ブロックチェーンを用いたクラウド内のアクセスコントロール管理

木下研究室　201503871 鎌田　啓太

研究背景

近年、ownCloud や AWS(Amazon Web Servise)などのクラウドサービスが開始し、企業、また一般ユーザーに広く普及される一方で情報漏えいのリスクが高まっている。

クラウド内において、得たい情報を本来意図されている通信経路とは別の経路を利用して情報を得ることが可能となるCovert channelが発生することで情報漏えいにつながってしまう。

先行研究

先行研究では、アクセス行列をグラフによってモデル化、またハイパーグラフによって、複数の推論から導出可能な組み合わせを算出することで現在発生しているCovert Channelを検知するものになっている

時間経過を考慮した動的な
Covert Channelに対応していない

目的

・従来のCovert Channel分析に時間の概念を追加することで、今まで検知できなかったCovert Channelに対応する

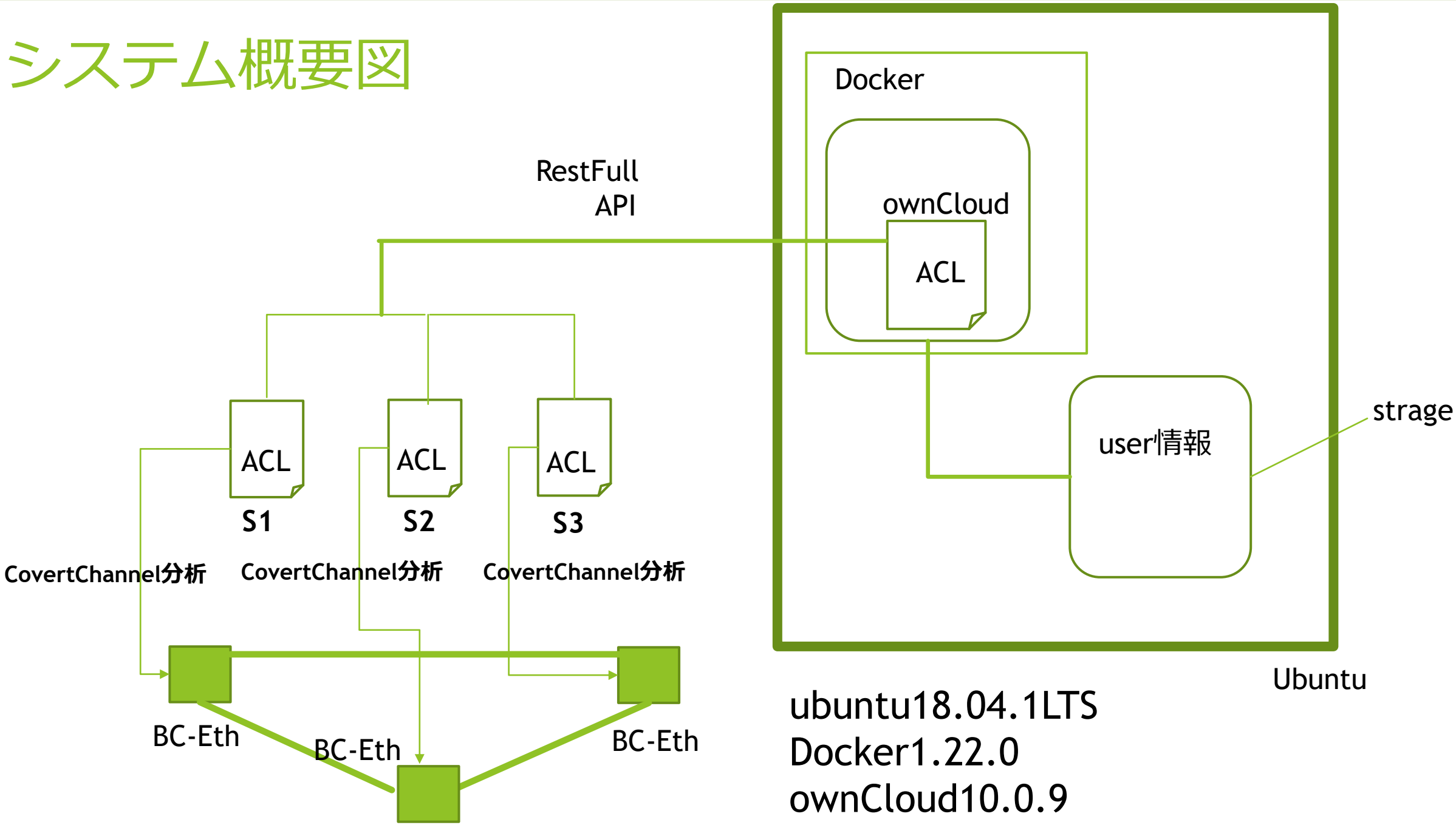
・ACLにブロックチェーン化を行うことで、管理者がいなくとも利用者間でアクセス権限の管理を行い、データの健全性を証明可能なシステムの提案

