Eternal Sunshine of the Spotless Machine: Protecting Privacy with Ephemeral Channels

Alan M. Dunn, Michael Z. Lee, Suman Jana, Sangman Kim,
Mark Silberstein, Yuanzhong Xu, Vitaly Shmatikov, Emmett Witchel
University of Texas at Austin

OSDI 2012
October 8, 2012

Wanted: Application Privacy

- Goal: Run programs without leaving traces

VoIP conversation with lawyer

Biomedical researcher accessing data

Website access

- Current state: Private browsing
  - Popular feature in web browsers
  - Ideal: When private browsing session terminates, all traces erased

A Privacy Problem

- Private browsing unachieved
  - Evidence of site visits leaks into OS [Aggrawal, 2010]
- Problem: No system support
  - Applications interact with user and world
  - Data leaks into OS, system services
  - Applications cannot remove traces they leave

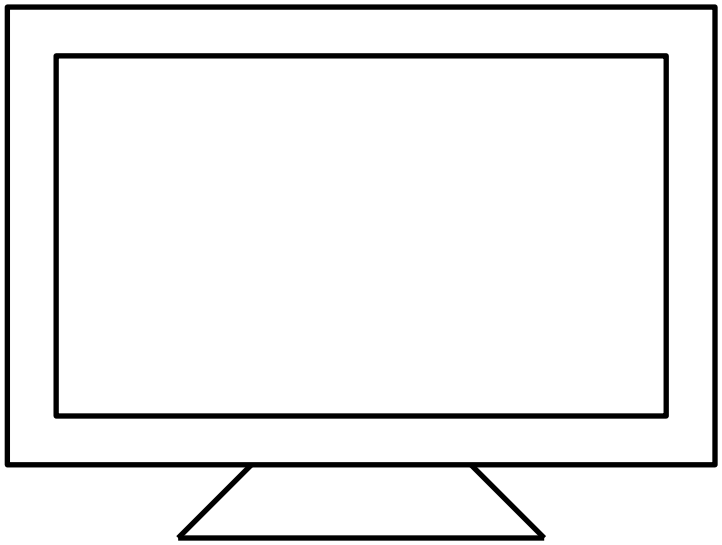
Example: Browsing a Website

Network

Audio

What traces still remain on the computer?

Leaks From Browsing



Network

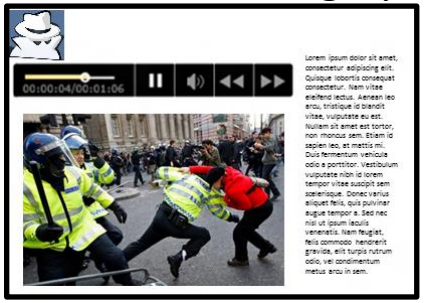
Memory contents:

Complete packets, like:

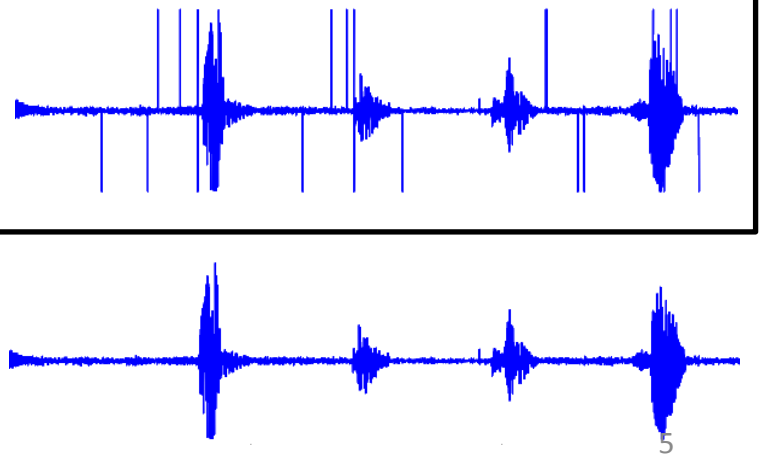
```
HTTP/1.1 200 OK
Date: Mon, 17 Sep 2012 …
Server: Apache/2.2.14 …
…
```

Audio

X server caches, graphics drivers



PulseAudio server



Secure Deallocation Is Not Enough

- **Secure deallocation**: Zero memory when freed
  - Research implementation [Chow, 2005]
  - PaX: Security patch for Linux kernel
- Sensitive data remains allocated
  - X caches, PulseAudio buffers not freed

Resisting a Strong Adversary

- Goal: Provide **forensic deniability** – no evidence left for non-concurrent attacker
- Once program terminated, protection maintained under extreme circumstances

Root-level compromise
(after program terminates)

Computer physically seized

Goals

- Provide privacy
  - Private sessions with forensic deniability
- Maintain usability
  - Simultaneous private/non-private applications
  - Support a wide variety of private applications
  - “Pay as you go” - costs only for private programs
  - Impose low overhead

