CS 535 Wireless Network Security

Fall 2018-2019

A Secure Design for Client & IoT Communications

Abstract

Key management and secure key distribution is always a challenge wireless network. This project proposes an authentication and authorization protocol with a secure key management scheme which uses two hash chains to provide forward and backward secrecy during the communication between client and IoT devices in smart home networks. In addition, using hash chains to create session key provide lightweight security which can be easily implemented and used without any overhead on IoT devices.

1. Introduction

IoT is one of the hottest topics in both the industry and academia. There are many current applications of IoT's and in the near future, this technology will be everywhere from factories to houses. Therefore, IoT systems must be secure enough to prevent widespread breaches since the consequences might be catastrophic. In this paper, we propose an IoT system where we imagine a smart house. IoT devices (tv, lights, thermostat etc.) are controlled by a gateway. The owner of the house can access these devices through the gateway by using a commercial client application. First of all, the gateway needs to authenticate any device that enters this network by communication with a trusted third party which will be the authentication server. Then clients must authorize themselves in order to access IoT devices. Authentication and Authorization allow the system to establish and dictate permissions.

2. System Design & Communication Protocols

The proposed structure contains four different entities representing; client, an IoT device, gateway, and authentication server. A client can be a desktop, web or mobile application which first enters into a network authenticates itself and requests for access to IoT devices in the

network for some data exchange. IoT devices are data transferring entities which are all connected to the gateway and are authenticated when they first joined to the network. Authentication Server is responsible for authenticating the IoT devices and clients with a challenge-response protocol. After authentication, clients can requests authorization for IoT's from AS (Authentication Server) where authorizations for each client or client type is stored.

a. Authentication Protocol

i. *C —> G: authenticationReq | CID*

ii. *G —> AS: authenticationReq | CID*

iii. *AS generates a nonce, AS —> G: nonce | HMAC(nonce)*

iv. *G checks whether the HMAC is correct, G —> C: nonce*

v. *C —> G: E(nonce | CID, H(P))*

vi. *G —> AS: E(nonce | CID, H(P))*

vii. *AS checks whether the decryption is successful,*
if yes, AS —> G: E(S1 | S2 , H(P)) | E(S1 | S2 | CID, GK)

viii. *G —> C: E(S1 | S2 , H(P))*

ix. *G generates hash chains for CID.*

x. *C generates hash chains.*

b. Authorization Protocol and Key Agreement

i. *C —> G: E(authorizationReq | IOTID, SK) | HMAC(E(authorizationReq | IOTID, SK))*

ii. *G —> AS: E(CID | IOTID, GK) | HMAC(E(CID | IOTID, GK))*

iii. *AS —> G: E(OK | CID | IOTID, GK) | HMAC(E(OK | CID | IOTID, GK))*

iv. *G checks wether the HMAC is correct, G —> C: E(OK | IOTID, SK)*

3. Performance Analysis

Performance tests are run on two different devices whether to see how much time elapsed cryptographic operations on each device. Since the speed of communication process highly depends on connection speed varying on different locations, measuring

cryptographic operation indicates more reliable results. During these experiments, Client and IoT devices have the i5-6360U and Cortex-A53 (ARMv8) processors respectively.

Cryptographic Operation	***Time Elapsed (s) in Client***	***Time Elapsed (s) in IOT device***
Initial hash chain generation	*0.0003*	*0.0025 - 0.0030*
Successive hash chain generations	*0.0004*	*NA*
Decrypting messages with AES (128-bit)	*0.0001*	*0.0019*
Encrypting messages with AES (128-bit)	*0.0017-0.0020*	*0.0041*
Encrypting messages with AES (128-bit) using the key derived from the hash chain	*0.0002 - 0.0003*	*NA*
Calculating HMAC of a message	*0.0008 - 0.0009*	*NA*
Generating hash of a password	*0.0000*	*0.0014*
Rekey operation	*0.0001*	*0.0006*

Table 3.1.1: Performance tests of the protocol on both client and IoT devices

According to Table 3.1.1, while both device approximately the same amount of time during rekey operations, it takes twice or more time to encrypt and decrypt messages with AES in IoT device since it has a relatively slower processor on its board in comparison with the client device. This difference gets bigger while devices are generating hash chain or calculating a hash value of the password by using the SHA-256 algorithm. The reason is that cryptographic operations such as calculating a hash of the message or encrypt/decrypt operations with AES

algorithm are CPU-bound operations. The speed of processor used in device highly affect time elapsed in these operations. Although a slowdown is observed while running the same protocol on the IoT device, time spent on calculations is still measured in milliseconds. Even though the slowdown on IoT devices wouldn't affect the communication between entities, the overhead should be reduced in further studies.

4. Conclusion and Future Work

Consequently, we suggest a scheme which handles key management and authorization processes by using a hash chain. Since hash chains are one-way functions, we ensure that backward and forward secrecy is guaranteed by the key generated from hash chains [2]. On other hand, there are also still available research areas in the proposed scheme. Firstly, gateway devices cannot differentiate between types of client and IoT devices connected to the network. Under these circumstances, any client can be authorized to retrieve data from other clients since the gateway cannot recognize types of devices in the authorization phase. Secondly, although we provided a key agreement scheme to provide key establishment between gateway to IoT devices and gateway to client devices, there is no data sharing protocol between client and IoT devices. Thirdly, in any case of short disconnections of devices from the network due to low signaling in extreme conditions, client devices cannot synchronize their shared keys generated from the common hash chain.

References

[1] HMAC: Keyed-Hashing for Message Authentication. (n.d.). Retrieved from https://tools.ietf.org/html/rfc21

[2] Lamport, L. (1981). Password authentication with insecure communication. *Communications of the ACM,24*(11), 770-772. doi:10.1145/358790.358797