Vol.2017-DPS-170 No.5
Vol.2017-CSEC-76 No.5
2017/3/2

スマホ認証を用いたIoT機器サービスの簡易利用方式

矢崎孝一[†1]　伊藤栄信[†1]　坂本拓也[†1]　二村和明[†1]

概要：さまざまな機器が生活をとりまくIoT時代において，従来型のパスワード方式では，利用者が多数のパスワードを管理・記憶することになり利便性が損なわれてしまう．そこで，パスワードを用いることなく，様々なIoT機器が提供するサービスの利用者認証を、利用者が普段利用する生体認証機能付きスマートフォンを用いて簡易化する手法を提案する．スマートフォンを用いたIoT機器の利用者認証では，接続するためのAPIをクラウドから攻撃され，IoT機器そのものが不正に操作されて周囲に被害を与えたり，IoT機器が蓄積するデータの漏洩を引き起こしたりする危険性がある．提案手法では，IoT機器とスマートフォンを近接無線で接続し，近年端末に導入された高いセキュリティ耐性と本人確認機能を持つFIDO機能を呼び出して利用者確認を行うことで、IoT機器の不正遠隔操作や利用者のなりすましを防止ができることを示す．

キーワード：スマートフォン，IoT機器，FIDO，パスワードレス，クラウドサービス

Easy to use method for cloud services through IoT devices using smartphone authentication

KOICHI YASAKI[†1]　HIDENOBU ITO[†1]　TAKUYA SAKAMOTO[†1]　KAZUAKI NIMURA[†1]

Abstract: In the IoT era, various devices are surrounding our daily lives, therefore conventional Password method would be obsoleted because of the horrible user experiences that would be needed to dealing with many Password that does not user could handle. Then in this paper, we propose new user authentication method for cloud services through IoT devices that take advantage of smartphone authentication. This solve the usability of the IoT device use however, it remains security consideration for IoT devices that would connect to server through the Internet may attacked then the IoT device would manipulated fraudulently and affect around it and data leakage from the IoT device might happen. To overcome the security issues we will introduce FIDO protocol which can provide high security tolerance and user verification capability and show the possibility of the prevention from the manipulation or spoofing.

Keywords: Smartphone, IoT device, FIDO, Password less, Cloud service

1. はじめに

2020年には，数百億個のIoT[a]機器がクラウドに接続され生活や産業を大きく変革するといわれている．様々な機器がつながることにより，新たなサービスが展開されることになる．例えば，クルマや宅配ボックス[1]などのシェアリングサービスのようなビジネスの拡大が期待されている．

IoT機器ごとに利用者認証が増えていくと，従来型のパスワードによる方法では，利用者が多数のパスワードを管理することになり利便性が悪い[2]．そこで，パスワードにかわる利用者認証技術として，指紋認証や手のひら認証などの生体認証をIoT機器に取り付け，サービス利用することが考えられる．しかし，利用するIoT機器が増える度に利用者がひとつひとつIoT機器に対して登録作業を行う必要があり利便性の観点から現実的とは言えない．このためクラウドに生体情報を登録して利用者認証する手法が、ATMサービスなどで用いられている．この方式では，クラウドでの個人情報に対する保護策が必要となり，安価なサービスを提供するIoT機器には向かず裾野が広がらない．

そこで本論文では，スマートフォン(以下，スマホと略す)をあらゆるIoT機器の鍵として利用することで，IoT機器の利用者認証を簡易化する手法を提案する．これにより利用者は，普段利用するスマホひとつを用いるだけで，IoT機器の違いに煩わされることなく，サービスを利用することが可能になる．一方で，スマホとIoT機器を連携させることにより，この間に新たなセキュリティ懸念が生じることになる．具体的には，スマホとの連携用APIを遠隔から不正に操作されてIoT機器が周囲に被害を与えるリスク，あるいはIoT機器が蓄積するデータの漏洩を引き起こすリスクをもっている．これに対し提案手法では，近年、端末に導入されている高いセキュリティ耐性と本人確認機能を持つFIDO(Fast IDentity Online) [3]機能を活用して，利用者の確認とIoT機器の前にいることを一括で確認することで，IoT機器の不正遠隔操作や利用者のなりすましを防止できるようにする．

また，本論文では，理論評価および提案手法の実装と動作評価を行うことで，提案手法が実用レベルであることを示す．

†1 (株)富士通研究所
Fujitsu Laboratories Ltd.

a) Internet of Things

Vol.2017-DPS-170 No.5
Vol.2017-CSEC-76 No.5
2017/3/2

2. 課題と提案手法

ここでは，本論文が対象とする課題をまとめ，解決に向けた要件および提案手法について述べる．

2.1 課題

IoT 機器における利用者認証の安全性と利便性には以下のような課題がある．

(a) 様々な IoT 機器においてパスワード，トークン，生体情報など異なる認証方式が用いられており，利用者がそれぞれの認証方法を管理する必要があり、利便性が悪い．(ユーザの利便性)

(b) IoT 機器あるいはクラウドにおいて、ユーザのプライバシー情報、たとえば生体認証の場合には生体情報が漏洩しないようにする対策コストがかかる．このため，安価な機器では費用対効果でみた場合に情報漏えい対策コストが見合わない(サービス構築の簡易性)

