HOKKAIDO UNIVERSITY

Title	Privacy Preserved IoT Data Inquiry System using DoT
Author(s)	Sagara, Hayato; Jin, Yong; Iida, Katsuyoshi et al.
Description	2024 7th Conference on Cloud and Internet of Things (CIoT). 29-31 October 2024. IEEE. Montreal, QC, Canada.
Citation	Cloudification of the Internet of Things (CIoT), 1-6 https://doi.org/10.1109/CIoT63799.2024.10757083
Issue Date	2024-11-22
Doc URL	https://hdl.handle.net/2115/94105
Rights	© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Type	conference paper
File Information	sagara_CIoT_2024.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

Privacy Preserved IoT Data Inquiry System using DoT

Hayato Sagara
Graduate School of
Information Science and Technology,
Hokkaido University,
Kita 11, Nishi 5, Kita-ku,
Sapporo-shi, 060-0811, Japan.
Email: sagara.hayato.y1@elms.hokudai.ac.jp

Yong Jin
Global Scientific Information
and Computing Center,
Tokyo Institute of Technology,
2-12-1 O-okayama, Meguro-ku,
Tokyo, 152-8550, Japan.
Email: yongj@gsic.titech.ac.jp

Katsuyoshi Iida
and Yoshiaki Takai
Information Initiative Center,
Hokkaido University,
Kita 11, Nishi 5, Kita-ku,
Sapporo-shi, 060-0811, Japan.
Email: {iida,ytakai}@iic.hokudai.ac.jp

***Abstract*—Privacy preservation has become one of the strict security requirements in some Internet services nowadays, especially in IoT-based e-health systems. This paper proposes a privacy-preserved IoT data inquiry system with DNS, one of the most scalable distributed databases on the Internet. In the proposed system, all IoT data is encrypted and transmitted over DNS payload via DNS over TLS (DoT) communication channels for the end user's inquiry. Moreover, the communication with IoT devices uses dedicated protocols, such as CoAP and MQTT, and the IoT data transmission uses DNS protocol only among IoT gateway, DNS servers, and end terminals that run client applications inquiring about IoT data. Consequently, the proposed system can preserve user privacy even if IoT data is stored in a cloud system and transmitted via the Internet. We implemented a prototype system in a local experimental network environment, confirming that the text-based IoT data was properly encrypted before transmitting the data and able to decrypt the inquiry on the end terminal as designed. Future work includes the extensions of the proposed system for handling binary-based IoT data transmission using DoT and performance evaluations in an actual network environment.**

Keywords: Privacy Preservation, Internet of Things, IoT, Domain Name System, DNS, DNS over TLS, and DoT.

I. Introduction

Internet of Things (IoT) technologies have been developed over a decade, enabling real-time data generation and seamless transferring between devices. According to the Statista report [1], the amount of IoT devices connected to the Internet was about 15 billion in 2023, projected to be about 30 billion by 2030. However, this rapid growth of IoT networks intensified cyber security concerns. According to the Statista report in 2023 [2], the number of attacks has increased yearly, suggesting a potentially worsening trend in the future.

Privacy is also crucial in IoT systems. IoT devices have characteristics of edge devices. Therefore, they handle more sensitive user data (like location and biometric information) than conventional devices. If attackers collect the data on IoT devices for an extended period, they may be able to specify the user's tendencies or preferences. According to the research on privacy threats in IoT by Torre et al. [3], privacy threats can be classified into six types. For example, the "Inventory" threat infers information about device configuration in an IoT system or user's preferences by unauthorized access or observing connection. Also, the "Profiling" threat infers the user's tendencies or preferences based on collected data from IoT networks and devices. In light of these potential threats, it is important to implement privacy protection measures in IoT systems as well as in IoT data transmission on the Internet.

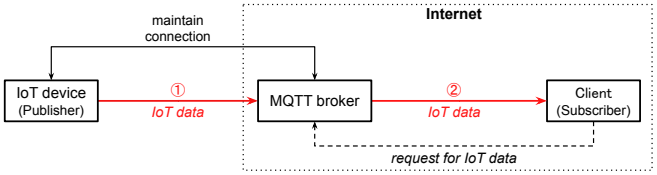


Fig. 1. Fundamental overview of IoT data inquiry system with MQTT

Existing IoT devices struggle to address security and privacy concerns due to their physical and runtime constraints. Limited resources and real-time processing requirements impede their ability to execute additional protective measures. This paper proposes a privacy-preserving IoT inquiry system that maintains compatibility with conventional IoT communication protocols, such as Message Queuing Telemetry Transport (MQTT) [4] and Constrained Application Protocol (CoAP) [5], between IoT devices and gateways. We leverage the Domain Name System (DNS), one of the Internet's most scalable distributed databases, as the carrier for IoT data inquiries. Furthermore, we encrypt all IoT data using Pretty Good Privacy (PGP) [6] and establish a secure communication channel via DNS over TLS (DoT) [7] for end-user IoT data queries. This approach protects user privacy during data inquiries and enhances data security, even when stored in cloud systems.

