分散台帳技術を用いた分散ファイルシステムの構想

中村 公洋[1]　新城 靖[1]　佐藤 聡[1]　中井 央[1]

1. はじめに

クラウドストレージサービスは、複数端末を持つ個人がファイルを同期するために利用されるだけでなく、友人同士のファイルの共有においても多く利用されている。このようなクラウドストレージサービスは、中央サーバに依存しているため、それに起因する様々な問題が生じる。例えば、中央サーバに障害が発生したとき、ユーザはサービスに接続できずファイルを扱うことができない。また、悪意ある中央サーバの管理者や侵入者によって、ユーザの通信やファイルの内容が監視される可能性がある。

中央サーバに依存しないファイルの共有方法として、P2P (peer-to-peer) に基づく方法が挙げられる。しかし、このようなファイルの共有方法では、相手がオンラインでないとファイルを共有できない。また、既存のサービスの多くはユーザ認証やアクセス制御などの機能を持っていない。

中央サーバに依存しない技術として近年分散台帳技術 (DLT: Distributed Ledger Technology) が注目を集めまた普及しつつある。例えば、分散台帳技術を用いた Ethereum [1] は、スマートコントラクトと呼ばれるプログラム実行環境を持っており、2019 年 11 月 20 日現在、1 日に約 70 万件のトランザクションが処理されている [2]。分散台帳技術ではトランザクションにユーザの識別子であるウォレットのアドレスが用いられ、これはユーザの公開鍵から生成される。

本研究の目的は、相手がオフラインでも利用可能な中央サーバに依存しない分散ファイルシステムを実装することである。そのために本研究では、分散台帳技術を用いる。従来の分散台帳技術で使われているユーザのアドレスは、PC で動作している普通の (集中型の) OS におけるユーザ識別子 (UID: User Identifier) と全く異なり、使いにくい。そこで本研究では、分散台帳技術のアドレスを集中型 OS のユーザ識別子にマッピングを行い、使いやすいものにする。具体的にはユーザ識別子を用いたファイルに対するアクセス制御を可能にする。本研究では、分散ファイルシステムのアプリケーションとして、分散型 SNS と電子出版を実装する。

[1] 筑波大学

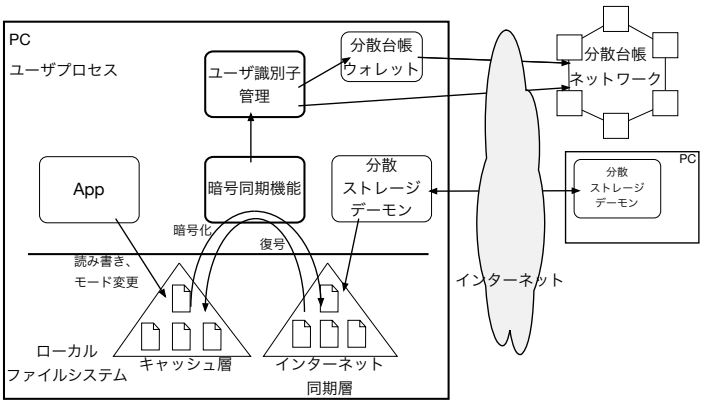

図 1　ファイルシステムの構成

2. 分散台帳技術を用いた分散ファイルシステムの設計

本研究で提案する分散ファイルシステムの設計を図 1 に示す。本研究の分散ファイルシステムは以下のモジュールから構成されている。

- ユーザ識別子管理機能
- ローカルファイルシステム。キャッシュ層、および、インターネット同期層から構成される。
- 暗号同期機能

その他に、各利用者の PC では、次のプログラムを実行する。

- 分散台帳技術のウォレット。
- 分散ストレージデーモン。IPFS (InterPlanetary File System) 等の、既存のインターネット上で利用可能な分散ストレージ・サービスの常駐プログラム。分散台帳上の高価なストレージの代わりとして、暗号化した大きなファイルを保存する。

2.1 ユーザ識別子管理

ユーザ識別子管理機能は、本分散ファイルシステムの各ユーザ (友人、出版社、購読者) についての次の情報を保持する。

- 分散台帳におけるウォレットのアドレス。一般的には、ウォレットの公開鍵から作られる。
- ローカル OS におけるユーザの識別子 (UID (User Identifier))。32 ビットの整数。
- ローカル OS における短い文字列の名前。ls -l での表

