Vol.2018-CSEC-83 No.12
2018/12/14

フォグコンピューティングを利用した IoT システムのセキュリティに関する研究

丹羽 雅哉[†1†2]　大久保 隆夫[†1]

概要：近年，さまざまな IoT 機器の数は年々増加しており，IoT 機器で取得した情報を収集・蓄積し，分析などを行うシステムの検討や開発が進められている．検討や開発が進み，複数の IoT 機器やサーバを含むシステムをサービス化していく中で，単体の機器だけでなく，システム全体としてセキュリティを考慮する必要があるが，IoT 機器には消費電力やメモリ等のリソースに制約がある場合が多く，IoT 機器単体でさまざまな脅威に対する十分な対策を打つことが困難である可能性が高い．こういった背景の中，IoT システムにおける想定される脅威とセキュリティ要件を整理し，その要件を実現するためのシステムアーキテクチャとしてフォグコンピューティングを検討した．本研究では，フォグコンピューティングの考え方を基に，監視カメラを想定したシステムを検証環境として構築し，セキュリティ要件の実現性確認や検証環境における性能評価/安全性評価を行うことを考えている．本稿では，セキュリティ要件の実現性確認の結果と性能評価の一部結果について報告する．

キーワード：IoT，フォグコンピューティング，セキュリティ

Study on the IoT system using fog computing

MASAYA NIWA[†1†2]　TAKAO OKUBO[†1]

Abstract: In recent years, the number of various IoT devices has increased year by year, and investigations or developments of systems for storing, and analyzing information acquired by IoT devices are under way. For such researches and developments, it is necessary to consider security not only for individual devices but also for whole system. However, IoT devices often have limited power and limited memory, therefore, it is difficult for IoT devices alone to take adequate security countermeasures against various threats. Under such circumstances, we consider to apply Fog Computing to a system architecture for organizing the assumed threats and security requirements in the IoT system and realizing the requirements. In this research, based on the concept of Fog Computing, we have built a system assuming a surveillance camera as a verification environment, and we are confirming the feasibility of security requirements, and evaluating performance / safety in this verification environment. In this paper, we report the result of the feasibility check of the security requirements and the part of the performance evaluations.

Keywords: IoT, Fog Computing, Security

1. はじめに

インターネット技術や各種センサー・テクノロジーの進化などを背景に，パソコンやスマートフォンなど従来のインターネット接続端末に加え，家電や自動車，ビルや工場など，世界中の様々な機器がインターネットへつながり，その数は爆発的に増加している．IoT 機器の数は 2015 年時点で 154 億個，2016 年時点で 173 億個，2020 年までに 304 億個まで増大すると推定され，分野別での成長率という観点で見てみると，コンピュータの普及率が拡大から成熟に向かう一方で，自動車（コネクテッドカー），産業用途の産業機器，家電などのコンシューマ機器の IoT 化が進んでいる [1].

こういった現状を踏まえ，現在，さまざまな IoT 機器の情報を収集・蓄積し，分析などを行うシステムの検討や開発が進められている．また，これまでクローズドであったネットワーク環境から，さまざまな機器が連携し，オープンなネットワーク環境へ移行しつつある．検討や開発が進み，複数の IoT 機器やサーバを含むシステムをサービス化していく中で，単体の機器だけでなく，システム全体としてセキュリティを考慮する必要がある．

本研究では，一般的な IoT 機器/IoT システムの特徴や IoT システムの想定脅威とセキュリティ要件を整理し，脅威に対するアーキテクチャとしてフォグコンピューティング [3]を検討している．そして，フォグコンピューティングの考え方を基に，監視カメラを想定した IoT システムを検証環境として構築し，セキュリティ要件の実現性確認や検証環境における性能評価/安全性評価を行い，IoT システムにおけるセキュアなアーキテクチャ設計のベストプラクティスについての研究を進めていく．なお，本稿では，セキュリティ要件の実現性確認の結果と性能評価の一部結果について報告する．

†1 情報セキュリティ大学院大学
Information Security INSTITUTE of INFORMATION SECURITY
†2 ＮＴＴテクノクロス株式会社
NTT TechnoCross Corporation

