MAISTS: Mutual Authenticated IoT-to-Smartphone communication with a Trusted Server

Philip Oguchi
School of Computing
University of Nebraska–Lincoln
eoguchi2@huskers.unl.edu

Uyen Tran
School of Computing
University of Nebraska–Lincoln
ktran21@huskers.unl.edu

***Abstract*—Secure and effective authentication mechanisms have grown more crucial as IoT and smartphone device usage has increased. This paper presents a simple, lightweight authentication technique that guards against man-in-the-middle and replay attacks. We use symmetric key and asymmetric key encryption and decryption. We formally analyze our protocol's correctness and robustness using Proverif. Our investigation also shows that our protocol has excellent performance and complexity and guarantees message integrity and confidentiality.**

***Index Terms*—Mutual authentication, IoT security, hash functions, message integrity, confidentiality.**

I. Introduction

The Internet of Things has changed our communication with our environment and physical devices. These devices can be smartphones, cameras, soil sensors, or others. As the number of IoT devices grows, security becomes a significant challenge to ensure the protection of sensitive shared data shared and an end-to-end guarantee of security. Many IoT devices need to be connected to the Internet to work, making them more vulnerable to various cyber-attacks [1]. Many commercial vendors do not consider security when building these devices, which has resulted in damaging effects like the Mirai Botnet DDOS attack on DNS infrastructure [2] and several others. Due to several constraints of IoT devices like battery, power, and memory limitations, it is challenging to implement complex security protocols on these devices, which makes them prone to attackers.

In this work, we concentrate on tackling the security issues in an IoT system where resource-constrained devices are connected to a trusted server through the internet and are controlled and monitored by a smartphone. The smartphone must also be connected to the trusted server using cryptographic protocols. The mutual authentication between an IoT device and the server, data integrity and confidentiality assurances for the data received from the IoT device to the trusted server, and mutual authentication between the smartphone user and the trusted server are some of the problems we tackled.

We propose MAISTS: **M**utual **A**uthenticated **I**oT-to-**S**martphone communication with a **T**rusted **S**erver. Our scalable approach guarantees message integrity, confidentiality, and mutual authentication for IoT devices and smartphones. MAISTS protocol is cryptographically and computationally secured against replay attacks and man-in-the-middle attacks.

We perform security analysis to evaluate our protocol for correctness and robustness under varying attack scenarios using Proverif. We also compute the performance complexity of our protocol. We establish that our protocol considers the resource constraints of the IoT device, making the implementation more efficient.

The remaining sections of this paper are as follows. Section 2 presents the preliminary background of our work. Section 3 provides the network model and assumptions. Section 4 presents the MAISTS protocol. Section 5 presents the security analysis. Section 6 provides automated security verification. Section 7 provides the performance analysis, and Section 8 concludes our paper.

II. Preliminary Background

Before diving into the network model and the proposed protocol, we briefly describe message integrity and mutual authentication in this section.

A. Mutual authentication

Two parties must authenticate each other's identities to provide access to the resources. Mutual authentication is more like a client-server authentication that performs a three-way handshake protocol to authenticate the transmitter and receiver fully. Mutual authentication provides an additional layer of security which means a client can authenticate a server, and a server can also authenticate a client. In our case, the client can be an IoT device or a smartphone, and the server is a Trusted server. Mutual authentication intends to reduce the risk of man-in-the-middle attacks, reply attacks, and other types of cyber threats. We assume that two communication parties already have pre-shared keys, the public-private key pairs. Then we implement mutual authentication [3] on them.

B. Message Integrity

During transmitting messages from the IoT device or Smartphone, ensuring that the message has not been tampered with is necessary. Therefore, we introduced a method for the receiver to have some form of guarantee that the message they received is the same as the one that was transmitted. In this paper, we use the Hash function for message integrity, which takes in an input and outputs a fixed-size message digest. The Hash function is a one-way function that makes generating the

original message infeasible. We assume that the receiver and the sender already agree on a chosen Hash algorithm.

III. Network Model And Assumptions

A. Network Model

1) Legitimate IoT Device (A): The legitimate device comprises a single lightweight IoT device, e.g., a camera or a smart bulb, that gathers sensory data and sends it to the trusted server via the internet. The legitimate device connects to a server via symmetric key cryptography. The legitimate node (device) is also scalable. Before establishing a session, the legitimate IoT device needs to bootstrap secrets with the Trusted server.

2) The Trusted Server (T): The server authenticates the message it receives from the legitimate IoT device (A) and the smartphone (P). It also coordinates the IoT and the smartphone to communicate securely. The connection to the trusted server is via the internet.

3) Smartphone (P): The smartphone can handle public key cryptography. The user can operate the IoT device with the smartphone through the trusted server via the internet.