The rest of this paper is organized as follows. Section II describes related work and existing problems, followed by the description of the proposed system in Section III. Then, Section IV describes the prototype implementation, followed by the evaluations described in Section V. Finally, Section VI concludes this paper and shows the future work.

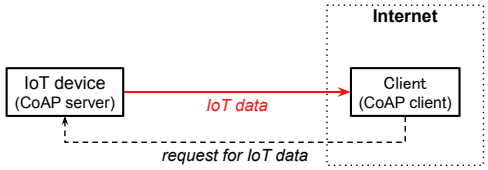


Fig. 2. Fundamental overview of IoT data inquiry system with CoAP

TABLE I
ACHIEVEMENT AND PRIMARY PROBLEMS OF RELATED WORKS

System	Achievement	Primary problems
MQTT	Fetching data in real-time via a broker with TCP	Lack of scalability
CoAP	Lightweight by using UDP	Unable to access IoT devices from the Internet
Ref. [8]	Using authentication	Not encrypting data and communications
Ref. [9]	Encrypting communications	· Not encrypting data · Lack of scalability
Ref. [10]	· Encrypting data · Using authentication · Improving scalability by DNS	· Not encrypting communications · Lack of name resolution process

II. Related Research and Technologies

A. Basic IoT data inquiry systems and existing problems

MQTT and CoAP are lightweight communication protocols widely utilized in IoT networks for IoT devices. Figures 1 and 2 show the overview of IoT data inquiry systems using MQTT and CoAP, respectively.

In MQTT, IoT devices initially establish and maintain a connection with the corresponding MQTT broker. Secondly, each IoT device sends IoT data to the MQTT broker with a specified topic (arrow number 1 in Fig. 1). Then, an end client accesses the MQTT broker and inquires about the IoT data. Lastly, the MQTT broker sends the inquired IoT data to the end client (arrow number 2 in Fig. 1).

In CoAP, each IoT device functions as a CoAP server, storing IoT data in a specified resource. When an end client requires information, it directly accesses the IoT device and requests the stored data. The IoT device then responds by transmitting the requested data to the end client.

Each inquiry system has its disadvantages. An MQTT-based system faces scalability issues, as IoT devices must maintain constant TCP connections with the MQTT broker, increasing the broker's workload as the number of devices grows. In contrast, a CoAP-based system poses a risk of intrusion into the home network. Unlike MQTT, end clients retrieve data directly from IoT devices without passing through a broker. This direct access to IoT devices introduces significant vulnerabilities in a remote IoT data inquiry system.

B. Related work on IoT data inquiry system

1) IoT systems for smart home networks: Paraschiv et al. [8] designed a remote monitoring system for senior citizens using IoT sensors and a cloud network. This system transmits IoT data from sensors to an IoT gateway via TCP or a cloud

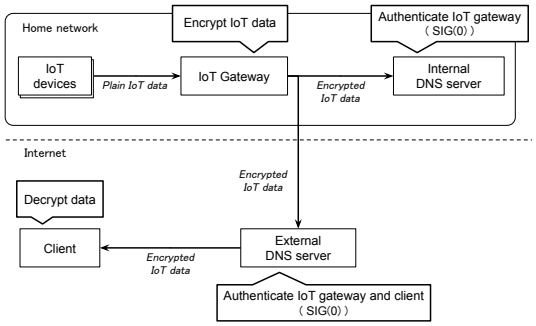


Fig. 3. Overview of a DNS-based IoT data inquiry system

database using the corresponding API. The data is accessible via the Internet, and OAuth 2.0 [11] is used for authentication. However, this system has security and privacy risks because it neither encrypts IoT data nor uses encrypted communications for data transmission over the Internet.

Arvandy et al. [9] designed a secure platform with communication protection for IoT devices in smart home networks. In this system, IoT devices encrypt data using a symmetric key shared with an IoT gateway. Each device and the gateway have unique IDs and exchange symmetric keys via Elliptic-curve Diffie-Hellman (ECDH) to prevent man-in-the-middle attacks. Additionally, a cloud-based server maintains a connection with the IoT gateway, and end users communicate with the server using the HTTPS protocol for IoT data inquiries. However, the IoT gateway's workload and communication with IoT devices can increase, as the gateway must exchange keys with all devices to encrypt data. Consequently, scalability becomes a challenge as the number of IoT devices grows. Moreover, the system remains vulnerable to data leakage during transmission and inquiry processes because the IoT data is not encrypted.

