クラウドにおけるPDEの考え方を用いたファイルシステムの検討

柴崎 凌我[1]　稲村 浩[2]　中村 嘉隆[2]

概要：インターネットのサービスにおいて，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，これだけでは鍵の開示の強要攻撃から保護ができない．機密情報の存在の否定を可能にする PDE(Plausibly Deniable Encryption) が提案されており，囮の鍵を開示することで鍵の開示の強要から利用者を保護することが可能になった．クラウドでの利用や仮想化技術の広まりによって実行時に主記憶装置が攻撃されることは検討すべき課題である．そこで我々は，PDE の考え方を用いた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) で実現する PTEE FS を提案している．本研究では攻撃者が囮鍵及び囮データの存在の知識に基づく攻撃への対策の検討を行う．PTEE FS の性能面については，クラウド上における実利用を模したモデルを利用し，クラウド上における実利用モデルとしてサーバ・クライアント間のファイル同期を用いて TEE 利用によるオーバヘッドが与える見積り性能の評価を行う．

1. 背景

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，PDE の既存研究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．主記憶装置への攻撃，メモリ検査攻撃は主記憶装置のスナップショットを不正に取得し，機密情報等を得る攻撃である．

総務省の白書 [2] によると図 1 からわかるように企業におけるクラウドの利用が約 60%となっており，図 2 よりデータの保管やバックアップ等がクラウドを利用して行われている事がわかる．このように，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，主記憶装置への攻撃も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．

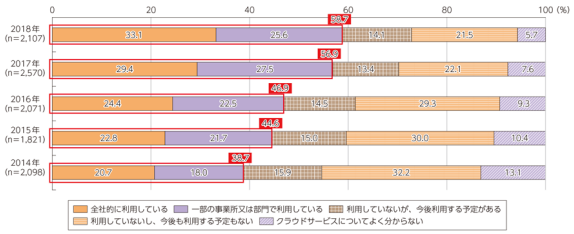
仮想化技術の発展として，このような特権ユーザや端末管理者の信用性がない場合にも，データの機密性を保証した処理を行うためにハードウェアでの支援機能が CPU に組込まれつつある [4].

これまで本研究では，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 を用いたシステムの提案 [5] を行った．

本稿では，PTEE FS(PDE with Trusted Execution En-

[1] 公立はこだて未来大学大学院 システム情報科学研究科
[2] 公立はこだて未来大学 システム情報科学部

vironment File System) の性能面について，よりクラウド上における実利用を模したモデルを利用し，TEE 利用によるオーバヘッドを考慮した性能の評価，並びに攻撃者が囮鍵及び囮データの存在を知っている前提で成立する攻撃への対策の検討を行う．PTEE FS の性能の評価に当たりクラウド上における実利用モデルとしてサーバ・クライアント間のファイル同期を用いる．



	集計企業数	比重調整後集計企業数	クラウドサービスの利用状況 (S) 利用している	全社的に利用している	一部の事業所又は部門で利用している	利用していない	利用していないが、今後利用する予定がある	利用していないし、今後も利用する予定もない	クラウドサービスについてよく分からない	無回答
全 体	2,119	2,119	1,236	697	539	751	297	453	119	13
[産業分類]										
建設業	299	91	59	39	20	29	16	14	2	0
製造業	371	567	307	172	135	236	96	141	22	1
運輸業・郵便業	316	194	98	48	50	82	24	58	12	1
卸売・小売業	296	452	287	165	122	134	63	72	24	6
金融・保険業	134	25	20	11	10	5	2	3	0	–
不動産業	132	31	22	16	6	7	3	4	1	1
情報通信業	243	105	89	63	26	15	11	4	1	–
サービス業、その他	328	654	354	183	171	241	83	158	56	3

図 1　企業におけるクラウドサービスの利用状況 (出典:文献 [2])

