機密情報の二次漏えいを防止するための プライベート・セキュア空間の提案

多田政美† 古市実裕†

近年，企業や組織内部からの機密情報流出に関する対策が不十分な状況にある．内部からの情報流出は件数が少なくても，その情報の正確性や一度に大量の情報が流出することにより，被害総額が多額になる傾向にある．特に，自社からの直接的な漏えいではなく，業務依頼先などからの間接的な漏えい防止に関しては，セキュリティ教育の徹底やセキュリティ製品の強制的な導入が困難で，大きな課題となっている．以前，ローカル HDD 上で排他的にアクセスを制御するセキュア空間と呼ぶ特殊な領域を作り出すことで，間接的な漏えいの防止策の実現を試みたが，運用上の制約や耐攻撃性に課題を残していた．本論文では，これらの制約や課題を解決するため，揮発性メモリや暗号を用いたプライベート・セキュア空間という手法で，情報の漏えいを防ぐ新たな方法を提案する．さらに，提案したシステムを実際に構築し，商用アプリケーションと組み合わせて定量的な性能評価を行い，その有効性を示す．

A Proposal of a Private Secure Area to Prevent Secondary Leakage of Confidential Information

MASAMI TADA† SANEHIRO FURUICHI†

1. はじめに

近年，ウィルス対策ソフトなどの浸透により，外部からの攻撃には十分な対策が取られるようになってきたが，未知のウィルスに対する脅威や，企業や組織の内部からの情報流出に関しては，不十分な状況にある．実際，2013 年に発表されたセキュリティの 10 大脅威[1]のひとつに内部犯行がランクインしており，さらに，11 位には，内部から発生する情報漏えいも候補に挙がっている．内部からの情報漏えいは，外部からの脅威に比べ，発生件数は少ないものの，その被害額は同等であるという統計[2]もあり，一件当たりの被害総額が大きくなる傾向にある．また，内部犯行では，機密性の高い情報を適確に大量に持ち出すことが可能であり，企業にとっては看過できない喫緊の課題である．情報セキュリティインシデントに関する調査報告書[3]によると，最も件数の多い情報漏えいインシデントは，不正な情報持ち出しである．さらに，情報漏えいの総件数のうち，3 割以上が誤操作に起因すると報告されており，不正を行う意思が無くとも，ヒューマンエラーにより情報漏えいが起きやすいことを示している．

さらに，昨今は専門業務の分業化が進んでおり，国内外に関わらず，他企業へ業務の一部を依頼する機会が増えている．自社の社員に対しては，セキュリティ教育などを通じて，不正や誤操作を防止する対策も施せるが，発注先の他企業に対して，発注元である自社の施策を徹底することは難しい．

本論文では，自社からの直接的な漏えいではなく，発注先の企業から間接的に自社の機密情報が漏えいすることを「二次漏えい」と定義する．著者らは，二次漏えいに対する一つの現実的な解法を以前示した[4]が，運用上の制約や耐攻撃性等の面で課題が残されていた．これらの課題を解決し，より実用的で安全性の高い新たな手法を提案する．また，提案手法を実装し，商用アプリケーションを用いて性能評価を行い，その実用性を示す．

2. 従来技術と問題点

セキュリティ強化のため，業務委託先の企業に対してシステムの大規模な再構築や，指定したアプリケーションの強制的な導入を要求することは，実際の現場では容易に受け入れられない．本章では，二次漏えいという観点から既存のセキュリティ対策技術の課題を検討する．

2.1 シンクライアント

PC からの情報漏洩を防止するために，あらゆるデータの処理をサーバー上で実行し，エンドユーザーの端末では画面表示と入力デバイス処理だけを行うシンクライアントが提案されている[5][6]．仮想化技術の進展に伴い，シンクライアントの実装方法の一つである VDI (Virtual Desktop Infrastructure)も普及しつつある[7][8]．

しかし，シンクライアントは自社内の閉じた環境で利用する形態が一般的であり，二次漏えいの防止を目的として，自社のシンクライアント環境を業務委託先の他社と共有する利用形態は現実的ではない．また，一般的なオフィス環

† 日本アイ・ビー・エム（株）ソフトウェア開発研究所
Software Development Laboratory, IBM Japan Ltd.

