データ提供時の不安に注目した非中央集権型データ連携フレームワークの形式検証

徳田 祥太[1,a)]　掛井 将平[1]　白石 善明[2]　齋藤 彰一[1]

概要：機密性の高いパーソナルデータの活用において，データ利用者による提供データの不正利用や不正取得の防止が重要である．そこで我々は先行研究として，データ利用者の機密実行環境で提供データを利用できる非中央集権型データ連携フレームワークを提案している．この研究では，データ提供者が同意するデータ利用条件を Intel SGX でデータ連携アプリに強制することで提供データの不正利用や不正取得ができないような設計としていたが，その安全性評価は机上での議論に留まっていた．本稿では，提供データの不正利用と不正取得の観点から，本フレームワークの安全性の形式検証を行う．本検証では ProVerif を用いて，秘匿性や認証に関する安全性要件に加え，データ利用制御の実施に関する安全性要件も検証対象とし，データ提供者が同意したデータ利用条件が順守されるかについても確認する．また，ProVerif が検出した攻撃に対する考察を行い，対策について議論する．

キーワード：形式検証，ProVerif，データ活用，データ利用制御，Trusted Execution Environment

Formal Verification of Decentralized Data Processing Framework Focused on Concerns with Data Provision

Shota Tokuda[1,a)]　Shohei Kakei[1]　Yoshiaki Shiraishi[2]　Shoichi Saito[1]

Abstract: We proposed a decentralized data processing framework that enables the use of provided data in the confidential execution environment of data consumers in our previous work. This framework was designed to prevent unauthorized use and acquisition of provided data by using Intel SGX to enforce data usage conditions agreed to by data providers to data processing apps. However, the security evaluation of this framework was only a theoretical discussion. In this paper, we use the formal verification tool ProVerif and present the security of this framework in terms of unauthorized use and unauthorized acquisition of provided data. In addition to the security requirements for secrecy and authentication, the security requirements for the enforcement of data usage control are also verified. Furthermore, this paper presents a discussion of attack methods identified from the verification results, along with the an analysis of countermeasures against these attacks.

Keywords: formal verification, ProVerif, data utilization, data usage control, trusted execution environment

1. はじめに

Internet of Things（IoT）技術やクラウド技術の発展により，様々なデータが収集，活用されることが期待されている．例えば，IoT デバイスから取得した施設情報や個人情報を用いて火災発生時における迅速な救命活動を実施できる [1]．この例では火災発生時に IoT センサから取得される建物の状況を取得し，避難プロセスの改善に役立てる．これらの情報に加え，建物内の被災者の位置情報や車いすを利用しているなどの属性情報を利用し，どの場所でどのような人が被災しているのかといった情報を消防士が取得することで，より正確な救命活動が可能となる．

[1] 名古屋工業大学 Nagoya Institute of Technology
[2] 神戸大学 Kobe University
[a)] s.tokuda.570@nitech.jp

データ提供者が提供データをデータ利用者と直接共有するような分散型でのデータ利用では，データ提供者による自身のデータの制御が困難となる．そのため，データ提供者がデータ利用者を信頼できずデータ主権が確保できない場合には，データ提供者が抱くプライバシに関する懸念からデータ共有が消極的となり，データ利用者が提供するサービスの質が低下する [2,3]．このような懸念に対処するため，我々はこれまでに Trusted Execution Environment（TEE）を活用することでデータ提供者が同意しないデータ開示を防止可能な非中央集権型データ連携フレームワークを提案している [4,5]．本フレームワークは TEE を用いることで一連のデータフローにおける提供データの機密性を確保する．また，日時や場所などのデータ提供者の同意したデータ利用条件に基づくデータ利用を実現するデータ利用制御技術を実装している．我々はこれまでに本フレームワークが期待するセキュリティ特性である提供データの不正取得と不正利用への耐性を満たすことを簡単な安全性評価を通して確認している．一方で，本フレームワークを構成するプロトコルの複雑さから網羅的な検証には至っていなかった．そのため本稿では，本フレームワークにおける提供データの不正取得や不正利用に関する安全性要件に対して ProVerif を用いた形式検証を行い，安全性を評価する．また，検証結果から本フレームワークで生じる攻撃に関して考察し，この攻撃に対する対策について議論する．

2. 関連技術

2.1 Intel SGX

TEE はカーネルやハイパーバイザが信頼できない環境で処理の完全性やデータの機密性を確保しつつ処理を行うための保護された隔離実行環境である．本研究では TEE として Intel Software Guard Extensions（SGX）[6] を用いる．SGX はメモリ上に *Enclave* と呼ばれる暗号的に保護された領域を生成し，その領域内で処理を行うことでデータの機密性を確保する．Enclave 内で動作するプログラムは *MRENCLAVE* と呼ばれるハッシュ値によって一意に識別可能である．SGX のシーリング機能では，データの機密性を確保しつつ不揮発性メモリにデータを保存できる [7]．

