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 (仮)

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

目 次

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 データ管理には主に次の 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, そのデータを保存する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. 本研究における提案システムは, 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：中央集権型管理から脱却し, 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 identifier（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, 多くが中央集権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．このような中央集権型モデルは, 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, ブロックチェーンおよび DID 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 InterPlanetary File System（IPFS）

IPFS は世界中のコンピュータ（ノード）に分散的にデータを保存する P2P 型のファイルシステムである. IPFS では中央集権的なサーバを介さず, データを分散的に管理しており, 耐故障性, 負荷分散, 耐検閲性, 改ざん耐性に優れている. 特徴は「コンテンツアドレス方式」である点で, 保存されたファイルはその内容を基に計算される Content Identifier（以下 CID）によって参照される. CID はファイル内容のハッシュ値であるため, 以下の性質を持つ.

- 同一内容のファイルは必ず同じ CID となる.
- 1 ビットでも内容が変更されれば別の CID となる.
- CID から元のデータを推測することはできない.

この仕組みにより, ファイルが改ざんされていないかを CID の比較によって確認できるため, 改ざん耐性に優れる.

本研究では, ユーザが保有する IoT データを IPFS に保存し, 得られた CID をブロックチェーンに記録することで, データの真正性と参照可能性を確保している.

3.2 ブロックチェーン

ブロックチェーンは, ネットワーク上の複数のノードが同一のデータを共有し, 合意形成に基づいて取引履歴を記録する分散型台帳技術である. 記録されるデータは複数の取引をまとめた「ブロック」に格納され, 各ブロックは直前のブロックのハッシュ値を保持することで鎖状に連結される. この構造により, 一部のブロックが改ざんされると以降すべてのブロックの整合性が崩れるため, 改ざんは即座に検知される. ここで重要なのは, ハッシュ値の性質である. ハッシュ値はデータから一方向的に算出される識別子であり, 内容にわずかな変更があっても全く別の値となる. また, ハッシュ値から元のデータを復元することはできない. ブロックチェーンではこの性質を利用し, データの完全性を保証している.

本研究では,IoT データの要約情報（CID）および後述する DID というユーザの識別子をスマートコントラクト経由でブロックチェーンへ記録することで, データ登録の証跡を改ざん不能に保持できるようにしている.

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.

3.3 Decentralized identifier（DID）

DID は特定の中央管理者に依存せずに個人が自分自身で生成・管理できる識別子であり, 分散型デジタルアイデンティティの基盤となる技術である. DID はその DID に紐づく公開メタデータである DID Document が存在している必要がある. DID Document には識別子の情報や DID の所有者が使用する公開鍵の情報が含まれており,DID の正当性や鍵の正しさを検証する場面では DID に紐づいた DID Document を取得して使用される. DID から DID Document を取得するという処理を DID 解決と呼ぶ. DID Document はブロックチェーンなどの改ざん耐性を持つ基盤に保存されることで, 第三者がその内容を検証可能となる.

本研究では, ユーザ A および発行者が発行した DID と DID Document をブロックチェーンに登録し,IoT データの所有者であることを証明するための基礎情報として利用する.

さらに本研究のシステムでは, 発行者がユーザに対して Verifiable Credential（以下 VC）と呼ばれる証明書を発行し, IoT データが正当なデバイスによって生成されたものであることを保証する仕組みを構築する. VC の検証時には,DID Document に記録された公開鍵により署名を確認し, データの真正性を確認することができる.