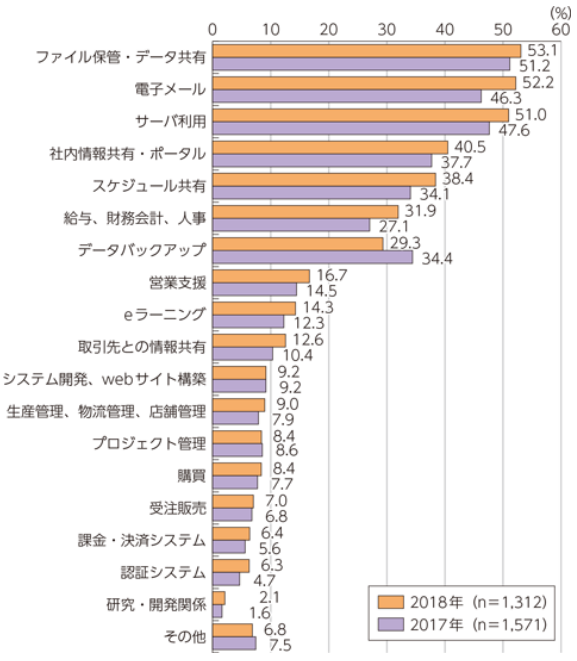


図 2　企業におけるクラウドサービスの利用内訳 (出典:文献 [2])

2.　関連研究と関連技術

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 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．これは，従来のディスク暗号化手法であるフルディスク暗号化では所有者が攻撃者から鍵の開示強要攻撃が起こった場合に保護が不可能であるという問題があるため，鍵の開示強要攻撃から所有者を保護する手法の 1 つである．PDE は囮鍵を用いる事による特性で鍵の開示強要攻撃から保護を可能にしている．PDE は図 3 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．所有者が攻撃者による鍵の強要攻撃にあった際，所有者は囮の鍵を攻撃者に開示する事ができる．攻撃者は囮の鍵を本来の秘密鍵であると思い込むため，攻撃者に囮の鍵と情報を与えることで，本来の情報の存在に気づかせない事が可能になる．一方，欠点として，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．さらに，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，これらを考え合わせると，実用的な方法とは言えない．しかし，PDE の基本的な考え方である，囮の鍵でアクセスを行うと囮の情報が得られ，秘密鍵でアクセスを行うと本来の機密情報が得られるという考え方は利用ができる．

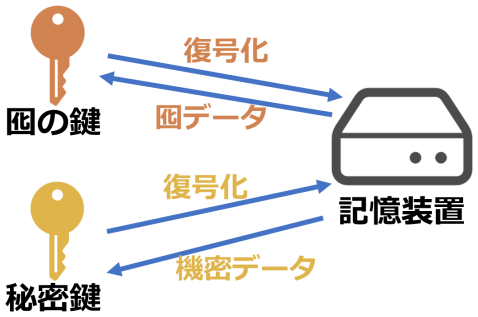


図 3　PDE の概要

この考え方を利用し，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 PDE 手法が Anderson ら [6] や Chang ら [7] 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．例えば，図 4 のように，画像ファイルのようなサイズの大きなファイルの一部に機密情報を埋め込み保存する．ステガノグラフィではこのような機密情報を埋め込んだファイルに変更を加えた場合に機密情報を上書きしてしまう危険性を持つ．このような上書きを回避するために機密情報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

率が悪化し，多くの機密情報を保持できない欠点を持つ．

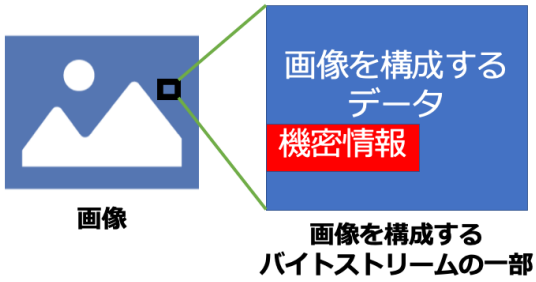


図 4　ステガノグラフィの例

隠しボリューム技術を利用した PDE は Jia ら [8] や Zuck ら [9] などによって提案されている．隠しボリューム技術では記憶装置上に，囮ボリュームを囮の鍵で作成し，隠しボリュームを秘密鍵で作成する．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．ユーザはシステムを利用する際，図 5 のように公開モードもしくは PDE モードでログインを行いファイルシステムを扱う．公開モードでは囮ボリュームのみを扱い，PDE モードでは隠しボリュームを扱うことができ，鍵の開示を強要された場合には所有者は公開ボリュームのログインパスワード及び囮の鍵を開示するため，攻撃者は隠しボリュームにアクセスできず機密データは保護される．隠しボリューム技術では，囮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．