B. Assumptions

In this work, we make the following assumptions:

1) There is no limit to the resources the trusted server can handle.
2) The trusted server has the pre-shared keys for decrypting and encrypting messages from the IoT device and the smartphone.
3) There is a chosen Hash algorithm agreed by and shared among all parties in the network.

C. Threat Model

Here, we consider an attacker (M) located anywhere and can spoof the messages sent by either the legitimate IoT device or the smartphone to launch a man-in-the-middle attack or a replay attack to achieve its goal. There is no limit to what the attacker knows about the system, but the attacker doesn't have the keys. The adversary may know the protocol and location of the devices but does not have physical access to the devices. We do not consider denial-of-service attacks and jamming attacks as this is orthogonal to this work. We considered two kinds of adversaries.

Adversary type 1 who attempts to perform a man-in-middle-attack on A's communication to T and P's communication to T.

Adversary type 2 who attempts to perform a reply attack on A's communication to T and P's communication to T.

IV. Proposed MAISTS Protocol

We present the proposed mutual authentication and message integrity protocol. The MAISTS protocol has two parts, the first one provides mutual and message authentication between the IoT device and the Trusted Server, while the second part provides mutual authentication between the Smartphone and the Trusted Server.

A. Protocol A: Mutual and message authentication between the IoT device and the Trusted Server

Here, when IoT device and the trusted server wants to talk to each other. They use simple symmetric key cryptography. The communication flow diagram, as shown in Fig. 1 is as follows.

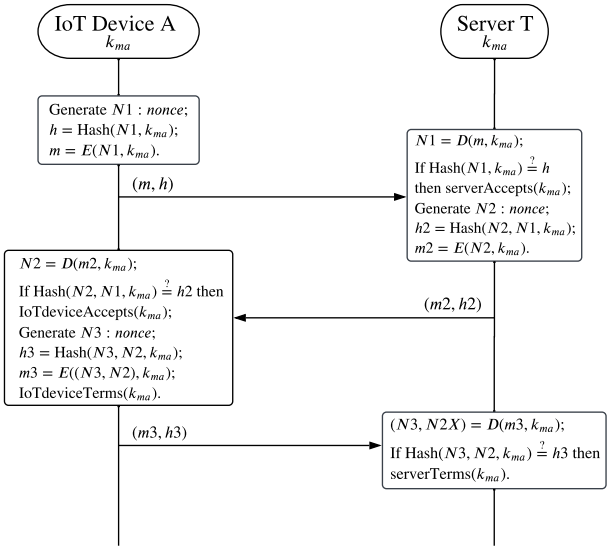

Fig. 1. Mutual authentication between the IoT device and Trusted Server

1) The legitimate IoT device (A) generates a random nonce, N_1 encrypts it with its key k_{ma}, it gets the hash of N_1 and k_{ma}, and sends the message and hash, (m, h) to the trusted server (T) through the communication channel.
2) T receives the message m, decrypts it with k_{ma}, and compares the hash with what it has, hash$(N_1, k_{ma}) \stackrel{?}{=} h$, if No, abort. Else, $serverAccepts$ process, then T generates a new nonce N_2, computes h_2 and m_2, and sends it to A.
3) A receives (h_2, m_2) from T. It decrypts the message and compares the hash $(N_2, N_1, k_{ma}) \stackrel{?}{=} h_2$, If No, aborts process. Else, $IoTdeviceAccepts$ process, then a new nonce N_3 is generated, and (m_3, h_3) is computed in a similar fashion and sent through the channel to the T, then $IoTdeviceTerminates$ process.
4) T receives (m_3, h_3), decrypts the message and compares the hash$(N_3, N_2X, k_{ma}) \stackrel{?}{=} h_3$. If No, abort. Else, $serverTerminates$ process. A secured session is then established and the protocol ends.

The protocol also includes a data integrity and confidentiality guarantee when a message (msg) is sent from the legitimate IoT device to the trusted server, as shown in Fig. 2.

1) A generates a random message (msg), gets its hash, and encrypts the message and the hash with k_{ma}. A sends the message, m to T.
2) T receives m, then decrypts the message using k_{ma} and compares the hash $(msg, k_{ma}) \stackrel{?}{=} h_2$. If No, abort process, else then accepts process.

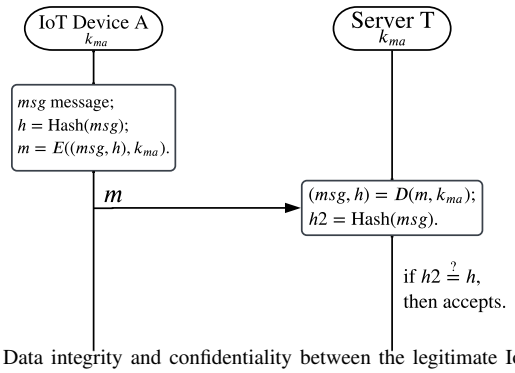

Fig. 2. Data integrity and confidentiality between the legitimate IoT device and the Trusted Server.

