IoTデバイスの通信セキュリティ向上のための ホームネットワーク仮想化フレームワークの提案

塚崎 拓真[1] 滕 睿[2] 佐藤 健哉[1]

概要：近年，IoT(Internet of Things) が注目を集めるようになり，今後あらゆるモノがネットワークに接続され，利用されることが予想される．しかし，IoT の発展により利便性が高まる一方で，セキュリティ上のリスクも高まっている．IoT デバイスはリソース制限により，適用できる機能が限られるという問題点があり，セキュリティ対策の適用は困難である．また，今後はホームネットワーク内で閉じたデバイス間の通信によって連携を行う形になることが想定され，各デバイスにおいてアクセス制御等の更なるセキュリティ対策を行う必要がある．そこで本研究では，各 IoT デバイスに柔軟にセキュリティ対策を適用するために，コンテナを用いて IoT デバイス間通信を中継することで，各デバイスに対して，仮想的にセキュリティ対策を適用できるシステムを提案した．また，ホームネットワーク内の通信のトラフィック情報は既知であるため，提案システムでは OpenFlow を用いて，ホームネットワーク内の通信を監視するフレームワークの構築も検討した．そして，IoT デバイス間で閉じた通信を行うシミュレーションの評価を行い，ホームネットワークにおいてセキュリティ要件を保つことを示した．

Proposal of Home Network Virtualization Framework to Improve Communication Security of IoT Devices

TAKUMA TSUKASAKI[1] RUI TENG[2] KENYA SATO[1]

1. はじめに

近年，IoT(Internet of Things) の可能性が注目され，今後あらゆるモノがネットワークに接続され，利用されることが予想される [1]．あらゆるモノがネットワークに接続されることで，ビッグデータを収集し，活用することが可能となる．また，ローカルネットワーク内においても，様々な IoT デバイスの普及により，従来，相互接続されていなかった機器同士の接続が想定される．このような変化により，生活や産業の効率・利便性を高めることが期待されている [2]．

しかし，IoT の発展により利便性が高まる一方で，従来ネットワークに接続されていなかったモノが接続されることにより，セキュリティ上のリスクも高まっている [3]．IoT デバイスは，十分なセキュリティを考慮せずに開発されたものが多く，脆弱なパスワードによる侵入やプライバシー保護の不十分さ等のセキュリティ対策不足が顕著である [4]．そのため，悪意のある攻撃者によるサイバー攻撃の標的になりやすい．脅威としては，ホームネットワークに侵入し，デバイスの遠隔操作による外部サーバへの攻撃や，マルウェア感染によるプライバシーに関わる機密情報の収集などが挙げられる．また，攻撃によってホームネットワーク内に侵入された場合，その内部においてもデバイス間で自由にアクセスできるため，マルウェア感染などのリスクがホームネットワーク全体のデバイスに広がる可能性がある．そのため，ホームネットワークのセキュリティ対策として，内部への侵入を前提に，攻撃を受けた際に被害を最小化することが望まれる．

しかし，IoT デバイスは従来の PC 等の既存機器と比較した場合，CPU 等のリソースを十分に保持していない [5] ため，デバイスの計算能力の制限やソフトウェア自体の脆弱性によって，適用できる機能が限られるという問題があ

[1] 同志社大学大学院 理工学研究科
Graduate School of Science and Engineering, Doshisha University
[2] 同志社大学モビリティ研究センター
Mobility Reserch Center, Doshisha University

る [6]．そのため，暗号化等のセキュリティ対策の適用は困難となり，全デバイスが接続するネットワークを利用したシステムを構築することや，仮想的にセキュリティ対策を施し，デバイスのリソース制限に捉われないシステムを構築することが望まれる．

そこで本研究では，コンテナ上にセキュリティ対策を施した Proxy を作成し，IoT デバイスに対して，仮想的にセキュリティ対策を適用するシステムを提案する．また，SDN(Software Defined Networks) の代表的プロトコルである OpenFlow[7] を用いて，ホームネットワーク内の通信を監視するフレームワークの構築も検討する．

