DOI:10.13203/j.whugis2008.10.003
第33卷 第10期 武汉大学学报·信息科学版 Vol.33 No.10
2008年10月 Geomatics and Information Science of Wuhan University Oct. 2008

文章编号:1671-8860(2008)10-1088-04 文献标志码:A

基于Cache时间特性的AES差分时间分析攻击

邓高明[1] 张 鹏[1] 赵 强[1] 刘晓芹[1]

(1 军械工程学院计算机工程系,石家庄市和平西路97号,050003)

摘 要:通过分析数组访问操作的索引值与Cache命中的关系,找到了数组索引值与访问时间的弱线性关系,并用数组访问实验进行了验证。在一般对称加密实现过程中数组访问索引值与明/密文以及最密钥之间的关系的基础上,提出了一种基于Cache时间特性的差分时间分析旁路攻击方法,通过实验验证了在4×10^4组随机样本条件下,将AES-128最后一轮子密钥16个字节的密钥搜索空间从2^{128}缩小到2^{98}。

关键词:旁路攻击;Cache时间特性;差分分析;AES

中图法分类号:TN918

基于Cache时间特性的密码旁路攻击[1]是密码旁路分析中新兴的一种方法,它利用Cache在PC机中的快速缓存作用,结合大多对称密码实现中的查表操作,通过精确测量加密程序在PC机上的运行时间,并对运行时间加以分析得到加密程序中的秘密信息。利用差分分析的统计方法能有效地消除噪声,突出与秘密信息相关的时间特性,进而从中萃取出秘密信息,对密码安全构成了巨大的威胁。本文提出一种针对AES密码的基于Cache时间特性的差分分析方法,并针对Linux系统下OpenSSL库中的AES快速实现进行了攻击。

1 数组操作的Cache时间特性

Cache的容量比主存储器要小得多,而且读写Cache和读写主存储器的时间有很大差别,所以CPU访问Cache时的命中(Cache Hit)和失效(Cache Miss)情况就会从时间延迟这个旁路泄漏出信息。现代一般CPU访问Cache时命中的情况下仅需要0.3 ns左右,而失效的情况下则需要从主存中获取数据,需要将近50 ns[2],这100多倍的时间差在多次操作时累积起来是非常明显的差距。

Cache中存储的是内存中最近被访问的数据的映像,当主存中的数据被访问的时候,这些数据就会将Cache中的旧数据替换出去。紧接其后的相同位置的主存访问就只需要从Cache中获取数据。CPU与Cache之间的数据交换以字为单位,而Cache与主存之间的数据交换以行为单位。为了说明Cache在进行数组操作时的时间效应,在表1所示配置的环境下进行了如图1所示的实验。

表1 实验设备参数

Tab.1 Configuration of the Device

CPU		Intel Pentium 4	1.70 Hz
L1 Cache	容量	8 kB	
	集联	4通道	
	行大小	64 byte	
L2 Cache	容量	512 kB	
	集联	8通道	
	行大小	64 byte	

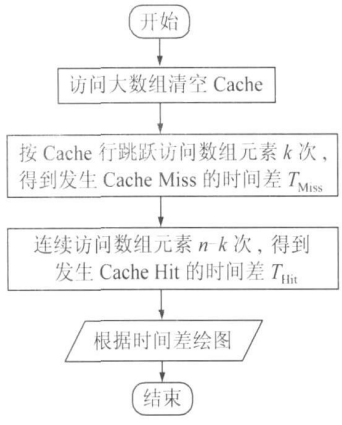

图1 Cache时间特性分析实验流程图

Fig.1 Flow Chart of Cache Timing Experiment

收稿日期:2008-08-28。

项目来源:国家自然科学基金资助项目(60571037);国家863计划资助项目(2007AA01Z454)。

观测 Cache 命中和失效情况下访问数据的时间延迟,得到的规律如图 2 所示。从图中可以看到,随着 Cache Miss 次数 M 从 1 到 16 的逐渐增多,16 次数组访问操作的运行时间 T 也逐渐增大,而且数组操作的运行时间和 Cache Miss 次数基本成正比例关系,$T=kM$。这里的 k 和配置环境有关,主存储器时间延迟越大,k 的值也就越大。至此,可以明确地得到结论,可以从数组操作运行时间推断操作中的 Cache Miss 的次数。

