Vol.2019-MBL-93 No.3
Vol.2019-ITS-79 No.3
2019/11/20

存在の否認が可能な暗号化ファイルシステムの Trusted Execution Environment を用いた実現

柴崎 凌我[1]　稲村 浩[1]　中村 嘉隆[1]

概要：　モバイルインターネットのサービスにおいて，プライバシーに関わる機密データの漏洩はデータ所有者のプライバシーを危険にさらし，漏洩させた組織の社会的信用の失墜に繋がるため，このようなデータの保護が重要な問題となっている．ストレージシステムにおいては，フルディスク暗号化等の従来手法が一般に使用されているが，これらの方法では計算機ハードウェアへのアクセスや管理者権限が攻撃者によって奪取された場合の機密性の維持が難しい．これに対し，新しい暗号化の考え方である Plausibly Deniable Encryption (PDE) が提案されている．PDE は，情報の存在の否定を可能にすることで機密情報を十分に保護するというものである．PDE は，囮の鍵を開示することで，攻撃者による復号鍵の開示の強要から保護する．しかし，ストレージシステムの構成の観点では，従来の手法は主に永続記憶装置での機密情報の保護を行っており，実行時における機密情報の存在を司る主記憶装置が攻撃されることは想定されていないが，近年の仮想化技術の広まりによって今や検討すべき課題となった．本研究では PDE の考え方を用いて存在の否認が可能な暗号化ファイルシステムの，信頼できる実行環境での実現を目的とする．ハードウェアによる保護された実行環境として Intel SGX を用いたシステムを提案し，作成中のプロトタイプに関して一部機能の評価を報告する．

キーワード：Intel SGX，Plausibly Deniable Encryption，OS Security

Ryoga Shibazaki[1]　Hiroshi Inamura[1]　Yoshitaka Nakamura[1]

1. はじめに

プライバシーに関わる機密データの漏洩はデータ所有者のプライバシーを危険にさらし，漏洩させた組織の社会的信用の失墜に繋がるため，このようなデータの保護が重要な問題となっている．ストレージシステムにおいては，フルディスク暗号化等の従来手法が一般に使用されているが，これらの方法では計算機ハードウェアへのアクセスや管理者権限が攻撃者によって奪取された場合の機密性の維持が難しい．これに対し，新しい暗号化の考え方である Plausibly Deniable Encryption (PDE) が提案されている [1]．PDE は，情報の存在の否定を可能にすることで機密情報を十分に保護するというものである．PDE は，囮の鍵を開示することで，攻撃者による復号鍵の開示の強要から保護する．暗号化ファイルシステムがシステムに存在することは認めるとして，攻撃者に囮の鍵を渡すことで，囮領域にアクセスさせるが，隠し領域の存在とその内容は秘匿される．ストレージシステムの構成の観点では，従来の手法は主に永続記憶装置での機密情報の保護を行っており，実行時における機密情報の存在を司る主記憶装置が攻撃されることは想定されていないが，近年の仮想化技術の広まりによってクラウドサービスが普及したため [2] 検討すべき課題となった．すなわち，クラウドサービスで用いられるハイパーバイザ等の仮想化技術のようなハードウェアを特権制御する技術の利用を前提に，システムの構成を理解するものが脆弱性をついて主記憶装置への攻撃を試みる危険性が存在する．このような攻撃の一例として，2019 年にはアメリカの銀行であるキャピタル・ワンでアマゾン・ウェブ・サービスに保存された約 1 億人分のデータが違法にアクセスされた事件 [3] が挙げられる．

本研究では，永続記憶装置だけでなく主記憶装置への攻撃耐性をもつ，PDE の考え方を用いた存在の否認が可能な暗号化ファイルシステムの構築を目的としている．暗号

[1] 公立はこだて未来大学　システム情報科学部
School of Systems Information Science, Future University Hakodate

化ファイルシステムの実現においてハードウェアによる保護された実行環境として Intel SGX を用いたシステムを提案し，プロトタイプに関して評価を報告する．

本稿は以下の通りに構成する．2 節にて関連研究，3 節にて脅威モデル，4 節にて研究の目的・課題，6 節から 7 節まで提案手法の前提及び設計，8 節に置いて提案システムのレイテンシの評価を述べ 9 節でまとめる．