B. Protocol B: Mutual authentication between Trusted Server and Smartphone

The mutual authentication protocol between the smartphone and the trusted is based on public key cryptography. This is the case because the smartphone has the capability to handle the higher computational overhead of public key encryption and decryption. The steps for the mutual authentication protocol for a smartphone and a trusted server, as shown in Fig. 3, are as follows.

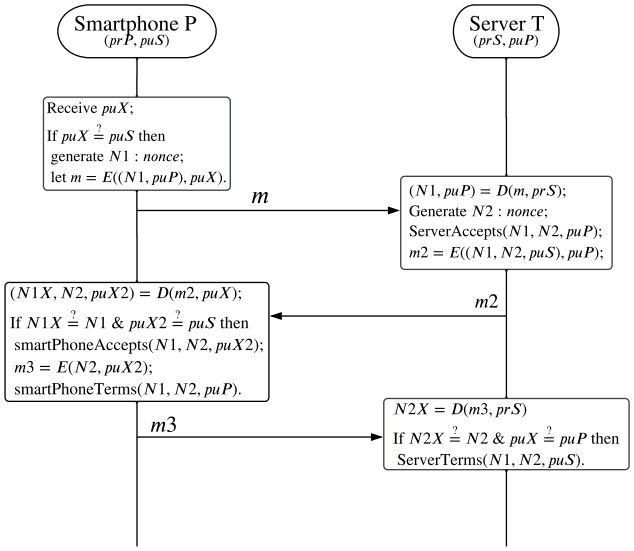

Fig. 3. Mutual authentication between the Smartphone and the Trusted Server.

1) The smartphone (P) receive a key puX and compare it to the server public key puS it has. If yes, it then generates a nonce N_1 and encrypts the nonce and its public key puP with puX to get m. It sends m to the trusted server (T).
2) T receives m, decrypts it with its private key, prS. It generates a new nonce N_2, then the $serverAccepts$ process. It encrypts N_1, N_2, puS, with the smartphone's public key puP to get m_2. It sends m_2 back to P.
3) P receives m_2, it decrypts the message with puX. It verifies that the nonce N_1 in the decrypted message is the same as the one it sent earlier, i.e., $N_1X \stackrel{?}{=} N_1$ and the public key of the server puX2 from the decrypted message, $puX2 \stackrel{?}{=} puS$. If No, abort. Else, $smartphoneAccepts$ process. It encrypts N_2 with $puX2$ sends m_3 to T, then $smartphoneTerminates$ its process.
4) T receives m_3, it decrypts it with its private key prS, and compares if the nonce it has is the same with the one it received, $N_2X \stackrel{?}{=} N_2$ and also compares $puX = puP$ on its side then the $serverTerminates$ its process. Now that the secret values are shared, a secured session is established.

V. Security Analysis

In this section, we first analyze the correctness of MAISTS, followed by its robustness analysis against the adversary we presented in Section III C. We implement our protocol using proverif, an open-source automated tool for formally verifying cryptographic protocols..

A. Correctness Analysis of MAISTS

For the correctness of MAISTS, we use the corresponding assertion to prove that we are confident that our protocol is correct and it meets its intended goal. We use corresponding assertions to prove that our protocol works and is procedural for the protocol execution.

1) Protocol execution: Here, we show that there is mutual authentication between the smartphone and the trusted server. We ensured that the assertion was true.

$$inj-event(smartPhoneTerms(\ldots)) \\ ==> inj-event(serverAccepts(\ldots)) = True, \quad (1)$$

$$inj-event(serverTerms(\ldots)) \\ ==> inj-event(smartPhoneAccepts(\ldots)) = True. \quad (2)$$

So, for the Smartphone to terminate its process, the server must accept, and for the server to terminate its process, the smartphone must accept, which guarantees that our protocol is correct and achieves mutual authentication.

Similarly, we show the mutual authentication between a legitimate IoT device and the trusted server. The following corresponding assertions must be true.

$$inj-event(IoTdeviceTerms(k)) \\ ==> inj-event(serverAccepts(k)) = True, \quad (3)$$

$$inj-event(serverTerms(k)) \\ ==> inj-event(IoTdeviceAccepts(k)) = True, \quad (4)$$

(3) and (4) show that for an IoT device to terminate its process, the server must first accept, and for the server to terminate its process, the IoT device must first accept. Finally, we prove the correctness of our protocol for message integrity

and confidentiality when we send a message from the IoT device to the trusted server.

$$event(NoChangeS(m_1)) \\ ==> event(NoChangeA(m_1)) = True. \quad (5)$$

(1), (2), (3), (4) show that mutual authentication was achieved between the smartphone and the trusted server and IoT device and the trusted server. In contrast, (5) guarantees message integrity and confidentiality between IoT device and the trusted server. Therefore, our protocol is correct.

