中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

Copker: Computing with Private Keys without RAM

Le Guan[%#], *Jingqiang Lin*[%#], Bo Luo[&], Jiwu Jing[%#]

%. Data Assurance & Communication Security Center, Chinese Academy of Sciences

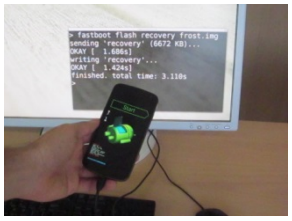
#. Institute of Information Engineering, Chinese Academy of Sciences

&. Department of Electrical Engineering and Computer Science, University of Kansas

Remanence Effect

- The remanence effect of RAM
  - The contents in RAM fade away gradually after power off, in several minutes or hours (low temperature)

- Lest We Remember: Cold Boot Attacks on Encryption Keys
  - USENIX Security 2008



- FROST: Forensic Recovery Of Scrambled Telephones
  - ACNS 2013

Photo from https://citp.princeton.edu/research/memory/media/
http://www1.informatik.uni-erlangen.de/frost

Cold-Boot Attack

- Based on the remanence effect, the cold-boot attacks can steal sensitive information in RAM
  - such as cryptographic keys
- An example:
  1. Steal a power-on laptop
     - The cryptographic keys are in RAM
  2. In low temperature, extract the RAM chips
  3. Put the RAM chips in another machine
  4. Read out the sensitive information

- **The system security mechanisms are useless**
  - **E.g., password, access control, encryption, authentication**

How to Prevent Cold-Boot Attacks?

- Another equal question
  - Where to store the sensitive information?
  - Not in RAM

- CPU-bound solution, for cryptographic keys
  - During the computation
  - No sensitive information in RAM, but within CPU

The storage units in CPU: **Register** *vs.* **Cache**

	Pro.	**Con.**
Register	**Easy to control** e.g., mov eax 0x3344	**Limited space** 256-bit register (AVX) 32/64-bit general register
Cache	**Large-size storage** L1D: 32 KB per core L2/L3: several MB	**Difficult to control** Limited instructions to control caches

Existing Register-based Solutions

- AES
  - *TRESOR* – USENIX Security *Amnesia* – ACSAC
- RSA - much more storages are needed
  - PRIME – ACSAC 2013, 2048-bit RSA
  - Low performance, about 10%
    - "by factor 9 in comparison to the best PolarSSL algorithm, and slower by factor 12 in comparison to the OpenSSL implementation"
    - "store well-chosen intermediate values of RSA in RAM"; otherwise, even worse
- **Can we find a different way? Better performance?**

Copker: Computing with Private Keys without RAM

- Our cache-based solution
- Implement 2048/3072/4096-bit RSA in caches
  - Good performance and reasonable overhead

- The private key and the intermediate states, **only in on-chip CPU caches** (and registers)
  - No cold-boot attack on caches, until now

How to Keep the Data in Caches?

- A similar question
- How to put an elephant in a refrigerator?