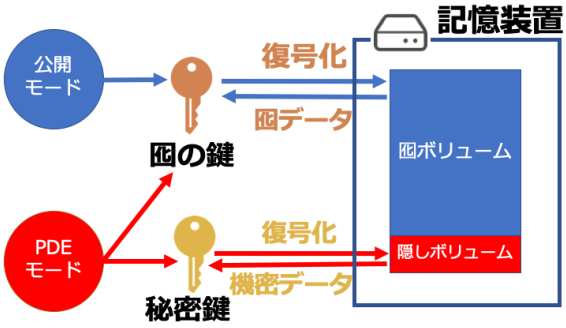


図 5　隠しボリュームシステムの概要

2.2　Intel Software Guard Extensions

Intel Software Guard Extensions(以下 Intel SGX と表す)[10] 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 6 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 (TEE:Trusted Execution Environment) をハードウェアレベルで提供する．この特性から Intel SGX はエンクレーブ内のプログラムとデータをメモリ検査攻撃から保護する事が可能である．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．これらの技術を利用するための環境として Intel Corporation は Intel Software Guard Extensions SDK[11] を提供している．しかし，Intel SGX のエンクレーブにはプログラムとデータのサイズが 100MB 程度が限度であるという制約が存在する．このため，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．Ahemed らの Intel SGX を用いた既存研究 [12] 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．Gjerdrum らの Intel SGX のパフォーマンスを測定した研究 [13] では，エンクレーブへ送信するバッファのサイズが 64kB を超えるとオーバヘッドが増大することが指し示されている．

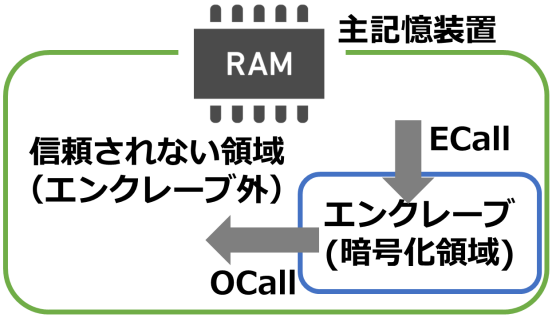


図 6　Intel SGX の概要

3.　目的

本研究の目的は鍵の強要攻撃に耐性を持ち仮想環境でのメモリ検査攻撃にも耐える分散ファイルシステムを実現することである．

3.1　これまでの我々の提案方式

これまでの我々の研究 [5] では鍵の強要攻撃に持つ分散ファイルシステムのプロトタイプについて以下のように設計してきた．

PTEE FS は暗号化したままのファイルを操作し，サー

バにおいて平文ファイルそのものは扱わずクライアント側でデータの暗号化や復号化を行う．サーバは囮空間と隠し空間の管理を行う．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(Trusted Execution Environment) を用いて保護し処理を行う．クライアントとのデータのやりとりについてはNFS(Network File System) プロトコルに必要な修正を加えて用いる分散システム構成とする．そのため，本構成はユーザアプリケーションからの侵害への耐性を有する．

本構成においては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．TEE として Intel SGX を用いるため，攻撃者が特権ユーザや端末管理者である場合についても，これらの構造操作のコードの機密性が保持できる．そのため，本構成はファイルシステムアクセス時における侵害への耐性を持ち得る．

Jia ら [8] のようにファイルシステム上で空である領域に対し，ランダムなビットで埋める等の処理を施すことによって永続記憶装置のスナップショットからの侵害への耐性を得ることが可能である．

3.2 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 7 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 7 のように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．

