「マルチメディア，分散，協調とモバイル
(DICOMO2021)シンポジウム」 令和3年6月-7月

Capabilityモデルに基づく スマートホームデバイスのネットワークアクセス制御

松本 直樹[1] 小谷 大祐[1] 岡部 寿男[1]

概要：一般家庭のネットワークには、スマートスピーカなどのスマートホームデバイスと呼ばれる多機能なデバイスが接続されつつある。しかし、そういったデバイスが情報の窃取などを目的としたユーザーの意図しない通信を受けた場合、デバイスを保護する手段が存在しない。また、クラウドを経由することなくデバイス同士が直接通信する場合において、全てのデバイスについて適切にアクセス制御を行うことができるとは限らない。そこで、ネットワーク側でポリシ設定を行いデバイスを保護する手法が提案されているが、実際には一般家庭のユーザーがデバイスごとに設定を行い保護することは困難である。本論文では、一般家庭におけるアクセス制御に求められる要件を整理し、ホームネットワークにおける Capability モデルに基づく認可アーキテクチャとアクセス制御手法を提案する。デバイスの各機能を Capability で表現する認可アーキテクチャにより、デバイスの機能ごとにどのような目的をもつ通信であるかをユーザーが確認しつつ認可することを可能にした。ユーザーに認可された Capability に基づき、OpenFlow によるフロー制御とパケットのペイロードに基づく制御を行うことでデバイス間や外部通信についてアクセス制御を実現した。さらに、提案手法に基づくプロトタイプの設計と実装を行い、実デバイスを用いた検証環境で提案手法が動作することを確認した。

Capability Based Network Access Control for Smart Home Devices

Naoki MATSUMOTO[1] Daisuke KOTANI[1] Yasuo OKABE[1]

1. 序論

家庭内ネットワーク（ホームネットワーク）には、パソコンのような従来の情報端末に限らず、スマートフォンやタブレット、スマートスピーカ、さらには施錠装置や空調機器、電球などの家電製品 [14] といったいわゆるスマートホームデバイスが多数接続されつつある。スマートホームデバイスによってユーザーは手元の情報端末や音声で家電を操作でき、さらに、製品の状態や記録を逐一確認するといったことが可能になった。しかし、そういったきわめてプライベートな情報のやり取りが行われる一方、ホームネットワークの特性上、ネットワークに参加している任意の機器間の通信が可能であったり、簡単なセキュリティ機構しか用意されていない場合がほとんどである。そのため、ネットワーク内に一つでもユーザーが意図しない動作をするデバイスが存在した場合、家庭内のあらゆるデバイスが、悪意のある第三者やデバイスベンダによるホームネットワーク内の探索や情報の窃取、機器の不正使用を目的とした、ユーザーの意図しない通信にさらされる。その結果、スマートスピーカや IP カメラからの個人情報の漏洩に限らず、施錠装置や空調機器、医療機器に対する不正な操作によって人命に関わる事態が発生する恐れがあるが、ユーザーにはホームネットワークに接続されたデバイスを保護する手段が存在しない。

また、現在のスマートホームデバイスはクラウド上のシステムと連携することでデバイス間の連携を可能にしているが、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る [21]。デバイス間で直接通信を行う場合、各デバイスでどういったデバイスとの通信を受け入れるかアクセス制御を行う必要がある。しかし、全てのデバイスがアクセス制御に対応しているとは限らず、デバイスの計算能力の制限によって実現できるアクセス制御に制限があったり、デバイスのソ

[1] 京都大学

フトウェア自体の脆弱性によってアクセス制御が機能しない場合が考えられる。また、デバイス間の通信プロトコル自体が認証やアクセス制御を行わない設計となっている場合もある。その結果、前述した問題と同様の事態が発生する可能性がある。

そこで、各デバイスのアクセス制御をユーザーが一つずつ通信が意図したものであるか確認しつつ設定を行う、いわゆるユーザー主体な制御を、インターネットの動的な制御によって行う提案 [1], [16] がなされている。これらの手法では OpenFlow[13] を用いることでデバイスの通信を制御し、アクセス制御を実現している。デバイスの利用意図に応じて設定を行いデバイスを保護することが可能であるが、ネットワークに関する知識が乏しいユーザーが多様なデバイスについて細かい制御を実現するためのポリシを記述をすることは困難である。

アクセス制御を実現するアーキテクチャとしては、Role Based Access Control[18] や Attribute Based Access Control が存在するが、セグメント単位やデバイス全体で統一された属性による制御であり、多様なデバイスが存在するホームネットワークには適していない。それらの問題を解決しデバイスの細かいアクセス制御を行う手法として、Capability モデルに基づくアクセス制御 [4] が提案されている。Capability というデバイスの機能毎に発行されるトークンによって認可を取扱い、Capability を発行すれば認可を行ったこととなるため、デバイスごとに持つ機能が異なるホームネットワークに適したアーキテクチャである。すでに産業用 IoT システムやスマートハウスの IoT 環境を想定した認可アーキテクチャの提案は複数 [5], [19], [22] なされているが、アーキテクチャの提案にとどまり、さらに、実際に適用するには全てのデバイスで手法に沿った実装が必要となり既存の多様なデバイスが存在するホームネットワークでは現実的ではない。多種多様なデバイスに対して画一的にアクセス制御を実現する方法としてはネットワークにおけるアクセス制御があるが、Capability モデルとホームネットワークアクセス制御を組み合わせた提案はなされていない。さらに、 Capability モデルではあらゆる認可についてその都度 Capability を発行する必要があるため、すべてのデバイスに関してユーザーが頻繁に認可判断を行う必要があり、アクセス制御の仕組みとして適切でない [23] という問題もある。

以上の通り、既存の研究では認可アーキテクチャとアクセス制御手法が分けて考えられていることや、一般家庭のユーザーが対象であることを踏まえた議論がなされていないため、実際のホームネットワークにおける利用を考えると不十分であり、ホームネットワークに適したアクセス制御について、一般家庭のユーザーが利用するという観点を含め認可アーキテクチャとアクセス制御手法を合わせて考える必要がある。