2) DNS-based IoT data inquiry system: Jin et al. [10] proposed an IoT data inquiry system by practically using DNS. Figure 3 illustrates an overview of the system. This system separates the network containing IoT devices in a home network from the Internet. The home network consists of an IoT gateway and an internal authoritative DNS server, while the Internet side includes end clients and an external authoritative DNS server. The internal DNS server handles inquiries within the home network, while the external DNS server processes inquiries from end clients on the Internet. In this setup, IoT devices and the IoT gateway can continue to use conventional communication protocols, such as MQTT and CoAP, without modification for IoT data exchange. This approach maintains compatibility with existing IoT communication protocols while enhancing the security and privacy of data inquiries.

The flow of IoT data inquiry is described in the following. First, an IoT gateway retrieves plain text IoT data from an IoT device with a conventional communication protocol, then encrypts the IoT data with PGP and encodes it with Base64. After that, the IoT gateway registers the data as a TXT record to the internal and external DNS servers. At the time, two DNS servers authenticate the IoT gateway with SIG(0) [12], a public

key authentication of the DNS extension mechanism. Next, an end client inquiries the TXT record to the external DNS server with SIG(0), retrieving encrypted IoT data. Finally, the end client transforms the encrypted data into plain data by decrypting it with PGP and decoding it with Base64.

In this system, the IoT gateway encrypts IoT data and handles authentication on behalf of the IoT devices. The internal and external DNS servers manage the IoT data. This architecture minimally impacts the performance of the IoT devices in the inquiry system, as they only need to send data. Furthermore, the system leverages DNS, a well-established protocol for large-scale name resolution on the Internet. The system also incorporates DNS-based authentication features to enhance security. Moreover, DNS improves data management flexibility by allowing arbitrary domain assignments to individual IoT devices.

This system meets three requirements: authentication of IoT devices, authentication of clients, and end-to-end encryption of IoT data. However, it does not address communication encryption, resulting in a risk of privacy leakage. Additionally, there is an issue with name resolution using conventional DNS, as the mechanism for automatic name resolution between end clients and the external server is absent. Consequently, end clients must query the external server directly using its IP address.

Above all, IoT data inquiry systems need to meet the following requirements to provide security measures and privacy preservation.

1) Authentication of IoT devices (or IoT gateway)
2) Authentication of clients
3) End-to-end encryption of IoT data
4) Encryption of communication

Table I summarizes the comparison of the previous works introduced in this section.

In this paper, we extend the previous research by adding a communication encryption feature using DoT to mitigate the privacy leakage risk by sniffing.

III. Proposed System

A. Overview of the proposed system

As described in Sec. I, the security and privacy requirements in an IoT data inquiry system are crucial and must be met. In this section, we first summarize the system's requirements before designing the system.

- IoT gateways should authenticate IoT devices.
- Clients should be authenticated.
- IoT data should be encrypted to protect the data itself.
- Communication should be encrypted to protect privacy and data leakage.

We use the DNS protocol to meet the above requirements because of its state-of-the-art enhancements: TSIG [13] and SIG(0) for authentication and DoT for encryption.

The IoT data inquiry system by Jin et al. described in Section II-B2 has a privacy leakage risk and low practicality due to a lack of encryption of communication and name

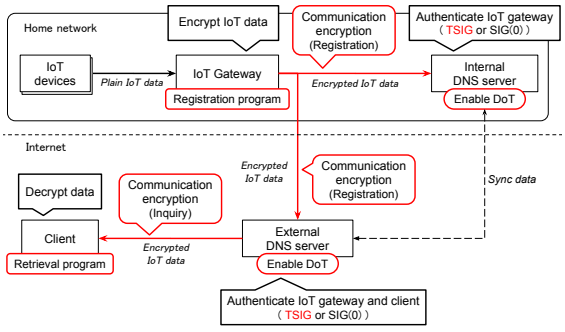


Fig. 4. Overview of the proposed IoT Data Inquiry System

resolution to the external DNS server. Therefore, the proposed system in this paper improves its system like the design shown in Figure 4. First, the proposed system enhances privacy protection by encrypting communication from the IoT gateway with DoT. Also, it improves practicality or convenience by adding TSIG besides SIG(0) for authentication. Furthermore, we adopt three DNS servers (the group of DNS servers for name resolution), a cache server, a root DNS server, and a TLD server between clients and the external DNS server. The clients can retrieve the IP address of the external DNS server with a recursive query to the cache server.

