2025 年度　卒業論文

DID に基づいた IoT データ管理システムの構築と評価

2026 年 2 月 10 日

システム工学部システム工学科
(学生番号: 60276128)

竹内 結哉

和歌山大学システム工学部

1　はじめに

近年,IoT 機器の爆発的な増加に伴い, 生成されるデータ量は急激に増加している. さらに,IoT は家庭や産業, 医療, 農業など多様な分野で活用されるようになり, 生成されるデータの種類や粒度も一層多様化している.

また, 従来の中央集権型による IoT データ管理には, 以下の 3 つの課題が存在する. 第一に, スケーラビリティの問題である.IoT デバイスの急増により, 中央サーバーへの負荷が指数関数的に増大し, 処理能力の限界に達する可能性がある. 第二に, セキュリティ上の問題である. 中央サーバーは単一障害点となりやすく, 攻撃対象として脆弱である. 第三に, プライバシー保護の問題である. 個人情報を含む IoT データが集中することで, 情報漏洩時の被害が甚大化するリスクがある.

これらの課題を解決するために, 本研究ではユーザ主権型 ID に基づいた分散型データ管理システムの実現を目指す. 本システムは, 以下の三点を重視して設計されている.

1. データの分散管理：中央集権型から脱却し, 分散型ファイルシステムである InterPlanetary File System(以下 IPFS) を用いることで, 単一障害点を排除しシステムの堅牢性を向上させる.
2. ユーザの真正性確保：分散管理環境におけるなりすまし防止のため, 分散型識別子である Decentralized Identity(以下 DID) を活用し, データ所有者の身元を保証する.
3. データの信頼性と改ざん防止：ブロックチェーンを活用し, データが改ざんされていないことを検証可能とする.

以上の要素を組み合わせることで,IoT データに対する分散型かつ信頼可能な管理基盤を構築することを目指す.

2　関連研究

IoT データ管理におけるプライバシー確保は, 従来より大きな研究課題とされている. 現行の IoT システムは, 多くが集中型のクライアントサーバーモデルに依存しており, 生成される膨大なデータはサービスプロバイダを介して管理される. このような集中型モデルは, ユーザの行動履歴や個人情報が第三者に漏洩・不正利用されるリスクを内包している. この問題に対処するため, 近年ではブロックチェーンを基盤とした分散型データ管理アーキテクチャの研究が進められている.

IoT とブロックチェーンの統合に関する典型的なユースケースとしては, (1) イベントの改ざん防止ログ, (2) アクセス制御の管理, (3) IoT センサーデータの購入などが挙げられる [1]. これらの研究では, ブロックチェーンが持つ分散性と改ざん耐性を活用し,IoT 環境におけるデータ完全性の確保を目指している. 一方で, 既存の研究の多くは概念設計やプロトコル提案に留まり,Proof-of-Concept レベルの実装や性能評価が十分に行われていないことが課題とされている.

Ali ら [2] は, ブロックチェーンと IPFS を組み合わせた**モジュラーコンソーシアムアーキテクチャ**を提案している. このモデルでは,IoT デバイスをプライベートな「サイドチェーン」にグループ化し, アクセス制御の管理を「コンソーシアムブロックチェーン」によって実現する. サイドチェーンはセンサーデータ生成イベントを記録し, コンソーシアムブロックチェーンはアクセス要求の不変なログを保持することで, プライバシー保護とアカウンタビリティを両立させている. さらに, データ自体は IPFS 上に保存され, ブロックチェーンはハッシュのみを記録することで, ストレージ効率とセキュリティの両立を実現している.

評価実験として,Ethereum(PoW) および Monax(PoS/Tendermint) を用いた性能比較が行われている. その結果,Monax はサイドチェーンレベルで低い処理オーバーヘッドを示す一方で, コンソーシアムレベルでは高いネットワークトラフィックオーバーヘッドが課題となった. Ethereum はコンソーシアムレベルで通信効率に優れるものの,PoW に基づく高い計算コストが問題点として指摘されている. このように, ブロックチェーンのコンセンサスメカニズムの選択は,IoT 分散アーキテクチャの実用性に直接影響を与えることが明らかとなっている.