本論文では、ホームネットワークに適したアクセス制御として、細かい認可をユーザー主体で容易に実現することができる Capability モデルに基づく認可アーキテクチャと、認可された Capability に基づくホームネットワークのネットワークアクセス制御を実現する手法を提案する。家庭におけるアクセス制御についてユーザーの観点から求められる要件 [11] を整理し、Capability モデルを利用することでそれら要件を満たす認可アーキテクチャを設計した。認可アーキテクチャには、Capability モデルの問題点であるユーザーへの認可判断の要求頻度を削減するために、認可判断をユーザーに代わって自動で行う自動認可が組み込まれている。自動認可を悪用した攻撃やプライバシ上のリスクを考慮し、特定デバイスや特定 Capability に関するポリシ設定をユーザーが記述することで自動認可を制御し、これらのリスクを抑制できるように設計した。

そして、設計した認可アーキテクチャでの認可に基づき、OpenFlow によるフローの制御やパケットのペイロードを解釈し転送の制御を行うことで、ユーザーがデバイスについて認めた通信のみ許可しユーザーが意図しない通信はすべて遮断するアクセス制御を実現した。

本論文の提案を実際のデバイスに対して適用する上で既存のデバイスには対応できない問題については、既存のデバイスに不足する機能を補完し物理デバイスに対応するソフトウェアとして、仮想デバイスを導入することで既存のデバイスに大きな改変を加えることなく本論文の提案を既存のシステムに組み込むことを容易にした。仮想デバイスでは、認可に関する Capability の処理だけでなく、物理デバイスのパケットを仮想デバイス内で解釈し転送を制御することでパケットペイロードに基づく通信の制御も行う。

これらのプロトタイプを作成し実機を用いた検証環境における検証によって、提案手法により Capability モデルに基づくアクセス制御でスマートホームデバイスの通信を、自動認可を含む認可の元で制御できることを確認した。

本論文では、第 2 章で関連する研究、第 3 章で自動認可を含む Capability モデルに基づく認可アーキテクチャを提案し、第 4 章で提案する認可アーキテクチャに基づいたアクセス制御を提案、第 5 章では提案手法の実装と検証、第 6 章では考察、第 7 章で結論をまとめている。

2. 関連研究

2.1 ホームネットワークにおけるアクセス制御

ホームネットワークにおけるネットワークの動的な制御としては a) 外部のシステムに管理を委託する手法 と b) ユーザー主体で設定・管理する手法が存在する。

a) 外部のシステムに管理を委託する手法

外部の専門家やシステムにホームネットワークの管理を委託する手法は、OpenFlow などの SDN 技術と組み合わせたものが提案されている。Feamster ら [3] は、ホームネッ

トワークの管理を外部に委託することで電子メールのスパム検出やボットネットの検出が可能であることを示した。また、Osman ら [17] は、ホームネットワーク内に接続されたデバイスを自動で検出し、クラウド基盤上に構築された解析システムでデバイスを解析し、自動的にネットワークの分離を行うシステムを提案した。

これらの手法は、ネットワークの管理主体をネットワークに関する知識が不足するユーザーから外部の専門家や自動化された仕組みに委譲することを提案している。しかし、これらの手法では各個人のデバイスやネットワークに関する情報を外部に送信するためプライバシの問題を原理上解決することが困難である。また、ネットワークの管理を担う外部システムが利用不可能になった場合、ホームネットワークも利用不可能になる可能性があるという可用性の点で大きなリスクをはらんでいるため、外部への依存を前提とすることは望ましくない。さらに、 Yao ら [23] によると、ユーザーは家庭内のデバイスの管理を外部に委譲することには否定的であり、外部に委譲するシステムはユーザーにとって望ましいものではないとしている。そのため、本論文では外部のシステムには依拠せず、ホームネットワーク内で完結する手法を提案する。

b) ユーザー主体で設定・管理する手法

一方で、ユーザー主体で同様な概念を実現する提案がなされている。Mortier ら [16] はルータと OpenFlow を用いたフローの制御を統合することで、ホームネットワークに接続される機器を個別に管理する手法を提案した。しかし、ユーザーがデバイスごとに手動で設定するのは困難であるため、Bakhshi ら [1] は上流回線の帯域を有効活用するために、デバイスごとに通信の特性を記述したプロファイルを設定し、帯域制限や QoS の制御を行う手法を提案した。さらに、Lear らは、IoT デバイスごとにベンダが事前にアクセス可能な範囲を定義し、自動で適用する Manufacturer Description Usage(MUD)[8] を提案した。デバイス側ではなくネットワーク機器側で動的に定義に従って制御するため、デバイス側に脆弱性があった場合であっても追加の防衛策として機能するとしている。

これらの研究では、フローの制御技術を利用することで多種多様な機器が接続されるホームネットワークの管理コストや上流回線の利用効率を改善する提案を行っている。しかし、ホームネットワークの制御について技術的な可能性を示したが、手動設定や事前定義されたプロファイルを利用しているため、ユーザーがデバイスごとに適切な設定を行い意図したアクセス制御が行われているか確認することが困難である。また、Mazurek らによる家庭内の情報共有に関するアクセス制御におけるガイドライン [11] によると、家庭内のアクセス制御では (1) 細かい制御が可能であること、(2) デバイスの貸与を考慮すること、(3) Reactive Policy Creation[12] が可能であること、(4) ログを含むこと、(5) 事前設定の複雑さを軽減すること、(6) 社会的な規則を考慮すること、(7) ポリシの頻繁な書き換えを考慮すること、(8) ユーザーのメンタルモデルを考慮すること、が求められている。既存のポリシやプロファイルを利用した設定では、(1) については既に述べたように一般ユーザーにとっては困難なことであり、(2)、(3)、(5)、(7) についてもデバイス毎にポリシを手動で設定する仕組みでは頻繁に変化する利用状況に追従しつつアクセス制御を実現することが困難である。このように、既存の研究では一般ユーザーが利用するという観点からの議論が不十分であり、本論文では一般ユーザーが利用する上で必要となる要件を整理し、要件に沿った認可アーキテクチャを提案する。

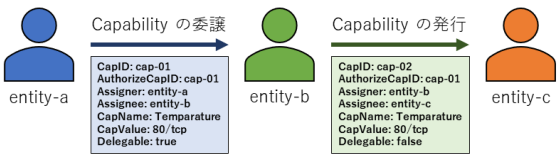

図 1 Capability の委譲

2.2 Capability Based Access Control

Capability モデル [9] は古くから提唱されたアクセス制御モデルであり、デバイスが利用者に対して Capability を発行し、利用者はデバイスに対して Capability を提示することで該当機能を利用できるようになるモデルである。Gusmeroli ら [4] は、Capability モデルに基づいたアクセス制御を提案した。従来の Access Control List や Role Based Access Control[18], Attribute Based Access Control では、IoT デバイスなどの機器の機能に合わせて設定を定義する場合、ロール爆発や属性の一貫性の担保が困難といった問題があった。Capability モデルを採用することでこれらの問題を解消し、デバイスの機能ごとにアクセス制御を行うことを可能にした。さらに、Capability を図 1 にあるように他者に委譲することによって、認可判断や Capability の発行をデバイス以外でも行える仕組みを実現している。