1) IoT device and IoT gateway: IoT devices generate plain text data and transmit it with conventional protocols such as MQTT and CoAP. The IoT gateway performs the following processes to register encrypted IoT data. First, it awaits data generation from each IoT device. Next, it retrieves the data from the IoT device and conducts PGP encryption and Base64 encoding. Then, it stores the processed data in a TXT resource record (TXT RR). Finally, it registers the TXT RR to the internal and external DNS servers using DoT with authentication by TSIG or SIG(0).

2) Internal DNS server and external DNS server: The internal DNS server is deployed in the home network, while the external DNS server is set on the Internet. Both DNS servers accept TXT RR registration requests from the IoT gateway with device authentication by TSIG or SIG(0). Moreover, the external DNS server accepts TXT RR queries from the clients. In inquiry, the external DNS server only allows DoT communication and authenticates clients with TSIG or SIG(0).

IV. Implementation

This section describes the implementation of the proposed system. Each component runs on the Ubuntu operating system except for the clients. BIND9 [14] was used for each DNS server, and libcoap [15] was used for IoT devices and IoT gateway. Additionally, the Go programming language [16] and GnuPG software [17] were used in the registration and retrieval programs shown in Fig. 4. Subsections IV-D and IV-E explain the details of each program.

A. Implementation of IoT device and IoT gateway

In this research, we utilize CoAP for communication between IoT devices and the IoT gateway. In addition, we reproduce their behavior on Ubuntu because of not using physical

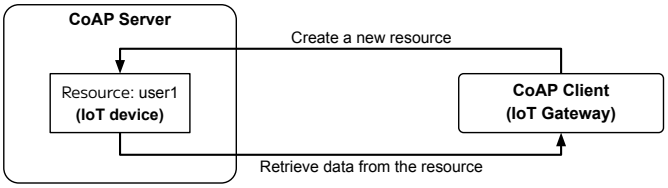


Fig. 5. Implementation overview of IoT device and gateway with CoAP

devices. Therefore, we implement each resource created on the CoAP server as an IoT device and the CoAP client as an IoT gateway. Figure 5 shows the implementation overview with CoAP. The CoAP client creates a new resource on the CoAP server beforehand. Then, the CoAP server stores IoT data in the resource. Finally, the CoAP client requests and retrieves data from the resource. The CoAP server is launched by "coap-server" command, and the execution of the CoAP client is described in Section IV-D.

B. Configuration of domains and keys

It is necessary to map IoT devices with their registered IoT data to retrieve desired data from the internal or external DNS server. Therefore, we assign each IoT device a unique domain name. The IoT gateway registers TXT RRs with the corresponding domain name of the IoT device, and the clients query the respective domain names to obtain the TXT RRs.

In this research, we created 1000 household domain names under the parent domain "exmaple.com.", such as "home1.example.com." and "home2.example.com.". Then, we made ten subdomains for each household domain, such as "IoTdevice1" and "IoTdevice2". IoT devices are assigned each subdomain of each household domain (e.g., "IoTdevice1.home1.example.com.") without duplication.

The keys of PGP, TSIG, and SIG(0) are created individually for each household domain name. Moreover, the keys of TSIG and SIG(0) are made for the IoT gateway and clients.

C. Settings of internal DNS server and external DNS server

This subsection describes the main settings of the internal DNS and external DNS servers. We apply the following settings to both. However, the parent domain "example.com." does not apply authentication settings because the cache server cannot retrieve SOA and A RRs from the external DNS server when authentication is enabled.

- In the "tls block", describe the Certification Authority (CA) for DoT. In this research, we utilize a private CA.
- In the "options block", enable DoT with port 853.
- Specify the TSIG key.
- In the "zone block" for each household domain name, specify the keys of TSIG and SIG(0) in the "allow-update clause" to enable authentication of the IoT gateway.

Furthermore, for the external DNS server, we apply the following additional settings:

- In the "zone block" for each household domain name, configure TSIG and SIG(0) in the "allow-query clause" to enable client authentication.

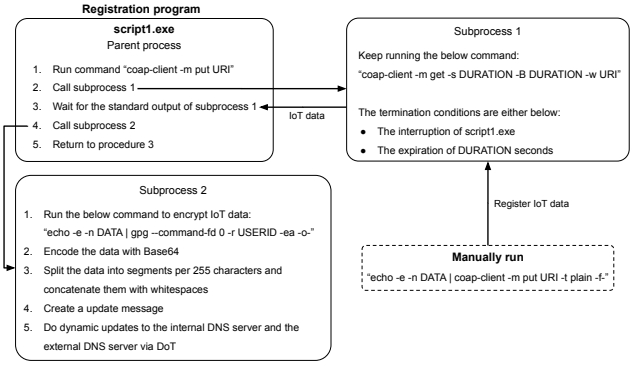


Fig. 6. The processing steps of the registration program (script1.exe)

D. Implementation of the registration program

The registration program consists of a built Go language program "script1.exe". The IoT gateway uses this program to retrieve data from IoT devices and register data to the DNS servers. In the sequence of operations, the program uses the external package dns [18] for DNS-related processes. Additionally, the program uses a coap-client command of libcoap for CoAP client operations. Furthermore, the program uses GnuPG to process encryption with PGP. Provided, GnuPG loads the public key for encryption beforehand.