Remote Attestation はリモートで動作する Enclave プログラムやプラットフォームの正当性と完全性を検証するために利用される．Remote Attestation を用いることで検証者は通信相手の Enclave プログラムの MRENCLAVE を取得し，意図したプログラムがリモート環境で動作しているかを検証できる．RA-TLS [8] は TLS 証明書に Remote Attestation に用いる情報を埋め込む手法である．

2.2 データ利用制御

IT アーキテクチャにおいて個人や組織が自身でデータの使用を制御する能力を表すデータ主権という概念は，プライバシの文脈でしばしば言及される [9]．データ主権を実現するためにデータ利用制御（Data Usage Control）の利用が注目されている [3]．データ利用制御は RBAC（Role-based Access Control）や ABAC（Attribute-based Access Control）などの従来のアクセス制御を拡張した概念で，アクセス時における認証や認可に加えてリソースが提供者から離れた後の制御も可能とし，義務[*1]や条件[*2]の順守を強制する [10]．データ利用制御を用いることでデータライフサイクル全体でデータ提供者の意図した利用が行われることが保証される．データ提供後における利用者側でのデータの継続的な制御の実施にはハードウェアベースのセキュリティ対策が重要な役割を果たしており，その一例としては TEE の利用がある [11]．

2.3 ProVerif

複雑化する暗号プロトコルの安全性の証明に形式検証ツールが活用されている [12]．ProVerif [13] は Dolev-Yao モデル [14] と呼ばれる形式モデルに基づく自動暗号プロトコル検証ツールである．ProVerif を用いることで暗号プロトコルにおける秘匿性や認証，強秘匿性，等価性に関する性質を証明できる．ProVerif は共通鍵暗号や公開鍵暗号，ハッシュ関数などの暗号プリミティブに対応している．

ProVerif では証明したい安全性要件をクエリとして表現する．秘匿性に関するクエリは以下のように記述される．

```
query attacker(d).
```

このクエリでは攻撃者が機密データ d にアクセスできるかについて検証する．検証結果が True の場合には攻撃者が機密データにアクセスできることを表し，False の場合は攻撃が発見されないため機密データの秘匿性が示される．

認証に関する安全性要件の検証にはイベントを用いる．イベントはプロセス中で呼び出され，検証対象のプロトコルがある処理に到達したタイミングとデータを記録するために利用される．これにより，認証を行ったタイミングやデータを送信したタイミング，認証に用いられた認証情報を識別できる．イベントを用いた安全性検証では一般的にイベント間の関係が用いられ，「あるイベント e が実行された場合にそれ以前にイベント e' が実行された」という形式で表現される．また，イベントに引数を用いることで，イベントの引数間の関係もクエリで検証できる．例えば，Alice の Bob に対する認証が適切に実行されるかについて検証する場合には，まずイベント end_auth(i) を Alice のプロセスが終了するタイミングに記述し，イベント begin_auth(i) を Bob のプロセスが開始するタイミングに記述する．各イベントの引数 i は ID などの認証に用いられる情報である．このとき，クエリは以下のように記述される．

[*1] データの利用を許可するために満たさなければならない要件
[*2] 主体から独立した環境要件やシステム要件

```
query i: bitstring;
  event(end_auth(i)) ==> event(begin_auth(i)).
```

このクエリではイベント end_auth(i) が発生した場合にイベント begin_auth(i) が発生しているかを検証する．検証結果が True である場合には，適切に認証されることが証明される．一方で，検証結果が False の場合には攻撃が発見され，BoB のなりすましが生じうることが判明する．

3. 関連研究

TEE や SGX を用いた暗号プロトコルの安全性評価に形式検証を利用できる [15, 16]．Alansari [15] はブロックチェーン技術と SGX を用い，属性ベースのアクセス制御ポリシーによるデータ共有フレームワークを提案した．安全性評価において，提案プロトコルが認証やデータの秘匿性に対する攻撃に耐性があることを ProVerif で証明している．Arshad ら [16] は属性ベース暗号を拡張し，アクセス制御を拡張した義務を実施可能とした OB-ABE を提案した．OB-ABE では SGX を用いてデータ所有者が指定した義務をデータ利用者側で強制する．この強制可能な義務に関して ProVerif を用いた形式検証を行い，適切に実施されることを示した．本研究でもこれらの研究と同様に提案フレームワークをモデル化し，ProVerif での形式検証によって安全性を検証する．

4. 提案フレームワーク

4.1 概要

提案フレームワークの概要を**図 1** で示す．本研究で対象とするデータ連携はデータ提供者とデータ利用者が直接データをやり取りをする非中央集権型の方式である．提案フレームワークでは TEE を用いて Enclave 内でデータ連携アプリを実行し，データ処理中の提供データの機密性を確保する．データ提供時には TLS に基づいた Remote Attestation である RA-TLS を用いてデータ提供者がデータ利用者とデータ連携アプリを認証する．また，TEE のシーリングを用いてデータ利用者側で提供データの一時保存を行う．本フレームワークのシーリングはデータ連携アプリに固有の ID である MRENCLAVE に基づいて行うことで，データ提供時にデータ利用を認可したデータ連携アプリでのみ提供データが利用可能となる．