この研究では、 Capability モデルに基づく認可アーキテクチャによってアクセス制御を行う提案をしているが、Capability モデルを利用してネットワークのアクセス制御を行うことに関しては、技術的な内容も含め深く検討されていない。本論文では、Capability モデルに基づく認可アーキテクチャとネットワークにおけるアクセス制御を行うアクセス制御手法を提案するだけでなく、それらのプロトタイプ実装を行い、実際にアクセス制御ができることを検証によって確認した。

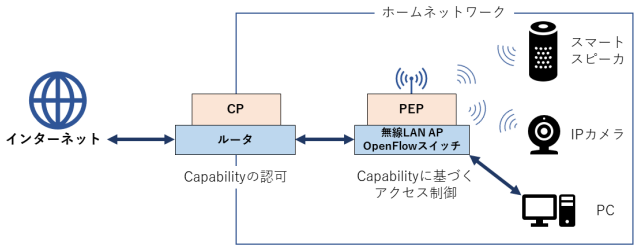

図 2　提案の構成

3. Capability モデルに基づく認可アーキテクチャ

3.1 要件設定

アクセス制御を行うホームネットワークは、図 2 にあるように外部への接続にルータを設置しデバイス間の通信は単一セグメント内で無線 LAN AP やスイッチを介して行われるネットワークを想定している。第 2.1 節で述べた Mazurek ら [11] によるガイドラインを元に認可アーキテクチャの要件を設定した。(6) については (3) の Reactive Policy Creation によって部分的に解決可能であるとしているため、(3) と合わせて考慮する。(8) については認可アーキテクチャ自体の問題ではなく、UI に関する問題が中心であるため今回は議論しない。これらより、ホームネットワークにおけるアクセス制御では、Capability モデルに基づく認可 ((1), (2), (3), (4), (5))、部分的な自動認可 ((5), (7)) を含む認可アーキテクチャを提案する。

3.2 提案手法の概要

第 3.1 節で整理した要件を満たす認可アーキテクチャとして、Capability モデルに基づく認可アーキテクチャを提案する。

本節で提案するアーキテクチャでは、デバイスの各機能を Capability として表現する。一連の認可を行うための構成要素として、デバイス群の Capability を管理する Capability Provider(CP)、Capability をアクセス制御に適用する Policy Enforcement Point(PEP) を置く。あらゆる Capability は CP に対して委譲され、 CP で Capability の発行やユーザーへの委譲が行われる。PEP の詳細は第 4 章で取り扱う。

ユーザーに対して認可を要求する頻度を減らすために Capability の委譲を用いた自動認可の仕組みが CP に組み込まれている。自動認可を受け入れる Capability には Capability を要求するデバイスの情報等を条件とした認可ポリシが組み込まれており、CP はその認可ポリシを元に認可するか否かの判断を自動的に行う。

図 2 にあるように CP と PEP は同一ないしは異なるホスト上で稼働し、CP はホームネットワーク内で常時稼働するホスト、 PEP は部屋に設置された無線 AP 付きのホストで稼働することを想定している。

3.3 構成要素

認可アーキテクチャはデバイスが保有する Capability、Capability Request と Capability Provider、 Policy Enforcement Point から構成される。

3.3.1 Capability と Capability Request

デバイスは自身の機能を表す Capability と他のデバイスに対して Capability を要求するための Capability Request を保有する。

Capability は発行者と被発行者、Capability が指し示す機能（温度取得や音声取得など）とそれに対応するプロトコルとポート番号、自動認可を行うための認可ポリシから構成される。認可ポリシでは、自動で認可するデバイスやデバイスベンダ の ID を指定することが可能である。さらに、外部通信に関する Capability の認可ポリシでは、自動で認可するドメインを `*.example.com` のようにワイルドカード付きで指定し `aaa.example.com` に対する外部通信の Capability は自動で認可するといったことができる。第 3.4 節で述べる認可の例 (図 3) では下記のような Capability が発行されている。

```
{
  // Capabilityの固有ID
  "capabilityID":"004bb012-3e39...",
  // Capabilityの発行者ID
  "assignerID":"ef4bfdac-5760...",
  // Capabilityの被発行者ID
  "assigneeID":"23c186a6-8ec9...",
  // Capabilityが表す機能名
  "capabilityName":"Temperature",
  // 通信に利用するプロトコルとポート番号
  "capabilityValue":"80/tcp",
  // 委譲された場合、元となったCapabilityのID
  "authorizeCapabilityID":"e1ee77ff-d7c6...",
  // 自動認可のポリシ
  "capabilityGrantPolicy":{
    // 認可判断に利用する情報
    "requesterAttribute":"VendorID",
    // 自動認可する場合のベンダID
    "requesterVendorID":"a726c1bd-e3d6-..."
  }
  "capabilitySignature":{
    // 署名者ID
    "signerID":"ef4bfdac-5760...",
    // 被署名者ID
    "signeeID":"23c186a6-8ec9...",
    // 署名
    "signature":"gI+wYKnYxN/XIOwWh..."
  },
}
```

Capability Request は Capability を要求するときに利用され、要求者と被要求者、要求する Capability に関する情報から構成される。3.4 節で述べる認可の例 (図 3) では下記のような Capability Request が送られている。

```
{
```

```
  // Capability Requestの固有ID
  "requestID":"8450ebba-5e9a...",
  // Capability要求者ID
  "requesterID":"993c0162-5b8f...",
  // Capability被要求者ID
  "requesteeID":"bcb851d2-f979...",
  // 要求するCapabilityの機能名
  "requestCapability":"Temperature",
  "requestSignature": ...
}
```

Capability はアクセスの認可を証明する要素であるため、第三者が不正な Capability を発行する可能性がある。そのため、Capability と Capability Request が正当なものであるか検証するために公開鍵暗号による署名 (PKCS#1[15]) が組み込まれている。各デバイスやユーザー、 CP、 PEP は Capability と Capability Request の署名と検証を行うために、4096bit RSA 秘密鍵と、認証局によって署名された RSA 公開鍵を含む証明書を持つ。