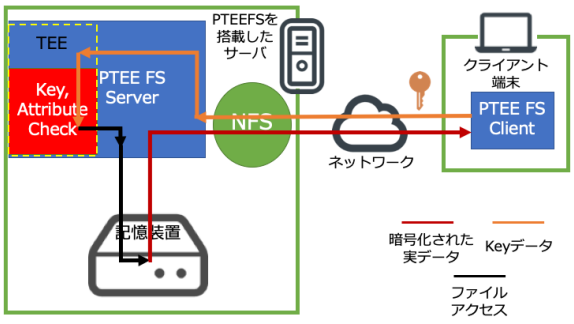


図 7 提案システムにおいて TEE を用いるデータフロー

4. 研究課題

囮鍵の開示による知識を用いた攻撃に対して一定の安全性を持つ設計を与え，クラウドサービスに適用した場合の性能見積もりから実用性の見通しを得る．我々がこれまで検討してきた攻撃方法に加え，これまで検討されてこなかった囮鍵の開示による知識を用いた攻撃について述べる．次に，本システムの実装の形態について説明し，クラウドストレージサービスでよく見られるファイル同期サービスにおけるアクセスパターンで稼動させた場合の性能見積りを行う．

4.1 囮鍵の開示による知識を用いた攻撃

囮鍵の開示による知識を用いた攻撃とは，囮鍵が開示された攻撃者は囮領域への自らのアクセス情報の時系列をネットワークトラフィックあるいはサーバへのメモリ検査攻撃にて取得できるとした場合，同じ前提から正統な利用者による隠し鍵による隠し領域へのアクセス情報の時系列がを取得できることになり，これらを比較照合することにより隠し領域の存在が露呈することである．

PTEE FS において囮鍵の開示による知識を用いた攻撃に関して，信頼できる空間である TEE のインターフェースにおいてどのように攻撃が成立するか，それらをどのように保護するかについて考察を行う．TEE のインターフェースにおけるデータフローを図 8 に表す．インターフェースとしてネットワーク・TEE 間と永続記憶装置・TEE 間の 2 つが存在する．それぞれのインターフェースにて観測可能な情報について以下のように定義する．

TS1 ： (TimeSeries1) ネットワーク・TEE 間の操作時系列で，修正された NFS プロトコルのやりとりが観測される．

TS2 ： (TimeSeries2) 永続記憶装置・TEE 間の操作時系列で，永続記憶装置への fetch や store などの操作列が観測される．

TS1 と TS2 で観測される内容の例を以下の表 1 に示す．

表 1 TS1 及び TS2 の例

TS1	Send	Recieve
	getattr File	(OK, Error) Result
	getattr Dir	(OK, Error) Result
	readdirplus Dir	(OK, Error) Result
	write File data	(OK, Error) Result
	read File	(OK, Error) Result data
TS2	Send	Recieve
	fetch ObjectID data	Result ObjectID (OK, Error)
	store ObjectID data	Result ObjectID (OK, Error)

TS1 は図 8 において青色の線で表され，TS2 は橙色の

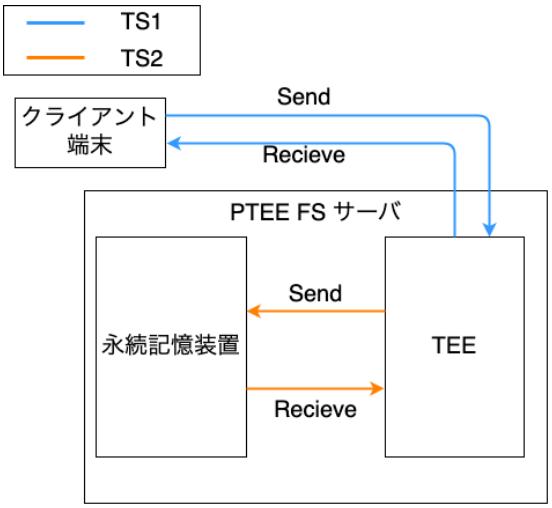


図 8　TEE のインターフェースにおけるデータフロー

線で表される．TS1 及び TS2 について攻撃者が任意に発生させたものを TS1_m，TS2_m(m:malicous) と表し，正統なユーザの操作によって発生させたものを TS1_l，TS2_l(l:legitimate) と表す．この時 TEE のインターフェースにて観測可能な情報から以下の 2 つの攻撃が考えられる．

攻撃可能性 1：　TS1_m と TS1_l の差異を攻撃者が観測することで隠し領域の存在が露呈する

攻撃可能性 2：　TS1_m の操作結果として TS2_m が外部観測可能な場合，TS2_m と TS2_l のペアの同一性から TS1 同士の一致判定が可能となり隠し領域の存在が露呈する