境において，従来の PC 環境に比べ，VDI 環境のユーザー当たりの総コストの方が高くなるという指摘もある[9].

2.2 Enterprise DRM

データとセキュリティポリシーを関連付けし，アプリケーション上での操作や閲覧制限を行う文書管理システムが存在する. 代表的なものに, Adobe LiveCycle[10]や Microsoft Windows Rights Management Services[11]がある. アプリケーションプログラム自身が編集権限や印刷権限などのセキュリティポリシーを解釈し，プログラムの操作を制限する.

しかし，アプリケーションがセキュリティポリシーを解釈できることが前提であり，現状では Microsoft Office や Adobe PDF などの一部の製品に限定される．企業で利用する業務ソフトや作り込みのアプリケーションの多くは対応しておらず，システムを導入するためには，業務ソフトの全面的な置き換えや業務形態の大幅な見直しが必要になる.

また，近年はクラウド環境上で文書を編集し，共有できるサービス[12][13]も普及している. 適切なアクセス権や編集権限を設定することで，業務委託先と容易に文書を共有できる利便性もあるが，利用可能な文書形式に限りがある上，社内規定で社外クラウドの業務使用を禁止している企業もあり，二次漏えい防止の決定的な解決策にはなっていない.

2.3 セキュア OS

SELinux[14]や AppArmor[15]などのセキュア OS は，管理者の設定に基づき機密情報へのアクセス制御を厳密に管理する強制アクセス制御機構を備えている.

しかし，現状では，多くの企業が Windows ベースのアプリケーションを使用しており，それらの業務ソフトを全面的に置換する必要がある．さらに，操作性も大きく変わるため，一般業務で本格的に用いるには敷居が高い.

セキュリティポリシーの設定作業が煩雑となりがちという課題もある．使いやすいポリシーエディタ[16]や少ない手間による情報漏洩防止機構[17]なども提案されているが，業務依頼先の企業に OS 全体のポリシーを強制することは難しい.

2.4 セキュリティ対策アプリケーション

市販のセキュリティ対策アプリケーションを導入することで，ファイル操作などをリアルタイムに監視して，機密データなどが複製されることを防ぐ，あるいは複製したことを記録としてログに残すこともできる．また，外部メディアなどに機密データを暗号化して保存するアプリケーションも数多く存在する[18][19].

しかし，発注者が指定したアプリケーションを業務依頼先に強制的に導入すると，業務依頼先の既存のセキュリティ製品と競合する可能性が大きく，実現が難しい.

3. 従来技術の課題に対する施策とその課題

従来の PC のセキュリティ対策手法は，社内のセキュリティを守るためのものであり，かつ，既存の業務ソフトや IT 資産の有効活用が難しく，業務形態の大幅な変更やユーザーの操作性の大幅な低下を招くことが多いため，二次漏えい対策のために導入するには前章で述べたような課題が存在している.

それらの課題に対応するために既存の業務形態や IT 資産を維持したまま，ユーザーが容易に扱えるセキュリティポリシーを遵守させることで二次漏えいを防ぐ方法として，ローカル HDD 上で排他的にアクセスを制御するセキュアな空間を作り出し，その中で商用アプリケーションを使ってファイルの閲覧や編集を行う手法を提案した[4]．この手法では，暗号化されたアプリケーションのデータと，セキュア空間を作り出すための API 監視モジュールを一つのパッケージファイルに梱包して配布する．データを編集する際は，図 1 に示すように，API 監視モジュールをロードして，HDD 上にセキュアな空間を作り出してから，データを展開する．詳細な手順を以下に示す.

