Article

User Authentication and Authorization Framework in IoT Protocols

Hasan Al-Refai and Ali Ahmad Alawneh

1. Faculty of IT-Department of CSIS, Philadelphia University; halrefai@philadelphia.edu.jo
2. Faculty of IT-Department of MIS, Philadelphia University
Correspondence: aalawneh@philadelphia.edu.jo

Abstract
The Internet of Things (IoT) has become one of the most attractive domains nowadays. It works by creating a special network between physical devices such as vehicles, home equipment, and other items. In recent days, the common technologies of communication such as Wi-Fi and 2G/3G/4G cellular are insufficient for the IoT networks because they are designed to serve appliances with immense processing capabilities such as laptops and PCs. Moreover, most of these technologies are centralized and use an existing infrastructure. Currently, the new communication technologies such as Z-Wave, 6LowPAN, and Thread are dedicated to the IoT and have been developed to meet its requirements. These technologies can handle many factors such as range, data requirements, security, power demands, and battery life. Nevertheless, the security issues in IoT systems have major concerns and matters because vulnerabilities in such systems may result in fatal catastrophes. In this paper, an enhanced IoT security framework for authentication and authorization is proposed and implemented to protect the IoT protocols from different types of attacks such as man-in-the-middle attack, reply attack, and brute force attack. The proposed framework combines an enhanced token authentication that has identity verification capabilities and a new sender verification mechanism on the IoT device side based on time stamp, which in turn can mitigate the need for local identity verification methods in IoT devices. The proposed IoT security framework is tested using security analysis with different types of attacks compared with previous related frameworks. The analysis shows the high capability of the proposed framework to protect IoT networks against many types of attacks compared with current available security frameworks. Finally, the proposed framework is developed using

Keywords: Internet of Things; security protocol; authentication; authorization; networks

1. Introduction

Kevin Ashton introduced the term Internet of Things (IoT) in 1999. Since then, it has been growing rapidly, and the number of installed IoT devices is expected to reach 38.6 billion at the end of 2025 according to Statista (Statista, 2021). Another statistic shows that the number of IoT-connected devices in 2017 was around 20 billion, and it will be about 22 billion in 2018 and more than double in 2025 (Statista, 2021). IoT is defined as a network of physical objects (sensors, actuators) that interact with one another to perform special tasks while maintaining connectivity to Internet services to obtain stationary control and monitoring over the Internet (connect humans with devices). The Internet is not only a network of computers but has evolved into a network of devices of all type and sizes, vehicles, smartphones, home appliances, toys, cameras, medical instruments and industrial systems, animals, people, and buildings that are all connected, communicating, and sharing information based on stipulated protocols to achieve smart reorganizations, positioning, tracing, safety and control, and even personal real-time online monitoring (De Donno et al., 2019). IoT can be classified into three categories, namely, people to people, people to machine, and machine to machine (M2M), all interacting through the Internet. IoT conforms to a paradigm that considers pervasive presence in the environment of various things or objects that through wireless and wired connections and unique addressing schemes can interact with one another and cooperate with other things or objects to create new applications and services to reach common goals. In this context, the research and development challenges to creating a smart world are enormous. The importance of IoT can be revealed by its wide range of applications, such as smart cities, smart workplaces, smart industries, smart cars, and smart homes. The IoT environment consists of a considerable number of smart devices such as sensors, actuators, and microcontrollers connected via different means of network communication such as wireless and wired networks. Smart devices inside IoT can connect, transfer information, and decide on behalf of people in a world where the real, the digital, and the virtual are converging to create smart environments that make energy, transport, cities, and many other areas more intelligent. IoT is characterized by real-world small things that are widely distributed and have limited storage and processing capacity, which involve concerns regarding reliability, performance, security, and privacy. However, cloud computing is considered the backbone of IoT technology because it relies on the Internet, can be accessed from everywhere, and has unlimited capabilities in terms of storage and processing power. IoT devices are required to perform the heavy task and statistical analysis of the gathered data in such cloud computing infrastructure. Cloud computing brings along a new cycle of development of the Internet. On this basis, cloud computing eliminates many limitations. With cloud computing, people will not be constrained by physical resources anymore. On the contrary, they can use the Internet anywhere and at any time (Song et al., 2011). Thus, a novel IT paradigm in which the cloud and the IoT are two complementary technologies merged expectedly to disrupt current and future smart services (Pawani, A. et al., 2018), and this merged technology or the new paradigm is called Cloud IoT. IoT also uses the cloud by exporting cloud services that can deliver services to users and allow them to control the IoT environment.

The concept of cloud computing has emerged widely in the past years because of the nature that facilitates the fast deployment of solution and services. However, the immense data transfer between cloud services and human applications revile high-priority demands for securing transmission protocols. Therefore, much research was conducted to overcome the available security issues in terms of user authentication and authorization processes in the Internet protocols used in the cloud services. (Choudhury et al., 2011) proposed a user authentication framework to prevent many types of attacks such as man-in-the-middle attack, impersonation attack, and phishing attack. Those attacks can access the server, damage it, or steal important information between users and servers. (Hasan Al-Refai et al., 2020) proposed an enhancement for the user authentication framework built by (Choudhury et al., 2011), but their work still had shortcomings. This research investigates the issues in the available security frameworks related to IoT protocols in terms of applicability and security levels, and proposes an enhanced security framework adapted for IoT devices based on token authentication technology and fingerprint (FP). The two main purposes of the proposed framework are to deliver a framework that understands the needs of IoT devices such as minimizing computational load and memory usage over IoT devices due to their capabilities from hardware design view such as limited memory and low processing power and to deliver the best practice security framework that overcomes the security issues and attacks available on the IoT protocols such as (man-in-the-middle attack, replay attack) by enhancing the registration, authentication, and authorization phases, and adding FP in the registration and authentication phase combined with token that contains information from the IoT device itself.

1.2 Research Context

This paper studies and analyzes the security needs for IoT protocols while communicating with cloud services over the Internet in terms of authentication and authorization schemas. IoT proposes new challenges over the cloud network due to its open architecture behavior. Moreover, standard security criteria used in the cloud network are not fit to be implemented by IoT protocols because they add extra load over the network, which uses very limited resources such as low memory, power consumption constraints, and limited CPU processing capabilities. Owing to those reasons, IoT protocols were created and concentrated on allowing IoT devices to communicate with one another and cloud services with the least power consumption and the fastest end-to-end packet delay. Therefore, IoT protocols did not contain solid standards for security implementations. Despite several efforts from IoT manufacturers to inject security features into their products, most of the implementations were not mature enough to protect the IoT networks and devices from attacks. Moreover, most of the IoT products did not consider the security factor, which led to serious harms due to different types of attacks over IoT networks (Deogirikar & Vidhate, 2017). This paper studies the different types of attacks and their effect on the IoT network to provide enhanced user authentication and authorization security framework for IoT protocols. This work proposes an enhanced framework of the one proposed by (Hasan Al-Refai et al., 2020), which overcomes the weakness points related to the IoT security preaches. One of the most important issues that appeared in the previous frameworks is token implementation over a session, which can be easily grabbed and used by a third party (i.e., attacker) using man-in-the-middle attack. This work also investigates the special needs of IoT devices in terms of low memory usage and low computation power. The previous security frameworks did not address such platforms and applied a very heavy load to the processor and the memory of devices to achieve high-security levels for user authentication and authorization. Cloud computing security itself is the set of control-based technologies and policies designed to adhere to regulatory compliance rules and protect the information, data applications, and infrastructure associated with cloud computing use. Different researchers developed cloud computing and merged it with many security protocols such as (Internet Protocol Security, Secure Shell Protocol, TLS, and ACE). All these protocols enhance the privacy of cloud computing to make it more secure from the attacker. Nevertheless, cloud security protocols do not suit the IoT technology because of their high complexity. Hence, many IoT protocols such as “Thread protocol and Z wave protocol” were proposed. Unfortunately, no security standard was adopted in these protocols. Therefore, this paper aims to present a standard security framework for IoT protocols, which allows IoT protocols to overcome the security issues available in IoT, such as authentication and authorization breaches —considering the low power consumption, CPU processing, and memory limitations of IoT devices.

This paper aims to propose an enhanced security framework for IoT protocols that overcomes the weakness of the previous frameworks in terms of attack vulnerabilities and process power utilization as follows:

- Identity information is added to the generated tokens to allow cloud services to verify the sender (Client, IoT device) identity and prevent stolen tokens from being used by attackers because stolen tokens sent by the attacker will have identity information that is different from the attacker request identity.
- All heavy processing to be performed on the cloud servers are relayed to reduce the load on the IoT devices by generating most of the encryption keys (public key [PK] and private key [P_RK]) on the cloud services. Moreover, most of the checking mechanisms and token generation are implemented on the cloud side. IoT devices only need to handle a simple preshared key encryption mechanism.
- A time stamp (TS) is added to each request or response from the cloud service to the IoT device only, which eliminates reply attacks to be done on IoT devices. IoT devices only use tokens to authenticate themselves on cloud services and cannot authenticate the server request using the token mechanism because it will add an immense load over the IoT devices. Therefore, the proposed security framework applies a special mechanism for requests and responses sent from the server side to the IoT devices by using preshared key data encryption to encrypt the sent data combined with a TS that restricts the hackers from using reply attack over IoT protocols to control IoT devices.
- The IoT registration phase with the FP authentication method by authorized clients to allow them to add new IoT devices securely and prevent other attackers from adding their own malicious IoT devices is added.
- A software tool for the proposed security framework that simulates all the framework phases to define a core software that can be integrated later inside IoT protocols is developed.

2. Related Work

This section investigates the main previously proposed frameworks and then compares them with our proposed work.

(Trnka and Cerny, 2016) presented a method for managing IoT device authentication and authorization rules that relies on a central identity store. Every device has a registered account in the identity store to confirm its identity against any device and application in the network. The method also provides scalability over the authorization rules because it allows central management for multiple IoT devices and services based on group policies. Token-based authentication is used inside the central identity store to enable fast data access. The results show a low time consumption for the authentication phase. However, this method used no means of data encryption, which allows different types of attacks, such as man-in-the-middle attacks and phishing attacks. Moreover, the token is not protected from being stolen. Moreover, the central identity store requires all the tokens to be verified directly through it, which disables the local token authentication from the service provider side and leads to extra time delay through the request–response procedures.

(Polat, H. et al., 2017) proposed new security authentication procedures based on a combination of token and machine ID authentication. They developed the proposed security authentication model to allow secure transmission of data for IoT M2M platforms. The new model leveraged the One-M2M model into the Database server, web service server, and web client server to allow the full support of Internet protocols and increase M2M scalability while decreasing system complexity by using Restful web services as a communication protocol because it supported the data structure of JSON. The NoSQL database was used for the database server. Finally, token-based authentication was used as a stateless authentication and authorization method to handle machine and human access requests. However, the presented M2M model lacked data encryption, which induced high levels of security breaches such as man-in-the-middle attack and phishing attack. Moreover, tokens were not secured from stealing through antiforgery mechanisms, which made the proposed security model open to reply attacks.

(Sciancalepore et al., 2017) presented a framework to solve the traditional approaches already adopted for web and cloud applications that cannot be used directly. Most require large computational and bandwidth capabilities (that cannot be reached with restricted devices) and provide low interoperability versus standardized communication protocols for IoT. The proposed work provided access control functions to the resources to which the IoT is exposed by using existing, widely accepted, and open standards and their proper alignment. The primary component of the OAuth-IoT framework was the gateway, which deals with the following: The first process is collecting the information produced by restricted devices through recently standardized lightweight protocols in the IETF context. The second process is monitoring permission requests provided by party applications through the well-known OAuth 2.0. The third process is supporting various symbolic formats to handle application authentication and authorization properly. The final stage of the system process is temporarily storing data collection. The limitation of this work is that it did not evaluate various scenarios in which the owners of the resources are not online or not identifiable with the client, whether the number of owners is one, two, or more. Table 1 summarizes the most substantial previous security frameworks related to the proposed IoT security framework and their used techniques.

