ブロックチェーン技術を用いたIoT機器向けセキュアアップデートフレームワーク

長柄 啓悟[†1,a)]　松原 豊[†1]　高田 広章[†1]

概要：近年，Internet of Things (IoT) 機器が普及し，その脆弱性も明らかになっている．その脆弱性を悪用して，IoT 機器を分散型サービス妨害攻撃の踏み台や不正マイニングに利用する攻撃も確認されている．そのため，脆弱性の修正や機能の向上が行えるセキュアなソフトウェアアップデートが必要である．本研究では，既存のフレームワークを元に，ブロックチェーン技術を活用した IoT 機器向けのセキュアアップデートフレームワークの提案と開発を行った．ブロックチェーン技術を用いることにより，耐改ざん性があり，低コストなフレームワークが実現できる．

キーワード：Internet of Things(IoT), セキュリティ，ソフトウェアアップデート，ブロックチェーン技術

A secure update framework using a blockchain for IoT devices

Keigo Nagara[†1,a)]　Yutaka Matsubara[†1]　Hiroaki Takada[†1]

Abstract: In recent years, the Internet of Things (IoT) equipment has become widespread, and its vulnerability is also clarified. Attacks that exploit the vulnerability of IoT devices for stepping stones of distributed denial of service attacks and illegal mining have also been confirmed. Therefore, a secure software update that can fix vulnerabilities and improve functions is required. In this research, based on the existing framework, we have proposed and developed a secure update framework for IoT devices utilizing the blockchain. By using the blockchain, the tamper-resistant and low-cost framework can be realized.

Keywords: Internet of Things(IoT), Security, Software update, Blockchain

1. はじめに

近年，Internet of Things (IoT) 機器が普及し，その脆弱性も明らかになっている．汎用的なシステムと比較して，リソース制限や多様性がある組込みシステム・IoT 機器において，脆弱性は発生しやすい．その IoT 機器の脆弱性を悪用し，分散型サービス妨害 (DDoS) 攻撃の踏み台や不正マイニングに利用する攻撃も確認されている [1], [2]．IoT 機器が長期間使用される間に，新たな脆弱性や脅威が発見される可能性は高い [3].

そのため，脆弱性の修正や機能の向上・追加といった事後対応が可能なソフトウェアアップデートフレームワークが必要となっている [4], [5]．また，そのフレームワークの要求として，アップデートファイル自体が改ざんされないというセキュリティの必要性が挙げられる．そして，そのフレームワークが，企業や製品を問わずに利用できることが望ましい．低コストで高可用性であるフレームワークがあれば，開発者は，企業や製品ごとにアップデートの仕組みを維持・管理する必要がなくなる．他にも，IoT 機器の長期使用に伴って，開発者がその稼働状況やアップデート適用情報を把握できる機能があることが望ましい．

本研究では，既存のフレームワークを元に，ブロックチェーン技術を活用した IoT 機器向けのセキュアアップデートフレームワークの開発を行った．ブロックチェーン技術とは，分散型台帳技術とも言われ，情報を格納されたリストであるブロックが，タイムスタンプと前のブロックのハッシュ値を持ち，チェーンのように連なっていくデータベースのような技術である．この技術は，分散型データベースといった既存技術と比較して，高い整合性や耐改ざん性を持ったシステムを低コストに構築できるというメリットを持つ．この技術を用いた本アップデートフレームワークにより以下のような利点が挙げられる．

[†1] 現在，名古屋大学
[a)] nagara@ertl.jp

(1) 耐改ざん性が高いブロックチェーンにより，高信頼性なフレームワークが実現できる [6]．ブロックチェーンは，理論上，記録を改ざんすることが不可能であり，Peer to Peer(P2P) ネットワークにより，中央サーバや管理者なしに稼働するためである [7]．
(2) 非中央集権型のブロックチェーンは，ダウンタイムがなく堅牢であるため，フレームワークにおけるシステムの保守・運用のコストが抑えられる．既存システムと比べ，同等のセキュリティを担保するためのプロシージャの時間と費用が削減できる．
(3) ブロックチェーンのデータをアップデート状況情報として二次利用することが可能である．ブロックチェーン内のデータを検索することで，開発者は機器の稼働状況やアップデート状況を把握することが可能である．
(4) クライアントサーバ方式によるアップデートフレームワークと比較して，サーバの負荷を各ノードに分散させることができる．PC やアクセスポイントをブロックチェーンのノードとして活用することも可能である．
(5) 後述の既存フレームワークに対し，PC/アクセスポイントを中継として利用し，プロシージャを IoT 機器ごとに独立させることで，その通信量と処理を減らし，軽量な組込みシステムにおいても適用できる．