2. IoT（システム）の特徴，想定脅威とセキュリティ要件

一般のシステムとの違いに着目した，IoT 機器や IoT システムの主な特徴を表 1 に示す．なお，これらの特徴は，分野やシステムによって異なると考えるが，一般的な特徴を挙げた．

表 1　IoT 機器・システムの特徴

IoT の特徴	説明
リソースの制約	PC 等と比較して，小型，安価であり，CPU・メモリ・ディスク・消費電力などリソースに制約がある場合が多い．
機器の管理	無人環境で管理がされない，また，スマートロックや AI スピーカーのように画面等がなく，監視が行き届きにくい(異常に気付きにくい)場合がある．
ライフサイクル	一般的なシステムに比べ，ライフサイクルが長く，時間の経過とともにセキュリティ対策が危殆化する可能性が高い．
データ	機微なデータ(例えば，位置情報，医療情報など)を扱う場合がある．
性能	医療や災害対応など，リアルタイム性が求められる場合がある．

また，文献 [2]によれば，IoT 機器が連携するシステムの想定脅威は表 2 の通りである．脅威は，利用者による操作に起因するものや正当な権限を持たない攻撃者が悪意で行う操作に起因するものがある．

表 2　IoT 機器が連携するシステムの想定脅威とセキュリティ要件 [2]

脅威	セキュリティ要件
不正アクセス，情報の改ざん，情報の漏えい	アクセス制御：各機器へのアクセス制御
なりすまし，不正利用，情報の漏洩	認証：ユーザ認証，サーバ認証，機器認証，メッセージ認証
盗聴，情報の漏洩	暗号化：通信，データの暗号化
ウイルス感染	ウイルス対策：ウイルスの予防，検出，回復
脆弱性をついた攻撃	アップデート：ソフトウェアの更新を容易に実施 ※対策が難しい場合は仮想パッチ等により前段階で攻撃を防御
情報漏えい，Dos 攻撃	侵入・攻撃対策：FW，IDS/IPS の導入，不正侵入検知，ログ取得
情報の消失，破損，改ざん	バックアップ：定期的にデータのバックアップ
不正アクセス，不正利用	ログ管理：アクセスログ，操作ログなどの管理

3. フォグコンピューティング

IoT 機器が連携するシステムにおいては，さまざまな脅威が考えられるが，末端の IoT 機器のリソース（CPU・メモリ・ディスクなど）は限られているため，十分なセキュリティ対策を打つことが困難であると考える．そこでフォグコンピューティングの考え方を検討し，末端の IoT 機器で対応するのではなく，中間層（フォグ層）でセキュリティ要件を満たすようなシステムアーキテクチャを考える．Cisco が提唱しているフォグコンピューティング [3]（クラウド/雲よりも機器に近いところに位置するのでフォグ/霧と呼ばれている）は，図 1 のように IoT 機器とインターネットの間に，フォグ層をおく構成である．IoT 機器のあるローカルなネットワーク内に，データを一次的に処理する IoT ゲートウェイやフォグノードなどを置き，処理されたデータを必要な IoT デバイスへ送り返したり，クラウドに上げたりすることで，即時応答とクラウドの負荷軽減を目的としている．例えば，以下のような課題へ対応することができる．

- 通信の速度はサーバとの距離に比例して遅くなるため，医療や災害の対応など即時性が求められるシステムでは，対応が遅れる．
- クラウドとの通信障害が発生した場合，各デバイスはクラウドにデータを送れず，クラウドからの応答もなくなる．

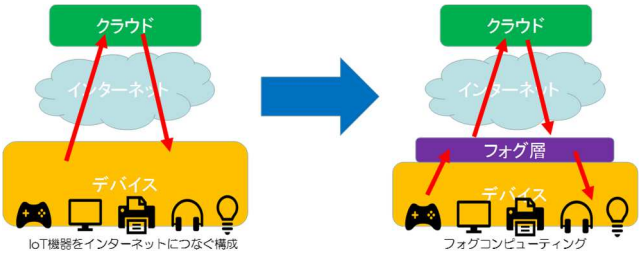

図 1　従来 IoT システム構成とフォグコンピューティング

つまり，フォグコンピューティングは，性能の向上（リアルタイム性の向上）やシステムにおける可用性を高めるためのシステムアーキテクチャであると考えられる．

一方，セキュリティの観点における脅威を考慮した場合も，このフォグコンピューティング，つまりフォグ層でさまざまな処理を行うことで，対応できることも多くあるのではないかと考える．以下に，実際にフォグ層でどのようなセキュリティ関連の機能を持てばよいか現段階で本研究において想定している機能の例を示す．