(1) パッケージ管理プログラムを起動する.
(2) PC のシリアル番号などの PC 固有情報を取得する．固有情報が一致しない場合，終了する.
(3) 暗号化されたデジタルデータを復号するためのパスワードをユーザーに入力させる. パスワードが一致しない場合は終了する.
(4) API 監視モジュールを展開する.
(5) API 監視モジュールをすべてのプロセスに対して注入し，監視を始める．この時点でセキュア空間（保護領域）が作成される.
(6) アクセス制御ポリシーとデジタルデータを復号してセキュア空間に展開する.
(7) ポリシーを API 監視モジュールに適用する.
(8) ポリシーで許可されたプロセスは，セキュア空間内のデータに対してアクセス可能になる.

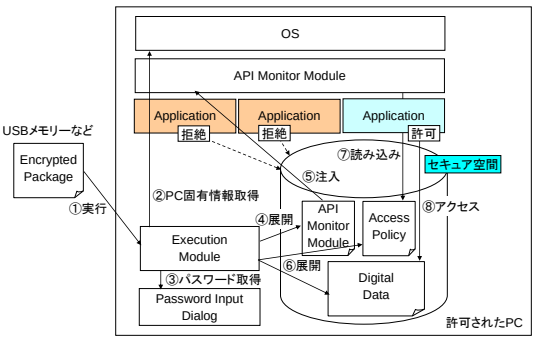

図 1 セキュア空間の概念図

セキュア空間とは，許可されたアプリケーション以外がファイルにアクセスできない領域である．例えば，C:¥shelter というフォルダをセキュア空間と定義し，さらに，

Adobe Reader[20]のみがこのフォルダへのアクセスを許されると定義したとき，Explorer や他のアプリケーションはこのフォルダにアクセスできないため，このフォルダから別のフォルダへ，ファイルを持ち出すことができない．一方，Adobe Reader には，このフォルダへのアクセスを許可すると同時に，このフォルダ以外へのアクセスを禁止することにより，Adobe Reader 自身を使用してファイルを持ち出すことも不可能になる．さらに，クリップボードへのコピー&ペーストの禁止（スクリーン・コピーも含む）やプリンタ出力の禁止なども同時に適用できるため，様々な経路からの情報漏えいを防止できる．

セキュア空間では，マルチユーザー環境への対応は行っていない．そのため，別なユーザーを介した情報漏えいを防ぐために，セキュア空間を透過的暗号化フォルダ（フォルダの暗号化機能）上に構築することを推奨している．また，PC の異常終了や強制終了などが起こった場合，セキュア空間にデータが取り残される問題がある． PC の再起動時にセキュア空間のデータを削除することにより，機密情報を保護するが，状況に応じて完全に削除できない可能性が残る．この問題を防ぐために， セキュア空間を RAM ディスク上に構築することを推奨している．ただし，これらの解決策は必ずしも万能ではない．

3.1 透過的暗号化フォルダ

Windows では OS の標準機能として, EFS (Encrypting File System)[21]でフォルダを暗号化できる．EFS で暗号化されたフォルダにデータを展開することにより，別なユーザー ID でログインした場合でも，セキュア空間上のデータを保護できる．しかし，EFS で暗号化されたフォルダに複数のユーザーを登録できるため，複数のユーザーが登録されたフォルダにはセキュア空間を構築できないようにするなどの制限が必要となる．さらに，システムを故意に異常終了させた場合や電源を強制的に切った場合，再起動後にデータを削除する前に，悪意あるプログラムによって情報が持ち出される可能性が残り，完全とは言えない．

3.2 RAM ディスク

RAM ディスク上にセキュア空間を展開した場合，異常終了や強制的な電源切断に対しても，データを守ることができる．別ユーザーからデータを保護するために，透過的暗号化フォルダと併用することが望ましい．ただし，一般に RAM ディスクを使用するためには，管理者権限を取得してデバイスドライバをインストールする必要があり，発注先の企業にそれを強制することは難しい．二次漏えい対策では，OS の再起動なしにセキュア空間を構築できることが必須要件の一つとされている．