(c) IoT 機器が利用者以外により遠隔から不正に操作される．(IoT 機器前の本人確認性)

2.2 要件

課題解決に向けた要件は以下のようになると考える．

(a) ユーザの利便性： さまざまな IoT 機器が存在する中で，ユーザがそれぞれの認証方法を管理しなくても IoT 機器のサービスを受けられること．

(b) サービス構築の簡易性： IoT 機器およびクラウドサービスにおいて、生体情報などのプライバシー情報を管理する必要がないこと．

(c) IoT 機器前の本人確認性： IoT 機器前にいる本人のみが API アクセスできること．

2.3 提案手法

ここでは課題解決のため，IoT 機器の利用者認証に，スマホの認証機能を用いる手法を提案する（図 1）．提案手法がそれぞれの要件を満たすことを示すと共に、その動作に関しても述べる．

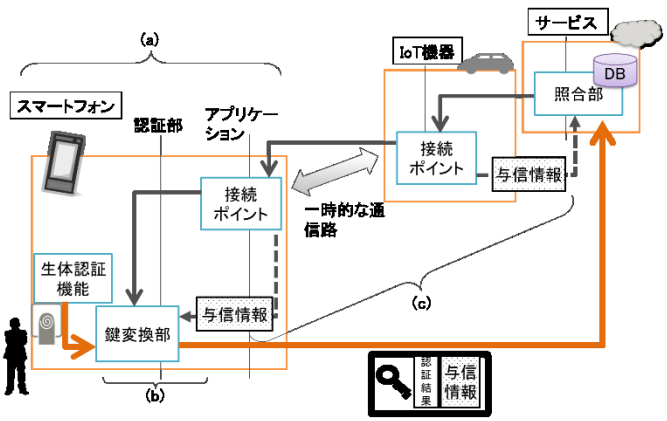

図 1　提案手法の概念図

Figure 1　Conceptual diagram of proposed method

(a)　ユーザの利便性：

図 1 の(a)に示しているように、利用者が普段利用しているスマホの生体認証機能を IoT 機器の利用者認証に用いることで、IoT 機器への生体認証機能の組み込みを不要とすると共に、利用者が IoT 機器ごとの認証方式の違いに煩わされることなく認証が行えるようにする．

(b)　サービス構築の簡易性：

図 1 の(b)に示しているように、生体認証による本人認証結果をスマホの中で公開鍵手法を用いた署名に変換する鍵変換部を持つことで，IoT 機器およびサービス側において生体情報を扱う必要がなくなり生体情報に対する管理コストを不要にする．

(c)　IoT 機器前の本人確認性：

図 1 の(c)に示しているようにサービスから本人に至るまでの通信路に関して、接続ポイントから各通信路の接続相手を与信情報としてサービス側に吸い上げ、分析照合する照合部を持つことで経路変化を検知する．これにより IoT 機器前にいるスマホ/本人のみからのアクセスを特定する．

2.3.1 登録

動作の前提として，スマホの鍵変換部が生成する鍵をサービスに登録しているものとする．

2.3.2 動作

提案手法による IoT 機器利用時の認証動作は以下のようになる．

- スマホと IoT 機器の間に、一時的な通信路を生成する．
- 生成された通信路を用いて、サービスからスマホへ利用者認証要求パケットを送出．
- スマホで生体認証をおこない、与信情報を追加したデータに署名．サービスはその署名データから IoT 機器前にいるスマホという経路と、そのスマホ上の本人を検証する．

2.3.3 効果

提案手法により，IoT 機器やサービスに生体情報を登録することなく，利用者が普段利用するスマホを用いて，様々な IoT 機器を一元的な認証で利用することが可能となる．また，認証の際に IoT 機器前にいることも測定しているため，IoT 機器が遠隔から不正に操作されることのない安心安全な環境でユーザは IoT 機器を利用することが可能になる．

2.3.4 ユースケース

提案手法のユースケースには，物理的なロックを伴う入退室管理や，シェアードサービスでの利用が考えられる．入退室管理では，託児所，訪問介護・家事サービス，セキュリティルームへの入室が考えられる．スマホを鍵にすることで，さまざまな鍵の解錠ができる．これにより，シェアードカー，民泊などのサービスにおいて，物理的な鍵の受け渡しをする必要がなくなる．

またユーザ同意の下で個人の特定もサービスから行うことが可能なので，シェアードカーのカーナビやコックピッ

トなどのパーソナライズ，運転履歴に応じた保険の提供などへの展開もできる．このように，IoT 機器の利用者認証が必要となる様々なサービスについて，利用者個人のスマホを用いることにより，安心安全な利用者認証が実現できるようになる．

3. 実装

提案手法の実装構成は，スマホ標準の OS 機能と通信ハードウェア，IoT 機器においても標準的な通信ハードウェアとソフトウェアのみで実装することで実現コストを最小に抑えた実用レベルの実装を追求した．

近接通信方式については NFC，Bluetooth を用い，認証については，国際標準化団体である FIDO アライアンスによって策定されている機能を利用した．以下では，この FIDO の特徴についてまとめるとともに，この機能を利用した提案手法の実装について述べる．