(Claeys et al., 2018) presented the IoT security framework for authentication and authorization. The framework was designed over OAuth 1.0, a model combined with a light version of the ACE security standard. The security framework also presented a new self-securing token that contains security keys for identity verification using a technique called Proof of Possession (PoP). The PoP method was used with the accompanying long-term replay window value maintained by the authorization server that allowed it to increment a special counter inside the token for each new fresh token request for each client. This approach eliminated stolen token reuse attacks. The proposed framework also conducted a new method for authentication and authorization between devices with indirect connection criteria through proxy servers. The results showed a high demand for IoT devices during the token generation and authentication phase due to asymmetric encryption, but the framework was very secure over different types of attacks.

However, the framework allowed local token authentication between IoT devices, which added high processing and energy demand over the IoT devices. Moreover, no biometric verification for the users was conducted, which may lead to brute force attacks.

(Oh, Kim, and Cho, 2019) presented an access control framework that uses OAuth 2.0. However, the main use of OAuth 2.0 is protecting the user-specific domain and role-based access control to protect resources in the domain. Specifically, the authors expanded OAuth 2.0 to release an Interoperable Access Token (IAT) that acted as a global connectivity range through the IoT platforms by utilizing multiple pairs of clients' credentials. After testing the interoperability scenario using IAT on the implementation results, they came out from implementing the proposed framework (Mobius), which is one of the M2M-based IoT and FIRMWARE. The framework showed a good result by rapidly determining if the user can access the domain with a token. Moreover, the role-related permissions were easily managed by administrators in the client-specific domain. The drawbacks of this paper were that the framework cannot work in a dynamic environment; it only can work in a static environment. The second drawback was that the framework had no mechanism for protecting against stolen token attack.

Section 5.2 discusses in detail the comparison between previous works and the proposed work.

3. Enhanced User Authentication and Authorization Framework

This work investigates the issues in the available security frameworks related to IoT protocols in terms of applicability and security levels, and proposes an enhanced security framework adapted for IoT devices based on enhanced token authentication technology and FP. The enhanced token adds two new features to the regular token. The first one is for identity verification, whereas the second one allows fast authorization through the token directly. The two main purposes of the proposed framework are to deliver a framework that understands the needs of IoT devices such as minimizing the computational load and memory usage over the IoT devices due to their capabilities from the hardware design view, limited memory, and low processing power, and the best practice security levels that overcome the security issues and attacks available on the IoT protocols such as man-in-the-middle attack and replay attack. This approach enhances the registration, authentication, and authorization phases by adding the FP in the registration and authentication phases combined with the token that contains information from the IoT device itself.

Regular authentication between IoT network devices uses (user name and password) as the key for accessing the network, which may be the default data as (admin, admin), comes with most devices. Such authentication type may lead to authentication key exposures; this can be explained depending on the login issuer. The following table presents the a list of acronyms used in this work.

List of Acronyms

Acronyms	**Description**
C	Client (User) of the IoT
CD	Client data (FP, CN, CP, and RII)
FP	Fingerprint
CN	Client name
CP	Client password
RII	Requester Identity information (MAC, IP, and browser version)
T	Terminal (laptop, mobile phone, tablet)

PK	Public key
P_RK	Private key
SK	Shared key
CA	Client authorization level
ED	Token expiration date
IoTRI	IoT requester identity information (MAC, IP)
IoTD	IoT data (IoTID, IoTP)
IoTID	IoT device identity
IoTP	IoT device password
TEK	Token encryption key
TS	Time stamp

3.1 Client Login via Terminal

In this scenario, the user enters the user name and password via terminal (such as PC, tablet, or mobile) to access the IoT-connected devices through cloud services to collect data or send a command for a specific device. Even if the protocol is encrypted, if the messages are intercepted by a hacker, then the hacker can easily resend the same messages to the IoT network to perform the same commands, which may lead to an immense damage to the devices. Moreover, the IoT network needs to create a session for the logged users to verify them, which in turn creates a load on the devices' memories.

3.1.1 IoT Device to Cloud Service over the IoT Network

This method is the same as the client login, but in this case, the IoT device tries to control others or obtain some information to perform dependent actions. The same scenario appears because the connection can be intercepted, and the sent messages can be regenerated between devices.
However, in this case, creating the session has a greater effect because it stays on the IoT network until the device is shut down or the connection is closed, which leads to extra memory usage on each device.
The enhanced security framework consists of seven main phases:

1. Cloud service initiation phase
2. Client registration phase
3. Client login phase
4. IoT registration phase
5. IoT login phase
6. IoT to cloud server data transmission
7. Cloud server to IoT data transmission

3.1.2 Cloud Service Initiation Phase

In this phase, the cloud service repairs itself to receive the requests and data transactions from the clients and the IoT devices by generating the asymmetric keys (PK and P_RK) to prevent the client's credentials from being stolen. Moreover, this phase generates the symmetric token encryption key (TEK), which is used only by the cloud service to generate the encrypted tokens and decrypt them in the validation. The generated keys are refreshed in a constant time based on the expiration time of the token assigned by the cloud service administrator to increase the security level of the encryption key by eliminating crypto analysis attacks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 August 2022 doi:10.20944/preprints202208.0188.v1

Using this mechanism in the proposed IoT security framework allows the cryptographic keys to be generated only once and used for all requests in other phases. This mechanism reduces the overall process time needed to generate separate keys for different clients and IoT devices. This phase is mainly adopted by the proposed framework to reduce the response time from the server side to the clients and the IoT devices. Different types of encryptions (symmetric, asymmetric) are used to eliminate different types of attacks such as man-in-the-middle attack, reply attack, and brute force attack. Even though asymmetric encryption is considered heavy and time consuming, it is mandatorily used in the presented security framework. As illustrated by the sequence diagrams in Figures 4 (Step 3.1), 5 (Step 3.2), and 7 (Step 3.1), the asymmetric key PK is used only once to prevent the client and IoT credentials during transmission to the server from being stolen and reused by hackers because the preshared key, SK, can be stolen if used instead of PK in the mentioned steps. This step is performed to achieve the first and second problem statements. However, asymmetric encryption over the IoT devices in the proposed security framework is only used once and restricted to the encryption only, which minimizes the load of using asymmetric encryption because generating the keys and decrypting the data are the cloud server's responsibility.

3.1.3 Client Registration Phase

The client starts the handshake by sending a registration request to the server, and the server replies to the client with an asymmetric PK. The registration phase starts by requesting the user to enter the user name and password, with valid conditions such as (password length, number and character combinations, and uppercase with special characters' combination). After successful input, the framework asks the user to enter three valid FPs, using the FP reader attached to the client device. Next, the terminal uses the hash function to sign the submitted data, encrypts all the entered data with the hash code using the PK, and sends them to the server. The server decrypts the data using the P_RK, stores them as user account record with the requested permissions, and replies to the client that the registration is finished successfully.

On the one hand, the hash function is used in this phase to sign the transmitted data to prevent data from being manipulated or changed by hackers using man-in-the-middle attack, as shown in Figures 1 (Step 5), 2 (Step 5), and 4 (Step 5). On the other hand, FP biometric authentication is performed to prevent brute force attacks, as shown in Figures 4 (Step 3) and 5 (Step 3) because it eliminates the automatic user name and password generation tools from having a successful authentication on the cloud service. Other mechanisms to defend against brute force attacks include using captcha code. However, it does not completely prevent the attacks from being applied, especially for brute force attacks that use artificial intelligent recognition systems such as optical character recognition (OCR). Therefore, using biometric verification in the proposed security framework is the best choice to prevent such attack types. Encryption is also added in this phase to secure the transmitted data and prevent them from being exposed through man-in-the-middle attack, as mentioned in Figure 4 (Step 3.1). Figure 1 shows the sequence diagram for the registration phase in the proposed security IoT framework.

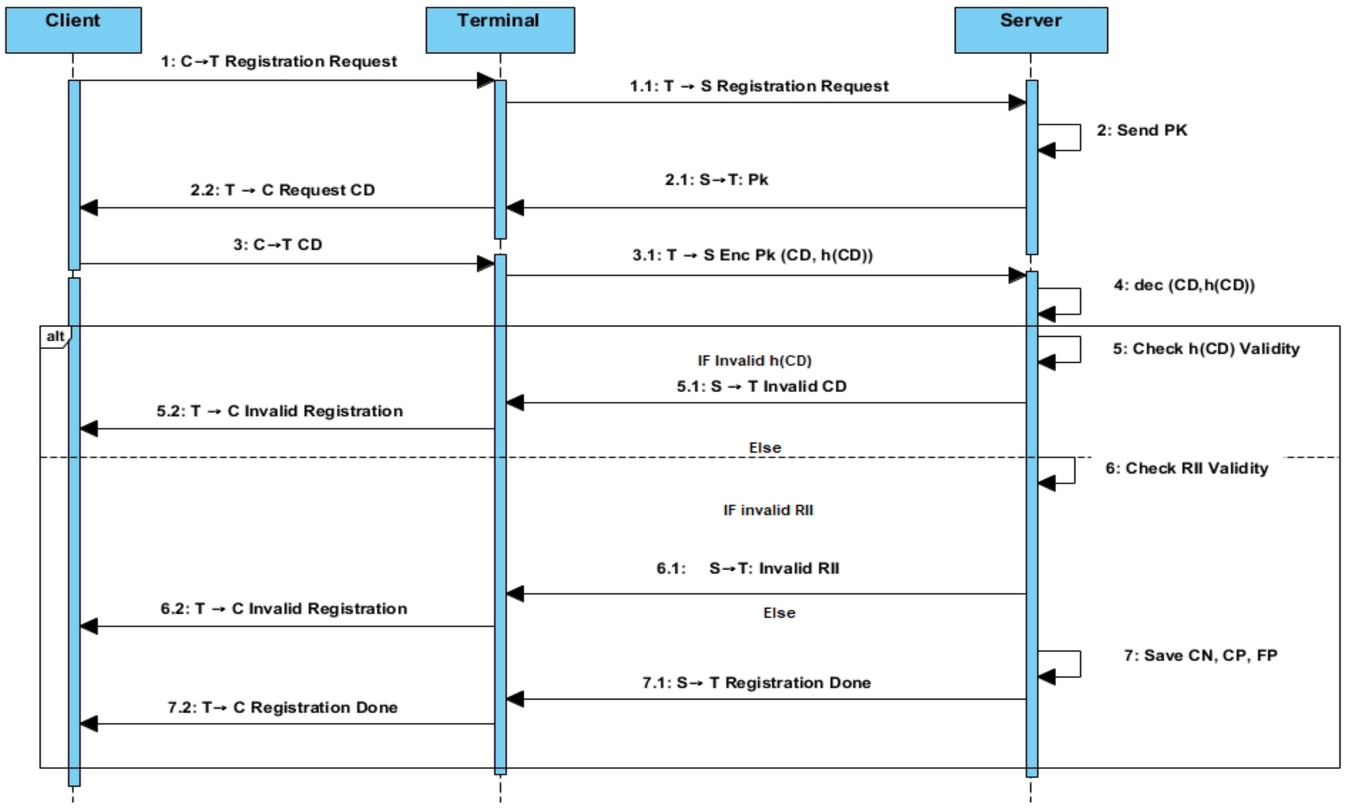


Figure 1 Sequence Diagram of Client Registration Phase

The steps of the client registration sequence diagram are illustrated below:

1- The client sends a registration request through the terminal device (i.e., laptop/mobile):

C→T: Registration Request

1.1- The terminal sends the request to the cloud server:

T→S: Registration Request

2- The server receives the request and sends *PK*.

2.1- The server sends *(PK)* to terminal and requests *CD*:

S→T: (PK)

2.2- Through the terminal, the client asks to insert *CD*.

3- The client inserts *CD*.

3.1- The terminal computes (CD,*h (CD))* and sends them to the server:

T→S:Enc $_{pk}$(CD, *h (CD))*

4- The server decrypts *(*CD, *h (CD))* and starts to check *h (CD), RII* validity.

5- The server checks *h (CD)* validity.

5.1- IF *h (CD) is* invalid.

S→T: Invalid CD

5.2- Through the terminal to client Invalid Registration.

T→C: Invalid Registration

6- Checks *RII* Validity.

6.1-IF *RII* Invalid.

S→T: Invalid RII.

6.2 - Through the terminal to client Invalid Registration.

T→C: Invalid Registration

7- The server saves *CN, CP, and FP*.
7.1- The server sends Registration Done to the terminal:
S→T: Registration Done.
7.2- Through the terminal to client Registration Done:
T→C: Registration Done.