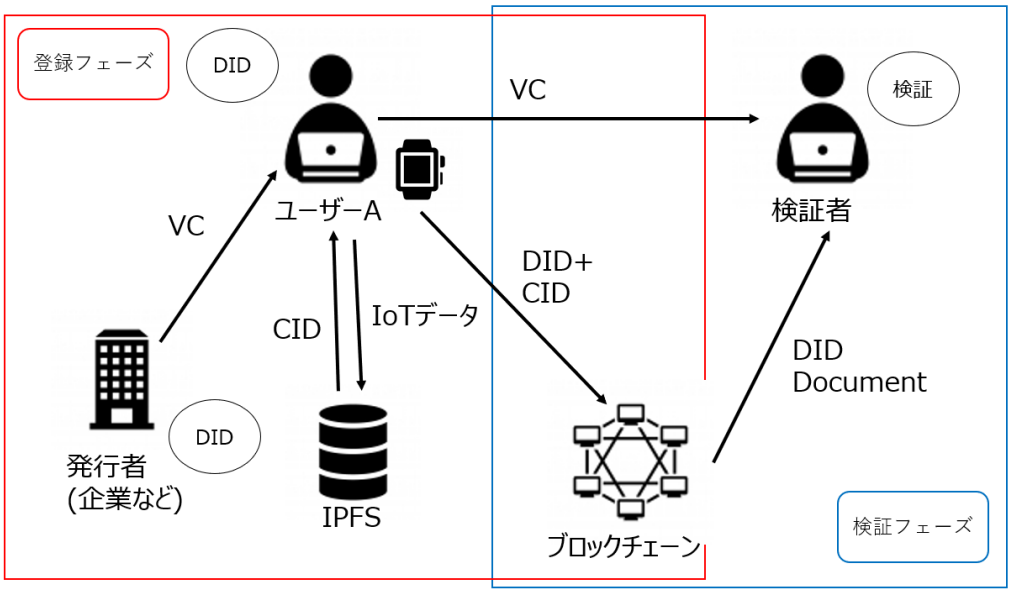

図 1　提案システムの概要図

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,IPFS, ブロックチェーンおよび DID 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.

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. 本研究で提案するシステムは,IoT データを保有するユーザ A, データの真正性を確認し VC を発行する発行者, および最終的に VC を検証する検証者により構成される. 図 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 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, 検証者がブロックチェーン上の DID Document と VC を突き合わせることで, ユーザ A およびデータの正当性を検証する.

4.1.1　データ保有者（ユーザ A）

ユーザ A は, 自身の識別子として DID を保持し,DID Document をブロックチェーンに格納する. さらに, 保有している IoT データを IPFS に格納し,IPFS から返される CID を取得する. 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および後に発行者から取得する VC に自身の秘密鍵で署名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 が保有している 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

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 Document 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ことができる．本研究で提案するシステムにおいては, 検証者が VC の正当性を確認した上でデータの取引を実行することを想定している．

5　システム全体のフロー

システムの処理は, **登録フェーズ**と**検証フェーズ**の 2 つに大別される. 登録フェーズでは DID・IoT データがブロックチェーンおよび IPFS に記録され, 発行者による VC 発行までが行われる. 一方, 検証フェーズでは, 提示された VC の署名検証を通じてユーザ A が真正なデータ提供者であることを確認し, 安全なデータ取引を可能にする.

以下では, 両フェーズの詳細な処理について説明する.

5.1　登録フェーズ

登録フェーズのフローチャートを図 2 に示す. 本フェーズは DID の生成・登録,IoT データの保存,CID の登録, および発行者による VC 発行, ユーザ A による VC への署名までの流れで構成される.

5.1.1　DID の生成と DID Document の登録

まず, ユーザ A および発行者はそれぞれ DID を生成し, それに対応する DID Document を作成する. 作成した DID Document はブロックチェーンへ登録される. DID Document のブロックチェーンへの登録手順はスマートコントラクト `registerDIDDocument()`(algorithm 1) で規定される.

本研究では,DID の制御主体として Ethereum アドレスを用いる設計を採用している. Ethereum アドレスは ECDSA(secp256k1) 鍵ペアに基づいて生成されるため, 当該アドレスで署名可能であることは対応する秘密鍵を保持していることを意味する. この対応関係をブロックチェーン上に登録することで, 後続処理における署名検証の基盤を構築する.

以下にユーザ A が生成した DID Document の一例を示す. 本 DID Document は最小構成とし,DID とその制御主体である Ethereum アドレスのみを記載している.

```
{
  "id": "did:example:userA",
  "controller": "0x2c854F81C990fDD856fC360f17Dc592366711f08"
}
```

5.1.2　IoT データの保存と CID の取得

次に, ユーザ A は自身が保有する IoT データを IPFS に保存する. この処理により,IoT データはブロックチェーンとは独立した分散ストレージ上に格納され, データ本体を直接ブロックチェーンに保存する必要がなくなる.

IPFS への保存が完了すると, 保存されたデータに対応する CID が取得される. 本研究では, この CID を IoT データを一意に識別する参照情報として扱い, 後続の処理においてユーザ A の DID と組み合わせてブロックチェーンへ登録する.

5.1.3　CID と DID のブロックチェーンへの記録

ユーザ A は取得した CID と自身の DID をブロックチェーンへ登録する. CID と DID のブロックチェーンへの登録はスマートコントラクト `registerIoTData()`(algorithm 2) で規定される. これにより,「どの DID がどの IoT データの所有者であるか」が改ざん耐性を持って記録され, 第三者は所有者を検証可能となる.