提案フレームワークでは分散台帳技術を用いて MRENCLAVE とデータ連携アプリの解析情報を紐づけて記録する．データ提供者は Remote Attestation で取得した通信相手の MRENCLAVE から分散台帳に記録されたデータ連携アプリの解析情報を取得する．分散台帳へのデータ連携アプリの登録にはスマートコントラクトを用いることで，登録処理の完全性を保証する．登録処理では，まずアプリのソースコードに対応する MRENCLAVE を取得する．また，データ利用制御の実装やデータ処理における提供データの開示の有無を検証する．これらの情報を分散台帳に記録し，データ提供者は登録されたアプリの解析情報を元にデータの提供判断を行う．例えば，データ提供者が秘匿にしたい提供データが処理結果として開示されることをアプリの解析情報から確認した場合には，データ提供者はデータ提供を拒否する判断できる．スマートコントラクトによるアプリ登録処理と TEE アプリケーションの完全性により，分散台帳に記録されたデータ連携アプリの解析情報と実際にデータ利用者が実行するアプリの動作は一致することが保証される．

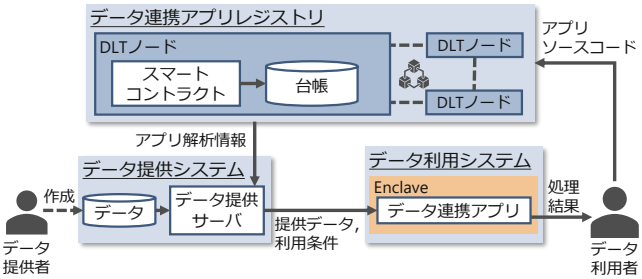

図 1 提案フレームワーク概要

提案フレームワークで用いられる提供データは個人情報を想定するため，データ連携アプリのデータ処理では基本的に提供データが開示されないことを想定するが，データ提供者のデータ提供ポリシー次第では提供データを開示するデータ処理を行うデータ連携アプリへのデータ提供も可能である．このように提案フレームワークは個人情報などの機密情報から交通情報などの公開情報まで様々なデータを処理できる．本稿では，提供データとして個人情報が利用され，提供データを開示しないデータ処理を行うデータ連携アプリを想定する．

提案フレームワークにおける一連の流れは 3 つのフェーズから構成される．データ連携アプリ登録フェーズでは，データ利用者が実装したデータ連携アプリをスマートコントラクトで検証し，分散台帳へアプリの情報を登録する．データ利用申請フェーズでは，データ利用者がデータ提供者に対して利用したいデータや処理を行うアプリの情報などを提示する．データ処理フェーズでは，データ連携アプリが提供データを取得してデータ処理を行う．次節では本稿でモデル化の対象となるデータ処理フェーズについて説明する．

4.2 データ処理フェーズ

データ処理フェーズでは，データ連携アプリがデータ提供サーバに対してデータ要求を行う．この要求の際に Remote Attestation が行われ，データ提供サーバは通信相手のアプリの ID である MRENCLAVE を取得する．データ提供サーバは取得した MRENCLAVE を用いてデータ

利用申請フェーズでデータ利用者から申請された情報を取得する．この情報からデータ利用条件を算出し，データとデータ利用条件をデータ連携アプリに返す．データ連携アプリは提示されたデータ利用条件を確認し，データ処理を行いデータ利用者に処理結果を応答する．提案フレームワークでは提供データの一時保存を行うが，提供データの機密性の維持のために提供データとデータ利用条件を紐づけて MRENCLAVE に基づいたシーリングを行う．一時保存データを用いる際には，シーリング済みデータに対してアンシーリングを行い，データ処理を行う．

5. 問題設定

5.1 前提

データ提供者は TEE を完全に信頼し，Remote Attestation で利用する Intel のサーバを信頼する．また，本研究では TEE に対するサイドチャネル攻撃は考慮しない．認証局（Certificate Authority，CA）は十分なセキュリティ対策が行われており，データ提供者は CA を信頼する．

5.2 検証内容と安全性要件

提案フレームワークはデータ提供者の提供データをデータ利用者に渡し，データ利用者の計算リソースを使用してデータを処理する．提供されたデータが不正に取得されたり，不正に利用されたりしないかといったデータ提供者の不安を鑑みると，以下の性質を満たすことが望ましい．

満たすべき性質：提供データはデータ提供者が指定する条件のもとで認可されたデータ連携アプリのみが処理でき，データ利用者はその処理結果のみを取得できる．

提案フレームワークが上記の性質を満たしているかを検証するため，本研究における形式検証はデータ提供者視点で行う．本検証では提供データの不正利用防止を考慮し，データ利用を認可したデータ利用者やアプリで提供データが利用されるかについて検証する．また，データ連携アプリでデータ提供者が提示したデータ利用条件が順守されるかについて検証する．加えて，提供データの不正取得防止を考慮し，一連の提供データのデータフローにおいて機密性が確保されているかについて検証する．

先述の検証内容より，本研究で検証対象とする安全性要件を定義する．

R1 提供データの秘匿性
R2 データ提供者によるデータ利用者の認証
R3 データ提供者によるデータ連携アプリの認証
R4 データ提供者の算出したデータ利用条件の順守