B. Robustness analysis of MAISTS

Here, we prove that MAISTS is robust against Adversary type 1 and Adversary type 2 described in Section III

1) Adversary Type 1: The authentication in this protocol depends upon the secrecy of the nonces on the ends of both entities. An attacker (M) who tries to intercept communication between the smartphone and the server and then impersonate either of them to perform a man-in-the-middle attack would fail if we ensure that the following assertions are True:

$$Query\ not\ attacker(secretAN1[]) \quad is \quad true, \quad (6)$$

$$Query\ not\ attacker(secretAN2[]) \quad is \quad true, \quad (7)$$

$$Query\ not\ attacker(secretBN1[]) \quad is \quad true, \quad (8)$$

$$Query\ not\ attacker(secretBN2[]) \quad is \quad true. \quad (9)$$

(6) and (7) prove the secrecy of the nonces on the devices' end is protected. (8) and (9) show that an adversary cannot compromise the secrecy of the nonces on the server's end.

An adversary who wants to perform a man-in-the-middle attack for communication between the IoT device and the trusted server would fail because the attack doesn't have access to the secret key used by the IoT device and the assertion (3) and (4) is true. (3) ensures that the server will only accept a message from a legitimate IoT device if the message contains a fresh nonce that has not previously been used. If an attacker modifies the nonce, the server rejects it. (4) ensures that the IoT device will only accept message from the server if the message has the nonce which was used in the original message from the IoT device. Any modification to this nonce will lead to a rejection from the IoT device. Therefore this guarantees the robustness of adversary type 1 for the IoT device and the trusted server.

2) Adversary Type 2: An adversary Type 2, (M), who records transmissions from the smartphone to the trusted server and then retransmits it will fail because the (1) and (2) is True. Assertion (1) ensures that the server only accepts the termination from the Smartphone if it did not receive it previously. Else, the server will reject the message. Similarly, assertion (2) ensures that the smartphone only accepts the terms from the server if it did not receive it previously. Else, it will reject the message.

An adversary who wants to perform a replay attack for communication between the IoT device to the server would fail because of freshness on the nonces used by the IoT device and the server. Since the server only accepts messages containing fresh nonces, an attacker can only generate new nonces with access to the IoT device's secret key, which is unknown to him. Assertion (3) and (4) is true, and $(Query\ not\ attacker(test[])is\ false)$ to ensure the attacker does not have access to the secret key used by the legitimate IoT device.

VI. Automated Security Verification

We implemented the security properties of MAISTS using experimentation on a formal verification tool called Proverif. We also tested the correctness of our protocol in Section V. We modeled the smartphone and the IoT device separately, as shown in (3) and (1). We explained the corresponding assertions, observations, and verification in detail in Section V. The simulation scripts are attached to the deliverables for this course.

VII. Performance Analysis

Here, we show computational complexity for MAISTS for the communication between smartphone and server and IoT device and server. If we assume using the universal hashing, the worst case is $O(n)$ [4] where n is the size of the message. Similarly, the complexity of symmetric key encryption and decryption is also proportional to the message size, hence taking up to $O(n)$. The computations for asymmetric cryptography are more expensive. Public-private key generation uses resources up to $O(m^3)$, where m is the key size. Encryption and decryption are computationally expensive and depend on the key size. Hence the complexity is $O(m^2)$. Thus the complexity of the proposed protocol is $O(n + m^3)$. Overall our computational complexity is reasonable.

VIII. Conclusion

We have established a reliable and secure protocol for secure communication between an IoT device, a trusted server, and a smartphone. MAISTS addresses several security concerns like reply attacks and man-in-the-middle attacks. It also guarantees message integrity and Confidentiality. MAISTS is lightweight, simple, and practical, making it suitable for deployment in various IoT systems. We use a formal analysis with Proverif to show the robustness and correctness of MAISTS. In the future, we plan to scale this protocol to achieve multiple IoT devices in a distributed way.

References

[1] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer networks*, vol. 76, pp. 146–164, 2015.

[2] The Guardian. (2016) DDoS attack that disrupted internet was largest of its kind in history, experts say. [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[3] G. Lowe, "An attack on the needham- schroeder public- key authentication protocol," *Information processing letters*, vol. 56, no. 3, 1995.

[4] M. Babka, "Properties of universal hashing," 2010. [Online]. Available: http://ktiml.mff.cuni.cz/~babka/hashing/thesis.pdf