The main purpose of the client registration phase is not only to add the client into the authenticated pool but also to prevent the client information from being stolen using man-in-the-middle attack. This step is performed by sending the user information through a secure tunnel and encrypting the information using asymmetric encryption. Moreover, the registration phase applies identity verification based on the Requester Identity Information (RII) sent by the terminal to verify the client identity and eliminate the reply attack. The usage of the biometric authentication also removes the possibility of having brute force attacks. The framework also uses the hashing mechanism to eliminate data changes and manipulation by hackers.

3.1.4 Client Login Phase

This phase is implemented to ensure that the user name, password, and FP information transmitted by the client are not compromised by any type of attacks while they are sent to the server through the network and to secure the data transmitted later between the clients and the server. Thus, the security framework uses asymmetric cryptography techniques based on PKs and P_RKs because it prevents the P_RK, which is needed to decrypt the data, from being shared over the network, as shown in Figure 2 (Steps 2 to 4). However, the complexity of asymmetric encryption is very high due to the long process required to generate PKs and P_RKs. The encryption and decryption of the algorithm is time consuming, and it is not the best way to handle data transmission over slow networks such as IoT networks. Hence, the security framework uses asymmetric key generation only in the cloud service initiation phase periodically based on the expiration time of the token. In addition, the asymmetric encryption/decryption in the login phase is only used in the first step to exchange the user credentials with the preshared encryption key (i.e., session key, shared key [SK]).
The client login phase starts by sending the login request to the cloud service. The cloud service then replies to the request with the generated PK. The client submits his credentials (user name, password, and FP) in the terminal. The terminal then generates a preshared key (SK) and uses hashing function on the submitted credentials and the generated preshared key. Then, the PK received from the cloud server is used to encrypt the client credentials, the SK, and the hash code. Next, the encrypted data are sent to the cloud server. The hash code is used to detect any manipulation in data by a hacker during their transmission over the IoT network, as shown in Figure 5 (Step 5). After that, the cloud server receives the encrypted data and decrypts them using its P_RK. Later, it authenticates the user name, password, and finger stamp. In case of a valid login, the cloud server generates the token using the client information and the TEK key, and sends it back to the client. The token generation is completely taken by the server starting from generating a payload string of data that contain client name, client authorization level, token expiration date, the RII, and preshared key. The generated payload string is then signed using a hash function. The generated payload string and the hash code are encrypted after that using the TEK key. Later, the final encrypted data are sent back to the client and stored in its memory as a token. The token is sent over the network without encrypting it using the SK key generated by the terminal because it is secured by itself using the TEK key. Moreover, the token cannot be reused by hackers because it contains identity verification information. The procedure of this phase is as follows (Figure 2):

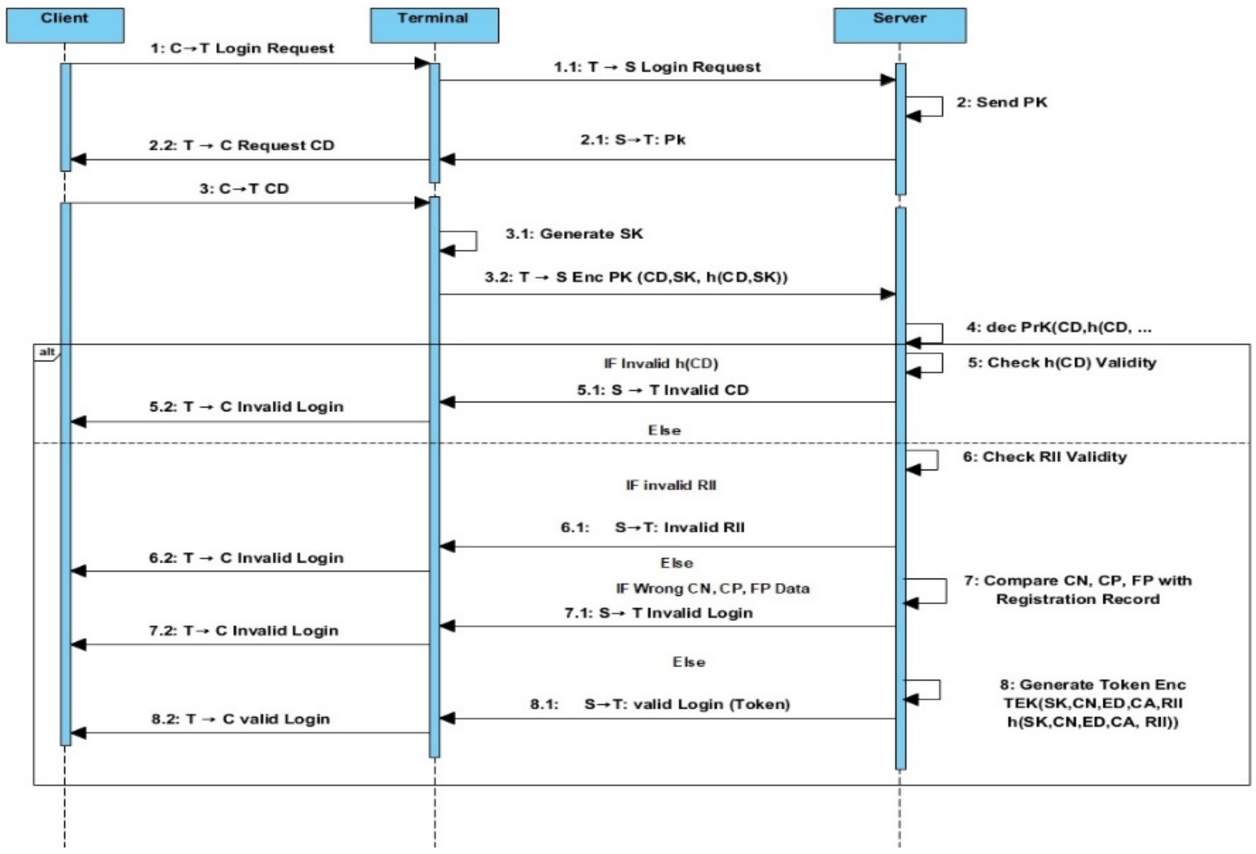


Figure 2: Client Login Phase Sequence Diagram

The steps of the client login sequence diagram are illustrated below:

1- The client sends a login request through his terminal (laptop/mobile):
C⟶T: Login Request
1.1- The terminal sends the request to the cloud server:
T⟶S: Login Request
2- The server receives the request and generates and sends *PK*.
2.1- The server sends *(PK)* to terminal and requests *CD*:
S⟶T: (PK)
2.2- Through terminal, the client asks to insert *CD*.
3- The client inserts *CD*.
3.1- The terminal computes (*SK).*
3.2- The terminal generates (*CD, SK, h (CD, SK))* and sends them to the server:
T⟶S:Enc $_{pk}$(CD, *SKh (CD, SK))*
4- The server decrypts *(CD, SK, h (CD, SK))* and starts to check *h (CD), RII* validity.
5- The server checks *h (CD)*validity.
5.1- IF *h (CD)*Invalid.
S⟶T: Invalid CD
5.2- Through the terminal to client Invalid Login:

T→C: Invalid Login
6- Checks *RII* validity.
6.1-IF *RII* Invalid.
S→T: Invalid RII.
6.2 - Through the terminal to client Invalid Login:
T→C: Invalid Login
7- The server compares *CN*, *CP*, and FP with Registration Record.
7.1- IF wrong, *CN*, *CP*, *FP* Data.
S→T: Invalid Login.
7.2- Through the terminal to client Invalid Login:
T→C: Invalid Login
8- The server generates *TEK.*
8.1 The server generates *Enc TEK (SK, CN, ED, CA, RII,) h (SK, CN, ED, CA, RII))*
8.2- The server sends Login Done to the terminal with the token:
S→T: valid Login (Token)
8.3- Through the terminal to client Login Done:
T→C: Valid Login.

The main purpose of the Client Login Phase is to prevent the IoT network from unauthenticated login breaches using man-in-the-middle attack by securing the user credential information using asymmetric encryption. Moreover, this phase protects the client authentication from being hacked using stolen tokens because of the identity verification mechanism. Furthermore, it prevents reply attacks due to encrypted RII data.

3.1.5 IoT Registration Phase

After a successful login (authentication) of the client through the framework to the cloud server, the client can send the IoT register request and submit all the needed information such as IoT ID, password, and IoT device-related information such as MAC address. The submitted IoT device information are then encrypted by the terminal using the preshared key (SK), which is used to protect the data from being compromised using man-in-the-middle attack, as shown in Figure 6 (Step 2.1). The terminal then sends the encrypted data with the authentication token to the cloud service. The framework then verifies the sender client token validity and the received IoT information, and saves the new registered IoT device information. Figure 3 shows the sequence diagram for the IoT registration phase.

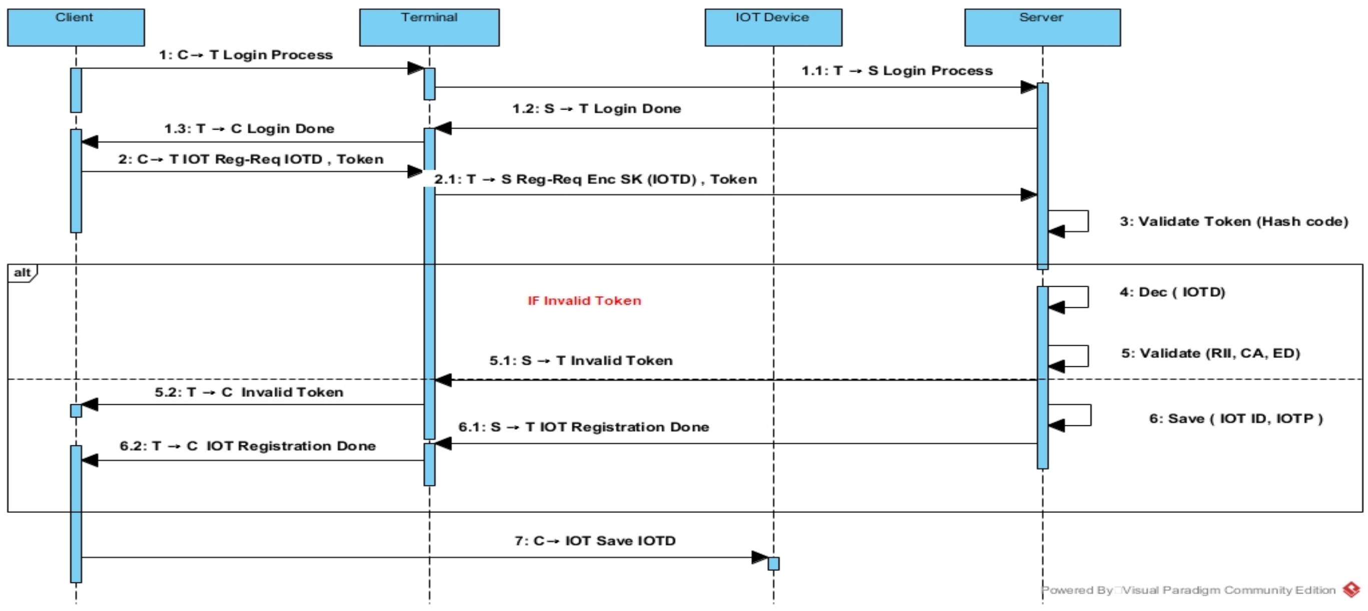


Figure 3: IoT Registration Phase Sequence Diagram

The procedure for this phase is illustrated as follows:

1- The client sends Login Process through his terminal (laptop/mobile):
C→T: Login Process
1.1- The terminal sends the request to the cloud server:
T→S: Login Process
1.2- The server receives the request and sends Login Done through the terminal to the client:
S→T: Login Done.
1.3 Terminal to Client Login Done:
T→C: Login Done
2- Client to the Terminal *IoT Registration Request* with, *IoT Data*, and *Token:*
C→T: IoT Reg-Req (IoT Data, Token)
2.1 Through Terminal to the cloud server *Registration Request* with *SK* for *IoTData, TOKEN:*
T→S: Enc SK (IoT Data), Token.
3- The server *Validate Token (Hash Code).*
4- The server *Decrypt IoTD.*
5- The server *Validate (RII, CA, ED).*
5.1 IF *Invalid Token:*
S→T: Invalid Token.
5.2 Through the terminal to the client *Invalid Token*:
T→C: Invalid Token.
6- The server saves (IoTID, IoTP)
6.1 The server sends IoT Registration Done to the terminal:
S→T: IoT Registration Done.
6.2 Through the terminal to the client IoT Registration Done:
T→C: IoT Registration Done.
7- Through the client to the IoT Device, save IoT Data:
C→IoT: Save IoTD.

doi:10.20944/preprints202208.0188.v1

The main purpose of the IoT registration phase is to protect the IoT network from malicious IoT devices injected by hackers who may affect the whole network operation. Moreover, this phase prevents the IoT device's credentials from being stolen using attacking techniques such as man-in-the-middle attack and reply attacks. Stealing the IoT device credentials may lead to severe damages depending on the operation type of the IoT device.