R1 は本研究で想定する個人情報のような機密性の高いデータを共有する際に考慮すべき点であり，データの不正取得防止のための要件である．R2 から R4 はデータの不正利用防止を考慮した要件であり，これらの要件が満たされることでデータ提供者が同意した通りにデータが利用される．

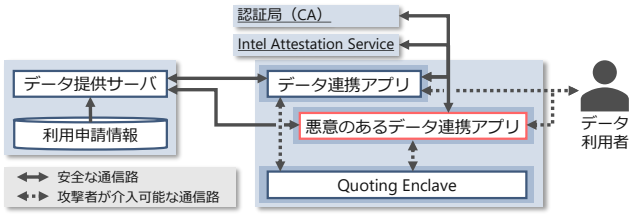

図 2 形式検証に用いる提案フレームワークのモデル

6. モデル化

6.1 提案フレームワークのモデル化

図 2 で形式検証で利用する提案フレームワークのモデルを示す．本検証では，提供データがやり取りされる際の安全性を評価するため，実際にデータ提供やデータ処理が実行されるデータ処理フェーズに注目する．そのため，既にデータ利用者が実装したデータ連携アプリが登録済みであり，そのアプリに対してデータ利用申請が完了していることを想定する．これは，各エンティティのプロセスの実行前にデータ連携アプリのソースコードとデータ利用者の識別名に対応する情報を生成し，これらの情報を用いて利用申請情報を記録することによりモデル化する．また，この記録と同一のタイミングでマシンのコンテキスト情報やデータ提供サーバで利用する証明書などを生成する．

データ提供者側で動作するデータ提供サーバとデータ連携アプリ間とデータ提供者の信頼する IAS や CA とデータ連携アプリ間の通信路は攻撃者によって侵害されないことを想定し，適切にコネクションが管理されている安全な通信路としてモデル化する．具体的には，これらの通信路を Private と明記し，各コネクションを識別する乱数を生成して各通信で相手から提示された値が一致するか確認する．一方で，データ提供者が信頼できないデータ利用者とデータ連携アプリ間の通信路や Local Attestation が行われる Quoting Enclave とデータ連携アプリ間の通信路は攻撃者が介入可能な通信路を想定する．

本検証では，明示的にデータ利用申請済みではない悪意のあるデータ連携アプリのプロセスを記述し，データ提供者による意図しないアプリへのデータ提供の有無を検証する．このアプリはデータ利用申請時にデータ利用者が宣言したデータ連携アプリとは異なるアプリであり，提示された利用条件を無視し，提供データを処理結果としてそのまま開示する処理を行うことを想定する．そのため，このデータ連携アプリではデータ利用条件は確認されず，データ処理は提供データを開示するようにモデル化する．