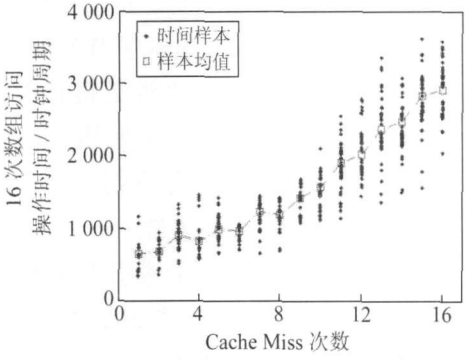

图 2　Cache 命中/失效比与程序运行时间的关系

Fig. 2　Relationship Between Cache Misses and Time

2　差分时间分析原理

差分时间分析是用统计的方法对多次不同明文的加密程序运行时间进行分析,利用大样本来消除不需要的噪声信号,提取有用的信号,具有较强的攻击能力。

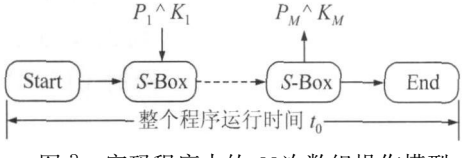

图 3　密码程序中的 M 次数组操作模型

Fig. 3　M Array Accesses in Cipher Model

首先给出如图 3 所示的密码程序数组操作模型。假设有对同一个 S-Box 的 M 次操作,使用的索引值分别为 $\text{index}[j]=P_j \hat{} K_j$,其中 $1 \leqslant j \leqslant M$。差分时间攻击的步骤如下。

1) 获得 N 个随机的明文输入 PT_l 和对第 l 次加密过程进行记录得到的时间 T_l,其中 $1 \leqslant l \leqslant N$;

2) 穷举猜测子密钥块 K_i 和 K_j 的值,根据每次猜测的值和所有明文推算计算:

$$D=f(PT,\ Kx)=(PT_i \hat{} Kx_i - PT_j \hat{} Kx_j)/\text{Cache\_ Line\_ Size} \quad (1)$$

并根据 D 的取值,把 $D=0$ 对应的明文和加密这个明文的时间划分到集合 S_0 中,把 $D \neq 0$ 对应的明文和加密这个明文的时间划分到集合 S_1 中;

3) 分别求集合 S_0 和 S_1 中时间的平均值 A_0 和 A_1:

$$A_0=\frac{1}{|S_0|}\sum_{T_i \in S_0} T_i,\ A_1=\frac{1}{|S_1|}\sum_{T_i \in S_1} T_i \quad (2)$$

式中,$|S_0|$ 和 $|S_1|$ 分别表示 S_0 和 S_1 的元素个数。然后对两个平均值作差,得到差分时间值 D:

$$D=|A_1-A_0| \quad (3)$$

4) 如果 D 的值接近于 0,则可以推断子密钥块 K_i 和 K_j 的猜测是错误的;如果 D 的值远大于 0,超过了 Cache 访问的时间延迟 ΔT,则可以推断子密钥块 K_i 和 K_j 的猜测是正确的。

因为如果对子密钥块 K_i 和 K_j 的猜测是正确的,那么集合 S_0 中的明文就都满足 $(PT_i \hat{} Kx_i - PT_j \hat{} Kx_j)/\text{Cache\_ Line\_ Size}=0$,也就是 Cache 操作都是满足 Cache Hit,而集合 S_1 中的明文都满足 $(PT_i \hat{} Kx_i - PT_j \hat{} Kx_j)/\text{Cache\_ Line\_ Size} \neq 0$,即 Cache 操作都是满足 Cache Miss,对应的时间平均值 A_1 和 A_0 之差就是 Cache 访问的时间延迟和主存访问的时间延迟的差,这个值就应该远大于 0;否则如果对子密钥块 K_i 和 K_j 的猜测是错误的,对集合 S_0 和 S_1 的划分就是随机的,在大样本的情况下,两个随机集合 S_0 和 S_1 中的时间平均值 A_0 和 A_1 就会非常接近,其差值会非常接近于 0。