攻撃可能性 2 の成立条件として以下の 2 つの仮定を置く．

攻撃者仮定 1：　TS1 と TS2 の対応関係

TS2 = TEE_exposed_func(TS1)

が推定可能である．これは NFS RPC の時系列から永続記憶装置への操作系列の対応づけが可能であることを意味する．

攻撃者仮定 2：　TS2 の要素同士の一致判定が可能である．つまり永続記憶装置への操作同士が識別可能で同一性が観測できることを意味する．

従って，攻撃者から提案手法にて機密情報を保護するには以下の 2 つが必要となる

(1) 攻撃可能性 1 を成立させない

(2) 攻撃者仮定 1 または攻撃者仮定 2 のいずれかを不可能にする事で攻撃可能性 2 を防御する

4.1.1　攻撃可能性 1 について

TS1 を構成する要素であるパケットの RPC のペイロード部分を暗号化することで，データサイズ以外について違いが観測不可能になり，攻撃者可能性 1 の発生を防ぐことができる．

4.1.2　攻撃可能性 2 について

攻撃可能性 2 について，攻撃者仮定 1 を成立させないため次のようなシステム構成をとる．攻撃可能性 1 の対策と同様にパケットの RPC に関する部分については暗号化を行う．TS2 に関しても永続記憶装置に格納されるデータそのものについては暗号化を行うこととする．本構成では永続記憶装置側は一般のディスクあるいは通常のファイルシステムを想定しているので，永続記憶装置で対象を指定する際に用いるオブジェクト ID はメモリ検査攻撃から保護されない．この時に攻撃者が得られる情報はオペレーションとオブジェクト ID，TEE への入出力タイミング，暗号化部分のサイズである．TEE への入出力におけるオブジェクト ID の出現のタイミングの相関を崩すために例えば囮領域のアクセス中に隠し領域の ID へのアクセスを混入させる手法が考えられる．この手法を実現するには以下の 2 つの問題について解決する必要がある．

(1) 操作が必要な囮領域の操作対象のオブジェクト ID が与えられた際の，目くらましに操作列に混入させる隠し領域のオブジェクト ID の適切な選択

(2) 追加された隠し領域のオブジェクト ID への操作列を尤もらしく見せること

4.2　クラウドサービスに適用した場合の性能

提案方式の実現モデルに対して，クラウドストレージサービスでよく見られるファイル同期サービスにおけるアクセスパターンで稼動させた場合の性能見積りを行う．我々は保護が必要な NFS RPC 一回呼び出し当りの処理時間へのオーバヘッドの算出モデルとして (1) 式を得た [5]．

OH(ms) は追加応答時間であるオーバヘッドを表し，i(MB) はエンクレーブ呼び出しの際の転送バッファのサイズを表す．o(MB) はエンクレーブ脱出の際の転送バッファのサイズを表す．オーバヘッドはエンクレーブ呼び出しの際の転送バッファが 64kB を超える時に増大することがわかっているため，オーバヘッドの計算時には i のサイズを 64kB で区切り，算出することとする．提案システムをクラウドサービス上で適用した場合に，TEE の利用の有無による応答時間の差異から，ファイルシステムの性能に与える影響を評価できることがわかる．

クラウド利用の応用に見られる実用的な例としてファイル同期が挙げられる．そこでこれを改めて評価対象とし，具体的なファイル同期システムとして RSYNC[14] を利用する．RSYNC の実行トラフィックをトレースし，やり取りされるサイズを (1) 式の引数として与え，見積もりのオーバヘッドを算出し，その影響を評価する．

我々のこれまでの研究ではプログラムのビルドを負荷としたベンチマークで評価を行ったが，クラウドで最も多く用いられる応用であるファイル共有で一般的に見られるサービスであるファイル同期を評価対象に加えることにした．

$$OH = \begin{cases} 0.0097 + 0.0354o + 0.0115 & i < 0.064 \\ 0.9560i + 0.0354o - 0.0002 & i >= 0.064 \end{cases} \quad (1)$$

