Omi

～　ウェブアプリケーションに透明性、検証可能性、検閲耐性を追加する　～

鹿目　黎*

マップ

1. 要旨・概要
    研究の要点をまとめる。
2. 問題提起・研究目的
    ムーブメントの詳細を記述。先行研究を列挙。目的と手段を記述。
3. 研究方法
    実際にどのように研究したのかを記述。使用する技術を列挙。
4. 結果
    純粋に結果だけ記述。(なぜそうなるのかなどは考察)
5. 考察
    理論を展開。2で示した先行研究などと比較。
6. 結論
    どのような成果が得られたのか、また課題は何かを記述。 (簡潔)

7. 参考文献　8. 謝辞　9. 図表

はじめに

　インターネットで調べた限り、車輪の再発明ではないと思うが正直分からない。本稿での提案システムは、JSSA2020での評価によっては実装を予定しており、現在は実装までには至っていない。(時間的に厳しかった)

　本稿の中核をなすところは”**Web2の透明性と検閲可能性についての問題を解決するような、次世代のWebアーキテクチャを提案すること**”である。そのため、主眼ではないところ(例えばP2Pネットワークにおける効率のよいデータの伝搬方法、P2PでのBootstrap問題、使用するハッシュ関数(例えばsha2-256やsha3-256は一方向ハッシュ関数という点において同じであり、それらのどれを使うかなどは詳述しない)など)はなるべく省略した。それらをすべて書くと文字数制限(約8000字)を大幅に超えてしまいそうだったし、新規性がないので書いても意味がないとおもった。実際、結果の章までで約8000字に達してしまった。

　考察の章以降は8000字を超えていて画像内に文字を貼り付けて文字数制限回避をしようかとも考えたが、意味がないと思ったのでそのまま書いた。考察のところで本稿の芸術性が顕れるので、文字数制限を超過しているが、考察まで読んでいただけると筆者としては喜ばしい。また、Eメールにコメントや質問をいただけると嬉しい。約11000字。以下から本稿。

* Eメール: yuki@grvti.sakura.ne.jp

要旨・概要

今日のウェブアプリケーションは透明性、検証可能性、検閲耐性といった性質を持たない。そのため、ブロックチェイン技術を中心としてWeb3ムーブメントが進んでいる。本稿では、ブロックチェイン技術とは異なる角度からウェブアプリケーションに透明性、検証可能性、検閲耐性を持たせる方法を提案した。

問題提起・研究目的

以下では本稿システムを提案する背景について記述した後、問題提起と研究目的を記述する。

背景

Web2とは

Web2では、ウェブアプリケーション(以降、アプリという)は単一の管理者によって管理される。現状のほとんどのアプリがこの中央集権的な方式である。Web2のアプリの例としては、私たちが普段使用するFacebook[1-1]やGoogle[1-2]などが挙げられる。Web2のアプリは以下の可能性を持つ。

1. 開発者が利用者の個人情報をみる (不透明性)
2. 得た個人情報を不正に利用する (不透明性)
3. 利用者の意図しない操作を行う (検証不可能性)
4. アプリの利用を禁止する (検閲可能性)

2はFacebookの選挙不正疑惑[2]が、3はTwitter[1-3]の内部ツールを標的としたソーシャルエンジニアリングによる影響力の強いアカウントからの意図しない投稿[3]、4はGitHub[1-4]のアクセス制限[4]が実際に起きた例として挙げられる。

曖昧さ回避のため、上記３つの性質を定義する。

不透明性
どのような情報が取得され、どのように分析され、どのように使用されているのかわからない

検証不可能性
受け取ったデータの差出人が本人か検証することができない

検閲可能性
特定のアプリ利用者のアプリ利用を制限・禁止することができる

Web3ムーブメントとは

上記4つの可能性があるにも関わらず、ユーザーはWeb2のアプリを利用している。理由としては、利便性やアプリの開発元を信頼しているということが考えられる。Web3ムーブメントはWeb2の持つ性質を踏まえ、アプリに透明性、検証可能性、検閲耐性を持たせようとする動きである。

先行研究・プロダクト

先行する研究・プロダクトとして以下の3つを紹介する。

IPFS
IPFS[5-1]は静的なファイルをP2Pネットワークで分散的に保存できるプロトコルである(ブロックチェインは使用しない)。P2Pネットワーク参加者に経済的インセンティブはなく、また静的なファイルしかホスティングできないためにインタラクティブなアプリが提供できないという問題が存在する。

Polkadot
Polkadot[5-2]は、異なるブロックチェインを繋げて大きな一つのブロックチェインを作ることで、ブロックチェインの持つスケーラビリティ問題と、インセンティブ不整合[5-3]などのセキュリティ問題を解決しようとしている。

DFinity
DFinity[5-4][5-5]の位置づけとしては、動的コードの実行と、インセンティブを備えたIPFSのようなものである。プロジェクト開始初期から現在に至るまで開発設計の指針がブレているが、現状はトークンによってネットワークのインセンティブが保たれている。具体的にはDFinityのネットワークに参加しているノード(キャニスターと呼ばれる)に対し、DFinityの開発元が発行する通貨(トークン)で対価が支払われる。(注[1])

[1] どのようなトークン発行形態をとるのか、まだ定かではない。('20-9-22時点)

