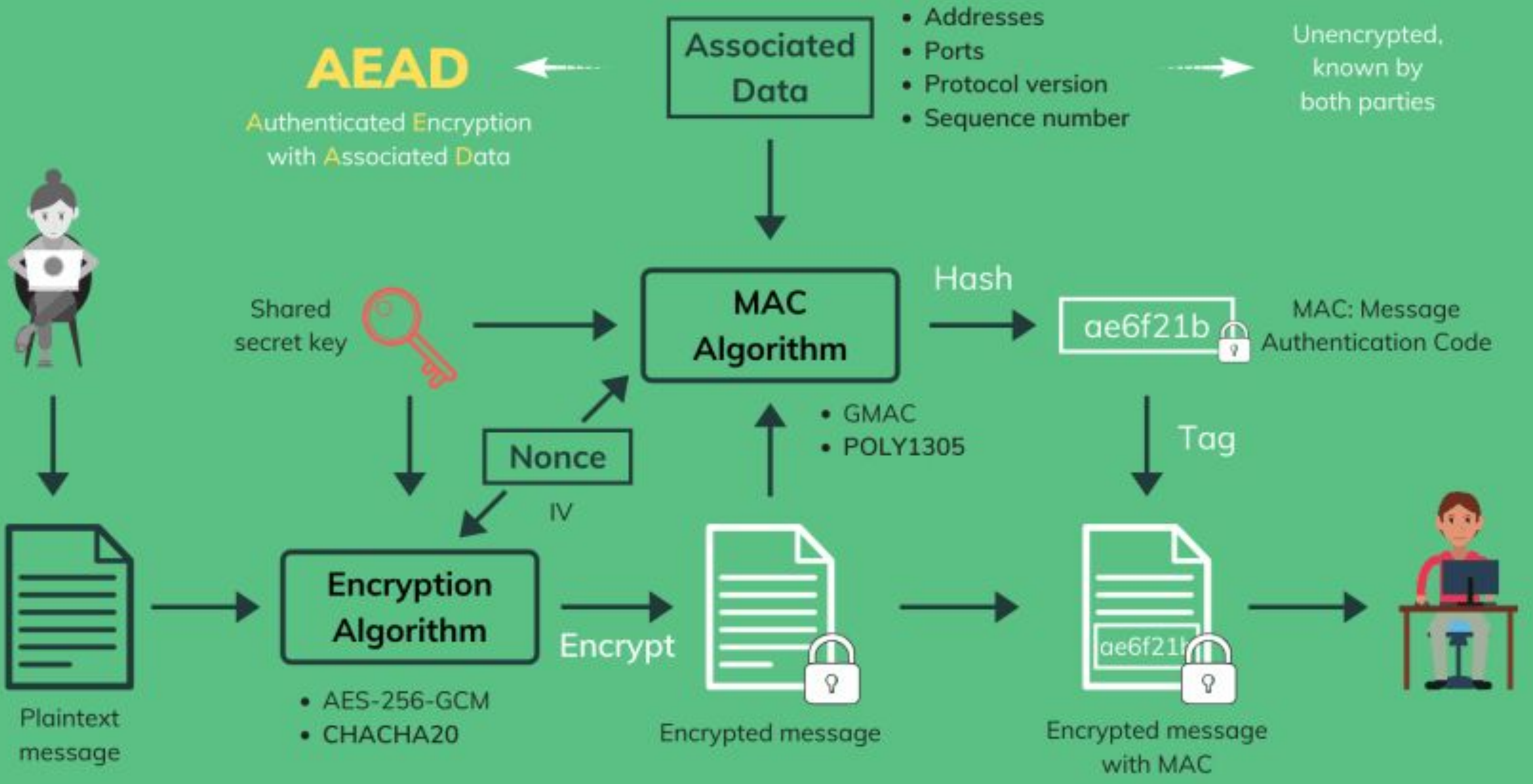
- Negotiate TLS protocol version
- Select cryptographic algorithms: cipher suites
- Authenticate by asymmetric cryptography
- Establish a secret key for symmetric encryption

2. Record protocol

- Encrypt outgoing messages with the secret key
- Transmit the encrypted messages
- Decrypt incoming messages with the secret key
- Verify that the messages are not modified