3.1.6 IoT Login Phase

After the client's successful registration of the IoT device, the IoT device sends a login request to the server. The server replies to the IoT device by sending its PK, which is used to prevent man-in-the-middle attack in earlier login stages, as shown in Figure 7 (Step 3.1). Then, a unique SK is generated on the side of the IoT device, which will be used later for securing data transactions between the IoT device and the cloud service using light simple encryption mechanism, as shown in Figures 7 (Steps 3 to 3.1), 8 (Step 1), and 9 (Steps 2 to 2.1). After that, the IoT device sends encrypted (IoTD, IoTRI, SK h (IoTD, SK, IoTRI)) to the cloud service. The server checks the validation of the (h (IoTD, SK, IoTRI)). This stage ensures that the data have not been changed by hackers using man-in-the-middle attack, as shown in Figure 7 (Step 5). The cloud service then responds to the IoT device with invalid login if the h (IoTD) is invalid. Otherwise, the server checks the validation of the IoRTI; if it is invalid, the server responds with invalid login to the IoT device. Next, the cloud service compares the IoT ID and the IoTP data. If IoT ID and the IoTP data are incorrect, the server responds with invalid login. Otherwise, the server responds to the IoT device with a valid login (token) encrypted with the TEK encryption key. Figure 4 shows the sequence diagram of the IoT login phase.

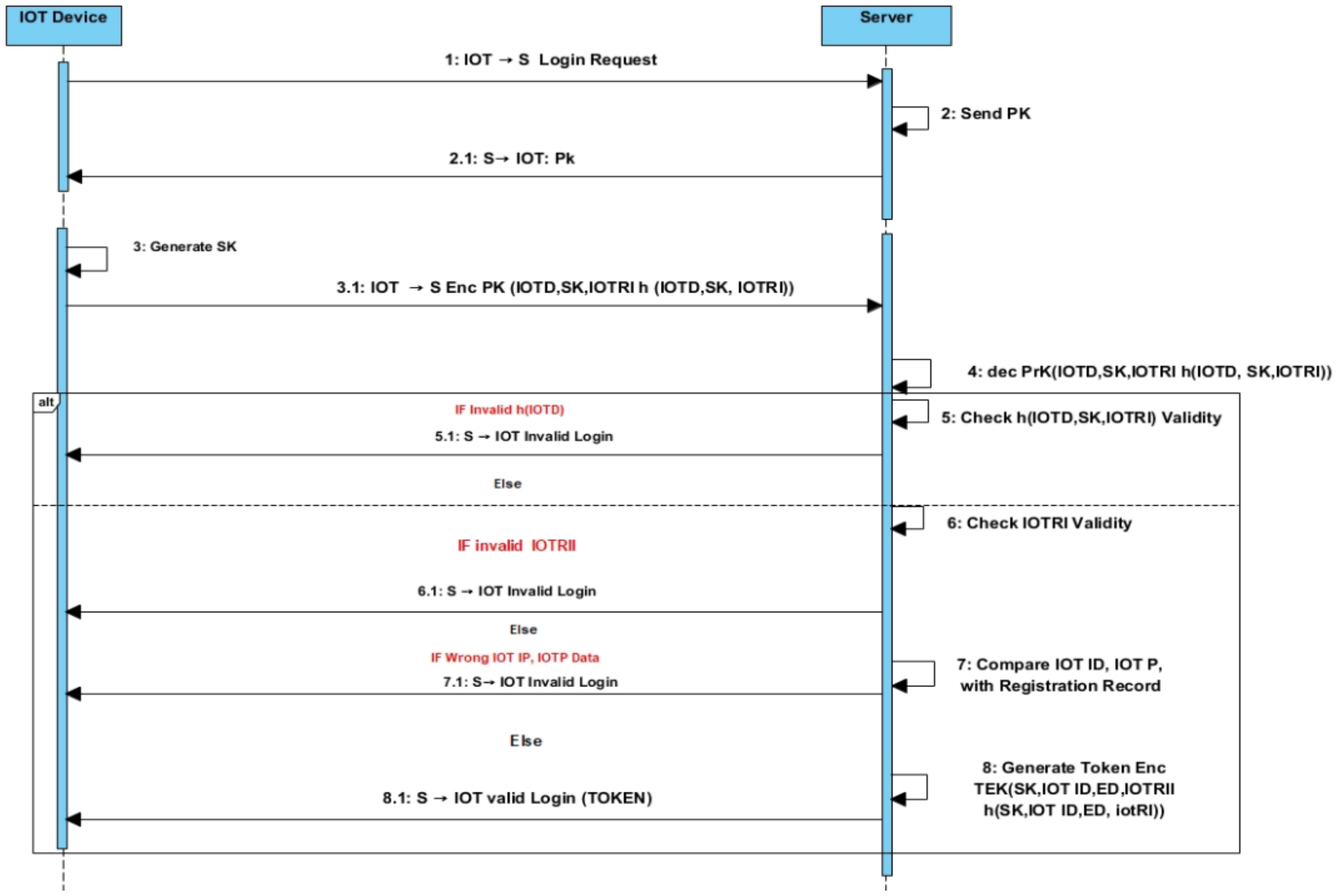

IOT Device
Server
1: IOT → S Login Request
2: Send PK
2.1: S→ IOT: Pk
3: Generate SK
3.1: IOT → S Enc PK (IOTD,SK,IOTRI h (IOTD,SK, IOTRI))
4: dec PrK(IOTD,SK,IOTRI h(IOTD, SK,IOTRI))
alt
IF Invalid h(IOTD)
5.1: S → IOT Invalid Login
5: Check h(IOTD,SK,IOTRI) Validity
Else
6: Check IOTRI Validity
IF invalid IOTRII
6.1: S → IOT Invalid Login
Else
IF Wrong IOT IP, IOTP Data
7.1: S→ IOT Invalid Login
7: Compare IOT ID, IOT P, with Registration Record
Else
8.1: S → IOT valid Login (TOKEN)
8: Generate Token Enc TEK(SK,IOT ID,ED,IOTRII h(SK,IOT ID,ED, iotRI))

Figure 4: IoT login phase

The procedure for this phase is illustrated as follows:

1- The IoT device sends Login Request to the cloud server:
IoT Device → S: Login Request.
2- The server receives the request and generates *PK* and P$_R$K:
S→IoT Device: Login Process.
2.1- The server sends (*PK)* to the IoT device:
S→IoT Device: (PK)
3- The IoT device generates *SK*
3.1- Through IoT device to cloud server with *SK* for *(IoT Data, SK, IoTRI), h (IoT Data, SK, IoTRI).*
4- The cloud server decrypts *(IoT Data, SK, IoTRI), h (IoT Data, SK, IoTRI).*
5- The cloud server checks *h (IoT Data, SK, IoTRI)* validity.
5.1 IF Invalid h (IoTD), the server sends IoT Invalid Login to the IoT device:
S→IoT Device: Invalid Login
6- The cloud server checks the IoTRI Validity.
6.1 IF Invalid *IoTRI*, the cloud server sends *Invalid Login* to the *IoT device.*
7- The cloud server compares *(IoTID, IoTP)* with Registration Record.
7.1- IF wrong *IoTID, IoTP Data*, the cloud server sends Invalid Login to the IoT Device.
8- The cloud server generates TEK (From Initiating Phase).
8.1- The cloud server generates Token *Enc TEK (SK, IoTID, ED, IoTRI) h (SK, IoTID, ED, IoTRI)).*

8.2- The cloud server sends Valid Login with Token to the IoT device.
S→IoT Device: Valid Login (Token).

The main purpose of the IoT device login phase is to protect the IoT network from unauthenticated login breaches using malicious IoT device or man-in-the-middle attack by securing the IoT device credential information using asymmetric encryption. Moreover, this phase prevents reply attacks using the identity verification technique.

3.1.7 IoT to Cloud Data Transmission

After authenticating the IoT device, the IoT device sends an encryption request to the server with preshared key (SK) and token. The server then decrypts the token and checks the h (SK, IoTID, ED, IoTRI). If the h (SK, IoTID, ED, IoTRI) is invalid, the server responds with invalid requests. Next, the server checks the validation of the IoTRI; if it is invalid, the server responds with invalid request. The server then checks the validation of ED. If it is invalid, the server responds with expired token. Lastly, the server decrypts (SK) data and sends a response with encrypted SK (Response, TS). In the IoT device side, it decrypts SK (Response, TS) and checks the validation of TS; if it is invalid, it ignores the response. Otherwise, it acts based on the response. Figure 5 shows the sequence diagram of the IoT to cloud data transmission.

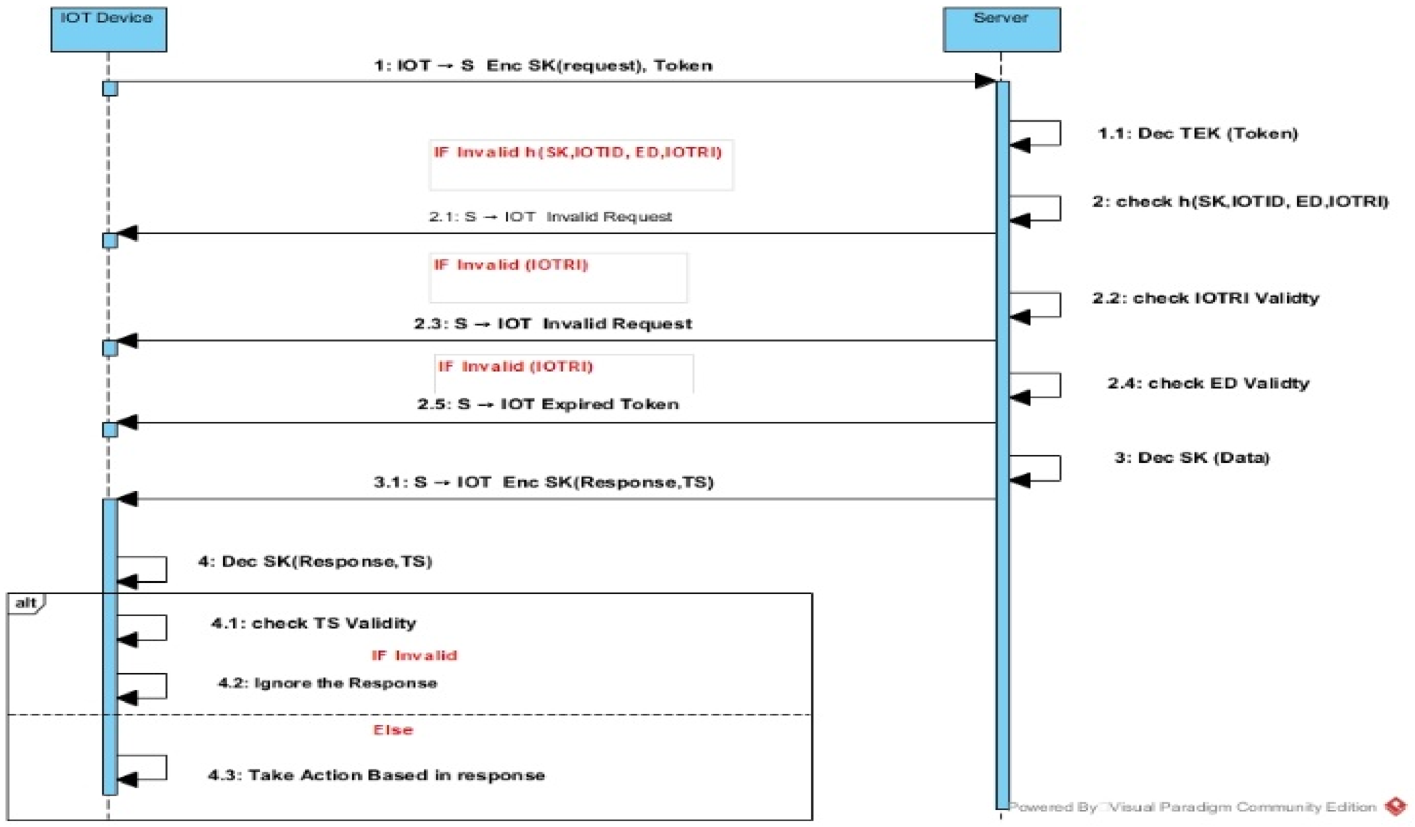

IOT Device
Server
1: IOT → S Enc SK(request), Token
1.1: Dec TEK (Token)
IF Invalid h(SK,IOTID, ED,IOTRI)
2: check h(SK,IOTID, ED,IOTRI)
2.1: S → IOT Invalid Request
IF Invalid (IOTRI)
2.2: check IOTRI Validty
2.3: S → IOT Invalid Request
IF Invalid (IOTRI)
2.4: check ED Validty
2.5: S → IOT Expired Token
3: Dec SK (Data)
3.1: S → IOT Enc SK(Response,TS)
4: Dec SK(Response,TS)
alt
4.1: check TS Validity
IF Invalid
4.2: Ignore the Response
Else
4.3: Take Action Based in response
Powered By Visual Paradigm Community Edition

