情報処理学会論文誌　Vol. 51　No. 10　1234–1241 (Oct. 2010)

管理 VM へのキーボード入力情報漏洩の防止

江川友寿[†1]　光来健一[†1,†2]

IaaS 型クラウドサービスにおいて，ユーザは提供された VM（ユーザ VM）に遠隔ログインすることで VM の操作を行うが，ユーザ VM に起因するネットワーク障害時には VM の操作が不可能になってしまう．そこでユーザ VM を管理している特別な VM（管理 VM）経由でユーザ VM にアクセスすることが望ましいが，IaaS において管理 VM は信頼できないため，キーボード入力情報が漏洩するリスクを伴う．この問題を解決するために，本稿では管理 VM 経由でのアクセスでもユーザ VM に安全にキーボード入力情報を送れるシステム *FBCrypt* を提案する．FBCrypt は，キーボード入力をユーザの VNC クライアントで暗号化し，クラウド側の仮想マシンモニタで復号化を行う．FBCrypt を Xen と TigerVNC に実装し，キーボード入力情報が漏洩していないことを確認した．

Preventing Information Leakage to Privilege VM regarding Keyboard Input

Tomohisa Egawa[†1] and Kenichi Kourai[†1,†2]

In IaaS cloud, the users manage their virutual machines (user VMs) via remote login, but network failure caused by the VMs makes such management difficult. Therefore it is desirable to manage user VMs via privileged VMs called management VMs. However, keyboard inputs may leak because management VMs are untrustworthy in IaaS. To solve this problem, this paper proposes *FBCrypt*, which securely sends keyboard inputs to user VMs via management VMs. FBCrypt encrypts keyboard inputs in VNC clients and decrypts them in the virtual machine monitor of IaaS. We have implemented FBCrypt in Xen and TigerVNC and confirmed that keyboard inputs do not leak.

†1 九州工業大学
†2 独立行政法人科学技術振興機構，CREST

1. はじめに

ネットワークを介してサービスを提供するクラウドコンピューティングの利用が広がっている．その一形態として，ユーザに仮想マシン（VM）を提供する Infrastructure as a Service（IaaS）と呼ばれるクラウドサービスがある．IaaS のユーザはサービス提供のためのハードウェアを用意することなく，クラウド上の VM を必要な時に必要なだけ利用することができる．ユーザは VNC や SSH などを用いて，VM に直接アクセスすることで VM 内部のソフトウェアの管理を行う．しかし，ファイアウォールの設定ミスなどによるネットワーク障害時には VM へのアクセスが不可能になってしまい，以降の VM の管理が困難になってしまう．

ユーザ VM の障害時でも管理を行えるように，管理に用いられている専用 VM（管理 VM）を経由してユーザの VM（ユーザ VM）にアクセスすることが望ましいが，その一方で情報漏洩のリスクが高まるという問題がある．これは IaaS においては管理 VM を十分に信頼できるとは限らないためである．例えば，管理 VM のセキュリティ対策が十分でない場合，攻撃者に侵入される可能性がある．また，悪意を持ったシステム管理者が攻撃を行う可能性も考えられる．その結果，管理 VM の VNC サーバが改ざんされたりすると，ユーザ VM へのキーボード入力情報が盗聴され，パスワードなどの機密情報が漏洩してしまう．

この問題を解決するために，管理 VM 経由でのアクセスでもユーザ VM に安全にキーボード入力情報を送れるシステム *FBCrypt* を提案する．FBCrypt はユーザが VNC クライアントに対して行ったキーボード入力を暗号化して管理 VM 上の VNC サーバに送信する．VNC サーバがキーボード入力をユーザ VM に渡す際に，仮想マシンモニタ（VMM）が間に入って復号化を行う．VMM は復号化したキーボード入力を従来と同じインターフェースでユーザ VM に送信するため，ユーザ VM のゲスト OS への修正は必要ない．FBCrypt を用いることにより，管理 VM 内ではキーボード入力情報が暗号化されているため，VNC サーバの改ざんによる盗聴が行われたとしても情報が漏洩することはない．FBCrypt はリモートアテステーションと呼ばれるハードウェアを用いた認証技術を用いることで，ハードウェアおよび VMM を信頼する．

我々は，準仮想化 Linux を対象として FBCrypt を Xen 4.0.2[1] および TigerVNC[2] に実装した．キーボード入力はストリーム暗号の RC4 により暗号化され，ドメイン 0 上の QEMU 内の VNC サーバは，ハイパーコールを用いて暗号化されたキーボード入力を VMM に渡す．VMM は受け取ったキーボード入力を復号化し，ユーザ VM のカーネル内の I/O