現状分析

Web2のなかで非中央集権にするべきアプリと、中央集権のままで十分なアプリに分類した。また、分散システムにおけるトークンの働きの概略を記述する。

中央集権のままで十分なアプリ

• 開発者への信頼が利用の絶対条件のアプリ。

非中央集権にすべきアプリ

• 中央集権のままで十分なアプリ以外のすべて。

アプリを非中央集権にすべき理由は、ユーザーが信頼できない開発者や管理者の提供するアプリを利用するという点にある。アプリの開発者はWeb2の性質(不透明性、検証不可能性、検閲可能性など)を利用することができる。本稿では、上記の非中央集権にすべきアプリのみを対象とする。

トークンのはたらき

通常のピュアP2Pネットワークのソフトウェアはネットワークのコントロールが不可能に近いことが多く、P2Pソフトウェア開発者やP2Pネットワーク参加者は金銭的な利益を得ることは難しいと考えられている。対してブロックチェインを使用したP2Pネットワークの場合、開発者およびP2Pネットワーク参加者のインセンティブは暗号通貨によって保証される。開発者は暗号通貨を(Initial Coin Offeringなどで)発行することによって法定通貨(または別の暗号通貨)を受け取り、P2Pネットワークの参加者は、発行される暗号通貨を得たいために働く。

問題提起

たしかにWeb2での不透明性や検証不可能性、検閲可能性は問題である。しかしそれら問題の解決方法としてブロックチェインを使用することが必至なのかと疑問を持った。ピュアP2Pネットワークの性質上、透明性、検証可能性、検閲耐性を持つアプリはピュアP2Pネットワークを使用して実現できるはずである。(注[2])PolkadotはWeb3ムーブメントの一プロジェクトとして位置付けられているが、なぜブロックチェインの問題を解決することがWeb3ムーブメント中に位置付けられるのか、筆者には分からない(注[3])。ブロックチェインが持つ非中央集権的特性について、Vitalik氏(Ethereum[6-1]の考案者)はブログ記事[6-2]で次のように述べている。

“*Blockchains are politically decentralized (no one controls them) and architecturally decentralized (no infrastructural central point of failure) but they are logically centralized (there is one commonly agreed state and the system behaves like a single computer)*”

ブロックチェインを分散システムおよび分散データベースなどと比較して明確に異なるところは、合意形成アルゴリズムを持ち、P2Pネットワーク全体で統一されたロジック(上記の”logically”に対応)を持つということである。ブロックチェインはシステムをlogical centralizedにつくることができるが、アプリは必ずしもlogically centralizedを必要としないないため、logically centralizedに処理することが要求された場合にのみ、外部のブロックチェインを使用することが最適であると考えた。

また、DFinityはトークンをネットワーク参加者へのインセンティブ付与の手段として使用するが、インセンティブ付与の手段としてトークンを使用することに疑問を感じた。アプリにlogical centralizedが不要であれば、logical centralizedを必要とする概念であるトークンを使う必要はないと考える。

---

[2] 検証可能性は暗号学(署名など)で保証され、透明性は誰でもネットワークに参加できることによって(完全ではないが)保証され、検閲耐性は複数の管理者による各自独立した管理によって保証される。

[3] Polkadotがブロックチェインにはびこる問題を解決できるというのは理解し得るが、Web3がブロックチェインを中心として進んでいることに疑問をもった。

(logically centralizedはビザンチン障害耐性と言い換えることができる。)

研究目的

本稿のシステムはウェブアプリケーションに透明性、検証可能性、検閲耐性を持たせる。それらはピュアP2Pネットワークを用いることにより可能となるが、ピュアP2Pネットワークではアプリ開発者、ネットワーク参加者ともに収益化を行うことが難しい。本稿では、ブロックチェインを使用せずにP2Pネットワーク参加者及び開発者にインセンティブ付与を行うためのアーキテクチャを提案することを目的とする。

研究方法

目的を達成するための思考実験を、ノートに書き溜めつつ繰り返し行った。これらの過程でウェブ検索を行い、先行研究を確認した。このような研究方法をとったのは、明確なデータをとることができる種類の研究ではないからである。具体的なデータを述べる代わりに、使用技術は適切か、任意の攻撃や場合が考えられるが対策や対応はどうか、などといったことをひたすらに考え、アーキテクチャを精査した。

使用技術

本稿で提案するシステムで使用した技術を列挙する(説明は省略)。

- ピュアP2Pネットワーク
- 一方向ハッシュ関数
- 非対称鍵暗号
- IDベース暗号
- Aoiの考えかた (注[4])
- Digital Subscriber Line (以降、DSLという)

結果

準備

本節では用語を説明した後、本稿のシステム中の登場人物とアドレスを紹介する。

用語

曖昧さ回避のため、アプリとサービスの違いを明確にしておく。「アプリ」というと本稿で提案するシステム構成要素の最小単位を指し、「サービス」というとDev(後述)によって開発され、Node(後述)が提供する機能である。

登場人物とインセンティブ

本システムの登場人物は以下の5種である。

Trusted Party - TP
TPはUserの電話番号などで本人確認を実施する役割をもつ。TPは重み割り当て問題を解決するために導入される。TPは信頼のおける者によって開発、管理されるのが望ましい。