Symmetric bulk encryption

AUTHENTICATED ENCRYPTION



AEAD

Authenticated Encryption with Associated Data

Associated Data

- Addresses
- Ports
- Protocol version
- Sequence number

Unencrypted, known by both parties

Shared secret key

MAC Algorithm

- GMAC
- POLY1305

MAC: Message Authentication Code

Nonce

Encryption Algorithm

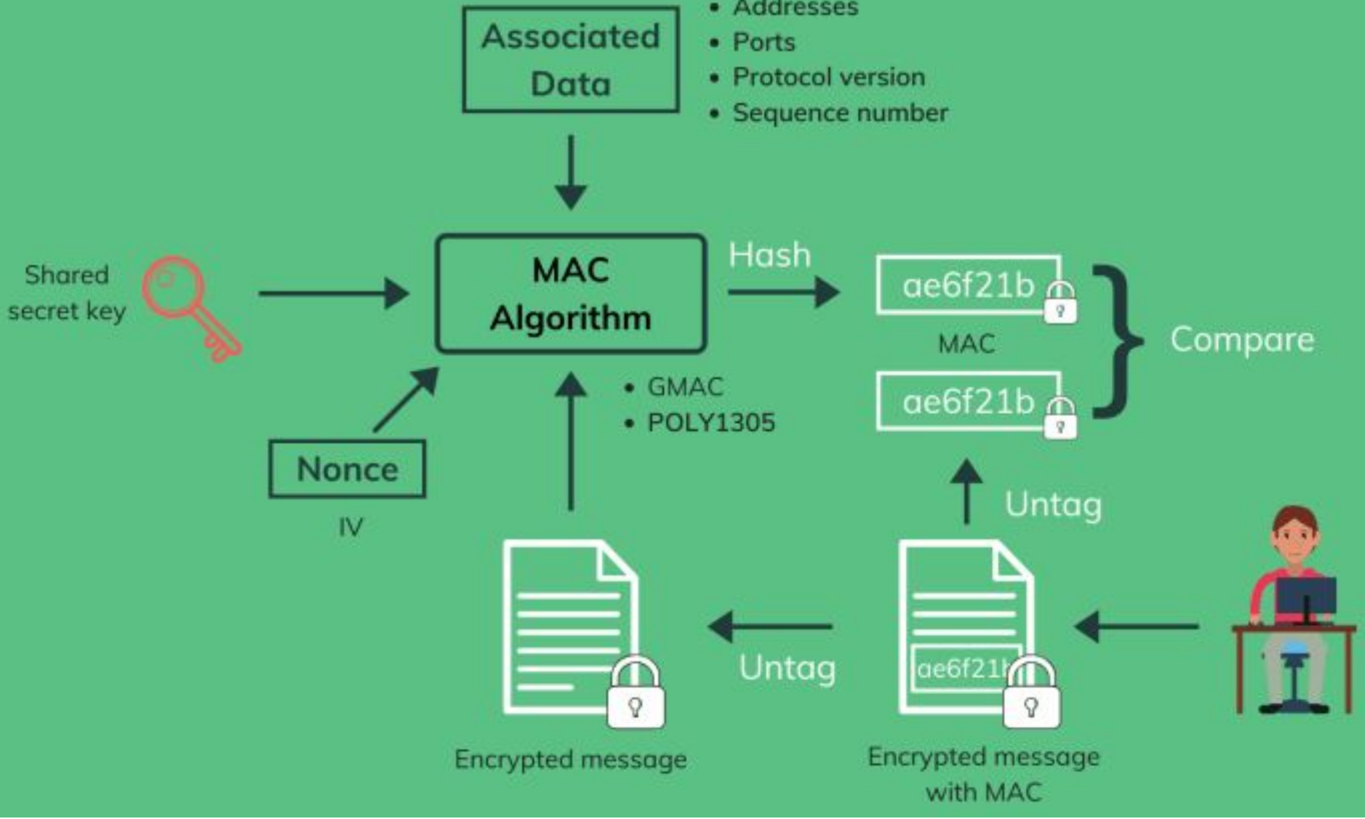
- AES-256-GCM
- CHACHA20

Plaintext message

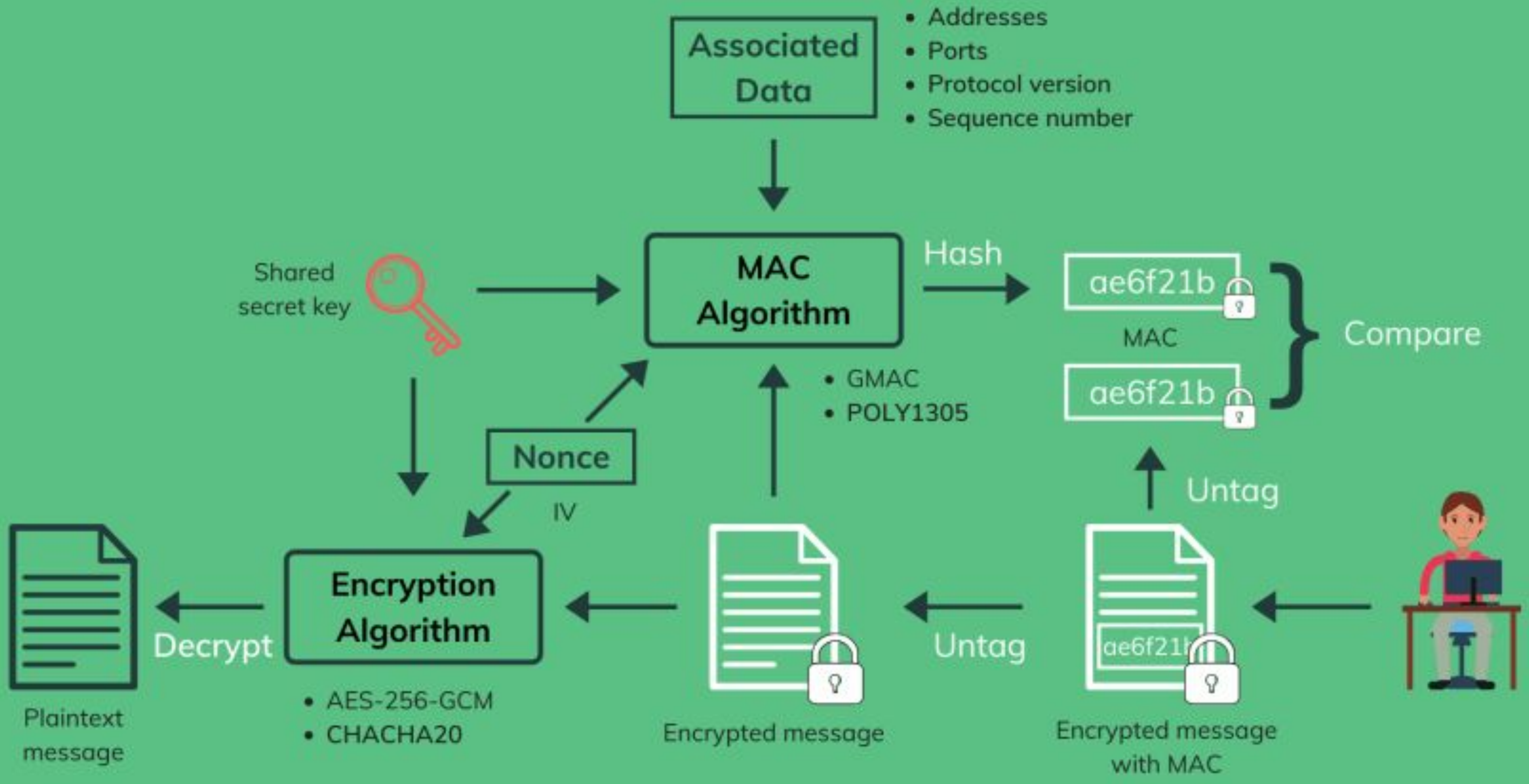
Encrypted message

Encrypted message with MAC

DECRYPTION + VERIFICATION



DECRYPTION + VERIFICATION



Associated Data