3.3.2 Capability Provider(CP)

Capability Provider は、デバイス間の Capability のやり取りを取りまとめる要素である。各デバイスから Capability と Capability Request を受け取り、自動認可や手動認可によって Capability を発行する。

最初に対象のデバイスから CP に対して Capability が委譲される。CP は、委譲された Capability が正当なものであるか検証し、正当なものであると判定したら自身に対して委譲された Capability としてまとめて管理する。

他のデバイスが CP に対して Capability Request を送った場合、CP は Capability Request が正当なものであることを検証し、自動認可または手動認可を行う。認可の流れについては第 3.4 節で述べる。認可対象のデバイスが個人的なデバイスであるといった理由で他のユーザーに利用してほしくない場合は、自動認可と手動認可をユーザーによるポリシ設定で制御することができる。ポリシでは、特定のデバイス全体やあるデバイスの特定の Capability に関して自動認可を停止する設定や、特定ユーザーの手動認可のみ認める設定を行うことが可能である。

3.3.3 Policy Enforcement Point(PEP)

Policy Enforcement Point は、デバイスから送られてきた Capability を元にネットワークのアクセス制御を行う要素である。PEP は受け取った Capability を検証し、検証した結果正当な Capability としてみなされた場合は、Capability に含まれる情報を元にネットワークアクセス制御を行う。アクセス制御の詳細は第 4 章で取り扱う。

3.4 認可の流れ

本認可アーキテクチャでは、自動認可を行う場合とユーザーによる手動認可を行う場合が存在し、それぞれ認可の流れが異なる。ここでは、デバイス間で温度取得を行う場合を例に認可の流れを説明する。デバイスから温度を取得するために必要な Capability を Cap(温度取得) 、Cap(温度取得) を要求するための Capability Request を CapReq(温度取得) とする。

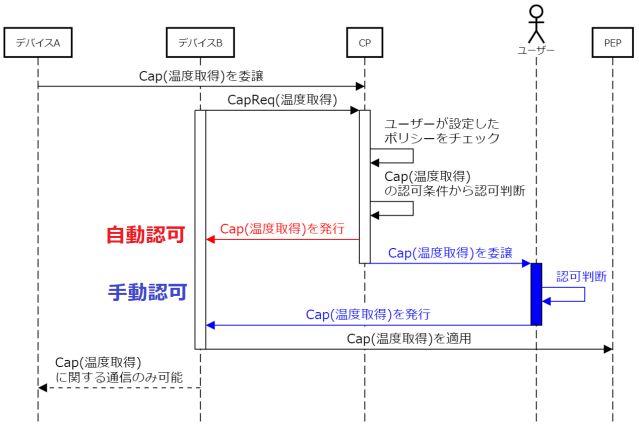

図 3 自動認可・手動認可の流れ

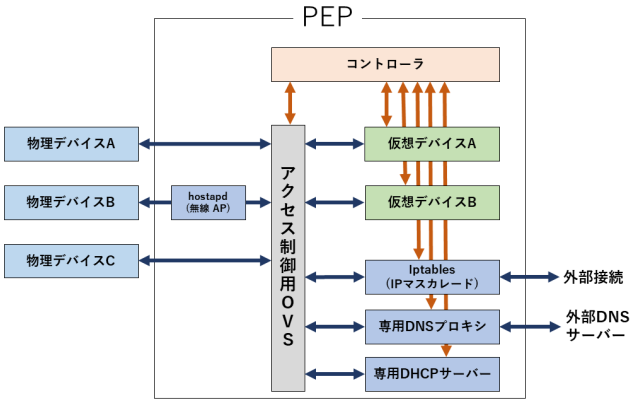

図 4 PEP の内部構成

自動認可によって Capability を発行する場合は図 3 の自動認可の流れに従う。デバイス A が Cap(温度取得) を CP に対して委譲し、 デバイス B が CapReq(温度取得) で要求するが、Cap（温度取得） にはデバイス B による要求の場合は自動的に認可するポリシが組み込まれているため自動的に CP がデバイス B に対して Cap(温度取得) を発行する。

手動認可によって Capability を発行する場合は図 3 の手動認可の流れに従う。デバイス B が CapReq（温度取得）で要求するが、自動認可に失敗するため、CP はユーザーに対して Cap（温度取得）を委譲して認可の可否を問う。ユーザーが認可した場合はユーザーがデバイス B に対して Cap（温度取得）を発行する。手動認可では Capability を CP ではなくユーザーが発行するため、誰が認可を行ったか容易に確認することができる。

4. Capability に基づくネットワークアクセス制御

4.1 提案手法の概要

Capability に基づくネットワークのアクセス制御を行う手法を提案する。本章で提案するネットワークアクセス制御は第 3 章の認可アーキテクチャで発行された Capability を元に、OpenFlow を利用したフロー制御とデバイス側でのパケットのペイロードに基づくパケット転送制御によって実現される。

フロー制御については OpenFlow の規格に準拠したソフトウェアスイッチである Open vSwitch[*1](OVS) を用いた。デバイス側でのパケットのペイロードに基づくパケット転送制御では、パケットのペイロードを解釈し Capability に準ずるパケットであれば転送し、そうでないパケットでなければ破棄することによってアクセス制御を行う。さらに、ユーザーにとって重要度が高い [23] が、OpenFlow やパケットのペイロードの解釈だけでは実現することが困難な外部通信に関するアクセス制御については、専用の DNS プロキシを実装することで実現した。

物理デバイス単体でのアクセス制御に加え、本提案が組み込まれていない既存の物理デバイスや計算能力が貧弱なデバイスについては図 4 にあるように PEP 内で仮想デバイスという仮想的なソフトウェアデバイスを利用してアクセス制御を実現している。仮想デバイスでは第 3 章における Capability の処理を物理デバイスに代わって行う。さらに、パケットのペイロードに基づく制御についても図 5 にあるように仮想デバイスにおいて行う。

仮想デバイスを利用する場合、物理デバイスごとに仮想デバイスを設定する必要があるが、事前設定の手間を削減するために仮想デバイスの自動構成を行う機能を設計し実装した。Capability の自動認可と仮想デバイスの自動構成によって、全自動で物理デバイスのアクセス制御を行うことが可能である。

