第34卷 第17期 计 算 机 工 程 2008年9月
Vol.34 No.17 Computer Engineering September 2008
·安全技术· 文章编号：1000—3428(2008)17—0141—03 文献标识码：A 中图分类号：TP309

针对 AES 的基于时间的缓存攻击

李 波，胡予濮，钟名富
(西安电子科技大学计算机网络与信息安全教育部重点实验室，西安 710071)

摘 要：基于时间的缓存攻击是指通过分析处理器中算法的不同执行时间来恢复密钥的攻击。该文分析针对AES的时间驱动缓存攻击，给出一种改进的攻击，它可以应用于大多数的AES实现软件。在PentiumⅢ, OpenSSL v.0.9.8.(a)和Miracl环境下的实验发现，只需要2^{24}个时间信息就可以恢复出密钥，少于原攻击的2^{28}个时间信息数据。给出抵抗这种攻击的对策。

关键词：边信道；AES 算法；S 盒；缓存攻击；缓存未命中

Time-based Cache Attacks on AES

LI Bo, HU Yu-pu, ZHONG Ming-fu
(Key Laboratory of Computer Networks & Information Security of Ministry of Education, Xidian University, Xi'an 710071)

【Abstract】 Time-based cache attacks analyzes the time difference in the execution of algorithm over a processor, and recovers the secret key. This paper investigates such an attack on AES. A modified version of this attack is shown. It can be applied in most AES software, and implemented against OpenSSL v.0.9.8(a) and Miracl running on PentiumⅢ. The attack is shown under optimal conditions to recover a full 128-bit AES key with 2^{24} timing samples, less than 2^{28} of the previously attack. It gives several countermeasures of such attack.

【Key words】 side channel; AES algorithm; S-boxes; cache attack; cache-miss

1 概述

在研究密码基本模块时，密码学家通常把密码模块作为一个数学函数来评估其安全性。在这个模式中，Alice 和 Bob 试图使用密码在公开信道进行秘密通话，而 Eve 监听通话信道，试图窃听 Alice 和 Bob 的讲话内容。从传统角度来看，在此模式中能抵抗 Eve 的攻击的密码被认为是安全的。但在现实世界中，在各种物理设备中实现密码会受具体实现环境的影响。电力设备(比如智能卡)消耗能量，产生电磁辐射，并且与周边环境温度相互反应等，这些物理效果将会被非法第三者所监控，并利用这些信息进行密码分析。此类攻击技术称为边信道攻击(side channel attack)技术，对很多密码的安全性产生巨大冲击。时间驱动的缓存攻击是边信道攻击中的一种重要方法，它利用密码算法执行时间所反映出的密钥信息，对密码系统进行攻击。

2 理论背景

2.1 AES 算法

AES[1]是一种迭代分组密码，由Square算法改进而成。分组长度和密钥长度可以是 128 bit, 192 bit或 256 bit，加解密时按字节组织成 4 行的矩阵。

算法迭代次数由分组长度和密钥长度共同决定，可为 10, 12 和 14 圈。每一圈包括 4 个基本操作：非线性的字节替换 ByteSub(S 盒)，列间扩散的行移位 ShiftRow，列内字节间扩散的列混合 MixColumn 和圈密钥加密 RoundKeyAddition。

2.2 OpenSSL

OpenSSL[2]是一个开放源代码的网络安全项目，被广泛应用于网络安全。它实现了SSL 和TLS等一系列协议，支持 Linux, Windows, Unix 等主流操作系统，可用于身份认证，保证传输数据的完整性和保密性。

在 OpenSSL 提供的加密算法库中，它用一个字的间隔(4 B)来执行 AES 的轮操作。每一轮 r 的状态字为

$$X_i^{(r)}=\left(x_{4i}^{(r)},x_{4i+1}^{(r)},x_{4i+2}^{(r)},x_{4i+3}^{(r)}\right)\ (i=0,1,2,3)$$