- Addresses
- Ports
- Protocol version
- Sequence number

Shared secret key

MAC Algorithm

- GMAC
- POLY1305

Hash

ae6f21b

MAC

ae6f21b

Compare

Untag

Nonce

IV

Plaintext message

Decrypt

Encryption Algorithm

- AES-256-GCM
- CHACHA20

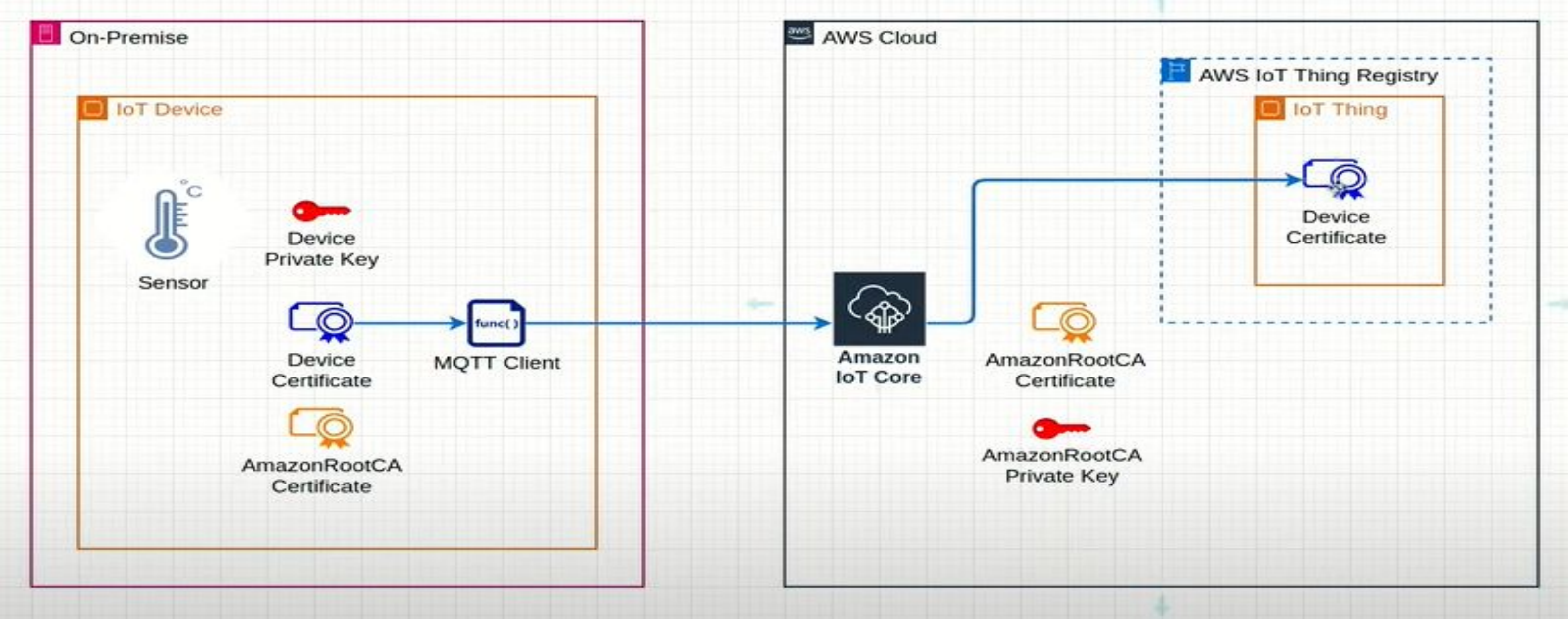