本フレームワークの概要が 2 章，実装の説明が 3 章，まとめが 4 章になる．

2. アップデートフレームワーク概要

2.1 全体概要

本節では，本研究におけるアップデートフレームワークの全体の概要について説明する．フレームワークのネットワーク対象として，開発者がアップデートファイルを置く Github のようなサーバ，そのファイルのテストを自動で行う Continuous Integration(CI) サーバ，フレームワークの管理サーバ，ローカル環境にある PC もしくはアクセスポイント，IoT 機器があると想定した（図 1）．また，本研究では，CI サーバのチェックを完了したアップデートファイルが，Github に置かれ，開発者による管理サーバへのアップデート開始通知を持ってアップデートを開始することを前提とした．

PC/アクセスポイントから，その通知を得た管理サーバへのバージョンチェックを行い，アップデートファイルを入手する．その後，ローカル環境内でアクセスポイントから IoT 機器へのアップデートを On The Air で行い，ブロックチェーンによる記録を行うサーバへ通信を行い，ブロックに記録するという流れが，本フレームワークの主なプロシージャになる（図 2）．本フレームワークでは，IoT 機器の限られたリソースや環境でも適用できるように，PC/アクセスポイントを中継してアップデートを行った．

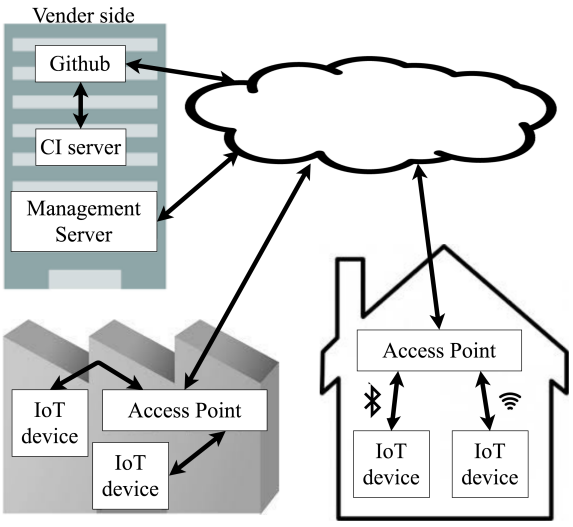

図 1　アップデートフレームワークの対象ネットワーク

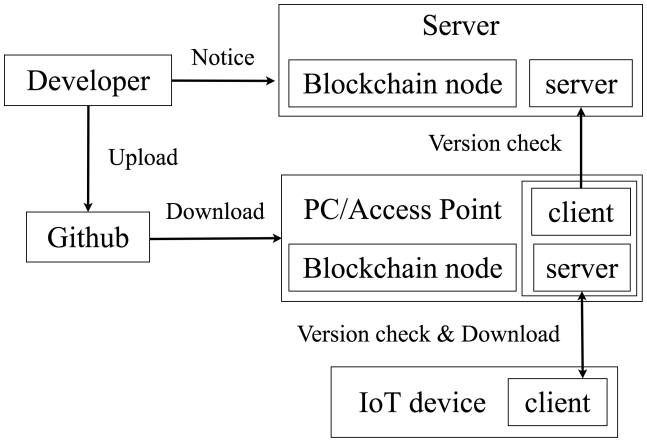

図 2　提案フレームワーク概要

2.2 既存フレームワーク

本節では，本フレームワークの開発にあたって参考にした先行研究のフレームワークについて説明する．この先行研究は，IoT 環境における組込みシステムのセキュアファームウェアアップデートの提案を行った研究 [6] である．この先行研究のフレームワークでは，IoT 機器と IoT 機器，もしくは IoT 機器とトラッカーの役割を果たすサーバの間で，P2P 通信を行い，アップデートファイルのチェック・配布を行い，他の IoT 機器のブロックチェーンのノードへそのログを書き込む．このフレームワークでは，バージョンチェックと必要に応じたアップデートにより，攻撃可能な時間の最小化を行い，また，分散モデル上で全プロシージャを実現している．

この既存フレームワークは，ノーマルノードとバリッドノードとされるブロックチェーンノードと，バリッドノードを管理するベンダーノードからなる．ノーマルノードからのブロードキャストによるリクエストに応じて，各ノードがレスポンスを行い，バージョンやファームウェアのハッシュ値の比較によるアップデートファイルのチェッ

表 1　先行研究と本研究のフレームワークにおける機器ごとに持つ役割の比較

機能		IoT 機器		PC/アクセスポイント		サーバ	
		先行研究	本研究	先行研究	本研究	先行研究	本研究
バージョンチェック		✓	✓	–	✓	✓	✓
アップデートファイル保持		✓		–	✓	✓	
ブロックチェーン	トランザクション	✓		–	✓		✓
	マイニング	✓		–	✓	✓	✓