5.1.4　発行者によるデータ真正性の確認と VC 発行

発行者は, ブロックチェーン上の DID と CID の整合性を確認し, ユーザ A が登録したデータが真正であるかを検証する. 正当性が確認された場合, 企業はユーザ A に対して VC を発行する. VC には発行者の DID による署名が付与され, 内容の真正性が保証される.

5.1.5　ユーザ A による VC への追加署名

最後に, ユーザ A は自身の DID に紐づく秘密鍵を用いて VC に追加署名を行う. これにより, 企業とユーザ A の双方が署名した VC が完成し, 検証フェーズにおいて提示・検証可能な証明情報となる.

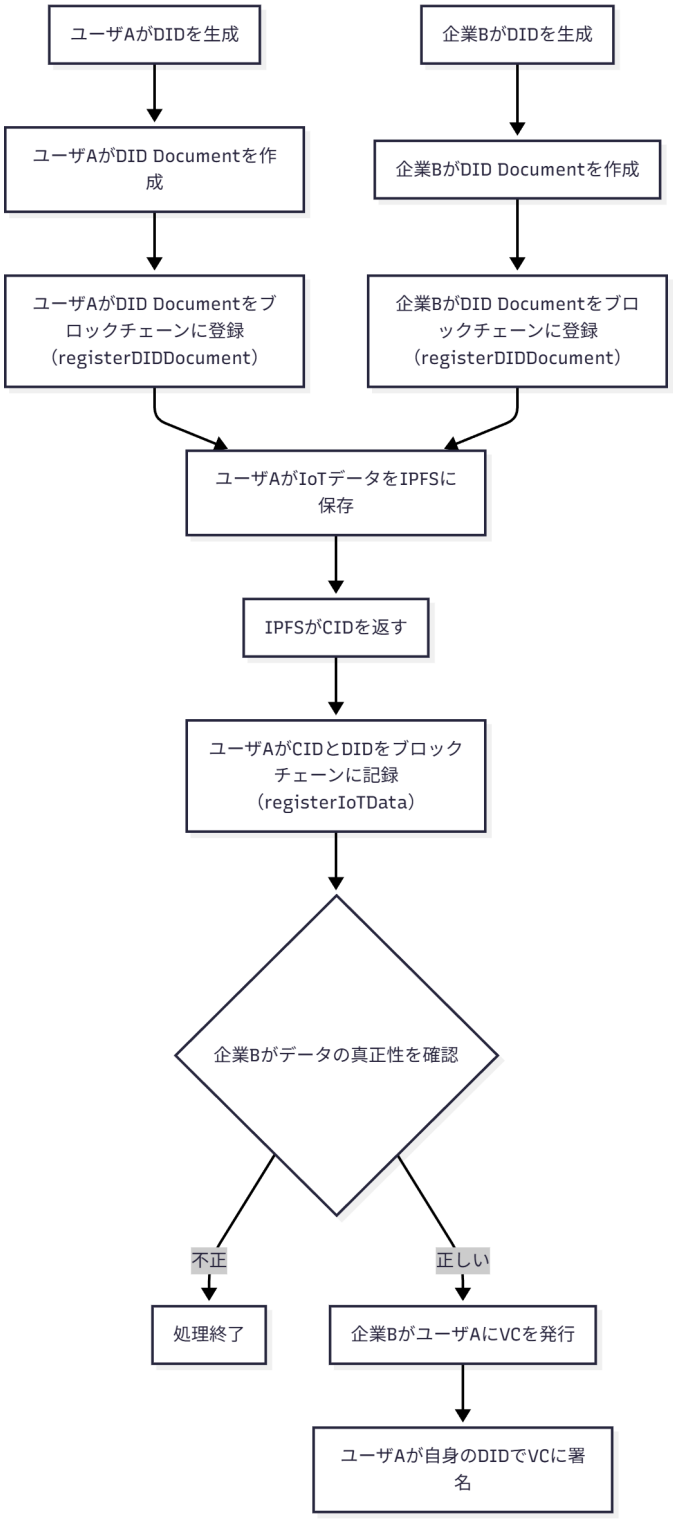

図 2　登録フェーズのフローチャート

5.2 検証フェーズ

検証フェーズのフローチャートを図 3 に示す．本フェーズでは, ユーザ A から提示された VC に含まれる署名を段階的に検証することにより, IoT データが正当なデバイスによって生成されたものであること, およびユーザ A が当該データの正当な保有者であることを確認する．