- 攻撃の検知/遮断

  FW，IDS/IPS，WAF の機能により，不正な通信を検知し遮断する．例えば，この機能により，外部からの攻撃を検知・遮断したり，内部からの予期しない通信を検知・遮断することが期待できる．具体例として IoT マルウェアである Mirai [4] [5]を考えると本機能により，まず不正な侵入を防ぐことができ，仮に侵入を許

したとしても，フォグ層で IoT 機器の不正なアウトバウンド通信を制御することで C&C サーバへの通信や DDos 攻撃となる通信，他の IoT 機器への通信を制御できるのではないかと考える．

- 監視
  機器のファイル，ソフトウェア，ファームウェアなどの構成を管理し，意図しない構成であった場合の異常を検知する．例えば，この機能により，不正なファイル操作や，ソフトウェア/ファームウェアの改ざんを検知することが期待できる．また，機器のプロセスや CPU，メモリなどのリソースを監視し，異常を検知する．例えば，この機能により，不正なプロセスが動作していた場合，それを検知することが期待できる．具体例として Smart Parking system [6]への攻撃を考える．例えば，物理的なアクセスにより，一部の IoT 機器（Sensor， IP Camera）が乗っ取られたり，改ざんされた場合は，フォグ層において IoT 機器の状態や構成を監視・管理をしたり，通常と異なる動作（例えば，全体のパーキングスペースの中で 1 箇所だけ利用率が明らかに低い場合など）を検知（振る舞い検知）することで，攻撃や異常が起こったことを検知でき，最終的な被害を防ぐことができるのではないかと考える．
- アップデート管理
  機器にセキュリティパッチをあてるなど，アップデート処理・管理を行う．
- 簡単な演算処理
  機器や上位のサーバからデータを受け取り，一次処理を行い，その結果を返す．例えば，事前にルールを設けておき，受け取ったデータやリクエストの不正を判断し，不正を検知することが期待できる．

なお，フォグコンピューティングと類似した概念でエッジコンピューティングがあるが，文献 [7]によれば，フォグコンピューティングは，フォグ層のノードが相互につながって連携しながら分散処理を行うと述べられている．

4. 関連研究

4.1 フォグコンピューティング

Kewei Sha らは，IoT システムのセキュリティを強化するために Edge レイヤーでのセキュリティサービス設計である EdgeSec を提案している [8]． EdgeSec では，3 層（Things レイヤー，Edge レイヤー，Cloud レイヤー）にレイヤーをわけ，システム設計を行っていて，Edge レイヤーは Things レイヤーよりもはるかに多くのリソースを持っている前提としている．そして，Edge レイヤーのリソースを利用して，データ暗号化，キー生成，侵入検出などの計算集約型タスクを実行したり，Things レイヤーの異種性をマスクしたり，Things レイヤーの機器を管理することに利用できると述べられている．また，Edgesec では，7 つのコンポーネント，プロセスで設計を行うことで IoT システムのセキュリティ問題を体系的に処理する提案が行われている．この EdgeSec では，IoT 機器が必ず Edge レイヤーを経由し，Edge レイヤーでハンドリングするように見受けられた．必ず Edge レイヤーを経由するため，場合によっては，性能とのトレードオフが発生するものと考える．また，全て Edge を経由することにより，Edge ノードが単一障害点となってしまうおそれがあると考える．さらに，Edge ノードが管理機能などの重要な機能を持っており，Edge ノードが乗っ取られてしまうと，システム全体に影響が出てしまう可能性が高い．システム設計を行う際には，対策を講じる必要があると考える．

