一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IA2023-10,ICSS2023-10(2023-06)

DoTを用いたプライバシー配慮型IoTデータ照会システムの検討

相良　隼†　　金　勇††　　飯田　勝吉†††　　高井　昌彰†††

† 北海道大学 工学部 情報エレクトロニクス学科 札幌市北区北 11 条西 5 丁目
†† 東京工業大学 学術国際情報センター 東京都目黒区大岡山 2-12-1
††† 北海道大学 情報基盤センター 札幌市北区北 11 条西 5 丁目
E-mail: †sagara.hayato.y1@elms.hokudai.ac.jp, ††yongj@gsic.titech.ac.jp, †††{iida,ytakai}@iic.hokudai.ac.jp

あらまし　近年、Internet of Things (IoT) 機器の普及に伴い、IoT 機器を対象としたサイバー攻撃が増加している。また、IoT サービスの増加により、特にヘルスケアなどプライバシー情報を扱っているシステムでは IoT 機器のデータを遠隔からセキュアに取得することが求められている。そこで、本研究では、DNS over TLS (DoT) を利用したセキュアな IoT データ照合システムを提案する。提案システムでは認証と IoT データの暗号化に加え、通信路の暗号化も行い、セキュリティリスクとプライバシー漏洩リスクの両方の問題を解決する。さらに本稿では、実装および仮想環境上での評価実験を行い、提案システムの機能が実際に動作することを示す。

キーワード　IoT, プライバシー, データ照会システム, DNS, DNS over TLS (DoT), DNS Zone Transfer over TLS (XoT)

A Consideration of Privacy Preserved IoT Data Inquiry System using DoT

Hayato SAGARA†, Yong JIN††, Katsuyoshi IIDA†††, and Yoshiaki TAKAI†††

† Advanced Information Network Lab., School of Engineering, Dept. Electronics & Inf. Engineering, Hokkaido University.
†† Global Scientific Information & Computing Center, Tokyo Institute of Technology, Tokyo, Japan.
††† Information Initiative Center, Hokkaido University, Sapporo-shi, Japan.
E-mail: †sagara.hayato.y1@elms.hokudai.ac.jp, ††yongj@gsic.titech.ac.jp, †††{iida,ytakai}@iic.hokudai.ac.jp

Abstract　Recently, with the increase of Internet of Things (IoT) devices, the number of cyber-attacks against IoT devices has also increased. Under this circumstance, it is expected to securely obtain IoT data especially in a system handling privacy information such as a healthcare IoT system. Accordingly, we propose a secure inquiry system for IoT data using DNS over TLS (DoT) in this research. In the proposed system, in addition to the authentication for the devices handling IoT data, not only the IoT data itself but also the communication path will be encrypted. Consequently, the proposed system can mitigate the two risks of security and privacy. The preliminary evaluation using the implemented prototype system in a local experimental environment confirms that the features of the proposed system worked correctly.

Key words　IoT, Privacy, Data Inquiry System, DNS, DNS over TLS (DoT), and DNS Zone Transfer over TLS (XoT).

1. はじめに

近年、利便性向上やデータ収集などを目的として、様々な電子機器をインターネットやネットワークに接続させる Internet of Things (IoT) という概念が普及している。“The Statistics Portal” によると、IoT を活用した機器 (IoT 機器) の総数は、2023 年において全世界で約 15 億台に上ると試算されている [1]。一方で普及に伴い、IoT 機器を対象としたサイバー攻撃も増加しており、2022 年度末には IoT 機器へ約 11 億回も攻撃が行われた [2]。こうした事情から、IoT 機器のデータ (IoT データ) をインターネット上からセキュアに取得できるシステムの構築が求められているといえる。

十分なセキュリティを確保して IoT データを取得するための要件を図 1 に示す。要件は 3 つあり、1 つ目の要件は IoT 機器の認証である。この要件を満たしていない場合、なりすましされた IoT 機器から、クライアントが不正なデータを取得するおそれがある。2 つ目の要件は、IoT データの暗号化である。平文の状態で IoT データをクライアントへ送信した場合、インターネット上の第 3 者がその内容を窃取するおそれがある。3 つ目の要件は、通信の暗号化である。この要件は、平文による IoT データの窃取対策だけでなく、通信情報の収集によるプライバシー漏洩のリスク対策でもある。後者について、例えば IoT データに関する通信の IP アドレス、送信元・送信先・対象 IoT 機器のホスト名やドメイン名といった情報を、長期間