Developer - Dev
DevはNodeとUser用のソフトウェアを開発する役割をもつ。また、収益化のためにアプリ内UserIDを発行する。

ユーザー - User
Userはアプリを使用する。UserはReqの要求に答えてNodeにデータを挿入することが役割である。

ノード - Node
NodeはDevが開発したソフトを自身の環境で動かし、サービスを提供する。Userのデータを自身のデータベースに挿入、削除、取得する役割をもつ。Userがアプリを使用する際に常用的に使用(接続)する登場人物はNodeのみである。また、Nodeにはアプリから絶対的な管理者を削除するための役割もあるため、独立性を保たなくてはならない(意図的に選択した特定のサーバーにアクセスするなどは禁止。)。

Requester - Req
ReqはNodeを介してUserへの要求を行い、Nodeに仲介料として対価を支払う。また、Devに対してUserの正当性確認要求を行い、Devにも対価を支払う。Reqはアプリのインセンティブを賄う(スポンサーのような役割を持つ)。

[4] ここでは**「複数のTrusted Party(TP)に対して同様の要求を行い、返答結果を複合的に判断することで、１TPあたりの信頼を低下させる」**働きとして捉えていただきたい。

1つのアプリに関係する登場人物は複数のTP、Dev、Node、Req、Userが存在し得る。各登場人物の持つインセンティブは以下のようになる。

- Devはソフトウェアを開発し、Reqからの対価を受け取りたいとする。
- Reqは正当なUserに対して広告やアンケートの実施など、何かしらの要求を通したいとする。
- NodeはDevの開発したソフトウェアをUserに対してホストし、Reqの要求をUserに伝えることで対価を受け取りたいとする。
- TPはユーザーに対する対価または公益を求めるとする。
- Userはサービスを継続的に使用したいとする。また、より良いサービスを求めるとする。

本稿での対価は経済的な利益を想定している(思想的な利益のみの場合は、ピュアP2Pネットワークを構築するのみで十分である)。

登場人物間の信頼関係

登場人物間の信頼関係を図示すると図1のようになる。番号の信頼度合いについて詳述すると以下のようになる。
1. UserはTPが自身の個人情報を漏洩しないことと、可用性について信頼する。
2. DevやReqはTPが正常にUserの正当性を確認しているということを信頼する。複数のTPに同様の操作を要求することで、1TPあたりに必要な信頼性は低下する。
3. UserはDevの可用性について信頼する。後述する諸々の方法により可用性への信頼は低下する。
4. Nodeは複数存在すると仮定しており、複数のNodeを同時に同様に使用することでNodeへの信頼性は低下する。
5. NodeはReqに要求の対価を支払うという信頼をする。Reqが支払わなかった場合にはReqからの要求を聞かないことができる。
6. ReqはNodeを信頼しなくとも成り立つ。
7. DevはReqへ個人情報を要求するため、個人情報を漏洩しないという信頼が必要である。
ReqがDevに与える個人情報によっては中程度の信頼になり得る。

アドレス

アプリ間のリンクシステムを構築するためには、アプリのアドレスを表す手段が必要である。アプリのアドレス構造は、Devの公開鍵のハッシュと(ユーザーID発行のためのアプリをホストしている)IPアドレスをキーとして(Devがつくったアプリをホストしている)NodeのIPアドレスが保持される。DevがIPアドレスを変更したい場合、公開鍵の所有を示すことで変更することができる。また、複数のDevが同一のアプリを開発している場合、キーが複数になってNodeのIPアドレスと紐づく。データ構造は以下のようになる。(注[5])

PUT(<Hex; Dev's hash>, <String; Node's IP address>)

GET(<Hex; Dev's hash>) → List<String; Dev's IP addresses>, List<String; Node's IP addresses>

アドレスの保持者などについては省略するが、非中央集権的な問合せシステムを構成することなどが考えられる。

ハンドシェイク

UserとNode、UserとDev、UserとTP
"Userによるサービスの利用開始"で後述する。

NodeとReq
NodeとReq間で通貨の種類、支払いの量について合意形成を図る。

DevとReq
ReqとDev間で通貨の種類、Reqが渡す個人情報の種類について合意形成を図る。

DevとNode
合意形成は不要。

合意形成が失敗した場合、その登場人物間の関係は構築されない。

動作手順(システム利用フロー)

本節では本稿のシステムを利用する際の手順を主にUser目線から記述する。

[5] 書き方が分からなかったので、てきとうに書きました。

レイヤ

図2は本稿のシステムの論理階層である。図中の任意の数字について、その数以下の数が書かれてある他のレイヤを必要と(依存)する。(以降の内容を理解する助けになれば幸いである。)

Devによるソフトウェアの配布

DevはNodeとUserに配布するソフトウェアに自身の公開鍵(以降、この鍵ペアをDevキーという)とIPアドレスを含める。ソフトウェアには内蔵した公開鍵に対応する秘密鍵で作成した署名をつけて配布する。

Userによるサービスの利用開始

前提として、DevはNodeとUser用のソフトウェア、ID発行用のアプリを開発し、十分な数のNodeに配布し終えているとする。ユーザーがサービスを利用開始する際のフローとしては、TPへのユーザ登録(UserとTPのハンドシェイク)→DevへのユーザーID発行要求(UserとDevのハンドシェイク)→Nodeへの接続(UserとNodeのハンドシェイク)となる。

TPへのユーザー登録
ユーザー登録の過程では各TPが要求する方法で本人確認を行い、TPの内部アカウントを作成する。内部アカウントにUserの公開鍵を紐付ける。 1 TPあたりの信頼性を下げるため、Userは複数のTPに対してユーザー登録を行う。