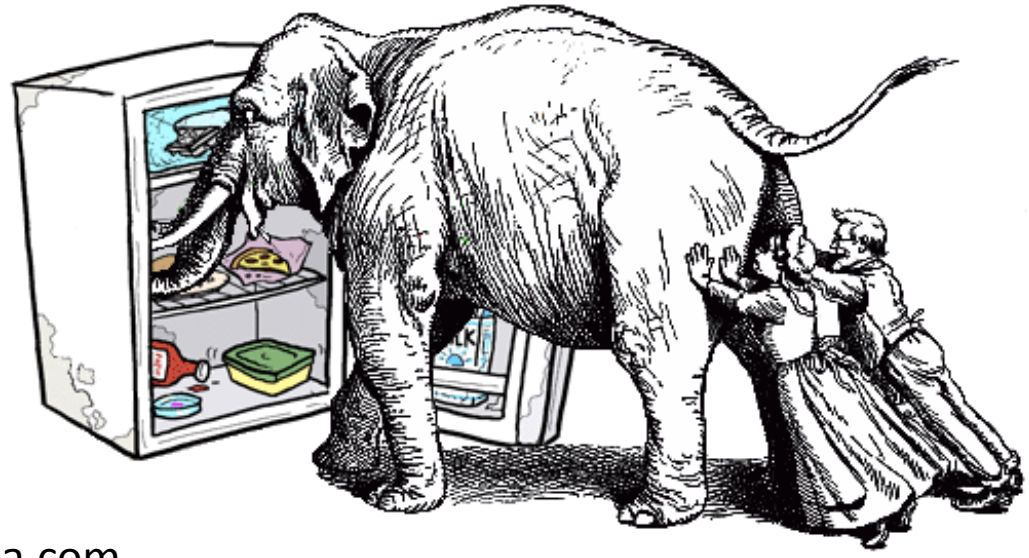


Photo from http://uncyclopedia.wikia.com

The “Standard” Answers

- How to put an elephant in a refrigerator?
  1. Open the refrigerator, put the elephant in
  2. Close the refrigerator

- How to Keep the Data in Caches?
  1. Take the sensitive data in caches
  2. Never leak them out of caches

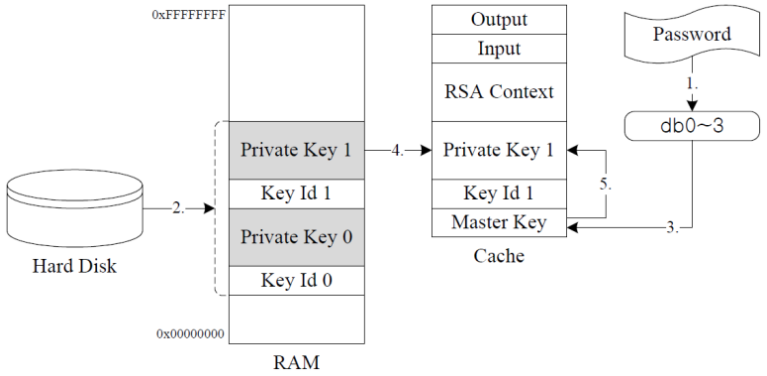
- Easy to understand, but **not so easy to do**

1. Take the sensitive data in caches

- Cache-fill Mode: **Write-Back (WB) Mode**
  - The most common mode
  - Accessing data will take these data into caches; and the following operations are performed only in caches

- Our basic idea – **A reserved space in kernel**
  - Reserve an address space in kernel
  - Perform the RSA computations in this reserved space, under WB mode

Detailed Steps



1. Employ TRESOR to store **an AES master key** in the registers
   - TRESOR: a register-based AES solution against cold-boot attacks
2. Load **encrypted** RSA private keys into RAM, from hard disks

On each request, **in the reserved space in kernel**, **WB mode**

3. Copy the AES master key
4. Read the encrypted RSA private key
5. AES-decrypt the private key, and perform RSA computations
6. Clean all data, except the result

- **During Steps 3-6, no event lets the sensitive data be synchronized from caches into RAM**

2. Never leak them out of caches

- In the following cases, the data in caches may be synchronized into the RAM
  a. Task scheduling and interrupt
  b. Input/output of the RSA computation
  c. Data access during the computation
  d. Memory access by share-cache cores
  e. Cache control by share-cache cores

Task Scheduling and Interrupt

- If a task is suspended and not resumed soon, the occupied cache lines may be evicted to RAM
- Interrupt – similar results

- Countermeasure in Copker, when computing
  - preempt_disable() to disable kernel preemption
  - local_irq_save() to disable interrupts locally
- Non-maskable interrupts (NMIs)
  - NMI handler needs to be modified to clean the sensitive data immediately, once NMI is triggered