2. 関連研究

2.1 フローレベルの攻撃検知・防止

Sivanathan らは，SDN と外部の解析エンジンを用いて，IoT デバイスのネットワークを常に監視し，フローレベルでのトラフィック検査を行う手法を提案した [8]．IP アドレスやポート番号等のフロー情報から攻撃を検知できることを示し，パケットベースのネットワーク監視と比較し，処理コストの大幅な削減を実現した．

しかし，ホームネットワーク内のトラフィック情報検査を外部で行っていることが問題点として挙げられる．今後の IoT デバイスは，ホームネットワーク内で閉じたデバイス間の通信によって，相互の連携を行う形になることが想定される [9]．デバイス間で直接通信を行う場合，各デバイスにおいてアクセス制御等のセキュリティ対策を行う必要がある．

2.2 ホームネットワーク運用の外部依存を避けた自己完結型システム

Zhang らは，クラウド上の遠隔サーバから制御されている現状のホームネットワークの問題点を挙げ，IoT デバイスの管理をクラウドや外部サービスに依存しないホームネットワークシステムを提案した [10]．トラストアンカーをローカルコントローラに置くことで，デバイスやアプリケーションの管理や，データ検索等の承認，制御をホームネットワークシステム内で行うことを可能にした．その結果，クラウドを用いたシステムより，遅延を低減した．

しかし，各 IoT デバイスに対応したセキュリティ対策を施す柔軟性を持ち合わせていない．ホームネットワーク内には，異なる規格のハードウェアや様々なアプリケーションが混在しているため，各デバイスに対して柔軟にセキュリティ対策を適用できるシステムを構築することが望まれる．

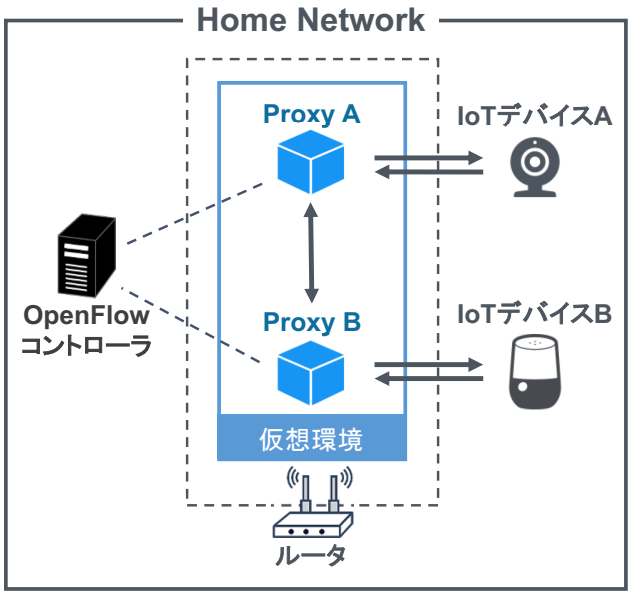

図 1　提案システムの構成

3. 提案システム

3.1 概要

本研究では，コンテナ上にセキュリティ対策を施した Proxy を作成し，IoT デバイスに対してセキュリティ対策を適用するシステムを提案する．提案システムでは，IoT デバイスがリソース量の制限により適用できないセキュリティ対策を Proxy にオフロードする．そして，IoT デバイス間の通信を中継することで，本来 IoT デバイスに適用したいセキュリティ対策を実現する．また，OpenFlow の機能も追加し，ホームネットワーク内通信のトラフィック情報は既知であることから，フロー検証等のネットワーク監視を OpenFlow コントローラで行う．詳細なセキュリティ対策については後述する．提案システムでは，IoT デバイスと Proxy 間の接続要求はユーザが行うものとする．

3.2 システム構成

提案システムの構成を図 1 に示し，詳細を以下に示す．本提案システムは，IoT デバイス，Proxy，ルータ，OpenFlow コントローラから構成される．

- IoT デバイス
  本研究で扱う IoT デバイスは，CPU 等のリソースを十分に保持しておらず，直接セキュリティ対策を適用できないデバイスと定義する．
- Proxy
  IoT デバイスに要求されるセキュリティ対策を，コンテナ上で実現したものである．そして，IoT デバイスからの通信を中継し，セキュリティ対策を適用する．各 IoT デバイスに必要なセキュリティ対策をそれぞれ作成，適用することで，対象デバイスに応じた必要な