4.2 PEP の内部ネットワーク

提案手法を実現するために図 4 にあるネットワークを設計した。アクセス制御を行う PEP の内部ネットワークにおいて、すべての物理デバイスと仮想デバイスはアクセス制御用 OVS を経由して接続されている。物理デバイスや仮想デバイスと外部との通信は、 iptables による IP マスカレードを行うことで、PEP ホストを経由して行われる。PEP の内部ネットワークは単一サブネットで構成されており、物理デバイスはブロードキャストパケットによる探索を含め、既存のホームネットワークと同じ条件下で稼働する。

仮想デバイスを利用する場合、物理デバイス間の通信は

*1 http://www.openvswitch.org/

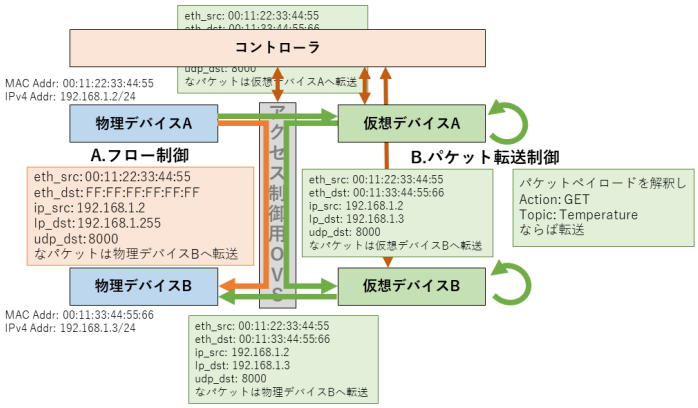

図 5 フロー制御とパケット転送制御

図 5 にあるようにアクセス制御用 OVS 内でのフロー制御と仮想デバイスでのパケットのペイロードに基づくパケット転送制御によって制御される。図 5 の A. フロー制御のように、フロー制御では、物理デバイスが接続されているアクセス制御用 OVS に対して、物理デバイス間のフローを Capability に基づいて追加し制御する。フロー制御は 5 タプルを元に制御を行うため、HTTP のように同じポートで異なる通信を行う場合は十分に制御をすることができない。そのような場合は、フロー制御だけでなく、仮想デバイス内でパケットのペイロードを解釈しパケット転送の制御によって対応する。図 5 の B. パケット転送制御にあるように、パケットはフローに従い物理デバイスから仮想デバイスに対して転送され、仮想デバイス内でペイロードを解釈して転送の制御を行う。仮想デバイスで転送されたパケットは相手側の仮想デバイスに転送される。相手側の仮想デバイス内でもペイロードをチェックされたのち、物理デバイスに対して転送される。パケット転送の制御はペイロードを全て解釈するため計算コストが高くなる。そのため、大容量ファイルの転送といった特定のフローで大量のパケットが流れる場合はフロー制御のみを利用し、ペイロードを解釈する必要があるパケットのみ仮想デバイスを経由する。

また、外部通信に関してはデバイス間通信のアクセス制御手法はそのまま利用できないため、外部との通信をドメインベースで制限するために専用の DNS プロキシによって、ドメインに対応する IP アドレスのフローを許可するという処理を行う。

物理デバイスとアクセス制御用 OVS との接続は有線に限らず、hostapd を用いたソフトウェアアクセスポイントによる無線 LAN 経由での接続も可能である。hostapd は標準では OVS に対応していないため、対応パッチを施したものを利用している。

コントローラ

コントローラは OVS や仮想デバイス、補助コンポーネントを管理し、全体の制御を行う。コントローラに対して

引き渡された Capability は正当性が検証され、検証に成功した場合は図 5 にあるようにフロー制御とパケット転送制御に必要な指示を行う。ブロードキャストパケット等のパケット転送制御を行わない場合や、仮想デバイスが存在しない物理デバイスがパケット転送制御に対応している場合、仮想デバイスがパケット転送制御に対応していない場合は、図 5 の A. フロー制御のように、コントローラは物理デバイス間のフローをアクセス制御用 OVS に追加する。DHCP サーバーとの通信や外部通信を行う場合は、物理デバイスと DHCP サーバー、DNS プロキシ、iptables が稼働するポート間のフローを追加する。

物理・仮想デバイス間の通信を行う場合や仮想デバイスにおいてパケットのペイロードを解釈して転送制御を行う場合は、図 5 の B. パケット転送制御のように、物理デバイスからの特定フローのパケットが仮想デバイスを通過するように適切なフローをアクセス制御用 OVS に追加し、仮想デバイスに対してはパケットのペイロード解釈と転送制御を指示する。

仮想デバイス

仮想デバイスの実体は Linux のプロセスであり、Linux network namespace を利用することでホストや他の仮想デバイスのネットワークと分離している。図 5 にあるように、仮想デバイスでは認可された Capability に応じてパケットのペイロードを解釈して転送制御を行う。パケット転送制御を行う場合、アクセス制御用 OVS に追加されたフローによって物理デバイス間の通信は図 5 にあるように仮想デバイスを通過するが、そのとき仮想デバイスでパケットのペイロードを解釈し、フロー制御では不可能な、ペイロードに基づく細かい制御を行う。

専用 DNS プロキシ

外部通信の制御において、フロー制御を利用してドメインベースのアクセス制御を行うためにはドメインと IPv4 アドレスの対応を取得する必要がある。専用の DNS プロキシによって、A レコードに関するクエリと結果を取得することで、ドメインと IPv4 アドレスの対応関係を把握し、アクセス制御を実現している。

専用 DHCP サーバー

物理デバイスに対して IPv4 アドレスを割り当てつつ、IPv4 アドレスとデバイスの対応関係をコントローラに対して通知する。また、物理デバイスに対応する仮想デバイスに関する情報を取得する。

4.3 デバイス間のアクセス制御

デバイス間のアクセス制御は図 5, 図 6 にあるように行われる。図 5, 図 6 では物理デバイスと仮想デバイスに分かれているが、物理デバイスが Capability の処理とパケット転送制御に対応しているならば仮想デバイスの役割を物理デバイスが担う。デバイスが保持する Capability を PEP に対して提示すると、コントローラがフローを生成しアクセス制御用 OVS に追加する。さらに、必要に応じて仮想デバイスに対してパケット転送の制御を指示することで Capability に応じたパケットの制御を行う。