リングと呼ばれるメモリ領域に書き込む．ドメイン 0 から I/O リングへのアクセスを禁止することで，VMM が復号化したキーボード入力情報を盗まれないようにする．FBCrypt を用いた実験を行い，管理 VM 経由の VNC 接続時において，ユーザ VM へのキーボード入力情報が漏洩していないことを確認した．

以下，2 章で IaaS 環境における情報漏洩の問題について述べる．3 章でこの問題を解決する FBCrypt について述べ，4 章でその実装の詳細について述べる．5 章で FBCrypt を用いて行った実験について述べる．6 章で関連研究に触れ，7 章で本稿をまとめる．

2. 管理 VM への情報漏洩

IaaS のユーザは，VNC クライアントを使用する場合にはユーザ VM の VNC サーバに接続することでユーザ VM 内部のソフトウェアの管理を行う．この管理方法の問題点は，VM を動作させているマシンが組織外部のクラウドに置かれているため，VNC サーバへの接続ができなくなると VM を管理することができなくなることである．ユーザ VM 内のでネットワークやファイアウォールの設定を間違えると，VNC サーバにネットワークアクセスすることができなくなる．また，ユーザ VM 内の OS を起動している間や OS がクラッシュした時には VNC サーバ自体が動作していないため，接続することができない．

このような状況でもユーザ VM の管理を継続できるようにするためには，**図 1** のように管理 VM で VNC サーバを動作させることが望ましい．管理 VM とはユーザ VM の起動や停止，マイグレーションなどの制御を行える特権を持った VM のことである．管理 VM 経由でユーザ VM の仮想ハードウェアを通してユーザ VM にアクセスすると，ユーザ VM の障害時でもローカルコンソールからログインしているかのように操作することができ，より柔軟な VM の管理が可能になる．例えば，ユーザ VM へのネットワーク接続ができなくなってもキーボード入力などを行うことができ，OS の起動時にも起動メッセージを見ることができる．既存の IaaS プロバイダはこのような管理方法を提供していないが，Xen VNC Proxy（xvp）プロジェクト[3] では，管理 VM 経由でユーザ VM を管理できるようになっている．

しかし，管理 VM 経由でユーザ VM にアクセスすると情報漏洩のリスクが高まる．クラウド環境においては管理 VM が十分に信頼できるとは限らないためである．IaaS 上の VM はデータセンタ間をマイグレーションで移動することがあり，どのデータセンタで自分の VM が動いているか分からないこともある．その結果，セキュリティ意識の低いシステム管理者のいるデータセンタで VM が動作する可能性も考えられる．このような環境では，システム管理者の怠慢により管理 VM に脆弱性が残っていたりすると，第三者の攻撃により管理 VM の制御が奪われる恐れがある．またシステム管理者自身に悪意があった場合，管理 VM の中で不正を働くことは容易である．

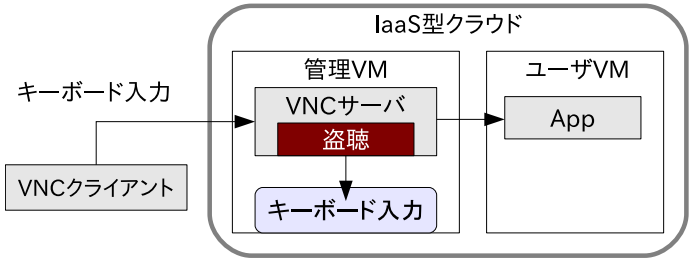

図 1　キーボード入力の盗聴

攻撃者やシステム管理者によって管理 VM の VNC サーバが改ざんされた場合，クライアントからのキーボード入力情報が容易に盗聴されてしまう．VNC クライアントで入力されたキーボード入力情報は，**図 1** のように VNC サーバに盗聴プログラムを組み込むだけでパスワードなどの機密情報が漏洩する．クライアントとサーバ間のネットワーク上では，VPN や SSH トンネリングなどで暗号化が可能であるが，VNC サーバが受信した時点で入力された情報は復号化されている．ユーザ VM で VNC サーバを動かす場合はユーザ VM 内で復号化されるため，このような管理 VM への情報漏洩のリスクはなかった．

3. FBCrypt

管理 VM が信頼できない IaaS 環境下でも，ユーザ VM に安全にキーボード入力を送れるシステム FBCrypt を提案する．

3.1 脅威モデル

FBCrypt は管理 VM 経由でユーザ VM にアクセスした際に，管理 VM のプログラム内でキーボード入力情報が盗まれる攻撃を想定している．よって管理 VM および管理 VM のシステム管理者は信頼しない．ただし，FBCrypt は攻撃者や管理 VM のシステム管理者がユーザ VM に直接侵入して情報を盗む攻撃は想定しない．ユーザ VM はユーザに正しく管理されているとし，パスワードやソフトウェアに脆弱性はないものとする．キーボード入力を行うクライアント環境も十分に信頼できるものとし，クライアント PC もしくは VNC ク