また,[1] では,IoT センサーから得られるデータを**二重チャネル (Dual Channel)** で配信するミドルウェアが提案されている. 一方は Ethereum と IPFS を用いたインテグリティチャネルで, データ完全性を保証する. もう一方は MQTT を利用したリアルタイムチャネルで, 低遅延かつ高速な配信を実現する. この方式は, 後からインテグリティチャネルを参照することで, リアルタイムチャネル経由で受け取ったデータの改ざん検出が可能になる点に特徴がある. ただし,Raspberry Pi を用いた評価では, インテグリティチャネルにおいて数百秒規模の遅延やデータ欠損が発生

するなど, ブロックチェーンのオーバーヘッドとリソース制約の影響が確認されており, 実運用に向けた課題も示されている.

本研究は, これらの先行研究の知見を基盤としつつ,IoT データ管理における **DID** と **VC** の統合に焦点を当てる. 既存研究が主に公開鍵基盤に依存したアクセス制御を行っていたのに対し, 本研究では DID/VC を導入することで, より柔軟かつ標準化された認証・検証モデルを提供する点に新規性がある. また,Ethereum および IPFS を用いたローカル環境での実装と性能測定を通じて, スケーラビリティと実用性の両面から評価を行うことを目的としている.

3 準備

本研究では, 分散型データ管理の基盤技術として IPFS,DID, およびブロックチェーンを用いる. 本章では, これらの技術の概要を説明し, さらに本研究で使用した実験環境について述べる.

3.1 IPFS

IPFS は, 分散型のファイルシステムであり, コンテンツ指向のアドレッシング方式を採用している. 従来の URL が「場所」に基づいてデータを参照するのに対し,IPFS ではデータ内容から生成されるハッシュ値を用いて「内容」を参照する. これにより, 同一のデータは同一のハッシュ値で一意に識別され, データの改ざん検知が容易になるとともに, ネットワーク上の複数ノードに分散保存することで耐障害性を高めることができる. 本研究では,IPFS の実装として `go-ipfs` を用いた. これは IPFS の公式実装であり,Go 言語で開発されている.

3.2 DID

DID は, 自己主権型アイデンティティを実現するための分散型識別子である. 従来のインターネットにおける認証・識別は, 中央集権的な認証局やサービス提供者に依存していたため, 単一障害点や利用者のプライバシー保護に課題があった. これに対し DID は, ユーザ自身が管理可能な識別子を提供することで, 第三者に依存せずに真正性を保証できる仕組みを実現する. DID Document と呼ばれる文書には公開鍵やサービスエンドポイントなどが含まれ, これにより利用者の認証やデータ検証を行うことができる.

3.3 ブロックチェーン

ブロックチェーンは, 分散型台帳技術の一種であり, 取引情報をブロックとして記録し, チェーン上に連結することで, データの改ざん耐性を保証する. 各ブロックは暗号学的ハッシュにより前後のブロックと結合されており, 一部のデータが改ざんされると以降のブロックすべてが不整合となるため, 改ざん検知が容易である. また, ネットワークに参加する複数ノード間で合意形成を行い, 信頼できる台帳を分散的に維持する仕組みを持つ. 本研究では実験環境として Ethereum 互換のローカルブロックチェーン環境である `Ganache` を使用した. Ganache は開発用に特化したブロックチェーンシミュレータであり, 高速なテスト実行やトランザクションの記録確認を容易に行うことができる.

さらに, 既存研究においても議論されているように,Proof-of-Work(以下 PoW) はエネルギー消費が大きく, リソース制約のある IoT 環境には適さないことが指摘されている [1]. そのため, 本研究においても PoW ベースのブロックチェーンは採用せず, より実装や評価に適した Ethereum 環境を主に使用することとした.

4 システム構成

本研究で提案するシステムは,IoT データを分散的に管理するために,DID,IPFS, およびブロックチェーンを連携させたものである. 本章では, まずシステム全体の流れを示した後, 各要素の役割について説明する.