4. プライベート・セキュア空間手法の提案

前章で述べたとおり，復号したデータを HDD 上に書き出すと，異常終了や強制終了によってデータが HDD に残存し，機密情報が漏えいする危険性が高まる．RAM ディスクや透過的暗号化フォルダと組み合わせることで，その危険性を制限のもと防ぐことができるが，あらゆる環境で汎用的に使える手段ではないため，不完全であった．

そこで，本論文では，セキュア空間の安全性を高めるために二つの新たな手法を提案する．図 2 はその概念図である．

第一の手法は，指定されたアプリケーションからしかアクセスできないプライベート（排他的）なデータ領域を RAM 上に確保し, そこに復号したデータを配置した上で，ファイルへのアクセスをリダイレクトする． RAM 上に展開されたデータは，当然のことながら，電源が切断された時点で自動的に消滅するので，異常終了や強制終了が起きても機密情報を含むファイルが残存する可能性は全くない．本論文では，第一の手法で実現されるセキュア空間を,「揮発性セキュア空間」と呼ぶ．

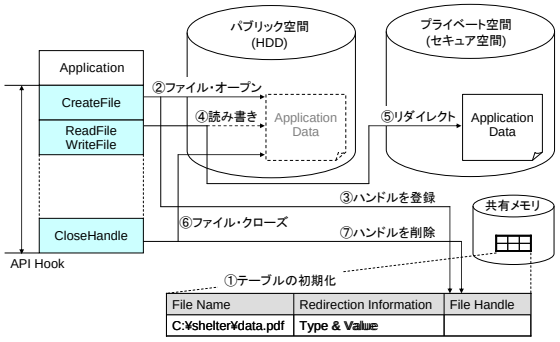

図 2 プライベート・セキュア空間の概念図

第二の手法は，復号されたデータをアプリケーション固有の一時的な鍵で再暗号化した上で HDD に書き出し，アプリケーションによるファイルの読み込みの際にオンザフライで必要な部分だけをプライベートなメモリ領域に復号する．異常終了や強制終了が起きると，再暗号化されたファイルの実体は HDD に残るものの，メモリ上にしか存在しない一時的な鍵が消滅するため，理論上，復号することはできない. この第二の手法で実現されるセキュア空間を,「再暗号化セキュア空間」と呼ぶ．

二つの手法は, いずれも Binary Interception[22][23]をベースにしたリアルタイム監視・制御技術[24][25]により実現する．これは文献 4 でセキュア空間を構築するために採用した技術と同等である．セキュア空間を構築するために，ファイルやクリップボードなどの主要リソースに対するアクセス制御を行うが，それに加えて，アプリケーションによるファイルへのアクセスをプライベートなセキュア空間に配置したデータにリダイレクトする必要がある．

4.1 揮発性セキュア空間の実現

この節では，二つの提案手法のうち，RAM 上にプライ

ベートなデータ領域を確保する揮発性セキュア空間の実現方法を述べる．

提案手法は二次漏えいの防止を目的としており，エンドユーザーによるアプリケーションのインストールや PC の再起動なしに実現することが求められる．そのため，本手法では，稼働中のアプリケーションに API 監視モジュールを動的に注入し，外部からアプリケーションの挙動を制御する技術[4]を用いる．その上で，通常のアクセス制御に加えて，ファイル・アクセスの対象をセキュア空間のデータにリダイレクトする機能を実現する．具体的な手順を図 2 を用いて以下に示す．

(1) データの初期化

パッケージ管理プログラムは，特定のアプリケーションのみがアクセスできる共有メモリ上に図 2 の①に示すようなテーブルを作成し，保護対象のファイル名と，リダイレクトのタイプ（この場合は共有メモリ），その参照値（この場合は名前付き共有メモリの名前）をテーブルに登録する．本方式では，図 2 におけるセキュア空間の実体は共有メモリであり，ファイルの内容をこの名前付き共有メモリ上に展開する．すべての共有メモリには適切な ACL(Access Control List)を設定する．HDD にはダミー・ファイルを生成する．