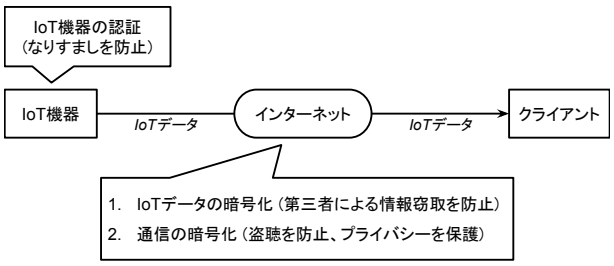

図 1　インターネットを経由して IoT データを取得する際に求められる 3 つのセキュリティ要件

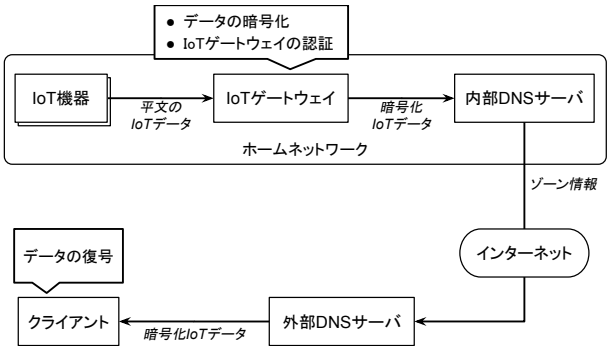

図 2　先行研究のシステムの概要

に渡って多数収集し分析された場合、IoT 機器が属するネットワークの内部構造や、クライアントが日々受信する IoT データの傾向を把握することが可能になるおそれがある。

ここで、上述の 3 つの要件をシステムへ組み込む際には、IoT 機器がもつ制約の存在を意識する必要がある。IoT 機器は省スペース・省エネルギー、およびリアルタイム性などを確保するために、PC といった従来型のデバイスと比べて、一般に処理性能は劣ることが多い。また処理性能を向上させるために、既存の IoT 機器を全て刷新する試みも考えられるが、掛かるコストや手間が非常に大きいため、現実的ではない。こうした理由から、IoT 機器単体で 3 つの要件を全て実現することは難しい。

そこで本研究では、主に名前解決で用いられる Domain Name System (DNS) のサーバを拡張したシステムを提案する。提案システムでは、IoT 機器がクライアントへデータを直接送信することはしない。代わりに DNS サーバへ IoT データを保存し、クライアントが DNS サーバに問い合わせることで、IoT データを取得する。DNS サーバは IoT 機器ではないため、認証や通信の暗号化を行うのに十分な性能を確保することができる。よって図 1 の 3 つの要件を満たしつつ、IoT 機器の性能に依存しない、セキュアなデータ取得システムが構築できると考えられる。

本論文では主に、関連研究、提案システムの内容、仮想環境における提案システムの実装、および 3 つの要件に関する提案システムの性能評価について述べる。

2. 関 連 研 究

本研究の提案システムは、金らが提案した、ヘルスケア用 IoT 機器向けのセキュアなデータ取得システム [3] を改良したものである。

金らのシステムについて述べる前に、提案システムで使用する、IETF によって標準化された DNS の 4 つの拡張方式について簡単に説明する。4 つの拡張方式とは、TSIG、SIG(0)、DNS over TLS (DoT)、および Zone Transfer over TLS (XoT) である。

まず TSIG は、RFC8945 [4] にて記載されている機能であり、共通鍵を利用した認証を提供する。クライアントと DNS サーバとの間で TSIG を使用するためには、DNS サーバ側に TSIG キーを登録する必要がある。TSIG キーには、共通鍵の内容、使用するアルゴリズムなどの情報を記載する。

SIG(0) は、RFC2931 [5] にて記載されている機能であり、公開鍵と秘密鍵を利用した認証を提供する。クライアントと DNS サーバ間において SIG(0) を使用する場合、クライアントに秘密鍵を、DNS サーバに公開鍵を配置する。