クを行う．ノーマルノード同士での比較の場合，ファームウェアのチェックを行い，お互いのファームウェアのバージョンが最新である場合はその結果をブロックチェーンに記録する．結果が異なる場合は，バリッドノードからの情報を元に，ファイル転送用プロトコルおよびそのソフトウェアである BitTorrent を用いて，最新のファームウェアを他の IoT 機器から入手する．バリッドノードはピアリストで管理するファームウェアのトラッキングを行い，P2P 通信の補助を行う．ベンダーノードは，ブロックチェーンのノードでなく，バリッドノードに対し，最新のファームウェアのハッシュ値を与える．

この先行研究では，このフレームワークの欠点として，サーバ補助を担うバリッドノードの負荷が大きい点や，各ノードへの物理的な攻撃やファームウェア改ざんの影響を受ける点，要求の開始がブロードキャストのため無駄な通信や処理の発生している点が挙げられている．他にも，全 IoT 機器がアップデートファイルを保持し，ブロックチェーンノードの役割を果たしており，高性能な IoT 機器のみを対象に想定したフレームワークであると考えられる．

このフレームワークの実装については，この研究論文を参考に行い，本研究におけるフレームワークの参考とした．

2.3　本研究のフレームワーク

本節では，本研究で提案するアップデートフレームワークの概要について説明する．本研究と先行研究のフレームワークを比較したものが，表 1 になる．この表の比較について，以下で説明する．

バージョンチェックにおいて，先行研究では他の IoT 機器やサーバへのブロードキャストによって通信を行っているが，この手法では無駄な通信や処理が発生する．そこで，本研究では，IoT 機器から最新バージョンのアップデートファイルを持つアクセスポイントへ確認を行うプロシージャを用いる．IoT 機器の起動時には，ローカル環境にある PC/アクセスポイントに対してバージョンチェックを行い，最新のアップデートファイルがある場合はそれをダウンロードしアップデートを行う．また，アクセスポイントからは，一日一回と言った頻度で，サーバに対してバージョンチェックを行い，最新のアップデートファイルがある場合はそれをダウンロードして保持する．これにより，IoT 機器が常にアップデートファイルを保持する必要や配布する必要がなくなり，より軽量な組込みシステムにおいても使用することが可能である．そして，同様の理由から，ブロックチェーンのマイニングにおいても，IoT 機器を除いた，PC/アクセスポイントとサーバで処理を行い，また，ブロックチェーンに対するトランザクション要求は，PC/アクセスポイントから行った．

3.　フレームワークの実装

3.1　概要

この章では，本研究のフレームワークの実装の詳細について説明する．本研究における開発と動作の環境には，PC(macOS Sierra 10.12.3) 上で Python 3.6.3 を用いた．また，Python 用の軽量なウェブアプリケーションフレームワークである flask(version 0.12.2) と Python 用の HTTP リクエストを行うライブラリである requests(version 2.18.4) を使用した．本研究におけるフレームワークの開発成果物は OSS として公開した．[*1]

3.2　ブロックチェーン機能の実装

ブロックチェーン機能の開発に関しては，Python によるブロックチェーンの開発を行った OSS[*2]を参考にした．ブロックチェーンは，ブロックという不変なレコードが，ハッシュ値を使って連鎖したものである．これを用いて，複数のサーバや性能の高い PC/アクセスポイントをノードとして，アップデートの記録を保存・活用する [6]．

起動しているノードに対して，マイニング，トランザクション，ブロックチェーンの記録確認，ノードの追加，コンセンサスという 5 種類の HTTP/HTTPS による要求を行うことで，それぞれの機能が実行される．トランザクション要求に応じて新たなトランザクションが作成される．マイニング要求によりそれらのトランザクションを含む新しいブロックがブロックチェーンに追加される．また，コンセンサス要求によりノード間での分散化されたブロックチェーンの矛盾を解決する．

本フレームワークにおけるブロックチェーンのブロックのデータ構造は表 2 の通りである．本フレームワークは先行研究と異なり，ブロックチェーンノードが全ての IoT 機器ではなく，開発者側のサーバと高性能な PC/アクセス

[*1] https://github.com/ertlnagoya/blockchain_IoT_update
[*2] https://github.com/dvf/blockchain

表 2　ブロックのデータ構造

	詳細
index	インデックス
message	レスポンスメッセージ
time stamp	タイムスタンプ（UNIX 時間）
previous hash	前のブロックのハッシュ値
proof	マイニングの結果となる PoW (Proof of Work)
transaction	トランザクションのリスト