提案

ownCloud内のストレージからACLを取得

再設定する過程でCovert channel分析を行い、その後ブロックチェーン化

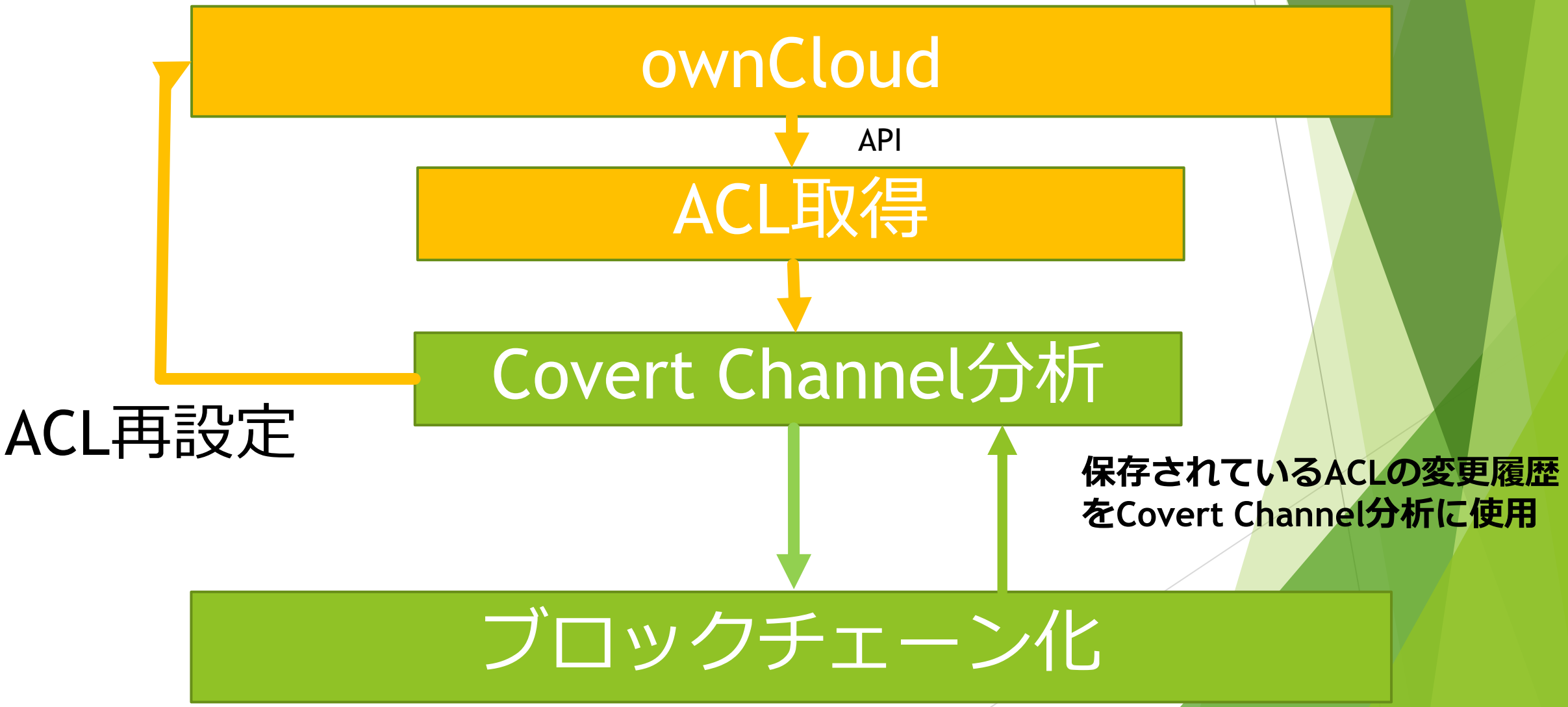
システム概要図



ubuntu18.04.1LTS
Docker1.22.0
ownCloud10.0.9

実装状況

下記図の黄色部分、（ownCloud環境、APIによるACLの取得、再設定）までの実装を行った。



実装環境

ubuntu18.04.1LTS

Docker1.22.0

ownCloud10.0.9

ブロックチェーン

Covert Channel修正後のACLにブロックチェーン化を行う。これによって、ユーザー全員がACLを共有し、自立分散型ネットワークの持つ相互監視によって、データの改ざんを防ぎ健全性を証明する。
(例)S1がACLの変更を行う。

- **Covert Channelが発生した場合修正**
- **変更履歴を全てのユーザーと共有**
- **ユーザー全員で了解を交換**
- **S2.S3の取引履歴をブロック化**

ACL(アクセスコントロールリスト)

・ファイルなど何らかの対象への利用者のアクセス権限
　を列挙したリスト。

・アクセス権限は読み取りや上書き・削除、
(プログラムの)実行、権限の変更などいくつかの種類がある。

・組み合わせによって段階的に権限を設定することができる。

ownCloud内のAPIを利用して取得

実装したownCloud内にweb上からtestファイルを作成

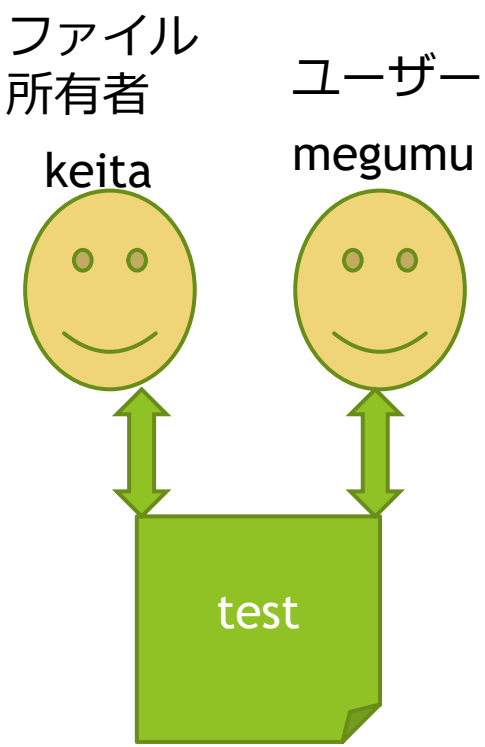
megumuというユーザに
Read、Writeなどを設定

APIを使用し、作成したtestファイルのACLを取得

```
curl
"https://172.17.0.1/ocs/v1.php/apps/files_sharing/api/v1/shares?path=test"
-k -u   keita:keita
<?xml version="1.0"?>
<ocs>
  <meta>
   <status>ok</status>
   <statuscode>100</statuscode>
   <message/>
  </meta>
  <data>
   <element>
    <id>7</id>
     <share_type>0</share_type>
      <uid_owner>keita</uid_owner>
      <displayname_owner>keita</displayname_owner>
      <permissions>31</permissions>
    <stime>1541662794</stime>
    <parent/>
    <expiration/>
    <token/>
    <uid_file_owner>keita</uid_file_owner>
<displayname_file_owner>keita</displayname_file_owner>
    <path>/test</path>
    <item_type>folder</item_type>
    <mimetype>httpd/unix-directory</mimetype>
    <storage_id>home::keita</storage_id>
    <storage>1</storage>
    <item_source>45</item_source>
    <file_source>45</file_source>
    <file_parent>9</file_parent>
    <file_target>/test</file_target>
     <share_with>megumu</share_with>
<share_with_displayname>megumu</share_with_displayname>
    <share_with_additional_info/>
    <mail_send>0</mail_send>
```

