Cache Attack on AES for Android Smartphone\*

Bo Li

School of Computer Science and Engineering   
Beihang Univeristy  
Beijing, China  
(+86)82338059

1st author's E-mail address

Bo Jiang†

School of Computer Science and Engineering   
Beihang Univeristy  
Beijing, China  
(+86)82338059

jiangbo@buaa.edu.cn

**ABSTRACT**

In this paper, we describe the formatting guidelines for ACM SIG Proceedings.

**CCS Concepts**

• **Information systems➝Database management system engines**   • **Computing methodologies➝Massively parallel and high-performance simulations.**This is just an example, please use the correct category and subject descriptors for your submission*.* The ACM Computing Classification Scheme:

<http://www.acm.org/about/class/class/2012>. Please read the [HOW TO CLASSIFY WORKS USING ACM'S COMPUTING CLASSIFICATION SYSTEM](http://www.acm.org/publications/article-templates/CCS-HOWTO-v6-12Jan2015.docx) for instructions on how to classify your document using the 2012 ACM Computing Classification System and insert the index terms into your Microsoft Word source file.

**Keywords**

Keywords are your own designated keywords separated by semicolons (“;”).

# INTRODUCTION

随着移动互联网近些年突飞猛进的发展，手机等移动设备已经成为我们生活必不可少的一部分。它在给我们生活带来便利的同时，也给我们到来了潜在的威胁，尤其是在手机能方便管理资产，存储着各种隐私的今天。因此，随着科技的进步，安全问题也越发得到人们的重视。虽然手机等设备的开发厂商以及Android系统的开发公司采取了各种措施来保护用户的隐私安全，包括可信执行环境，虚拟内存管理，权限管理等。然而，由于电脑，手机等设备结构设计的缺陷，并随着安全领域不断的研究，一些手机设备的安全漏洞和攻击方式逐渐的显露出来，其中包括cache旁路攻击。Cache旁路攻击是通过探测程序执行时cache的hit或miss，访存时间的多少等信息来进行攻击的方法。其中攻击程序与被攻击程序没有直接的交互，他们并行执行在同一个核或不同的核上，访问各自的地址空间，因此攻击时不需要多余的权限。

在最近10年中，基于x86平台的CPU cache攻击得到了人们越来越多的重视。Kocher[19]和Kelsey[18]等人提出了通过分析高速缓冲存储器在运行时泄露出的旁路信息来破解计算机中的加密算法的方法。这一思想在计算机安全相关人员的重视下得到快速的发展。在最近几年不断提出了在x86平台上的cache攻击方法，并在监测用户键盘输入，AES T-table加密破解方面得到了实验，也证明了cache攻击的有效性。比如，Page等人提出了一种针对DES加密技术的cache模型攻击的方法，将DES密钥的搜索空间从56位降低到32位。Tsunoo[34]通过分析DES查表索引以及Cache的访问特征首次实现了针对DES的Cache 攻击，并在一台600-MHz Pentium III 的个人计算机上，通过223个明文样本成功获取DES 全部密钥。Bernstein[38]在强制消除网络传输时延条件下实现了一种针对OpenSSL[27]中AES 的远程时序驱动计时攻击，其中加密服务端采集AES 加密时间。Bonneau[5]提出了一种利用密码程序在加密中的内部数据访问冲突导致的Cache 命中和失效信息来进行密码分析的方法。Percival[30]提出多线程间共享Cache 存储器访问引发的漏洞，它不仅提供了线程间一个简单、高带宽的隐秘通道，也给恶意线程监视其他线程提供了入口，使得恶意线程能够窃取加密密钥，他依次设计实现了一种针对RSA 的计时攻击。Osvik[29]借鉴Percival信息采集方法实现了多例针对AES 的Cache 计时攻击。Neve[24][25][26]将Osvik攻击[33]切入点转移到最后一轮，提出了一种新的AES 最后一轮访问驱动的Cache计时攻击方法。可见，通过这些漏洞，攻击者不仅可以获得用户键盘输入等的隐秘信息，还能获得价值更高的密钥等信息，进而威胁到用户的财产安全。

