Object Security for IoT

Karan Luthra

September 2021

1 Introduction

The objective of this report is to demonstrate the implementation of Object Security on the Internet of Things. This report will provide a proof of concept of how such security measures can be implemented.

A secure communication will be established between the client and server to enable confidential, integrity, and replay protection for constrained IoT devices.

2 Architectual design

Although TCP is more robust and reliable than UDP, it has more overhead. For instance, a TCP packet has a minimum size of 20 bytes up to a maximum of 60 bytes and a UDP packet consists of only 8 bytes for each packet. For small IoT devices, the choice of packet has a great effect on device performance, especially battery-powered devices and thus UDP is a wise choice.

In terms of security, the Elliptic Curve Diffie-Hellman Key Exchange algorithm was preferred due to its smaller key size in comparison to e.g., RSA. Also, generating keys with the algorithm requires far less power than RSA. These benefits are highly desirable for constrained IoT devices. The Python library *cryptography* is equipped with ECDH protocol. The one used in the report is the 384-bit prime field Weierstrass curve (SECP384R1).

If a private key happens to get leaked and the attacker has previously saved communication sessions, then the attacker can use the leaked private key to decrypt the saved data. It ensures that pas session keys will not be compromised even if the private key is compromised. To mitigate the issue, communication between objects and hosts should implement *perfect forward secrecy*. This is can be achieved by using ephemeral shared keys. Shared keys are regenerated every new session, so the key generation process must be time-efficient. Luckily, ECDH provides fast key generation.

Publicly shared keys in an insecure channel via UDP are serialized and converted into bytes. *X962* encodes the keys into binary format and compresses it with *CompressPoint*. User keys are authenticated through derivation with HKDF which uses SHA-256. The keys are derived with HKDF based on HMAC using SHA-256

Once the initial handshake with key exchange is complete, the entities can begin sending data among themselves.

Instead of encrypting the data as it is sent, encryption can be done before transmission. Then the data is considered to be an object. To achieve *object security* transmitted data is encrypted with AES with Cipher Feedback (CFB) using the PyCrypt library. Advantages are that encrypted objects can be cached in the network to increase performance; low session maintenance. Drawbacks are increased overhead and less able to integrate with other sources.

2.1 Handshake and transmission steps

Client and server running key generation and exchange:

1. Server initalizes and listens for client
2. Client sends a request to server
3. Server initalizes handshake with client
4. Client/Server generate a private key using ECDH algorithm

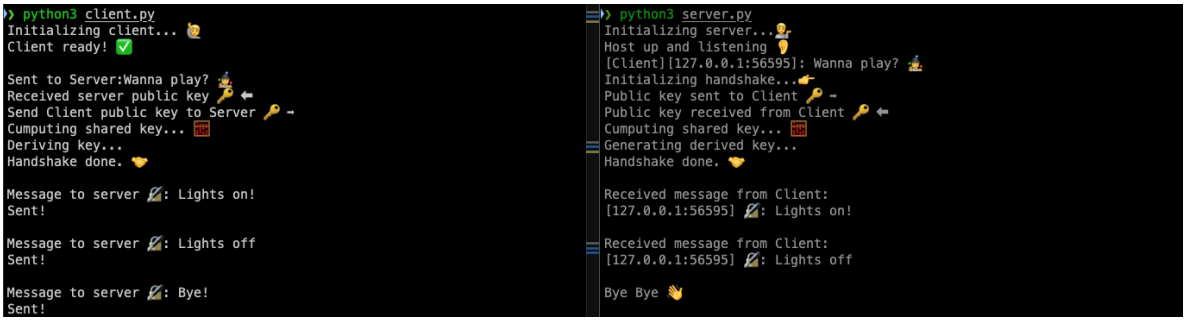

Figure 1: Client (left) Server (right) handshake print log.

5. Client/Server generate public key from the previously generated private key
6. Public keys are sent to one an other
7. A shared key is computed from the recieved public key and its own private key
8. The shared key is derived with HKDF
9. Handshake sequence is done

Transmission of data:

1. Data is encrypted with AES with default block size of 16 bytes
2. Data packets have a maximum size of 64 bytes
3. Encrypted data is encoded to base64 and sent over UDP
4. Received data is decoded and then decrypted with derived key
5. Data is then consumed by the party.

For detailed overview, please see Appendix.

3 Evaluation

Using the present proof-of-concept, it is possible to achieve integrity, confidentiality and theoretically be protected against replay attacks with IoT technologies. Nevertheless, no replay attacks were tested. The design does not take into account intermediate caching. The implementation of caching would further demonstrate the benefits of object security by simulating a man-in-the-middle attack. Only authorized users can access the data sent, even if the sent objects are temporarily captured by a third party. A caching implementation would demonstrate object security's robustness.

4 Conclusion

Object Security principles are applied to IoT devices in this report as a proof of concept. Devices with hardware constraints can communicate in a secure and lightweight manner. The technology seems promising, despite not being subjected to rigorous testing during the initial examination. In the opinion of the senior expert, object security should be tested on company IoT products and adopted in the next security patch release.