3.1 FIDO

FIDO アライアンスはパスワード不要のオンライン認証技術を策定する団体で，Google, DoCoMo, VISA, ARM, Bank of America など 250 を超える組織が加入している．FIDO が定める仕様には，UAF，U2F がある．UAF は，「Universal Authentication Framework」の略で，生体認証もふくめさまざまなローカル認証をサービスから利用するためのプロトコルである．U2F は，「Universal Second Factor」の略で，主に ID/パスワードといった第 1 の認証に加えて，第 2 の認証トークンを所持しているかをサービスから確認するためのプロトコルである．

UAF では，スマホのセキュア空間で生体認証を行い，その認証結果を公開鍵暗号手法に変換してサービスに送付する．スマホからは公開鍵手法による署名だけが送付されるため，サービス側は公開鍵だけをもてばよく，従来のように生体情報・パスワードなどの個人認証情報の管理が不要な方式となっている．

UAF は以下 4 つのモジュールから構成され，利用するときは UAF をサービスに登録後に利用可能となる．

- Authenticator(認証器) : 指紋認証などユーザを認証する方式毎に用意されている．このモジュールはユーザの生体情報を扱うため，スマホのセキュアな空間で動作し、認証結果を署名に変換する役割を担う．このモジュールは ASM からの呼び出し I/F だけを持つ．
- ASM (Authenticator Specific Module) : セキュアな空間に実装された Authenticator をアプリケーションから利用可能にするためのブリッジ機能を提供し，アプリケーションから Authenticator を呼び出すための公開インターフェースを提供している．
- FIDO Client : UAF で規定されているプロトコル(UAF パケット)を解釈し，Authenticator を利用しようとするアプリケーションが Authenticator を制御するための UAF パケット生成を助ける役割を担う．SDK としてアプリケーションの中に含まれる場合もあれば，独立ソフトウェアとして存在する実装形態もある．
- FIDO Server : UAF パケットを解釈し，署名データの検証と、その結果からユーザ ID に変換しサービスに提供する役割を担う．

3.1.1 登録

UAF を利用する場合には, UAF が生成した公開鍵をサービスに登録する必要がある．その公開鍵には、通常端末ベンダによる署名[b]が添付されているので，サービスはそれを検証して問題なければデータベースに保存する．利用時には，その公開鍵を用いて認証結果を確認し、本人性を確認することができる．

3.1.2 動作

登録後に、UAF が利用可能となる．図 2 は、サービスにログインするときの動作を示したものである．

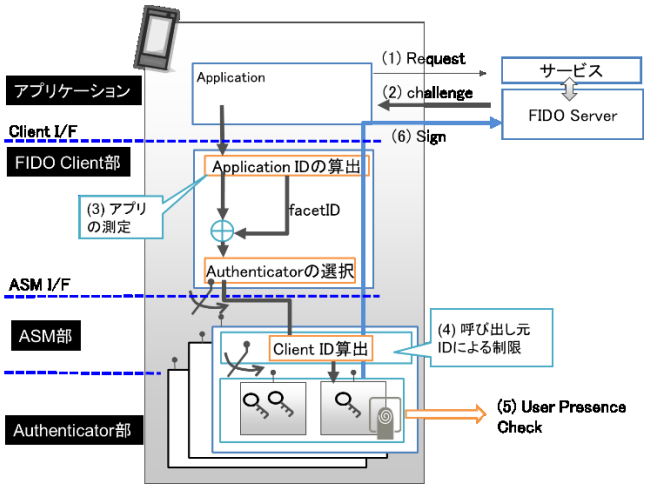

図 2 FIDO UAF プロトコルの処理

Figure 2 Operation Flow of FIDO UAF protocol.

(1) 認証依頼(アプリケーション) : まずスマホ上のアプリケーションがサービスログイン要求をサービスに対して発行する．

(2) 認証依頼(FIDO Server) : サービスは FIDO Server と連携して利用者認証のプロトコルを開始する．FIDO Server からは FIDO alliance で決められた UAF パケットが発行される．サービスから UAF パケットを受け取ると，アプリケーションはその処理依頼を FIDO Client に処理依頼する．

(3) 認証依頼(FIDO Client) : 処理依頼を受けると，依頼元アプリケーションを測定して facetID[4]を算出し、UAF パケットに追加．その後、UAF パケットに記載されている Policy[5]を解釈しサービスが期待する ASM、Authenticator を選択して処理を依頼する．

(4) 認証依頼(ASM) : 依頼を受けた ASM は，呼び出し元

b) その秘密鍵が端末内でどのように保護されているか、そのパラメータを入れてベンダの秘密鍵(Attestation Key と呼ばれる)で署名したもの

Client ID[c]の測定を行う．この Client ID と UAF パケットの内容をもとに、セキュア空間にある Authenticator を駆動して処理依頼を行う．

(5) 認証(Authenticator):処理依頼をうけた Authenticator は、ユーザを確認し，公開鍵手法によって署名データを生成し、結果を返す ．

(6) 認証確認(FIDO server)： FIDO Server は，署名データに含まれる KeyID [d]から、データベースに登録されている公開鍵をロードして署名を検証，署名データに含まれる情報からユーザの本人性を確認し，サービスの提供を開始する．