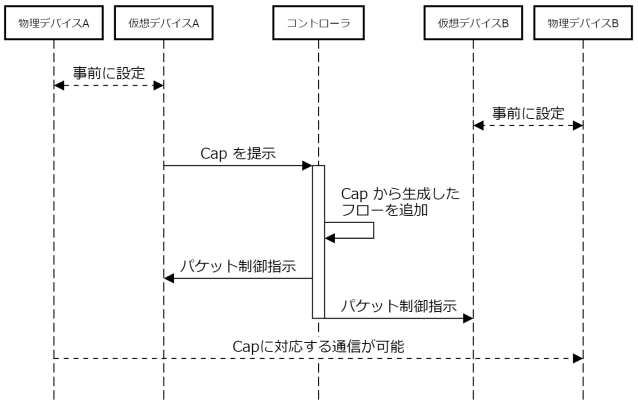

図 6 デバイス間通信の制御

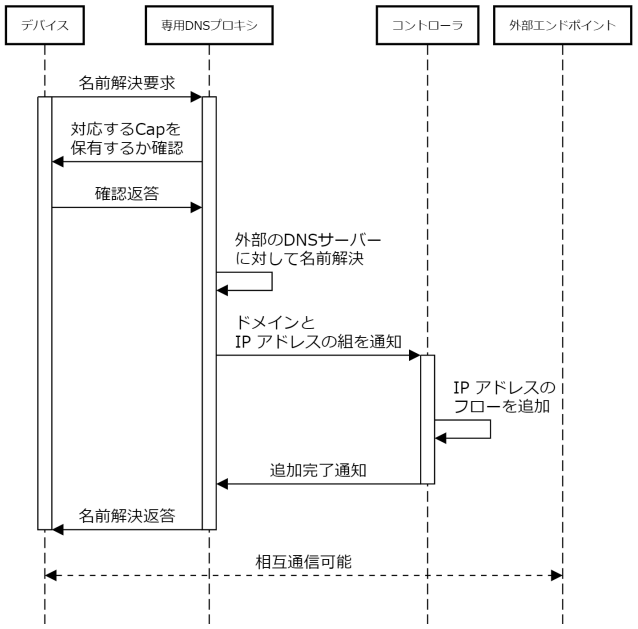

図 7 外部通信の制御

4.4 外部通信の制御

デバイスと外部エンドポイント間のアクセス制御はデバイス間のアクセス制御と異なり、図 7 にあるように行われる。外部通信については、IPv4 アドレスが変化する可能性があることや、DNS を利用した冗長化、負荷分散に対応するためにドメインベースで制御が行われる。名前解決を行うことでドメインに対応するエンドポイントへの通信が可能になる。

外部通信に関する Capability は PEP が保有し、第 3 章の認可アーキテクチャによってデバイスに発行される。専用 DNS プロキシは、デバイスからの A レコードのクエリを待ち受ける。受け取ったクエリについて、ドメインに対

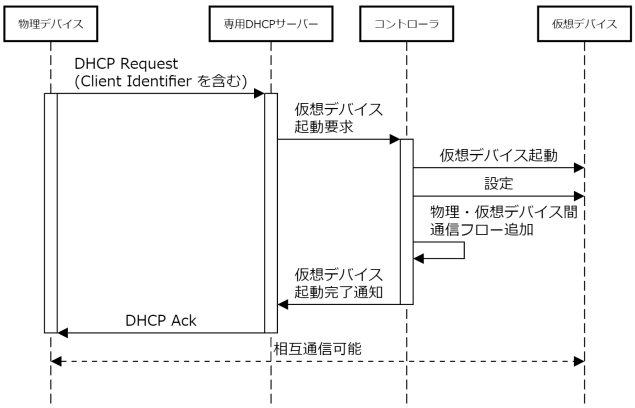

図 8　仮想デバイスの自動構成

応する Capability をデバイスが保有している場合はクエリを外部 DNS サーバーに対してリダイレクトする。受け取ったクエリとその結果から、デバイスが通信しようとしている外部エンドポイントのドメインと対応する IPv4 アドレスのペアが得られるため、コントローラはドメインに対応する IPv4 アドレスのフローをアクセス制御用 OVS に追加することで通信が可能になる。

4.5　仮想デバイスの自動構成

ユーザーが手動で設定する必要がある仕組みが好ましくないことは第 3.1 節で述べた。そこで、物理デバイスを PEP のネットワークに接続すると、物理デバイスから仮想デバイスに関する情報を取得し、自動的に適切な仮想デバイスを起動する仕組みを実装した。

物理デバイスから仮想デバイスに関する情報を取得するために、DHCP による IPv4 アドレスの割り当てにおいて Option 61 Client Identifier[20] というクライアントに関する識別子のオプションを利用した。図 8 にあるように、専用 DHCP サーバーは物理デバイスの DHCP Request に含まれる ClientIdentifier からコントローラに対して対応する仮想デバイスを起動する。ClientIdentifier を利用できない物理デバイスについては MAC アドレスと仮想デバイスの対応関係を設定しておくことで等価な動作を行うように設計されている。

5.　実装・検証

第 3 章と第 4 章で提案した認可アーキテクチャとアクセス制御手法についてプロトタイプ実装を行い、実デバイスを用いた検証を行った。プロトタイプ実装は CREBAS[*2] というプロジェクト名で公開している。

5.1　検証環境

検証には図 9 にあるように以下の機材を利用した。

- PEP, CP 稼働用 PC: ThinkPad X260

[*2] https://github.com/naoki9911/CREBAS

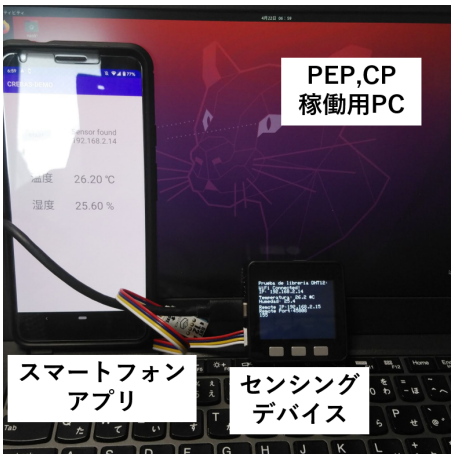

図 9　検証環境の機材

- センシングデバイス: M5Stack + 温湿度センサ
- スマートフォン: Google Pixel 3a XL

センシングデバイスでは温度と湿度を測定し、他のデバイスやアプリからの要求に応じて測定値を返す機能が備わっている。ユニキャストでセンシングデバイスに対して要求することで測定値を得ることができる。また、センシングデバイスと自動で連携するために近傍探索機能が存在する。ブロードキャストされた探索パケットを受け取るとセンシングデバイスは探索パケットの送信元に対してユニキャストで返答を行う。センシングデバイス自体はアクセス制御を行わないためあらゆるアプリやデバイスが測定値を取得することが可能である。センシングデバイスは Capability として近傍探索を許可する Cap(近傍探索)、測定した温度の取得を許可する Cap(温度取得)、測定した湿度の取得を許可する Cap(湿度取得) を持つ。