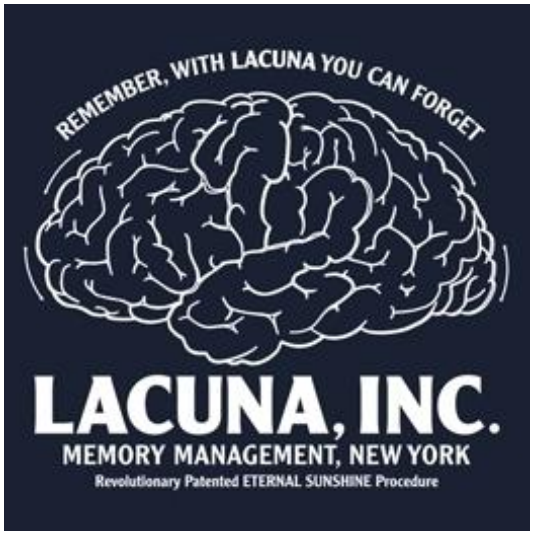
Lacuna

- System to accomplish our privacy and usability goals
- Host OS (Linux), VMM (QEMU-KVM) modified
- Applications unmodified

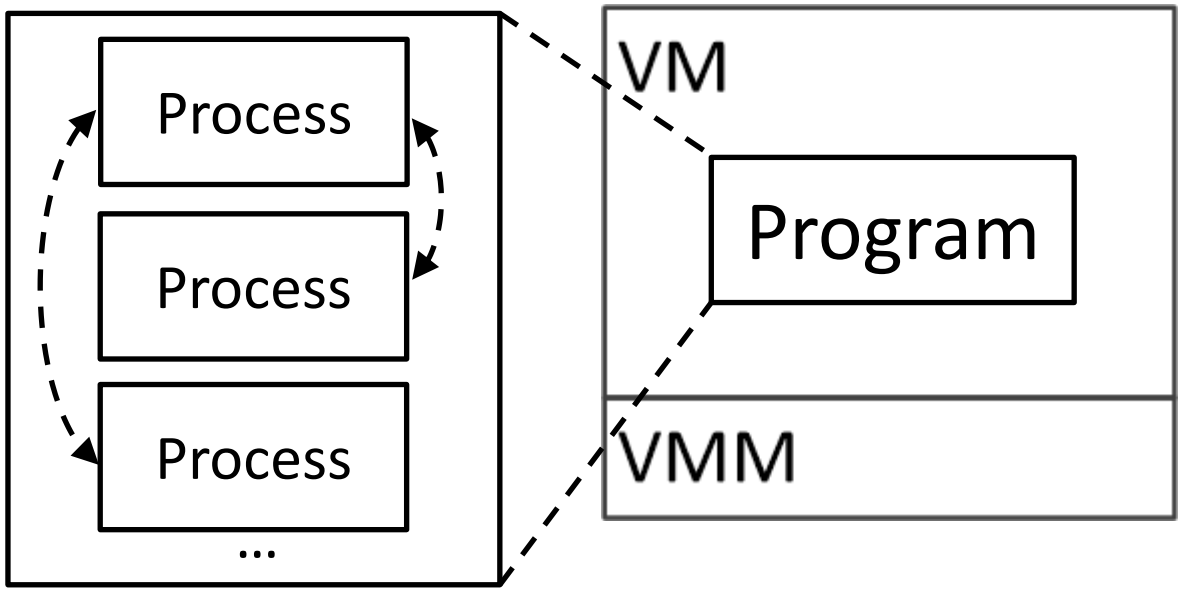
la·cu·na [luh-kyoo-nuh]
1. a gap or missing part, as in a manuscript, series, or logical argument...

Outline

- Design
  - Erasable program container
  - Allow communication with peripherals
- Evaluation
  - Lacuna provides privacy
  - Lacuna maintains usability