次に DoT は、RFC8310 [6] にて記載されている機能であり、通常 UPD 上で行われるクエリの問い合わせと応答を、TLS 上で実現させる。これにより、クエリに関する通信が暗号化される。

最後に XoT は、RFC9103 [7] にて記載されている機能であり、通常平文のまま行われるゾーン転送の通信を、TLS によって暗号化する。

さて、先行研究のシステムの概要を図 2 を用いて説明する。先行研究では、IoT 機器と IoT ゲートウェイが属するネットワークを、ホームネットワークとしてインターネットから分離する。そしてホームネットワーク内に DNS サーバ (内部 DNS サーバ) を 1 台、インターネット上に DNS サーバ (外部 DNS サーバ) を 1 台配置する。システムの処理の流れとしては、まず IoT 機器が IoT ゲートウェイへ IoT データを送信する。次に、IoT ゲートウェイは IoT データを Pretty Good Privacy (PGP) を用いて暗号化するとともに、Base64 形式でエンコードし、TXT レコードとして内部 DNS サーバへ登録する。なおレコードの登録時、内部 DNS サーバは SIG(0) によって IoT ゲートウェイを認証する。そして登録後、内部 DNS サーバが外部 DNS サーバへ自身のゾーン情報を転送する。最後に、クライアントが外部 DNS サーバへ問い合わせてデータを取得し、これを Base64 形式でデコード、および PGP で復号化することで、平文の IoT データを得る。

先行研究のシステムは、図 1 に記載された 2 つの要件、IoT 機器 (IoT ゲートウェイ) の認証とデータの暗号化は満たしている。しかし通信の暗号化という要件までは満たしていないため、1. 章で述べたプライバシー漏洩のリスクが残存している。

3. 提案システム

前章において、先行研究のシステムでは図 1 の要件である通信路の暗号化が満たされないことを説明した。そこで提案システムでは、通信路を含めた暗号化を実現するため、先行研

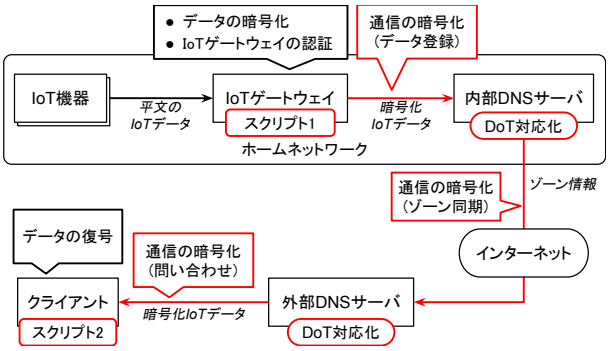

図 3　提案システムの概要

表 1　Ubuntu の構成情報

OS	Ubuntu Desktop 22.04.2 LTS
Python	Python 3.10.6
DNS ソフトウェア	BIND 9.18.12-0ubuntu0.22.04.1-Ubuntu (ESV)
CoAP ソフトウェア	libcoap3 Version 4.3.1

スクリプト1
- filewatch.py
- main.py
- update.py

図 4　スクリプト 1 の構成

filewatch.py

1. 引数を読み取る
2. 指定ディレクトリ内の指定ファイルを監視し続ける
   a. 指定ファイルの変更を検知 → 処理終了

図 5　filewatch.py の処理の流れ

update.py

1. 引数を読み取る (TSIGかSIG(0)の指定を含む)
2. データをPGPで暗号化
3. 2.のデータをBase64形式でエンコード
4. 3.のデータを255文字ごとに分割した後、空白で連結
5. 4.のデータを含むTXTレコードを作成
6. 5.のTXTレコードを内部DNSサーバへ登録
   a. TSIGの場合　→ dnspythonパッケージのudp.query.tlsを実行
   b. SIG(0)の場合 → nsupdateコマンドを実行

図 6　update.py の処理の流れ

究に次の 3 点の拡張を施す。1 つ目の拡張は、IoT ゲートウェイによる内部 DNS サーバへのレコード登録時と、クライアントによる外部 DNS サーバへの問い合わせ時の両方で、DoT を使用することである。次に 2 つ目の拡張は、内部 DNS サーバと外部 DNS サーバ間におけるゾーン転送で、XoT を使用することである。最後に 3 つ目の拡張は、認証機能を強化するために、SIG(0) だけでなく TSIG も使用可能とすることである。