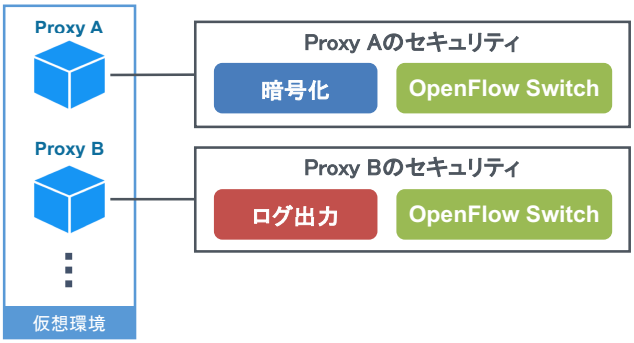

図 2　Proxy のセキュリティ対策

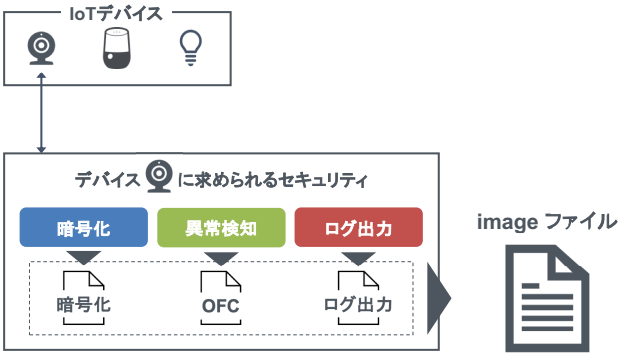

図 3　image ファイルの作成

セキュリティ対策を実現できる．

- ルータ
  IoT デバイス間通信の中継機器として用いる．ルータ上に Proxy の実行環境を生成する．Proxy が作成される際に必要とされるリソースを提供することが可能である．
- OpenFlow コントローラ
  Proxy 内に作成された OpenFlow スイッチと通信を行い，ホームネットワーク内の通信を監視する．ホームネットワーク内部に設置する．

3.3　Proxy のセキュリティ対策

本研究におけるセキュリティ対策は，Proxy ごとに異なるセキュリティ対策を適用可能である．Proxy のセキュリティ対策の適用例を図 2 に示す．これにより，リソース制限が原因で IoT デバイスに直接適用できないセキュリティ対策を適用できる．また，前述の問題点で述べたような様々なハードウェアやアプリケーションへの適用や，様々なセキュリティ要件の変更に対しても柔軟に対応が可能となる．

また，コンテナ上で作成されるセキュリティ対策は，各セキュリティ対策に対応したコンテナの image ファイルで定義される．image ファイルの作成例を図 3 に示す．IoT デバイスに適用したいセキュリティ対策が複数ある場合においても，対象デバイスの規格に対応したセキュリティ対策をそれぞれ作成し，ソフトウェアモジュールのような形で組み合わせて定義することで，image ファイルを作成することが可能となる．