3.1.3 FIDO のセキュリティ

UAF パケットには、リプレイ攻撃を防ぐ仕掛けが組み込まれている．その１つがモジュールを通過する毎にパケットに値が追加されていく仕様である．最初は FIDO Server が設定するチャレンジデータ[6]、さらに client が測定した FacetID，SSL セッション情報(サーバ証明書，時刻など)}、そして Authenticator が生成する{ユーザの確認結果}，および鍵を特定するための{鍵 ID}，そして{NONCE}が加算されている．NONCE は Authenticator が加える乱数で、FIDO Server と Authenticator 両方で UAF パケットに乱数を入れることで、リプレイ攻撃耐性を強くしている．

また、それぞれのモジュールが追加した情報は署名データの中に入り、FIDO Server で照合される．チャレンジデータとの比較からサービスが使用しているどのセッションに紐付けられたものかを検証することができ、FacetID からは不正なアプリケーションによる認証要求でないことを確認することができる．

Client 部では、さらに FacetID によるアクセス制限も行っている．アクセス可否の情報は UAF パケット内の AppID パラメータに入っており、その情報を Client 部が解釈して実施する．この仕組みは、悪意あるアプリケーションが FIDO Client 経由で ASM & Authenticator にアクセスし，ユーザがどんなサービスを登録しているのか見られないようにするためのプライバシー保護策となっている．

3.2 提案方式の実装

ここでは上述の FIDO をベースに，2.3 節の提案手法(a)ユーザの利便性，(b) サービス構築の簡易性，(c) IoT 機器前の本人確認性，それぞれの具体的な実装方法について述べる．また登録および動作についても記述する．

(a) ユーザの利便性

FIDO 機能を持つスマホと IoT 機器を接続し、IoT 機器の利用者認証を FIDO 機能で代用することで、ユーザはパスワードなどを管理する必要がなくなる．これにより IoT 機器を認証するときの利便性問題が解決される．その一方で IoT 機器からスマホに入っている FIDO 機能を呼び出すために毎回接続する必要があり、その手間がユーザの利便性を悪くしている．そこで，図 3 に示したようにサービスに 1 回登録すれば，あとはその登録された情報を複数の IoT 機器から呼べるようにするための自動接続(自動ペアリング)機構を導入した．これを可能とするために，接続ポイントとして IoT 機器には【送信部】、スマホには【受信部】を実装している．

【送信部】は、IoT を識別する ID(たとえば URL)を発信し、【受信部】は，その ID の正しさを検証して、IoT 機器からの依頼を FIDO 機能に中継する役割を持つ．

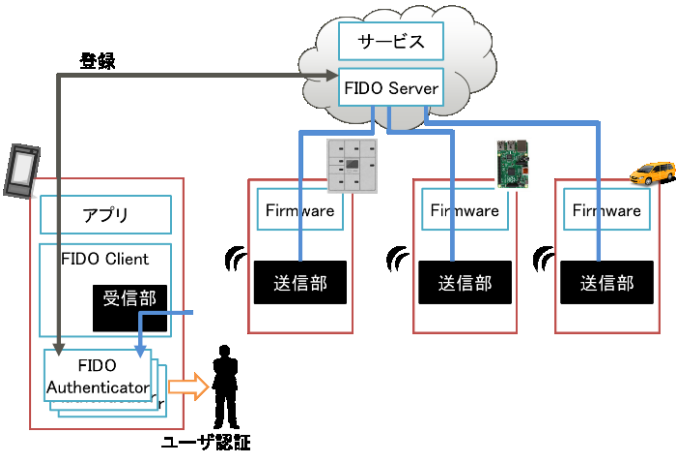

図 3 提案手法の構成

Figure 3 System configuration of proposed method

(b) サービス構築の簡易性：

FIDO 機能の「認証結果を署名データに変換する部分」(Authenticator 部)を図 1 にある「鍵変換部」としてそのまま利用することでサービス構築の簡易化を実現している．

(c) IoT 機器前の本人確認性：

FIDO 機能である「Client ID に基づいて呼び出し元を制限する部分」(ASM 部)はそのまま利用し、Client 部の「アプリケーションを測定する機能」に関しては、その測定対象をアプリケーションではなく IoT 機器に変更し【受信部】として利用している．

受信部から IoT 機器との接続を測定する機能は新規に実装する必要がある．また測定対象である IoT 機器が同じ端末内にはないため Client 部が実装しているような OS 機能を用いた手法は使えない．そのため【受信部】と IoT 機器双方から相手を測定し、その測定データを与信情報としてサービスに報告し、照合することで通信路の正しさを検証できるようにしている．具体的には、【送信部】【受信部】がスマホと IoT 機器の接続を担当し、通信確立に用いた情報(NFC タッチした時刻、BLE 通信時の MAC アドレス、セッション ID/鍵など)を逐次測定し、送信・受信部でないと確認できない情報を用い、その情報をハッシュ化して通信