DevへのユーザーID発行要求
ID発行フローは図3に示した。(以下の番号は図中の番号に対応する)

1. Userは、IDリクエストの際に自身の鍵(新しく作成した非対称鍵ペアの公開鍵。以降、この鍵ペアをIDキーという)のハッシュを送信する。

2. Devはソルト(十分に長く、規則性がなく、毎回異なる値の文字列)をUserに伝える。受け取ったハッシュは保持しておく。

3. Userは、送信したハッシュに受け取ったソルトを加えてハッシュ化し、仮のIDをつくる。仮のIDとDevキーの公開鍵とIPアドレスを、TPに登録した鍵での署名つきでTPに送信する。送信するTPは複数であることが望ましい(信頼性の低下)。TPは署名を検証し、正常であれば仮のIDとDevの公開鍵ハッシュを結合する。結合規則は例えば「Devの公開鍵ハッシュ-仮のID」といったように、予め定める。

4. 結合したものを(IDベース暗号での)IDとする秘密鍵をDevにDevの公開鍵で暗号化して送信する。Devは秘密鍵を受け取り、TPが自身の許可リストに存在するか、またTPからの秘密鍵が正常に動作するかを検証する。正常であればユーザーIDとソルト、TPからの秘密鍵を内部システムに登録し、Userに登録の完了を通知する。通知はUserがDevにアクセスして受け取る。通知内容はDevによるユーザーIDへの署名である。ここで、Devが署名に使用する鍵はNodeへのソフトウェア配布時に同封したものを使用する。通知を受け取った後、呼び方として仮のIDはユーザーIDへと変わる。

Nodeへの接続
Userは任意の場所からNodeの情報を取得し、複数のNodeを同時・同様に一つのサービスとして使用する。アプリによってはサービスを提供しているNode間でピュアP2Pネットワークを形成して協調する。協調すると例えばネットワーク内での情報の検索を行ったりすることができる。協調するかどうかはアプリの設計による。具体的な協調方法などは主眼ではなく、また先行する論文などでカバーされてあるので省略する。UserはNodeの使用開始時に各Nodeとハンドシェイクを行う必要がある。ハンドシェイクの過程では、IDキーの公開鍵とDevの署名とユーザーIDとユーザーIDの所有権を証明したもの(例えば、Nodeの情報とUnixtimeをIDキーの秘密鍵で署名するなど)を送信した後、Nodeがレスポンスを行う。このレスポンスでは、Userを受け入れるか受け入れないかを通知する。Devの情報がNodeの受け入れ判定の一つの材料として存在する。

Userによるサービスの利用開始後

UserがNodeの使用を開始した(ハンドシェイクを正常に終えた)後、Nodeに対して(1)データの操作、(2)Reqからの要求と反応、(3)使用Nodeの変更の3種類の操作が考えられる。

データの操作
UserはNodeにデータを挿入する際、パブリックデータかプライベートデータかを選択できる。パブリックデータであればUserのIDキーでの署名が必須となり、プライベートデータであれば暗号化して送信する。

Reqからの要求と反応
NodeとDevの収益化についてのフローを図４に図示した。以下は図中に対応する番号についての記述である。

1. ReqはNodeにUserへの要求を提示する。

2. NodeはReqからの要求をUserに提示する。

3. Reqからの要求に従い、UserはNodeに対して署名つきでデータを挿入する。このデータは、例えば広告であれば「見ましたよ」という意味のあるデータを挿入するなど、Reqからの要求に対してUserのなんらかの反応をデータとして挿入する。

4. Reqは定期的にNodeに保管されているUserの挿入したデータを要求する。

5. Userからのデータを取得後、データを挿入したUserが正当かを確認するために、Devに確認を要求する。DevはID発行処理の際に複数のTPから得た秘密鍵を使用し、Userの正当性を証明する。ReqはUserの正当性を複数のTPの保証から複合的に判断する。

ReqはDevにUser正当性確認の対価、Nodeに仲介料としての対価を支払う。

使用Nodeの変更
使用しているNode以外のNodeとのハンドシェイクは常時可能である。

考察

考える項目をインセンティブ、セキュリティ、現状との比較に分類した。

インセンティブ

「登場人物とインセンティブ」で示した各登場人物が持つインセンティブを満たしているかを確認する。

TP
TPはUserから、登録時または定期的にお金をもらうことで経済的インセンティブを満たす。

Dev
DevはReqからの対価を受け取ることでインセンティブを満たす。

Node
NodeはReqからの対価を受け取ることでインセンティブを満たす。

Req
ReqはNodeを通してUserに要求し、DevにUserの正当性確認を要求することでインセンティブを満たす。

User
UserはNodeにデータを挿入することによって、Nodeが利益を上げることができ、またDevは得た利益で新しいアプリを開始する可能性があり、インセンティブを満たす。

セキュリティ

鍵や宛先のすり替え

Devはソフトウェアを配布する際、DevキーとIPアドレスを含めるが、配布する際に何者かがDevキーとIPアドレスを書き換え、収益のみを横取りする可能性が考えられる。この問題への対策として、最初は信頼できるNodeへソフトウェアを配布し、大多数がDevを認知した段階で不特定多数にソフトウェアを公開するといった方法が挙げられる。しかし結局はUserやNodeの自由意志によって、Devではない者にID発行要求を行うことが可能であるため、UserやNodeのインセンティブ次第である。正当なDevに対するID発行要求を行うインセンティブとして、既存のアプリを改善したり新しいアプリへの期待などが挙げられる。つまりDevは登場人物から感謝や応援、期待をされなければ、他の者に利益を横取りされる。