(2) ファイルを開く際の処理

アプリケーションがデータを読み込む際，最初にファイル・オープン API を呼び出す．例えば Windows の場合は，CreateFile 関数が呼び出される．提案手法では，このような API が呼び出された際，テーブルを探索し，ファイル名が存在した場合，HDD 上のダミー・ファイルを開き（図 2 ②），ファイル・ハンドルをテーブルに保管した上で，呼び出し元にファイル・ハンドルを返す（図 2 ③）．

(3) ファイル操作時の処理

ファイル・ハンドルを必要とする他の API が呼ばれた場合（図 2 ④），与えられたファイル・ハンドルがテーブル内に存在すれば，リダイレクトすべきファイル・ハンドルであると認識できる（図 2 ⑤）．例えば，ReadFile 関数等のファイルの読み込み API が呼び出された場合，ファイル・ハンドルから，共有メモリのアドレスと現在のファイル・ポインタ（共有メモリのオフセット）の位置を取得し，共有メモリ上の現在のオフセットから指定されたバイト数を与えられたバッファにコピーするだけでよい．

また，SetFilePointer 関数等のファイル・ポインタを制御する API では，共有メモリのオフセットを書き直すだけでよい．

(4) 終了処理

ファイル・クローズ API，例えば，CloseHandle 関数が呼び出されてファイルが閉じられる（図 2 ⑥）と，テーブルから対象のファイルに関連した領域を破棄する（図 2 ⑦）ことで処理を終了する．

4.2 再暗号化セキュア空間の実現

再暗号化セキュア空間とは，透過的暗号化フォルダ等の汎用的な手法でセキュア空間全体を透過的に暗号化することではなく，セキュア空間に配置される個々のファイルを特定アプリケーションだけの排他的な一時鍵で再度暗号化することを意味する．前節の揮発性セキュア空間と同様，一連のファイル操作 API を監視することにより，アクセス対象を再暗号化されたファイルからメモリ上に復号されたデータにリダイレクトする．以下，具体的な手順を図 2 を用いて以下に示す．

(1) データの初期化

パッケージ管理プログラムは，特定のアプリケーションのみがアクセスできる共有メモリ上に図 2 の①に示すようなテーブルを作成する．保護対象のファイル名とリダイレクトのタイプ（この場合は再暗号化），その参照値（この場合は一時的に生成された鍵）を登録する．その鍵でファイルを再暗号化した上で HDD に保存する．本方式では，図 2 におけるセキュア空間の実体は一時的に復号される時に使用されるメモリ領域である．

(2) ファイルを開く際の処理

アプリケーションがファイルを開く際（図 2 ②），共有メモリ上のテーブルを参照し，再暗号化ファイルに指定されていれば，ファイル・ハンドルをテーブルに保管する（図 2 ③）．以後，再暗号化ファイルに対するファイル操作では，暗号・復号処理が追加される．

(3) ファイル操作時の処理

アプリケーションがファイルを読み込む場合（図 2 ④），再暗号化されたファイルからデータを読み込み，リダイレクト先のメモリ領域内で一時的に復号し，そのデータを返す（図 2 ⑤）必要がある．揮発性セキュア空間では，ファイルの読み込みとファイル・ポインタの操作は容易に実現できたが，再暗号化されたファイルの場合，実装が複雑になる．

(4) 終了処理

揮発性セキュア空間と同じ処理を行う．

5. 実装と評価

本章では，提案手法を実際に Windows 7 で実装し，実現可能性を検証すると同時に，様々な条件で性能測定を行い，提案した二つの手法の比較とそれらの実用性に関する評価を行う．

5.1 揮発性セキュア空間の実装

先に述べた揮発性セキュア空間を Windows 上に実装するため，主要なファイル関連の Win32 API を監視する．通常のアクセス制御に加え，ファイル・アクセスを RAM 上に確保したプライベートなセキュア空間にリダイレクトするため，CreateFile 関数をはじめとする主要なファイル I/O API が呼び出された際の動作を拡張する．