5.2.1 VC の提示と受領

ユーザ A は完成した VC を検証者へ提示する．検証者は VC を受け取り, 内容に含まれる署名情報と DID 情報を確認する．

5.2.2 企業の署名検証

検証者はまず VC に含まれる発行者の署名を検証する．署名の公開鍵は, ブロックチェーンへ登録された企業の DID Document から取得される．署名が不一致であった場合,VC は不正と判断され処理を終了する．

5.2.3 ユーザ A の署名検証

企業の署名が正当であった場合, 次にユーザ A の署名が検証される．ユーザ A の公開鍵は同様に DID Document から取得され, 署 ‘名が一致する場合, ユーザ A が IoT データの正当な保有者であることが確認される．不一致の場合, 処理は終了する．

5.2.4 データの取引の実行

企業およびユーザ A の署名がいずれも正当であれば, 検証者はユーザ A が真正なデータ提供者であると判断できる．この確認を基に, 検証者はユーザ A と安全にデータ取引を実行する．

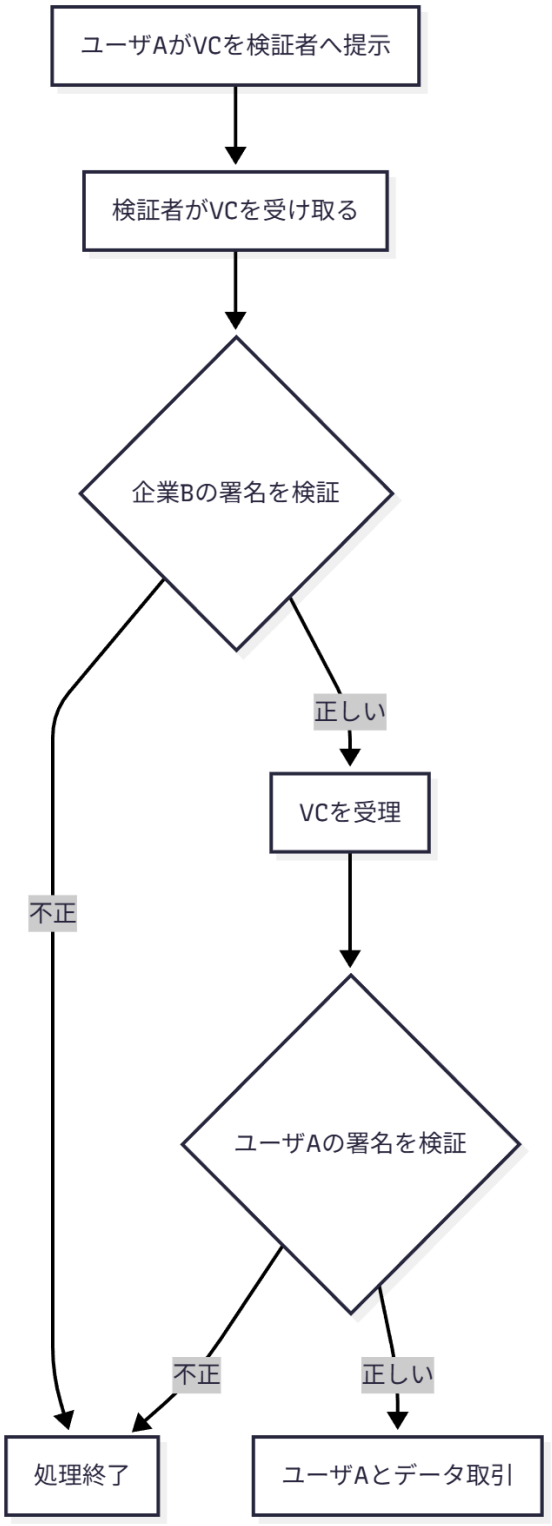

図 3　検証フェーズのフローチャート

5.3 スマートコントラクトによる登録処理

本研究で使用したスマートコントラクトでは, ユーザの DID Document および IoT データ（CID）をブロックチェーンに記録するために `registerDIDDocument` および `registerIoTData` の 2 つの関数を提供している．それぞれの処理内容を擬似コードとして以下に示す.

Algorithm 1 DID Document の登録処理（registerDIDDocument）

Require: DID, DID Document（JSON 形式）

1: 呼び出し元アドレスを取得する（これをユーザ識別子として扱う）
2: DID Document を以下の形式で保存する:
3: `records[呼び出し元アドレス].append（（DID, DID_Document））`
4: DID 登録イベントを発行する

Algorithm 2 IoT データ（CID）の登録処理（registerIoTData）

Require: DID, CID（IPFS で取得したハッシュ値）

1: 呼び出し元アドレスを取得する
2: IoT データを以下の形式で保存する:
3: `IotData[呼び出し元アドレス].append（（DID, CID））`
4: IoT データ登録イベントを発行する

6　実験と考察

本章では, 本研究で構築した DID・VC・IPFS・Ethereum を用いた分散型データ管理システムについて, 機能動作試験および性能評価を通してその有効性を検証する.