Encrypted message

Untag

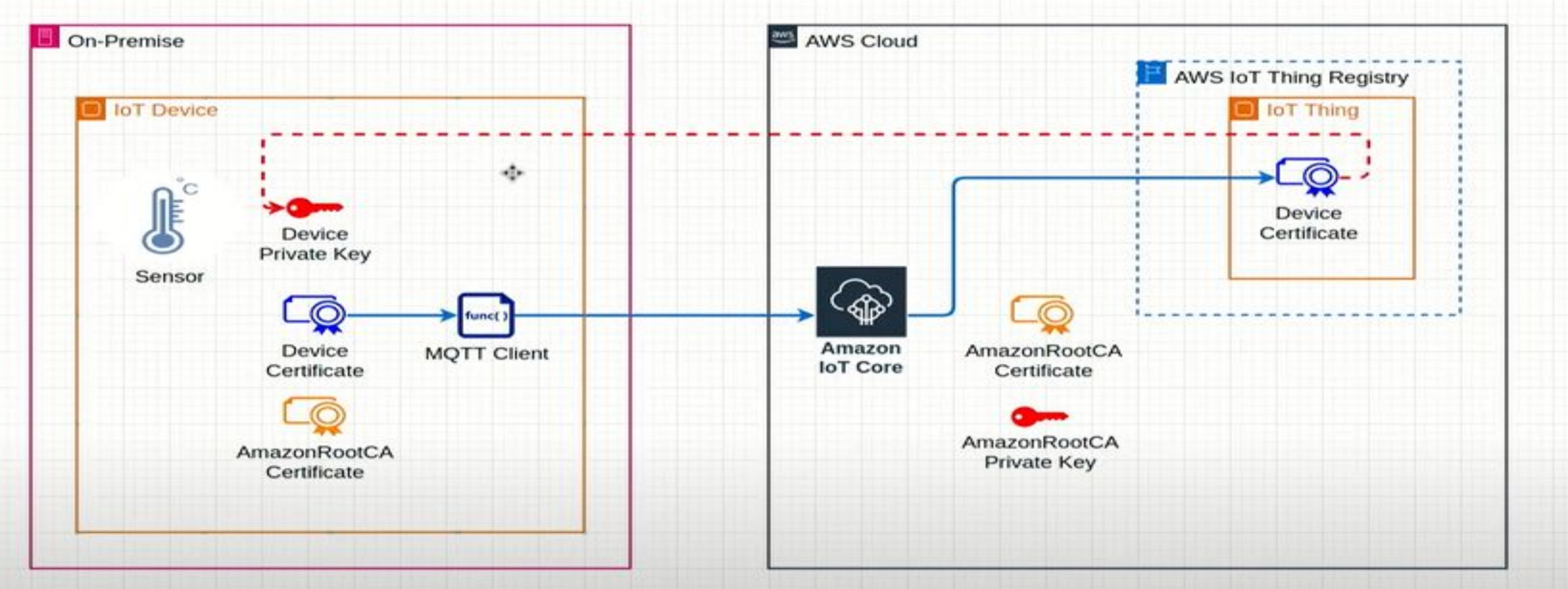
ae6f21b

Encrypted message with MAC

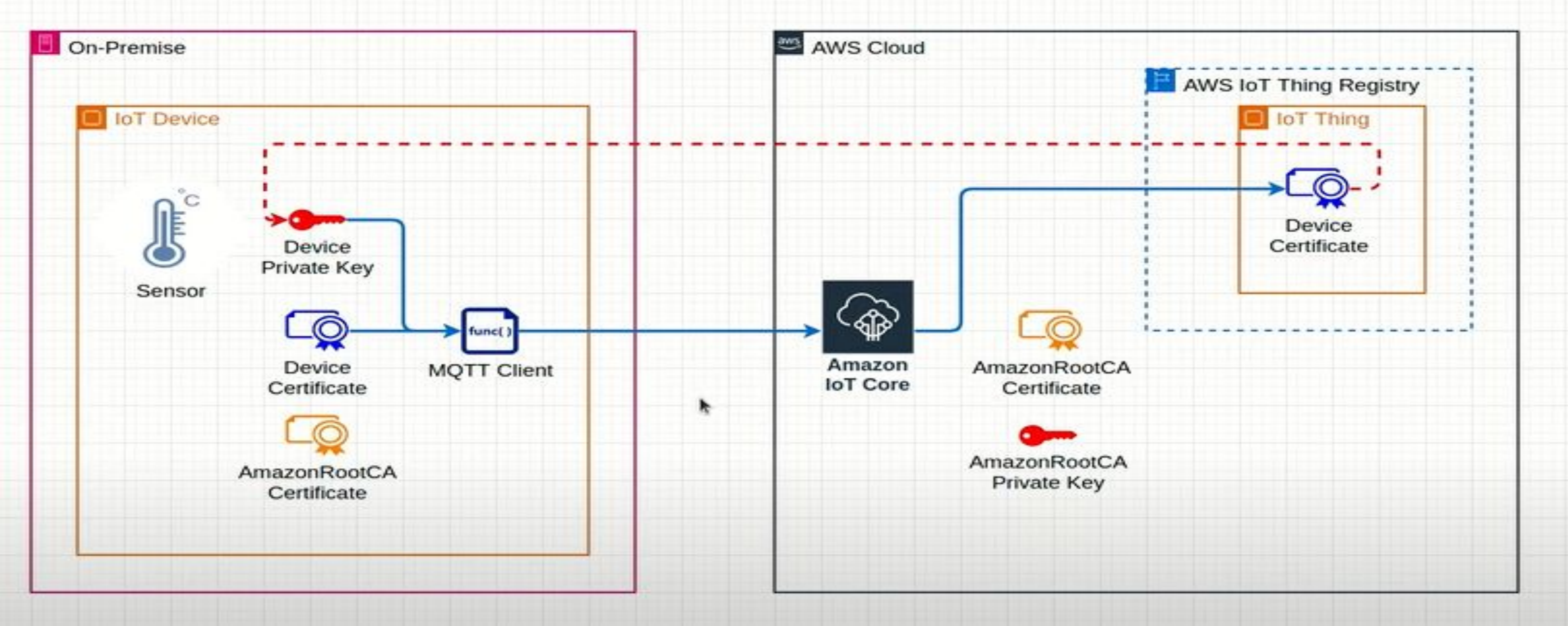
The iot device presents the device certificate, and the AWS IOT core verifies is the register in the Things. (policy attached and connection is active).