6.1.1 データ連携アプリの証明書取得

データ連携アプリの証明書取得の流れを**図 3** で示す．本稿のシーケンス図では，各ライフラインの最初で各プロ

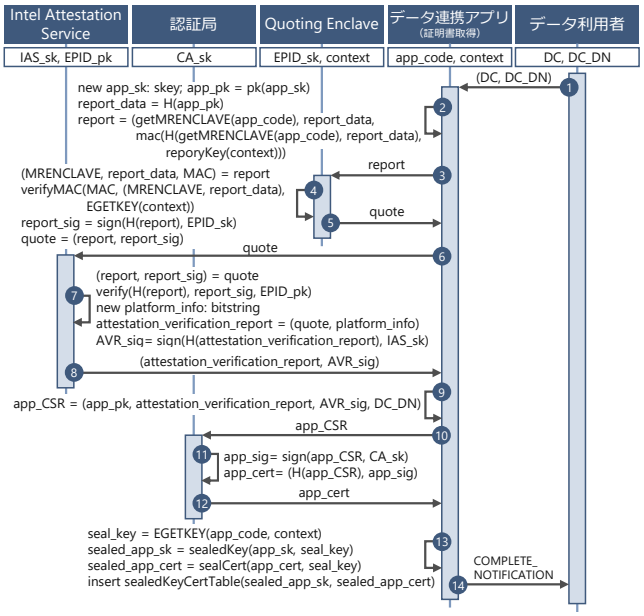

図 3　モデル化された証明書取得の流れ

セスへ入力するデータを示し，各プロセスでの処理は疑似コードで示す．まず，データ利用者はデータ連携アプリの証明書発行処理を実行してデータ要求に利用するクライアント証明書を取得する．データ利用者は証明書発行のためにデータ連携アプリに識別名 DC_DN を入力する（Step 1）．なお，データ連携アプリの実行者をモデル化するために実行者の識別子 DC を DC_DN とともに渡しており，DC は後述するイベントで利用される．次に，データ連携アプリは MRENCLAVE を含む report と呼ばれる構造体を取得し，Quoting Enclave に送信する（Step 2，3）．report は Enclave アプリのソースコード app_code に対応する MRENCLAVE と Enclave 内で生成された公開鍵のハッシュ値とこれらのデータに対する CPU 固有の鍵を用いた MAC（Message Authentication Code）となるようモデル化する．Quoting Enclave では report の MAC の検証によって同一マシン上で report が生成されたことを確認後，EPID 秘密鍵 EPID_sk を用いて report に対して署名し（Step 4），report と署名を quote として出力する（Step 5）．データ連携アプリは取得した quote を外部サーバの Intel Attestation Service（IAS）に送信し（Step 6），quote の署名の検証後に Attestation Verification Report を作成し出力する（Step 7，8）．データ連携アプリは取得した Attestation Verification Report を証明書署名要求 app_CSR に埋め込み（Step 9），CA は署名を行い証明書 app_cert を返す（Step 10–12）．データ連携アプリは取得した秘密鍵とクライアント証明書に対してシーリングを行い，table に保存する（Step 13）．提案フレームワークでは，Enclave の ID である MRENCLAVE に基づいてシーリングを行う．そのため，シーリング鍵はマシンを識別する情報 context に加えてデータ連携アプリのソースコード

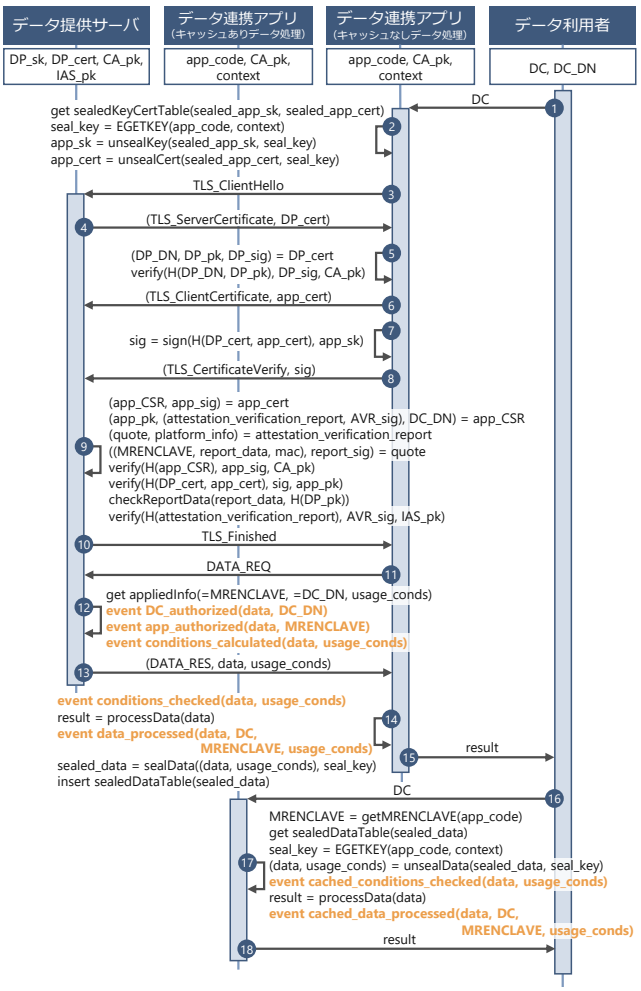

図 4　モデル化されたデータ処理の流れ

app_code から算出するようにモデル化する．最後に，データ利用者へ取得完了が通知される（Step 14）．

6.1.2　データ連携アプリのデータ処理

次にデータ処理の流れを**図 4** で示す．データ利用者はデータ処理を行うデータ連携アプリを実行する（Step 1）．データ連携アプリでは，まず table からシーリングされた秘密鍵と証明書を取得し，アンシーリングを行う（Step 2）．次に，この秘密鍵と証明書を用いてデータ要求をデータ提供サーバに対して行う．このハンドシェイクではデータ連携アプリ側でサーバ証明書を元にデータ提供サーバを認証し（Step 3–5），サーバ側でクライアント証明書を元に通信相手のデータ連携アプリとデータ利用者を認証する（Step 6–9）．Step 7 ではアプリが秘密鍵を保有していることを証明するためにアプリ秘密鍵を用いた署名を行い，CertificateVerify でサーバに署名を送信する．Step 9 ではこの署名も検証し，アプリがクライアント証明書に対応する秘密鍵を保有しているかを確認する．本来の CertificateVerify で送信される署名はそれ以前にやり取りをしたペイロードのハッシュ値に対する署名であるが，本モデル化ではサーバ証明書とクライアント証明書のハッシュ値に対して署名を行う．データ連携アプリの認証では