6.1　実験目的

本研究の実験目的は, 大きく 2 つである. 1 つ目は提案システムが問題なく動作することを検証することである. 具体的には,DID の生成から DID Document の登録,IoT データの IPFS への保存, 取得した CID のブロックチェーンへの保存, 発行者による VC の発行, および検証者による VC の検証に至るまでの一連の処理を実装し, それらが想定通り正しく実行されることを確認する. これにより,DID・VC・IPFS・ブロックチェーンを統合した提案システムが, データとその所有者の正当性を一貫して保証できることを検証する.

2 つ目は, 提案システムの性能を定量的に評価し,DID および VC を統合したことがシステム全体の性能に与える影響を明らかにすることである. 特に,IPFS およびブロックチェーンを用いた IoT データ管理に関する先行研究が多数存在することを踏まえ, これらを基準的な処理として位置づけたうえで, DID および VC に関連する処理が新たな性能上のボトルネックとなり得るかどうかに着目して評価を行う.

6.2　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

6.3　実験方法

本節では, 本研究で構築したシステムに対して実施した実験方法を述べる. 実験は大きく次の 2 段階に分けて行った.

1. **機能動作試験** DID Document の登録,IoT データの IPFS への保存,CID のブロックチェーンへの保存, 発行者による VC 発行, 検証者による VC 検証という一連の処理が正しく実行されることを確認した.
2. **性能評価**システム内の主要な処理について処理時間を計測し, 応答時間を評価した. 測定対象は以下のとおりである.
   - IoT データの IPFS アップロード時間
   - ブロックチェーンへの CID 登録時間
   - VC 発行処理時間
   - VC 検証処理時間

なお, 本研究における「データ 1 件」とは, センサから取得された 1 時点分の IoT データを表す JSON オブジェクト 1 つ分を指す. 具体的には, タイムスタンプ, 温度, 湿度, 照度を含む以下の形式の JSON データ 1 つを 1 件とする擬似データを使用している.

```
{
  "timestamp": "2025-12-14 18:32:58",
```

```
  "temperature": 21.65,
  "humidity": 63.41,
  "light": 926
}
```

したがって, データ数 10 件,100 件,1 000 件とは, 上記形式の JSON データをそれぞれ 10 個,100 個,1 000 個まとめたファイルを対象として測定を行ったことを意味する.

6.4 実験結果

6.4.1 機能動作試験の結果

本研究で構築したシステムが, 設計時に想定した一連の処理を正しく実行できるかを確認するため, 各処理を順に実行し, コマンドプロンプト上の出力結果をもとに機能動作試験を行った.

まず, 図 4 に示すように, ユーザおよび発行者がそれぞれ DID を生成し, 対応する DID Document を作成した後, スマートコントラクトを通じてブロックチェーンへ登録する処理を実行した. 出力結果から,DID および DID Document を正常が正常に生成され, トランザクションが成功していることを確認できる.

次に, 図 5 に示すように,IoT データを IPFS へアップロードする処理を実行した. コマンドプロンプトには, アップロード対象ファイルに対応する CID が表示されており, データが IPFS 上に正常に保存されたことが確認できる.

続いて, 図 6 では,IPFS 上に保存されたデータの CID と, そのデータの所有者を示す DID をブロックチェーンへ登録する処理を示している. 出力結果から,CID および DID が対応付けられた形でスマートコントラクトに記録され, 登録処理が正常に完了していることが確認できる.

図 7 は, 発行者がユーザに対して VC を発行する処理の実行結果を示している. コマンドプロンプトには,VC の生成および発行処理が正常に完了した旨が出力されており, 発行者による VC 発行が正しく行われたことが確認できる.

次に, 図 8 に示すように, ユーザ A が発行された VC に対して署名を行う処理を実行した. 出力結果から, ユーザ A の秘密鍵を用いた署名処理が成功していることが確認でき,VC がユーザ本人によって承認された状態となっていることが分かる.

最後に, 図 9 では, 検証者が VC の検証処理を実行した結果を示している. コマンドプロンプトの出力から,VC の署名検証および DID Document との照合が成功し, VC の正当性が確認されたことが分かる.