2. 関連研究

本節では Plausibly Deniable Encryption の概念とそのファイルシステムへの適用，主記憶装置への攻撃耐性の実現に適用を検討する Intel SGX，そしてプロトタイピングに利用する Filesystem in Userspace について述べる．

2.1 Plausibly Deniable Encryption

Plausibly Deniable Encryption(以下 PDE と表す) は暗号化手法の 1 つとして Canetti ら [1] によって提案された．図 1 のように従来の暗号化とは異なり機密情報に囮の鍵と秘密鍵の両方で復号化が可能な特殊な暗号化を施す．囮の鍵を用いて復号化をすると囮の平文が得られ，秘密鍵で復号化をすると本来の平文が得られる．PDE はこの特性を利用し，攻撃者に本来の情報を気づかせないというものである．しかし，暗号文のサイズが極端に大きくなるため，攻撃者に特殊な暗号を施していることに疑念を持たせる可能性がある．また，機密情報の痕跡がファイルシステム及び物理記憶媒体層等から取得される可能性があり，実用的な方法とは言えない．このことより，単純な暗号化を使用するのではなく，ステガノグラフィと隠しボリュームの 2 種類の技術を使用して機密性をもたらす方法が提案された．まず，ステガノグラフィの考え方を用いた手法が Anderson ら [4] や Chang ら [5] によって提案された．基本的な考え方は機密情報を通常の情報の中に隠すことである．これは，上書きを回避するためにデータを複数コピーして保存することで軽減するが，記憶装置の使用効率が悪化する．隠しボリューム技術は Jia ら [6] や Zuck ら [7] などに提案されている．記憶装置上に，囮ボリュームを囮の鍵で作成し，隠しボリュームを隠し鍵で作成する．囮ボリュームは記憶装置全体に配置され，隠しボリュームは通常，永続記憶装置上の隠しボリュームの初期位置である隠しオフセットから記憶装置の末尾に向かって配置される． 隠しボリューム技術では，公開ボリュームを扱うシステムでは隠しボリュームの存在や隠しオフセットが未知となるため，囮ボリュームに格納されたデータが隠しボリュームを上書きする可能性がある．

2.2 Intel Software Guard Extensions

Intel Software Guard Extensions(以下 Intel SGX と表す) は Intel Corporation が提供する CPU の拡張アーキテクチャである．特権ユーザや端末管理者の信用性がない場合にも，データの機密性を保証した処理を行うためのものである．図 2 ように主記憶装置上にエンクレーブと呼ばれる暗号化された領域を作り，データの機密性を保ちながらプログラムの実行を可能にするため，信頼できる実行環境をハードウェアレベルで提供する．エンクレーブは信頼されない領域から，ECall を用いて呼ばれる．そして，エンクレーブ内で処理された結果が OCall を用いて信頼されない領域へと引き渡される．エンクレーブは特権システムソフトウェアを含むエンクレーブ外のコードが，エンクレーブの実行を検査改竄できない特別なモードで CPU によって実行され，ECall および OCall は，外部ソフトウェアがキャッシュされたアドレスを使用してエンクレーブの専用メモリへアクセスするのを防止するため，機密性を実現することが可能である．しかし，Intel SGX のエンクレーブにはサイズが 100MB 程度が限度であるという制約が存在する．このため，Ahemed らの Intel SGX を用いた既存研究 [8] では秘密鍵のみを保持し，それに関わる処理のみエンクレーブ内で行うといった方針を取っている．この既存研究と同様にエンクレーブで処理する内容は最低限としなければならない．