プライベートなセキュア空間は，Windows の名前付き共有メモリとして実装し，パッケージ管理プログラムが復号したデータを書き出すと同時に，適切な ACL を設定することで他のプログラムからの不正なアクセスを防止する．

共有メモリに書かれたデータは，アプリケーションに注入された API 監視モジュールにより，ファイル・ハンドルから共有メモリのポインタにリダイレクトされることで，アクセス可能となる．

さらに許可されたアプリケーションからセキュア空間外にファイルを書き込むことを禁止する．同時に，クリップボードへのコピーやスクリーンショットの取得や印刷等を禁止する追加のポリシーを適用することにより，情報漏えいを完全に防止できる．情報の機密レベルに応じて，印刷だけは許可するなど，柔軟な運用も可能である．

HDD には復号データの代わりに中身が 0 で埋められたダミー・データを書き込む．これにより，Explorer などのファイル操作アプリケーションに対して，ファイルが物理的に存在するかのように見せかけることができる．ファイル・アイコンのダブルクリックでアプリケーションからファイルを開けるなど，操作性は損なわれない．

5.2 再暗号化セキュア空間の実装

再暗号化セキュア空間を Windows 上に実装する方法は，揮発性セキュア空間の実装方法とほぼ同様である．HDD にダミー・データを書き出す代わりに，一時的な鍵で再暗号化したファイルを保存する．

暗号アルゴリズム[26]は大きなデータの暗号化に最適な対称鍵暗号を用いることにした．ブロック暗号であればいずれの方式も使用可能であるが，一般的に普及したアルゴリズムとして 128-bit キー長の AES(Advanced Encryption Standard)[27]を採用した．ランダム・アクセスを考えるとき，ECB(Electric Code Block)モードより，CBC(Cipher Block Chaining)モードの方が複雑になる．ECB モードのときは，任意のブロックが鍵によって復号できるが，CBC モードの場合，任意のブロックを復号するには，XOR 演算のために，その一つ前のブロックも必要となる．ECB モードの方が実装は容易だが，セキュリティの観点から，CBC モードを採用した．許可されたアプリケーションは，再暗号化されたファイルを開くことができ，さらにデータを読み込む際に，暗号化されたデータがプライベートなセキュア空間に復号される．

5.3 評価

提案した二つの手法はいずれも，通常のファイル処理に加えて，リダイレクトや復号などの追加処理が必要である．そこで，追加処理に伴うオーバーヘッドを定量的に評価するため，特定サイズのファイル（約 30MB の PDF ファイル）に対して本手法を適用した場合の性能評価を行った．表 1 にテスト環境の仕様を示す．

表 1 性能評価の実行環境

OS	Microsoft Windows 7 Professional
CPU	Intel® Core™ 2 Duo CPU T7300 @ 2GHz
RAM	2048MB

最初に，ファイルの読み込み処理に与える影響を評価するため，すべてのデータを起動と同時に読み込むベンチマーク・アプリケーションを作成し，読み込み処理に要する時間を比較した．一般的な商用アプリケーションと同様，ファイル全体を一度に読み込まず，一定サイズの一時バッファを介して繰り返しファイルの内容を読み込む方法にした．表 2 に測定結果を示す．

表 2 ファイルの読み込み時間（秒）

バッファサイズ	通常	共有	暗号
1KBytes	0.148	0.046	2.231
1MBytes	0.031	0.026	2.009

比較のため，1KBytes と 1MBytes の二通りのバッファサイズで測定した．表中，「通常」は本手法を使用していない場合，「共有」は揮発性セキュア空間を使用した場合，「暗号」は再暗号化セキュア空間を使用した場合を示す．

通常方式で読み込んでもファイル・キャッシュの効果により，それほど時間がかからない．バッファサイズを大きくすれば，読み込みの繰り返し処理に伴うオーバーヘッドが少なくなるため，いずれの方式でも性能は改善される．