Figure 5: IoT to Cloud Data Transmission

The procedure for this phase is illustrated as follows:

1- The IoT device sends the encryption request with token to the cloud server:
IoT Device → S: Enc SK (Request), Token
1.1- The server receives the request and decrypts using TEK (Token).
2- The server checks *h (SK, IoTID, ED, IoTRI)* validity.
2.1- IF Invalid *h (SK, IoTID, ED, IoTRI),*
The cloud server sends Invalid Request to the IoT device:
S→IoT Device: Invalid Request.
2.2- The cloud server checks *(IoTRI)* validity.
2.3- IF Invalid *(IoTRI),* the cloud server sends Invalid Request to the IoT device:
S→IoT Device: Invalid Request.
2.4- The cloud server checks *ED* validity.
2.5- IF Invalid *ED*, the cloud server sends *Expired Token* to the IoT device:
S→IoT Device: Expired Token.
3- The cloud server decrypts the data.
3.1- The cloud server sends encrypted response with TS to the IoT device:
S→IoT Device: Enc SK (Response, TS).
4- The IoT device decrypts the response with the TS and checks the validity.
4.1- The IoT device checks *TS* validity.
4.2- IF Invalid *TS*, the IoT Device rejects the response.
4.3- IF valid, act based on the response.

The main purpose of the IoT to cloud server data transmission phase is to protect the connection between the IoT device and the server from man-in-the-middle attack by securing the IoT device request information using symmetric encryption and identity verification through the enhanced token, as shown in Figure 6 (Step 1). Moreover, the reply attack is prevented in this phase by having token verification with RII. The TS in this phase is defined as the real time of the cloud service when the response to the IoT device is sent. The TS used in this phase protects the IoT device from being hacked using a reply attack without the need of a local authentication method. This new mechanism leads to the reduction of the load over the IoT device, which solves the fourth problem statement and enhances the IoT device performance in terms of response time and IoT network overall end-to-end delay. Moreover, the cloud service does not need to save the special SK for each IoT device because it depends on the token to obtain the SK.

3.1.8 Cloud Server to IoT Data Transmission

In this phase, the server sends a token request to the IoT device, and then the IoT device sends a response to the server using the same token. This initial step of requesting the token allows the server to obtain the SK, which is used in encrypting the sent data to the IoT device. The next step is for the server to send an encrypted request with (TS) to the IoT device. Then, the IoT device side decrypts (Request, TS) to check its validation; if it is invalid, the IoT device checks the TS validation; otherwise, it ignores the request. Figure 6 shows the sequence diagram of the server to IoT data transmission.

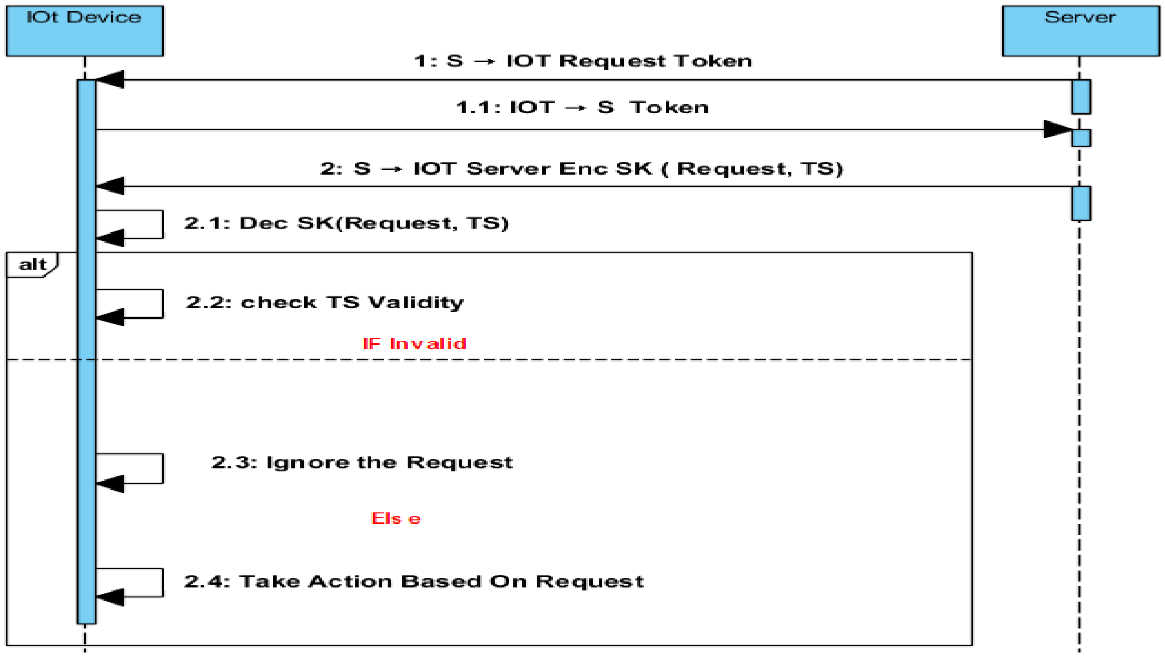


Figure 6: Server to IoT Data Transmission

The procedure for this phase is illustrated as follows:

1- The cloud server asks the IoT device to send the token:
S⟶ IoT Device: Request Token.
1.1- The IoT device receives the request and sends his token to the cloud server.
2- The cloud server receives the token from the IoT device and sends the encrypted request to the IoT
S⟶ IoT Device: Enc SK (Request, TS).
2.1 The IoT device decrypts the request.
2.2 Check TS validity.
2.2- IF Invalid *TS,*
2.3- The IoT device ignores the request.
2.4- Take action based on the request.

The main purpose of the IoT to cloud server data transmission phase is to protect the connection between the IoT device and the server from man-in-the-middle attack by securing the IoT device request information using symmetric encryption. Moreover, in this phase, the TS is used to protect the IoT device from being hacked using a reply attack. The TS is used to replace the local authentication model needed by the IoT device to validate the cloud service authentication and identity. This model can be explained by the fact that the cloud service is the only platform that can decrypt the information inside the enhanced token. Therefore, the cloud server is the only entity that can send encrypted data with the correct SK key related to each IoT device. Moreover, the encrypted TS inside the sent packet by the server cannot be changed by hackers because changing it directly without decryption will corrupt the TS leading to ignore the request by the IoT device finally.

4. Proposed Security Framework Defense over Man-in-the-Middle attack and Reply Attack Test Cases

The standard IoT protocols do not provide sufficient built-in security procedures (Wardana & Perdana, 2018) such as data encryption or authentication even though in more mature protocol such as thread IoT protocol. The later protocol, which is originally inherited from the Internet protocol suite, contains several simple security procedures, but these protocols do not contain IoT device identity verification.
Figures 7 and 8 show different examples of IoT protocols to connect the client with the cloud server. In Figure 9, the client sends the login request with his client data (CD) to the cloud server without encryption to respond to the client. However, in the case of interruption due to man-in-the-middle attack that presents a new connection (hacker), in the connection between the client and the cloud server, the hacker receives the CD and sends it to the cloud server to log in to the system as a client. Figure 10 shows another case, where the client sends the login request with his data (CD) to the cloud server to respond to the client, but in this case, the data are encrypted. In the man-in-the-middle attack, the hacker interrupts the connection and receives the packet from the client and sends it as it is to the cloud server, and the cloud server responds to the hacker as a client.

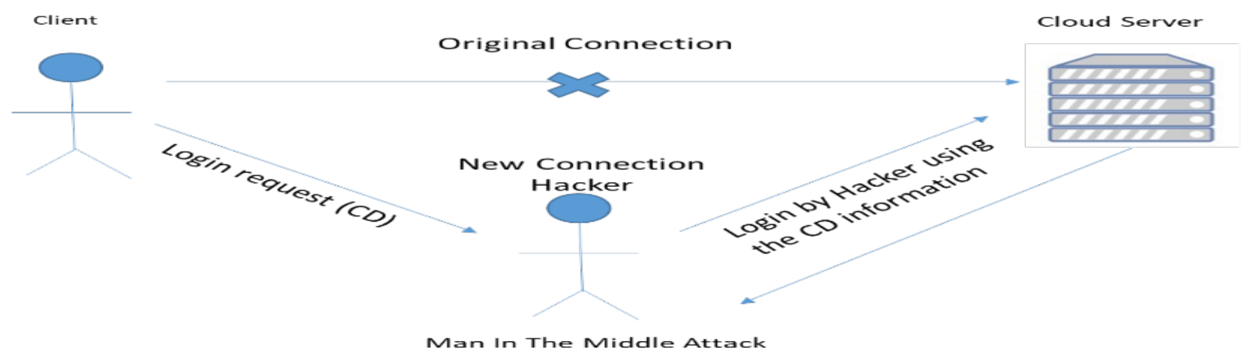


Figure 7: Man in the middle attack over standard IoT protocol without encryption

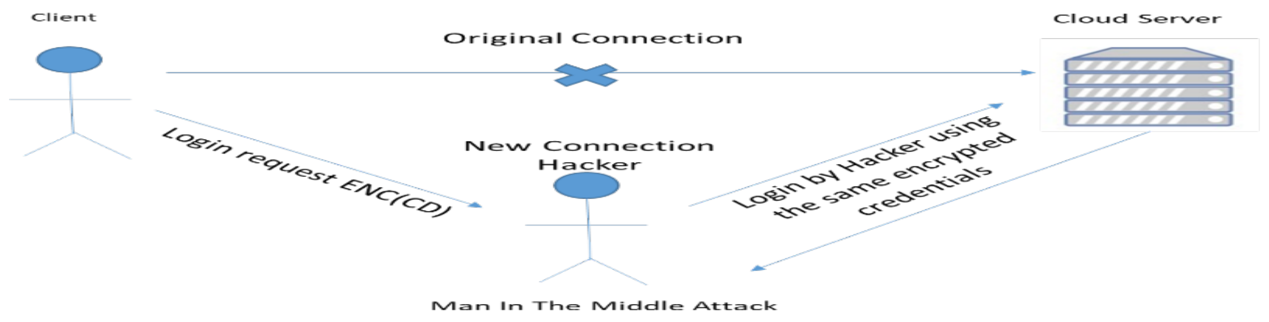


Figure 8: Man in the middle attack over standard IoT protocol without encryption

The proposed framework shown in Figure 9 prevents man-in-the-middle attack by sending the login request with encrypted (CD) and RII. Thus, if the attacker tries to steal the data and send them to the cloud server, the attacker will not obtain access to the cloud because the RII is changed.