Devへの信頼性度合い

DevはUserに対するTPの保証をReqに販売しているため、Devへの信頼が高くなるにつれ、ReqはDevにUserの正当性確認を行わなくなる可能性がある。

Devの可用性

Devがオフラインになった場合、新規Userがサービスを使用できなくなり、また、ReqはUserを確認できなくなるため、アプリのインセンティブが崩壊する。インセンティブ崩壊の可能性を低下させるための方法として、共同開発者や出資者のIPアドレス及び公開鍵もソフトウェアに内蔵するなどして予備のDevを確保することでDevの可用性が上がり、それに伴いアプリの可用性が向上するといった対策がある。DevはReqから対価を受け取れるため、予備のDevが存在しない場合に誰がDevになるかといった争いが起きることが予想される。争いの結果として、アプリに複数のDevが存在することとなる可能性がある。

Devによるバックドア設置

DevはNodeやUserのソフトウェアにバックドアの設置や意図しないコードを組み込む可能性がある。実装段階での対策として、それらのソフトウェアをDSLで記述することが挙げられる。そうすることで、DevからDSLの開発者へ信頼を(DSL開発者への信頼は必須であるが)移行することができる。

TPからのトラッキング

TPはUserの個人情報を知っているため、その個人情報とアプリ内IDを紐付ける可能性がある。

アプリ内ユーザーID発行フローで、UserはTPに対してアプリ内ユーザーIDとは別のIDを伝えている。そのため、TPによるトラッキングの可能性はTPとDevが共謀してトラッキングを行う場合を除いて排除される。TPとDevが共謀してトラッキングを行う場合、アプリ内ユーザーIDと個人情報は紐付けられる。

悪意を持つNode

以下ではNodeについて考えられる場合や攻撃について考える。

データの改竄など
NodeはUserから受け取ったデータを改竄することはできない。パブリックなデータには署名が必須であり、プライベートなデータには暗号化がかかっているためである。しかし、データを受け取ったにも関わらず、受け取っていないように振舞うことは可能である。そのためにUserはランダムに選択した複数のNodeを同様に使用する。

Devの開発したソフトウェアではない場合
NodeはDevが開発したソフトウェアをホストしていないのにもかかわらずDevが開発したサービスのように振る舞う可能性がある。Devのソフトウェアと同じ機能を提供していれば問題はない。ただしDevが開発したソフトウェアではない場合、ソフトウェア内部で何を行なっているかが不透明となる。

シビルアタック
上記ではUserがランダムに選択した複数のNodeを使用することでデータの損失は防がれると書いたが、そのNodeがすべて単一の管理元であった場合、簡単にNodeの悪意が通るため、データは損失する。この問題の対策として、Nodeが複数のTPにユーザ登録を行い、自身を保証してもらうことなどが挙げられるが、TPの権力が強まるという欠点が存在する。

悪意を持つReq

ReqはDevに対してUserの正当性の確認を行い、その正当性を他者に転売する可能性が考えられる。DevはTPから受け取った秘密鍵で自由に署名を作成することができるため、署名にReqの個人情報を含めることが可能である。転売の際には自身の個人情報も含めて転売することになるため、Reqによる転売は一定数防ぐことができる。この対策は、ReqがDevに個人情報を与え、Devは個人情報を検証する必要があるが、”保証の転売”には有効な対策であると考える。

ReqがNodeへの対価の支払いを怠った場合、NodeはいつでもReqとの関係を切ることができ、またNode間でP2Pネットワークが構築されている場合に、その評判を落とすことができる。ただし、Nodeが不当に評判を落とす可能性もあるため、評判を判断する際には、評判を落としているNodeへのある程度の信頼が必要である。

迷惑なUser

Userがすべきこととしてreqが要求したデータのNodeへの挿入があった。NodeはUserが提供するReqの要求に対する反応のデータによって利益が生まれるため、Userがそのデータの挿入を拒否した場合、そのUserに対して、Nodeは自身の利用を拒否する可能性が考えられる。

また、NodeとUserが手を組んでReqからの要求反応を過剰に行う可能性がある。ReqはNodeから受け取ったUserのデータをReq自身で設計した判定アルゴリズムを使用するなどして解析することで、一定数は防ぐことができると考える。

課題

インセンティブ面

”鍵や宛先のすり替え”や”Devへの信頼性度合い”で記述したように、Devのインセンティブをシステムとして完全に保証することができていない。

セキュリティ面

“TPからのトラッキング”で記述したように、アプリの一部分に不透明性が残る。

Web2との比較

透明性と検証可能性

UserはNodeへのデータ送信時に公開か非公開か選択することができ、自身のプライバシーを暗号に保証してもらうことが可能である。(透明性)
また、公開データには署名が必須であり、Userではない者がUserのように振る舞うことはできない。(検証可能性)

Nodeの単一サーバーへのアクセスは禁止されているが、NodeがDevの開発したソフトウェアではないソフトウェアを使用し、かつNode内部または外部サーバーで密かにユーザーのログを集積して分析する可能性は排除できない。また、”課題”の”インセンティブ面”で記述した課題も存在する。従って、本稿のシステムではアプリの完全な透明性は保証できない。