Input/Output of the RSA Computation

- The Copker RSA computation is implemented as system functions in kernel
- When a user-space process calls these functions, only the reserved space is protected
  - But the caller's stack is not in the reserved space
  - Sensitive data may be generated in stack, on computing

- Countermeasure: Stack Switch
  - Change `ebp`/`esp` firstly
  - Let the stack be within the reserved space

Data Access during the Computation

- Ensure that all data accesses are within the reserved space

- Heap variables are prohibited in Copker
  - This issue only relates to the long integers
    - Modify PolarSSL v1.2.5 to eliminate heap variables
  - A little more memory
    - Cache is large enough

Memory Access by Share-cache Cores

- Caches are shared by several cores
- If another concurrent task take very frequent memory accesses – **cache replacement**
  - The data of Copker may be evicted to RAM

- Cache-fill mode in Intel CPU: **No-Fill Mode**
  - Data in caches are still effective; but read/write misses accesses the RAM directly
  - No cache-filling or replacement
- Copker tasks let other cores enter no-fill mode
- The number of concurrent Copker tasks == the number of separate cache sets

Cache Control by Share-cache Cores

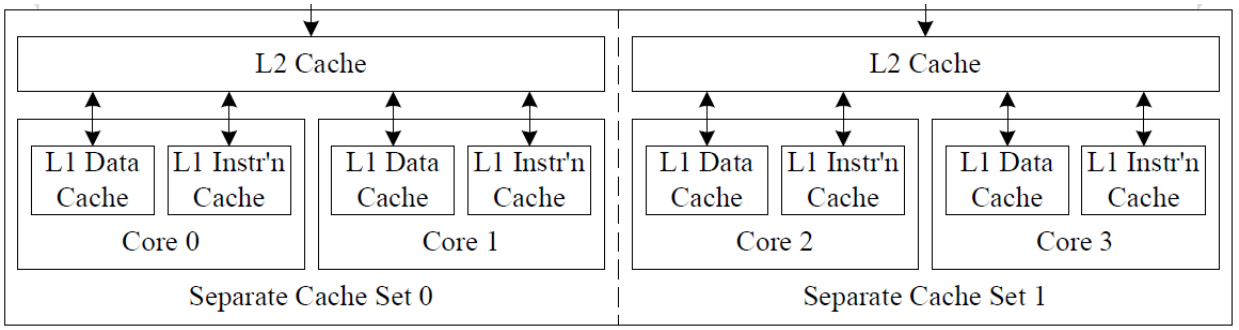
- Malicious binaries in other cores may control caches
- wbinvd instruction, needs ring 0 privilege
  - Write back all modified cache lines to RAM and invalidate the caches.

- Patch Linux, 2 system calls
  - Only write_cr0() and wbinvd(), can execute these cache-related instructions
- When Copker tasks are running
  - Other core can not leave from no-fill mode
  - Other core can not execute wbinvd♪

Trustworthy OS Kernel

- Copker needs a trustworthy OS kernel
  - Task scheduling/interrupts are disabled effectively
  - Patches are effective
  - Task isolation without vulnerabilities
    - No malicious process accesses the reserved space