示等で使われる。

これに加えて、ユーザ識別子管理機能は、ユーザグループも管理する。具体的には、グループ識別子 (GID)、グループの名前、グループのメンバのリスト等を保持する。

本研究では、ユーザ認証機能、すなわち、分散ファイルシステムへのログイン機能を分散台帳技術で実装する。分散台帳で個人所有のウォレットは、個人の認証にも利用可能である。しかしながら、これらの仕組みは、一般的な OS が提供する仕組みとはほど遠く、使いにくいものである。本研究では、これをローカル OS の UID にマップして扱いやすくする。すなわち、本研究で実装するアプリケーションや既存の ls -l 等のコマンドは、システム・コールでローカル OS の UID を扱えば良いようにする。ユーザ識別子管理機能は、上記の情報を利用して UID で指定されたユーザを、ウォレットにより認証されたユーザに対応付ける。

2.2 ローカルのファイルシステムと暗号同期機能

本分散ファイルシステムは、次の 2 層のローカルファイルシステムを利用して実装する。

- **キャッシュ層**。この層は、暗号を解いた状態のファイルを保持する。アプリケーションは、キャッシュ層のファイルを読み書きする。
- **インターネット同期層**。この層は、暗号化されたファイルを保持する。IPFS 等の分散ストレージデーモンは他のノードと暗号化されたファイルを双方向で同期する。

本分散ファイルシステムは、利用者空間で動作するプログラム「暗号同期機能」により、キャッシュ層と同期層の間で双方向にファイルをコピーすることで実装する。このプログラムは、キャッシュ層でファイルが更新された場合、それを読み出し、暗号化して同期層へ書き込む。逆にインターネット同期層でファイルが更新された場合、それを読み出し、復号してキャッシュ層へ書き込む。暗号の鍵は、ユーザ識別子管理機能から取得する。

本ファイルシステムでは、ローカルの (集中型の) OS で設定された読み書きのアクセス権を、インターネット上で実現する。一般に、インターネットにおいて限られた他の利用者にだけファイルの読み出しを許可することは容易であるが、書き込みを許可することは容易ではない。本研究では、分散台帳技術が提供するスマートコントラクトを利用してそれを実現する。

ファイルの内容を暗号化するとき、読み出し可能なユーザが少数なら、それらの利用者のウォレットの公開鍵による暗号化を繰り返す。多数なら、属性ベース暗号を使い、1 度の暗号化で済ませる。この場合、ウォレットの公開鍵を利用し、属性秘密鍵を事前に配布しておく。

3. 本分散ファイルシステムを用いたアプリケーション

分散ファイルシステムのアプリケーションは、ローカルファイルを操作するものとして開発する。具体的には、本研究では以下のものを実装する。

- 分散型 SNS。/etc/group と同様の仕組みを用いて、友人の UID をグループとして管理する。友人に読み書きを許すファイルのモードを chmod g+r や chmod g+w で設定する。
- 自由な電子出版。分散型 SNS と同様にアクセスできる人のグループを作る。出版物の購入者の UID をグループに追加する。

4. 関連研究

論文 [3] は、Ethereum、IPFS、および、属性ベース暗号を組み合わせて分散ファイルシステムを実現することを提案している。その特徴は、属性秘密鍵等の保存と受け渡しに分散台帳を利用している点にある。

本研究ではアクセス制御にユーザ識別子とグループ識別子を用いる。また、分散台帳技術のアドレスとローカル OS のユーザ識別子をマッピングしてアクセス制御の設定を簡単に行えるようにする。

5. まとめ

本研究では、相手がオフラインでも利用可能な分散台帳技術を用いた分散ファイルシステムを提案する。本分散ファイルシステムではアプリケーションはローカルファイルと同じ方法でファイルを読み書き、および、アクセス制御を可能にする。本分散ファイルシステムの実装はユーザ識別子管理機能および暗号同期機能で構成される。ファイルのデータは、高価な分散台帳上のストレージではなく、インターネットで安価に利用可能な分散ストレージを用いて保存する。

現在までに既存の分散ストレージである IPFS、Storj および、DAT を動作させ、本研究の分散ストレージとして適しているか仕組みを調査した。今後、提案方式に基づき分散ファイルシステムを実装し、そのアプリケーションを実装することでその機能を評価する。

参考文献

[1] Ethereum, https://www.ethereum.org. Accessed: 2019-10-25.
[2] Ethereum Transaction History, https://etherscan.io/chart/tx. Accessed: 2019-10-25.
[3] S. Wang and Y. Zhang and Y. Zhang, “A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems,” IEEE Access, pp.38437–38450, 2018.