検閲耐性

Nodeは複数存在しており、いろいろな思想を持つNode管理者が存在すると考えられる。Devへの圧力により”Devの可用性”で考察した内容と同じ(ReqがUserを確認できない)状況が生じる場合があるが、NodeがDevの役割を果たすことで検閲耐性を持つ。(検閲耐性と開発者へのインセンティブ付与についてはトレードオフの関係が成り立つ。)

先行研究・プロダクトとの比較

Pokadotと本稿のシステムはシステムを構築する主旨が異なる(注[6])ので、比較するべきではない。IPFS及びDFinityと比較した。

IPFS

IPFSはシステムをピュアP2Pネットワークの参加者のみで構築するが、本稿はピュアP2Pネットワークの参加者ではない者も考慮することで、関係者へのインセンティブ付与とP2Pネットワーク参加者によるインタラクティブなアプリの提供を実現した。

DFinity

トークンによるネットワークのインセンティブ設計はビザンチン将軍問題を考えなければいけないが、本稿ではトークンによるインセンティブを設計を行わずに各登場人物間の直接的な取引によるインセンティブを設計した。直接的な取引により、ビザンチン将軍問題を考慮する必要がなく、登場人物は個々の自由意志にしたがって行動する。

登場人物が個々の自由意志に従って行動することによる本稿のシステム全体としての統一性がないことについて、筆者としては大きな利点であ

[6] Polkadotがブロックチェインの問題を解決するのに対し、本稿はブロックチェインとは関係がないため。Polkadotは背景の説明という目的で記述した。

ると考える(プラットフォーム化がなされていないことにより、より最適なアーキテクチャやシステムが提案され、インターネットのように自由に発展していく可能性があるため)。

結論

本稿では現状ブロックチェイン技術が中心として進むWeb3ムーブメントを別の角度から考え、ウェブに透明性、検証可能性、検閲耐性を追加した。

考察では提案した手法に対して定性的な評価を行ったが、秘密鍵の漏洩などといった一般的な諸問題への対処方法を記述していない。

開発者へのインセンティブ付与と検閲耐性が競合してしまうこと、透明性が完全に保証されていないことが課題である。

またReqが支払いを行わない場合の評判システム、アドレスの問合せシステムのインセンティブについては今後、詳細を考える必要がある。

参考文献

問題提起・研究目的

既存のウェブアプリケーション [1]

1. [Facebook] https://facebook.com
2. [Google] https://google.com
3. [Twitter] https://twitter.com
4. [GitHub] https://github.com

Facebookの選挙不正 [2]

1. [The Guardian] The Cambridge Analytica Files - https://www.theguardian.com/news/series/cambridge-analytica-files

---

Twitterの内部ツールへのハッキング [3]

1. [Twitter] An update on our security incident - https://blog.twitter.com/en_us/topics/company/2020/an-update-on-our-security-incident.html

---

GitHubの不当なアクセス制限 [4]

1. [ZDNet] GitHub starts blocking developers in countries facing US trade sanctions - https://www.zdnet.com/article/github-starts-blocking-developers-in-countries-facing-us-trade-sanctions/
2. [GitHub] GitHub and Trade Controls - https://docs.github.com/en/github/site-policy/github-and-trade-controls

---

先行研究・プロダクト [5]

IPFS

1. [IPFS] IPFS - https://ipfs.io

Polkadot

2. [Polkadot] Polkadot Wiki - https://wiki.polkadot.network/en/

インセンティブ不整合問題

3. [首藤 一幸] Bitcoinの革新性が導くWeb 3 - http://www.shudo.net/article/202001-IPSJ-cryptoeconomics/

DFinity

4. [DFinity] FAQ - https://dfinity.org/faq/
5. [DFinity] A Technical Overview of the Internet Computer - https://medium.com/dfinity/a-technical-overview-of-the-internet-computer-f57c62abc20f

問題提起 [6]

1. [Ethereum] Ethereum - https://ethereum.org/
2. [Vitalik Buterin] The Meaning of Decentralization - https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274

謝辞

まず、今の情報科学を作りあげた先人たちに感謝する。先人というのは、情報科学者のみならず数学者や経済学者など、広域にわたる。また、至らないところ(主に文字数の**over4000字**)があるにも関わらず、本稿を読んでくださったJSSA審査員の方々に感謝する。

黎美さん。僕は中学生の頃、学校にはほとんど行ってなかった。経緯は控えるが一種の精神衰弱になっていて、夜はずっと眠れなかったし朝もおきれなかった。考えることといえば死にたい(というかこの世界からどこまでも逃げたい)だとかだった。久しぶりに先生に連れられて学校に行った際、転校生がひとりいた。その子はそんなに可愛くなかった。でもその子の笑った顔は特別だった。しとやかだった。現実から逃げるようにずっと本を読んでいた。その子はLINEも何も持たない人なので連絡手段がない。まあ、時々夢にでてくるので、その日は一日うれしい。ただ、でてきてほしくない気もする。中学の頃、「黎美」という名前が気になって「黎」を辞書でひいてみた。すると「くらい」だとか書いてあった。だからあの子はいつも暗いのかと思った。今、名前の意味を考えてみる。すると、まず真っ先に「黎」という漢字で思い浮かぶのは「黎明期」である。君がいるから、黎明期は美しい。

