Dependable and Secure Remote Management in IaaS Clouds

Tomohisa Egawa
Kyushu Institute of Technology
egawan@ksl.ci.kyutech.ac.jp

Naoki Nishimura
Kyushu Institute of Technology
naonishi@ksl.ci.kyutech.ac.jp

Kenichi Kourai
Kyushu Institute of Technology
kourai@ci.kyutech.ac.jp

***Abstract*—In Infrastructure-as-a-Service (IaaS) clouds, the users manage the systems in the provided virtual machines (VMs) called *user VMs* through remote management software such as Virtual Network Computing (VNC). For dependability, they often perform out-of-band remote management via the *management VM*. Even in the case of system failures inside their VMs, the users could directly access their systems. However, the management VM is not always trustworthy in IaaS. Once outside or inside attackers intrude into the management VM, they could easily eavesdrop on all the inputs and outputs in remote management. To solve this security issue, this paper proposes *FBCrypt* for preventing information leakage via the management VM in out-of-band remote management. FBCrypt encrypts the inputs and outputs between a VNC client and a user VM using the *virtual machine monitor (VMM)*. Sensitive information is protected against the management VM between them. The VMM intercepts the reads of virtual devices by a user VM and decrypts the inputs, whereas it intercepts the updates of a framebuffer by a user VM and encrypts the pixel data. We have implemented FBCrypt in Xen and TightVNC and confirmed that any keystrokes or pixel data did not leak.**

***Keywords*-Virtual machine, remote management, information leakage**

I. Introduction

Infrastructure as a Service (IaaS) provides virtual machines (VMs) hosted in data centers. Its users can set up the systems in the provided VMs called *user VMs* and use them as necessary. They usually manage their systems through remote management software such as Virtual Network Computing (VNC). To allow the users to access their systems even on failures inside their VMs, IaaS often provides *out-of-band remote management* via a special VM called the *management VM*. Unlike traditional remote management, management servers are run in the management VM, not in user VMs, and directly interact with virtual devices for user VMs, such as virtual keyboard and video devices. Even if the networks of user VMs are disconnected due to configuration errors or if the systems crash in user VMs, the users can continue to manage their VMs.

However, this out-of-band remote management increases security risks because the management VM is not always trustworthy in IaaS [1], [2], [3]. The management VM may be compromised by outside attackers if it is not well-maintained. If some of the administrators are malicious, they may mount insider attacks [4]. Such attackers can easily eavesdrop on the inputs and outputs in remote management by replacing the management servers with malicious ones. For example, they can extract passwords from keystrokes sent from the clients and take screenshots of user VMs to steal sensitive or private information. In addition, they may execute arbitrary commands inside user VMs by sending keyboard events.

To solve this security issue, we propose *FBCrypt*, which protects sensitive information in out-of-band remote management against the attackers in the management VM. FBCrypt encrypts the inputs and outputs in remote management between a VNC client and a user VM using the *virtual machine monitor (VMM)*. It can prevent information leakage via the management VM between them in a manner transparent to a user VM. The inputs to a user VM are encrypted by a VNC client and decrypted by the VMM when a user VM reads them from virtual devices. When a user VM updates a framebuffer, the pixel data are encrypted by the VMM and decrypted by a VNC client. As such, only encrypted data are passed to the management VM. In addition to the confidentiality, the VMM checks the integrity of the inputs. It verifies the message authentication code generated from inputs and detects the modification before passing the inputs to a user VM.

To guarantee the integrity of the VMM inside IaaS, FBCrypt performs remote attestation of the VMM with a trusted server outside IaaS. Remote attestation certifies the authenticity of the VMM by tamper-resistant hardware such as the trusted platform module (TPM) [5]. Although the management VM often has high privileges, the code and data of the VMM are still protected against the management VM. Thanks to the memory protection by the VMM, the attackers in the management VM cannot steal secret keys for encryption in the VMM or modify the code in the VMM to invalidate the proposed security mechanisms.

We have implemented FBCrypt in the Xen VMM [6] and TightVNC [7] for para-virtualized Linux guest operating systems. To securely pass decrypted keyboard inputs to a user VM, the VMM identifies the keyboard queue in the user VM and directly writes decrypted inputs into the queue. To encrypt pixel data on a screen for the management VM, the VMM replicates a virtual framebuffer (VFB) that holds pixel data in a user VM and provides the original and encrypted VFBs to the user VM and the management VM, respectively. It synchronizes two VFBs when the user VM updates its

VFB. Our experimental results show that the attackers in the management VM cannot steal keystrokes and screenshots and that the overheads of FBCrypt are not so large.