Figure 6 illustrates the processing steps of the registration program, and the following describes the details.

1) The parent process executes the command "coap-client -m put URI" to create a new resource on the CoAP server (IoT device). The URI is in the format "coap://[CoAP server's IP address]/resource name".
2) The parent process executes the command "coap-client -m get -s DURATION -B DURATION -w URI" in a separate process (subprocess 1) to sequentially retrieve the data of the modified resource and output its contents.
3) Manually execute the command "echo -e -n DATA | coap-client -m put URI -t plain -f-" to register IoT data to the specified resource. Here, DATA is the string that is registered as IoT data.
4) The parent process reads the output of subprocess 1 to retrieve IoT data. It then launches another process (subprocess 2) and returns to waiting for standard output.
5) Subprocess 2 executes the command "echo -e -n DATA | gpg –command-fd 0 -r USERID -ea -o-" to encrypt the IoT data with PGP.
6) Subprocess 2 performs Base64 encoding on the PGP message. Then, it splits encoded data into segments per 255 characters and concatenates with white space because the string length in one TXT RR is limited to 255 characters [19].
7) Subprocess 2 signs the update message with TSIG or SIG(0) and then sends it to the internal and external DNS servers over DoT. In a TLS negotiation for DoT, the verify function of the Go language standard package x509 [20] verifies the certificate.

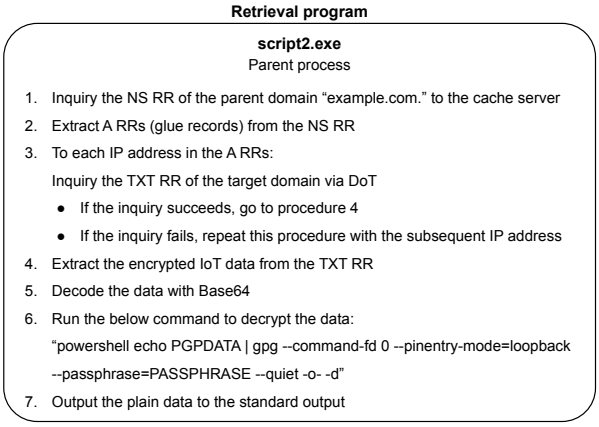


Fig. 7. The processing steps of the retrieval program (script2.exe)

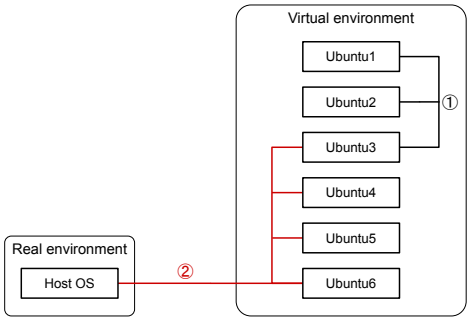


Fig. 8. Network configuration in the virtual environment

E. Implementation of the retrieval program

The retrieval program “script2.exe” is programmed in the Go language. Clients use this program to retrieve and decrypt IoT data from the external DNS server. Likely the registration program, it utilizes the external package dns for DNS-related operation and GnuPG for decryption processing.

Figure 7 illustrates the processing steps of the retrieval program, and the following describes the details.

1) Inquiry to the cache server via UDP for the NS RR of the parent domain example.com.
2) Obtain candidate IP addresses of the external DNS server from the glue records accompanying the NS RR.
3) For each candidate IP address, sequentially inquiry the TXT RR of the target IoT device’s domain over DoT with TSIG or SIG(0) authentication. Also, the Verify function of x509 conducts certificate verification.
4) Extract data from the TXT RR and decode it with Base64. Then, execute the command “powershell echo PGPDATA | gpg –command-fd 0 –pinentry-mode=loopback –passphrase=PASSPHRASE –quiet -o- -d” to decrypt with PGP.
5) Output the plain text IoT data to the standard output.