提案システムの概要を図 3 に示す。提案システムは、IoT 機器、IoT ゲートウェイ、内部 DNS サーバおよび外部 DNS サーバから構成される。これらのうち IoT 機器、IoT ゲートウェイおよび内部 DNS サーバについては、同一のネットワーク (ホームネットワーク) に属する。クライアントは、ホームネットワーク外に属する外部 DNS サーバに対して問い合わせを行い、IoT データを取得する。さらに提案システムでは、スクリプト 1 およびスクリプト 2 という 2 つのプログラムを使用する。スクリプト 1 は IoT ゲートウェイ上で動作し、IoT データに対する加工と内部 DNS サーバへのデータ登録を行う。スクリプト 2 はクライアント上で動作し、外部 DNS サーバからのデータ取得とデータ内容の表示を行う。

さて、IoT 機器と IoT ゲートウェイの動作を 3.1 節で、内部 DNS サーバと外部 DNS サーバの動作を 3.2 節で、それぞれ詳細に説明する。

3.1　IoT 機器と IoT ゲートウェイ

IoT 機器は、IoT データを生成するとともに、IoT ゲートウェイと通信を行う。通信の際に用いるプロトコルは、CoAP や MQTT といった、既存のプロトコルを使用する。

IoT ゲートウェイは、スクリプト 1 を使用して次の動作を行う。まず、各 IoT 機器が新たにデータを生成したかどうかを監視する。そして、ある IoT 機器のデータ生成を検知した場合、当該 IoT 機器からデータを取得する。その後、取得データの暗号化と Base64 形式によるエンコードを行う。最後に、データを TXT レコードとして整形し、DoT で内部 DNS サーバへ登録する。なお、データを暗号化するためのプロトコルとして、先行研究と同じ PGP を用いる。

3.2　内部 DNS サーバと外部 DNS サーバ

内部 DNS サーバは、IoT ゲートウェイからのレコード登録要求を受け付けて、暗号化された IoT データを保存する。レコードに登録可能な機器であるかどうかは、TSIG または SIG(0) によって認証される。

外部 DNS サーバは、ホームネットワーク外に存在し、内部 DNS サーバと同じゾーン情報を保有する。また、クライアントからの問い合わせを DoT で受け付け、対応する IoT 機器の TXT レコードを返す。

内部 DNS サーバへのレコード登録が完了次第、内部 DNS サーバは外部 DNS サーバへ、UDP ポート 53 番で NOTIFY メッセージを送信する。これを受信した外部 DNS サーバは内部 DNS サーバへゾーン転送を要求し、自身が有するレコードを最新のものに更新する。なお、ゾーン転送は XoT で行う。

4.　実装と評価実験

4.1　実　　装

図 3 を元に、提案システムの具体的な実装内容について述べる。まず、クライアントを除いた全ての機器について、表 1 に示した構成の Ubuntu 上で動作を行う。表 1 のうち、Python はスクリプト 1 およびスクリプト 2 を実行するために使用し、BIND9 は内部 DNS サーバおよび外部 DNS サーバを、libcoap3 は IoT 機器および IoT ゲートウェイを動作させるために使用する。

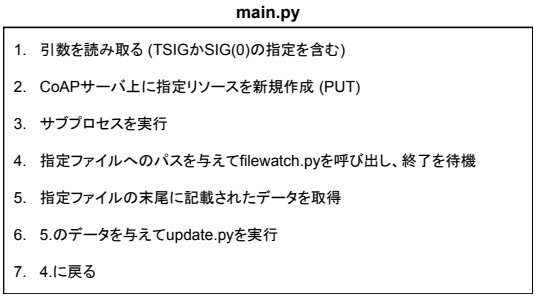

図 7　main.py のメインプロセスの処理の流れ

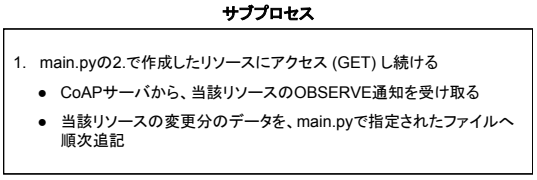

図 8　main.py のサブプロセスの処理の流れ

さて、内部 DNS サーバに記載するゾーン情報を 4.1.1 項で、BIND9 による内部 DNS サーバと外部 DNS サーバの設定を 4.1.2 項で、IoT 機器と IoT ゲートウェイの実装を 4.1.3 項で、スクリプト 1 の実装を 4.1.4 項で、スクリプト 2 の実装を 4.1.5 項でそれぞれ説明する。