The organization of this paper is as follows. Section II describes issues in dependable remote management using the management VM. Section III proposes FBCrypt for protecting sensitive information in dependable remote management and Section IV explains the implementation details in Xen. Section V shows our experimental results. Section VI describes related work and Section VII concludes this paper.

II. Motivation

A. Dependable Remote Management

To manage a user VM in a cloud, the user usually connects a management client to a management server running in the user VM. This is called *in-band* remote management because the user directly accesses the user VM. Let us consider remote management through VNC. Whenever the user presses a key or pointer button or moves a pointing device, an input event is generated and sent from a VNC client to the server. When the user VM draws graphic objects in a screen, a framebuffer update is sent from the VNC server to the client in response to a request from the client. A framebuffer is an area of memory used to hold pixel data. Since the communication between the VNC client and server can be encrypted with a virtual private network or SSH tunneling, sensitive information in the inputs and outputs is protected. However, this in-band remote management is not dependable. If the user just fails the configurations of the network or firewall in the user VM, the VNC client cannot access the VM at all. If the system in the VM crashes, the user cannot obtain any information via VNC.

To increase dependability in remote management of the user VM, *out-of-band* remote management is desired. In this style of remote management, a VNC server is run in the *management VM*, as illustrated in Figure 1. The management VM is provided in the type-I VMM, which runs directly on hardware, such as Xen and Hyper-V and has privileges for accessing all user VMs. It also emulates virtual devices for each user VM. The VNC server in the management VM can directly access the virtual devices to interact with a user VM. This out-of-band remote management does not rely on the network or the VNC server in the user VM. The user can access the user VM as if he locally logged in the VM even on network failures of the VM. For example, if the user fails network configuration in the user VM, he could fix the problems by modifying the configuration through the virtual keyboard. Even when the system in the user VM crashes, the user may check kernel messages through the virtual video device.

This dependable remote management relies on the management VM, but the management VM is not always trustworthy in clouds [1], [2], [3]. Since the user VMs can be migrated between data centers, it is not guaranteed that they are

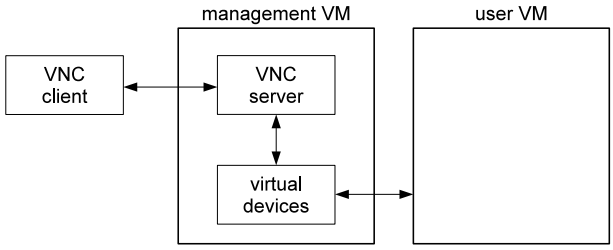


Figure 1. Dependable remote management of a user VM.

run in data centers where all the administrators are trusted. If the management VM is managed by *lazy* administrators, it may have vulnerabilities in software or configurations. In this case, vulnerable management VMs may be penetrated by outside attackers. Worse, if administrators themselves are *malicious*, they can act as inside attackers [4].

If such attackers abuse the privileges of the management VM, they can eavesdrop on or modify the inputs and outputs in remote management. Even if the network is encrypted between a client host and the management VM, the data processed by the VNC server in the management VM is not encrypted. For the inputs to a user VM, the attackers can easily obtain keyboard and pointer inputs by modifying the VNC server. The VNC server has to receive the input events from the client and write them to the virtual devices for the user VM. For example, the attackers can extract passwords and credit card numbers from keystrokes. In addition, they can send keyboard events to a user VM and make the VM execute arbitrary commands.

For the outputs from a user VM, the attackers can take screenshots of the user VM through the framebuffer in a virtual video device. The attackers may steal sensitive information displayed on the screen. For example, when passwords have to be written in configuration files, the attackers can know displayed passwords even if they cannot eavesdrop on keyboard events to the user VM. If the user uses a software keyboard to avoid keyloggers, the attackers can steal information on pressed keys displayed on the screen.

B. Threat Model and Assumptions

We assume that the management VM can be compromised by outside attackers or abused by IaaS administrators. Such attackers could take the root privilege in the management VM and even modify the operating system kernel. In this paper, we focus on the attempts to steal and modify sensitive information sent between a VNC client and a user VM in out-of-band remote management.

We assume that IaaS providers themselves are trusted. This assumption is widely accepted [2], [3]. To guarantee the trustworthiness, a small number of trusted senior administrators are responsible for the maintenance of the VMM and the hardware. If average administrators that may be lazy

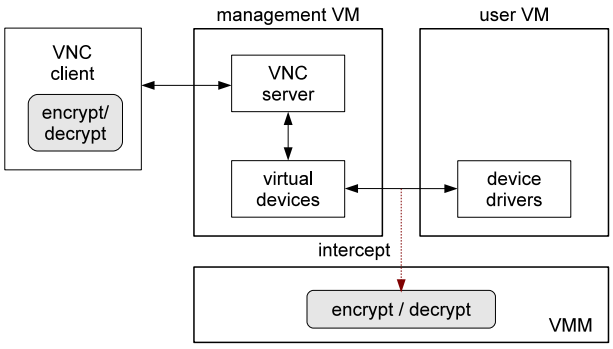


Figure 2. The architecture of FBCrypt.

or malicious maintain the VMM or the hardware, senior administrators should verify it. Consequently, the VMM is well-maintained and has no vulnerabilities. Also, we do not consider physical attacks because server rooms should be strictly protected in data centers.