V. Evaluation

A. Environment of experiment

We evaluate the proposed system by setting up a virtual environment on the host OS. We launched six instances of Ubuntu, configured each with the network settings shown in Figure 8, and assigned them to the corresponding devices in the system. Finally, we constructed an experimental local network, as depicted in Figure 9, including a group of DNS servers for name resolution.

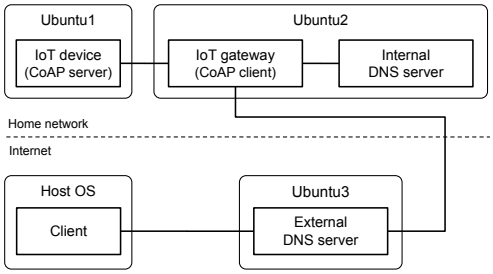


Fig. 9. Experimental local network topology (IoT data inquiry system)

TABLE II
The registered TXT RR
in both the internal and external DNS servers

The content of the TXT RR (excerpt)
IoTdevice1.home1.example.com. 3600 IN TXT
“LS0tLS1CRUdJTiBQR1AgTUVTU0 · · · 0tLQo=”

TABLE III
The result of executing the retrieval program

Command:
.\script2.exe -s 172.16.0.4 -a example.com. -d IoTdevice1.home1.example.com. -m 0 -w 3Yf5xUF3QcaN -l 5000 -t .\keys\tsig\Kexample.com.tsig.client_1 -g 1
Result: IoT encrypt (This string shows IoT data is properly decrypted.)

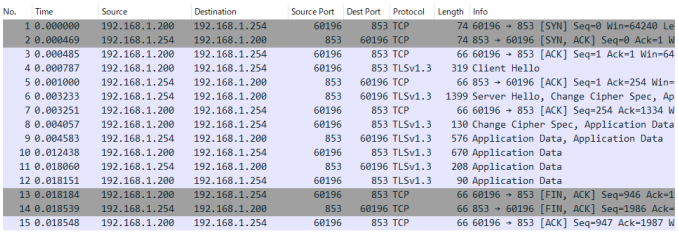
B. Functional evaluation

1) IoT data encryption and decryption: We verified the effectiveness of data encryption by the IoT gateway and decryption by the client.

First, we executed the registration program to create a new resource named “user1” on the CoAP server. Next, we executed a command manually to write the string ”IoT encrypt” to this resource. Then, the registration program registered the encrypted IoT data to the internal DNS server and external DNS server. Here, the destination domain for registration is “IoTdevice1.home1.example.com.”. After that, we used the dig command to query the TXT RRs of both DNS servers directly and got the result shown in Table II. From this result, it turns out that the IoT data on both DNS servers is encrypted.

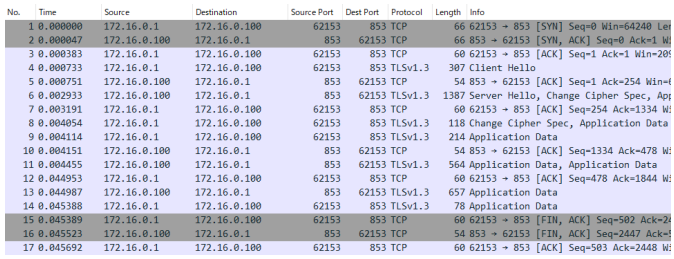
Next, we executed the retrieval program to retrieve IoT data from the destination domain on the external DNS server. The result is shown in Table III, confirming that the registered string “IoT encrypt” can be decrypted.

2) Communication encryption: To confirm that the communication link is encrypted, we used the packet visualization software Wireshark [21]. Here, we do not verify the communication in data registration to the internal DNS server because the IoT gateway and the internal DNS server operate on the same Ubuntu2, as shown in Fig. 9.