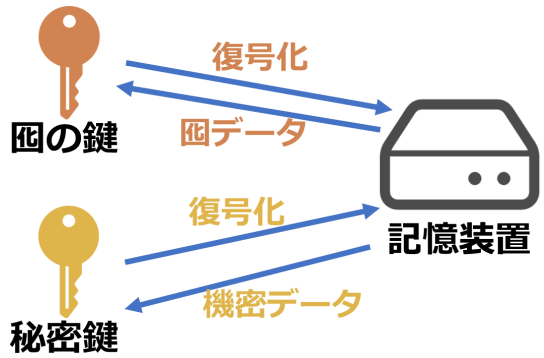

図 1 PDE の概要

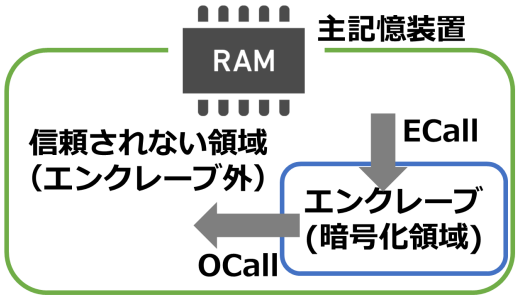

図 2 Intel SGX の概要

2.3 Filesystem in Userspace

Filesystem in Userspace(以下 FUSE と表す)[9] はユーザがカーネルを改変することなく独自のファイルシステムを

作成する機能を提供するシステムである．図 3 のようにインストールされた FUSE のカーネルモジュールが VFS とユーザ作成のファイルシステムの仲介を果たすため，ユーザはカーネルを改変することなくファイルシステムを作成することができる．

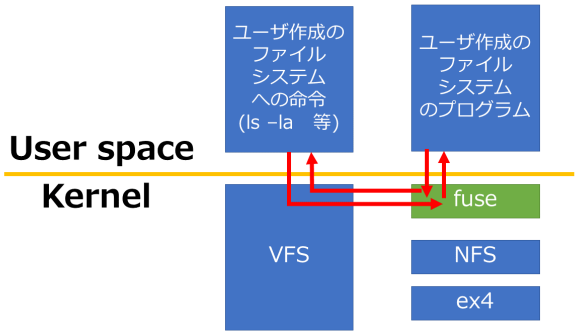

図 3　FUSE の構成

3. 脅威モデル

このセクションでは攻撃によって否認性が保証されない状況について述べる．いずれの攻撃シナリオにおいても攻撃者は永続記憶装置及び主記憶装置のスナップショットの取得が可能であると仮定する．

3.1 ユーザ使用のアプリケーションからの侵害

攻撃者は主記憶装置のスナップショットを得ることで PDE が稼働しているファイルシステム上であっても，外部アプリケーションが機密情報を使用している場合は，主記憶装置上に機密情報の平文を得られる．このため，攻撃者は外部アプリケーションが使用した機密情報の取得が可能であり，PDE の否認性が失われる．

3.2 ファイルシステムアクセス時の侵害

攻撃者は主記憶装置のスナップショットを得ることで，PDE ファイルシステムのアクセスパターンに違いがあることがわかる．例えば，ユーザが機密情報に READ アクセスしようとした際に，攻撃者が観測するファイルシステム上では空き領域にアクセスしようとするような事が想定される．この場合攻撃者はこの空き領域に機密情報の存在を疑うことができる．このことから，PDE の否認性が失われる．

3.3 永続記憶装置のスナップショットからの侵害

攻撃者が永続記憶装置のスナップショットを得る事で PDE の否認性が失われる可能性がある．例えば，攻撃者が観測可能なファイルシステム上では未使用領域であるにも関わらず，実際のバイトストリームでは何かが書き込まれていたりした場合に機密情報の存在が疑われる．

4. 研究の目的・課題

4.1 目的

本研究は PDE の考え方を用いて存在の否認が可能な暗号化ファイルシステムを信頼できる実行環境 (TEE:Trusted Execution Environment) を活用して実現する PTEE FS(PDE with Trusted Execution Environment File System) の提案を行う．ハードウェアによる TEE として Intel SGX を用いたシステムを提案する．

4.2 課題

PDE の概念及び TEE を用いた暗号化ファイルシステムの構築を行いその有効性を確認するために以下の課題を検討する．

[課題 1]3.1 節で述べた脅威モデル (以下脅威モデル 3.1) を分析し実現可能なシステム構成を導く

[課題 2]暗号化ファイルシステムの構造やその操作コードをメモリ検査攻撃に対して隠蔽する (脅威モデル 3.2,3.3)