3　针对 AES 的攻击

3.1　AES 密码算法快速实现

AES(advanced encryption standard)[3] 密码算法支持 128 位、192 位和 256 位密钥的不同版本,本文只关注 128 位密钥的 AES、192 位和 256 位密钥版本,只是采用了不同的密钥扩展算法和更多的加密轮数。

AES 是一个基于有限域运算的迭代密码。第 i 轮迭代根据 16 字节的输入状态 S_i 和一个 16 字节的子密钥 K_i,产生一个 16 字节的输出状态 S_{i+1}。除最后一轮之外,每一轮迭代运算都包括对 S_i 的以下 4 个代数运算:字节代换(SubBytes),行移位(ShiftRows),列混淆(MixColumns),然后和轮子密钥 K_i 作异或运算(AddRoundKey)。

为了提高性能,现在常用的 AES 软件实现将前三个操作合并并且预先进行计算,结果存储在几个大的查找表(T_0, T_1, T_2, T_3)中,每一个查找

表将1字节的输入映射到4字节的输出。每一轮的运算都是首先将S_i分割成16字节的小块S_i^0，S_i^1，…，S_i^{15}，K_i分割成16字节的小块K_i^0，K_i^1，…，K_i^{15}，之后进行如下的轮加密过程：

$$\begin{cases}(S_{i+1}^0,S_{i+1}^1,S_{i+1}^2,S_{i+1}^3)=T_0[S_i^0]\oplus T_1[S_i^5]\oplus T_2[S_i^{10}]\oplus T_3[S_i^{15}]\oplus(K_i^0,K_i^1,K_i^2,K_i^3)\\(S_{i+1}^4,S_{i+1}^5,S_{i+1}^6,S_{i+1}^7)=T_0[S_i^4]\oplus T_1[S_i^9]\oplus T_2[S_i^{14}]\oplus T_3[S_i^3]\oplus(K_i^4,K_i^5,K_i^6,K_i^7)\\(S_{i+1}^8,S_{i+1}^9,S_{i+1}^{10},S_{i+1}^{11})=T_0[S_i^8]\oplus T_1[S_i^{13}]\oplus T_2[S_i^2]\oplus T_3[S_i^7]\oplus(K_i^8,K_i^9,K_i^{10},K_i^{11})\\(S_{i+1}^{12},S_{i+1}^{13},S_{i+1}^{14},S_{i+1}^{15})=T_0[S_i^{12}]\oplus T_1[S_i^1]\oplus T_2[S_i^6]\oplus T_3[S_i^{11}]\oplus(K_i^{12},K_i^{13},K_i^{14},K_i^{15})\end{cases}\tag{4}$$

轮计算的软件实现采用这种方法效率比较高，因为只进行了16次查表操作和16次4字节的异或操作。在初始密钥被扩展为10个轮密钥之后，一个完整的加密过程首先进行一次16字节明文P和16字节初始密钥K的异或操作(AddRoundKey)，紧接着9次普通的轮加密操作，再加上一次更为简单的最后一轮。最后一轮没有列混淆(MixColumns)操作，这一步是关键的，因为它使得软件实现在最后一轮采用了一个新的T_4表。