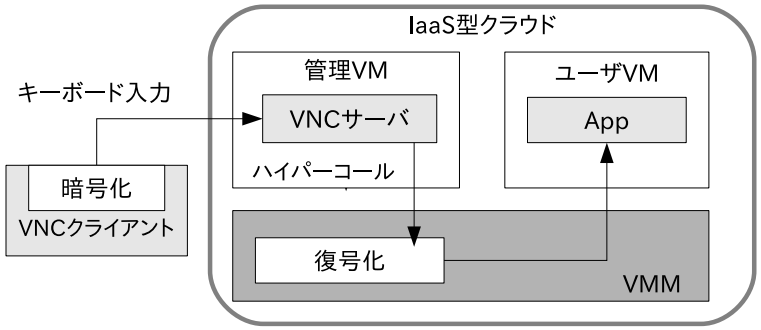

図 2　FBCrypt の構成

ライアントからの情報漏洩はないものとする．

FBCrypt は VMM が不正に改ざんされていないものとして，クラウド内の VMM を信頼する．VMM が改ざんされていないことはリモートアテステーションによって検証することができる．リモートアテステーションはマシンの起動時に VMM のハッシュ値を計算し，それを第三者が検証することでコード改ざんの有無を安全に確認する技術である．これはシステム管理者であっても改ざんが不可能な TPM などのハードウェアの機能を使うことで実現される．ただし，VMM をインストールし，リモートアテステーションの設定を行う管理者は信頼できるとし，システムの導入時点で不正な改ざんはないものとする．

3.2　FBCrypt

FBCrypt はユーザの VNC クライアントとユーザ VM が動いている VMM の間でキーボード入力情報を暗号化する．FBCrypt のシステム構成を**図 2** に示す．ユーザの VNC クライアントに対して行われたキーボード入力は VNC クライアント内で暗号化され，管理 VM の VNC サーバに送信される．VNC サーバが VMM に暗号化されたキーボード入力を送ると，VMM は受け取った情報を復号化し，ユーザ VM の仮想キーボードデバイスへの入力とする．これにより，管理 VM 内ではユーザ VM へのキーボード入力が暗号化されるため，盗聴されても情報が漏洩することはない．また，ユーザ VM のゲスト OS は従来と同じインターフェースで仮想キーボードデバイスにアクセスすることができる．VMM 内ではなくユーザ VM 内の仮想キーボードのデバイスドライバ等で復号化する手法も考えられるが，ゲスト OS への変更が必要になる．このことは様々な OS を利用できるようにする上での障害となる．

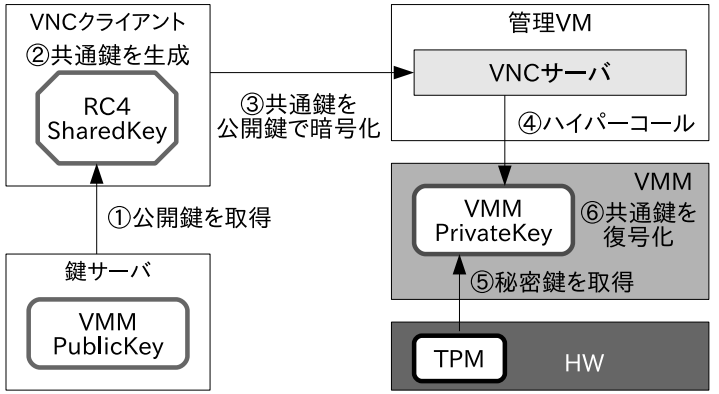

図 3　リモートアテステーションを利用した鍵の共有

管理 VM がユーザ VM のメモリから復号化されたキーボード入力情報を取得するのを防ぐために，VMCrypt[4] を用いてユーザ VM のメモリを暗号化する．管理 VM はユーザ VM を起動したりマイグレーションしたりするために，ユーザ VM のメモリにアクセスすることができる．VMCrypt は管理 VM がユーザ VM のメモリにアクセスする時にはいくつかの例外を除いて暗号化したページを見せることで，管理 VM からユーザ VM のメモリの内容を盗み出せないようにするシステムである．VMCrypt を用いることで，VMM が復号化してユーザ VM に渡したキーボード入力情報はユーザ VM しか見られないようにすることができる．

FBCrypt では，キーボード入力の暗号化と復号化に用いる共通鍵を VNC クライアントと VMM の間で安全に共有する．リモートアテステーションを利用した鍵の共有の手順を**図 3** に示す．ユーザが VM にアクセスする際には，鍵サーバから接続先の VMM の公開鍵を取得する．この際に，接続先の VMM が改ざんされていないことをリモートアテステーションにより確認する．鍵サーバには信頼できる IaaS の管理者によりあらかじめ正当な VMM の公開鍵を登録しておく．VNC クライアントは取得した VMM の公開鍵を使って共通鍵を暗号化して管理 VM に送信する．管理 VM は暗号化された共通鍵をそのまま VMM に送り，VMM は自身の秘密鍵で共通鍵を復号化する．これにより，各ユーザの VNC クライアントと VMM 間で VNC 接続を開始するたびに新しい共通鍵の共有が可能となる．VMM の秘密鍵は TPM 内に保存され，正しい VMM が起動されてアテステーションに成功した時だけ TPM から取り出すことができる．そのため，管理 VM から VMM のバイナリを見