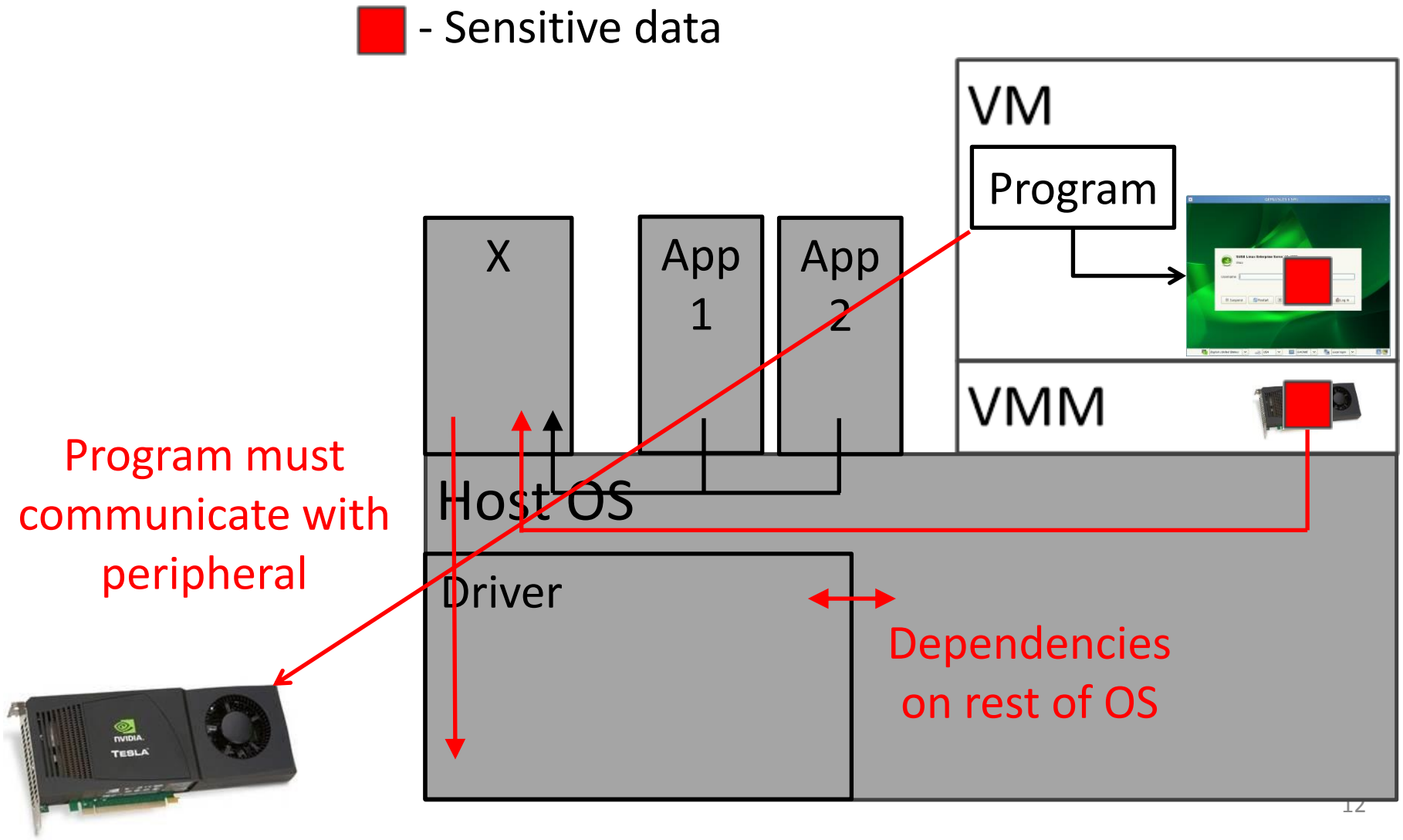
Erasable Program Container



VM contains Inter-Process Communication

VM alone is insufficient

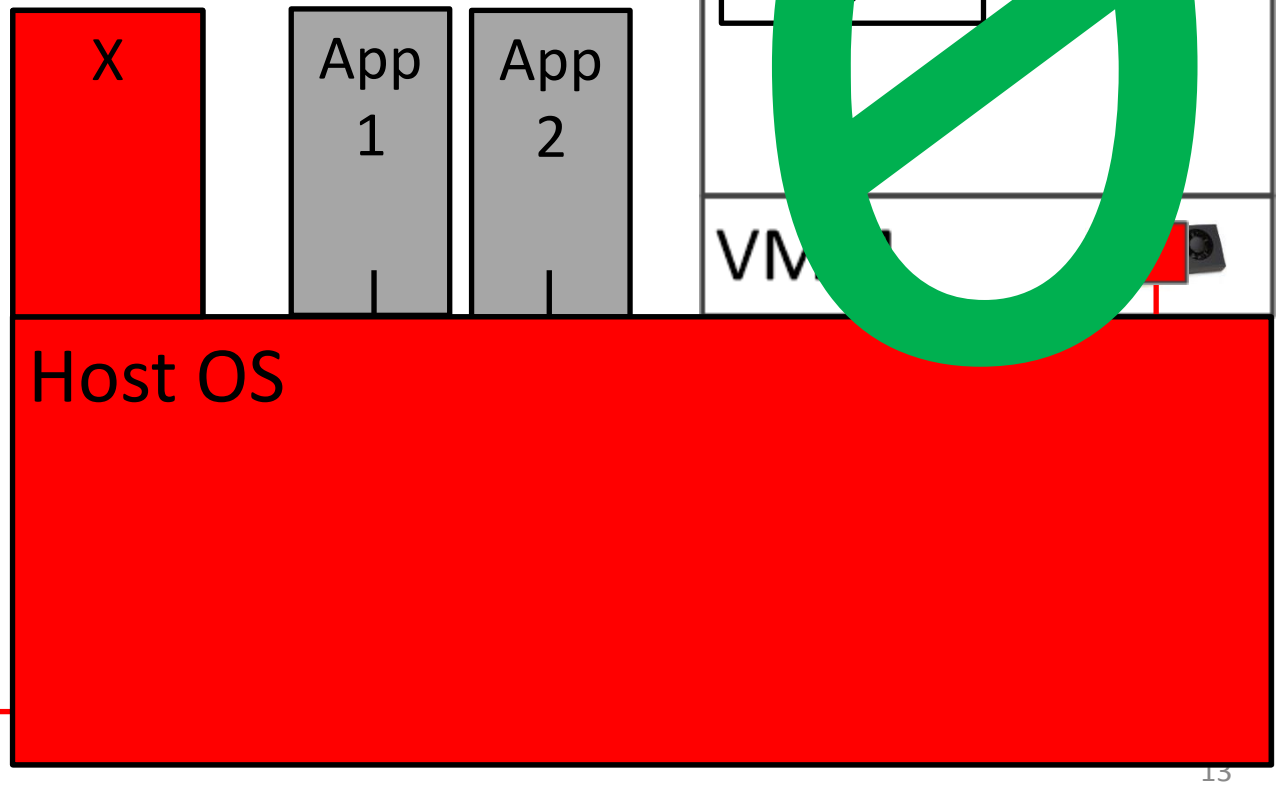