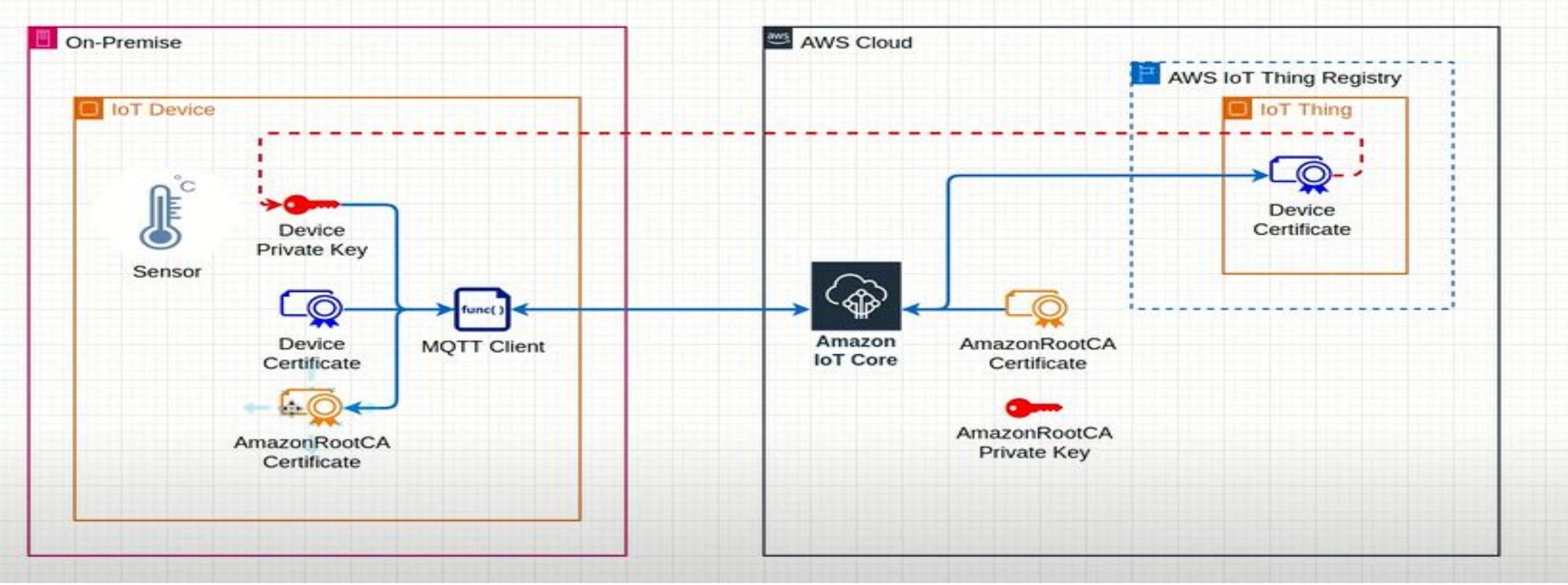
The device certificate interacts with device private key, the private key is stored in the device.