然而，由于Android等移动设备的体系结构与Intel x86结构有很大的区别，在指令集合cache组织方式以及cache替换策略等与cache攻击相关结构也有很大的不同。因此，直到最近为止，才有人提出在非root的手机上的有效的跨核 cache攻击方法。Moritz Lipp et al. [21] 提出了通过prime + probe， flush + reload， evict + reload以及flush + flush对ARM处理器的跨核攻击模型，并且不需要root权限。这些模型能够有效的探测到在被攻击程序运行时cache无意间泄露出来的信息，通过对这些信息进行统计分析，并将其作用于cache攻击的模型即可提取用户的私密信息。然而当前并没有在android端完整实现AES攻击的例子。由于基于Android系统的移动设备和x86设备的cache结构不同，在将x86平台的攻击方式应用到移动设备时会出现一些问题。比如cache的替换策略，在x86平台上，因为cache使用的是LRU替换策略，可以很简单的将cache中指定set中的数据驱逐到内存中。然而由于android使用的是伪随机替换策略，在将cache中指定set中的数据驱逐到内存时需要使用额外的方法。此外，为了得到稳定的数据访问时间，通常需要对访存或访cache操作进行预热。而之前的攻击方式通常直接使用第一次访问cache或访问内存的时间来度量cache命中与否，从而容易引进误差。

# Background

## Determining the Best Eviction Strategy

在x86平台下可以使用cflush指令将cache中的内容驱逐到内存中，虽然arm平台也提供了类似的操作指令，但通常只在特权模式下才能使用，因此需要更加普适的驱逐策略来实现cache的驱逐。连续地址访问就是一种更加通用的一种驱逐策略，其多次读取能够映射到同一cache中set的数据，通过新读取的数据将之前缓存到cache中的数据驱逐到内存的方式实现对指定set的驱逐操作。虽然读取大量的地址能够大概率的保证将关联set中的数据都驱逐出内存，但大量的访存操作不仅仅会增加驱逐所花的时间，存储相关地址的内存也会增大，而且由于移动设备的cache采用伪随机替换策略，连续读取多个内存数据并不能保证cache中的内容能够彻底的驱逐到内存中。因此找到快速有效的驱逐策略对cache旁路攻击来说是至关重要的。

Gruss发现对驱逐策略来说cache set的大小、驱逐时不同地址的访问顺序及重复访问的次数这三个因素是至关重要的。据此实现了如下的驱逐算法：

|  |  |
| --- | --- |
| 驱逐算法 | |
| 输入：  N：待驱逐set中可以存放的不同地址数  D：每个循环访问的不同地址数  A：每个循环每个地址的访问次数 | |
| 输出：无 | |
| 1:  2:  3:  4:  5:  6: | for i = 0; i < N – D; i++ do  for j = 0; j < A; j++ do  for k = 0; k < D; k++ do  Access(i + k)  end for  end for  end for |

因此，为了保证驱逐的成功率，需要对具体的设备做大量的实验以得到快速有效的驱逐模式。本文中使用脚本的方式对可能的策略组合进行检测，最终选择驱逐效果最好的策略组合作为目标机的驱逐策略。

## The Prime Probe Strategy

为了能够通过cache来获取用户的私密信息，攻击程序需要有能够获取cache状态的能力。可以使用Prime+Probe的方式来判断用户程序在运行时有没有访问到cache中的指定set中的数据。该方法主要由下面几个步骤组成。

Prime+Probe：

1. 驱逐并占用指定cache set
2. 被攻击程序运行
3. 检测之前的数据是否还在cache中

在第一阶段，攻击程序驱逐并占据了一个或多个cache中的set，通常根据驱逐策略以一定得方式连续读取多个地址中的数据来实现。在第二阶段，被攻击程序正常执行，执行过程中的访存操作也许会占用cache中的一些区域。第三阶段攻击程序检测还有多少数据缓存在cache中，如果第一阶段占用的数据仍然保存在cache中则表明被攻击程序在运行过程中没有访问到制定的cache set，反之，如果第一阶段中的部分数据被驱逐到内存中，则说明被攻击程序执行时访问的数据占用了制定cache set中的部分line。

## Precise Measurement of Time

一个精确的计时方式是cache攻击的前提，它需要将cache hits和cache misses区分开来，供攻击者获取待攻击程序运行导致的cache状态的变化。并针对不同的攻击对象获取不同的相关信息，比如对共享库的攻击需要获取共享库相关地址在被攻击程序运行期间的cache缓存状况。然而获取这些状况的前提都是拥有能够准确区分cache hits和cache misses的能力。Moritz Lipp et al.虽然提出了几种非特权的计时方式，包括perf\_event\_open、POSIX的clock\_gettime函数以及专用的线程计时器。但这些接口并非对所有Android版本，对所有处理器都开放，因此需要对待攻击机型确定能精确且能稳定测量的时间源，以此提供对cache攻击的支持。

除了通过读取寄存器获取CPU周期来度量时间外，还有其他3种可选方式来度量访存或访问cache的时间。

* clock\_gettime系统调用

"clock\_gettime"是基于Linux C语言的时间函数，他可以精确到纳秒。

#include<time.h>

int clock\_gettime(clockid\_t clk\_id， struct timespec \*tp);

clk\_id : 检索和设置的clk\_id指定的时钟时间。

CLOCK\_REALTIME:系统实时时间，随系统实时时间改变而改变，即从UTC1970-1-1 0:0:0开始计时，中间时刻如果系统时间被用户改成其他，则对应的时间相应改变