c) Client ID は，Android 実装においてはソフトウェア署名者の公開鍵 ID が用いられている．

d) https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html#keyid-typedef

路 ID としている．

3.2.1 登録

提案方式を、FIDO を用いて実装したのが図 4 である．

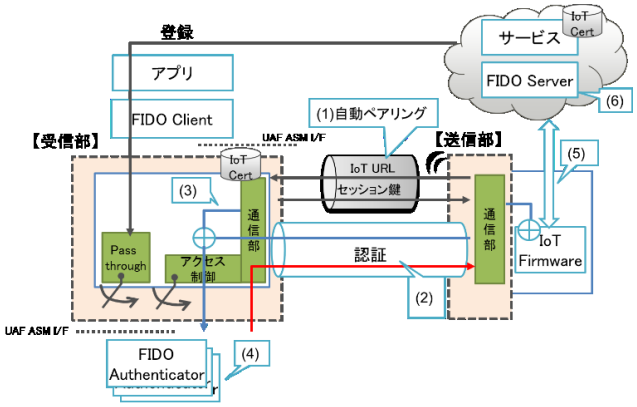

図 4　自動ペアリングによる FIDO 機能の呼び出し

Figure 4　FIDO function using pairing free method

どのスマホと IoT 機器を自動ペアリングするかは、近接接続を用いて決定している．具体的には、スマホが近接によって IoT 機器を感知するところからはじまる．スマホは、近接通信を用いて IoT から URL を取得し、その正しさをサービスから取得した URL の証明書(IoT Cert)リストで検証し、自動ペアリング可能な IoT 機器かどうかを判断している．近接接続の代表的なデバイスは NFC であるが、小さな IoT 機器では NFC を搭載していないものも少なくない．そういった場合も想定し、NFC 以外に BLE モードでの近接検知も実装している．BLE モードでの自動ペアリングフローを示したのが図 5 である．

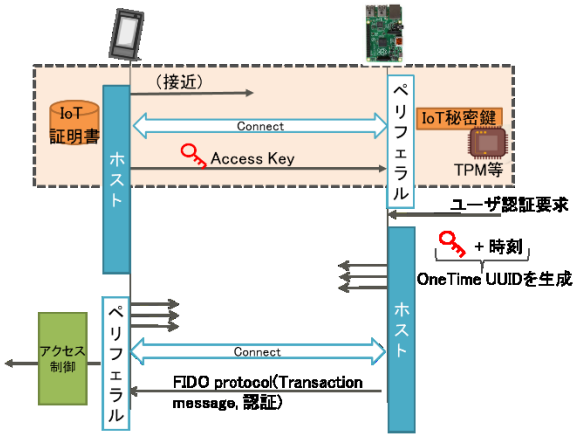

図 5　BLE 方式での自動ペアリング

Figure 5　Pairing free using BLE method

BLE では、機能を提供するペリフェラルがビーコンを発する．そのため FIDO 認証機能を提供するスマートフォンがビーコンを発行する必要がある．しかしそれではスマートフォンのバッテリ消費が大きいため、まずは IoT 機器がビーコンを発行し、その電波レベルでスマートフォンが近接にある IoT 機器を認識し、セッション鍵（Access Key）を交換した後に、それぞれの動作モードを交換して一時接続を行う実装を行っている．すなわち近接検出時は、IoT 機器がペリフェラル、利用時にはスマートフォンがペリフェラルになるよう実装することで、バッテリ消費を抑えている

また【受信部】は、自動ペアリング以外に重要な役割を果たしている．それは他のアプリケーションがサービスに登録した認証器を、複数の IoT 機器から利用するための機能である．具体的には、スマホ上から Authenticator を利用するときは UAF パケットをそのまま転送(Pass Through)し，IoT 機器から認証要求がきたときは，適切な Authenticator に転送する役割を担う．これにより Authenticator からみて同じ ID を持つソフトウェアが認証器にアクセスしていることになり、複数のアプリケーションからの認証器共有が可能となる．

3.2.2 動作

提案手法の動作は以下になる．主に FIDO との違いを中心にまとめる．

(1) 認証依頼(通信部)：利用するときは，スマホを IoT 機器に近接させる．その近接通信を通して IoT 機器から URL を取得して検証する．その後自動ペアリングが行われ、適切な Authenticator を呼び出すための API が有効になる．

(2) 認証依頼(IoT Firmware)： スマホ側に UAF パケット処理依頼を転送する．

(3) 認証(受信部)：Client 相当．UAF パケット内の Policy を解釈し，適切な Authenticator に処理を依頼するとともに、通信部が測定した情報をパケットに追加する．

(4) 認証(ASM & Authenticator)：変更なし

(5) 認証結果送信(IoT Firmware)：IoT 通信部が測定した経路情報を追加して、サービスに応答を返す．

(6) 認証確認(FIDO server)：署名データを照合して【IoT 機器前の本人確認性】を検証

4. 評価

以下では，提案手法の理論評価と動作評価を行った結果を示す．理論評価では提案した経路保証が攻撃に対する耐性をもっているかを見るために、経路に対する攻撃を検討し、その耐性を検証する．動作評価では，ドアロックを例にして基本動作確認により要件を満たしていることを評価する．

4.1 理論評価

ここでは，提案手法の脅威分析による理論評価を行う．攻撃箇所をまとめたものが図 6 であり、それぞれの対策内容をリストアップしたのが表 1 である．攻撃箇所は，サービスと IoT 機器間(①)、IoT 機器上のマルウェア(②)、IoT 機器とスマホ間(③)，スマホ上マルウェア(④)、IoT 機器の偽造(⑤)に分類できる．