以上の結果より,DID の生成・登録,IPFS へのデータ保存,CID と DID のブロックチェーン登録, VC の発行・署名・検証という一連の処理が, 提案システムにおいて問題なく実行できることを確認した.

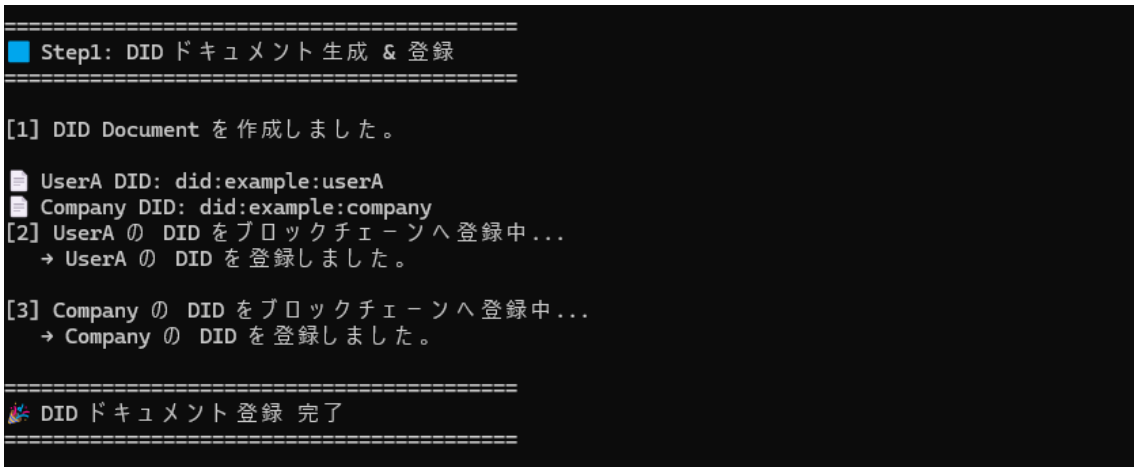

図 4　DID および DID Document の生成・登録処理の実行結果

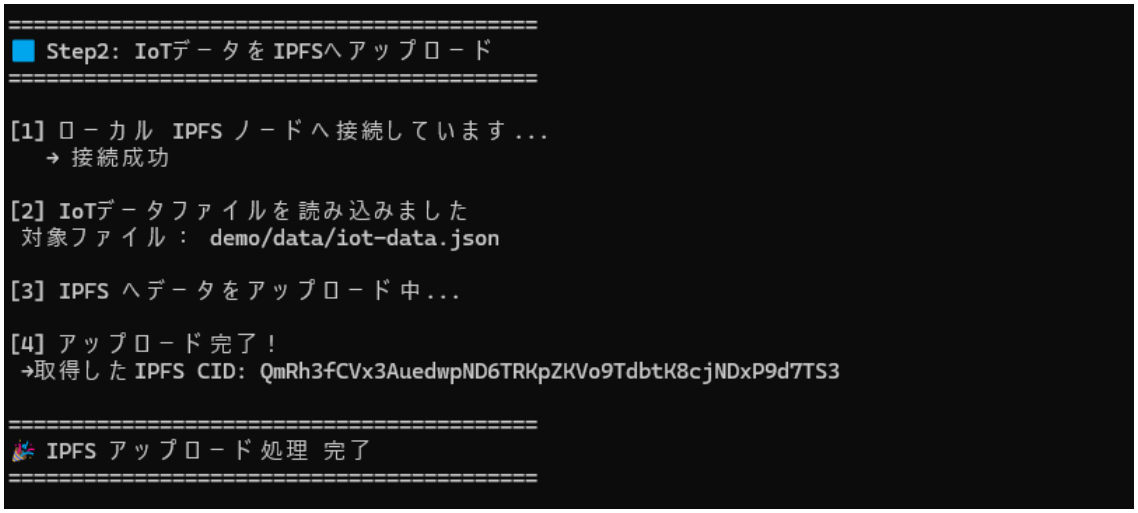

図 5　IPFS へのデータアップロードの実行結果

図 6　CID および DID のブロックチェーンへの登録処理

```
==========================================
🟦 Step4: 企業が IoT データの真正性を保証する VC を発行
==========================================

[1] Company の Ethereum アドレス:
   → 0x041653bCaB0e61d24236212DdcECc8F9b9E4745A

[2] VC 生成に使用する情報:
   CID         → QmRh3fCVx3AuedwpND6TRKpZKVo9TdbtK8cjNDxP9d7TS3
   User DID    → did:example:userA
   Company DID→ did:example:company

[3] VC オブジェクトを生成中...

[4] 発行者が秘密鍵による VC署名処理を実行中...

[5] VC の発行が完了しました！
==========================================
Step4 完了: VC 発行処理が正常に終了しました
==========================================
```