CLOCK\_MONOTONIC:从系统启动这一刻起开始计时，不受系统时间被用户改变的影响

CLOCK\_PROCESS\_CPUTIME\_ID:本进程到当前代码系统CPU花费的时间

CLOCK\_THREAD\_CPUTIME\_ID:本线程到当前代码系统CPU花费的时间

struct timespec

{

time\_t tv\_sec; /\* 秒\*/

long tv\_nsec; /\* 纳秒\*/

};

* Perf性能分析工具

Perf是内置于Linux内核源码树中的性能剖析(profiling)工具。它基于事件采样原理，以性能事件为基础，支持针对处理器相关性能指标与操作系统相关性能指标的性能剖析。常用于性能瓶颈的查找与热点代码的定位。

从Linux Kernel2.6.31版本开始，Linux内核开始提供\_\_NR\_perf\_event\_open的系统调用。使用这个系统调用我们可以像使用文件一样打开一个Performance counter，通过设置不同的参数让这个Performance Counter统计不同的软件或硬件事件，然后就可以向读文件一样来读取这些事件的统计结果。

static struct perf\_event\_attr attr;

attr.type = PERF\_TYPE\_HARDWARE;

attr.config = PERF\_COUNT\_HW\_CPU\_CYCLES;

fd = syscall(\_\_NR\_perf\_event\_open， &attr， 0， -1， -1， 0);

PERF\_COUNT\_HW\_CPU\_CYCLES：统计CPU周期数

* 线程计时模拟器

如果没有足够精确的计时接口可用，攻击者可以通过运行一个循环自增一个全局变量的线程来得到CPU周期的一个倍数估计值。因为一次循环自增操作可看做是由固定次数个时间周期组成的，所测时间的整数倍即为所对应的时钟周期，因此能够将cache hits和cache misses区分开来，从而也可以用于cache攻击。

# Attacking AES Algorithms

## The First Round Attack

\* This research is supported in part by NSFC (project no. 61772056), the Research Fund of the MIIT of China (project no. MJ-Y-2012-07), and the research fund of the State Key Laboratory of Virtual Reality Technology and Systems.

† Correspondence author

本文使用prime+probe的方式对Aes进行攻击，攻击主要分为第一轮和第二轮攻击，第一轮攻击主要根据可以通过密钥和明文获取第一轮t-table的访问索引，及给一个16字节的密钥k = (k0, . . . , k15)，在Aes加密过程中会扩展到10轮的内部密钥，其中r=1,…,10，每一轮密钥被划分为4个字，每个字4个字节，表示为，其中第一轮扩展密钥时访问的T-table索引可由p^k求得，及明文的每个字节与密钥的每个字节相异或得到的就是第一轮攻击时访问到的table索引，其中第0、4、8、12字节求得的索引访问的是table0，第1、5、9、13访问的是table1，2、6、10、14访问的是table2，3、7、11、15访问的是table3。

本文通过使用ks检验的方式来获取度量分数，通过ks检验来判断两个分布是否相同。其中第一个分布的样本获取的是prime之后紧接着probe所测得时间，其中第二个样本获取的是prime之后执行Aes加密操作后对table所映射到的set进行的probe操作所获取的时间。此时如果ks检验得到这两个样本总体属于同一个分布，则表示在执行Aes期间Aes并没有访问到这个set。KS-检验（Kolmogorov-Smirnov test） -- 检验数据是否符合某种分布Kolmogorov-Smirnov是比较一个频率分布f(x)与理论分布g(x)或者两个观测值分布的检验方法。其原假设H0:两个数据分布一致或者数据符合理论分布。D=max| f(x)- g(x)|，当实际观测值D>D(n,α)则拒绝H0，否则则接受H0假设。KS检验与t-检验之类的其他方法不同是KS检验不需要知道数据的分布情况，可以算是一种非参数检验方法。当然这样方便的代价就是当检验的数据分布符合特定的分布事，KS检验的灵敏度没有相应的检验来的高。在样本量比较小的时候，KS检验最为非参数检验在分析两组数据之间是否不同时相当常用。

## The Second Round Attack

第二轮加密访问到的索引比较复杂，通过数学推到，可得到第二轮访问索引如下：

⊕⊕⊕⊕2•⊕⊕3•⊕⊕⊕

⊕⊕⊕⊕3•⊕⊕⊕⊕⊕⊕

⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕1

⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕⊕

其中x2表示中间密钥的第三个字节，x5,、x8、x15同理，其中x2对应的是table2的索引，x5对应的是table1的索引，x8对应的是table0的索引，x15对应的是table3的索引，这些索引是实现Aes攻击的关键。本文通过假设检验的方式来进行Aes攻击，及遍历所有密钥可能的取值。根据假设的取值和密钥可以计算出相应访问的索引，再和实际访问索引进行对比则可获得该假设的可信度。