Communicating with Peripherals



Communicating with Peripherals

- Sensitive data

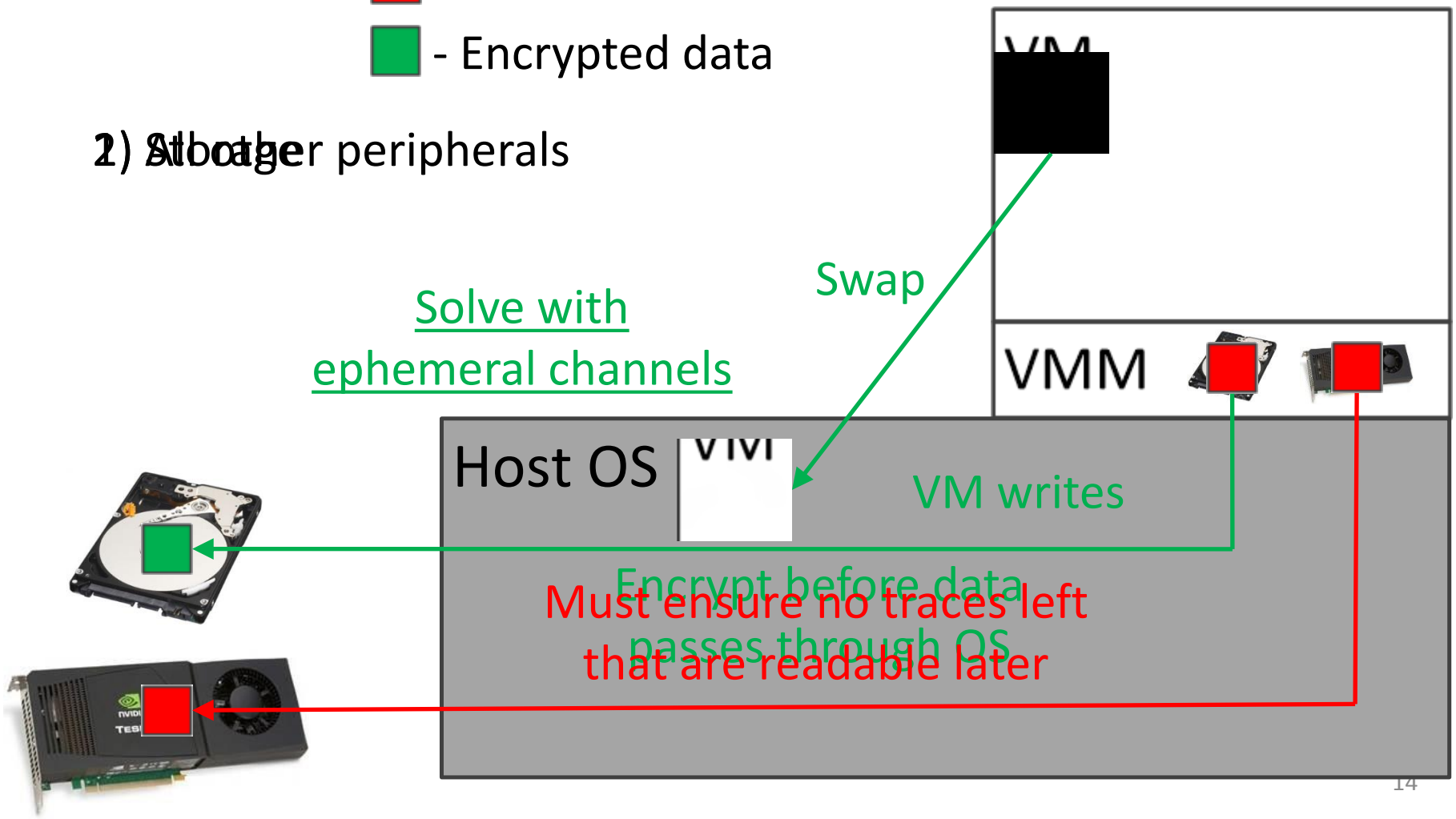
Code with potential data exposure



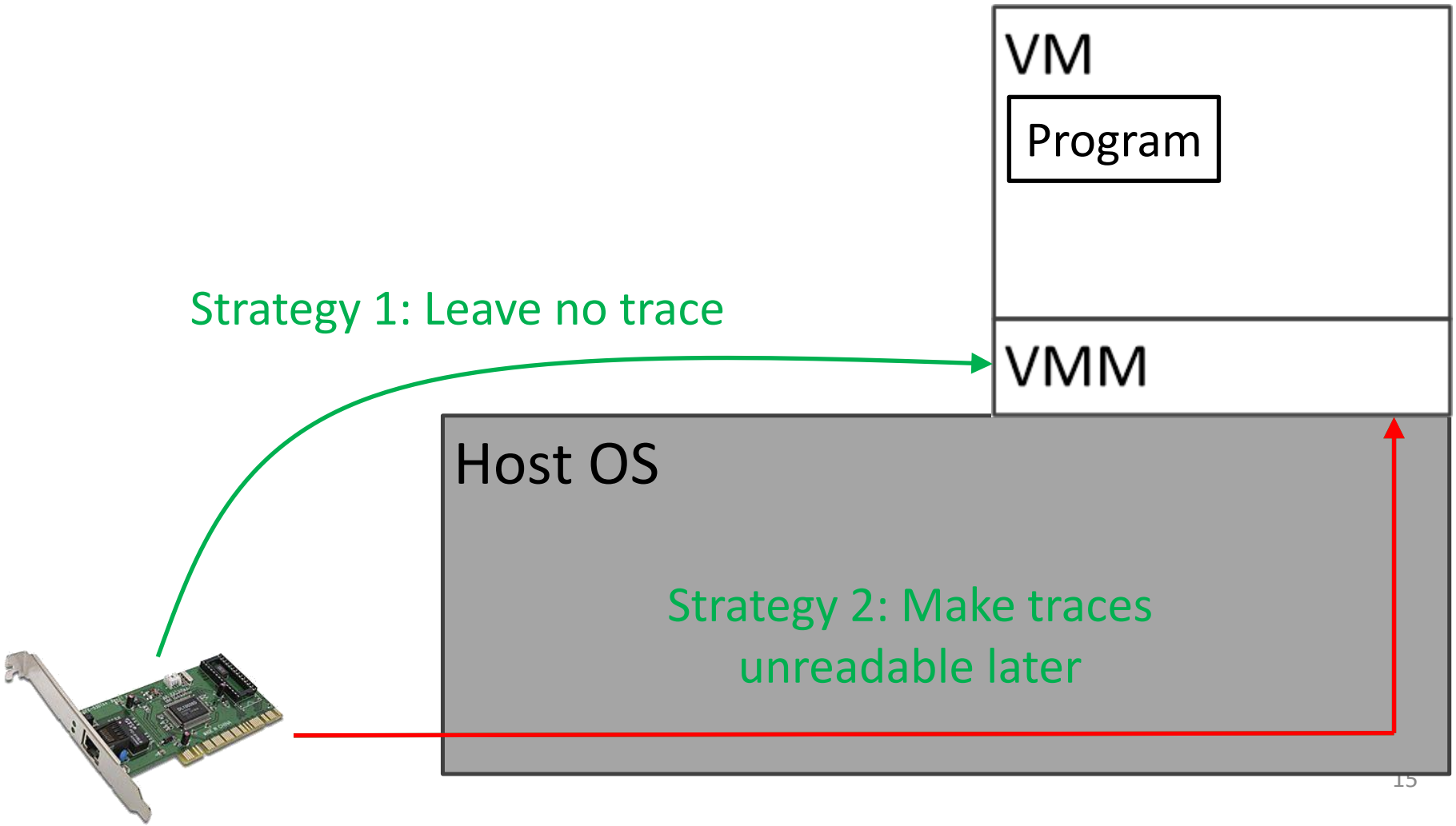