スマートフォンアプリはセンシングデバイスの測定値を取得し表示する機能を持つ。センシングデバイスに対してユニキャストで測定値を取得する。また、センシングデバイスを検出するためにブロードキャストで探索パケットを送信し、自動でセンシングデバイスを発見、測定値を取得する。アプリは Cap(近傍探索)、Cap(温度取得)、Cap(温度取得) を要求する。

センシングデバイスとスマートフォンは PC の hostpad による無線 LAN AP 経由で PEP のネットワークに接続される。センシングデバイスとスマートフォンそれぞれに対応する仮想デバイスが存在し、デバイス、スマートフォンの MAC アドレスと関連付けられている。仮想デバイスにはセンシングデバイスとの通信に利用するパケットを解釈し転送を制御する機能が実装されている。

5.2　検証シナリオ

本実験では、実験用に開発した温湿度を測定するセンシングデバイスと取得した温湿度を表示するスマートフォンアプリを用いて、認可アーキテクチャの自動認可及び手動

認可とアクセス制御手法におけるフロー制御とパケットのペイロードに基づく転送制御、仮想デバイスの自動構成を実現できていることを確認した。

1. 仮想デバイスの自動構成、自動認可およびフロー制御

センシングデバイスの Cap(近傍探索) はスマートフォンアプリからの Capability Request であれば自動で認可する自動認可が設定されている。そのため、自動構成によって起動したスマートフォン用の仮想デバイスが正常に Capability を要求し、適用できていることを確認する。

センシングデバイスとアプリ間の通信は UDP 8000 番ポートで全て行われる。近傍探索については、アプリが UDP 8000 番ポートへブロードキャストで探索パケットを送信し、探索パケットを受け取ったセンシングデバイスがユニキャストでアプリに対して返答を行うことでセンシングデバイスの IPv4 アドレスを取得する。そのため、Cap(近傍探索) では、センシングデバイスとアプリ間の探索用ブロードキャストパケットに関するフロー制御を行うことで制御する。正常に Cap(近傍探索) を適用できていれば探索用ブロードキャストパケットに関するフローが追加され、アプリがセンシングデバイスを発見できるため、これについて確認する。

2. 手動認可およびパケットのペイロードに基づく転送制御

センシングデバイスの Cap(温度取得), Cap(湿度取得) は自動認可による認可は行われない。そのため、手動で認可することで該当通信をできるようになる。Cap(温度取得) と Cap(湿度取得) について、それぞれ手動で認可を行うことでアプリで値を取得できることを確認する。

前述したように、センシングデバイスとアプリ間の通信は、全て UDP 8000 番ポート間で行われる。そのため、フロー制御に利用する 5 タプルだけでは、温度取得に関する通信であるか、湿度取得に関する通信であるか判定することは不可能である。そのため、仮想デバイス内でパケットの中身を解析し、温度取得に関する通信であるか、湿度取得に関する通信であるか判定する。認可されている通信であればパケットを転送し、そうでなければパケットを破棄するため、Capability に即した通信の制御を行うことが可能となるため、これについて確認する。

5.3 検証結果

それぞれのシナリオについて用意した検証環境を用いて検証を行った。

1. 仮想デバイスの自動構成、自動認可およびフロー制御

図 10 は実際にセンシングデバイスとスマートフォンアプリ間の通信が仮想デバイスの自動構成と自動認可によって認可された Cap(近傍探索) に基づきフロー制御が行われている様子である。フロー制御によって、センシングデバイスとスマートフォンアプリ間の近傍探索用のブロードキャストパケットのみ許可されているため、センシングデバイスの発見はできている。しかし、値取得に関するユニキャスト通信は許可されていないため温度と湿度に関しては取得できなかったことを示す NO VALUE が表示されている。

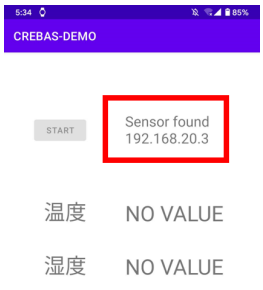

図 10 近傍探索のみ成功

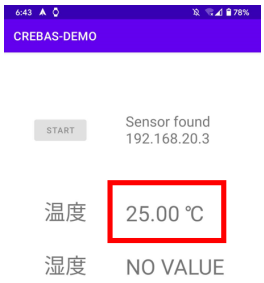

図 11 温度取得に成功

2. 手動認可およびパケットのペイロードに基づく転送制御

手動で Cap(温度取得) を認可すると、図 11 にあるように温度は取得できるが湿度は取得できない状態になっていることが分かる。

6. 考察

認可アーキテクチャとアクセス制御手法について、家庭におけるアクセス制御に関するガイドライン [11] の観点から考察する。

(1) 細かい制御が可能である

本提案では Capability に基づき OpenFlow によるフロー制御と仮想デバイスにおけるパケットのペイロードに基づく転送制御によるアクセス制御を行っている。そのため、本論文の提案における制御の粒度の限界は Capability の定義の粒度に基づき、Capability の定義を適切に行えば、細かい制御が可能である。

(2) デバイスの貸与を考慮する

本提案ではデバイスの貸与は考慮していない。一般の家庭では複数のユーザーが存在し、ユーザー間でのデバイスの貸与や共有 [6], [7], [10] が行われるため同一デバイスについてユーザーごとに異なるアクセス制御を行う必要がある。本提案では Capability を仮想デバイスに対して発行することが可能であるため、貸与される物理デバイスについてユーザーごとに仮想デバイスを切り替えることでデバイスを貸与する場合であってもアクセス制御をユーザーごとに変更することは可能である。一方で、デバイスの Capability のうち、デバイスの所有者のみが認可できるものや家族は認可できるもの、訪問客も認可できるものといった、認可可能な範囲が変化する場合があり、ユーザーごとに認可の制限を設けることができることが望ましい。複数ユーザーに関する仕組みは本提案には含まれていないが、認可アーキテクチャへの追加は容易であり、実装と検証が今後の課題である。