られても，秘密鍵が漏洩することはない．

4. 実　　装

我々は FBCrypt を Xen 4.0.2 および TigerVNC に実装した．Xen においては管理 VM はドメイン 0，ユーザ VM はドメイン U となる．管理 VM 内で動作する QEMU に VNC サーバが含まれている．ユーザ VM 内で動作させるゲスト OS として準仮想化 Linux を対象とした．

4.1 キーボード入力の暗号化と復号化

FBCrypt の暗号化と復号化の処理の流れを**図 4** に示す．VNC クライアントがキーボード入力を暗号化して QEMU 内で動作している VNC サーバに送信する．VNC サーバはハイパーコールを用いて VMM に暗号化されたキーボード入力情報を渡し，VMM はそれを復号化してドメイン U の I/O リングに書き込む．I/O リングはドメイン 0 とドメイン U の間で入出力データをやりとりするためのリングバッファである．従来は，VNC サーバがキーボード入力を受け取ると直接，ドメイン U の I/O リングに書き込みを行っていた．FBCrypt では VMM が従来と同様に I/O リングにキーボード入力情報を書き込むことにより，ドメイン U は従来通りに I/O リングからキーボード入力情報を読み出すことができる．

VMM は VNC サーバからキーボードもしくはマウスの入力情報を受け取る．この入力情報には，キーボード入力，マウスの相対位置と絶対位置の 3 種類が含まれる．マウスのクリックに関する情報はキーボード入力情報に含まれる．VMM は渡された入力情報がキーボードに関するものである場合だけ復号化を行う．復号化されたキーボード入力は ASCII コードのため，キーコードに変換する．キーコードとはキーボードの配列順に割り当てられているハードウェア依存の値であり，I/O リングにはキーコードでキーボード入力を渡す必要がある．従来は VNC サーバが ASCII コードをキーコードに変換していたが，FBCrypt では ASCII コードが暗号化されているため VMM でしか変換を行うことができない．VMM は変換によって得られたキーコードを I/O リングに書き込む．VNC サーバから渡された入力情報がマウスに関するものであった場合はそのまま I/O リングに書き込む．

キーボード入力情報の暗号化にはストリーム暗号の RC4 を用いた．OpenSSL[5] ライブラリの RC4 をオープンソースの VNC クライアントである TigerVNC[2] に組み込み，キーボード入力を一文字ごとに暗号化して VNC サーバに送る．ストリーム暗号はバイト単位での暗号化が可能であり，平文のデータサイズと暗号化済みのデータサイズが常に一致する．RC4 は，疑似乱数生成器の内部状態を初期化する鍵スケジュールアルゴリズム（KSA）と，

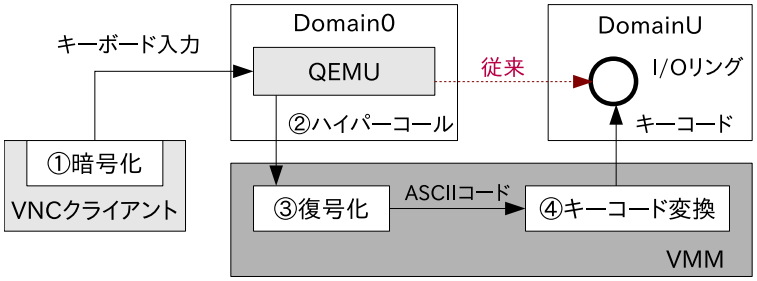

図 4　FBCrypt の暗号化と復号化

内部状態を更新しながらキーストリームを生成する疑似乱数生成アルゴリズム（PRGA）からなる．KSA はあらかじめ登録された鍵を使って内部状態を初期化し，PRGA は内部状態の配列の要素を入れ替えることでキーストリームを生成する．このキーストリームと平文もしくは暗号文との排他的論理和をとることでそれぞれ暗号化と復号化を行う．内部状態の初期化に使うための鍵は，VNC クライアントと VMM で共通の鍵を使用する．VNC クライアントを終了した後で接続を行うと VNC クライアントの内部状態が初期化され，VMM が保持している内部状態との不一致が起き復号化が行えなくなる．このため，VNC サーバは VNC 接続のたびに VMM の内部状態の初期化を行うハイパーコールを発行する．改ざんされた VNC サーバがこのハイパーコールを発行しなかったとしても，VNC クライアントが正しく通信できないだけで情報が漏洩することはない．