这样，就产生了以下 4 个真值表：

$$X_0^{(r)}=T_0\left[x_0^{(r-1)}\right]\oplus T_1\left[x_5^{(r-1)}\right]\oplus T_2\left[x_{10}^{(r-1)}\right]\oplus T_3\left[x_{15}^{(r-1)}\right]\oplus K_0^{(r)}$$

$$X_1^{(r)}=T_0\left[x_4^{(r-1)}\right]\oplus T_1\left[x_9^{(r-1)}\right]\oplus T_2\left[x_{14}^{(r-1)}\right]\oplus T_3\left[x_3^{(r-1)}\right]\oplus K_1^{(r)}$$

$$X_2^{(r)}=T_0\left[x_8^{(r-1)}\right]\oplus T_1\left[x_{13}^{(r-1)}\right]\oplus T_2\left[x_2^{(r-1)}\right]\oplus T_3\left[x_7^{(r-1)}\right]\oplus K_2^{(r)}$$

$$X_3^{(r)}=T_0\left[x_{12}^{(r-1)}\right]\oplus T_1\left[x_1^{(r-1)}\right]\oplus T_2\left[x_6^{(r-1)}\right]\oplus T_3\left[x_{11}^{(r-1)}\right]\oplus K_3^{(r)}$$

这里的T_0,T_1,T_2, T_3是 4 个 1 Byte输入、1 个字输出的表。由于它们每一轮都由单独的查找和逐比特加运算组成，这 5 个预先计算的查找表在加密过程中增加了时间，因此这些表的最大缺点就是它所引起的内存消耗。

2.3 缓存处理器

Cache 是一种存储空间较小但存取速度却很高的存储器，它位于 CPU 和存储容量比较大、操作速度比较低的主存储器之间，也可以集成在 CPU 内部。

Cache由一个目录(也称为标记或标签)存储器和一个数据存储器组成。每当CPU需要读/写数据时，首先查找Cache的地址部分(标签字段)是否存在该地址。如果存在，则表示命中Cache(Cache-hit)，读取Cache中该地址对应的数据部分；若不存在，则称为未命中Cache (Cache-miss)，这时仍然从主内存单元中读取数据，同时将该地址和对应的数据分别写入Cache的标签字段和数据部分。下次访问该地址时，即可 命中[3]。

基金项目：国家自然科学基金资助项目(60673072)；国家密码发展中心基金资助项目
作者简介：李 波(1981－)，男，硕士研究生，主研方向：分组密码分析技术；胡予濮，博士生导师；钟名富，硕士研究生
收稿日期：2007-10-12 **E-mail：**routerlee@163.com

3 基于时间的缓存攻击

3.1 密钥恢复攻击描述

如前所述，如果明文加密时间过长，将会引起频繁的缓存未命中。在Bernstein的攻击方法[4]中，认为在加密过程中，当S盒查找表被使用时，数据存取就会发生。在这一过程中，有一些条件会引起缓存未命中。当数据第 1 次需要使用S盒查找表，由于查找时缓存中并没有这个数据在表中的定位，因此需要从主内存中读取并被装入到缓存中，这时就发生了一次缓存未命中。当还有值要查表时，如果S盒的输入值与刚才被装入的值相等或相近，这时数据的定位就可以直接从缓存中读取，缓存未命中不会发生。但是，如果这个值并不是刚才被装入的值或是相差很远时，需要的数据就不会在缓存中被找到，这时需要从主内存中查找，缓存未命中也随之发生。因此，在加密过程中，如果多个S盒查找表被使用，缓存未命中的数量将会随着输入S盒不同值数量的增长而 增长。

基于以上原因，在加密中，如果有大量不同的值进入 S 盒中，加密时间将会加长。这样，通过对一段明文加密时间的测量，可以决定明文是否会产生大量的缓存未命中。

由此可见，缓存攻击需要监视数据进出处理器不同内存层次的时刻，也就是进出缓存时。这样的攻击一般包括收集和分析 2 个阶段，前者用于给攻击者提供执行程序时的数据；后者是攻击者利用上一阶段的数据来恢复密钥。