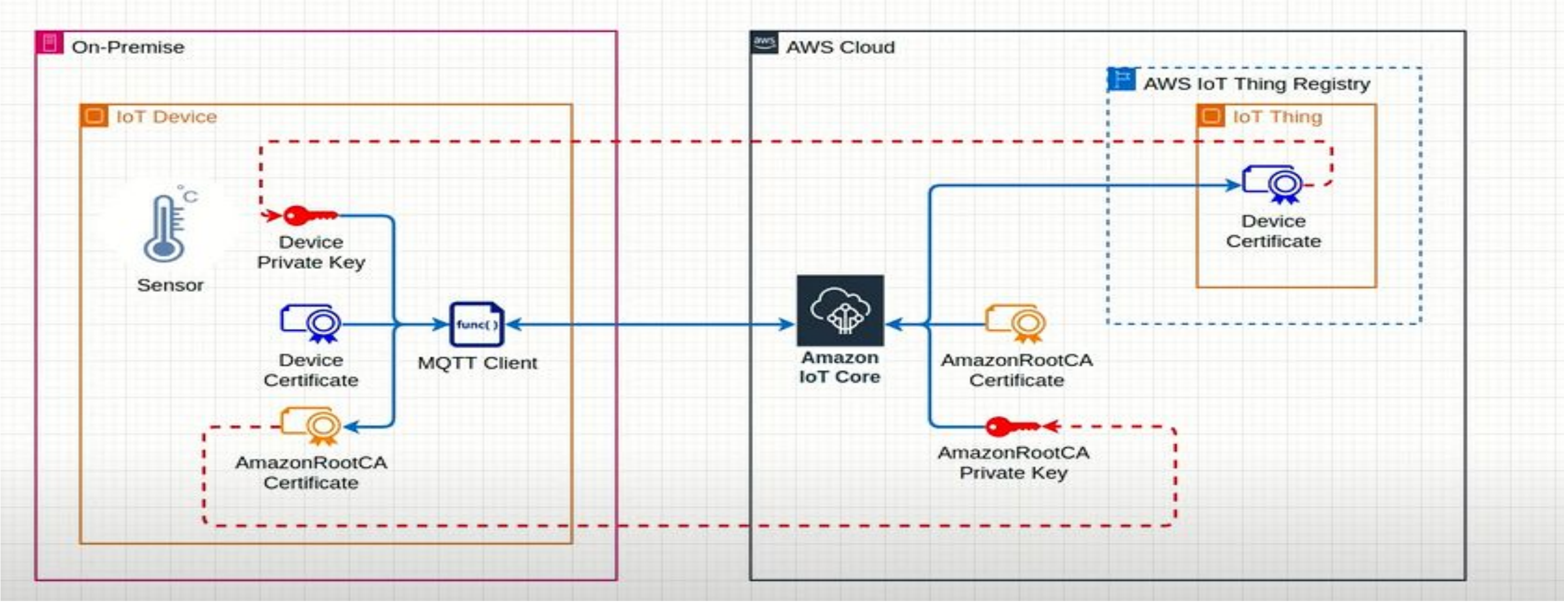
The device private key and certificate used has a proof to identify the IOT Device.



The aws iot core presents the certificate, the device, the device verify wether the certificate is valid, so that it ensures the communication is valid, (Aws iot endpoints should be valid)



The amazonroot certificate corresponds with the private key to check wether its secure. And both establish trust and confirms the identity and starts,


On-Premise
IoT Device
Sensor
Device Private Key
Device Certificate
MQTT Client
func( )
AmazonRootCA Certificate
AWS Cloud
AWS IoT Thing Registry
IoT Thing
Device Certificate
Amazon IoT Core
AmazonRootCA Certificate
AmazonRootCA Private Key