128位AES总共采用了10轮加密，但是需要11个16字节的子密钥，因为在轮加密之前有一次16字节明文P和16字节初始密钥K的异或操作。这176字节的密钥都是从16字节的原始密钥产生出来的，方法是反复地从前一个16字节块通过一个非线性变换得到后一个16字节块，直到得到所有的176字节密钥。这个密钥扩展结构在给出扩展密钥的16个连续字节的情况下是可逆的[2]。这一点对攻击者来说是很有用的，因为只要恢复了最后16字节扩展密钥(或者其他16字节)，就相当于恢复了原始密钥。

$$\begin{cases}(C^0,C^1,C^2,C^3)=(T_4[S_{10}^0]\oplus K_{10}^0,T_4[S_{10}^5]\oplus K_{10}^1,T_4[S_{10}^{10}]\oplus K_{10}^2,T_4[S_{10}^{15}]\oplus K_{10}^3)\\(C^4,C^5,C^6,C^7)=(T_4[S_{10}^4]\oplus K_{10}^4,T_4[S_{10}^9]\oplus K_{10}^5,T_4[S_{10}^{14}]\oplus K_{10}^6,T_4[S_{10}^3]\oplus K_{10}^7)\\(C^8,C^9,C^{10},C^{11})=(T_4[S_{10}^8]\oplus K_{10}^8,T_4[S_{10}^{13}]\oplus K_{10}^9,T_4[S_{10}^2]\oplus K_{10}^{10},T_4[S_{10}^7]\oplus K_{10}^{11})\\(C^{12},C^{13},C^{14},C^{15})=(T_4[S_{10}^{12}]\oplus K_{10}^{12},T_4[S_{10}^1]\oplus K_{10}^{13},T_4[S_{10}^6]\oplus K_{10}^{14},T_4[S_{10}^{11}]\oplus K_{10}^{15})\end{cases}\tag{5}$$

式中，C是16字节的密文输出；T_4是AES最后一轮的查找表。对密文的任意两个字节C^i和C^j，设存在一个x使得$C^i=T_4[S_{10}^x]\oplus K_{10}^i$，存在一个$y$使得$C^j=T_4[S_{10}^y]\oplus K_{10}^j$。不考虑$x$和$y$的值，只要当$S_{10}^x=S_{10}^y$，在查找$T_4$时就会发生一次Cache命中。假设$S_{10}^x=S_{10}^y$，则$T_4[S_{10}^x]=T_4[S_{10}^y]=\alpha$，那么$C^i=\alpha\oplus K_{10}^i$，$C^j=\alpha\oplus K_{10}^j$，也就是$C^i\oplus C^j=K_{10}^i\oplus K_{10}^j$。反之，若是$C^i\oplus C^j\neq K_{10}^i\oplus K_{10}^j$，则在查找$T_4$表的时候就会得到两个不同的值$\alpha$、$\beta$。那么就会有$\gamma=\alpha\oplus\beta=C^i\oplus C^j\oplus K_{10}^i\oplus K_{10}^j$，因为$K_{10}^i\oplus K_{10}^j$是由密钥决定的，对于固定的明文字节差值$C^i\oplus C^j$来说，$\gamma$的值就是一个常数。

攻击的方法就是记录每次随机明文加密得到的密文C和对应的加密时间t。在获得足够的样本之后考察每个密文中所有的C^i和C^j，猜测对应的密钥字节K_{xi}和K_{xj}，根据式(1)计算D值并划分集合，对两个集合中的时间值求平均之后差分，根据差分值判断猜测密钥块的正确性。

3.2 针对AES最后一轮的攻击

为了利用访问Cache时的索引不同而导致的时间差异对AES进行攻击，考虑加密的最后一轮。正如前面所说，AES加密的最后一轮没有列混淆操作而引入了另一个表T_4，将式(4)转化为：

3.3 实验及分析

根据前面介绍的攻击原理，编写针对AES-128的Cache Hit攻击算法，其中利用了根据CPU时间戳进行精确计时的指令RDTSC[4]对AES的加密过程进行精确计时，为了保证在每一次AES加密之前所有的查找表都不在Cache中，需要在每次AES加密之前进行一次Cache“清空”操作，可以首先用CPUID指令[4]查找得到Cache大小S，然后在每次AES加密之前访问内存中大小为S的一块连续区域来实现确保将AES表元素都替换出Cache，同时CPUID指令还能获得Cache的行大小。