[課題 3]ファイルシステムのベンチマーク [10] を用いて有効性を検証する

5. 実現可能なシステム構成

課題 1 について検討する．ユーザ空間で動作するアプリケーションの実行を暗号化ファイルシステムと同一ホスト上で許す場合のメモリ検査による攻撃 (脅威モデル 3.1) の自由度が高いため，以下の 2 つのシステム構成とし，対処可能なメモリ検査攻撃への前提を加える．ここで，ユーザ空間で動作するアプリケーションは平文データを必要とし，実行中にメモリ検査が行われた場合には自らのメモリ空間に展開された平文データを保護する手段を持たないものとする．

[常時検査型]任意の時点でのメモリ検査攻撃に対処するため PTEE FS サーバで平文ファイルを扱わない．メモリ検査に脆弱なユーザ空間で動作するアプリケーションは攻撃者から安全なクライアント側でのみ実行されるものとする．

[マウント時検査型]ユーザ空間で動作するアプリケーションの PTEE FS サーバでの実行を許容するために，メモリ検査が行われるのは隠し領域のマウント直後の時点のみとし，ユーザ空間で動作するアプリケーションからの隠し領域に属する情報の機密漏洩を排除できるものとする．

6. システム構成と仮定

システム構成と仮定について，常時検査型とマウント時検査型のそれぞれに関して述べる．提案手法はいくつかの仮定を置くことで PDE の否認性が成り立つ．ここでは既

存の PDE 暗号化手法 [5][6][7] 等でも必要としている前提を含め議論する．

6.1 常時検査型

常時検査型では，PTEE FS サーバが公開する名前空間を外部のクライアント端末が NFS[11] 等の遠隔アクセスプロトコルにてマウントして利用することを仮定する．

6.1.1 システムモデル

本システムを使用する環境としてクラウドプロバイダが提供する計算機であって，Intel SGX などの TEE が利用可能であることを仮定する．OS，ブートローダー等は全てマルウェアフリーである．本構成に固有な仮定として以下の常時システム仮定を置く．

常時システム仮定 1：提案システムが稼働しているサーバにアクセスする正規ユーザーは，必ずサーバをマウントしてデータを扱う．

6.1.2 攻撃者モデル

攻撃者は所有者に復号鍵開示の強要が可能である．攻撃者は復号鍵が明らかになった時点で鍵の強要をやめる．攻撃者はデバイスに対し何らかの方法でアクセスする方法があり，主記憶装置及び永続記憶装置のスナップショットを複数取得できる．攻撃者は本システムの構成を知っているが，復号化鍵及び復号化パスワードを知らない．本構成に固有な仮定として以下の常時攻撃者仮定を置く．

常時攻撃者仮定 1：この複数のスナップショットは攻撃者が任意のタイミングで取得が可能であると仮定する

常時攻撃者仮定 2：攻撃者はマウントをしているクライアント PC に対するアクセス権を持たない

6.2 マウント時検査型

マウント時検査型では，PTEE FS を搭載したサーバ上で機密情報の処理を行うことを想定する．以降，常時検査型の仮定との差分を述べる．

6.2.1 システムモデル

常時システム仮定 1 は仮定しない．以下，本構成に固有の仮定は以下の通り．

マウント時システム仮定 1：提案システムが稼働しているサーバは隠し領域に属する機密情報を扱う際に機密情報が含まれたボリュームをマウントする

6.2.2 攻撃者モデル

常時攻撃者仮定 2 は仮定しない．常時攻撃者仮定 1 を以下に変更する．

マウント時攻撃者仮定 1：主記憶装置におけるスナップショットは，隠し領域に属する機密情報を含んだボリュームがマウントされたタイミングでのみ取得が可能である

7. PTEE FS の設計

7.1 常時検査型における攻撃耐性を有するシステム構成

常時検査型における詳細なシステム構成に関して述べる．本構成では 5 節で述べたように PTEE FS は暗号化したままのファイルを操作し，平文ファイルそのものは扱わないためデータの暗号化や復号化は重要ではない．クライアントから提示された鍵によって囮領域と隠し領域とにアクセス先を振り分け，ファイルシステムの構造を切り替えることが必要になるため，その構造操作のコードへの主記憶装置のスナップショットによる漏洩と検査を防ぐ為，TEE を用いて処理を行う．個別のデータのやりとりについては NFS を用いてクライアントが PTEE FS にアクセスする分散システム構成とする．