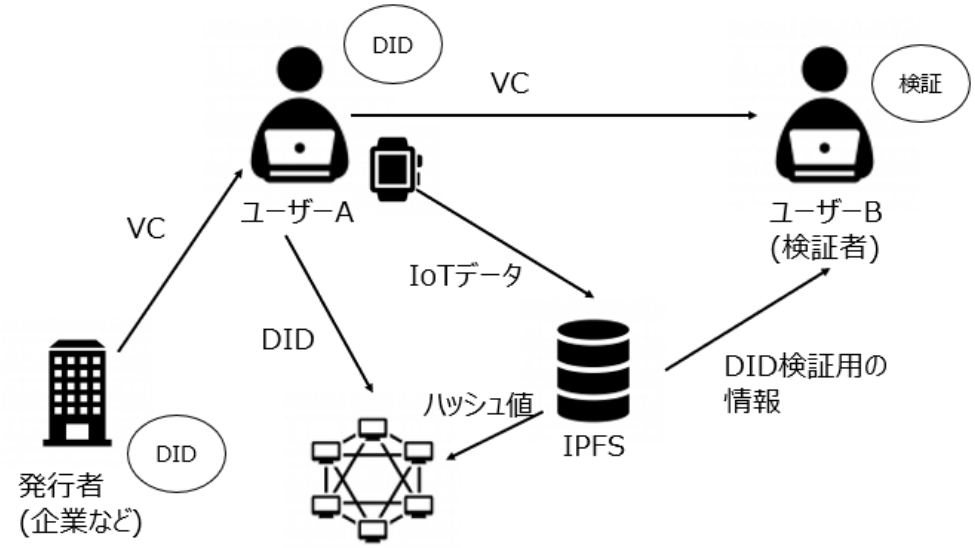

図 1　提案システムの概要図

4.1　システムの概要

本研究で提案するシステムの全体像を図 1 に示す. 1 に示すように, ユーザ A は自身の IoT データを IPFS に格納し, その結果として得られるハッシュ値 (以下 CID) を, 自身の識別子である DID とともにブロックチェーンに記録する. その後, 発行者が IoT データの真正性を保証する Verifiable Credential(以下 VC) を発行し, 検証者であるユーザ B がブロックチェーン上の DID 情報と VC を突き合わせることで, ユーザ A の正当性を検証する.

4.1.1　データ所有者 (ユーザ A)

ユーザ A は, 自身の識別子として DID を保持し, その検証用情報 (公開鍵など) をブロックチェーンに格納する. さらに, 保持している IoT データを IPFS に格納し,IPFS から返される CID を取得する. この CID はデータの一意なハッシュ値であり, データ改ざん検知に利用できる. ユーザ A は,CID と自身の DID を組み合わせてブロックチェーンに登録することで, データの所有者であることを保証する.

4.1.2　発行者 (企業など)

発行者とは, ユーザ A に対して VC を発行する主体である. 本研究ではユーザ A が所持している IoT 機器の製造元企業などを想定している. 発行者はユーザ A が保持している IoT データが自社製品によって生成されたデータであることを確認し, その真正性を保証する VC をユーザ A に発行する.

4.1.3　検証者 (ユーザ C)

検証者は, ユーザ A と IoT データを取引する相手, すなわちデータの受領者を想定している. 検証者は, ユーザ A から提示された VC を受け取り, ブロックチェーン上の DID 情報と照合することで, ユーザ A が真正なデータ所有者であることや, IoT データについて企業が保証していることについて確認する.

4.2　データフロー

本システムの具体的な処理の流れを図 2 に示す. 1 に示すように, ユーザ A は IoT データを IPFS に格納し, 得られた CID と自身の DID をブロックチェーンに登録する. その後, ユーザ B が VC を発行し, ユーザ A はこれをユーザ C