report_data から Enclave の情報のすり替えがないかを確認し，クライアント証明書に埋め込まれた IAS による Attestation Verification Report の署名を検証する．TLS ハンドシェイク完了後にデータ連携アプリはデータ要求を行い（Step 11），データ提供サーバはデータ利用条件 usage_conds を MRENCLAVE とデータ利用者の識別名 DC_DN から取得する（Step 12）．そして，データ提供サーバはデータとデータ利用条件をデータ連携アプリに送信する（Step 13）．Step 12 で発生するイベント DC_authorized(data, DC_DN) はデータ data に対して識別名 DC_DN に対応するデータ利用者のデータ利用を認可したことを表す．また，イベント app_authorized(data, MRENCLAVE) はデータ data に対して認証したデータ連携アプリ MRENCLAVE のデータ利用を認可したことを表す．Enclave アプリは MRENCLAVE で一意に識別可能であるため，イベント app_authorized の第二引数では MRENCLAVE を用いる．データ連携アプリは提示されたデータ利用条件を確認後にデータ処理を行い，データ連携アプリは取得したデータとデータ利用条件を紐づけてシーリングを行う（Step 14）．イベント conditions_checked(data, usage_conds) はデータ data に対してデータ利用条件 usage_conds を確認したことを表す．データ利用条件の確認後，データ連携アプリはデータ処理を行う．イベント data_processed(data, DC, MRENCLAVE, usage_conds) はデータ data に対してデータ利用者 DC がデータ連携アプリ MRENCLAVE でデータ利用条件 usage_conds に基づいて処理したことを表す．データ連携アプリはデータ利用者に対して処理結果とシーリング済みデータを返す（Step 15）．

一時保存データを利用する際には，データ利用者はデータ連携アプリに対してシーリング済みデータを入力する（Step 16）．一時保存データ利用時にはデータとデータ利用条件に対してアンシーリングを行い，データ利用条件の確認を行う（Step 17）．そしてデータ処理を実行し，処理結果をデータ利用者に返す（Step 18）．一時保存データありのプロセスにもイベント conditions_checked と data_processed と同様にイベント cached_conditions_checked と cached_data_processed を記述する．これにより，一時保存データを利用する場合に実行されるデータ処理を対象とした安全性評価を実施する．

6.2 クエリ

各安全性要件に対応するクエリを**表 1** で示す．R1 は到達可能性，R2 から R4 は対応表明性のクエリを用いてそれぞれの安全性要件を表現する．表 1 は一時保存データを利用しない場合のデータ処理に関するクエリである．提案フレームワークでは一時保存データの有無でデータ連携アプリの動作が異なり，一時保存データが存在しない場合にはデータ連携アプリがデータ提供者に問い合わせてデータを取得する．一方で，一時保存データが存在する場合にはシーリングされたデータとデータ利用条件を用いてデータ処理を行う．一時保存データを用いる場合のデータ処理を対象とした要件の検証を行うためには，イベント data_processed と conditions_checked をそれぞれ cached_data_processed と cached_conditions_checked に置き換える．本節では一時保存データなしの場合のクエリについて説明する．

R1 に対応するクエリは攻撃者が提供データ data にアクセスできるかについて検証するクエリである．R2 は「データ data がデータ利用者 dc によって処理されたならば，それ以前にデータ提供サーバはデータ data の利用をデータ利用者 dc に対して認可した」ということを意味し，データ提供者の同意したデータ利用者によってデータ利用が行われているかについて検証するクエリである．ここで，イベント data_processed と DC_authorized 間でデータ利用者の表現方法が異なっているため，関数 getDN() を用いてデータ利用者を識別名に変換する．この変換により，各イベントの引数に含まれるデータ利用者の情報が同一のデータ利用者を表しているかを検証する．R3 は「データ data がデータ連携アプリ app によって処理されたならば，それ以前にデータ提供者はデータ data の利用をデータ連携アプリ app に対して認可した」ということを意味し，データ提供者の同意したデータ連携アプリでデータ利用が行われているかについて検証するクエリである．R4 は Arshard ら [16] がデータ利用制御の義務の検証に用いたクエリを参考に作成した．このクエリは「データ data がデータ利用条件 conds で処理されたならば，それ以前に条件の確認が同一のデータとデータ利用条件に対して実行され，データ利用条件を確認する前にデータ提供者がデータ data に対してデータ利用条件 conds を算出した」ということを表しており，データ提供者の同意したデータ利用条件に基づいてデータ利用が行われているか検証するクエリである．

7. 安全性評価

7.1 検証結果

ProVerif（バージョン 2.05）での検証結果を**表 2** で示す．一時保存データを利用する場合と利用しない場合で同一の結果が出力され，提供データの秘匿性とデータ提供者によるデータ利用者の認証とデータ提供者の算出したデータ利用条件の順守に対しては True と出力された．この結果より，提案フレームワークがこれらの安全性要件を満たしていることが示された．一方で，データ提供者によるデータ利用者の認証に対しては False と出力され，一時保存データの有無に関わらずデータ提供者の意図しないデータ利用者によるデータ利用が行われるような攻撃手法が存在することが示された．

表 1　安全性要件とクエリ（一時保存データなしの場合）

安全性要件	クエリ
R1: 提供データの秘匿性	query attacker(data).
R2: データ利用者の認証	query data: bitstring, dc: bitstring, app: bitstring, conds: conditions; event(data_processed(data, dc, app, conds)) ==> event(DC_authorized(data, getDN(dc))).
R3: データ連携アプリの認証	query data: bitstring, dc: bitstring, app: bitstring, conds: conditions; event(data_processed(data, dc, app, conds)) ==> event(app_authorized(data, app)).
R4: データ利用条件の順守	query data: bitstring, dc: bitstring, app: bitstring, conds: conditions; event(data_processed(data, dc, app, conds)) ==> (event(conditions_checked(data, conds)) ==> event(conditions_calculated(data, conds))).