Two Peripheral Types

- Sensitive data
- Encrypted data

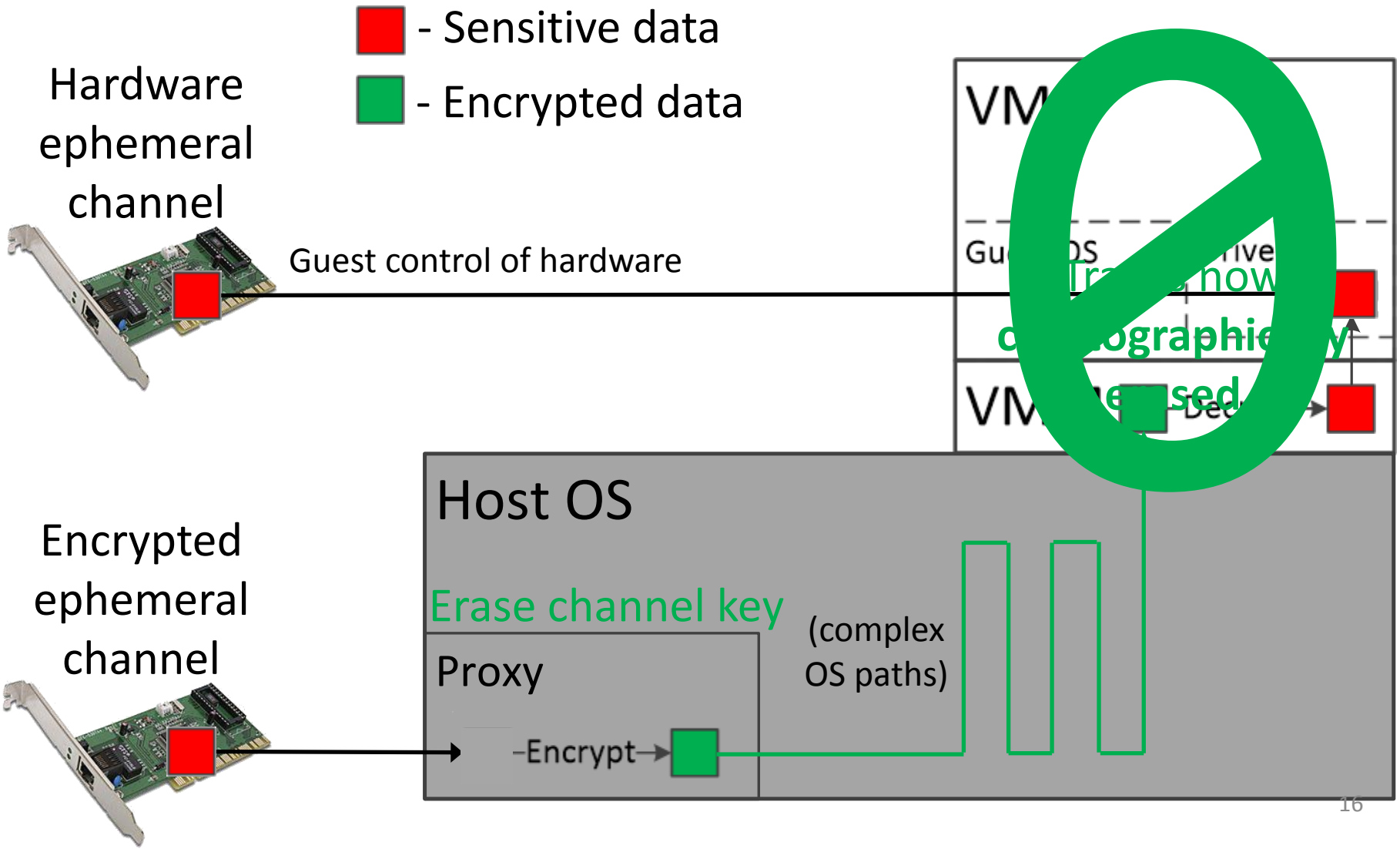
1) Storage
2) All other peripherals



Ensuring No Readable Traces