用攻击程序对OpenSSL v.0.9.8(a)库中的AES加密程序进行攻击，使用RedHat v2.4.20-8和gcc v3.2.3，在如表1所示的环境下进行了实验。实验表明，在Cache行大小为通常的64字节的情况下，获取4×10^4组随机样本之后，每一次对2个字节密钥块的差分时间分析能够获取每个字节前2位的异或关系，对AES-128的最后一轮(即第10轮)的子密钥的全部16个字节分析之后，可以获取后15个字节中任1个字节前2位与第1个字节前2位的异或关系，因此在密钥搜索的时候猜测第1个字节的前2位之后就可以推测后15个字节中每个字节的前2位，共确定$2\times15=30$位。可以看出，这种分析方法能够将这16个字节的密钥搜索空间从2^{128}缩小到2^{98}。

在猜测获得16字节的第10轮扩展密钥之

后,对每一个猜测的第 10 轮子密钥的 16 字节组,攻击程序反推原始密钥并且用一组已知的明文/密文来检验获得的密钥。

查表操作时 Cache 表现出的与索引值相关的"命中"和"未命中"的行为特征为密码分析提供了一条新的途径,而差分分析的统计方法可以有效地去除随机噪声并凸显出有用的数据。由于查表操作是现代 AES 加密快速而有效的实现途径,使得基于 Cache 时间特性的 AES 差分时间分析旁路攻击具有普遍的可行性。这种分析技术是以查表时 Cache 行为的时间特性为前提的,因而采用没有查表操作的 AES 实现方式是防御这种攻击的有效途径,Bernstein 提出过一种没有查表操作的 AES 加密算法的实现方式[5],另一种防御这种攻击的途径就是在加密运算之前对 Cache 进行"预热",即将表的内容预先调入 Cache 之中,这两种方法都是以牺牲加密运算的效率为代价。

参 考 文 献

[1] Kocher P. Timing Attacks on Implementations of Diffie-hellman, RSA, DSS, and Other Systems[J]. LCN '96, New York, 1996

[2] Osvik D A, Shamir A, Tromer E. Cache Attacks and Countermeasures: the Case of AES[C]. CT-RSA '06, Berlin, 2006

[3] Daemen J, Rijmen V. The Design of Rijndael: AES-the Advanced Encryption Standard[M]. New York: Springer-Verlag, 2002

[4] Intel. IA-32 Intel Architecture Software Developer's Manual[R]. Chapter 10. Intel Corporation, 2004

[5] Bernstein D J. Cache-timing Attacks on AES[OL]. http://crypto/antiforgery/ cachetiming-20050414.pdf, 2005

第一作者简介:邓高明,博士生,研究方向:信息安全。
E-mail: denggaoming26@163.com

Difference Timing Attack Against AES Based on Cache Timing Character

DENG Gaoming[1]　*ZHANG Peng*[1]　*ZHAO Qiang*[1]　*LIU Xiaoqin*[1]

(1 Dept. of Computer Engineering, Ordnance Engineering College, 97 West Heping Road, Shijiazhuang 050003, China)

Abstract: By analyzing the relationship between the indexes of array accesses and Cache hit or miss, weakly linear dependence between array indexes and its access time was found and verified with array accesses experiment. Based on analyzing the relationship between the indexes of the array accesses during general crypto implementation and the plaintext/ciphertext with the key, the Cache timing based timing difference analysis side channel attack is proposed. The experiment shows that with 4×10^4 samples, the search space of the sub-key used in the last round of the AES-128 can be reduced from 2^{128} to 2^{98}.

Key words: side channel attacks; Cache timing character; difference analysis; AES

About the first author: DENG Gaoming, Ph. D candidate, he is mainly engaged in research on information security.
E-mail: denggaoming26@163.com