III. Dependable and Secure Remote Management

To solve the security issue caused by using the untrusted management VM in IaaS, we propose *FBCrypt* for enabling dependable and secure remote management.

A. FBCrypt

FBCrypt encrypts the inputs and outputs between a VNC client and a user VM using the VMM in a manner transparent to the user VM. The attackers in the management VM between them cannot steal sensitive information included in the interaction. Figure 2 shows the architecture of FBCrypt. When the user generates an input event, the VNC client encrypts the input with a stream cipher and sends it to the VNC server in the management VM. The VNC server writes the encrypted input to the corresponding virtual device such as keyboard and pointing devices. When the user VM reads the encrypted input from the virtual device, the VMM intercepts this read and decrypts the input. It also checks the integrity of the input, that is, whether the input is not changed after sent from the VNC client. The existing operating system in the user VM can read the decrypted input from the virtual device using the traditional interface.

In FBCrypt, the attackers in the management VM and in the network cannot eavesdrop on keyboard or pointer inputs sent from a VNC client. They cannot decrypt any inputs because only the VNC client and the VMM share a session key for encryption. In addition, they cannot send arbitrary keystrokes to user VMs to execute malicious commands. The encryption and integrity check of inputs by FBCrypt prevent the attackers from generating their own input events. Also, the attackers cannot reuse encrypted inputs to perform replay attacks. Thanks to the stream cipher used by FBCrypt, encrypted inputs captured by the attackers cannot be decrypted correctly when they are sent to the VNC server later. The stream cipher encrypts even the same message in a different way. FBCrypt can detect such inputs that do not correctly decrypted by the integrity check.

For the outputs from a user VM, on the other hand, FBCrypt encrypts the framebuffer of a virtual video device in the management VM. When an application such as an X server in a user VM draws graphic objects, the user VM updates the framebuffer by accessing the device. The VMM intercepts this update and encrypts the updated pixel data. In response to a request from the client, the VNC server reads the framebuffer and sends encrypted pixel data to the client. Then the VNC client decrypts the received pixel data and draws them in its window. Encrypting the framebuffer does not cause any problems because the VNC server is not aware of the contents of the framebuffer. It simply deals with encrypted pixel data as if they were not encrypted. Since the virtual video device provides the same interface to the user VM, no modification to the operating system is needed.

The attackers in the management VM and in the network cannot eavesdrop on framebuffer updates sent to a VNC client. The sent updates are a part of the encrypted framebuffer, which can be decrypted only by either the VMM or the VNC client. The attackers in the management VM can directly access the entire framebuffer but cannot decrypt it. In addition, they cannot modify the encrypted framebuffer arbitrarily. Even if they copy some area in the encrypted framebuffer to other areas, the copied areas cannot be decrypted correctly because FBCrypt encrypts pixel data with the information on their positions. Furthermore, malicious framebuffer updates generated by the attackers can be easily detected. Since such updates cannot be decrypted correctly, meaningless objects are just drawn in its window. Consequently, the user could notice such attacks soon.

B. Protecting FBCrypt in IaaS

To guarantee the integrity of the VMM in IaaS, FBCrypt performs remote attestation of the VMM with a trusted server outside IaaS. Remote attestation certifies the authenticity of the VMM by tamper-resistant hardware such as the trusted platform module (TPM) [5]. It measures the VMM by calculating its hash value, sends the signed measurement to the trusted server, and verifies its integrity. The VNC clients can check the integrity of the VMM when connecting to the VNC server in the management VM. According to our assumption in Section II-B, only a small number of trusted senior administrators are allowed to register the hash value of a legitimate VMM to the trusted server for remote attestation.

The VMM is protected even against malicious management VMs by using the protection mechanisms of the VMM itself. The management VM usually has high privileges and can access most of the hardware without limitations. However, the management VM is still a sort of VM. Similar to the other VMs, it cannot access the state of the CPUs or

the memory used by the VMM. Therefore, the attackers in the management VM cannot modify the code in the VMM to invalidate the proposed security mechanism. They cannot steal data in the VMM, such as secret keys for encryption.