黎美さんへ捧ぐ

図表

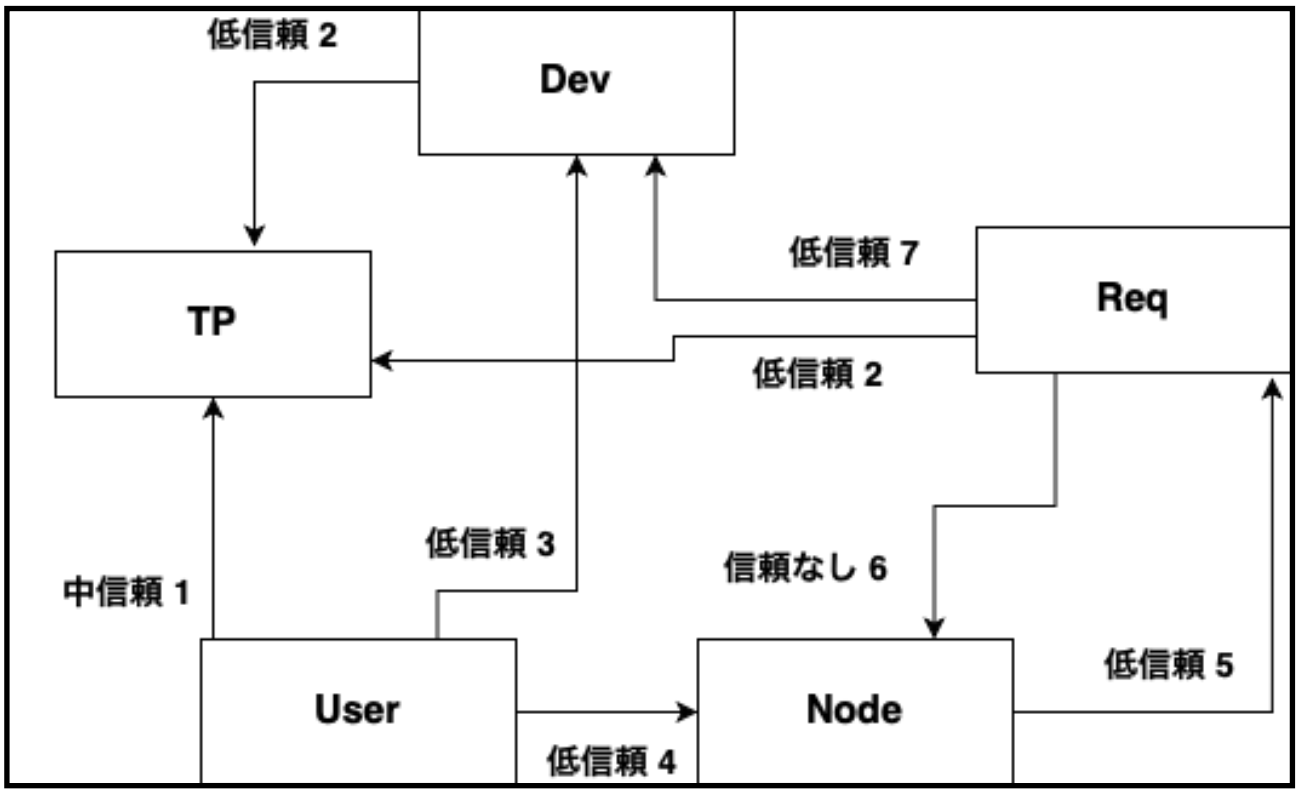

図 1　登場人物の信頼関係

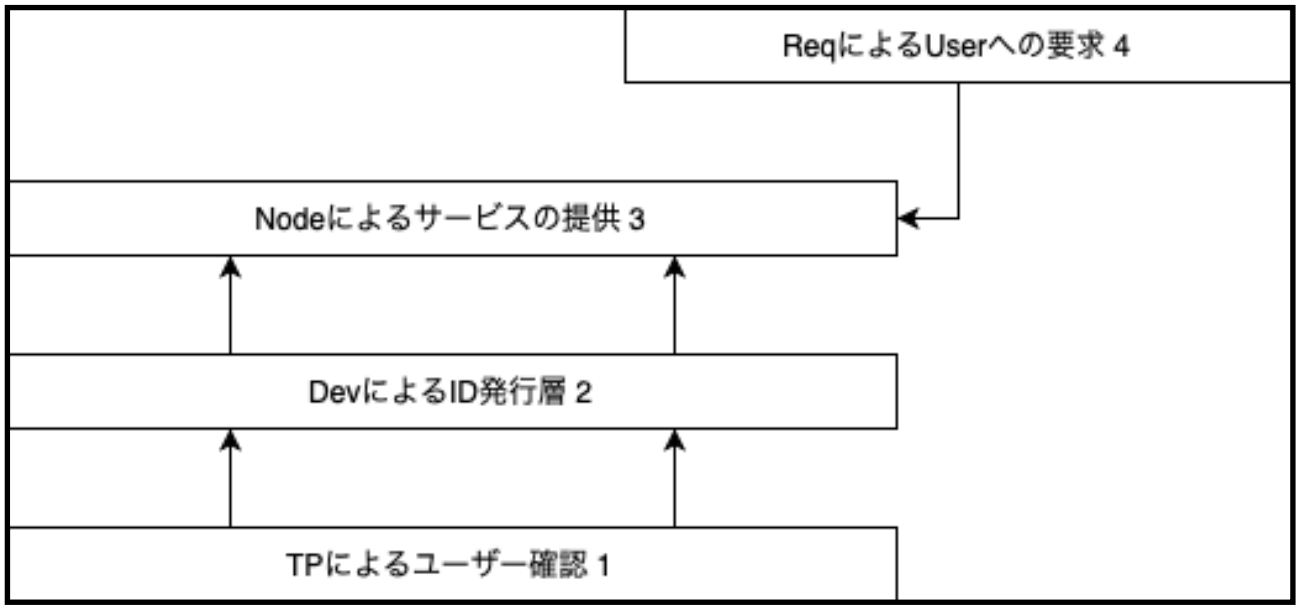

図２　アーキテクチャのレイヤ