5. 提案手法

4.1.2 節で述べた 2 つの問題を解決するには囮領域と隠し領域の間で，攻撃者から TS2 で観測される永続記憶装置での指定に用いるオブジェクト ID をできるだけ同じにすれば良い．これを実現するために，任意の隠し領域に存在するファイルは，図 9 のように囮領域のどれかのファイルに内部的に埋め込まれているようにする．ここで囮領域のみを扱うシステムからは図 9 における隠しファイルは空き領域として認識される．

図 9 隠しファイルの囮ファイルへの埋め込み

同様にある隠し領域のひとつのディレクトリは，囮領域のどれかのディレクトリの内部に埋めこまれているようにする．前提として，実際に読みこまれた永続記憶オブジェクトに囮側もしくは隠し側データがどこに存在し，今どちらをアクセスすべきかは永続記憶オブジェクトの暗号化された領域に記録され TEE 内で安全に確認・操作できるものとする．ディレクトリエントリにはこれを指定するフラグを設け，値を D(Decoy, 囮領域), H(Hidden, 隠し領域) とする．

この方式の実現にあたり 2 つの手続きを提案し利用する．

手続き 1： トップディレクトリで与えられた 2 つのディレクトリツリーを，できるだけ相似するもの同士でマージする

手続き 2： ディレクトリツリーへの更新に伴うマージからの逸脱を，囮側と隠し側で別に管理する

手続き 1 に関して，図 10 のようにファイル及びディレクトリの区分とサイズについてできるだけ相似するものでマージを行う．例えば隠し領域空間が一時的に大きくなった場合，囮領域に埋め込まれていない隠し領域のファイルが存在することとなる．そのためファイルのメタデータに，隠し領域のファイルのみ含まれるか，囮領域のファイルのみ含まれるか，その両方かが区別がつくように H,D フラグにて管理する．

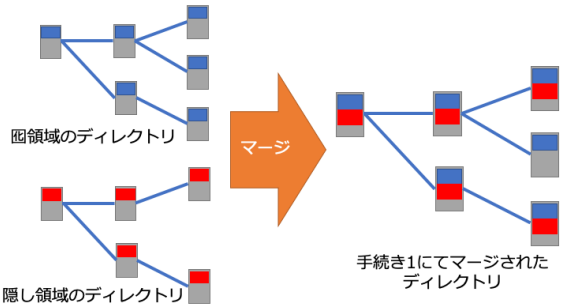


図 10 囮領域と隠し領域のディレクトリのマージ

ファイルの追加や削除があった場合に，手続き 1 で実現したマージによる埋め込み状態から外れてしまう．そのため，手続き 2 はファイル内部にマージされた埋め込み状態はそのままに，この変更の有無を，(UM:Un-mergerd, マージされていない状態を示す) フラグをディレクトリエントリに付けて管理を行う．追加されたディレクトリエントリが示す新規作成されたファイルはこの時点で内部の囮あるいは隠し領域の片方にのみしかデータが存在していないので UM フラグを付与する．ファイルの削除があった場合、ディレクトリエントリで指示されるファイルの中身で埋め込まれたもののうち、削除されていない側が残るので UM フラグでこの状態を記録する．追加や削除が起った場合に可能なら適切なファイルにマージする．同じディレクトリのファイルのマージ状態を H,D フラグで確認し反対側の領域にのみデータが存在するファイルあればそれを用いる．解消された UM フラグを消去する．あるディレクトリが保持するエントリの殆どが H,D フラグのどちらか片方、つまり片側の領域のファイルのみで構成される状態になった場合はディレクトリそのものの再マージが必要なため、親ディレクトリにて UM フラグで記録する．

これら 2 つの手続きを用いて PTEE FS で囮領域に隠し領域のファイル空間を埋め込む流れを説明する．

初期状態からの立ち上げ まずルートディレクトリを作り，手続き 2 運用下で囮側と隠し側でそれぞれユーザのアクセスを受け入れる．ある程度変更点が両側で溜ったら，ルートディレクトリから UM フラグを探索して最初に見付けたディレクトリをトップとして手続き 1 を