Share type
0=ユーザー
1=グループ

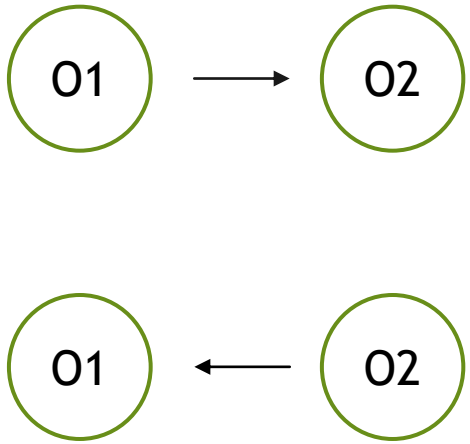
Permissions
1=読み込み
2=更新
4=作成
31=全て



読み込み、更新、作成の全てを許可

Covert Channel

	S1	S2
O1	R	Φ
O2	RW	R



アクセス行列からS1がO1をreadし,その情報をO2にwriteしてしまうと, そのO2をS2がreadすることによってO1に記述されていた情報がS2に漏えいしてしまう

ブロックチェーン化によってACLの変更履歴を保存することで、時間経過を考慮したCovert Channelを検知する

③重ね合わせる

	S1	S2
O1	R	
O2		

t=0
①S1がO1をRead
Covert Channelは発生していない

	S1	S2
O1		
O2	W	R

t=1
②S1がO2Write
S2がO2をRead
Covert Channelは発生していない

	S1	S2
O1	R_{t0}	
O2	W_{t1}	R_{t1}

④
時間経過によって発生する動的なCovertChannelを検知

ACLの再設定

ownCloud内のAPIを利用し、HTTPメソッドによってストレージ内ファイルのACLを再設定する。
対象となるファイルを指定、共有するユーザーまたはグループを決定し、読み込み、書き込みといった権限のレベルを設定する。