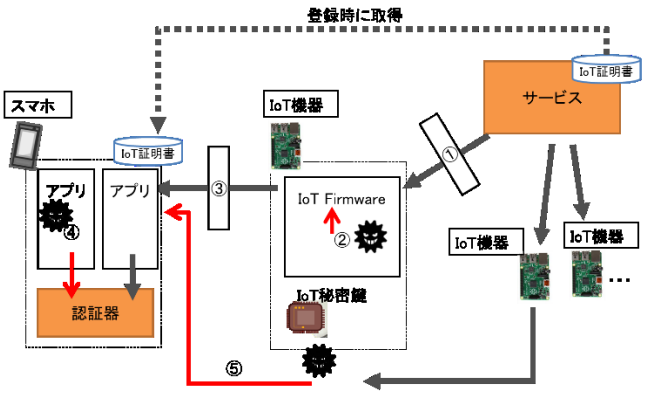

図 6　攻撃に関する検討

Figure 6　Attack point in this System

表 1　検討結果

Table 1　Countermeasure analysis for attack

攻撃箇所	内容	対策
①	中間者攻撃	サービスが持つ IoT 証明書で検出
②	不正アクセス	IoT 秘密鍵へのアクセスを適切なソフトウェアのみに許可するよう IoT 機器を構成する(IoT 秘密鍵保護に TPM[7])を利用)
③	中間者攻撃	スマホが持つ IoT 証明書で検出
④	不正アクセス	アプリ ID を識別し、認証器へのアクセスをブロック
⑤	フィッシング	操作される IoT 機器と、スマホ両方から与信情報を集めサービス照合で検出

①は、IoT 機器が使用している WAN 接続に介入し、通信内容の改ざんを行うものである．IoT 機器がサービスに接続する際に、お互いの証明書を交換しセッション鍵を交換するときに介入する．そのため IoT 機器が電源断するなどサービスとセッションを張り直すタイミングが狙われる．その一方で③は IoT 機器とスマホ間の一時的な接続に介入する．この接続は、ユーザが入れ替わるたびに張り直されるので介入されるタイミングは①よりも多い．介入する際は、偽の証明書を提示して通信に介入するため、提案方式では IoT 機器の証明書をサービスとスマホで持ち，IoT 機器と通信路を生成するときに、この証明書と照らし合わせることで、攻撃に対する耐性を持たせている．

一方、Firmware Update や、ユーザによる誤操作で IoT 機器/スマホにマルウェアが入ることも想定される．それが②④である．IoT 機器に入ったマルウェアは、サービス・スマホへの不正アクセスを行うことが考えられるが、アクセスするには秘密鍵による署名が必要なため、IoT 機器の秘密鍵へのアクセスをブロックすることで、不正アクセスへの攻撃耐性を持たせている．スマホ上のマルウェアに関しても、FIDO 準拠の認証器がアクセスしてくるマルウェアを測定しアクセスをブロックしており、その機能をそのまま利用して攻撃に対する耐性をもたせている．

また⑤は、他の IoT 機器をエミュレーションしユーザをだます手法で、IoT 機器前の本人確認性を攻撃しているものである．提案手法にあるようにスマホからの与信情報（近接時の時刻、一時的な通信で使用している MAC アドレス）と、IoT 機器からの与信情報を比較し偽造 IoT によるエミュレーションのタイムラグ、物理アドレスの不一致をサービスで検出しブロックすることで、攻撃に対する耐性を持たせている．

4.2 動作評価

ドアロックシステムに関して、提案手法を図 7 のように実装し動作することを確認した．

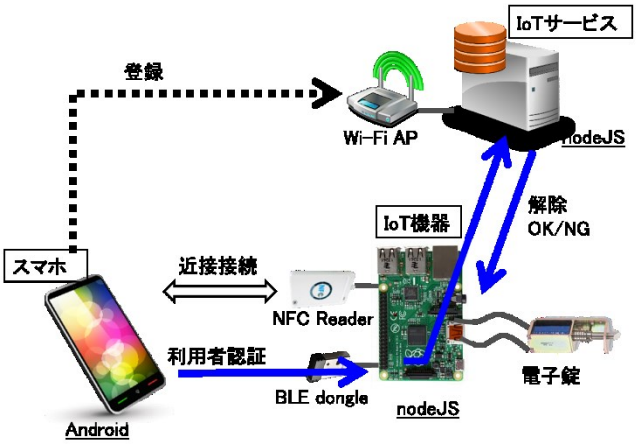

図 7　評価環境

Figure 7　System configuration for evaluation .

ユーザ認証を行うスマホとして FIDO 機能搭載 Android スマホ (認証タイプ : 指紋, OS : 6.0)、電子錠を制御する IoT 機器として Raspberry Pi (NodeJS で GPIO 制御)、IoT サービスの実装にはノート PC (NodeJS で Web サービス実装)を用いた．スマホと IoT 機器は、WiFi アクセスポイントにつながっており、スマホの FIDO 機能をサービスに登録するときや、IoT 機器がサービスにユーザ確認を行うときに用いる．図 8 が近接接続時の動作で、図 9 が利用者認証時の動作を示している．

評価環境では、近接接続の方法として NFC 方式を用いている．スマホを IoT 機器が持つ NFC Reader にタッチすることで近接かどうかを判断し、自動ペアリングをしている．その様子が図 8 であり、スマホを近接(①)してから IoT 機器の検出(②)がおこなわれ、その後通信路生成 (②→③)が行われる．接続時間を計測すると①～③までにかかる時間は約 2 秒であった．実際の利用者視点では、スマホを IoT 機器にタッチして手元にもってくるまでに接続は完了しており、遅れなどの違和感は無い．