Salem Alharbi らは，IoT デバイスへの通信を保護するためにフォグ層に VPN や DDoS 攻撃に対するセキュリティ機能を展開した FOCUS（FOg CompUtingbased Security System）を提案している [9]．FOCUS はエンドユーザに近いフォグコンピューティングで実装されているため，応答待ち時間が非常に短く，クラウドでセキュリティ機能を展開した場合と比較して 41%程度性能が向上したと述べている．このように，フォグコンピューティングにより性能向上が期待できると考える．

Mithun Mukherjee らは，IoT デバイスにサービスを提供するフォグノードは，サービスを要求するデバイスが本物であるかどうかを検証することができなければならない．また，データおよび他の重要な処理要求を送信する IoT デバイスは，意図された認証が実際に安全であるかどうかを検証することができなければならない．さらに，フォグノード間の通信には，マルチホップパスに含まれるノードが信頼できないため，エンドツーエンドのセキュリティが必要と述べている [10].

4.2 IoT システムにおける暗号，通信，認証

Abebe Abeshu Diro らは，IoT 機器同士が互いに通信すると，個々の IoT 機器ではセキュリティキーの処理，格納，および通信する能力が不足しているという課題があるため，フォグノードをブローカー（プロキシ）として利用し，ECC ベースの ElGamal 暗号でプロキシ再暗号化する方式を提案している [11]．ECC を利用することにより，RSA より高速に処理できることが示されており，メモリ要件についても同じセキュリティレベルで RSA と比較した場合に有利であることが述べられている．

Mohammed Al-Soh らは，高度にスケーラブルな MQTT プロトコルに基づくユビキタス評価システムのためのアーキテクチャを提案している [12]．MQTT は，軽量，省電力であり，スケーラビリティの高いイーサネット・メッセージング・プロトコルであり，高遅延で信頼性の低いネット

ワークもサポートしている．短いメッセージを大量にやりとりする際に有利であり，スマートウォッチとのメッセージのやりとりで MQTT を用いている．

Luciano Barreto らは, IoT システムにおける認証について認証はエンドユーザ，クラウドプロバイダ，製造者を考慮しなければならず，特に製造者がアップデートすることも考慮した上で，認証認可の仕組みは考慮する必要があると述べている [13]．ここでは，IoT Cloud Provider が IdM としてユーザの立場，及び製造者の立場から SSO のフローが示されている．

5. 実機検証

5.1 検証目的

「4 関連研究」で述べた通り，IoT システムにおいてリソース不足という課題やリアルタイム性が必要であるという要件に対して，軽量な暗号 [11]，通信 [12] [14]，フォグを用いたことによる性能 [9]等，個々の仕組みに関する研究は多いが，これらを組み合わせた研究，実験は少ないと考える．そこで，表 1,2 であげたセキュリティ要件を満たすようなシステムを，フォグコンピューティングの考え方で構築し，セキュリティ要件の実現性検証，性能評価，安全評価を行い，課題の抽出を行うことを考えている．

5.2 検証内容

図 2 に構築した検証環境の構成を示す．また，表 3 に検証環境のスペックや利用している OS/ミドルウェア等の情報を示す．

Web カメラが接続された Raspberry Pi を IoT デバイス層とし，その上位にもう一台の Raspberry Pi をフォグ層に配置する．ここまでをローカルな通信とし，フォグ層からクラウド層の間はインターネットを経由する構成とした．なお，IoT デバイスについては，実際の市販されているネットワークカメラであると通信や暗号化などの仕組みがパッケージ化されていて検証できない可能性があるため，今回は仮想的な環境として Web カメラが接続された Raspberry Pi を IoT デバイスとした．

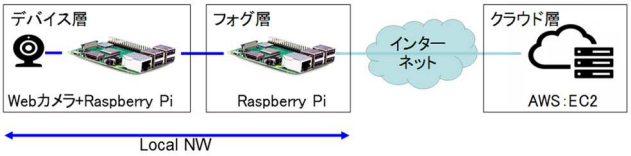

図 2　検証環境の構成

この検証システムの概要は以下の通りである．

- 監視カメラにより動体検知行い，画像を収集しクラウドサーバに保存．
- デバイスは貧弱であると仮定してフォグノードでセキュリティ機能を展開．
- クラウドとフォグの通信は軽量な通信(MQTT)を利用．
- ローカルネットワーク内は信頼できる前提．