7.1.1 鍵管理について

本構成では復号はクライアントで行われるため，PTEE FS との鍵のやりとりはマウント時におけるアクセス認証と認可に用いる．クライアントから送付された鍵が囮か真正かを判定し操作の切り替えを行う認可制御部は TEE を用いて保護し処理を行う．この時のシステム構成は図 4 のようになる．この図ではクライアントが Read コールでマウントした永続記憶装置にアクセルする時のデータフローを表している．クライアント端末で復号化や暗号化を行うシステムを FTEE FS Client と表し，サーバで鍵の判定及び操作の切り替えを行うシステムを　FTEE FS Sever と表す．クライアント端末を用いるユーザは図 4 のようにして暗号化されたデータをクライアント端末に受け取り，FTEE FS Client で復号化して用いるといった流れとなっている．

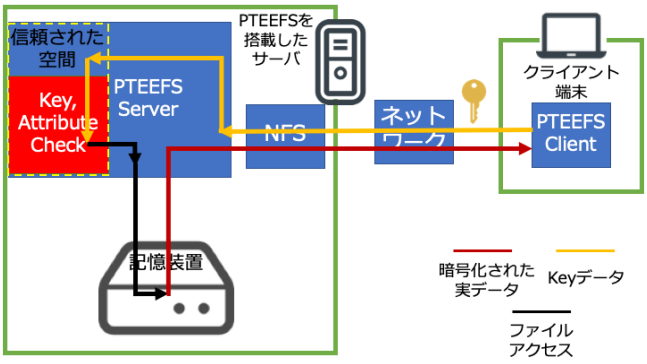

図 4　常時検査型におけて TEE を用いるデータフロー

7.2 マウント時検査型における攻撃耐性を有するシステム構成の考察

マウント時検査型における詳細なシステム構成に関して述べる．提案するファイルシステムでは，保護すべき情報

に関しては主記憶装置のスナップショットによる漏洩を防ぐ為，TEE を用いて処理を行う．ファイルシステムの構成には FUSE を用いて構成する．以下保護すべき観点に注目し提案手法のシステム構成を述べる．

7.2.1 暗号化・復号化の保護及び鍵管理について

機密情報を平文で扱う事は主記憶装置のスナップショットより情報の機密性が失われる事を意味する．この為，情報の暗号化及び復号化に関しては TEE を用いて処理を行う．鍵管理については実際に PTEE FS ホストでの暗号と復号に用いる他，常時検査型と同様の議論により，構造操作の切り替えを行う認可制御部は TEE を用いて保護し処理を行う．この時のシステム構成は図 5 のようになる．この図ではユーザが FTEE FS を搭載した端末に ssh クライアント等を用いて接続し，Read コールで永続記憶装置にアクセスした際のデータフローを表す．Read コールが行われると VFS や FUSE によって物理アドレスを得てで永続記憶装置にアクセスし，暗号化された実データを得る．暗号化された実データは FTT FS へと渡され，TEE において復号化される．その復号化されたデータをユーザが閲覧するといった流れとなっている．