In addition, the CPU state and the memory of a user VM can be protected against the management VM by using the secure runtime environment (SRE) [2] and VMCrypt [8]. The management VM can usually access all the resources of a user VM to enable VM management such as migration. The SRE and VMCrypt encrypt the CPU state and the memory of a user VM only for the management VM. With them, the management VM cannot steal keyboard and pointer inputs read by a user VM via its CPU registers or the memory. It cannot read unencrypted pixel data to be written in the framebuffer by a user VM. Also, it cannot modify the code in a user VM so that the user VM itself sends such sensitive data to the attackers, for example.

C. Key Management

A VNC client securely shares a session key with the VMM whenever it establishes the connection to a VNC server. When it connects to a user VM, it first generates a new session key. Then the VNC client communicates with the trusted server for remote attestation and checks that the VMM on which the user VM runs is legitimate. If so, the VNC client can obtain the public key of the VMM from the server. We assume that the public key is securely registered to the server in advance. Next, the VNC client encrypts the session key with the public key and transfers it to the VNC server in the management VM. The VNC server passes it to the VMM and the VMM decrypts it with its private key. The attackers in the management VM cannot decrypt the session key because they cannot obtain the private key of the VMM. The private key is sealed by TPM and can be unsealed only when a legitimate VMM are booted.

IV. Implementation

We have implemented FBCrypt in Xen 4.1.1 [6] and TightVNC Java Viewer 2.0.95 [7]. We added only 4197 lines of code to the VMM. In Xen, domain 0 is the management VM, while domain U is a user VM. A VNC server and virtual devices are a part of QEMU running in domain 0. The target guest operating system is para-virtualized Linux.

A. Encrypting Inputs

FBCrypt securely delivers user's inputs from a VNC client to the para-virtualized keyboard driver named kbdfront in domain U, as illustrated in Figure 3. Currently, FBCrypt supports only keyboard inputs, but it can support pointer inputs in the same way. When the user presses a key, the VNC client sends an encrypted keyboard input to the VNC server and the VNC server writes it to a virtual keyboard device. Then the device passes it to the VMM using a new hypercall. In the hypercall, the VMM decrypts the encrypted

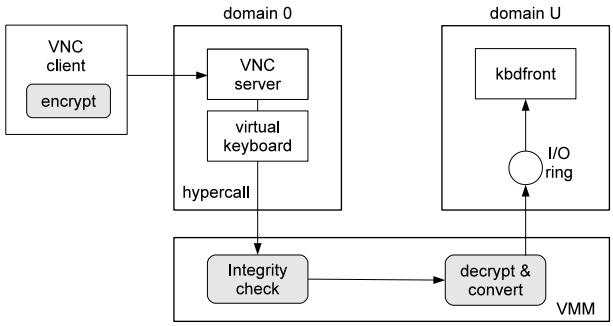


Figure 3. The encrypted delivery of keyboard inputs.

input and writes the decrypted one to the *I/O ring* for kbdfront. The I/O ring is a queue used for passing data between domains. From the I/O ring, kbdfront obtains the decrypted input. In the original Xen, virtual devices directly write unencrypted inputs to this I/O ring. In FBCrypt, the I/O ring is encrypted by SRE [2] or VMCrypt [8] and can be accessed only by domain U and the VMM.

The VMM identifies this I/O ring when domain U is booted. Originally, the VMM does not recognize the I/O ring because only domain U and domain 0 share it. On initialization, kbdfront allocates and sets up the xenkbd page, which is a memory page containing the I/O ring. Then it registers the frame number of the page to XenStore in domain 0. XenStore is a filesystem-like database containing information shared between domains. The VMM monitors this registration from domain U to domain 0 and obtains necessary information. Domain 0 cannot interfere with this identification mechanism. We describe how to monitor the interaction in Section IV-C.

For encryption and decryption of inputs, FBCrypt uses AES in CTR mode (AES-CTR) as a stream cipher. We used a modified version of the CyaSSL library [9] in the VMM and the Java standard API in TightVNC Java Viewer. AES-CTR generates a key stream by encrypting counter block values with AES and encrypts data by combining them and the next key in the key stream via XOR. When AES-CTR uses up all keys in the key stream, it increments the counter values and generates a new key stream from them. When the VNC client terminates, the internal state such as a key stream is lost. However, the VMM cannot recognize that termination and it continues to preserve the internal state. To synchronize the internal state when the VNC client re-connects to the server, the VNC server issues a hypercall for resetting the internal state in the VMM.