このようなシステムにおいて，表 2 で述べたセキュリティ要件の実現性を確認する．各セキュリティ要件に対して，様々な対応方法があるが，本稿においては導入が容易であると考えられるものを選択し，実機でどのように対応したかを表 4 に示す．

表 3　検証環境のスペック，OS，ミドルウェア等

レイヤー	スペック	OS，ミドルウェア等
クラウド 	EC2 t2.micro CPU：vCPU　1 Memory：1GiB Disk：10GB 米国東部 (オハイオ) リージョン	Ubuntu16.04.5　LTS (GNU/Linux 4.4.0-1066-aws x86_64) mosquitto　version 1.4.8 mosquitto-clients Python 2.7.12 paho-mqtt zabbix-3.4.4
フォグ 	Rapberry Pi 3 Model B CPU　:　ARMv7 Processor rev 4 (v7l) Memory：1GB Disk：サンディスク microSDHC カード 32GB	Ubuntu16.04.5 LTS(Xenial Xerus) mosquitto-clients Python 2.7.12 paho-mqtt Suricata 3.0 zabbix-3.4.4
デバイス 	Rapberry Pi 3 Model B+LOGICOOL ウェブカム HD 画質 120 万画素 C270 CPU　:　ARMv7 Processor rev 4 (v7l) Memory：1GB Disk　:　Transcend microSDHC カード 16GB Memory　1 GB	Ubuntu16.04.5 LTS(Xenial Xerus) motion　Version 3.2.12(動体検知) zabbix-3.4.4

表 4　セキュリティ要件と実機での対応

セキュリティ要件	実機での対応
アクセス制御：各機器へのアクセスを制御	ネットワークアクセス：フォグで IDS/IPS，FW 物理的アクセス：PWD 認証
認証：ユーザ認証，サーバ認証，機器認証，メッセージ認証	クライアント，サーバ証明書，電子署名
暗号化：通信，データの暗号化	TLS
ウイルス対策：ウイルスの予防，検出，回復	予防：フォグで通信の遮断 検出，回復：監視をフォグで実施し異常を検知
アップデート：ソフトウェアの更新を容易に実施 ※対策が難しい場合は仮想パッチ等で前段階において攻撃を防御	仮想パッチの考え方（フォグで攻撃パケットをブロック）
侵入・攻撃対策：FW，IDS/IPS の導入，不正侵入検知，ログ取得	フォグで IDS/IPS，FW
バックアップ：定期的にデータのバックアップ	クラウドサーバで管理

ログ管理：アクセスログ，操作ログなどの管理	クラウドサーバでログ管理
監視：動作（正常，異常），許可していないプログラムの動作，ファイル構成の監視	クラウドがフォグを監視 フォグがデバイスを監視

検証環境に対する評価については，性能評価と安全性の評価を行うことを考えている．性能については，フォグを用いた構成と，フォグ経由しない構成で処理時間を比較するための基礎となるデータを収集する．安全性の評価については OSS ツールを用いて脆弱性診断を行い，構築したフォグノードの安全性を確かめると共に，DDoS 攻撃等を行い，脅威からデバイスを守ることができるか評価することを考えている．さらに，実機での検証の中で課題を抽出し，可能であればその対応策を立て，対策する．

5.3 検証結果

Web カメラで動体検知する基本機能を実現し，クラウドとデバイス間のフォグで FW，IDS/IPS，監視等のセキュリティ機能を導入した．特に監視については，表 5 の通り，セキュリティの観点で，不正アクセスを検知するためのアクセスログ監視や設定ファイル等ファイルの改ざんがないかの監視の実現性を確認するとともに，基本機能である動体検知プロセスの死活監視の実現性を確認した．