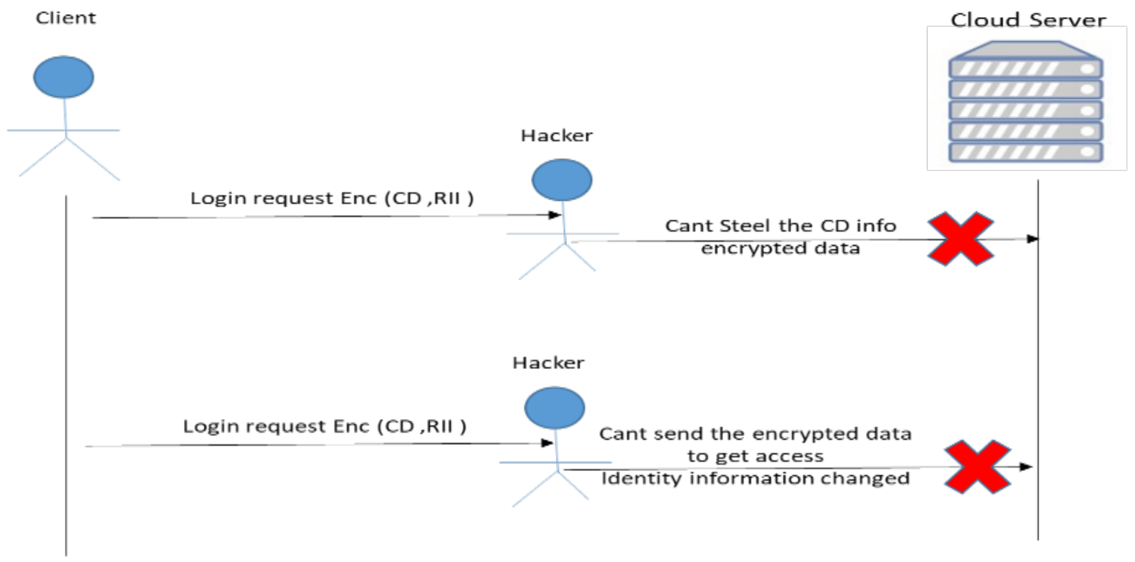

Client
Cloud Server
Hacker
Login request Enc (CD ,RII)
Cant Steel the CD info
encrypted data
Hacker
Login request Enc (CD ,RII)
Cant send the encrypted data
to get access
Identity information changed

Figure 9: Man in the middle attack defense over the proposed IoT security protocol in the client login phase

Figure 10 shows another type of attack, which is the reply attack. In this scenario, the cloud server sends a specific action command to the IoT device to take a specific action. When sending the command, the attacker captures the action command and waits for a specific time to resend the same action command to the IoT device to corrupt the normal operation of the IoT device.

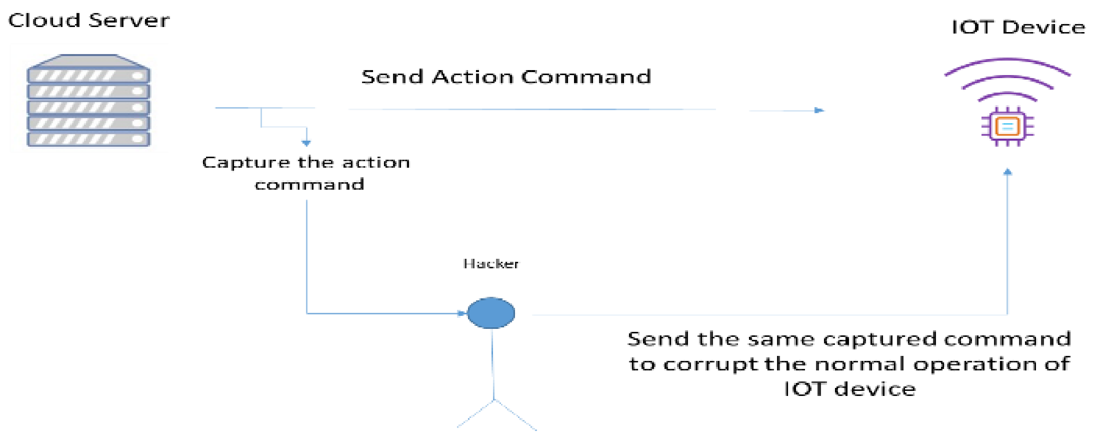


Figure 10: Reply attack over standard IoT protocol

Figure 11 shows how the proposed framework prevents the reply attack. The IoT device sends a token to the cloud server based on a request from the cloud server. This token contains a shared (SK). The server then sends an encrypted SK with the request and TS to the IoT device. The IoT device decrypts the SK and checks the validation time of the TS by checking the sending time from the server. For example, if the sending time from the server to the IoT device is two minutes, the IoT device will accept the request if the sending time is within two minutes.
However, if the time is more than two minutes, the device will ignore it. The advantages of this approach are preventing any request sent by the reply attacker and reducing the load on the IoT device by removing the local identity verification and authentication procedures from the IoT device side.

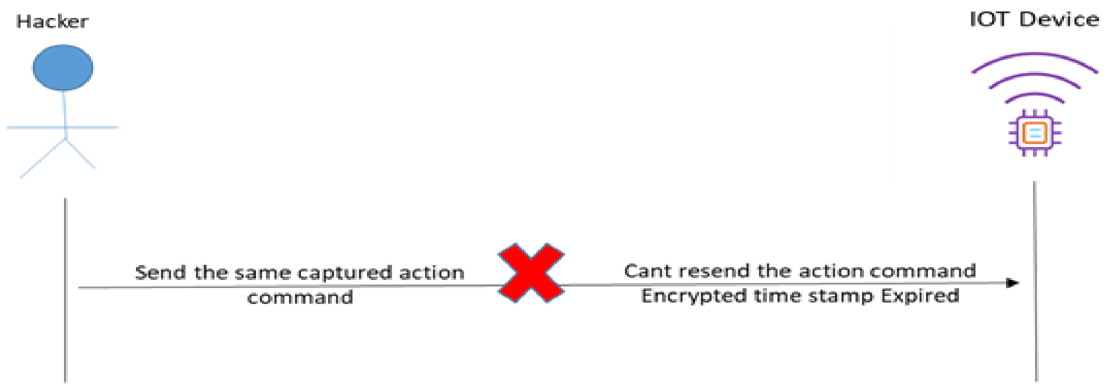

Hacker
IOT Device
Send the same captured action command
Cant resend the action command Encrypted time stamp Expired

Figure 11: Reply attack defense using the proposed IoT Security Framework in the data transmission phase

5. Results and Discussion: Security Analysis

In this section, the security requirements needed by the proposed IoT security framework to prevent attacks such as reply attack, man-in-the-middle attack, and brute force attack are first investigated. Then, the proposed framework is compared with other previous secure IoT frameworks.

5.1. Security Requirements

Replay Attack
A replay attack occurs when the attacker captures the requests and response packets between the clients, the IoT devices, and the cloud services to repeat the request when the session is ended. The proposed framework includes the enhanced token that contains an expiration date and identity verification information that eliminate such attack. Moreover, on the IoT side, reply attacks cannot be established because the proposed security framework deploys a smart authentication mechanism using a preshared key and customized TS, which prevents such attack and reduces the load on the IoT device by cancelling the need for local authentication on IoT devices.

Man-in-the-Middle Attack
This attack is considered one of the most effective attacks because it is implemented by a hacker by intercepting the communication between the clients, the IoT devices, and the cloud service. All sent requests and response commands between IoT devices, clients, and cloud services are sent by the hacker itself, and this enables the hacker to manipulate the requests by changing the data inside the requests, stealing the credentials, or even

stealing the tokens assigned with each request. The proposed IoT security framework can prevent such attacks because the sent data on all framework phases are encrypted and cannot be compromised. Moreover, the identity verification methods used by the proposed security framework disable the capability of reusing the intercepted requests. In addition, data manipulations or changes in request information cannot be done because the sent requests of the framework are signed using hash code, which allows the detection of any change in content.

Brute Force Attack
This attack uses user name and password generation tools that allow testing the large amount of credential patterns in a limited time to obtain a valid guess of IoT devices or client's credentials. Despite defense mechanisms such as the captcha code, which forces the clients to enter randomly generated patterns, the overall complexity of the attack time increases. New mechanisms including smart artificial intelligence tools such as OCR can break such protection mechanisms. The proposed security can defend from such attacks because it uses the FP biometric verification method, which cannot be broken by brute force attacks. Moreover, the framework has a special property that detects such attacks as a three-time login fail blocks the IoT device by its MAC address until it is released by the administrator.

Stolen Token Attack
The reuse of stolen token can provide the hackers access privilege as the authenticated client or IoT device, and this can lead to severe damage in a critical environment, such as IoT networks. The proposed IoT security framework prevents such attack type by using identity verification information injected inside the enhanced token.

5.2. Comparative Analyses Between Different IoT Frameworks and the Proposed One

Table 1 compares previous IoT frameworks and the proposed one in terms of security techniques used and the extra load added to the IoT devices.

Table 1 Security techniques used in the framework

Testing criteria	Author Name						
	Trnka and Cerny	Polat, H.et al	Sciancalepore et al	Claeys et al	Oh,Kim and Cho	A.alrefai	Proposed Framework
Protect Token	NO	NO	Yes	Yes	No	No	Yes
Verify cloud service Identity	Yes	Yes	Yes	Yes	Yes	No	Yes
Use Time Stamp	NO	NO	NO	No	No	Yes	Yes
Optimize load over IoT devices	NO	Yes	No	No	Yes	NO	Yes
Use Biometric Verification	NO	NO	NO	No	No	Yes	Yes
Use Asymmetric Encryption	NO	NO	Yes	Yes	No	Yes	Yes

doi:10.20944/preprints202208.0188.v1

Use symmetric encryption	NO	NO	Yes	Yes	Yes	Yes	Yes

Table 2 compares the different types of attack defense capability of previous IoT security frameworks and the proposed one.

Table 2 Attacks prevention comparison

Attack Type	Author Name						
	Trnka and Cerny	Polat, H.et al	Sciancalepore et al	Claeys et al	Oh,Kim and Cho	A.alrefai	Proposed Framework
Replay attack	NO	NO	Yes	Yes	No	No	Yes
Brute force attack	Yes	Yes	No	No	No	Yes	Yes
Man in the Middle Attack	No	No	Yes	Yes	Yes	Yes	Yes
Phishing Attack	NO	NO	Yes	Yes	Yes	Yes	Yes
Token Reuse attack	No	No	Yes	Yes	No	No	Yes

Table 3 illustrates the execution time overhead added on the IoT protocols for the types of cryptography methods used in the proposed framework. The results show low overhead for the token validation and the symmetric encryption/decryption processes, which means the proposed framework does not add high redundant computational time because the rest of the processes are used in very low frequencies and performed on the cloud service side.