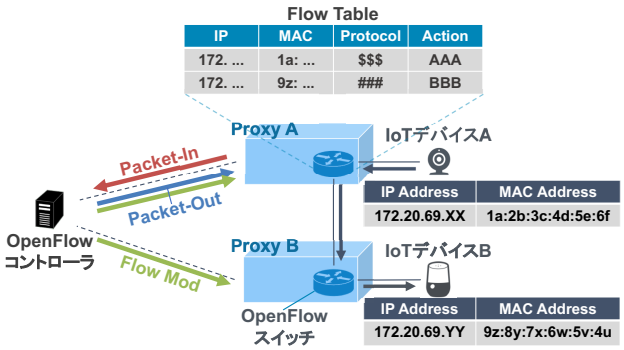

図 4　OpenFlow におけるフローチェック

3.4　OpenFlow によるフローチェック

本研究におけるネットワーク監視を OpenFlow を用いて行う．OpenFlow によるフローチェックを図 4 に示す．一つの IoT デバイスに対し，コンテナ上に OpenFlow スイッチの機能を生成する．IoT デバイスは，この OpenFlow スイッチを中継し，デバイス間通信を行う．OpenFlow コントローラは事前に IoT デバイスの情報を保持しており，OpenFlow スイッチとの通信が確立でき次第，デバイス間通信のフローテーブルを作成する．ホームネットワークの特性である各 IoT デバイスのトラフィック情報は既知であることや，変化が大きくないことを考慮し，IP アドレスや通信頻度の確認を行い，フローレベルにおける異常の検知を行う．異常を検知した場合は，そのパケットの Drop 処理を指示したフローテーブルに変更し，その後フローテーブルを削除する．また同時に，ユーザへ異常検知を通知する．

3.5　動作手順

本提案システムを利用する際のユーザの利用手順と，デバイスと Proxy の通信が確立するまでの提案システムの処理手順を図 5 に示し，詳細を以下に示す．

(1) ユーザは，デバイスをホームネットワークに接続した後，Proxy の実行環境へアクセスし，Proxy を作成するよう要求．
(2) 実行環境は，ホームネットワーク内に IoT デバイスが接続されたことを確認．
(3) 実行環境は，image ファイルのレジストリから image ファイルを取得．
(4) 取得した image ファイルを基に，実行環境内に Proxy を作成．
(5) Proxy は，デバイス情報を基に IoT デバイスに接続を行い，Proxy を経由して通信を行うように設定．

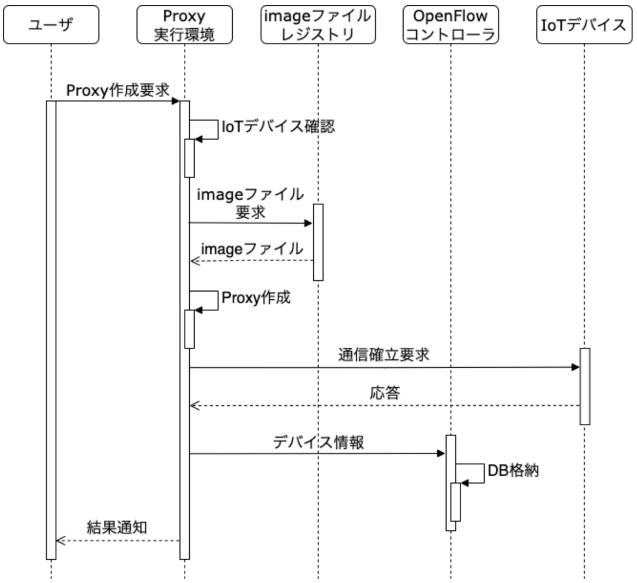

図 5　提案システムの動作手順のシーケンス図

表 1　実装環境

種類	項目	説明
Proxy	使用ソフト	Docker
	OS	Ubuntu 20.04
	CPU	3.60GHz Intel Core i9
	メモリ	5GB
OpenFlow コントローラ	使用ソフト	Ryu
	OS	Ubuntu 20.04
	CPU	3.60GHz Intel Core i9
	メモリ	5GB
	使用言語	Python

(6) Proxy は，OpenFlow コントローラへデバイス情報を送信．

(7) 実行環境は，Proxy の設定が完了し，IoT デバイス・Proxy 間の通信が確立された後，Proxy の作成状況を報告．

4. シミュレーション実験

4.1 実装環境

本研究の実装に用いた環境を表 1 に，実装環境の構成を図 6 に示す．Proxy は，コンテナ仮想化を用いてアプリケーションを開発・実行するためのオープンプラットフォームである Docker を用いて作成した．Docker で作成されるコンテナ上で Proxy を稼働させることで，複数の Proxy をリソースやオーバーヘッドを抑えて作成できる．また，これらの Proxy は，Docker Hub より配布される Docker Image を用いることで容易に作成でき，コンテナをアップロードして公開・共有できる．また，OpenFlow コントローラは，SDN を実現するための開発フレームワークである Ryu を用いて作成した．