To check the integrity of inputs, FBCrypt uses a message authentication code (MAC). When the VNC client sends an encrypted input to the server, it calculates a SHA-1 hash value from the input before encrypted, a sequence number, and the session key for encryption. The sequence number is incremented whenever one input is handled. Thanks to

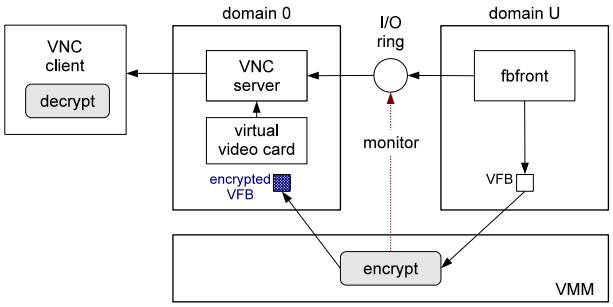


Figure 4. The encrypted delivery of pixel data.

a secret session key and a sequence number, the attackers cannot calculate the hash value correctly or reuse a pair of captured encrypted input and MAC value. To send the MAC value with an encrypted input, we have extended the RFB protocol [10] used in VNC. After the VMM decrypts the encrypted input, it compares the MAC value with the hash value calculated from the decrypted input. If the two values are different, the VMM discards the input.

For keyboard inputs, the VMM converts the encoding of keys when it writes decrypted inputs to the I/O ring for kbdfront. The I/O ring is designed so that it receives keycodes as keyboard inputs. A keycode (scancode) is data generated when a keyboard is pressed. However, decrypted inputs are keysyms, which are used in VNC. A keysym is defined by the X Window System and the same as the corresponding ASCII value for ordinary keys. To fill this gap, the VMM converts keysyms to keycodes using the mapping table. This conversion was originally done by the VNC server. In FBCrypt, the VNC server cannot perform this conversion because keyboard inputs received from the VNC client are encrypted.

B. Encrypting a Framebuffer

FBCrypt securely delivers updated pixel data from the para-virtualized video driver named fbfront in domain U to a VNC client, as illustrated in Figure 4. Originally, fbfront allocates a virtual framebuffer (VFB) in domain U and the virtual video device in domain 0 shares it. In FBCrypt, the VMM replicates the VFB for the virtual video device and encrypts the replicated one. When an application in domain U draws graphic objects, fbfront updates its own VFB and sends an update event to the virtual video device using the I/O ring for fbfront. At this time, the VMM synchronizes the original VFB with the replicated one. The VNC server periodically monitors updates in the replicated VFB and sends framebuffer updates to the client if pixel data are changed. Since the original VFB in domain U can be encrypted only for domain 0 by SRE [2] or VMCrypt [8], domain 0 cannot access unencrypted pixel data in the VFB.

To replicate the VFB for the virtual video device, the VMM first identifies the VFB in domain U. When domain U is booted, fbfront sets up the xenfb page, which contains the pointer to the VFB. When fbfront registers this page to XenStore in domain 0, the VMM intercepts it and allocates a new VFB in domain 0 as an encrypted replica. At the same time, it rewrites the xenfb page so that the page points to the replicated VFB. Since the xenfb page stores a VFB in a structure like page tables, the VMM constructs a replicated VFB with the same structure. Then it sends the rewritten xenfb page to XenStore in domain 0 and the virtual video device uses the replicated VFB. As such, the virtual video device is given an illusion as if it shared the same VFB with domain U.

The VMM synchronizes the original VFB with the replicated one when fbfront sends update events for the VFB to the virtual video device. An update event consists of an updated area in the framebuffer and is passed via the I/O ring shared between domain U and domain 0. Since the xenfb page also contains the I/O ring, the VMM can easily identify it. When an update event is sent using the I/O ring, the VMM intercepts it, encrypts the pixel data in the specified area of the original VFB, and writes them to the replicated VFB. After this synchronization, the VMM sends the update event to the virtual video device. The VNC server reads the updated pixel data from the encrypted VFB and sends them as a framebuffer update to the VNC client. The VNC client decrypts the received data and re-draws its window.

For encryption and decryption of pixel data, FBCrypt uses a modified version of RC5 [11], which uses the non-standard block size of 48 bits to accommodate two pixels. It is desirable to encrypt a VFB by one pixel because an updated area can be an arbitrary rectangle. However, the block size of 24 bits is less than the standard minimum size of 32 bits and therefore we chose the 48 bits as the block size. In addition, FBCrypt considers the position of each pixel to be encrypted. If all the pixels were encrypted by the same key without considering their positions, the attackers in domain 0 could obtain approximate screen images.

Since FBCrypt encrypts two pixels together, the VMM rewrites update events in the I/O ring so that the updated areas are aligned by the boundary of two pixels. For example, when the X-position or the width of an updated area is odd, the VMM needs to expand the updated area. If the VNC server transferred only one of two pixels, the VNC clients could not decrypt the received pixel data correctly. Fortunately, in the current implementation of Xen, the VMM does not need to expand updated areas. The X-position and the width of an updated area are always aligned by 8 bits in text mode. In graphics mode, they are always the same as the display width, which is usually even.