近接接続がおわると、利用者認証の動作 (図 9)が始まる．スマホに搭載されている FIDO 機能を IoT サービスから呼

び出し、利用者認証を行う．UAF 仕様では認証時に表示するメッセージをサービスからコントロールすることもでき、評価環境ではユーザによる確認を促すために「You are about to Open 3F office door. Is it OK? 」というメッセージを IoT サービスから発行している．それらを受信したスマホはユーザ認証時にメッセージを表示し、NFC タッチ時刻(2017/2/10/11:17:35)、通信路で使用している BLE アドレス(6C:A7:D1:A5:82:90)を署名データに入れて IoT サービスに送信する．IoT サービスからも接続時のデータ(NFC タッチ時刻、BLE アドレス)が報告されるので、そのデータと照合して問題なければ IoT 機器に利用 OK 命令を発効し、ドアが開く．

図 8　近接接続時の動作

Figure 8　Behavior of detect IoT in the System

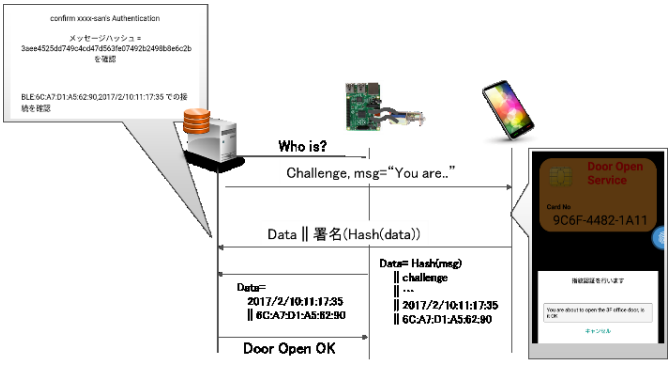

図 9　利用者認証時の動作

Figure 9　Behavior of authentication in the System

5. 関連技術

IoT 機器の利用者認証方法として，パスワード方式，生体認証方式，トークン方式が用いられている．以下では，これらの概要をまとめるとともに，2.2 節の要件をベースに提案手法との違いについて述べる．また，その比較を表 2 にまとめる．また FIDO アライアンスでは，ブラウザから，スマホの認証器を呼ぶ仕組みとして CTAP(Client To Authenticator Protocol) [8]が検討されている．これは IoT 機器向けではないが，これについても概要および提案手法との違いについてまとめる．

5.1 Password 方式

Password 方式は，IoT 機器の利用者認証に用いるパスワードをユーザが管理する[9]．認証が IoT 機器ごとに行われることから，認証情報の保持は，IoT 機器あるいはその機器と連動するサービスで行われることになる．パスワードを入力するユーザインタフェースも IoT 機器ごとに異なるため，ユーザは IoT 機器ごとのパスワードに加え，それを入力する方法も覚えておく必要があり利便性が悪い．

以下では，パスワード方式が要件を満たすかどうか評価をまとめる．

(a) ユーザによるパスワードの管理が必要で，煩わしい．
(b) ユーザのプライバシー情報を IoT 機器/サービスで管理する必要があり、コストは高い．
(c) ユーザが直接 IoT 機器を操作するので機器前の本人性は問題なし．

5.2 生体認証方式

生体認証方式[10]では，IoT 機器ごとに生体センサーが取り付けられ，そのセンサーで IoT 機器を利用するユーザを認証している．ユーザはセンサーによるスキャンだけで利便性よく IoT 機器が利用可能になる．その反面，生体情報はユーザが変えることの出来ない ID でもあるため，その情報が漏れないようにする対策や，IoT 機器へのセンサー設置が必要で，システムコストが高くなってしまう．

以下では，生体認証方式が要件を満たすかどうか評価をまとめる．

(a) 生体情報の提示のみなので利便性がいい
(b) ユーザの生体情報を IoT 機器/サービスで管理する必要があり、コストは高い
(c) ユーザが直接 IoT 機器を操作するので機器前の本人性は問題なし

5.3 トークン方式

トークン方式[11]は、そのトークンを封入するデバイスの接続形態から Bluetooth トークン、スマートカードトークン、USB トークンなどと呼ばれている．トークンはデジタル署名や、アルゴリズムに伴って変化するワンタイムパスワードに分かれる．デジタル署名の場合には、サービス側に公開鍵を登録し、ワンタイムパスワードの場合には,アルゴリズムを登録する必要がある．この登録作業をオフラインで行うのがクレジットカードなどの IC カード方式、一方 OAuth や OpenID connect 基盤で用いられるトークンはオンラインで発行され、Web ログインの連携やサービス間連携で用いられている．

5.3.1 IC カード方式

IC カード方式の代表的なものは Suica で、ショッピングや電車に乗る時にワンタッチで利用者認証を行うことができる．このプロトコルを崩さずにスマホに搭載したのが Apple Pay [12]である．これらは利用者が IC カードを持っ

ているかどうかをチェックするプロトコルになっており、さらに盗難対策を目的として、利用時にユーザ認証が呼び出されるものもある．このユーザ認証の設定はリアルタイムに変更できず､また認証方法を選択することもできない．

以下では，IC パスワード方式が要件を満たすかどうか評価をまとめる．

(a) IC カードをタッチするのみで利便性がいい．ただしユーザ認証をもとめるタイプのものは、盗難対策を主目的としているため利便性が悪い．

(b) 物理的な IC カードを用いる場合は発行するコストがかかるものの、プライバシー情報がないので構築は容易

(c) カードがどんな認証方法で本人確認しているのかサービス側には見えない．

表 2　既存方式との比較

Table 2　comparison with various related study.