4.1.1　内部 DNS サーバに記載するゾーン情報

DNS サーバから所望の IoT 機器のデータを取得するために、登録する IoT データの TXT レコードを、取得元の IoT 機器と対応付ける必要がある。そこで、ホームネットワーク内の全ての IoT 機器に対して、内部 DNS サーバに記載されたあるドメインの下で重複なく対応するように、サブドメインを個別に割り当てる。例えば、内部 DNS サーバのドメイン example.com. 下で IoT 機器 2 台を割り当てる場合、1 台目にはサブドメイン IoTDevice1.home.example.com. を、2 台目にはサブドメイン IoTDevice2.home.example.com. をそれぞれ与える。こうして割り当てられた IoT 機器のデータは、スクリプト 1 によって対応するサブドメイン下へ登録される。逆に所望の IoT 機器のデータを取得する際は、対応するサブドメインを指定して DNS サーバに問い合わせる。

4.1.2　内部 DNS サーバと外部 DNS サーバの設定

内部 DNS サーバの BIND9 の設定ファイルについて、IoT データを格納する対象ドメインを次のように設定した。まず、認証によってデータの登録可否を制御するため、allow-update 句に TSIG 共通鍵と SIG(0) 公開鍵を指定した。次に XoT を使用するため、allow-transfer 句に TSIG 共通鍵と SIG(0) 公開鍵を指定するとともに、port 853 および tls を指定した。最後に DoT によるクエリの通信路暗号化を行うため、listen-on 句に port 853 および tls を指定した。

外部 DNS サーバの設定ファイルについて、内部 DNS サーバの場合と同様に指定し、DoT が行われるようにした。また、スレーブのゾーンにおける masters 句に port 853 および tls を指定して、内部 DNS サーバと XoT が行われるようにした。

4.1.3　IoT 機器と IoT ゲートウェイの実装

本論文において、IoT 機器と IoT ゲートウェイは実機でなく Ubuntu 上で再現する。そのために本論文では、IoT 機器を CoAP サーバ、IoT ゲートウェイを CoAP クライアントとみなして実装する。CoAP サーバはリソースの管理、および CoAP クライアントへのデータ送信を行う。CoAP クライアントは、CoAP サーバに対するリソースの新規作成と指定リソースのデータ取得を行う。このとき、CoAP サーバ上の各リソースは各 IoT 機器に該当する。また CoAP サーバに対する CoAP クライアントのデータ取得は、CoAP を用いた IoT 機器から IoT ゲートウェイへのデータ送信に該当する。よって、実機と同様の処理を実現できているといえる。なお CoAP サーバの立ち上げには、libcoap3 で標準実装されている coap-server コマンドをそのまま使用する。

4.1.4　スクリプト 1 の実装

スクリプト 1 の構成を図 4 に示す。スクリプト 1 は主プログラム main.py と、副プログラム filewatch.py および update.py という 3 つのプログラムから成る。スクリプト 1 を使用する際は、これらのうち main.py のみを直接実行する。

副プログラム filewatch.py は、指定ディレクトリ内の指定されたファイルに変更が加えられたかどうかを検知するプログラムである。その処理の流れを図 5 に示す。このプログラムは変更を検知するまで指定ファイルを監視し続け、検知した場合は何も出力せずに処理を終了する。