C. Monitoring a XenStore Ring

To intercept the registration to XenStore in domain 0 from domain U, the VMM monitors a XenStore ring, which is shared between domain U and domain 0, as

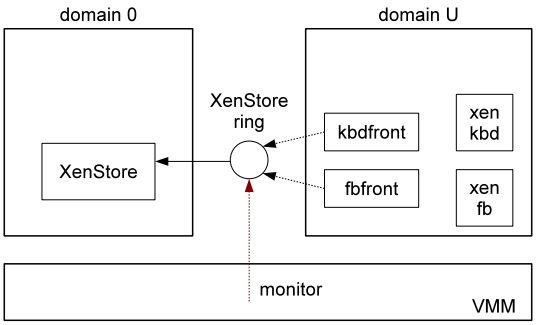


Figure 5. Intercepting the registration to XenStore.

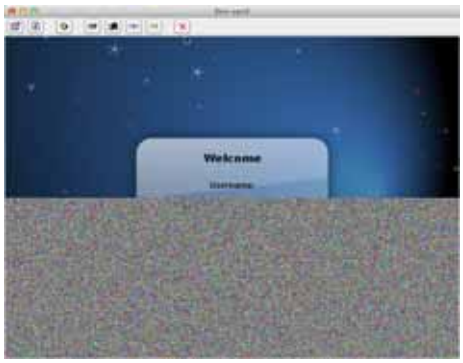


Figure 6. The screen whose only lower half is encrypted.

shown in Figure 5. When domain U registers data to XenStore, it writes a pair of path and value to the XenStore ring. For example, the path for a virtual keyboard is `device/vkbd/0/page-ref` and the value is the frame number of the xenkbd page. The VMM inspects the XenStore ring when domain U sends an inter-domain event to domain 0 using a hypercall after it writes data to the ring. If the written path is one of the above, the VMM obtains the information on a shared page.

Since the original VMM in Xen is not aware of a XenStore ring, our VMM identifies the ring from the start_info page, which points to the page containing the XenStore ring. This page is passed from domain 0 to domain U when domain U is booted. The VMM first obtains the virtual address of the start_info page from the RSI register of a virtual CPU for domain U. The address is set by domain 0 at the build time of domain U. Then the VMM translates the virtual address into the frame number of the start_info page.

V. Experiments

We conducted experiments for confirming the prevention of information leakage by FBCrypt and for examining its overheads. For server and client machines, we used two PCs with one Intel Core 2 Quad processor Q9550 2.83 GHz and a Gigabit Ethernet NIC. In the server machine, we ran a modified version of Xen 4.1.1 for the x86-64 architecture. The server machine had 4 GB of memory and we allocated 512 MB for domain U. We ran Linux 3.1.1 in domain 0 and Linux 2.6.32.21 in domain U. In the client machine, we ran a modified version of TightVNC Java Viewer 2.0.95 on Linux 2.6.38.8. The client machine had 8 GB of memory.

We used AES-CTR with a 128-bit key for encrypting inputs and SHA-1 for calculating MAC. For RC5 used for encrypting pixel data, we used a 48-bit block, a 192-bit key, and 16 rounds.

A. Attempts at Eavesdropping

To confirm that FBCrypt prevents domain 0 from eavesdropping on keystrokes, we embedded a custom keylogger into the VNC server in domain 0. This keylogger simply recorded keystrokes sent from the VNC client. Using the VNC client, we logged in to domain U by typing a user name and a password. Without FBCrypt, the plain-text password was recorded. When FBCrypt was enabled, the password was encrypted by the VNC client and the keylogger recorded encrypted one. Nevertheless, the user could log in domain U as usual.

Next, we embedded a program for screen capture into the VNC server to confirm that FBCrypt prevents domain 0 from stealing the pixel data of domain U. This screen capture periodically saved the pixel data in the VFB to a file. Figure 6 shows the screen whose lower half is encrypted for demonstration. With FBCrypt disabled, the pixel data was recorded as in the upper half and the attackers could read displayed texts. FBCrypt could randomize the pixel data as in the lower half, so that the attackers cannot recognize the contents.

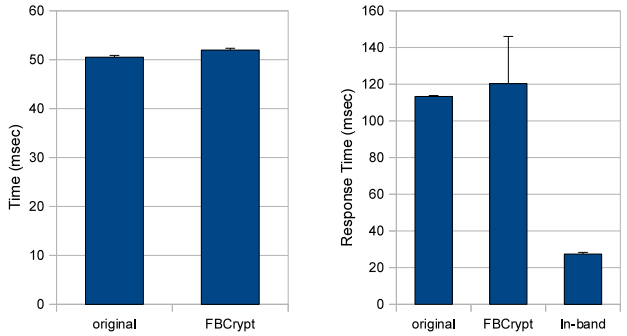
Table I
The overheads in the client and server sides (msec).