実行する．これを名前空間全体に実行し，埋め込み状態を作成する．

維持　追加削除が多く発生するとマージが最適ではなくなるので，定期的にディレクトリのトップから手続き 1 を実行する

6. 実験と評価

4.2 節のクラウドサービスに適用した場合の性能について本稿では，TEE の利用がファイルシステムの性能に与える影響を増加する応答時間の差から評価を行う．本実験では性能の評価に用いるベンチマーク負荷としてクラウドでのファイル同期を想定し，RSYNC によるクライアント・サーバ間のファイル同期を行った．TEE を用いることによる応答時間の増加をオーバヘッドとし，ファイル同期が完了するまでの応答時間と比較し評価を行う．

提案システムの実行時オーバヘッドを把握するために，実利用を模した状況でのファイルサーバの応答時間と，トレースデータを基にした TEE 利用による応答時間の増加の見積りを比較する．主要な機能のうち，暗号化はクライアントにて行われるので無視し，ファイル操作と鍵判別のための処理，それらのアクセスパターンを保護するための TEE 利用にかかる応答時間の増大を評価対象とする．まず，NFS を利用したベンチマーク負荷実行時のタイムスタンプつきネットワークトレースを取得する．トレースデータから RSYNC 実行時の代理サーバによる NFS 遠隔手続呼び出しを抽出し，その引数サイズから TEE の入出力データサイズを求める．NFS 手続き毎に 1 回の TEE 呼び出しを行うものとして，入出力サイズに対する TEE 利用時の応答時間を求める近似式から増加する応答時間を算出する．

6.1 実験方法

RSYNC 通信が可能なサーバ・クライアントを用意し，ファイルの同期を行った．RSYNC のサーバ構成が図 11 ようであると仮定し，会社や学校では主にファイルサーバを直接利用して作業を行うが，リモートワークのために自宅にあるクライアント端末にて変更を同期する状況を想定する．変更されたファイルのアップロードとダウンロードに読み込まれるデータのサイズは同じであると仮定し，ここではアップロードするクライアント・サーバ間のトラフィックをトレースする．データ取得を行う際の構成を図 12 とし，代理サーバでサーバをマウントし，クライアントから代理サーバに rsync を行い，代理サーバ・サーバ間の NFS のトラフィックを観測した．トレースデータから，RPC の READ，WIRTE コール引き渡される一連のバッファサイズを取得する．その一連のバッファサイズを，(1) 式上でエンクレーブの呼び出しと脱出のバッファサイズとして利用し，実行時間を計測し本システム利用時の Intel SGX のオーバヘッドとして評価を行った．処理全体の応答時間の測定は同様のサイズのファイル同期に関して 10 回行い，その平均の処理時間を利用した．RSYNC で同期で変更されるファイルは表 2 に示されるファイル群を用いた．テキスト形式のファイルの平均サイズは 11263 byte であり，バイナリ形式のファイルの平均サイズは 216269 byte であった．ファイル群のうちファイルサイズが TEE の応答時間が増大する 64kB を超えているのは 4 つである．

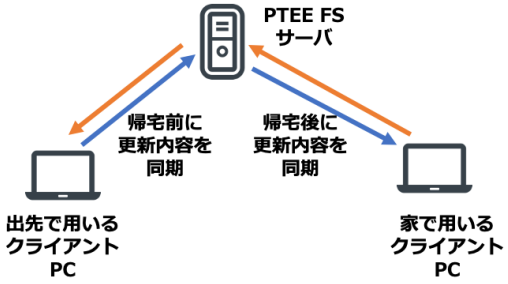


図 11　実験で想定するユースケース

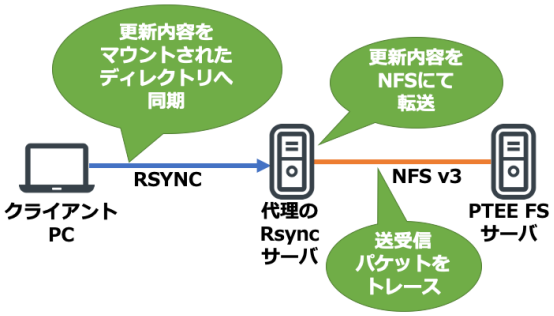


図 12　実験におけるサーバ構成

表 2　実験に用いたファイルのサイズと種類

ファイルサイズ (byte)	ファイルタイプ	ファイルサイズ (byte)	ファイルタイプ
55358	テキスト	1444438	バイナリ
22113	テキスト	223065	バイナリ
11602	テキスト	97252	バイナリ
11101	テキスト	96516	バイナリ
4284	テキスト	35184	バイナリ
2651	テキスト	33033	バイナリ
2607	テキスト	10244	バイナリ
2317	テキスト	6527	バイナリ
567	テキスト	165	バイナリ
28	テキスト		