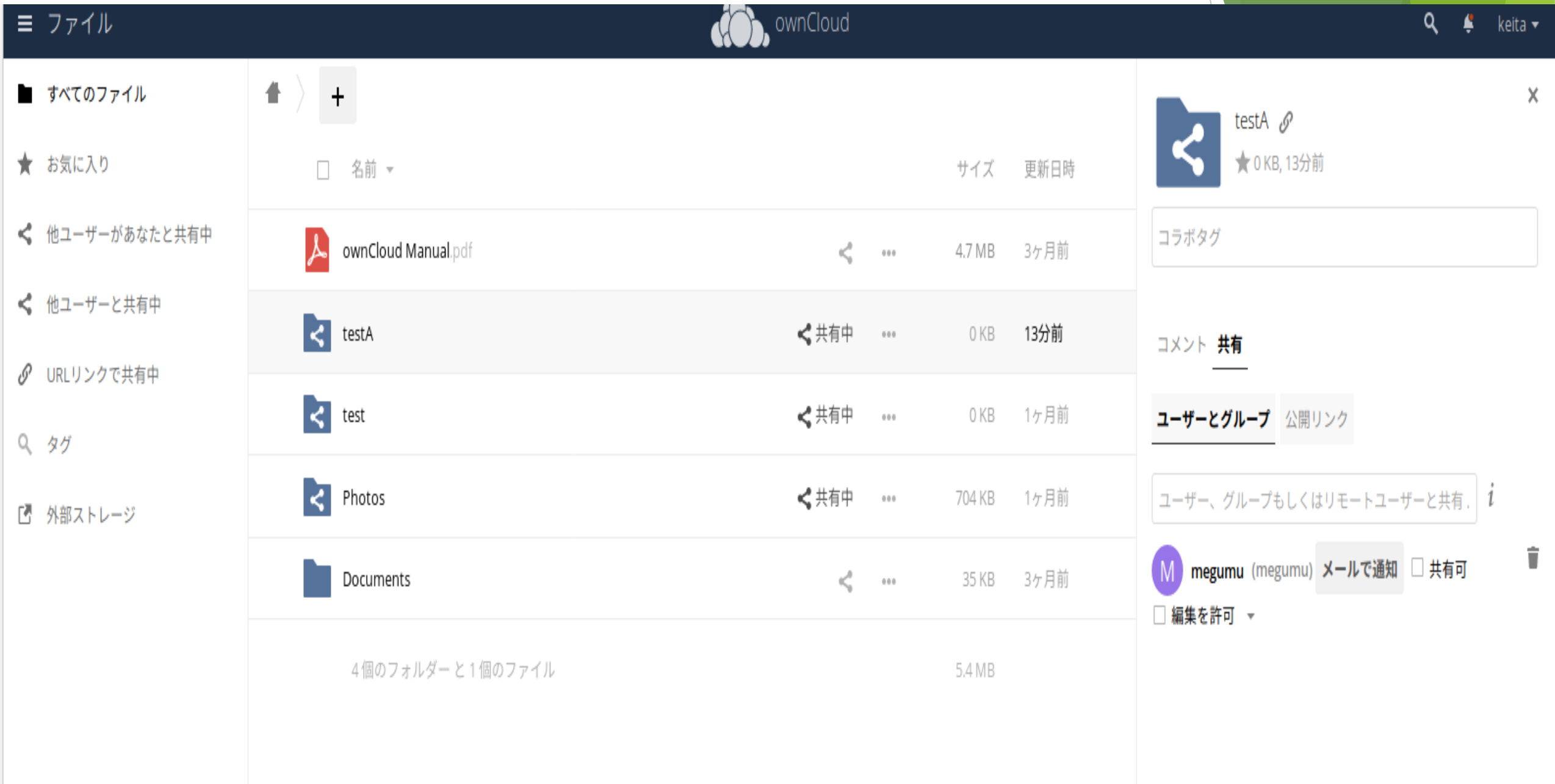
APIを使用しtestAファイルにACLの再設定を行う

```
keita@keita-NUC7i7BNH:~$ curl "http://172.17.0.1/ocs/v1.php/apps/files_sharing/api/v1/shares" -l -k -u keita:keita -X POST --data-urlencode "path=/testA" --data-urlencode "shareWith=megumu" --data "shareType=0&permissions=1"
<?xml version="1.0"?>
<ocs>
  <meta>
   <status>ok</status>
   <statuscode>100</statuscode>
   <message/>
  </meta>
  <data>
   <id>9</id>
   <share_type>0</share_type>
   <uid_owner>keita</uid_owner>
   <displayname_owner>keita</displayname_owner>
   <permissions>1</permissions>
   <stime>1545273019</stime>
   <parent/>
   <expiration/>
   <token/>
   <uid_file_owner>keita</uid_file_owner>
<displayname_file_owner>keita</displayname_file_owner>
```

Share type
0=ユーザー
1=グループ

Permissions
1=読み込み
2=更新
4=作成
31=全て

ACLの再設定を行ったtestAファイルをweb上から確認



結論

本研究では、ACLの管理にブロックチェーンを使用することで、ACLの変更履歴を保存し、従来の方法に時間の概念を追加したCovert Channel分析を提案した。