No.	Time	Source	Destination	Source Port	Dest Port	Protocol	Length	Info
1	0.000000	192.168.1.200	192.168.1.254	60196	853	TCP	74	60196 → 853 [SYN] Seq=0 Win=64240 Le
2	0.000469	192.168.1.254	192.168.1.200	853	60196	TCP	74	853 → 60196 [SYN, ACK] Seq=0 Ack=1 W
3	0.000485	192.168.1.200	192.168.1.254	60196	853	TCP	66	60196 → 853 [ACK] Seq=1 Ack=1 Win=64
4	0.000787	192.168.1.200	192.168.1.254	60196	853	TLSv1.3	319	Client Hello
5	0.001000	192.168.1.254	192.168.1.200	853	60196	TCP	66	853 → 60196 [ACK] Seq=1 Ack=254 Win=
6	0.003233	192.168.1.254	192.168.1.200	853	60196	TLSv1.3	1399	Server Hello, Change Cipher Spec, Ap
7	0.003251	192.168.1.200	192.168.1.254	60196	853	TCP	66	60196 → 853 [ACK] Seq=254 Ack=1334 W
8	0.004057	192.168.1.200	192.168.1.254	60196	853	TLSv1.3	130	Change Cipher Spec, Application Data
9	0.004583	192.168.1.254	192.168.1.200	853	60196	TLSv1.3	576	Application Data, Application Data
10	0.012438	192.168.1.200	192.168.1.254	60196	853	TLSv1.3	670	Application Data
11	0.018060	192.168.1.254	192.168.1.200	853	60196	TLSv1.3	208	Application Data
12	0.018151	192.168.1.200	192.168.1.254	60196	853	TLSv1.3	90	Application Data
13	0.018184	192.168.1.200	192.168.1.254	60196	853	TCP	66	60196 → 853 [FIN, ACK] Seq=946 Ack=1
14	0.018539	192.168.1.254	192.168.1.200	853	60196	TCP	66	853 → 60196 [FIN, ACK] Seq=1986 Ack=
15	0.018548	192.168.1.200	192.168.1.254	60196	853	TCP	66	60196 → 853 [ACK] Seq=947 Ack=1987 W

Fig. 10. Packet samples of the external DNS server update by the IoT gateway



No.	Time	Source	Destination	Source Port	Dest Port	Protocol	Length	Info
1	0.000000	172.16.0.1	172.16.0.100	62153	853	TCP	66	62153 → 853 [SYN] Seq=0 Win=64240 Le
2	0.000047	172.16.0.100	172.16.0.1	853	62153	TCP	66	853 → 62153 [SYN, ACK] Seq=0 Ack=1 W
3	0.000383	172.16.0.1	172.16.0.100	62153	853	TCP	60	62153 → 853 [ACK] Seq=1 Ack=1 Win=20
4	0.000733	172.16.0.1	172.16.0.100	62153	853	TLSv1.3	307	Client Hello
5	0.000751	172.16.0.100	172.16.0.1	853	62153	TCP	54	853 → 62153 [ACK] Seq=1 Ack=254 Win=
6	0.002933	172.16.0.100	172.16.0.1	853	62153	TLSv1.3	1387	Server Hello, Change Cipher Spec, Ap
7	0.003191	172.16.0.1	172.16.0.100	62153	853	TCP	60	62153 → 853 [ACK] Seq=254 Ack=1334 W
8	0.004054	172.16.0.1	172.16.0.100	62153	853	TLSv1.3	118	Change Cipher Spec, Application Data
9	0.004114	172.16.0.1	172.16.0.100	62153	853	TLSv1.3	214	Application Data
10	0.004151	172.16.0.100	172.16.0.1	853	62153	TCP	54	853 → 62153 [ACK] Seq=1334 Ack=478 W
11	0.004455	172.16.0.100	172.16.0.1	853	62153	TLSv1.3	564	Application Data, Application Data
12	0.044953	172.16.0.1	172.16.0.100	62153	853	TCP	60	62153 → 853 [ACK] Seq=478 Ack=1844 W
13	0.044987	172.16.0.100	172.16.0.1	853	62153	TLSv1.3	657	Application Data
14	0.045388	172.16.0.1	172.16.0.100	62153	853	TLSv1.3	78	Application Data
15	0.045389	172.16.0.1	172.16.0.100	62153	853	TCP	60	62153 → 853 [FIN, ACK] Seq=502 Ack=2
16	0.045523	172.16.0.100	172.16.0.1	853	62153	TCP	54	853 → 62153 [FIN, ACK] Seq=2447 Ack=
17	0.045692	172.16.0.1	172.16.0.100	62153	853	TCP	60	62153 → 853 [ACK] Seq=503 Ack=2448 W

Fig. 11. Packets in data retrieval from the external DNS server by the client

We captured packets in data registration as follows: First, we executed the tcpdump command on the IoT gateway (Ubuntu2) to start packet capture. Then, we executed the registration program to register IoT data and terminated the tcpdump.

Capturing in data retrieval similarly: We executed the tcpdump command on the external DNS server (Ubuntu3) and the retrieval program on the client to retrieve IoT data, then terminating the tcpdump.

By displaying the two sets of captured data in Wireshark, we got the results in Figure 10 and 11. They show that TLS communication worked in both cases, and the queried domain and content were encrypted. Therefore, we confirmed that communication encryption works appropriately.

3) Authentication with TSIG or SIG(0): To verify the authentication features of TSIG and SIG(0), we used unauthorized keys altered partially from legitimate ones. To validate the authentication in data registration, we executed the registration program with the tampered TSIG or SIG(0) keys and then attempted to register IoT data. With both authentications, the registration program resulted in an error, preventing the data from being registered to the internal and external DNS servers. Similarly, regarding the authentication in data retrieval, the retrieval program with the tampered TSIG or SIG(0) keys failed to retrieve IoT data.