	Client-side	Server-side
Key input	0.802	0.015
Full-screen update	47	37

B. Overheads

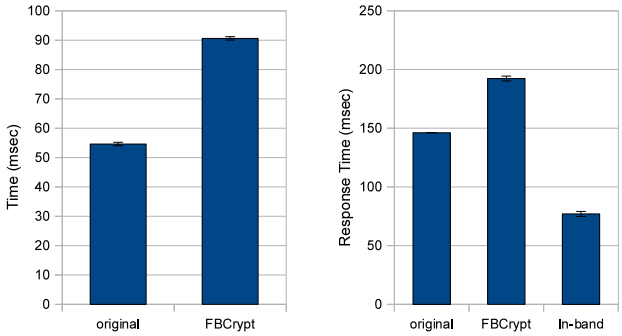
First, we examined the overheads in a keyboard input when we pressed one key to a user VM. In the client side, we measured the time from when the VNC client received a keyboard input until it sent the input event to the VNC server. In the server side, we measured the time from when the VNC server received the event until the input was written to the I/O ring. We measured these 100 times and obtained the average. As shown in the first row of Table I, the overhead in the client side was 802 μs. Most of the overhead comes from sending extra data for the MAC.

Next, we examined the overheads in a framebuffer update when we updated the full screen (800×600) of a user VM. In the server side, we measured the time from when the VMM intercepted the update event for the VFB until it completed to synchronize VFBs. In the client side, we measured the time from when the VNC client received pixel data until it



(a) Server side (b) Client side

Figure 7. The response time of a keyboard input.



(a) Server side (b) Client side

Figure 8. The response time of a full-screen update.

drew the data in its window. We measured these 10 times and obtained the average. The second row in Table I shows the overheads. Note that these overheads become smaller if an updated area is smaller.

C. Response Time of a Keyboard Input

To exclude the influences of the interaction between the VNC client and server, we first examined the response time of a keyboard input in the server side. We measured the time needed from when the VNC server received a keyboard event until the virtual video device received an update event for the VFB from domain U. Domain U displayed the character corresponding to an input on the screen and generated an update event. We measured the response time 100 times and obtained the average and standard deviation. As shown in Figure 7(a), the increase of the response time was 1.5 ms in FBCrypt. This overhead is caused by decrypting a keyboard input and encrypting pixel data for a displayed character.

Next, we examined the response time in the client side. We measured the time from when the VNC client received a keyboard input from the user until it completed to draw updated pixel data sent from the server in its window. We performed this measurement in the original Xen and FBCrypt. For comparison, we also measured the response time using in-band remote management, which ran a VNC server in domain U. Figure 7(b) shows the results. The response time in FBCrypt was 7.0 ms longer than that in the original Xen.

Compared with these two systems, the response time in in-band remote management was too short, which was 27 ms. One cause of the difference is the overhead of out-of-band remote management, but another is due to the timer interval used by the VNC server implemented in Xen. The VNC server checks updates in the VFB at some interval and sends the updates to the client if any. The interval is set to 30 ms when the VNC server receives a keyboard event. Since the framebuffer update caused by the keyboard input does not occur in 30 ms in out-of-band remote management, the next interval is set to 80 ms. This means that the framebuffer update occurs in 110 ms after a keyboard event is received. This is the reason why the response times in the original Xen and FBCrypt are more than 110 ms.

D. Response Time of a Full-screen Update

We first examined the response time of a full-screen update in the server side. We measured the time as explained in the previous section. We ran a screen saver that made the screen black out in advance. By the keyboard input, domain U terminated the screen saver, re-drew the full screen, and generated an update event. We measured the response time 10 times and obtained the average and standard deviation. Figure 8(a) shows the results in the original Xen and FBCrypt. The response time in FBCrypt was 36 ms longer than that in the original Xen. Most of this overhead is caused by encrypting 800×600 pixel data for the full screen.

To examine the response time in the client side, we measured the time as in the previous section. As shown in Figure 8(b), the increase of the response time in FBCrypt was 46 ms. This is nearly equal to the time needed for decrypting all pixel data in the screen. Thanks to the long timer interval used by the VNC server, as analyzed in Section V-B, the overhead of the synchronization in FBCrypt was almost hidden in the server side. For in-band remote management, the response time was shorter because of the same reasons as the previous section.

VI. Related Work

Xoar [12] runs QEMU including a VNC server in a separated VM called QemuVM. In Xen, QEMU can be run in a special VM called a stub domain. Since a small operating system named mini-os is run in these special VMs, it is more difficult for the outside attackers to compromise them. However, the VNC server can be compromised because it is open to the Internet. If it is compromised, the attackers can eavesdrop on sensitive information in remote management. In addition, this architecture does not improve the security against insider attacks by IaaS administrators.