表 2　ProVerif での検証結果

安全性要件	検証結果	
	一時保存データなし	一時保存データあり
R1: 提供データの秘匿性	True	
R2: データ利用者の認証	False	False
R3: データ連携アプリの認証	True	True
R4: データ利用条件の順守	True	True

7.2　考察

検証結果より，5.1 節の前提の下で提案フレームワークにおいてデータ提供者がデータ利用者を信頼できない場合にも提供データの機密性は確保されることが示された．同様に，データ提供者が同意した日時や場所，回数に基づいた利用が行われ，提案フレームワークに組み込まれたデータ利用制御は適切に実施されることが示された．

攻撃が発見されたクエリに対して ProVerif は攻撃手法をグラフなどで出力する．本節ではこの出力から確認されたデータ提供者の意図しないデータ利用者によるデータ利用の攻撃手法を説明し，検出された攻撃への対策を講じたモデルを再検証する．

提案フレームワークでは，攻撃者はデータ連携アプリを実行することで正当なデータ利用者に対して発行された秘密鍵とクライアント証明書を利用できる．そのため，データ提供者はデータ連携アプリの実行者を識別できず，攻撃者によるデータ利用の攻撃が生じる．また，一時保存データ利用時に関しても攻撃者がデータ連携アプリを実行することで一時保存データを利用できるため，攻撃者によるデータ利用が生じる．いずれの攻撃も攻撃者がデータ連携アプリを実行できることに起因する．これらの攻撃に対して，アプリの実行時にアプリ側でデータ利用者を認証することで対処できる．この認証で証明書取得時と同一のデータ利用者がアプリを実行しているかを確認することで，不正なデータ利用者によるデータ利用を防止できる．

以上のポイントを踏まえて 6.1 節で示したモデルでデータ利用者を認証できるように修正し，同様の安全性要件に対して形式検証を実施する．モデルの変更点を**図 5** で示す．まずプロセスの実行前のセットアップとして，正当なデータ利用者 DC とデータ利用者の認証に用いる検証情報 DC_verify_info を紐づけて table に記録する．また，

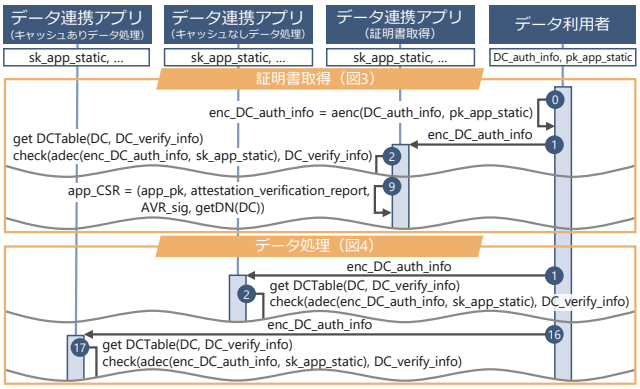

図 5　モデルの変更点（図 3 と図 4 の抜粋）

データ利用者の認証情報 DC_auth_info をデータ利用者とデータ連携アプリ間で安全にやり取りするための秘密鍵 sk_app_static と公開鍵 pk_app_static を事前に生成し，秘密鍵をデータ連携アプリのプロセスに，公開鍵をデータ利用者のプロセスに渡しておく．データ利用者は pk_app_static を用いて認証情報を暗号化し（Step 0），データ連携アプリの実行時にアプリに入力する．データ連携アプリでは暗号化された認証情報を復号し，この情報と table から取得した検証情報を用いてアプリの実行者を識別する．

この修正後のモデルに対して同様の検証を行った結果，すべての安全性要件が満たされることを確認した．この検証結果より，データ連携アプリの実行時にデータ利用者の認証を実施することで，データ提供者が意図したデータ利用者によるデータ利用が行われることが示された．

本修正は簡易なモデル化であったが，実際にはパスフレーズや公開鍵証明書を用いる手法に置き換えることができる．パスフレーズを利用する場合は，DC_auth_info をパスフレーズ，DC_verify_info をパスフレーズのハッシュ値とみなし，データ連携アプリの初回起動時などにパスフレーズを設定することで後から攻撃者がデータ利用者に成り代わることを防ぐ方法が考えられる．公開鍵証明書を利用する場合は，DC_auth_info をデータ利用者の公開鍵証明書と秘密鍵による署名値，DC_verify_info をトラストアンカーとなる認証局の公開鍵証明書とし，チャレンジ&レスポンス方式などでデータ利用者を認証できる．提供データは高いプライバシが求められるデータであるため，たとえ

攻撃者から秘匿されていたとしても利用されること自体が望ましくない．パスフレーズは簡単に漏えいする可能性があるため，TPM（Trusted Platform Module）やスマートフォンなど物理的な要素と公開鍵証明書を組み合わせてより堅牢な認証メカニズムとすることが望ましい．