(3) Reactive Policy Creation が可能である

図 3 で示したように、自動で認可されなかったリクエストに関してはユーザーが手動で認可することができる。

(4) ログを含む

すべての Capability は CP を介して発行ないしは委譲が行われるため、アクセス制御に利用される Capability は CP 上で保持されている。そのため、アクセス制御に関するログは CP から容易に取得することができる。

(5) 事前設定の複雑さを軽減する

Capability が事前に組み込まれた仮想デバイスと仮想デバイスの自動構成を用いることで、物理デバイスを PEP の制御下にあるネットワークに接続するだけでアクセス制御を行うことが可能である。そのため、デバイスごとに個別の設定を行う必要はなく、ユーザーは手動認可における認可判断のみでアクセス制御を実現できる。手動認可については、細かい制御を行う上でユーザーに対する認可の要求頻度が増加するため対策として自動認可を導入したが、ユーザーごとにプロファイル [2] を導入し、プロファイル上で認めたデバイスや Capability についても自動で認可する拡張が可能である。

(7) ポリシの頻繁な書き換えを考慮する

デバイスが保持する Capability の更新については仮想デバイスを更新することで容易に実現できる。

以上より、アーキテクチャやアクセス制御の手法としてはガイドラインをおおむね満たしているが、今後の課題となる点も存在する。

7. 結論

本論文では、ホームネットワークにおけるアクセス制御の手法として、Capability モデルに基づいたユーザー主体の認可アーキテクチャと、認可された Capability に基づくネットワークアクセス制御手法を提案した。デバイスが行う通信を機能単位で Capability によって表現することで、細かい制御をユーザー主体で行うという、ホームネットワークの制御に求められる要件を満たす認可アーキテクチャを設計した。また、設計した認可アーキテクチャに Capability の委譲を利用した自動認可を組み込むことで、一部の認可をユーザーに委譲することなく自動で認可判断を下し、 ユーザーの負担を軽減しつつ、適切な Capability の発行を可能にした。認可された Capability に基づき、OpenFlow によるフロー制御と、パケットのペイロードを解釈しパケットの転送制御を行うことでデバイス間のネットワークアクセス制御を可能にした。さらに、仮想デバイスを導入することで物理デバイスに対する改変を最小限にとどめ、仮想デバイスで Capability の処理やパケットのペイロードに基づく制御を行い、仮想デバイス自体も自動構成によって起動することでユーザーの負担を増やすことなく本論文の提案を適用することを可能にした。

今後は、本論文の提案を拡張することで、ホームネットワークにおけるエッジコンピューティングに求められる要件や解決できる問題について模索する予定である。

参考文献

[1] Bakhshi, T. et al.: User-centric traffic optimization in residential software defined networks, *ICT 2016*, pp. 1–6 (2016).

[2] Egelman, S. et al.: Family Accounts: A New Paradigm for User Accounts within the Home Environment, *CSCW '08*, ACM, p. 669–678 (2008).

[3] Feamster, N.: Outsourcing Home Network Security, *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, HomeNets '10, ACM, p. 37–42 (2010).

[4] Gusmeroli, S. et al.: IoT Access Control Issues: A Capability Based Approach, *IMIS 2012*, pp. 787–792 (2012).

[5] Hernández-Ramos, J. L. et al.: DCapBAC: Embedding Authorization Logic into Smart Things through ECC Optimizations, *Int. J. Comput. Math.*, Vol. 93, No. 2, p. 345–366 (2016).

[6] Jacobs, M. et al.: Caring About Sharing: Couples' Practices in Single User Device Access, *GROUP '16*, ACM, p. 235–243 (2016).

[7] Jang, W. et al.: Enabling Multi-User Controls in Smart Home Devices, *IoTS&P '17*, ACM, p. 49–54 (2017).

[8] Lear, E. et al.: Manufacturer Usage Description Specification, RFC 8520 (2019).

[9] Levy, H. M.: 1 - Capability- and Object-Based System Concepts, *Capability-Based Computer Systems* (Levy, H. M., ed.), Digital Press, pp. 1 – 20 (1984).

[10] Matthews, T. et al.: *"She'll Just Grab Any Device That's Closer": A Study of Everyday Device & Account Sharing in Households*, p. 5921–5932, ACM (2016).

[11] Mazurek, M. L. et al.: *Access Control for Home Data Sharing: Attitudes, Needs and Practices*, p. 645–654, ACM (2010).

[12] Mazurek, M. L. et al.: Exploring Reactive Access Control, *CHI '11*, ACM, p. 2085–2094 (2011).

[13] McKeown, N. et al.: OpenFlow: Enabling Innovation in Campus Networks, *SIGCOMM Comput. Commun. Rev.*, Vol. 38, No. 2, p. 69–74 (2008).

[14] Mendoza, I. C. P. et al.: ImHome: An IoT for Smart Home Appliances, *IEEE ICIEA 2020*, pp. 761–765 (2020).

[15] Moriarty, K. et al.: PKCS #1: RSA Cryptography Specifications Version 2.2, RFC 8017 (2016).

[16] Mortier, R. et al.: Control and understanding: Owning your home network, *COMSNETS 2012*, pp. 1–10 (2012).

[17] Osman, A. et al.: Transparent Microsegmentation in Smart Home IoT Networks, *HotEdge '20*, USENIX, pp. 1–6 (2020).

[18] Sandhu, R. S. et al.: Role-based access control models, *Computer*, Vol. 29, No. 2, pp. 38–47 (1996).

[19] Sivaselvan, N. et al.: Authentication and Capability-based Access Control: An Integrated Approach for IoT Environment, *ICCSN 2020*, pp. 110–117 (2020).

[20] Swamy, N., Halwasia, G. and Unit, S.: Client Identifier Option in DHCP Server Replies, RFC 6842 (2013).

[21] Vallati, C., Virdis, A., Mingozzi, E. and Stea, G.: Mobile-Edge Computing Come Home Connecting things in future smart homes using LTE device-to-device communications, *IEEE Consumer Electronics Magazine*, Vol. 5, No. 4, pp. 77–83 (online), DOI: 10.1109/MCE.2016.2590100 (2016).

[22] Xu, R. et al.: A federated capability-based access control mechanism for internet of things (iots), *Sensors and Systems for Space Applications XI*, Vol. 10641, International Society for Optics and Photonics, p. 106410U (2018).
[23] Yao, Y. et al.: Defending My Castle: A Co-Design Study of Privacy Mechanisms for Smart Homes, *CHI 2019*, ACM, p. 1–12 (2019).