也就是首先枚举密钥可能的取值，也就是对于密钥中的每一个字节，从0到255有256种可能，且16个字节得同时进行猜测，及控制变量法。对于每一次猜测，可以获得一个度量分数m，根据m的高低则可判断哪个猜测组合的可能性最高，即可猜测出Aes的密钥。其中度量分数主要是通过获取与该猜测密钥k在Aes t-table上对应关系决定的。密钥k与明文p相异或所得的值就是Aes在执行期间在t-table的查找索引，而table的查找索引对应到cache中的不同set中，最后反映出的就是对cache中set进行prime和probe操作统计时间所反映出来的时间差异。

## The K-S statistical Test

In statistics, the Kolmogorov–Smirnov test (K–S test or KS test) is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K–S test), or to compare two samples (two-sample K–S test). It is named after Andrey Kolmogorov and Nikolai Smirnov. The two-Sample K-S test is one of the most useful and general nonparametric methods for comparing two samples. 本文使用k-s检验来判断被攻击程序运行时是否访问到了cache中的指定set。首先通过攻击程序对指定cache set进行驱逐占领操作，及通过prime操作读取指定数据到cache中，并将之前缓存的数据驱逐到内存中去，然后紧接着进行probe操作，探测prime阶段访问的数据是否还缓存在cache中，因为是prime之后紧接着的probe操作，及可以假设prime阶段缓存的数据仍然缓存在cache中

## Performance Optimization

Please leave 3.81 cm (1.5") of blank text box at the bottom of the left column of the first page for the copyright notice.

Table 1. Table captions should be placed above the table

|  |  |  |  |
| --- | --- | --- | --- |
| **Graphics** | **Top** | **In-between** | **Bottom** |
| Tables | End | Last | First |
| Figures | Good | Similar | Very well |

# Experimental Study

本文对Aes进行了第一轮攻击，并能够成功获取到密钥中每一位的前4个位，结果如下所示：

图 11 Aes第一轮攻击结果

在实验中使用到的密钥k=(0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07)，如上图11所示，能够比较明显的获取到密钥的前4位。

# Related Work

逐个描写相关工作。

Place Tables/Figures/Images in text as close to the reference as possible (see Figure 1). It may extend across both columns to a maximum width of 17.78 cm (7”).

Captions should be Times New Roman 9-point bold. They should be numbered (e.g., “Table 1” or “Figure 2”), please note that the word for Table and Figure are spelled out. Figure’s captions should be centered beneath the image or picture, and Table captions should be centered above the table body.

# Conclusion

结论。

The heading of a section should be in Times New Roman 12-point bold in all-capitals flush left with an additional 6-points of white space above the section head. Sections and subsequent sub- sections should be numbered and flush left. For a section head and a subsection head together (such as Section 3 and subsection 3.1), use no additional space above the subsection head.



Figure 1. Insert caption to place caption below figure.

.

# ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

# REFERENCES

1. Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= <http://doi.acm.org/10.1145/161468.16147>.
2. Ding, W. and Marchionini, G. 1997. *A Study on Video Browsing Strategies*. Technical Report. University of Maryland at College Park.
3. Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 01 - 06, 2000). CHI '00. ACM, New York, NY, 526-531. DOI= <http://doi.acm.org/10.1145/332040.332491>.
4. Tavel, P. 2007. *Modeling and Simulation Design*. AK Peters Ltd., Natick, MA.
5. Sannella, M. J. 1994. *Constraint Satisfaction and Debugging for Interactive User Interfaces*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
6. Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* 3 (Mar. 2003), 1289-1305.
7. Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM, New York, NY, 1-10. DOI= <http://doi.acm.org/10.1145/964696.964697>.
8. Yu, Y. T. and Lau, M. F. 2006. A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions. *J. Syst. Softw.* 79, 5 (May. 2006), 577-590. DOI= <http://dx.doi.org/10.1016/j.jss.2005.05.030>.
9. Spector, A. Z. 1989. Achieving application requirements. In *Distributed Systems*, S. Mullender, Ed. ACM Press Frontier Series. ACM, New York, NY, 19-33. DOI= <http://doi.acm.org/10.1145/90417.90738>.

Authors’ background

\*This form helps us to understand your paper better, the form itself will not be published. Please fill in every author’s information.

\*Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Your Name | Position\* | Email | Research Field | Personal website |
| Bo Li |  |  |  |  |
| Bo Jiang | Associate Professor | gongbell@gmail.com | Mobile testing and security | http://jiangbo.buaa.edu.cn |