表 5　監視機能の実現性検証

監視対象	詳細設定
アクセスログ	/var/log/auth.log について，ssh アクセスがあればアラート通知する． トリガー詳細： {enddevice01:log[/var/log/auth.log].regexp(ssh)}=1
ファイル改ざん	設定ファイルなどについて，ハッシュをとり変更があればアラート通知する． トリガー詳細： {enddevice01:vfs.file.cksum[/var/tmp/test.log].diff(0)}>0
プロセスの死活	動体検知プロセスについて，プロセスが正常に起動していなければアラート通知する． トリガー詳細： {enddevice01:proc.num[motion].last(#1)}=1

図 3 に，設定ファイル等のファイル改ざんを検知した際のアラート例を示す．例えば，不正アクセス等により，デバイス上のファイルが改ざんされた場合に，図 3 のように管理者にアラートを挙げることを想定している．

次に，性能評価について述べる．図 4 に，フォグからクラウドのインターネット経由の ftp によるファイル転送速度と，デバイスからフォグへのローカルネットワーク内でのファイル転送速度を比較した結果を示す．なお，転送したファイルは，1MB 程度の jpg ファイルであり，1 回の転送にかかる全体時間を測定した．

今回の検証ではインターネット経由のファイル転送は，ローカルネットワーク内におけるファイル転送の 3 倍近くの時間を要することが分かった．つまり，フォグ層で，データの圧縮やデータの選別など一次処理を行い，クラウド層とやりとりを行うことが非常に有効であるということがわかる．なお，参考として，インターネット経由とローカルネットワーク内での ICMP の速度比較を図 5 に示す．

図 3　ファイルの改ざんを検知した際のアラート例

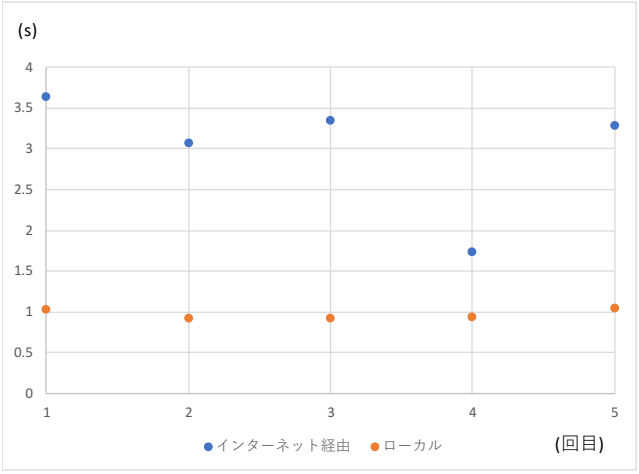

図 4　インターネット経由とローカルネットワーク内の ftp ファイル転送速度の比較

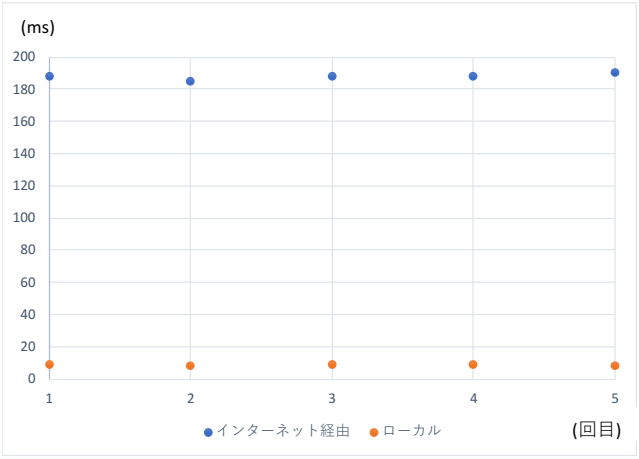

図 5　インターネット経由とローカルネットワーク内の ICMP 速度の比較

図 6 に MQTT において TLS の暗号化あり/なしでインターネット経由のファイル転送速度を比較した結果を示す．この結果より，暗号化することにより 2 割程度性能が劣化することがわかった．暗号化についてはマシンリソースを収集し，デバイス層でどの程度リソースを必要とするのか引き続き検証する．