副プログラム update.py は、与えられたデータを TXT レコードとして DNS サーバへ登録するプログラムである。その処理の流れを図 6 に示す。このプログラムは、まず入力データを PGP で暗号化し Base64 形式でエンコードする。その後、TXT レコードとして登録できるようにデータを整形する。具体的には、データとなる文字列を 255 文字ごとに分割し、さらに分割した各文字列を空白で連結するといった処理を行う。これは、1 バイト長の文字に対して、256 文字以上の文字列は一度に TXT レコードへ格納することができないためである [8]。データの整形後、update.py は TSIG または SIG(0) を使用して、指定された DNS サーバへ TXT レコードを登録する。なおレコードの登録処理において、TSIG を指定した場合は dnspython パッケージの udp.query.tls が、SIG(0) を指定した場合は nsupdate コマンドがそれぞれ実行される。

主プログラム main.py は、IoT 機器のデータを自動的に DNS サーバへ登録するプログラムである。その処理の流れを図 7 および図 8 に示す。main.py は、まず CoAP サーバ (IoT 機器) に対して、指定したリソース名のリソースを PUT メソッドで新規作成する。次に、サブプロセスとして GET メソッドで当該リソースへアクセスし、OBSERVE 通知の受け取りおよびデータの自動取得を行う。取得したデータは、指定フォルダ内の指定ファイル (データ取得用ファイル) に順次追記する。そしてメインプロセスは、データ取得用ファイルの変更を検知す

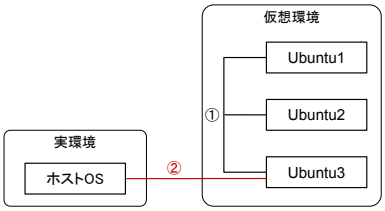

図 9　仮想環境の構成

表 2　実験環境におけるホスト PC とホスト OS の構成情報

ホスト PC のマシン構成	
OS バージョン	22H2
OS ビルド	19045.2965
プロセッサ	Intel(R) Xeon(R) Bronze 3204 CPU 1.90 GHz
プロセッサ数	2
RAM	32 GB
ホスト OS の構成	
OS	Windows 10 Pro for Workstations
OS バージョン	22H2
OS ビルド	19045.2965
仮想環境ソフトウェア	VirtualBox 7.0.8

表 3　実験環境における各 OS のネットワーク設定

ホスト OS	
IPv4 ②	172.16.0.1
サブネットマスク	255.255.255.0
Ubuntu1	
IPv4 ①	192.168.1.100
サブネットマスク	255.255.255.0
Ubuntu2	
IPv4 ①	192.168.1.200
サブネットマスク	255.255.255.0
Ubuntu3	
IPv4 ①	192.168.1.254
IPv4 ②	172.16.0.100
サブネットマスク (両方)	255.255.255.0

るよう指定して filewatch.py を呼び出し、その実行完了を待つ。実行完了後、データ取得用ファイルの末尾に記載された最新のデータを取り出し、これを update.py に与えて呼び出す。以降 main.py が終了されるまで、filewatch.py の呼び出し時点から処理を繰り返す。

4.1.5　スクリプト 2 の実装

スクリプト 2 は単一のプログラム retrive.py から構成され、外部 DNS サーバからの IoT データ取得を行う。retrieve.py は最初に、指定された FQDN のドメインを外部 DNS サーバへ問い合わせ、TXT レコードを得る。次に TXT レコードからデータとなる文字列を取り出すとともに、文字列から空白を除いて連結する。その後、base64 形式のデコードと PGP の復号化を行うことで文字列を平文化し、標準出力する。

4.2　実 験 環 境

提案システムの動作を検証するため、3 台の Ubuntu を図 9 に示した仮想環境上で動作させた。仮想環境におけるホスト

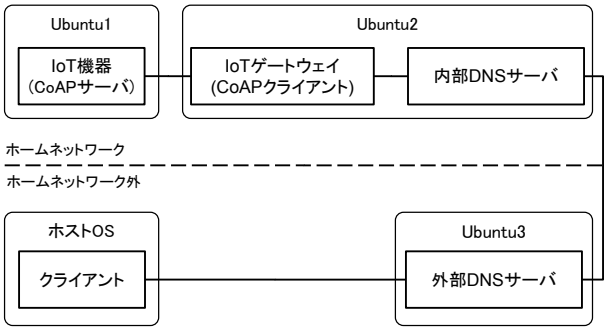

図 10　実験環境のネットワークトポロジー

```
>python retrieve.py IoTdevice1.home.example.com
response:
IoT data was encrypted and decrypted.
```