図 7　発行者が VC を発行する処理

```
==========================================
🟦 Step5: UserA による VC 署名
==========================================

[1] UserA の Ethereum アドレス:
   → 0x2c854F81C990fDD856fC360f17Dc592366711f08

[2] 企業が発行した VC を読み込み中...

🔍 読み込んだ VC の概要:
   Issuer DID   → did:example:company
   Subject DID  → did:example:userA
   CID          → QmRh3fCVx3AuedwpND6TRKpZKVo9TdbtK8cjNDxP9d7TS3

[3] UserAによる VC署名処理を実行中...

[4] UserA による署名が完了しました。
   以下が UserA の署名情報です：

===== 🔍 UserA の署名情報 =====
{
  "type": "EcdsaSecp256k1",
  "created": "2025-12-12T07:49:43.109Z",
  "verificationMethod": "did:example:userA#key-1",
  "hash": "0x9911a002516a2b7e7009192b89c07131fdc5561d3c4fee8677f2929d07d61268",
  "signature": "0x73ff394e52afac70b0f57c99c019bdc9ebf05039175574313a3aafb8fa895e770c1b2ec394fb710fc352
3f420b3ec9f0e00909b3eeb3b6e695af447bf730d3431c"
}
================================

==========================================
Step5 完了: UserA による VC 署名処理が正常に終了しました
==========================================
```

図 8　UserA が VC に署名する処理

```
====================================
Step6: Verifiable Credential(VC)の検証
====================================

[1] 検証対象の VC:
{
  "id": "vc:device-auth:userA",
  "issuer": "did:example:company",
  "subject": "did:example:userA",
  "claim": {
    "cid": "QmRh3fCVx3AuedwpND6TRKpZKVo9TdbtK8cjNDxP9d7TS3",
    "verifiedByDevice": "company-original"
  },
  "proof": {
    "type": "EcdsaSecp256k1",
    "created": "2025-12-12T06:56:57.139Z",
    "verificationMethod": "did:example:company#key-1",
    "hash": "0x01d28843637f345e1dc64a46a2fb298a5113d2114923dc04e9e690ca595a6709",
    "signature": "0x1ae86371302c24991e4d86938fbe18c18f091a0b89db56af1ecd82c22a6fd3920a9a86358e571a7471
3fadd767835bd80d2ecef4b782c366cfb30837a7f8ddad1c"
  },
  "userProof": {
    "type": "EcdsaSecp256k1",
    "created": "2025-12-12T07:49:43.109Z",
    "verificationMethod": "did:example:userA#key-1",
    "hash": "0x9911a002516a2b7e7009192b89c07131fdc5561d3c4fee8677f2929d07d61268",
    "signature": "0x73ff394e52afac70b0f57c99c019bdc9ebf05039175574313a3aafb8fa895e770c1b2ec394fb710fc3
523f420b3ec9f0e00909b3eeb3b6e695af447bf730d3431c"
  }
}

[2] Issuer の DID Document を検索...
   ▶ 所有者アドレス: 0x041653bCaB0e61d24236212DdcECc8F9b9E4745A
   ▶ DID Document: {
  '0': 'did:example:company',
  '1': '{"id":"did:example:company","controller":"0x041653bCaB0e61d24236212DdcECc8F9b9E4745A"}',
  __length__: 2,
  did: 'did:example:company',
  doc: '{"id":"did:example:company","controller":"0x041653bCaB0e61d24236212DdcECc8F9b9E4745A"}'
}

[3] Subject(UserA) の DID Document を検索...
   ▶ 所有者アドレス: 0x2c854F81C990fDD856fC360f17Dc592366711f08
   ▶ DID Document: {
  '0': 'did:example:userA',
  '1': '{"id":"did:example:userA","controller":"0x2c854F81C990fDD856fC360f17Dc592366711f08"}',
  __length__: 2,
  did: 'did:example:userA',
  doc: '{"id":"did:example:userA","controller":"0x2c854F81C990fDD856fC360f17Dc592366711f08"}'
}

[4] IoTデータの記録を検索...
   ▶ DID: did:example:userA
   ▶ CID: QmRh3fCVx3AuedwpND6TRKpZKVo9TdbtK8cjNDxP9d7TS3

[5] Issuer の署名を検証中...
   ▶ recover結果: 0x041653bCaB0e61d24236212DdcECc8F9b9E4745A
   ▶ 登録Issuerアドレス      : 0x041653bCaB0e61d24236212DdcECc8F9b9E4745A
   ✅ Issuer の署名は正しい

[6] UserA の署名を検証中...
   ▶ recover結果: 0x2c854F81C990fDD856fC360f17Dc592366711f08
   ▶ UserAアドレス 0x2c854F81C990fDD856fC360f17Dc592366711f08
   ✅ UserA の署名は正しい

================================
Step6 完了: VC検証処理が正常に終了しました
================================
```