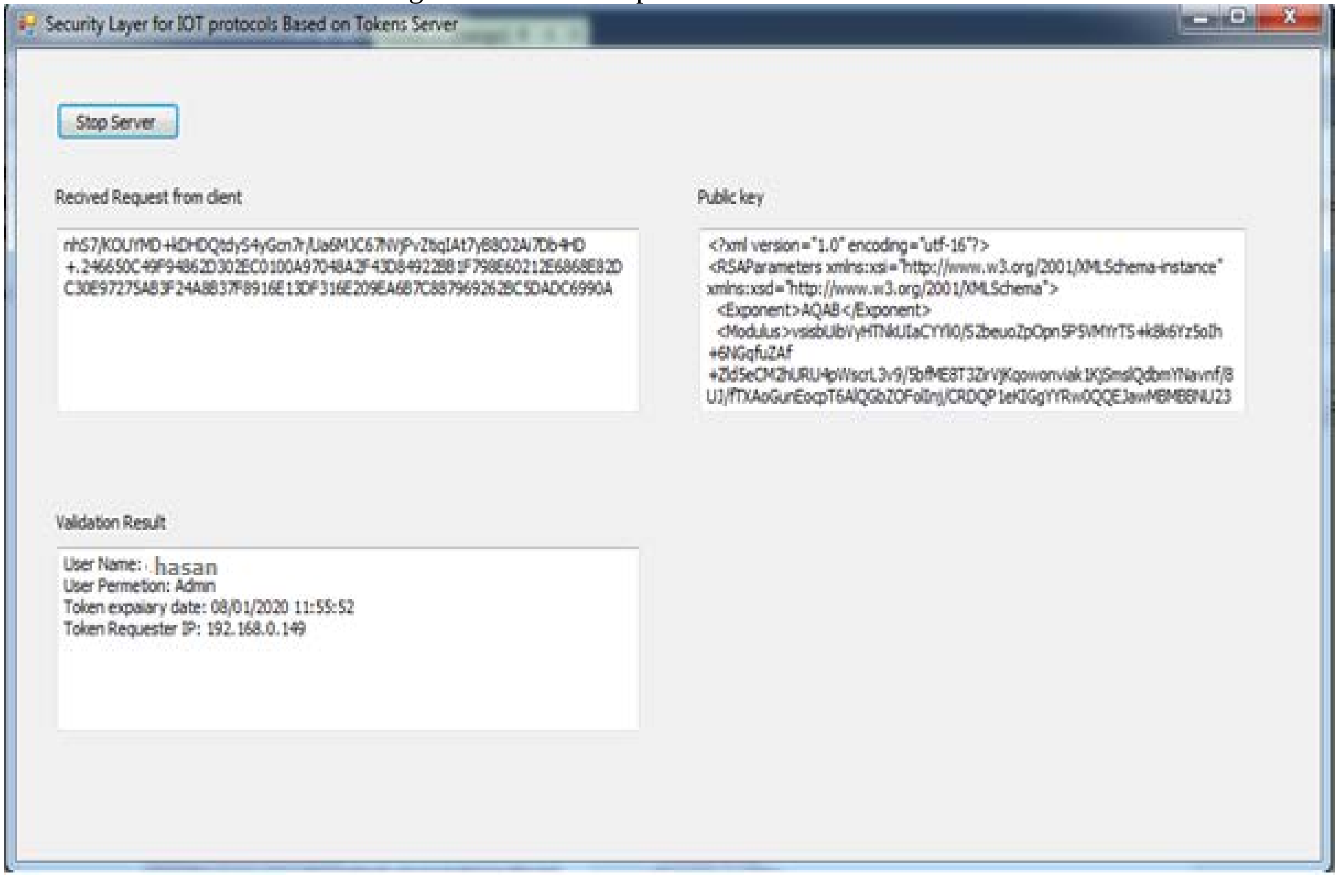
Table 3 CRYPTOGRAPHIC OVERHEAD (MS)

Cryptography Criteria	CRYPTOGRAPHIC OVERHEAD (ms)	
	Proposed framework	
	Encrypt /Sign	Decrypt/Verify
Symmetric encryption (AES 128)	0.062	0.132
Asymmetric encryption	126	197
HASH (SHA 256)	0.322	NA
Token Generation	167	213

Key generation (Server side)	690	Na
Key generation (AES 128) Client, IoT side	96	Na

5.3. Simulation Test Using the Developed Windows Application

A Windows application is developed to simulate the IoT framework phases to test the attack prevention performance and extract the execution time overhead, as shown in Table 3. Figure 15 shows the cloud server simulation with the initiated PK and request received from the client side. Figure 12 shows the simulation application receiving the request with the authentication token, the token validation, and the information extracted from the token containing the authorization permissions.


Security Layer for IOT protocols Based on Tokens Server
Stop Server
Recived Request from client
Public key
Validation Result

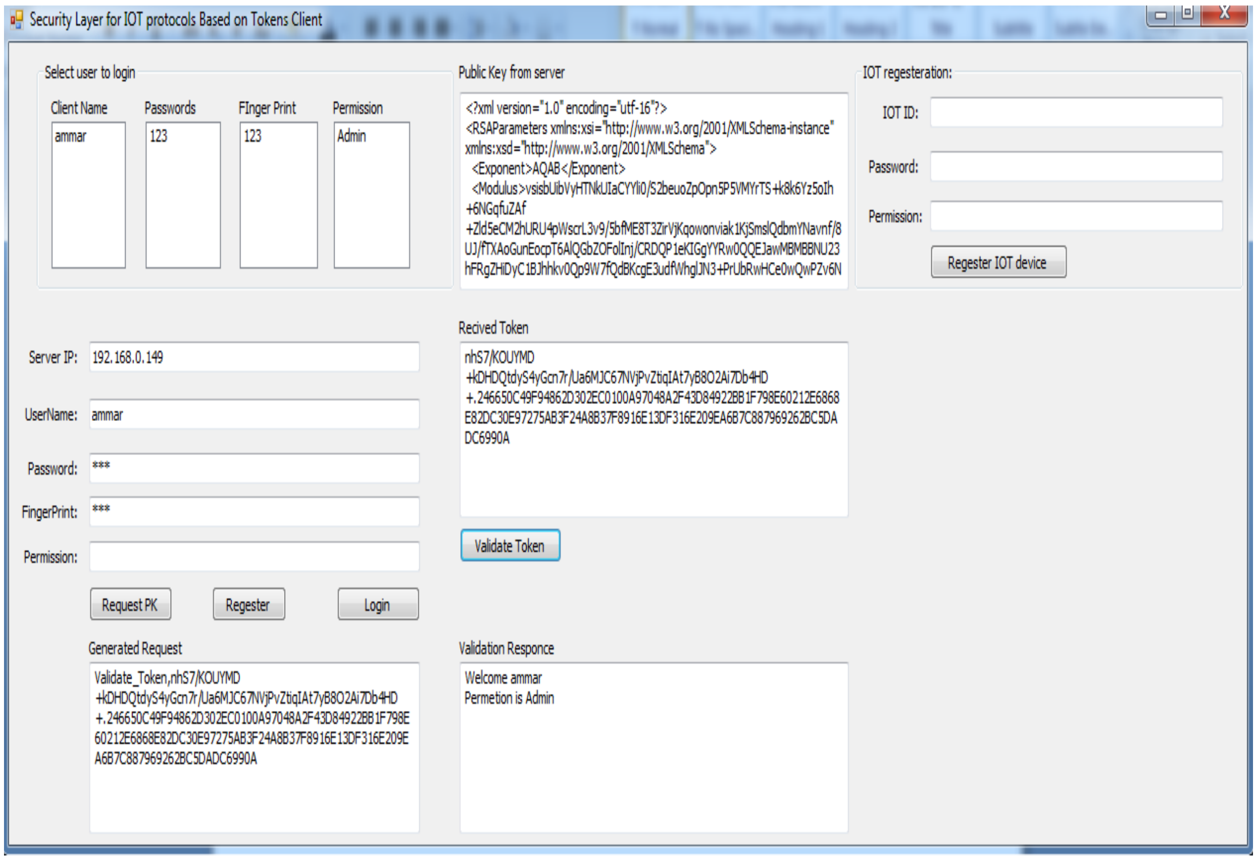
```
nhS7/KOUYMD+kDHDQtdyS4yGcn7r/Ua6MJC67NVjPvZtiqIAt7yB8O2Ai7Db4HD
+.246650C49F94862D302EC0100A97048A2F43D8492288 1F798E60212E6868E82D
C30E97275A83F24A8B37F8916E13DF316E209EA687C887969262BC5DADC6990A
```

```
<?xml version="1.0" encoding="utf-16"?>
<RSAParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Exponent>AQAB</Exponent>
  <Modulus>vsisbUibVyHTNkUIaCYYl0/S2beuoZpOpn5P5VMYrTS+k8k6Yz5oIh
+6NGqfuZAf
+Zid5eCM2hURU4pWscrL3v9/5bfME8T3ZrVjKqowonviak1KjSmslQdbmYNavnf/8
UJ/fTXAoGunEocpT6AlQGbZOFolInj/CRDQP1eKIGgYYRw0QQEJawMBMBBNU23
```

```
User Name: hasan
User Permetion: Admin
Token expaiary date: 08/01/2020 11:55:52
Token Requester IP: 192.168.0.149
```

Figure 1: Cloud service side of the proposed security framework simulation

Figure 13 shows the simulation of the client-side proposed framework operation. The simulation shows the received PK, the registration phase, and the received token from the server after a successful login. The figure also shows the requests generated by the framework to be sent to the cloud service.


Security Layer for IOT protocols Based on Tokens Client

Select user to login

Client Name	Passwords	FInger Print	Permission
ammar	123	123	Admin

Public Key from server

```
<?xml version="1.0" encoding="utf-16"?>
<RSAParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Exponent>AQAB</Exponent>
  <Modulus>vsisbUibVyHTNkUIaCYYli0/S2beuoZpOpn5P5VMYrTS+k8k6Yz5oIh
+6NGqfuZAf
+Zld5eCM2hURU4pWscrL3v9/5bfME8T3ZirVjKqowonviak1KjSmslQdbmYNavnf/8
UJ/fTXAoGunEocpT6AlQGbZOFolInj/CRDQP1eKIGgYYRw0QQEJawMBMBBNU23
hFRgZHiDyC1BJhhkv0Qp9W7fQdBKcgE3udfWhglJN3+PrUbRwHCe0wQwPZv6N
```

IOT regesteration:

IOT ID:

Password:

Permission:

Server IP: 192.168.0.149

UserName: ammar

Password: ***

FingerPrint: ***

Permission:

Recived Token

```
nhS7/KOUYMD
+kDHDQtdyS4yGcn7r/Ua6MJC67NVjPvZtiqIAt7yB8O2Ai7Db4HD
+.246650C49F94862D302EC0100A97048A2F43D84922BB1F798E60212E6868
E82DC30E97275AB3F24A8B37F8916E13DF316E209EA6B7C887969262BC5DA
DC6990A
```

Generated Request

```
Validate_Token,nhS7/KOUYMD
+kDHDQtdyS4yGcn7r/Ua6MJC67NVjPvZtiqIAt7yB8O2Ai7Db4HD
+.246650C49F94862D302EC0100A97048A2F43D84922BB1F798E
60212E6868E82DC30E97275AB3F24A8B37F8916E13DF316E209E
A6B7C887969262BC5DADC6990A
```

Validation Responce

Welcome ammar
Permetion is Admin

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 August 2022 doi:10.20944/preprints202208.0188.v1

Figure 2 Client side of the proposed security framework simulation

As mentioned before, a Windows applications is developed to simulate the phases of the proposed security framework and the methods and algorithms used inside each phase, test and evaluate the proposed security framework performance through its normal operation, test the framework behavior against several attacks such as man-in-the-middle attack, reply attack, and brute force attack, and verify the internal methods' execution time and complexity.

The developed Windows application is split into two main categories. The first category is the server-side application, which simulates all the operations inside each phase of the proposed security framework related to the cloud service side. The second category is the client-side and the IoT-side Windows application. The second category shows the functions related to the client and the IoT side. Figure 14 shows the input and output diagram of the server-side Windows application.