図 11　スクリプト 2 の実行結果

表 4　内部 DNS サーバと外部 DNS サーバに登録された TXT レコード

内部 DNS サーバ
IoTdevice1.home.example.com. 36000 IN TXT ”LS0tLS1CRUdJTiBQR1AgTUV...(中略)...UtLS0tLQo=”
外部 DNS サーバ
IoTdevice1.home.example.com. 36000 IN TXT ”LS0tLS1CRUdJTiBQR1AgTUV...(中略)...UtLS0tLQo=”

PC のマシン構成およびホスト OS の構成は表 2 であり、各 OS のネットワーク設定は表 3 である。ここで表 3 中の丸数字は、図 9 中の各セグメントに付記された丸数字と対応している。さらに仮想環境上の各 OS を提案システムの各機器と対応させ、図 10 に示したネットワークトポロジーを構築した。具体的には、1 台目の Ubuntu (Ubuntu1) が IoT 機器を、2 台目の Ubuntu (Ubuntu2) が IoT ゲートウェイおよび内部 DNS サーバを、3 台目の Ubuntu (Ubuntu3) が外部 DNS サーバを、ホスト OS がインターネット上のクライアントをそれぞれ再現するよう実装した。

4.3　実 験 結 果

4.2 節の実験環境において、図 3 中の提案システムの各機能が実現されていることを確かめる。具体的に、IoT データの暗号化について 4.3.1 項で、内部 DNS サーバによる IoT ゲートウェイの認証について 4.3.2 項で、通信路の暗号化について 4.3.3 項でそれぞれ検証する。

4.3.1　IoT 機器のデータの暗号化

内部 DNS サーバおよび外部 DNS サーバ内の IoT データが暗号化されていること、およびそれら IoT データが復元できることを確かめる。

はじめに Ubuntu1 上で coap-server コマンドを実行し、CoAP サーバを立ち上げた。次に Ubuntu2 上でスクリプト 1 を実行して、CoAP サーバへの新規リソース作成およびドメイン IoTdevice1.home.example.com. へのデータ登録の待機を行った。続いて Ubuntu1 上で coap-client コマンドを実行し、作成したリソースを更新した。このとき、当該リソース内には文字列”IoT

data was encrypted and decrypted.”が保存される。さらにスクリプト 1 によって、当該リソースのデータは自動的に内部 DNS サーバへ登録される。最後にクライアント上でスクリプト 2 を実行し、ドメイン IoTdevice1.home.example.com. のデータ取得と表示を行った。そのときの実行結果は図 11 のとおりである。

さて上記の操作後、内部 DNS サーバおよび外部 DNS サーバにドメイン IoTdevice1.home.example.com. の TXT レコードを問い合わせたところ、表 4 の結果が得られた。この表 4 から、IoT データは適切に暗号化されていることが確認された。また図 11 より、暗号化された IoT データはクライアント上で復号可能であることも確かめられた。

4.3.2 IoT ゲートウェイの認証

IoT ゲートウェイの認証失敗によって、IoT データが DNS サーバへ登録されないことを確かめる。

はじめに、偽の TSIG 共通鍵と SIG(0) 秘密鍵に該当するダミーファイルを、それぞれ dummy.key および dummy.private というファイル名で作成した。これらダミーファイルは、対応する正規の鍵ファイルの一部分を書き換えたものである。

偽の TSIG 共通鍵を用いた場合について検証する。まず、coap-server コマンドで Ubuntu1 上に CoAP サーバを立ち上げた。次に、dummy.key を用いて TSIG の認証を行うようスクリプト 1 を設定し、実行した。最後に Ubuntu2 上で coap-client コマンドを実行し、CoAP サーバ上のリソース内のデータが文字列”IoT_Dummy_Data_1”となるよう更新した。このとき、スクリプト 2 を用いて外部 DNS サーバから IoT データを取得したところ、前節の実行結果である図 11 と同じ結果になった。これは、スクリプト 1 による登録処理が内部 DNS サーバによって拒否されたことを示している。