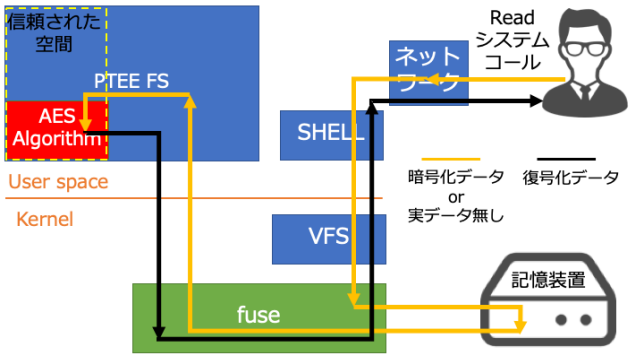

図 5 マウント時検査型において TEE を用いるデータフロー

8. 実験と評価

4.2 節で述べた課題 3 について部分的な評価を行う．今回は，FUSE を利用したファイルシステム構成における性能と Intel SGX で保護された手続き呼び出し性能の 2 つに関し検証を行う．

8.1 実験環境

Corei7-6700 とマザーボードの ASRock E3V5 WS，OS に Ubuntsu18.04LTS を搭載した計算機を使用した．

8.2 FUSE を利用したファイルシステム構成における性能の検証

8.2.1 実験方法

FUSE がファイルシステムへ及ぼす影響の検証として，ファイルシステムのコンパイルベンチマークを測定して検証した．FUSE で作成したファイルシステム上と Linux のファイルシステムである ex4 上で OpenSSH-8.0p1 の ビルド及び削除を 10 回行い実行時間を計測した．この実験で使用した FUSE ファイルシステムは既存のファイルシステムである ex4 のファイルシステムと同様の動作をするよう構成されたものである．実行時間計測には，Linux のコマンドである time コマンドを利用し計測を行なった．

8.2.2 実験結果・考察

FUSE で作成したファイルシステム上と Linux のファイルシステムである ex4 上で OpenSSH-8.0p1 の ビルド及び削除を 10 回行った実行時間を表 1 に表す．この表から FUSE ファイルシステムを用いた際でも大きなレイテンシがないことがわかる．

表 1 FUSE ファイルシステムと ex4 のコンパイルベンチマーク

ファイルシステム	Real	User	Sys
FUSE	6m39.646s	5m47.655s	1m22.637s
ex4	6m41.323s	5m47.406s	1m22.859s

8.3 Intel SGX で保護された手続き呼び出し性能

8.3.1 実験方法

Intel SGX がファイルシステムへ及ぼす影響の検証として，Intel SGX を用いたことによるプログラムのレイテンシの取得と評価を行った．エンクレーブ内に配置された何もしない関数を呼び出すプログラムをシングルスレッド版評価プログラムとし，Intex Software Guard Extensions SDK[12] で提供されるマルチスレッド対応ライブラリを組み込んだ上で何もしない関数を呼び出すプログラムをマルチスレッド版評価プログラムとした．実行時間の計測には，Intel 製 CPU の命令の 1 つ rdtscp を用いて CPU のクロックカウントを取得し計測を行なった．

8.3.2 実験結果・考察

マルチスレッドを含むプログラムによる測定結果を表 2 に表す．ここから，SGX を用いてマルチプロセスの処理を行うと大きなレイテンシがあることがわかる．

シングルスレッドのみのプログラムによる測定結果を表 3 に表す．ここから，シングルプロセスのプログラムの処理には今回の場合 1ms 程度のレイテンシがかかっていることがわかった．これに関しては Gjerdrum らの SGX を用いることによるレイテンシの計測結果 [13] と同様の結果が得られている．

表 2 と表 3 からマルチスレッドにおいては非常に大きなレイテンシが発生することがわかった．このマルチスレッドにおけるレイテンシに関しては今後調査していく必要があると考える．

表 2 FUSE ファイルシステムと ex4 のコンパイルベンチマーク

Intel SGX	クロックカウント [sec/3.4GHz]	実行時間 [ms]
有り	325938396	95.9
無し	8147382554	2396.3
有無による差	7821444158	2300.4

表 3 FUSE ファイルシステムと ex4 のコンパイルベンチマーク

Intel SGX	クロックカウント [sec/3.4GHz]	実行時間 [ms]
有り	1809916	0.532
無し	5392791	1.586
有無による差	3582875	1.054