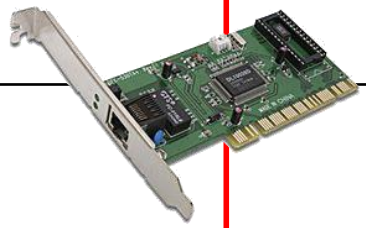


Ephemeral Channels



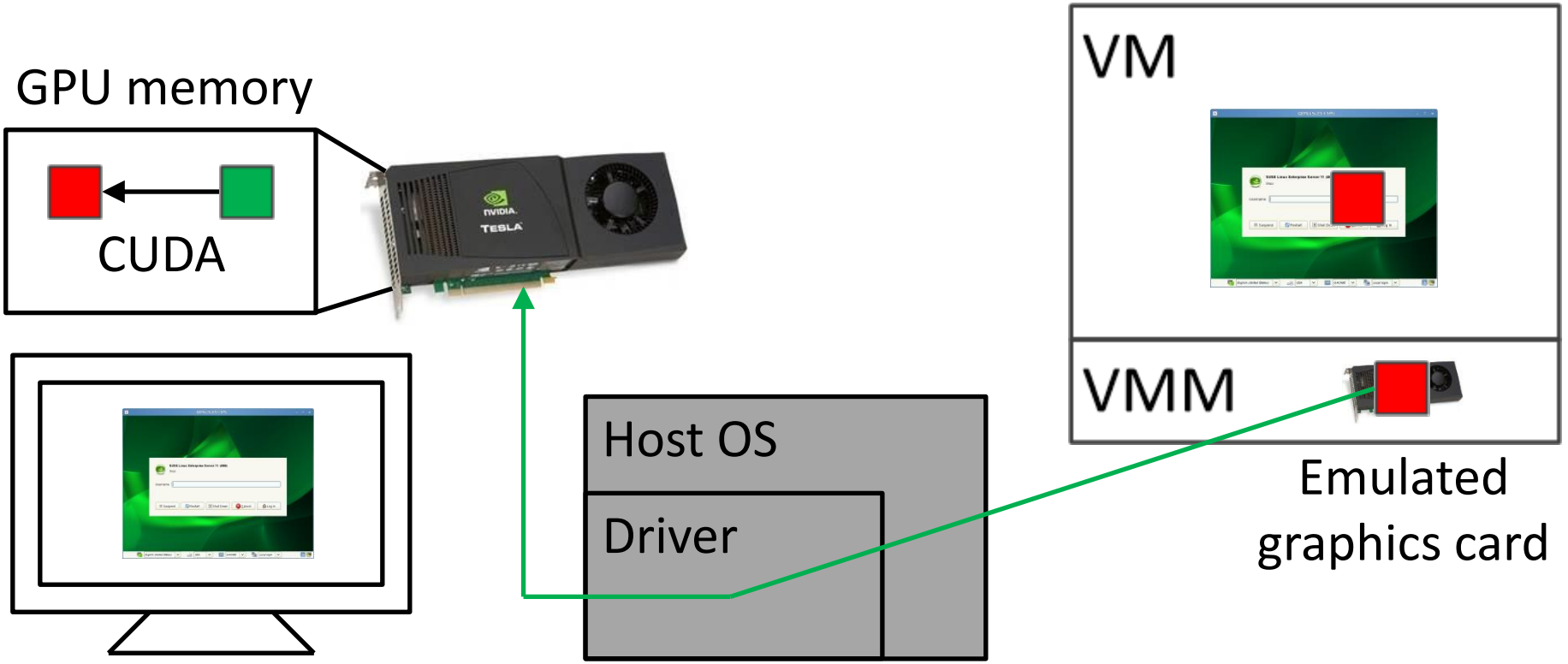
Channel Type Comparison

	Hardware	Encrypted
Host drivers unmodified	✓	✗
Host code never sees unencrypted data	✓	✗
Hardware virtualization support unnecessary	✗ (No graphics)	✓
Guest modification unnecessary	✗	✓ (Run Windows, Linux, unmodified programs)