Prototype

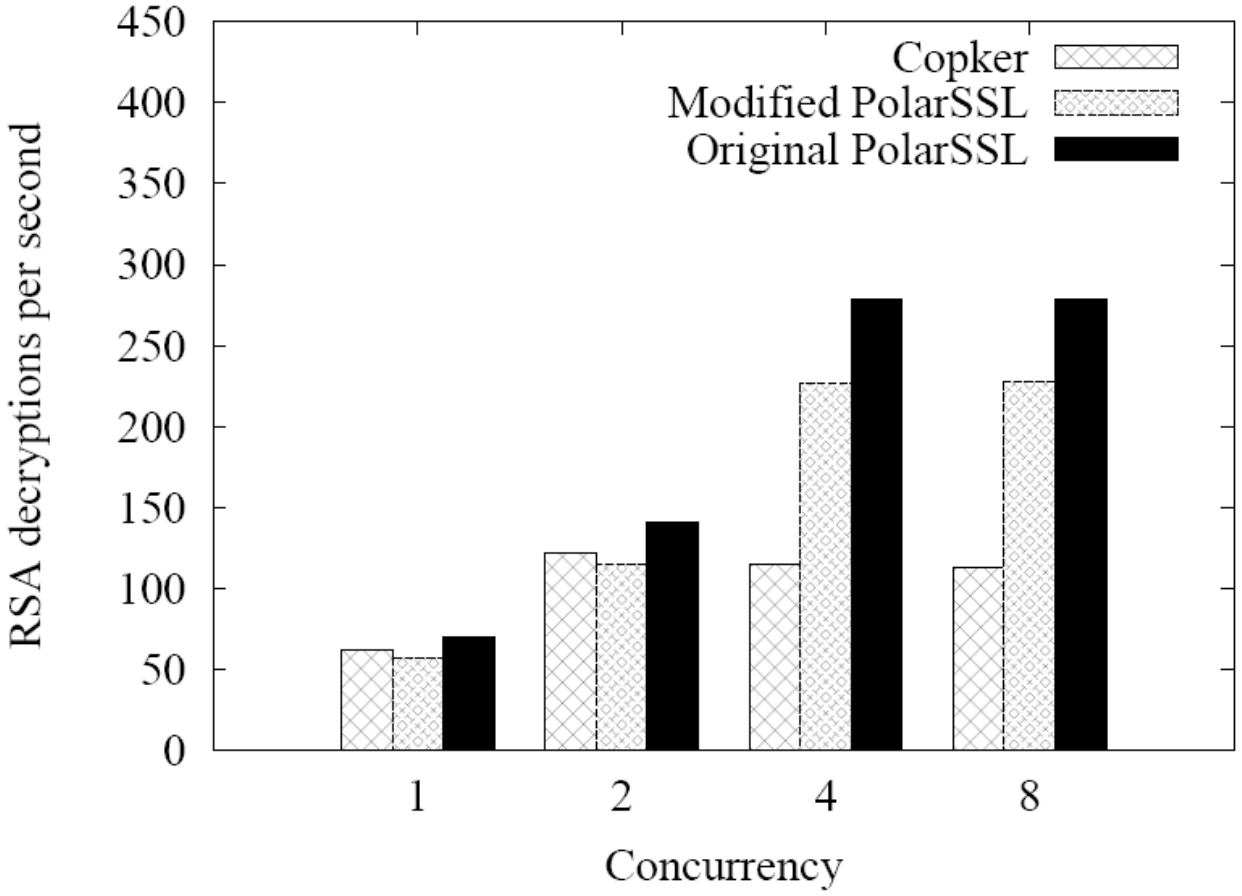


- Intel Core2 Quad Q8200
  - 4 cores, 2 separate cache sets
- Linux kernel 3.9.2
- RSA - A modified version of open-source PolarSSL
  - No heap data
  - Sliding windows changed from 6 to 1
    - Less cache requirement
- 2048/3072/4096-bit RSA

Performance [1]