4.2 I/O リングのアドレス取得

FBCrypt の VMM がドメイン U の I/O リングに復号化したキーボード入力を書き込むためには，VMM が I/O リングのアドレスを取得する必要がある．I/O リングのアドレスを入手するための方法はいくつか考えられる．一つは，ドメイン 0 の VNC サーバからハイパーコールを使って I/O リングのアドレスを VMM に通知してもらう方法である．しかしこの方法では，ドメイン 0 が偽のアドレスを通知することで，VMM で復号化された後の情報が盗み見られてしまう．別の方法としては，ドメイン U から I/O リングのアドレスを通知してもらう方法がある．I/O リングはドメイン U が確保するので，ハイパーコールを使って VMM に I/O リングのアドレスを通知することができる．しかし，ドメイン U のゲスト OS がハイパーコールを呼び出すように修正を行う必要がある．

そこで FBCrypt では VMM が独力で I/O リングのアドレスを取得する．ドメイン 0 には，XenStore と呼ばれる各 VM の仮想マシン情報を共有・管理するための簡易データベー

スがあり，ドメイン U は起動時に XenStore と構成情報のやり取りを行う．I/O リングのアドレスは，XenStore とドメイン U 間でやり取りされる情報の中のひとつであり，この通信を VMM が監視することでドメイン U やドメイン 0 にハイパーコールを追加することなく，I/O リングのアドレスを取得することができる．また，この手法はドメイン 0 に依存しないため I/O リングのアドレスの詐称を防ぐことができる．

ドメイン U が XenStore にアクセスするためのインターフェースは XenBus と呼ばれる．XenBus は準仮想化環境におけるドメイン間通信のためバスの抽象化を提供しており，送受信用の 2 つの XenStore リングを含んだ共有メモリとイベントチャネルから構成される．ドメイン U が XenStore に構成情報を送信する際には，16 バイトのヘッダ情報に続けて XenStore に格納される形式のパス情報とデータを XenStore リングに書き込む．仮想キーボード用の I/O リングの情報を登録する時には，*device/vkbd/0/page-ref* というパス情報と I/O リングが置かれているメモリページのマシンフレーム番号が書き込まれる．

ドメイン U はハイパーコールを用いてドメイン 0 にイベントを送ることで XenBus を使った送信を行う．イベント送信のタイミングで XenStore リングの監視を行うことで，VMM は I/O リングのマシンフレーム番号を取得することができる．イベントが送信されるたびに VMM は XenStore リングから 1 つの要求を盗み見て，パス名が対象の I/O リングのものかどうかを調べる．パス名が一致すればデータ部から I/O リングのマシンフレーム番号を取得でき，それ以降の監視は行わない．XenStore リングを監視するために，VMM をはどの要求まで調べたかを保持するポインタを管理する．1 つの要求を調べるたびにこのポインタが更新される．XenStore リング自体もどこまで要求がドメイン 0 に処理されたかを保持するポインタを管理しているが，XenStore が要求を処理するまでは更新されないため，VMM がどこまで調べたかを表すポインタを別に管理する必要がある．

4.3　XenStore リングのアドレス取得

VMM は XenStore リングを監視するためにドメイン内の XenStore リングのアドレスを取得する．XenStore リングのマシンフレーム番号はドメイン 0 とドメイン U が共有している起動情報ページに記載されているが，VMM は起動情報ページを管理していないため容易に取得することはできない．このため，ドメイン U 起動時に仮想 CPU のレジスタにセットされるアドレスから起動情報ページを特定し，XenStore リングのアドレスを取得する．

このレジスタにセットされているアドレスはドメイン U の仮想アドレスであるため，VMM はこのアドレスをマシンフレーム番号に変換する．通常，仮想アドレスは疑似物理フレーム番号に変換してから，ドメイン U の持つ P2M テーブルを用いてマシンフレーム番号に変換する．しかしドメイン U 起動時には P2M テーブルが作成されていないため，P2M テーブルからマシンフレーム番号を直接求めることはできない．そこで VMM が管理している M2P テーブルを用いて，全てのページについてマシンフレーム番号から疑似物理フレーム番号に変換していき，疑似物理フレーム番号が一致した時のマシンフレーム番号が起動情報ページのマシンフレーム番号となる．起動情報ページのマシンフレーム番号を入手したら，そのメモリページを VMM のメモリ空間にマップすることで XenStore リングのマシンフレーム番号を取得することができる．さらに取得した XenStore リングの置かれているメモリページをマップすることで，VMM による XenStore リングの監視が可能となる．