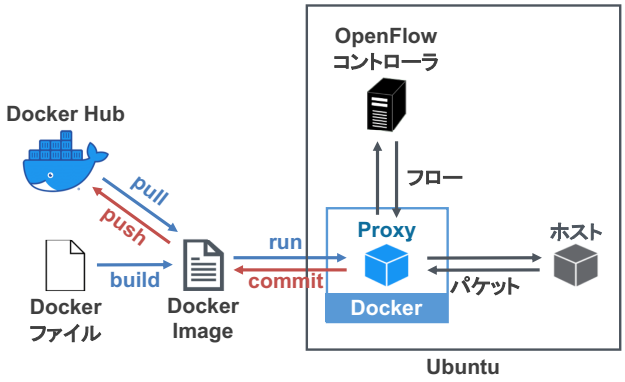

図 6　実装環境の構成

4.2 評価内容

本研究の評価として，まずセキュリティ対策が適用されているかを検証した．今回の異常通信としては，登録されていない IoT デバイスから通信があった場合と，あるデイバイスからの通信頻度が通常と異なる場合を想定した．その状況において，コンテナ上で作成した OpenFlow によるフローチェックが行われているかを検証した．

また，提案システムの有効性を示すため，提案システムを適用した上で，IoT デバイス間通信を 20 回行なった際のラウンドトリップタイムを計測した．比較対象として，セキュリティ対策を適用せず，ルータを経由してデバイス間通信を行う場合についても計測を行った．

4.3 評価環境

今回のシミュレーション実験においては，IoT デバイスや通信に関するログの収集・出力機能を実装し，セキュリティ対策として適用した．また，OpenFlow スイッチのイメージも取得し，OpenFlow によるフローチェックも行った．事前に 2 台のホストを設置した環境に，新たに 1 台ホストを追加し，そのホストが通信要求を行う環境を作成した．

5. 結果と考察

5.1 評価結果

登録済みの IoT デバイスから通信要求が来た場合，登録していない IoT デバイスや，通常の通信頻度と異なる等の異常の通信がなされている場合のフローテーブルの結果を図 7 に示す．通常時は他のデバイスに対し，通信を許可するフローテーブルが作成されている．一方で，異常時はパケットを Drop 処理するフローテーブルが作成されており，その後，そのフローテーブルが削除されていることがわかる．

また，セキュリティ対策を施した提案システムとセキュリティ対策を施してないシステムにおけるデバイス間通信の比較結果を図 8 に示す．提案システムは，セキュリティ

```
*** s3 ----------------------------------------------------------------------
cookie=0x0, duration=41.578s, table=0, n_packets=11, n_bytes=1022, priority=1,in_port="s3-eth1",dl_dst=86:49:46:a5:a8:74 actions=output:"s3-eth2"
cookie=0x0, duration=12.852s, table=0, n_packets=10, n_bytes=924, priority=1,in_port="s3-eth2",dl_dst=4e:98:97:5f:fc:6e actions=output:"s3-eth1"
cookie=0x0, duration=89.081s, table=0, n_packets=73, n_bytes=9112, priority=0 actions=CONTROLLER:65535

*** s3 ----------------------------------------------------------------------
cookie=0x0, duration=6.029s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s3-eth1",dl_dst=86:49:46:a5:a8:74 actions=output:"s3-eth2"
cookie=0x0, duration=6.020s, table=0, n_packets=3, n_bytes=294, priority=10,in_port="s3-eth2",dl_dst=4e:98:97:5f:fc:6e actions=drop
cookie=0x0, duration=53.532s, table=0, n_packets=65, n_bytes=8258, priority=0 actions=CONTROLLER:65535

*** s3 ----------------------------------------------------------------------
cookie=0x0, duration=11.782s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s3-eth1",dl_dst=86:49:46:a5:a8:74 actions=output:"s3-eth2"
cookie=0x0, duration=59.285s, table=0, n_packets=65, n_bytes=8258, priority=0 actions=CONTROLLER:65535
```