要件		(a)*	(b)*	(c)*
パスワード方式		×	×	○
生体認証方式		○	×	○
トークン	IC カード	△	△	×
	オンライン	○	○	×
	提案方式	○	○	○

*(a) : ユーザの利便性，(b) : サービス構築の簡易性，(c) : IoT 機器前の本人確認性

5.3.2 オンライン型トークン

オンラインでトークンを発行する代表的な例は、スマホなどで利用できる電子航空券[13]である．サービス会社の Web サイトにログインしてトークンを発行してもらい、そのトークンを QR コードなどに変換して会場に入場する際に読み取ってもらうことで利用者認証が行われる．Web サイトへのログインに FIDO を用いてユーザ認証を強化している例もある．このトークンを IoT 機器に提示することで、IoT 機器の利用者認証にすることも可能である．ただし、トークンを持っていれば誰でも IoT 機器を利用できてしまい、トークンが第三者の手に渡ってしまうと遠隔から IoT 機器を操作可能となるリスクを持つ．

以下では，直接方式が要件を満たすかどうか評価をまとめる．

(a) スマホをタッチするのみなので利便性がいい

(b) プライバシー情報を管理しないので構築は容易．

(c) トークンが漏洩すると，トークンと本人との紐付けが無意味になり，本人確認性が失われる

5.4 FIDO CTAP

FIDO CTAP は，タブレットや PC などのブラウザから多様な認証器を利用するための共通のインターフェース仕様で，U2F/UAF に対応したスマホやドングルタイプのデバイスをつなぐことができる．CTAP を使うには端末と認証器のペアリング作業が伴うため，端末側にペアリングを行うための UI や接続インターフェースが必要になる．IoT 機器においてはリソースが限られているものが多く、ペアリング作業が不向きである．IoT 機器向けに自動ペアリングで接続できるようにしたのが本方式である．

6. おわりに

IoT 機器の利用者認証において，パスワード方式では，利用する IoT 機器が増えると利用者が多数のパスワードを管理･記憶することになり利便性が損なわれる課題に対し，本論文ではスマホの生体認証機能を用いて IoT 機器の利用者認証を行う手法を提案した．また提案手法の実装では，スマホの FIDO プロトコルおよび接続経路の測定を行うことで不正遠隔操作や誤操作のリスクを排除し、IoT 機器前にいる本人確認性をサービスから確認できることを検証した．提案手法により，利用者が普段利用するスマホを用いることで，パスワードの管理や，プライバシー漏洩を気にすることなく，利便性を確保した状態で安心安全な環境で IoT 機器を利用することが可能となることを示した．

参考文献

[1] 「宅配受取ロッカー」，いよいよ JR の首都圏 100 駅に設置スタート, http://www.rbbtoday.com/article/2016/05/10/141886.html(2017 年 2 月確認)
[2] “パスワード地獄”からの解放なるか スマホ普及で指紋認証が再び脚光, http://itnp.net/article/2014/06/17/804.html (2017 年 2 月確認)
[3] FIDO alliance, https://fidoalliance.org/ (2017 年 2 月確認)
[4] FIDO AppID and Facet Specification v1.0, https://fidoalliance.org/specs/fido-u2f-v1.0-ps-20141009/fido-appid-and-facets-ps-20141009.html (2017 年 2 月確認)
[5] FIDO UAF Policy, https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html#processing-rules-for-the-server-policy (2017 年 2 月確認)
[6] FIDO Security Reference, http://fidoalliance.org/specs/fido-uaf-v1.0-rd-20140209/fido-security-ref-v1.0-rd-20140209.pdf (2017 年 2 月確認)
[7] Trusted Platform Module, https://trustedcomputinggroup.org/ (2017 年 2 月確認)
[8] Trusted Platform Module, https://trustedcomputinggroup.org/ (2017 年 2 月確認)
cs/fido-v2.0-rd-20161004/fido-client-to-authenticator-protocol-v2.0-rd-20161004.html (2017 年 2 月確認)
[9] ネットワークカメラや家庭用ルータ等の IoT 機器は利用前に必ずパスワードの変更, https://www.ipa.go.jp/security/anshin/mgdayori20161125.html (2017 年 2 月確認)
[10] 生体認証(Wikipedia), https://ja.wikipedia.org/wiki/生体認証 (2017 年 2 月確認)
[11] セキュリティトークン, https://ja.wikipedia.org/wiki/セキュリティ (2017 年 2 月確認)
[12] Apple pay, http://www.apple.com/jp/apple-pay/ (2017 年 2 月確認)
[13] 電子航空券, https://ja.wikipedia.org/wiki/電子航空券 (2017 年 2 月確認)