VMware vSphere Hypervisor [13] runs a VNC server in the VMM and enables dependable remote management without the management VM. The VNC server in the VMM can directly access virtual devices for user VMs. In vSphere, information leakage via the management VM does not occur. However, the attackers can steal sensitive information in remote management if they compromise the VNC server in the VMM. They can take over even the control of the VMM itself. FBCrypt preserves the confidentiality in remote management by the VMM even in the case that the VNC server in the management VM is compromised.

The secure runtime environment (SRE) [2] and VMCrypt [8] prevent information leakage from the memory of the user VMs to the management VM. When the management VM maps memory pages of a user VM, the VMM encrypts their contents. This architecture is complementary to FBCrypt in that it prevents the management VM from stealing information inside the user VMs. Note that the management VM cannot use the VFB encrypted by these systems, instead of the one replicated and encrypted by FBCrypt. These systems synchronize the unencrypted and encrypted VFBs only on memory mapping.

CloudVisor [3] runs the security monitor underneath the VMM and encrypts the memory and storage of the user VMs in the security monitor. Since it distrusts not only the management VM but also the VMM, it can prevent information leakage even from the VMM. However, the security monitor does not encrypt the inputs and outputs in remote management.

BitVisor [14] can prevent information leakage from storage and network of the user VM. It is similar to FBCrypt in that the VMM transparently encrypts I/O of the user VM without the help of the management VM. However, BitVisor does not provide the means of remote management.

VII. Conclusion

In this paper, we proposed FBCrypt for enabling dependable and secure remote management in IaaS clouds. To prevent information leakage via the management VM in out-of-band remote management, FBCrypt encrypts the inputs and outputs between a VNC client and a user VM using the VMM. The VMM decrypts the inputs encrypted by a VNC client when a user VM reads them. When a user VM updates a framebuffer, the VMM encrypts the updated pixel data, which are decrypted by a VNC client. As such, sensitive information is protected against the management VM, which is located in the middle. We have implemented FBCrypt in Xen and TightVNC Java Viewer and confirmed that the security in dependable remote management was improved and that the overheads of FBCrypt were not so large, particularly, for emergency use.

Currently, we are working on supporting fully-virtualized guest operating systems in FBCrypt. Unlike para-virtualized Linux, the guest operating systems access the virtual devices through standard interfaces without I/O rings. Future work is to apply FBCrypt to other remote management software such as SSH.

Acknowledgment

This research was supported in part by JST, CREST.

References

[1] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," in *Proc. Workshop Hot Topics in Cloud Computing*, 2009.

[2] C. Li, A. Raghunathan, and N. K. Jha, "Secure Virtual Machine Execution under an Untrusted Management OS," in *Proc. Intl. Conf. Cloud Computing*, 2010, pp. 172–179.

[3] F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization," in *Proc. Symp. Operating Systems Principles*, 2011, pp. 203–216.

[4] TechSpot News, "Google Fired Employees for Breaching User Privacy," http://www.techspot.com/news/40280-google-fired-employees-for-breaching-user-privacy.html, 2010.

[5] Trusted Computing Group, "TPM Main Specification," http://www.trustedcomputinggroup.org/, 2011.

[6] 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. Symp. Operating Systems Principles*, 2003, pp. 164–177.

[7] TightVNC Group, "TightVNC," http://www.tightvnc.com/.

[8] H. Tadokoro, K. Kourai, and S. Chiba, "Preventing Information Leakage from Virtual Machines' Memory in IaaS Clouds," *IPSJ Online Transactions*, vol. 5, pp. 156–166, 2012.

[9] yaSSL, "CyaSSL Embedded SSL Library," http://www.yassl.com/.

[10] T. Richardson, "The RFB Protocol Version 3.8," http://www.realvnc.com.

[11] R. L. Rivest, "The RC5 Encryption Algorithm," in *Proc. Workshop Fast Software Encryption*, 1994.

[12] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield, "Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor," in *Proc. Symp. Operating Systems Principles*, 2011, pp. 189–202.

[13] VMware Inc., "VMware vSphere Hypervisor," http://www.vmware.com/.

[14] T. Shinagawa, H. Eiraku, K. Tanimoto, K. Omote, S. Hasegawa, T. Horie, M. Hirano, K. Kourai, Y. Oyama, E. Kawai, K. Kono, S. Chiba, Y. Shinjo, and K. Kato, "BitVisor: A Thin Hypervisor for Enforcing I/O Device Security," in *Proc. Intl. Conf. Virtual Execution Environments*, 2009, pp. 121–130.