表 3　ブロック内のトランザクションのデータ構造

	詳細
state	機器の稼働状態
recipient	受信者
sender	送信者
digital signature	デジタル署名
ver	アップデートファイルのバージョン情報
verifier	アップデートファイルのハッシュ値

ポイントで稼働していることを想定している．特定の用途のために作られた組込みシステムの限られた性能によっては，ブロックチェーンノードのリソース消費が望ましくない場合があるためである．本フレームワークでは，ブロックチェーンのトランザクションやマイニングに関する要求は，PC/アクセスポイントから行い，ノードの追加やコンセンサスに関する要求は，管理サーバから行った．

3.3　プロシージャ機能の実装

本節では，本研究のフレームワークにおける実装について説明する．本フレームワークの機能に関する全体図が，前述した図 2 になる．本実装では，IoT 機器からリクエストを行うクライアント機能，PC/アクセスポイントでレスポンスを行うサーバ機能，PC/アクセスポイントからリクエストを行うクライアント機能，管理サーバでレスポンスを行うサーバ機能，ブロックチェーンノードのサーバ機能の 5 つに分けてプログラムを実装した．

PC/アクセスポイントのクライアント機能と管理サーバのサーバ機能において，TCP 通信によりそれぞれのアップデートファイルを比較し，バージョンやファイルのハッシュ値といった情報のチェックを行った．その比較結果が異なる場合，クライアント機能が Github からアップデートファイルの取得を行い，その記録のために，ブロックチェーンノードのサーバ機能へ HTTPS リクエストを行い，トランザクションを追加する．そして，IoT 機器のクライアント機能と PC/アクセスポイントのサーバ機能において，同様にアップデートファイルを比較し，結果が異なる場合，アップデートファイルの受け渡しを行い，アップデートを実行する．通信においては，先行研究のフレームワーク同様に，RSA による暗号化やリプライ攻撃対策の乱数の含有を行った．

本フレームワークにおけるブロックチェーンのデータ構造におけるトランザクション内のデータ構造は表 3 のようになっている．ここでは送信者やハッシュ値といった情報を記録し，これらを検索することでアップデートが行われた機器とそのバージョンを把握できるようにした．

4.　おわりに

本研究では，先行研究を参考に，ブロックチェーン技術を活用した IoT 機器向けのセキュアアップデートフレームワークの検討と開発を行った．本フレームワークに対する脅威とその対策の考察・評価を行い，セキュリティの強化や機能の充実を目指すことが今後の課題といえる．

機能の充実として，例えば，本フレームワークにおけるバージョン確認の通信はサーバクライアント方式であるが，これをブロックチェーンに対して通知の記録と参照をするように替えることで，より高い耐改ざん性が得られる．また，ブロックチェーンの情報を元に機器のアップデート状況を可視化して把握しやすくすることや，機器の稼働状況をブロックチェーンに書き込み同様にその情報も把握できるようにすることが考えられる．

そして，そのブロックチェーンの記録を介したバージョンチェックのプロシージャにおけるセキュリティの考察や，先行研究のフレームワークや，ブロックチェーンではなく分散型データベースを用いたフレームワークとの定性的な比較と評価を行う予定である．

謝辞　本研究の一部は JSPS 科研費 16K21097 の助成を受けて行われた．

参考文献

[1] ANTONAKAKIS, Manos, et al. Understanding the mirai botnet. In: USENIX Security Symposium. 2017.
[2] DULAUNOY, Alexandre, et al. An extended analysis of an IoT malware from a blackhole network.
[3] NAGARA, Keigo, et al. Portable DoS Test Tool for IoT Devices. In: Proceedings of the 2017 Workshop on Internet of Things Security and Privacy. ACM, 2017. p. 57-58.
[4] HOLMBACKA, Simon, et al. Lightweight Framework for Runtime Updating of C-Based Software in Embedded Systems. In: HotSWUp. 2013.
[5] KUPPUSAMY, Trishank Karthik, et al. Uptane: Securing Software Updates for Automobiles. The 14th escar Europe (escar EU 2016). Munich, Germany, 2016.
[6] LEE, Boohyung; LEE, Jong-Hyouk. Blockchain-based secure firmware update for embedded devices in an Internet of Things environment. The Journal of Supercomputing, 2017, 73.3: 1152-1167.
[7] Andreas M.Antonppoulos: Mastering Bitcoin: Programming the Open Blockchain. O'Reilly. 2017.
[8] BOUDGUIGA, Aymen, et al. Towards Better Availability and Accountability for IoT Updates by means of a Blockchain. In: Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on. IEEE, 2017. p. 50-58.