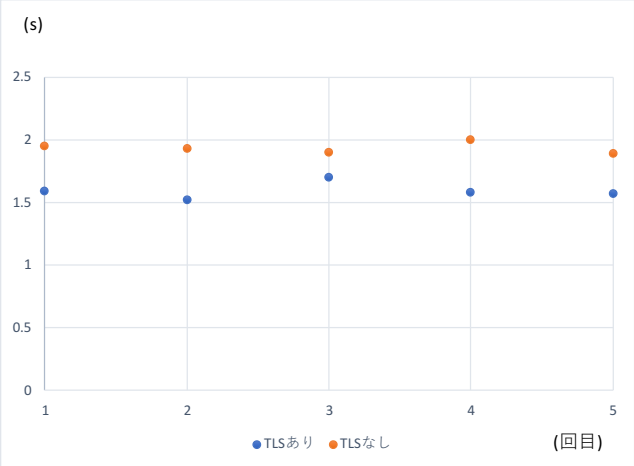

図 6　TLS あり/なしでの MQTT ファイル転送速度の比較

図 7 に, フォグからクラウドのインターネット経由の http によるファイル転送速度と，デバイスからフォグへのローカルネットワーク内でのファイル転送速度を比較した結果を示す. MQTT の TSL なしの転送速度と http によるインターネット経由のファイル転送速度を比較すると，今回の結果からは，MQTT の方が http よりも速い．ただし，ネットワークとしてインターネットを経由しているため，時間等によってネットワークの帯域/遅延等，大きな差異が生じることには注意したい．

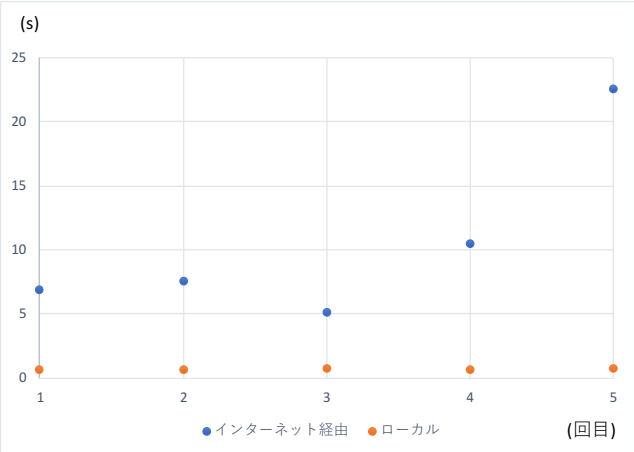

図 7　インターネット経由とローカルネットワーク内の http ファイル転送速度の比較

図 7 においては，インターネットを経由していたため，MQTT と http の純粋な差分を把握することができなかった．そこで，ネットワーク遅延等の影響を受けにくいローカルネットワーク内での http によるファイル転送速度と，MQTT によるファイル転送速度を比較した．その結果を図 8 に示す. 両者の差分はほぼないと考えられる. MQTT は，ヘッダサイズが http より小さく，短いメッセージを大量にやりとりする際に有利である [12]．よって，大きいサイズのリクエストや単発のリクエストでは，この特徴は現れないためと考える．今後，サイズの小さい複数回のリクエスト処理をした場合の性能差について引き続き検証する．

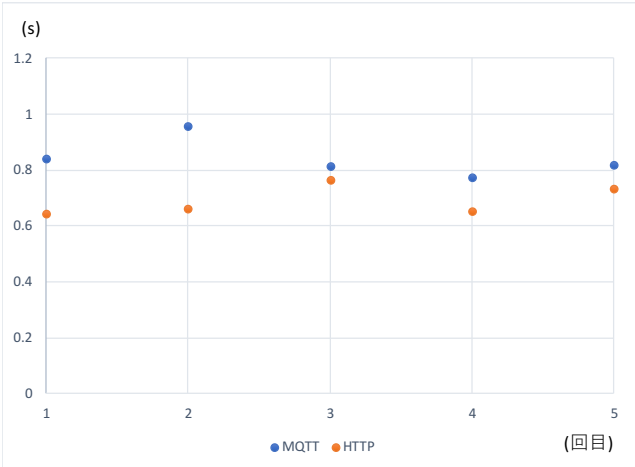

図 8　ローカルネットワーク内における MQTT と http のファイル転送速度の比較

6. おわりに

今後，IoT 機器が増えサービス化していく中で，IoT システム全体のセキュリティ設計は非常に重要となるが，リソースに制約のある IoT 機器にセキュリティやその他の要件を満たすようろいろな機能を持たせることは困難であると考える．セキュリティを強化し，脅威や攻撃から IoT システムを守るためには，フォグコンピューティングの構成が有効であると考えた．IoT システムへの脅威に対して，フォグ層において FW，IDS/IPS，監視等の機能を導入することで，さまざまな問題を解決できると想定している．