図 9　検証者が VC を検証する処理

6.4.2 性能評価の結果

本研究では, 提案システムの実運用を想定した際に,DID および VC を統合したことが性能上のボトルネックとなり得るかを明らかにすることを目的として, 各処理に要する時間の計測を行った.

性能測定は, 各処理は 20 回繰り返し実行した際の平均処理時間を算出することで行った. まず,IPFS への IoT データアップロード処理の性能を評価した. その結果, データ数 10 件（931bytes）から 1 000 件（96 793bytes）までのいずれの場合においても, 平均処理時間は約 19 21ms で推移しており, データサイズの増加に対して大きな遅延は確認されなかった. また, スループットはデータ数の増加に伴い向上しており,IPFS が IoT データの集約保存に対して十分な性能を有することが確認できた.

次に, ブロックチェーンへの CID 登録処理に要する時間を計測した. 本処理では, データ内容に依存しない条件下において, 平均処理時間は 63.60ms であり, トランザクション処理として一定の時間を要するものの, 安定した性能を示した.

これらの処理と比較し検討するため,VC の発行処理および検証処理に要する時間を計測した.

まず,VC の発行に要する時間を計測した結果, 平均処理時間は 10.45ms であり, IPFS アップロード処理およびブロックチェーン登録処理と比較しても短い処理時間であった. このことから,VC の発行処理がシステム全体の性能に与える影響は小さく, 性能上のボトルネックとなる可能性は低いと考えられる.

一方,VC の検証処理に要する平均時間は 155.78ms であり他の処理と比較して相対的に長い処理時間を要する結果となった.

以上の性能評価結果を表 1 にまとめる.

表 1　各処理における性能評価結果

処理内容	条件	平均処理時間
IPFS アップロード	10 件（931bytes）	19.11ms
IPFS アップロード	100 件（9 279bytes）	20.78ms
IPFS アップロード	1 000 件（96 793bytes）	21.20ms
ブロックチェーン登録	—	63.60ms
VC 発行	—	10.45ms
VC 検証	—	155.78ms

6.5 考察

本章では, 前章で示した実験結果を踏まえ, 提案システムにおいて DID および VC を統合したことがシステム全体の性能に与える影響について考察する.

6.5.1 DID/VC が性能に与える影響

まず,VC の発行処理に着目する. 性能評価の結果より,IPFS への IoT データアップロード時間は約 19 21ms で推移し, ブロックチェーンへの CID 登録処理は平均 63.60ms を要することが確認されている. これらの処理と比較すると,VC の発行処理に要する平均時間は 10.45ms と短く, システム全体の処理性能に与える影響は小さい. このことから,VC の発行処理が提案システムの動作において性能上のボトルネックとなる可能性は低いと考えられる.

次に,VC の検証処理に着目する. VC の検証にかかる処理は 155.78ms であり,IPFS へのアップロード処理やブロックチェーンへの CID 登録処理と比較して, 相対的に長い処理時間を要する結果となった.

しかしながら,VC の検証処理は,IoT データの保存や更新といった高頻度に実行される処理とは異なりデータの取引時といった特定のタイミングでの実行を想定している. そのため, 通常の運用形態においてはシステム全体の性能に与える影響は限定的であると考えられる一方, 検証処理の実行頻度が高くなる場合には, 性能上のボトルネックとなる可能性も否定できない.

6.5.2　今後の課題

本稿で提案したシステムは, 中央集権的な管理主体に依存せず, 分散的に IoT データの管理を行うことを目的として設計した. しかしながら, ユーザが保持するデータの正当性を保証する点においては,VC の発行主体として企業などの組織に依存せざるを得ず, この点は部分的に中央集権的な管理構造となっている.

したがって, 完全に分散化されたデータ管理システムの実現という観点からは, データの正当性保証を特定の組織に依存しない仕組みについてさらなる検討が必要であると考えられる.

7　まとめ

ここはまとめの章です.

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.