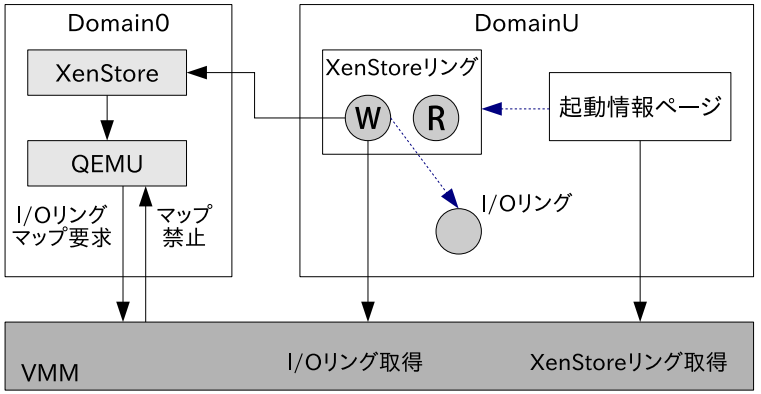

図 5　ドメイン U 起動時の FBCrypt の処理

4.4　I/O リングへのアクセス禁止

VMM が復号化した後にキーボード入力情報をドメイン 0 に盗み見されないようにするには，ドメイン 0 から I/O リングにアクセスできないようにする必要がある．VMM は暗号化されたキーボード入力情報を復号化して，ドメイン U の I/O リングに格納する．ドメイン 0 は XenStore から I/O リングのマシンフレーム番号を取得することができるため，そのメモリページを自身のメモリ空間にマップすることで，復号化後のキーボード入力を盗み見ることができてしまう．

そこで FBCrypt の VMM はドメイン 0 に対して，I/O リングの置かれているドメイン U のメモリページをマップすることを禁止する．ドメイン 0 がドメイン U のメモリページをマップするためにはページテーブルへのエントリの追加が必要となる．物理メモリを管理しているのは VMM なので，ドメインのページテーブル変更の際には，ハイパーコールを

```
[root@Domain0 ~]# tail VNC_KEYLOG2
r o o t  p a s s w a r d
```

図 6　管理 VM からのキーボード入力の盗聴

```
[root@Domain0 ~]# tail VNC_KEYLOGG
_ ? ? ? % w ? $ ? V f  [00 1F]: %
 ? t [00 03] " ? ? | " ? ? ? " [00 16]? ? ?
```

図 7　暗号化されたキーボード入力情報

実行する必要がある．このハイパーコールがドメイン 0 によって発行されており，かつ事前に XenStore リングから取得した I/O リングのマシンフレーム番号と，要求されたマシンフレーム番号が一致した場合，ページテーブルへのエントリの追加を行わないことでドメイン 0 が I/O リングにアクセスすることを禁止することができる．VMCrypt を用いる場合は，ドメイン 0 が I/O リングの置かれているメモリページをマップすると暗号化されるため，アクセスを禁止する必要はない．

5. 実　　験

FBCrypt により情報漏洩防止ができていることを確認し，FBCrypt のレスポンスタイムとオーバヘッドを測定するための実験を行った．実験には Intel Core 2 Quad Q9550 2.83GHz の CPU を搭載したマシンを VNC クライアント用と VM 用にそれぞれ用意し，ギガビットイーサネットを用いて接続した．VM 用のマシンでは VMM として FBCrypt を実装した Xen 4.0.2 を動作させ，ドメイン 0 とドメイン U では Linux 2.6.32.21 を動かした．このマシンは 3.5GB のメモリを搭載しており，ドメイン 0 には 3GB，ドメイン U には 512MB をそれぞれ割り当てた．VNC クライアントに使用したマシンは 8GB のメモリを搭載しており，Linux2.6.32.21 を動作させ，VNC クライアントには FBCrypt を実装した TigerVNC を使用した．

5.1　キーボード入力情報の漏洩防止の確認

FBCrypt を用いることで管理 VM からのキーボード入力情報の漏洩を防げることを確かめるために，管理 VM 上の VNC サーバを改ざんしキーロガーを組み込んだ．このキーロガーは VNC クライアントから送られてきたキーボード入力情報をファイルに保存するというものである．管理 VM を経由してユーザ VM に VNC 接続する従来システムでは，VNC クライアントから送られてくるキーボード入力は**図 6** のように盗聴され，ログインパスワードがそのまま記録されてしまっている．それに対して，FBCrypt ではキーボード入力が**図 7** のように暗号化されて記録されており，盗聴を行うことができなかった．このことから，FBCrypt により管理 VM にキーボード入力情報を漏洩させることなく安全にユーザ VM を操作できることが確認できた．