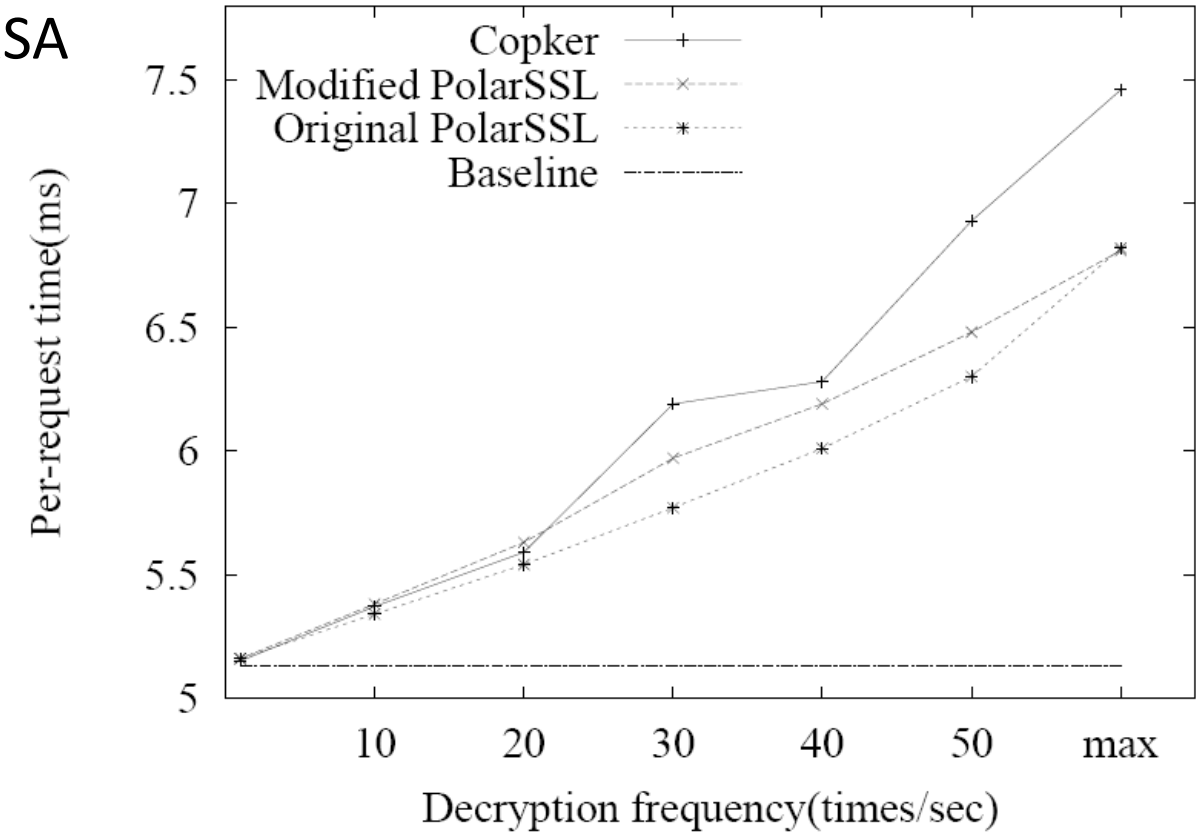
- Intel Q8200, 2 separate cache sets / 4 cores
- Up-to-2 concurrent Copker tasks
- 2048-bit RSA

- About 100%
  - Rare requests
- About 50%
  - Frequent requests



Performance [2]

- Impact on concurrent applications
  - Forced to be in no-fill mode, when Copker computing
- SysBench: the task to find prime numbers
  - x: the frequency of RSA
  - y: time per task
  - About 10:9



Conclusion

- The first cache-based solution against cold-boot attacks
  - Simple and effective
- Cache-based solution
  - 4096-bit RSA, CRT-enabled
  - The algorithms can be implemented by high-level programming languages, such as C

Other Issues [1/2]

- Cache-based side channel attack
  - Exist in some traditional RSA implementations
  - No such side channel in Copker
    - Side benefit
    - All computations are in caches only

- APCI S3 (suspend-to-RAM) and S4 (suspend-to-disk)
  - Do not mater
  - Interrupts are disabled, so these events are handled after the Copker RSA computations

Other Issues [2/2]

- Loadable kernel module (LKM)
  - Not support
  - Because the module may use `wbinvd`, for example
- kdump/kexec
  - When OS crashes, the kernel is dumped to the disk automatically
  - Quickly boot to a dump-capture kernel
  - Not support
  - The dump file may contain sensitive keys

Thanks!

Any questions or comments?

Jingqiang Lin <LINJQ@is.ac.cn>