図 7 登録済みのホストのフローテーブル (上)，異常時のパケットを Drop 処理するフローテーブル (中)，その後のアクションが削除されたフローテーブル (下)

対策を施していないシステムにおける通信より，ラウンドトリップタイムの平均値が約 1.06ms，最大値が約 10.62ms 大きくなっている．

5.2 信頼性に関する考察

ホームネットワーク内においても，IoT デバイス情報を基に，フローレベルで制御できることを示した．今回は，送信元 IP アドレスや通信頻度の情報を用いたが，宛先 IP アドレスやメッセージ情報を用いた追加のフロー制御を行うことで，更にセキュリティを高めることが可能である．

IoT デバイスを用いたシステムの安心安全を確保するための機能として，IPA により IoT 高信頼化機能が定義されている．また，IoT 高信頼化要件として，開始，予防，検知，回復，終了の 5 つの局面に分けてそれぞれセキュリティ要件が定義されている [11]．今回は前述の 5 つより，システム稼働中の局面である予防，検知，回復の 3 つにおける高信頼化要件に対し，提案システムの信頼性について考察する．

- 予防の局面における考察
  予防の局面での高信頼化要件は，稼働中の異常発生を未然に防止できることである．これに対応する IoT 高信頼化機能としては，ログ収集機能，暗号化機能等があり，異常の予兆の把握や機能・資産の保護を実現する．提案システムを用いることで，リソース量の制限により IoT デバイスに適用できない機能であっても適用可能となる．
- 検知の局面における考察
  検知の局面での高信頼化要件は，稼働中の異常発生を早期に検知できることである．これに対応する IoT 高信頼化機能としては，OpenFlow によるネットワーク監視機能，ログ収集機能があり，異常発生の検知や発生原因の特定を実現する．提案システムを用いることで，予防の局面同様，デバイスのリソース量に依存せず，要求機能を実現できる．また，Proxy は各 IoT デバイスごとに作成するため，個々のデバイスに応じた検知対策を適用可能となる．
- 回復の局面における考察

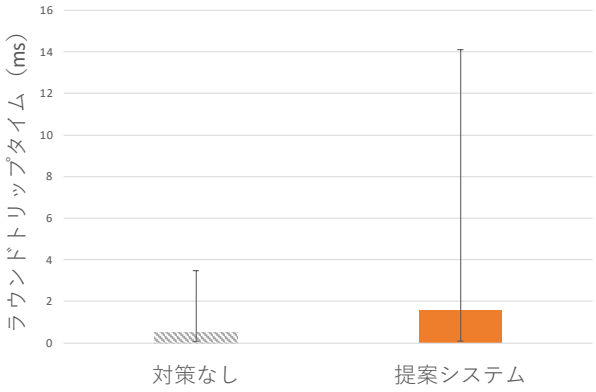

図 8 デバイス間通信の平均ラウンドトリップタイム

回復の局面での高信頼化要件は，異常が発生した場合に稼働の維持や早期の復旧ができることである．ホームネットワークでは，様々なデバイスが相互通信を行うため，事前に予測していなかったインシデントが発生することが想定される．本提案システムにおいて，各セキュリティ対策は Docker Hub から Docker Image として提供する．そのため，事前に作成したセキュリティ対策だけでなく，追加のセキュリティ対策も配布・適用することが容易である．従って，システムの早期復旧の実現が可能となる．

以上より，本提案システムは，システム稼働中の局面における高信頼化要件を満たしているため，信頼性を保っていることがわかる．

5.3 通信性能に関する考察

本提案システムは，セキュリティ対策なしの場合と比較し，平均値では約 1.06ms，最大値では約 10.62ms の遅延が発生した．これは，Docker によって適用されたセキュリティ対策や OpenFlow によるフローチェックを行った際の負荷によるものである．しかし，M2M(Machine to Machine) 接続におけるデバイス遠隔操作の遅延要件は約 100ms とされている [12] ため，許容範囲内である．