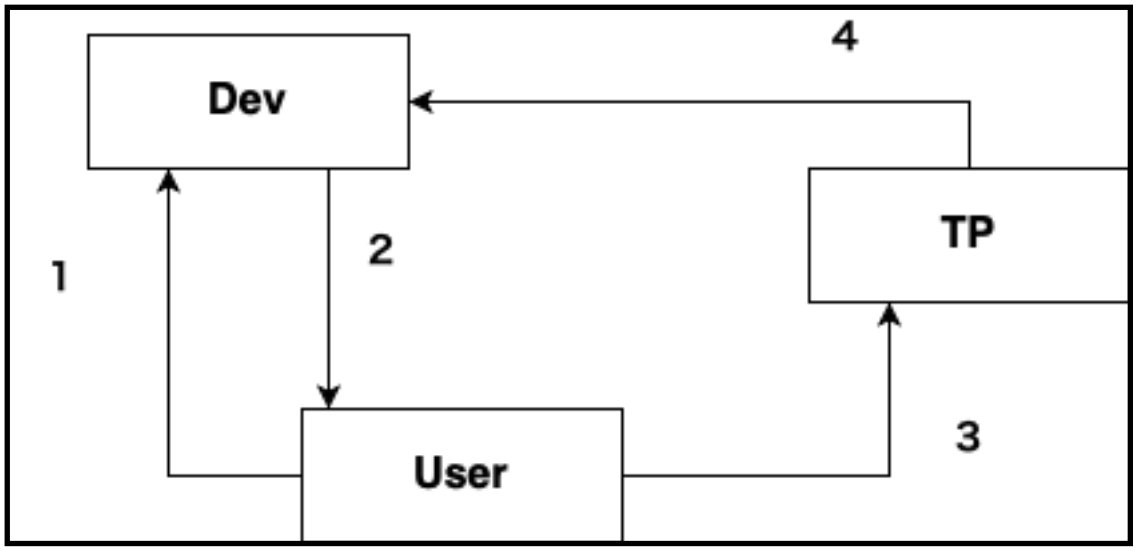

図３　サービス利用時のID発行フロー

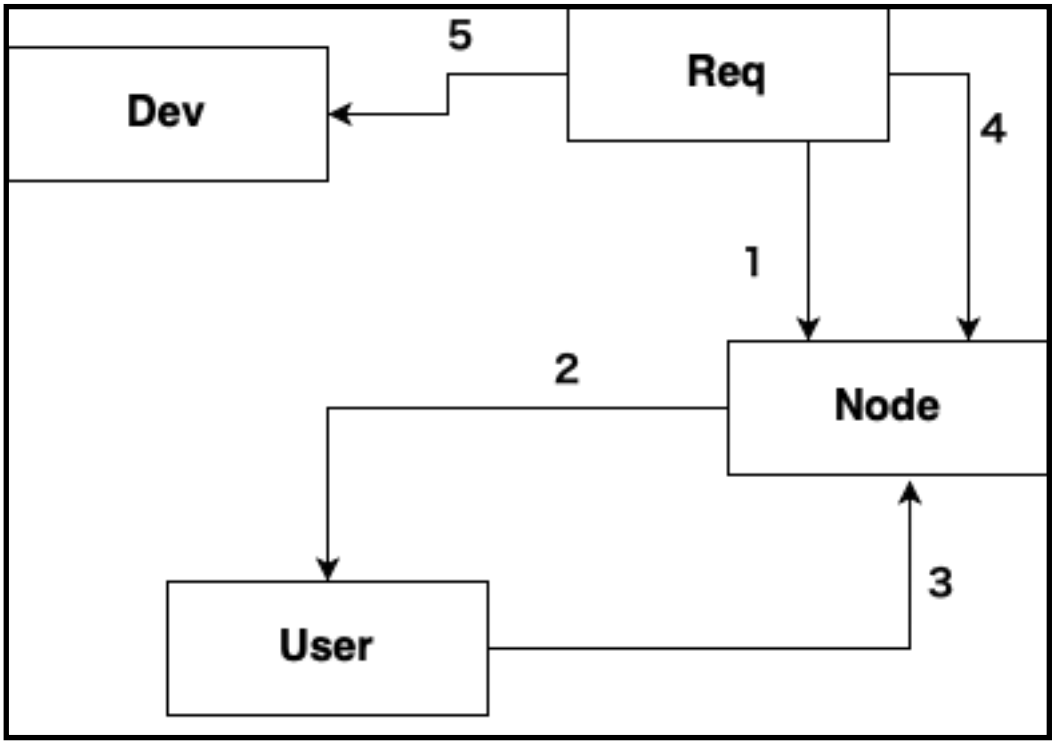

図４　Reqの要求とUserの正当性確認