時間の概念を追加することによって、過去から現在を通して発生する動的なCovert Channelの検知を可能とするシステムの提案ができる。

今後の課題

ownCloud、ブロックチェーン環境、APIによるACLの取得、再設定までの実装を行った。

今後の課題は、時間の概念を追加したCovert Channel分析のアルゴリズムの作成、また、ブロックチェーンのコントラクトコードの作成である。

メモ
先行研究　ハイパーグラフ

アクセス行列とcovert channel は有向グラフによってモデル化できる．グラフとは頂点集合とそれらを結びつける辺の二つで構成され，「点とそれを結ぶ線」の「つながり方」に着目して問題を考えるためのデータ構造である．つながり方だけではなく，辺でつながれた 2 頂点の順序を考慮してその有向性を矢印で表現したものを特に有向グラフと呼ぶ．その場合，有向性のないグラフは無向グラフと呼
ばれ，有向グラフと区別される．一般に，グラフは$G = (V;E)$ と記述される．ここで，Vは頂点集合，E はV の 2 頂点対の集合である．$u; v \ 2 \ V$,$(u; v) \ 2 \ E$ のとき，G が無向グラフならば，$(v; u) \ 2 \ E$ かつ$(u; v) = (v; u)$ であるが，G が有向グラフのときはそうとは限らない．ある頂点$u \ 2 \ V$ から，ある頂点$v \ 2 \ V$ へ辺をたどって到達可能な時，その到達経路を$(u; v)$ 歩道といいどの頂点も高々 1 回しかたどらない$(u; v)$ 歩道を$(u; v)$ 道と呼ぶ．
Fig.1 の左図のアクセス行列をグラフで表現したものを
Fig.2 に示す．
図2: アクセス行列のグラフ表現このグラフにおいて，実線はread，破線はwrite を意味する．グラフG 上で，Oi からSi への直接の矢印が無い,即ち，$(Oi; Si) =2 \ E$ にも関わらず，Oi からSi への$(Oi; Si)$
歩道が存在するときその歩道はcovert channel を意味する．したがって，グラフによってモデル化することでACL上のcovert channel 検知問題に対してグラフの歩道発見アルゴリズムを適用できるようになる[8]．また，グラフ化することで人間に対して視覚的に危険を訴えることができる

メモ
先行研究　マイナンバー制度におけるアクセス制御

ハイパーグラフによって個人番号と個人番号に紐づけられたデータを複数の推論から導出可能な組み合わせをすべて算出し, 推論による頂点着色を満たすリストの組み合わせが一つも存在しなければそのアクセス制御リストは推論による情報漏洩を考慮した時読み込めないオブジェクトが存在する不自由なACL ををを発見する問題設定を行うことを目的とする。
推論によるリスト着色が一つでも存在しなければそのACL はread することができないオブジェクトが存在することになる。そのようなACL はアルゴリズムの目的は与えられた依存関係とリストで構成されたグラフがリストの組み合わせが一

22

提案23

つでも存在していれば, そのACL は推論に対して安全なリストの組み合わせが存在していることを示す. 上記の図において(O1,O2) → (O5),(O3,O4) → (O5) だったときO1 とO2 がS0 で塗られ,O3 とO4 がS3 で塗られていたらO5 をS0 とS3 のどちらで塗るのかという問題がある. (O5,O6),(O7) で,O6 にS3 が塗られていて,O7 のL にS3 が無いときもしO5 をS0 で塗ったときO5 とO6 がS3 →推論でO7 もS3 になってしまうケースを見逃す可能性がある. このような場合はO5 にS0 を与える場合とS3 を与える場合の両方を記述することにする. このようにアクセス制御リストの修正が必要な場合もあり, 効果的に修正するためにはどのような問題設定をすればよいかを考察す

メモ
API

API
あるコンピュータプログラム（ソフトウェア）の機能や管理するデータなどを、外部の他のプログラムから呼び出して利用するための手順やデータ形式などを定めた規約のこと.

RESTful API
RESTful APIとは、Webシステムを外部から利用するためのプログラムの呼び出し規約（API）の種類の一つで、RESTと呼ばれる設計原則に従って策定されたもの。RESTそのものは適用範囲の広い抽象的なモデルだが、一般的にはRESTの考え方をWeb APIに適用したものをRESTful APIと呼んでいる。
RESTful APIでは、URL/URIですべてのリソースを一意に識別し、セッション管理や状態管理などを行わない（ステートレス）。同じURLに対する呼び出しには常に同じ結果が返されることが期待される。
また、リソースの操作はHTTPメソッドによって指定（取得ならGETメソッド、書き込みならPOSTメソッド）され、結果はXMLやHTML、JSONなどで返される。また、処理結果はHTTPステータスコードで通知するという原則が含まれることもある。