6. まとめ

本研究では，IoT のセキュリティ上のリスクにおいて，

今後，ホームネットワーク内で閉じたデバイス間通信が多くなり，各 IoT デバイスにおいてアクセス制御等の更なるセキュリティ対策を行う必要があることに注目した．そこで，コンテナを用いた IoT デバイスへのセキュリティ対策の適用と，OpenFlow を用いたホームネットワーク監視を行うフレームワークの構築を提案した．提案システムでは，コンテナ上に Proxy を作成し，ログ出力や暗号化，OpenFlow スイッチ等のセキュリティ対策をオフロードし，IoT デバイス間の通信を中継することで，本来 IoT デバイスに適用したいセキュリティ対策を実現した．そして，IoT デバイス間で閉じた通信を行うシミュレーション評価の比較を行い，提案システムはホームネットワークにおいてセキュリティ要件を保つことと，通信性能も許容範囲であることを示した．

今後は，ユーザの利便性を考慮し，オーケストレータ等を用いて，新しい IoT デバイスがホームネットワークに追加された際に，自動的にコンテナが Proxy として配備される仕組みを検討する．また，IoT デバイスが普及した際には，クラウドへの通信やデバイス間の通信が増加するため，帯域の輻輳や通信の干渉などの問題も発生すると予想される．そのため，より実環境に近い検証を行うために，デバイスの増加を想定した環境や，Raspberry Pi 等の実機を用いた提案システムの性能検証を行う予定である．

参考文献

[1] Sinha, S.: State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion, IOT ANALYTICS (online), available from ⟨https://iot-analytics.com/number-connected-iot-devices/⟩ (accessed 2022-04-02).

[2] 境野哲：IoT への期待と課題　〜IoT システム開発者・利用者の心得〜，情報の科学と技術，Vol.67，No.11，pp.560-565（2017）．

[3] 総務省：IoT・5G セキュリティ総合対策 2020，総務省（オンライン），入手先 ⟨https://www.soumu.go.jp/main_content/000698567.pdf⟩（参照 2022-03-28）．

[4] Ferrara, P., Mandal, A.K., Cortesi, A. and Spoto, F.: Static analysis for discovering IoT vulnerabilities, International Journal on Software Tools for Technology Transfer, Vol.23, No.1, pp.71-88(2021).

[5] Abdalla, P.A. and Varol, C.: Testing IoT Security: The Case Study of an IP Camera, 2020 8th International Symposium on Digital Forensics and Security (ISDFS), pp.1-5(2020).

[6] Serror, M., Henze, M., Hack, S., Schuba, M. and Wehrle, K.: Towards In-Network Security for Smart Homes, Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018), No.18, pp.1-8(2018).

[7] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J.: OpenFlow: enabling innovation in campus networks, SIGCOMM Computer Communication Review, Vol.38, No.2, pp.69-74(2008).

[8] Sivanathan, A., Sherratt, D., Gharakheili, H.H., Sivaraman, V. and Vishwanath, A.: Low-cost flow-based security solutions for smart-home IoT devices, 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp.1-6(2016).

[9] Pawar, P. and Trivedi, A.: Device-to-Device Communication Based IoT System: Benefits and Challenges, IETE Technical Review, Vol.36, No.4, pp.362-374(2019).

[10] Zhang, Z., Yu, T., Ma, X., Guan, Y., Moll, P. and Zhang, L.: Sovereign: Self-contained Smart Home with Data-centric Network and Security, IEEE Internet of Things Journal, pp.1-15(2022).

[11] 独立行政法人情報処理推進機構（IPA）技術本部　ソフトウェア高信頼化センター（SEC）：「つながる世界の開発指針」の実践に向けた手引き［IoT 高信頼化機能編］，独立行政法人情報処理推進機構（IPA），pp.1-96（2017）．

[12] Warren, D. and Dewar, C.: Understanding 5G: Perspectives on future technological advancements in mobile, GSMA Intelligence (online), available from ⟨https://www.gsma.com/futurenetworks/wp-content/uploads/2015/01/2014-12-08-c88a32b3c59a11944a9c4e544fee7770.pdf⟩ (accessed 2022-4-13).