本研究では，データ利用者側で動作するデータ連携アプリでデータ処理が行われる非中央集権型のモデルを想定している．このモデルでは，データ提供者側でのデータ提供時における認証のみではデータ利用者の継続的な制御は困難であり，本検証で発見されたような意図しないデータ利用者によるデータ利用が生じうる．このような攻撃を防止するためには，データ提供者側における認証に加えて，データ連携アプリでのデータ利用者の認証も重要となる．

8. おわりに

本稿では，我々が提案している TEE を用いた非中央集権型データ連携フレームワークの形式検証を行った．提案フレームワークではデータ利用者側でデータ提供者の提供データを処理するため，データ提供者は提供データの不正利用や不正取得に関する不安を抱く．本検証ではこれらの不安を考慮して安全性要件を定義した．安全性評価は提供データの機密性とデータ利用者とアプリの認証，データ利用条件の順守に対して実施した．これらの安全性要件を形式検証ツールの ProVerif を用いて検証した結果，データ提供者による提供データの機密性が確保され，データ提供者の意図したデータ連携アプリで提示したデータ利用条件が順守されることが示された．また，データ提供者が意図しないデータ利用者によるデータ利用が生じることが示されたが，データ連携アプリでのデータ利用者の認証によって攻撃が防止できることを確認した．

今後は本稿でモデル化を行わなかったスマートコントラクトによるソースコードの解析やデータ利用申請をモデル化し，提案フレームワーク全体のプロトコルに対して形式検証を行う予定である．また，データ提供者が意図しないデータ利用者によるデータ利用を防止する手法についても検討する．

謝辞 本研究の一部は，JSPS 科研費 JP22K17881，JP23K03847 の助成を受けたものです．

参考文献

[1] Imran A. Zualkernan, Fadi A. Aloul, Vikram Sakkia, Hassan Al Noman, Salman Sowdagar, and Omar Al Hammadi. An IoT-based Emergency Evacuation System. In *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, pp. 62–66, 2019.

[2] Internet Society. Concerns Over Privacy and Security Contribute to Consumer Distrust in Connected Devices. `https://www.internetsociety.org/news/press-releases/2019/concerns-over-privacy-and-security-contribute-to-consumer-distrust-in-connected-devices/`, 2019. (Accessed 2024-06-04).

[3] Gonzalo Gil, Aitor Arnaiz, Francisco Javier Diez, and Maria Victoria Higuero. Evaluation Methodology for Distributed Data Usage Control Solutions. In *2020 Global Internet of Things Summit (GIoTS)*, pp. 1–6, 2020.

[4] 徳田祥太, 掛井将平, 齋藤彰一. 検証可能なデータ利用ポリシーに基づく Intel SGX を用いた非中央集権型データ連携フレームワークの提案. 研究報告コンピュータセキュリティ（CSEC）, Vol. 2023, No. 17, pp. 1–8, 2023.

[5] 徳田祥太, 掛井将平, 白石善明, 齋藤彰一. 非中央集権型データ連携の機密性向上のための TEE と分散台帳技術を活用したデータ利用制御の提案. コンピュータセキュリティシンポジウム 2023 論文集, pp. 431–438, Oct 2023.

[6] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative Instructions and Software Model for Isolated Execution . In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, New York, NY, USA, 2013. Association for Computing Machinery.

[7] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, Vol. 13. ACM New York, NY, USA, 2013.

[8] Thomas Knauth, Michael Steiner, Somnath Chakrabarti, Li Lei, Cedric Xing, and Mona Vij. Integrating Remote Attestation with Transport Layer Security, 2019.

[9] Patrik Hummel, Matthias Braun, Max Tretter, and Peter Dabrock. Data sovereignty: A review. *Big Data & Society*, Vol. 8, No. 1, p. 2053951720982012, 2021.

[10] Jaehong Park and Ravi Sandhu. The UCONABC usage control model. *ACM Trans. Inf. Syst. Secur.*, Vol. 7, No. 1, p. 128–174, feb 2004.

[11] Johannes Lohmöller, Jan Pennekamp, Roman Matzutt, Carolin Victoria Schneider, Eduard Vlad, Christian Trautwein, and Klaus Wehrle. The unresolved need for dependable guarantees on security, sovereignty, and trust in data ecosystems. *Data & Knowledge Engineering*, Vol. 151, p. 102301, 2024.

[12] 中林美郷, 奥田哲矢. 暗号プロトコルの形式検証ツールとその活用例. 応用数理, Vol. 33, No. 2, pp. 83–88, 2023.

[13] Bruno Blanchet. Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. In *Foundations of Security Analysis and Design VII*, pp. 54–87. Springer, 2014.

[14] Danny Dolev and Andrew Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, Vol. 29, No. 2, pp. 198–208, 1983.

[15] Shorouq Alansari. *A Blockchain-based Approach for Secure, Transparent and Accountable Personal Data Sharing.* PhD thesis, University of Southampton, August 2020.

[16] Hamed Arshad, Pablo Picazo-Sanchez, Christian Johansen, and Gerardo Schneider. Attribute-based encryption with enforceable obligations. *Journal of Cryptographic Engineering*, Vol. 13, No. 3, pp. 343–371, Sep 2023.