揮発性セキュア空間の場合，通常方式より性能がよい．ファイル・キャッシュが効いているとはいえ，メモリ上に全てのデータを保持している方が優位である．

一方，再暗号化セキュア空間の場合，通常方式に比べ非常に遅い結果となった．詳細に検証したところ，アプリケーション起動直後は一時的に過負荷となり，復号処理に不利な条件であることがわかった．そこで，アプリケーションを改変し，起動後一定時間経過してからファイルを復号すれば，読み込み時間が約 1 秒に短縮されることがわかった．しかし，通常方式に比べると，性能低下の度合いは相変わらず大きい．

以上の結果より，再暗号化セキュア空間を使うと性能が大幅に低下することから，使い勝手が悪くなると思われがちである．しかし実際の商用アプリケーションでは，起動時にすべてのデータを一度に読み込むことは稀で，必要な部分のみを選択的に読み込み，必要に応じて他の部分を後から読み込むのが一般的である．特に大規模なデータを扱う商用アプリケーションほど，そのような設計を採用する傾向にある．

そこで，実際の商用アプリケーションに各方式を適用し，ファイルを読み始めてから表示するまでの初期化時間を測定した．アプリケーションとしては Adobe Reader Version 11

[20]を使用した．

表 3 に，各方式における Adobe Reader の初期化時間を示す．通常方式と比べ，再暗号化セキュア空間を採用しても，約 7.5%の性能低下にとどまることがわかる．

表 3 Adobe Reader の初期化時間（秒）

通常	共有	暗号
1.185	1.166	1.274

さらに，オーバーヘッドのほとんどが復号処理に費やされていることから，暗号化ロジックを高速化することを試みた．表 4 にその結果を示す．

表 4 暗号高速ロジックを使用した場合（秒）

通常	暗号	高速
1.185	1.274	1.197

「高速」は，演算のビット幅やメモリ使用効率等を最適化し，標準的な暗号方式の 4 倍程度速い暗号ロジックを採用している．高速化暗号ロジックを適用することで，1.274 秒が 1.197 秒まで縮まり，通常方式に比べても，約 1%の性能低下に留まることがわかる．これらの結果から，暗号化方式の性能は，新しく追加された 監視対象 API の数ではなく，暗号・復号ロジックの実装方法が支配的であることがわかる．

以上の結果，Adobe Reader のような一般的な商用アプリケーションで使用する場合，いずれの方式でも性能的にほとんど問題ないことがわかった．

6. まとめ

以前，提案した二次漏えいを防止する手法[4]では，故意による異常終了や強制的な電源切断を使用した攻撃，あるいはマルチユーザー環境における運用上の制約を持っていた．今回，それらの課題に対応できる新しい手法を二つ提示した．

一つは，プライベートなセキュア空間を共有メモリ上に構築し，HDD 上に展開すべきファイルを共有メモリにリダイレクトする方式である．この方式では，HDD を介さずにメモリ上のデータに直接アクセスするため，アプリケーションの性能に与える影響は少ない．ただし，サイズの大きいファイルを開く場合，システム・メモリを大量に消費するため，システム全体の性能に影響を与える可能性がある．

もう一つは，排他的な一時鍵で再暗号化したデータを HDD に展開し，読み込みの際にプライベートなセキュア空間でランタイムに復号する方式である．この方式では，システム・メモリを大量に消費することはないため，システム全体の性能を著しく下げることはないが，データを読み込むたびに毎回復号処理を行うため，アプリケーションの設計次第では起動に膨大な時間を要する場合がある．ただし，ファイル I/O が分散化された商用アプリケーションでは影響は少なく，また，高速な暗号プログラムを用いることにより，さらに性能が改善することがわかったので，実用上問題は少ない．

提案した二つの手法を状況に応じて使い分けることで，より安全に実用的な二次漏えい防止策を提供できる．

今後の課題として，Adobe Reader 以外の様々な商用アプリケーションへの適用が考えられる．単純なファイル・ビューアである Adobe Reader の場合，数種類のファイル操作 API を監視するだけで提案手法を実現できたが，より大規模なアプリケーションでは，ファイル・マッピングなどの高度な手法を使ってデータ処理をする場合もあり，それらの対応も行う必要がある．