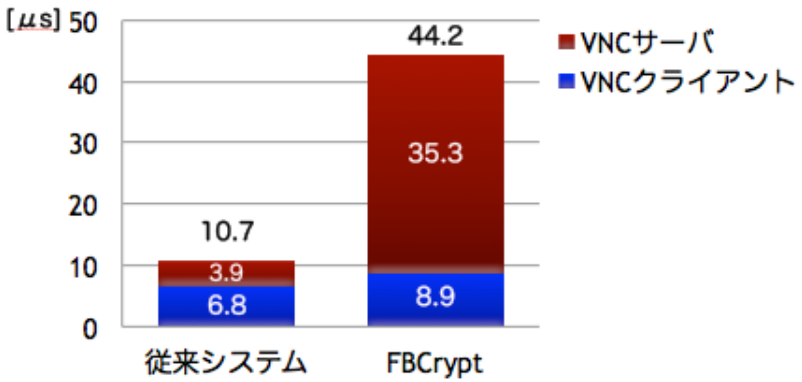

図 8　キーボード入力の処理時間

5.2　FBCrypt のオーバヘッド

次に FBCrypt による VNC クライアントおよび VNC サーバにおけるオーバヘッドを測定するために，従来システムとキーボード入力処理時間の比較を行った．VNC クライアントではキーボード入力を受け取ってから VNC サーバに送信するまでの時間を測定した．FBCrypt ではストリーム暗号を使ってキーボード入力が暗号化される．一方，VNC サーバでは送られてきたキーボード入力を受け取ってからユーザ VM の I/O リングに書き込むまでの時間を測定した．従来システムでは VNC サーバが直接 I/O リングに書き込むのに対し，FBCrypt では VMM が復号化および I/O リングへの書き込みを行う．測定結果は**図 8** のようになった．測定値はキーボード入力を 100 回行った際の平均値である．

VNC クライアントについては，処理時間の差が 2μs であることからストリーム暗号の RC4 による暗号化処理は十分に高速であることがわかる．一方，VNC サーバにおいて復号化にかかる処理は従来システムとの処理時間の差が 31μs であり，FBCrypt では 10 倍程度遅くなっていることがわかった．その内訳を調べたところ，VMM を呼び出すハイパーコー

表 1　キーボード入力一回あたりのレスポンスタイム（ms）

	SSH トンネリングなし	SSH トンネリングあり
従来	1.19	1.35
FBCrypt	1.28	1.48
ユーザ VM 直接	9.74	9.92

ルにかかる処理時間が 28μs であり，復号化に要する時間は 3μs であった．このことより，ハイパーコールの処理自体がボトルネックになっていることがわかる．

5.3　レスポンスタイム

キーボード入力一回あたりのレスポンスタイムを FBCrypt と従来システムで比較した．VNC クライアントでキーボード入力を行い，文字を表示して画面が書き換わることで送られてくるフレームバッファ変更要求をクライアントが受け取るまでの時間を測定した．キーボード入力を 100 回行った際の平均値を**表 1** に示す．SSH トンネリングを行わずに従来システムと FBCrypt を比較すると，FBCrypt は 0.09ms 遅くなった．しかし，SSH トンネリングで通信路を暗号化した従来システムと FBCrypt を比較すると，FBCrypt の方が 0.07ms 速かった．FBCrypt ではキーボード入力が暗号化されるため SSH トンネリングを行う必要はない．SSH と FBCrypt で用いている暗号方式が異なるため単純な比較はできないが，FBCrypt によるレスポンスタイムの悪化は小さいといえる．

一方，ユーザ VM で VNC サーバを動かしてユーザ VM に直接アクセスした時のレスポンスタイムは SSH トンネリングを行うと 9.9ms となり，管理 VM 経由の接続と比べて 9 倍程度遅かった．これはユーザ VM の VNC サーバにアクセスすると仮想ネットワークの遅延が発生したり，VNC サーバとアプリケーション間でのコンテキストスイッチが発生したりするためだと考えられる．

6. 関 連 研 究

GuardedID[6] や KeyScrambler[7] などは，OS にキーロガーが埋め込まれていたとしても，キーボード入力情報を安全にアプリケーションに渡す．例えば，ブラウザでオンライン決済を行う場合などに個人情報やパスワード等の機密情報を盗聴されないようにすることができる．これらはキーボード入力時にデバイスドライバレベルで暗号化を行い，ウェブブラウザ自身が復号化を行う．これにより OS 内のキーボード入力処理の途中で盗聴を行う典型的なキーロガーはキーボード入力情報を取得することができない．この点では，管理 VM 内のキーロガーに盗聴されないようにする FBCrypt と似ている．しかし，暗号化を行う処理が同じ OS 内で行われており，デバイスドライバレベルで盗聴を行うキーロガーに対しては無力である．アプリケーション毎の対応が必要という問題もある．