9. おわりに

本研究は PDE の考え方を用いて存在の否認が可能な暗号化ファイルシステムの，信頼できる実行環境での実現の提案を行った．ハードウェアによる保護された実行環境として Intel SGX を利用した．本システムでは従来の PDE では想定されていなかった主記憶装置への攻撃に対して耐性のあるシステムにする事で PDE をクラウド上で利用する際にも PDE の否認性が保証を実現する．今回は 4.2 節における課題 1 に沿って脅威モデルを分析し実現可能なシステム構成を定義し，課題 3 に挙げた，本提案の性能面についての部分的な評価を行った．提案システムにおいて FUSE ファイルシステムを用いることによるレイテンシはほとんど存在せず，実用的な範囲であるという結果が得られた．Intel SGX を用いた構成について保護された手続きの呼び出し性能の初期検討を行った．

参考文献

[1] Canetti, R., Dwork, C., Naor, M. and Ostrovsky, R.: "Deniable Encryption", *Advances in Cryptology — CRYPTO '97* (Kaliski, B. S., ed.), Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 90–104 (1997).

[2] 総務省："総務省 | 令和元年版情報通信白書 | 企業におけるクラウドサービスの利用動向", `http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r01/html/nd232140.html`. (accessed 2019-10-22T23:57:56Z).

[3] Kumparak, G.: "米金融大手キャピタル・ワンで 1 億人の個人データが流出", `https://jp.techcrunch.com/2019/07/30/2019-07-29-capital-one-hacked-over-100-million-customers-affected/`. (accessed 2019-10-22T14:53:49Z).

[4] Anderson, R., Needham, R. and Shamir, A.: "The Steganographic File System", *Information Hiding*, Springer, Berlin, Heidelberg, pp. 73–82 (1998).

[5] Chang, B., Zhang, F., Chen, B., Li, Y., Zhu, W., Tian, Y., Wang, Z. and Ching, A.: "MobiCeal: Towards Secure and Practical Plausibly Deniable Encryption on Mobile Devices", *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 454–465 (online), DOI: 10.1109/DSN.2018.00054 (2018).

[6] Jia, S., Xia, L., Chen, B. and Liu, P.: "DEFTL: Implementing Plausibly Deniable Encryption in Flash Translation Layer", *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, New York, NY, USA, ACM, pp. 2217–2229 (online), DOI: 10.1145/3133956.3134011 (2017).

[7] Zuck, A., Shriki, U., Porter, D. E. and Tsafrir, D.: "Preserving Hidden Data with an Ever-Changing Disk", *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, HotOS '17, New York, NY, USA, ACM, pp. 50–55 (online), DOI: 10.1145/3102980.3102989 (2017).

[8] Ahmed, R., Zaheer, Z., Li, R. and Ricci, R.: "Harpocrates: Giving Out Your Secrets and Keeping Them Too", *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, Seattle, WA, USA, IEEE, pp. 103–114 (online), DOI: 10.1109/SEC.2018.00015 (2018).

[9] libfuse/libfuse: "Filesystem in Userspace", `https://github.com/libfuse/libfuse`. (accessed 2019-06-24T04:57:30Z).

[10] Traeger, A., Zadok, E., Joukov, N. and Wright, C. P.: "A Nine Year Study of File System and Storage Benchmarking", *Trans. Storage*, Vol. 4, No. 2, pp. 5:1–5:56 (online), DOI: 10.1145/1367829.1367831 (2008).

[11] University of Wisconsin–Madison: "Sun's Network File System (NFS)", `http://pages.cs.wisc.edu/~remzi/OSTEP/dist-nfs.pdf`. (accessed 2019-10-27T00:40:45Z).

[12] Zone, I. S. D.: "SDK — Intel® Software Guard Extensions", `https://software.intel.com/en-us/sgx/sdk`. (accessed 2019-10-26T12:24:11Z).

[13] Gjerdrum, A. T., Pettersen, R., Johansen, H. D. and Johansen, D.: "Performance of Trusted Computing in Cloud Infrastructures with Intel SGX:", *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, Porto, Portugal, SCITEPRESS - Science and Technology Publications, pp. 696–703 (online), DOI: 10.5220/0006373706960703 (2017).