VI. Conclusion

Security and privacy are crucial for IoT systems, particularly when retrieving IoT data from remote servers over the Internet. In this study, we proposed a DNS-based IoT data inquiry system that meets four key requirements described in section III-A. We also enhanced the system by adding name resolution for clients and extending authentication to improve usability. A prototype was implemented in a local network, and evaluations confirmed that the system's core functions—end-to-end IoT data encryption, DoT-based communication link encryption, and client authentication—worked as intended.

The future work includes extensions for handling binary-based IoT data and performance evaluations in a real network environment.

Acknowledgement

This work was partly supported by JSPS KAKENHI Grant Number 22K12007.

References

[1] L. Vailshery, "Number of internet of things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030," Statista, Accessed: Apr. 8, 2024. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[2] A. Petrosyan, "Monthly number of internet of things (IoT) malware attacks worldwide from 2020 to 2022," Statista, Accessed: Apr. 8, 2024. [Online]. Available: https://www.statista.com/statistics/1322216/worldwide-internet-of-things-attacks/

[3] D. Torre, A. Chennamaneni, and A. Rodriguez, "Privacy-preservation techniques for IoT devices: A systematic mapping study," *IEEE Access*, vol. 11, pp. 16 323–16 345, Feb. 2023.

[4] C. Sengul and A. Kirby, "Message queuing telemetry transport (MQTT) and transport layer security (TLS) profile of authentication and authorization for constrained environments (ACE) framework," IETF RFC 9431, Jul. 2023.

[5] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," IETF RFC 7252, Jun. 2014.

[6] H. Finney, L. Donnerhacke, J. Callas, R. Thayer, and D. Shaw, "OpenPGP message format," IETF RFC 4880, Nov. 2007.

[7] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over transport layer security (TLS)," IETF RFC 7858, May 2016.

[8] E. Paraschiv, E. Tudora, E. Tirziu, and A. Alexandru, "IoT & cloud computing-based remote healthcare monitoring system for an elderly-centered care," in *Proc. IEEE Int'l Conf. e-Health and Bioengineering (EHB)*, Nov. 2021, pp. 1–4.

[9] Arvandy and Y. Bandung, "Design of secure IoT platform for smart home system," in *Proc. IEEE Int'l Conf. Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Sep. 2018, pp. 114–119.

[10] Y. Jin, M. Tomoishi, and N. Yamai, "Secure remote monitoring and cipher data sharing for iot healthcare system with privacy preservation," in *Proc. ACM Int'l Conf. Cloud & Big Data Computing (ICCBDC 2021)*, Aug. 2021, pp. 46—51.

[11] A. Parecki, "OAuth 2.0," OAuth, Accessed: Jul. 12, 2024. [Online]. Available: https://oauth.net/2/

[12] D. Eastlake 3rd, "DNS request and transaction signatures (SIG(0)s)," IETF RFC 2931, Sep. 2000.

[13] F. Dupont, S. Morris, P. Vixie, D. Eastlake 3rd, O. Gudmundssonn, and B. Wellington, "Secret key transaction authentication for DNS (TSIG)," IETF RFC 8945, Nov. 2020.

[14] Internet Systems Consortium, "BIND 9," ISC, Accessed: Apr. 19, 2024. [Online]. Available: https://www.isc.org/bind/

[15] O. Bergmann, "libcoap," libcoap, Accessed: Apr. 19, 2024. [Online]. Available: https://libcoap.net/

[16] Google, "The Go programming language," Go, Accessed: Apr. 19, 2024. [Online]. Available: https://go.dev/

[17] W. Koch, N. Ellmenreich, M. Ashley, L. Cappelletti, D. Shaw, T. Wittek, and B. McGinnes, "The GNU privacy guard," GnuPG, Accessed: Apr. 19, 2024. [Online]. Available: https://www.gnupg.org/index.html

[18] M. Gieben, "dns package," Go Packages, Accessed: Apr. 19, 2024. [Online]. Available: https://pkg.go.dev/github.com/miekg/dns

[19] P. Mockapetris, "Domain names - implementation and specification," IETF RFC 1035, Nov. 1987.

[20] Google, "x509 package," Go Packages, Accessed: Apr. 19, 2024. [Online]. Available: https://pkg.go.dev/crypto/x509

[21] Wireshark Foundation, "Wireshark," Wireshark, Accessed: Apr. 25, 2024. [Online]. Available: https://www.wireshark.org/