6.2 実験環境

実験のサーバ及びクライアントには RSYNC[14] 及び，NFS Version 3[15] がインストールされた計算機を用いた．トラフィックのトレースには Wire Shark[16] を用いた．実験環境においてのネットワークの帯域は 6.90MB/s であった．

7. 実験結果・考察

実験環境下でのクライアント・サーバにて処理全体の応答時間の平均は 18336.2ms であり，標準偏差は 2373 であった．RSYNC 実行時における NFS トラフィックのトレースに現れる一連の転送データサイズを図 13 に示した．いずれも [5] で示された応答時間が増大する 64kB 地点を超えていないことが確認できる．この転送データサイズの系列を NFS 一回呼び出し当りのオーバヘッド (OH) となる応答時間の増加を求める式 1 に代入し，系列の総和から 1.9ms となる．このため，提案システムを用いた際の見積もり処理時間は 18338.1ms である．よって，処理時間に占めるオーバヘッドの割合は 0.010% であり TEE を用いる際にかかるオーバヘッドは許容されると考える．

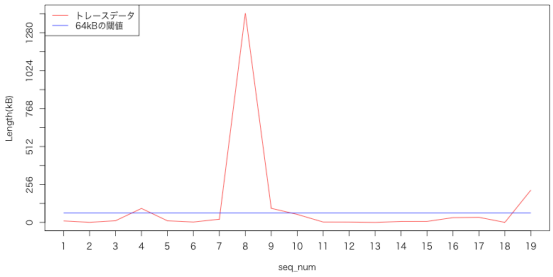


図 13 rsync 負荷時に観測された転送データサイズと NFS Call の系列の一部

8. おわりに

これまでの我々の提案方式 [5] である PTEE FS に対し，囮鍵の開示による知識を用いた攻撃への耐性を得るためのシステムの設計の検討を行い，PTEE FS の性能面に関してよりクラウドサービスに適用したユースケースにて実験並びに評価を行った．rsync によるファイル同期環境にて，処理における TEE 利用による増加応答時間の測定を行い，提案システム上において処理時間の見積もりにおける TEE 利用による応答時間の割合は 0.010%であり，TEE を用いる際にかかるオーバヘッドは許容される可能性があることがわかった．

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 2019-10-22).

[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 2019-10-22).

[4] 濱本亮，佐々木貴之，森田佑亮，三好一徳，小林俊輝："Trusted Execution Environment 搭載デバイスのためのセキュア空間へのアクセス権限設定法の基礎検討"，コンピュータセキュリティシンポジウム 2017 論文集，Vol. 2017, No. 2 (2017).

[5] 柴崎凌我，稲村浩，中村嘉隆："PDE の考え方を用いた暗号化ファイルシステムの検討"，第 82 回情報処理学会全国大会講演論文集，Vol. 82, No. 1, pp. 103–104 (2020).

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

[7] 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).

[8] 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, Dallas, Texas, USA, ACM, pp. 2217–2229 (online), DOI: 10.1145/3133956.3134011 (2017).

[9] 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, Whistler, BC, Canada, ACM, pp. 50–55 (online), DOI: 10.1145/3102980.3102989 (2017).

[10] Intel: "インテル® ソフトウェア・ガード・エクステンションズ (インテル® SGX)", `https://www.intel.com/content/www/jp/ja/architecture-and-technology/software-guard-extensions.html`. (accessed 2020-02-13).

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

[12] 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).

[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).

[14] rsync: "rsync", `https://rsync.samba.org/`. (accessed 2020-05-08).

[15] Callaghan, B., Pawlowski, B. and Staubach, P.: "NFS Version 3 Protocol Specification", `https://www.ietf.org/rfc/rfc1813.txt` (1995). (accessed 2019-12-24).

[16] WIRESHARK: "Wireshark", `https://www.wireshark.org`. (accessed 2020-02-13).