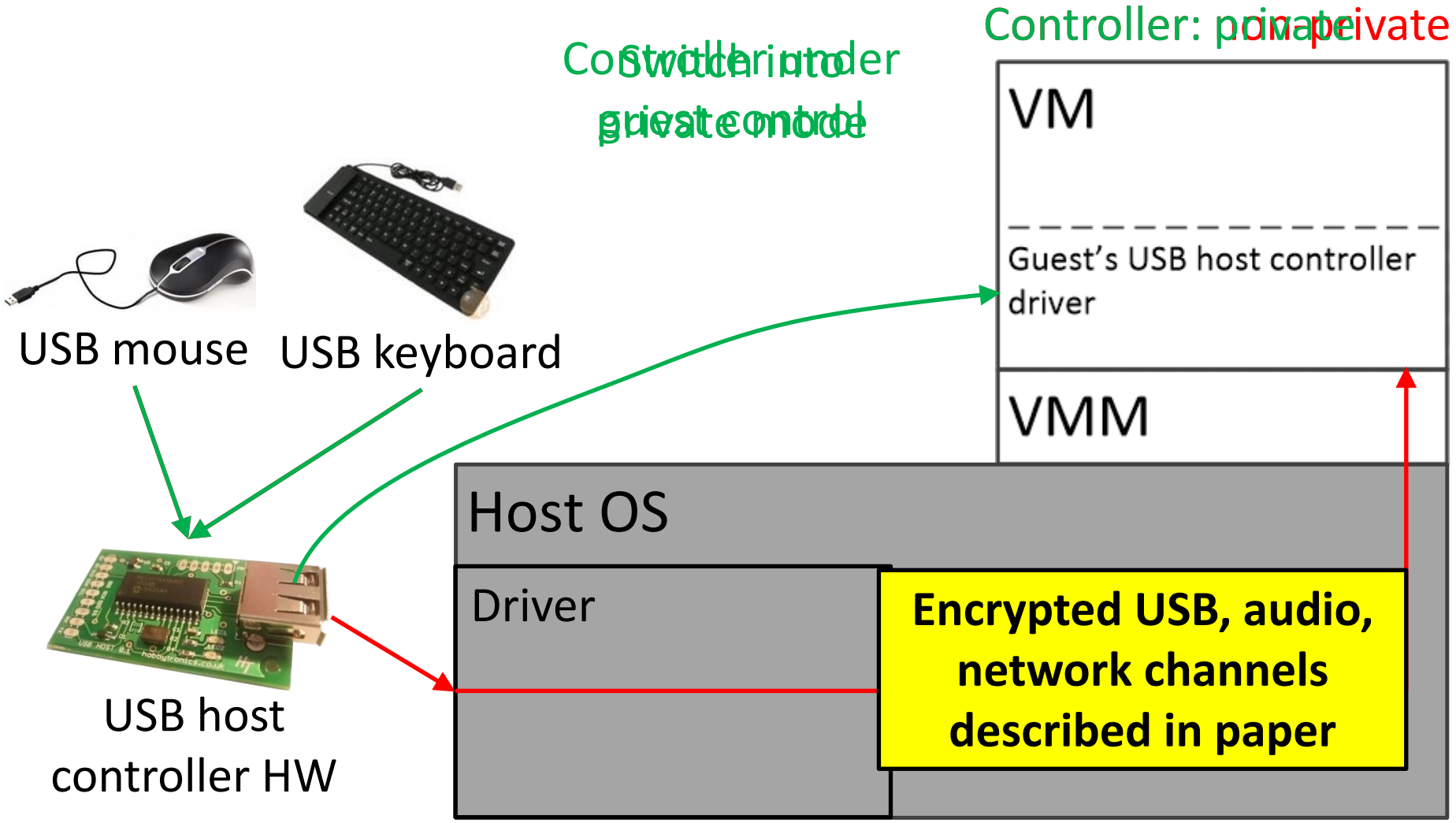


Encrypted Graphics Channel

- No hardware virtualization support for graphics
- Solution: Encrypt VM output to GPU memory



Hardware USB Channel



Sanitizing Storage

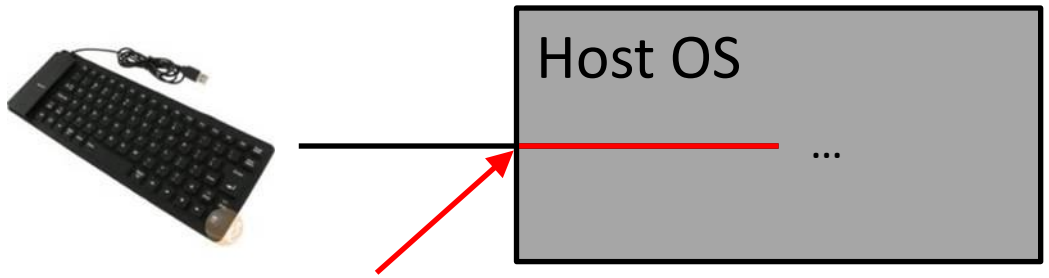
- Encrypt VM writes to storage
  - VM image file unmodified
  - **Diffs file** contains VM writes to storage
  - Diffs file encrypted
- Leave no evidence of which storage locations read
  - Free buffer cache pages for VM image file only
- Encrypt swapped memory from private VM
  - Encrypt swapped pages for VMM process only
- Encryption keys erased on VM exit
- Techniques here “pay as you go”

Evaluation

- Lacuna provides privacy
  - Measure that Lacuna does not leak private data
  - Quantify size of code that handles sensitive data
- Lacuna maintains usability
  - Low switch time to private environment
  - Application performance near that of running program in VM
- More evaluation in paper

Lacuna Protects Privacy

- Experiment to locate leaks
- Inject random “tokens” into peripheral I/O paths, scan memory to locate [Chow, 2005]
- Tokens <u>almost always</u> found without Lacuna
- Tokens <u>never</u> found with Lacuna



Little Code Handles Sensitive Data

Subsystem	Lines of Code
Graphics	725 (CUDA)
Sound	200 (out) 108 (in)
USB	414
Network	208

- Measurements are lines of code outside of QEMU that handle unencrypted data
  - Data within QEMU erased at VM exit

Time to Switch to Private Programs is Low

Channel Type	Switch Time (s)
USB passthrough (encrypted)	
keyboard	1.4 ± 0.2
keyboard + mouse	2.3 ± 0.2
PCI assignment (hardware)	
keyboard	2.4 ± 0.2
keyboard + mouse	3.8 ± 0.2

- USB driver disconnect significant (0.8-1.0 s)
- Switch time achieved by eliminating two extra disconnects in guest USB initialization

Impact on Full-System Workloads is Low

- Benchmarks
  - MPlayer: Watch video in across network
  - Firefox: Browse Alexa top 20 websites
  - LibreOffice: Create 2,994-character, 32-image document
- No execution slowdown, higher CPU utilization

	Video (75 s)	**Browser** (20 s)	**Office Suite** (175 s)
QEMU	32.2 ± 7.4	25.9 ± 1.3	8.1 ± 1.2
Lacuna	49.7 ± 0.3 (+ 17.5)	46.2 ± 1.5 (+ 20.3)	21.1 ± 0.6 (+ 13.0)

Measurements are % CPU util

Worst case: additional 20 percentage points

- CPU utilization lowered by hardware AES (AES-NI)

Conclusion

- Modern computer systems leak secrets
- Lacuna provides **forensic deniability**: secrets removed after program termination
- **Ephemeral channels** provide private peripheral I/O
- Lacuna runs full-system workloads efficiently