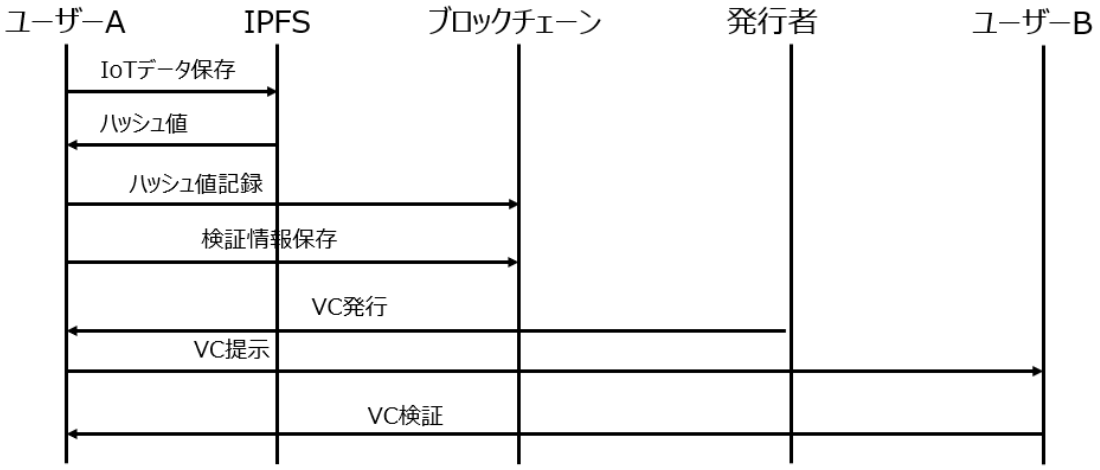

図 2　データフロー図

に提示する. ユーザ C はブロックチェーン上の DID 情報と VC を照合することで, ユーザ A の正当性を検証する.

4.3　システムフロー

Algorithm 1 DID ドキュメント登録処理

Require: Ethereum ノード URL, DIDRegistry(ABI, Address)
Ensure: UserA/Company の DID Document がブロックチェーンに登録される

```
web3 ← Web3(ノード URL)
accounts ← web3.eth.getAccounts()
userA ← accounts[0]
company ← accounts[1]
didUserA ← “did:example:userA”
didCompany ← “did:example:company”
didDocUserA ← {id: didUserA, controller: userA}
didDocCompany ← {id: didCompany, controller: company}
JSON として didDocUserA を保存
JSON として didDocCompany を保存
networkId ← web3.eth.net.getId()
registry ← Contract(ABI, Address)
registry.registerDIDDocument(didUserA, didDocUserA) を userA から send
registry.registerDIDDocument(didCompany, didDocCompany) を company から send
```

5　実験と考察

5.1　実験目的

本研究の実験目的は, 提案システムが実社会の運用に耐えうる性能と有効性を備えているかを検証することである. まず,IoT データの記録から VC による検証までの一連の処理を再現し,DID・VC・IPFS・ブロックチェーンを統合したシステムとして正しく動作することを確認する. また, 分散ストレージとして利用する IPFS に対して, ファイルアップロード処理の処理時間・スループット・エラー率を測定し, 膨大なデータ量を扱う IoT 環境において実用的に運用可

能かを評価する. さらに,DID および VC を用いたユーザ検証処理, およびブロックチェーンへのトランザクション記録に要する処理時間を測定し, 提案システム全体の性能を定量的に評価する.

5.2 実験環境

実験は以下の環境で行った.

- OS:Windows 11 Home
- CPU:AMD Ryzen 5 PRO 7530U with Radeon Graphics(2.00Hz)
- メモリ:16GB
- IPFS:go-ipfs v0.35.0
- ブロックチェーン環境:Ganache v7.9.2
- Solidity: v0.5.16
- Node.js: v18.20.7
- Truffle: v5.11.5
- Web3.js: v1.10.0

5.3 実験方法

5.4 実験結果

5.5 考察

6 まとめ

参 考 文 献

[1] Simon Krejci, Marten Sigwart, and Stefan Schulte. Blockchain- and IPFS-based data distribution for the internet of things. In Antonio Brogi, Wolf Zimmermann, and Kyriakos Kritikos, editors, *Service-Oriented and Cloud Computing*, pp. 177–191, Cham, 2020. Springer International Publishing.

[2] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. IoT data privacy via blockchains and IPFS. In *Proceedings of the Seventh International Conference on the Internet of Things*, IoT '17, New York, NY, USA, 2017. Association for Computing Machinery.