ソフトウェアキーボードは，ハードウェアのキーボードで行う入力をソフトウェアで実現したものである．画面上にキーボードを表示してマウスで画面上のキーをクリックすることで任意の文字を入力できる．VNC クライアントでソフトウェアキーボードを用いることで管理 VM にキーボード入力を盗まれることなく安全に入力を行うことができる．しかし，管理 VM からはマウス入力情報や画面情報を取得することもできるため，これらの情報から入力されたキーを特定される可能性がある．また，ソフトウェアキーボードによる入力は非常に手間がかかる．

VMCrypt[4] は，ユーザ VM のメモリから管理 VM のシステム管理者への情報漏洩を防ぐシステムである．管理 VM がユーザ VM のメモリにアクセスをしようとした時に，必要に応じて VMM がユーザ VM のメモリを暗号化することで，メモリの内容を盗み出せないようにする．VMCrypt は本研究と同じく管理 VM を信頼しないという前提で設計されており，本研究と VMCrypt を併用することにより，管理 VM に対するユーザ VM のセキュリティをさらに向上させることができる．

BitVisor[8] ではゲスト OS のストレージやネットワークの暗号化を VMM が行う．ゲスト OS からの I/O は VMM が監視しており，ウイルス感染や USB メモリの盗難，紛失といった事態が発生してもクライアント PC からの情報漏洩を防ぐことができる．BitVisor には管理 VM にあたるものがないため，暗号化・復号化処理は VMM で行われる．そのため，VMM が信頼できれば情報漏洩の危険性はない．しかし，ゲスト OS の障害時でも管理を行えるようにするには VNC サーバを VMM に組み込む必要があり，VMM が肥大化してしまう．

Xen VNC Proxy[3]（xvp）は，管理 VM 経由でユーザ VM を操作するオープンソースツール群である．xvp はウェブブラウザでサーバホストやその上で動作する VM を管理でき，新規に VM を作成したり，VM を起動・停止したりすることも可能である．xvp の通信は全て VNC プロトコルを拡張したものが使用されている．xvp の VNC ビューアは Java で書かれており，クロスプラットフォームになっている．FBCrypt 用に改造した VNC クライアントは現在のところ Linux でしか動作しないが，xvp の VNC ビューアに暗号化機能を実装することで，他のプラットフォームでも動かすことができる．

7. ま と め

本稿では，管理 VM 経由でのアクセスでもユーザ VM に安全にキーボード入力情報を送れるシステム FBCrypt を提案した．FBCrypt は，ユーザが VNC クライアントに対して行ったキーボード入力を暗号化して管理 VM 上の VNC サーバに送信し，VNC サーバがキーボード入力をユーザ VM に渡す際に VMM が復号化を行う．これにより，管理 VM 内ではキーボード入力は暗号化されているため，盗聴されても情報が漏洩することはない．我々は FBCrypt を Xen および TigerVNC に実装し，キーボード入力情報が漏洩しないことを確認した．

今後の課題は，VNC クライアントと VMM の間での鍵交換を実装することである．そのためには VMM 内で公開鍵暗号の処理を行えるようにする必要がある．次に完全仮想化ゲスト OS への対応を行う．完全仮想化に対応することにより，Windows など準仮想化 Linux 以外の OS でも FBCrypt が使用できるようになる．完全仮想化のキーボード入力処理は準仮想化 Linux とは大幅に異なるため，一から実装を行う必要がある．またマウスの情報や画面情報の暗号化も行っていく予定である．

参 考 文 献

1) P.Barham, B.Dragovic, K.Fraser, S.Hand, T.Harris, A.Ho, R.Neugebauer, I.Pratt, and A.Warfield. Xen and the Art of Virtualization. *In Proc. of the 19th Symposium on Operating Systems Principles*, pp. 164–177, 2003.
2) RedHat, Inc. TigerVNC. http://tigervnc.sourceforge.net.
3) xvp Project. Xen VNC Proxy：Cross-platform VNC-based and Web-based Management for Citrix XenServer and Xen Cloud Platform. http://www.xvpsource.org/.
4) 田所秀和, 光来健一, 千葉滋. IaaS 環境における VM のメモリ暗号化による情報漏洩の防止. 第 117 回 OS 研究会, 2011.
5) OpenSSL Project. OpenSSL. http://www.openssl.org/.
6) Inc. Strike Force Technologies. GuardedID：Protecting Your Identity At Every Keystroke. http://www.guardedid.com/.
7) QFX Software Corporation. Keyscrambler：The best Keystroke protection. http://www.qfxsoftware.com/.
8) Takahiro Shinagawa, Hideki Eiraku, Kouichi Tanimoto, Kazumasa Omote, Shoichi Hasegawa, Takashi Horie, Manabu Hirano, Kenichi Kourai, Yoshihiro Oyama, Eiji Kawai, Kenji Kono, Shigeru Chiba, Yasushi Shinjo, and Kazuhiko Kato. BitVisor. *Proc. Intl. Conf. Virtual Execution Environments and VEE'09*, pp. 121–130, 2009.