先行研究においては，IoT システムにおける基本的な要件やセキュリティ要件に対して，軽量な暗号や通信，フォグを用いたことによる個々の仕組みに関する研究は多いが，これらを組み合わせた研究や実験は少ないと考える．そこで，監視カメラを想定した実機検証を行い，要件の実現性検証，性能比較，安全性の確認，課題抽出を行い，IoT システムにおけるセキュアなアーキテクチャ設計のベストプラクティスについての研究を進めていく．

現状，監視カメラを題材として，基本機能を実現し，監視機能により，不正アクセスやファイルの改ざん等を検知する実現性を確認できた．また，実機での検証を通じて，ローカルネットワーク内の通信とインターネット経由の通

信とではかなり差が出ることが具体的に把握でき，フォグノードで一次処理(圧縮処理,クラウドへの送信データの選別等)を行うことが非常に有効であることがわかった．

今後，暗号化する際に，マシンリソースを実際にどの程度必要なのか等，実機での検証を行うと共にセキュリティ機能を導入したことによる安全性の評価を行い，システム上留意すべき点を整理し，必要に応じて改善していく予定である．その中で，最低限デバイスで持っていなければならないリソースやフォグ層でどの機能を持つべきか役割分担を再度明確化すると共に，フォグコンピューティングにおける処理性能向上やノードが増えることによるコスト増など，メリットデメリット整理していく．

参考文献

[1] 総務省，“平成 29 年版情報通信白書,” 2017. http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h29/pdf/n3300000.pdf.

[2] IPA, “ＩｏＴ開発におけるセキュリティ設計の手引き,” 4 2018. https://www.ipa.go.jp/files/000052459.pdf.

[3] CISCO, “フォグ コンピューティング,” https://www.cisco.com/c/m/ja_jp/solutions/internet-of-things/iot-system-fog-computing.html.

[4] 長柄啓悟, “組込みシステム向けマルウェア Mirai の攻撃性能評価,” 情報処理学会, 2017.

[5] Trend Micro, “「Mirai」による DDoS 事例から IoT の「エコシステム」を考察する,” 2016. http://blog.trendmicro.co.jp/archives/13956.

[6] D. V. D. Brian Russell, Practical Internet of Things Security, 2016.

[7] 八子知礼, IoT の基本・仕組み・重要事項が全部わかる教科書, SB クリエイティブ, 2017.

[8] K. S. W. W. T. A. Y. Ranadheer Errabelly, “EdgeSec: Design of an Edge Layer Security Service to Enhance IoT Security,” IEEE, 2017.

[9] P. R. R. M. I. N. S. Z. Y. Salem Alharbi, “Secure the Internet of Things with Challenge Response Authentication in Fog Computing, ” IEEE 36th International Performance Computing and Communications Conference (IPCCC), 2017.

[10] R. M. L. S. Mithun Mukherjee, “Security and Privacy in Fog Computing: Challenges,” IEEE Access, 2017.

[11] N. C. Y. N. Abebe Abeshu Diro, “Analysis of Lightweight Encryption Scheme for Fog-to-Things Communication,” IEEE Access 2018, 2018.

[12] I. A. Z. Mohammed Al-Soh, “An MQTT-based Context-aware Wearable Assessment Platform for Smart Watches,” 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT), 2017.

[13] A. C. M. V. M. F. A. P. Luciano Barreto, “An Authentication Model for IoT Clouds,” IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2015.

[14] D. X. P. W. Heng Wang, “A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices,” IEEE Access, 2017.