続いて偽の SIG(0) 秘密鍵を用いた場合について検証する。こちらも、まず coap-server コマンドで Ubuntu1 上に CoAP サーバを立ち上げた。次に、dummy.private を用いて SIG(0) の認証を行うようスクリプト 1 を設定し、実行した。最後に Ubuntu2 上で coap-client コマンドを実行し、CoAP サーバ上のリソース内のデータが文字列”IoT_Dummy_Data_2”となるよう更新した。このときスクリプト 2 を実行すると、やはり図 11 と同じ結果になった。

以上 2 つの実行結果から、TSIG および SIG(0) のいずれを使用しても、内部 DNS サーバは IoT ゲートウェイを適切に認証することが確かめられた。

4.3.3 通信路の暗号化

内部 DNS サーバと外部 DNS サーバ間におけるゾーン転送の通信路が暗号化されていること、および外部 DNS サーバへのクエリの通信路が暗号化されていることを確かめる。なお内部 DNS サーバへのクエリの通信路暗号化については、図 10 より IoT ゲートウェイと内部 DNS サーバが同じ OS 上で動作するため、本評価実験では省略する。

まず Ubuntu2 上で tcpdump コマンドを実行し、Ubuntu2 のトラフィックを pcap ファイル DNS_internal.pcap へキャプチャさせた。続いて 4.3.1 項と同様の操作を行い、IoT データを内部 DNS サーバへ登録させた。最後に DNS_internal.pcap へのキャプチャを終了させた。

次に Ubuntu3 上で tcpdump コマンドを実行し、Ubuntu3 のトラフィックを pcap ファイル DNS_external.pcap へキャプチャさせた。こちらも 4.3.1 項と同様の操作を行った後、DNS_external.pcap へのキャプチャを終了させた。

上記で得られた 2 つの pcap ファイルをソフトウェア Wireshark 上で表示したところ、内部 DNS サーバと外部 DNS サーバの場合それぞれにおいて、内部 DNS サーバが外部 DNS サーバと TLS を使用してゾーン転送を行っていること及びクライアントと外部 DNS サーバ間の通信路も TLS によって暗号化されることが確認された。

5. ま と め

IoT 機器のデータをセキュアに取得するためのシステムについて、先行研究では通信路の暗号化や SIG(0) による認証が行われていないという問題があった。本研究ではそうした問題点を解消するシステムを新たに提案し、さらに評価実験を通じて実際にシステムが構築可能であることを示した。

今後の課題として、次の 2 点が挙げられる。1 つ目は、外部 DNS サーバの IP アドレスを直接指定するのではなく、ルートサーバからの再帰問合せによって IP アドレスが取得できるよう変更することである。2 つ目は、セキュリティをより向上させるために、インターネットからホームネットワークへのアクセスを遮断できるようにすることである。

謝　　辞

本研究の一部は、JSPS 科研費 19K20254 の助成を受けたものです。

文　　献

[1] Statista, “Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030,” https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed 2023-05-02).

[2] Statista, “Monthly number of Internet of Things (IoT) malware attacks worldwide from 2020 to 2022,” https://www.statista.com/statistics/1322216/worldwide-internet-of-things-attacks/ (accessed 2023-05-02).

[3] Y. Jin, M. Tomoishi, and N. Yamai, “Secure Remote Monitoring and Cipher Data Sharing for IoT Healthcare System with Privacy Preservation,” *Proc. ACM Int’l Conf. Cloud & Big Data Computing (ICCBDC 2021)*, Aug. 2021, pp. 46–51.

[4] F. Dupont, et al., “Secret Key Transaction Authentication for DNS (TSIG),” *IETF RFC8945*, Nov. 2020.

[5] D. Eastlake 3rd, “DNS Request and Transaction Signatures (SIG(0)s),” *IETF RFC2931*, Sept. 2000.

[6] S. Dickinson, D. Gillmor, and T. Reddy, “Usage Profiles for DNS over TLS and DNS over DTLS,” *IETF RFC8310*, Mar. 2018.

[7] W. Toorop, et al., “DNS Zone Transfer over TLS,” *IETF RFC9103*, Aug. 2021.

[8] P. Mockapetris, “Domain Names: Implementation and specification,” *IETF RFC1035*, Nov. 1987.