参考文献

1) 独立行政法人情報処理推進機構セキュリティセンター： 2013 年版 10 大脅威 ～身近に忍び寄る脅威～ (2013/03)
2) 独立行政法人情報処理推進機構セキュリティセンター：組織内部の不正行為によるインシデント調査 (2012/07)
3) 独立行政法人情報処理推進機構セキュリティセンター： 2011 年情報セキュリティインシデントに関する調査報告書 (2012/7)
4) 多田政美，古市実裕，新村泰英：一時的セキュア空間作成による情報の二次流出防止, ProVISION, Winter 2011, No.68, pp.88-95 (2011)
5) 新井利明，溝口幸信：モバイルセキュリティを強化したシンクライアントソリューション，情報処理，Vol.47, No.10, pp.1127-1136 (2000)
6) Nieh, J., Yang, S. J. and Novik, N.: A comparison of thin-client computing architectures, Technical Report CUCS-022-00, Columbia University (2000/11)
7) VMware: VMWare Horizon, http://www.vmware.com/products/horizon-view/
8) Citrix: XenDesktop, http://www.citrix.com/products/xendesktop/overview.html
9) Microsoft: VDI TCO Analysis for Office Worker Environments (2010)
10) Adobe: Adobe LiveCycle Enterprise Suite 4 (ES4), http://www.adobe.com/products/livecycle/
11) Microsoft: Technical overview of Windows Rights Management Services for Windows Server 2003 (2003/11)
12) Microsoft: Office 365, http://office.microsoft.com/en-us/products/microsoft-office-365-for-home-or-for-business-office-online-FX101825692.aspx
13) Google: Google Cloud, https://cloud.google.com/
14) Loscocco, P. and Smallley, S.: Integrating flexible support for security policies into the Linux operating system, Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference (2001)
15) Bauer, M.: Paranoid penguin: an introduction to Novell AppArmor, Linux Journal, Vol.2006, No.148, p.13 (2006/08)
16) 中村雄一，鮫島吉喜 ：Security-Enhanced Linux のアクセス制御ポリシ設定の簡略化, SCIS2003, pp.831-836 (2003)
17) 田端利宏，箱守聡，大橋慶，植村晋一郎，横山和俊，谷口秀夫：機密情報の拡散追跡機能による情報漏えいの防止機構，情報処理学会論文誌, Vol.50, No.9, pp.2088-2102 (2009)
18) 日立ソフト：秘文, http://www.hitachi-solutions.co.jp/hibun/
19) Verdasys: Digital Guardian, https://www.verdasys.com/products/index.html

20) Adobe Systems: Adobe Reader, http://get.adobe.com/jp/reader/
21) Microsoft: Encrypting File System, http://technet.microsoft.com/en-us/library/cc782901(WS.10).aspx
22) Hunt, G. and Brubacher, D.: Detours: Binary Interception of Win32 Functions, Proceedings of the 3rd USENIX Windows NT Symposium, pp.135-144 (1999)
23) Hunt, G. C. and Scott, M. L.: Intercepting and Instrumenting COM Applications, Proceedings of the 5th Conference on Object-Oriented Technologies and Systems (COOTS '99), pp.45-56 (1999)
24) 古市実裕，三品拓也，大谷佑： PC におけるプログラムのリアルタイム監視・制御技術の開発と情報セキュリティシステムへの応用, SCIS2006 (2006)
25) 古市実裕，多田政美，池部敦巳： PC セキュリティ強化のための COM の透過的ポリシー強制手法, SCIS2007 (2007)
26) Schneier, B.: Applied Cryptography Second Edition, Wiley, ISBN 0-471-11709-9 (1996)
27) Daemen, J. and Rijmen, V.: AES Proposal: Rijndae, AES Algorithm Submission (1999/09/03)