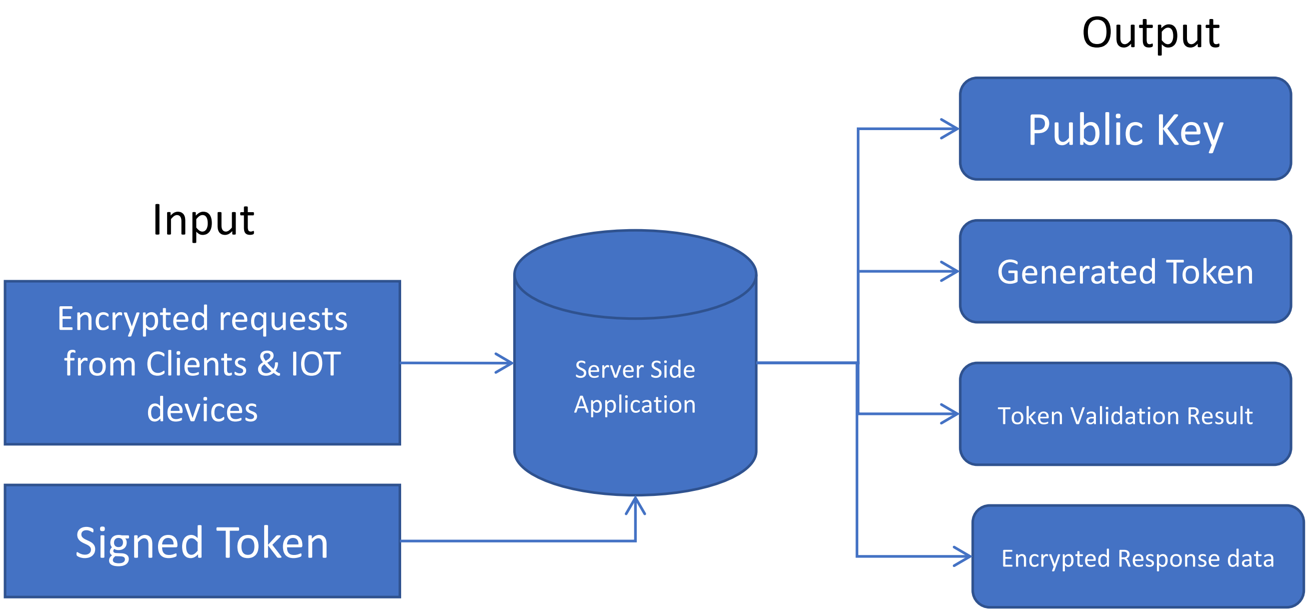
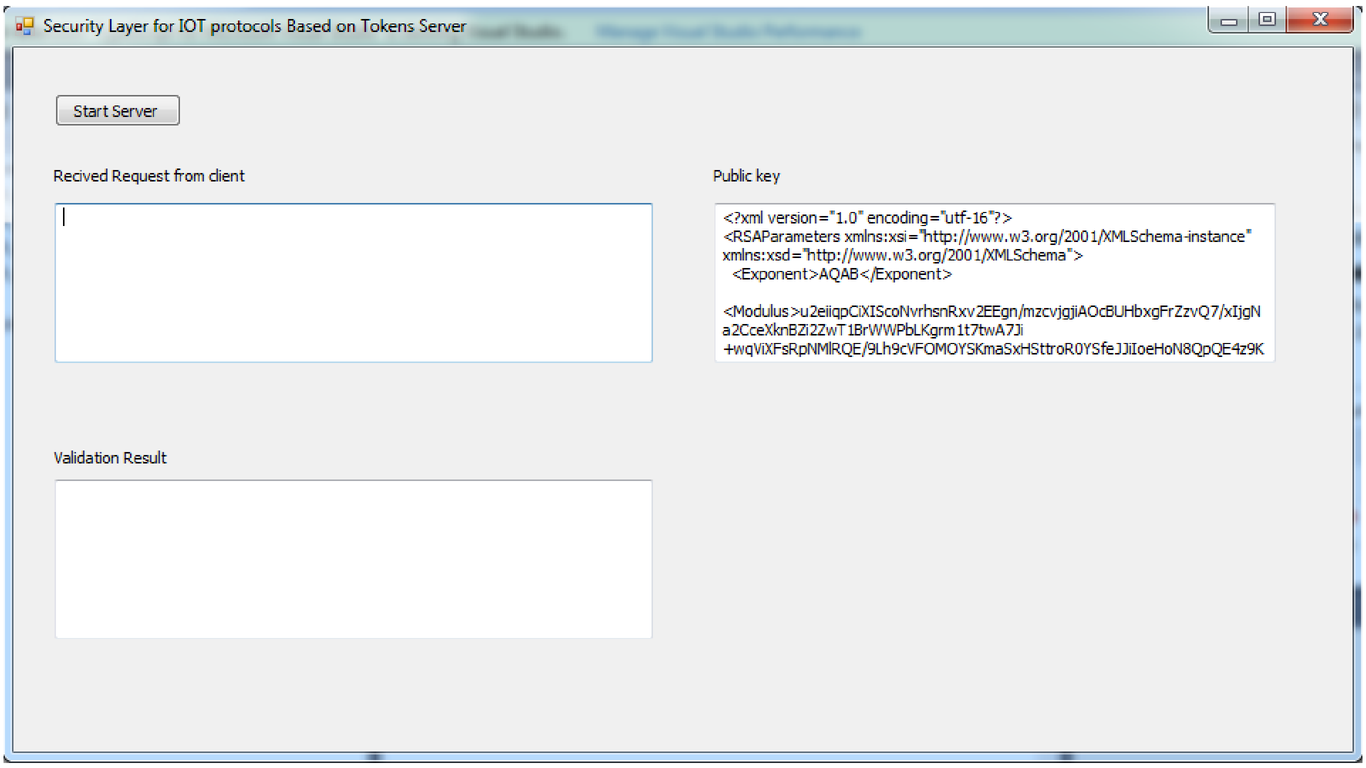


Figure 34: inputs and output diagram for the developed server-side IoT security framework

Figure 14 shows that the server-side application's main functionality is to respond to any request coming from the client and the IoT devices. The inputs of the server-side application are the received encrypted requests and the signed token, whereas the outputs of the server-side application can be classified into the public encryption key from the server, the generated token for the logged-in client or IoT device, the token validation result in case of validation failure, and the encrypted response data or commands from the server side, combined with the TS.

The operation of the server-side application starts by implementing the first phase of the proposed IoT security framework, namely, generating the PK and P_RKs for initial data exchange with the clients and the IoT devices during the registration and login stages. The application also generates the preshared key for encrypting the payload of the token and an extra secret key for hash generation used in signing the encrypted payload inside the token.

The PKs and P$_R$Ks are generated using the Rivest–Shamir–Adleman asymmetric algorithm. Moreover, the symmetric encryption algorithm used is the Advanced Encryption Standard (AES) 128-bit key. AES is used because it is much more secure than the Data Encryption Standard (DES) or 3DES given that it uses a 128-bit key instead off 64 bit key in DES. Moreover, the AES 128 is still a relatively considered lightweight encryption algorithm. In addition, SHA-512 cryptographic algorithm is used for hash code generation. The server-side application then initiates socket listeners to start receiving requests from the client-side application or the IoT side application. Figure 15 shows the server-side Windows application implementation.


Security Layer for IOT protocols Based on Tokens Server
Start Server
Recived Request from client
Public key
Validation Result

```
<?xml version="1.0" encoding="utf-16"?>
<RSAParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Exponent>AQAB</Exponent>

<Modulus>u2eiiqpCiXIScoNvrhsnRxv2EEgn/mzcvjgjiAOcBUHbxgFrZzvQ7/xIjgN
a2CceXknBZi2ZwT1BrWWPbLKgrm1t7twA7Ji
+wqViXFsRpNMlRQE/9Lh9cVFOMOYSKmaSxHSttroR0YSfeJJiIoeHoN8QpQE4z9K
```

Figure 15: Cloud service side of the proposed security framework simulation.

6. Conclusion

In this paper, an enhanced IoT security framework for authentication and authorization is proposed and developed. The IoT security framework uses an enhanced token authentication method that expands the standard token by adding identity verification information and authorization permission level inside the token payload data. The token payload is then encrypted by the cloud service to prevent stolen token data from being compromised by hackers. The security framework restricts the token validation only on the cloud service side only. The cloud server requests to the IoT devices are authenticated using a smart technique based on an encrypted modified TS that reflects the real time of the request sent from the cloud service to the IoT device. The TS is generated from the cloud service using the preshared key received securely via asymmetric encryption from the IoT device during the login phase, which only exists inside the encrypted payload data of the token. Therefore, it cannot be decrypted by anyone but the cloud service itself. The IoT device can validate any request by decrypting the received requests using its preshared key only using this technique. Therefore, it ignores any misencrypted data using man-in-the-middle attack. Moreover, reply attacks using stolen requests from the server will fail due to the TS embedded in the request that will be ignored if expired by the IoT device.

The proposed protocol also uses FP biometric encryption to increase authentication security level and prevent brute force attacks.
The framework is evaluated using a security analysis that justifies each property and defense technique used to prevent the issues mentioned in the problem statement section, especially the types of attacks such as man-in- the-middle attack, reply attack, and brute force attack. Another evaluation is conducted using a developed Windows application that simulates the proposed IoT security framework, which is used to calculate the time load added to the IoT protocols because of the proposed framework. Moreover, real testing of attack types is deployed over the developed simulation to test security framework security efficiency. The testing results show that the proposed security framework protects IoT protocols and networks from different types of attacks while adding a very low load over the IoT device.

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References:

Choudhury, A. J., Kumar, P., Sain, M., Lim, H., & Hoon, J. L. (2011). A strong user authentication framework for cloud computing. Proceedings - 2011 IEEE Asia-Pacific Services Computing Conference, APSCC 2011, 110–115. https://doi.org/10.1109/APSCC.2011.14

Claeys, T., Rousseau, F., & Tourancheau, B. (2018). Securing Complex IoT Platforms with Token Based Access Control and Authenticated Key Establishment. Proceedings - 2017 International Workshop on Secure Internet of Things, SIoT 2017, 1–9. https://doi.org/10.1109/SIoT.2017.00006

DE Donno M., KOEN TANGE , & NICOLA DRAGONI.(2019). Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. IEEE Access, Digital Object Identifier 10.1109/ACCESS.2019.2947652

Deogirikar, J., & Vidhate, A. (2017). Security attacks in IoT: A survey. Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017, 32–37. https://doi.org/10.1109/I-SMAC.2017.8058363

H Al-Refai, A Al-Awneh, K Batiha, AA Ali, YME Rahman (2011). Efficient Routing Leach (Er-Leach) Enhanced On Leach Protocol In Wireless Sensor Networks. International Journal of Academic Research 3 (3).

Hasan Al-Refai, Khaldoun Batiha, Ahmad M. Al-Refai. AN ENHANCED USER AUTHENTICATION FRAMEWORK IN CLOUD COMPUTING. International Journal of Network Security & Its Applications (IJNSA) Vol. 12, No.2, March 2020

H Al-Refai, A Alawneh, KA Jarah (2014). Enhanced model of Payment Phase for SET Protocol. Citeseer.

H Al-Refai, A Alawneh, K Batiha (2014). A COMPARATIVE STUDY IN WIRELESS SENSOR NETWORKS. International Journal of Wireless & Mobile Networks 6 (1), 61.

H Al-Refai, K Batiha, A Alwaneh, SB Hani (2014). Improved SPI Calculus for Reasoning on Cryptographic Protocols. International Journal of Video&Image Processing and Network Security IJVIPNS.

Oh, S. R., Kim, Y. G., & Cho, S. (2019). An interoperable access control framework for diverse IoT platforms based on oauth and role. Sensors (Switzerland), 19(8). https://doi.org/10.3390/s19081884

Pawani Porambage , Student Member, Jude Okwuibe, Student Member, IEEE, Madhusanka Liyanage, Member, IEEE, Mika Ylianttila, Senior Member, IEEE, and Tarik Taleb , Senior Member, IEEE (2018). Survey on Multi-Access Edge Computing for Internet of Things Realization. IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 20, NO. 4, FOURTH QUARTER.

Polat, H., & Oyucu, S. (2017). Token-based authentication method for M2M platforms. Turkish Journal of Electrical Engineering and Computer Sciences, 25(4), 2956–2967. https://doi.org/10.3906/elk-1608-6

Sciancalepore, S., Piro, G., Caldarola, D., Boggia, G., & Bianchi, G. (2017). OAuth-IoT: An access control framework for the Internet of Things based on open standards. Proceedings - IEEE Symposium on Computers and Communications, 676–681. https://doi.org/10.1109/ISCC.2017.8024606

SA Aljawarneh, A Alawneh, R Jaradat (2017). Cloud security engineering: Early stages of SDLC. Future Generation Computer Systems 74, 385-392.

Song, W., & Su, X. (2011). Review of Mobile cloud computing. 2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011, 1–4. https://doi.org/10.1109/ICCSN.2011.6014374

statista 2021. IoTdevs.https://www.statista.com/statistics/471264/IoT-number-of-connected-devices- worldwide/. Accessed: 2021-21-4.

Trnka, M., & Cerny, T. (2017). Authentication and Authorization Rules Sharing for Internet of Things. Software Networking, 2017(1), 35–52. https://doi.org/10.13052/jsn2445-9739.2017.003

Wardana, A. A., & Perdana, R. S. (2018). Access control on internet of things based on publish/subscribe using authentication server and secure protocol. Proceedings of 2018 10th International Conference on Information Technology and Electrical Engineering: Smart Technology for Better Society, ICITEE 2018, 118–123. https://doi.org/10.1109/ICITEED.2018.8534855