在学习阶段，攻击者通过控制服务器或是克隆服务器的环境获得密钥k。让明文$P=\{p0,p1,\cdots,pl\text{-}1\}$为长为$l$的随机明文组。客户机C将这些明文依次提交给服务器S。服务器S将这些明文分别加密，并计算时间，如$time(E_{AES}(p_i,k))$。然后将这些时间信息返回给客户机。客户机C上的程序跟踪每个字节的时间信息的平均值，并将这些值储存在初始值为 0 的数组t[16][256]中。同时，对每一字节和明文，客户机C计算

$$t[j][p_{j,i}] := t[j][p_{j,i}] + time(E_{AES}(p_i,k))$$

由于明文是随机选择的，每一字节值的数量被储存在数组 $tnum$[16][256]中，其他一些统计数据也会被储存下来。在加密 l 次后，可以统计出每一字节的数据，这些数据显示了一个字节所取的每一个值的平均加密时间。每一个可能的值所消耗的平均处理时间(减去整体平均加密时间)如下：

$$v[j][y] = \frac{t[j][y]}{tnum[j][y]} - \frac{\sum_j \sum_x t[j][x]}{\sum_j \sum_x tnum[j][x]}$$

由此，在学习阶段建立了在随机明文和已知密钥情况下的 S 盒的数据信息。

在攻击阶段，攻击者连通服务器或服务器的克隆，但这时服务器不在其控制之下。因此，这时服务器所使用的密钥 k ' 对攻击者来说是未知的。仍然采用学习阶段的方法，客户机 C 提交一组随机明文 p'$=\{p0', p1',\cdots,pl\text{-}1'\}$给服务器 S，服务器以同样的方式处理时间信息，并将数据存储在数组 t'[16][256]和 $tnum$'[16][256]中，然后计算

$$t'[j][p'_{j,i}] := t'[j][p'_{j,i}] + time(E_{AES}(p'_i,k'))$$

$$tnum'[j][p'_{j,i}] := tnum'[j][p'_{j,i}] + 1$$

类似地，计算 v'[j][x]。

在相关性阶段，根据相关性结合学习阶段和攻击阶段中的数据，创建数组 c[16][256]，计算 v 和 v ' 的相关性

$$c[j][u] := \sum_{w=0}^{255} v[j][w] \cdot v'[j][u \oplus w]$$

数组 c 中的元素按递减顺序储存，相关性高的结果按照一个合理的偏离值保存下来。这样，只有相关性高的密钥值被作为密钥的候选值保留下来。

在暴力破解阶段，攻击者根据对密钥候选值进行暴力破解，而前 3 个阶段的工作使得暴力破解的范围缩小很多。

3.2 改进的攻击

在 Bernstein 的攻击中，需要客户机向服务器发送长度固定的随机明文，之后从服务器返回的数据中统计时间信息，这种简化的模型可以用来说明攻击的步骤和结果，但是缺乏效率。服务器只是被用来测量加密时间，在网络上不需要 2 台计算机，因此，这种 B/C 模型可以被简化；另一方面，服务器可以代替客户机，自己根据加密时间进行分析。因此，对其服务器代码进行改进，只使用一台计算机来进行攻击。

之前的服务器程序 server.c 通过监听端口，对客户机发送过来的数据进行加密，然后返回时间信息，而客户机程序 client.c 先发送随机数据给服务器，再接收服务器返回的时间数据，这样就需要服务器、克隆服务器和客户机 3 台计算机。对服务器程序 server.c 和客户机程序 client.c 中主要功能(加密和测量时间)进行改进，产生了一个新的攻击程序版本。在改进的版本中，不需要客户机向服务器发送随机数据，而是由服务器自己生成随机数据，并加密，然后统计时间信息，这样就只需要 1 台服务器，并且消除了在服务器和客户机之间的时间消耗，避免了网络延迟等方面对数据的影响，从而提高了效率。

在之前的攻击中，需要客户机先后给服务器发送长度为 800 Byte, 600 Byte和 400 Byte的数据。在改进的版本中，只产生 16 Byte的数据，通过这 16 Byte数据的加密时间信息，同样可以恢复出密钥。通过实验发现，改进的攻击只需要2^{24}个时间信息数据，少于原版本的2^{28}个。

对不同处理器及不同 AES 加密库进行攻击，得到表 1 所示的数据集。

表 1 数据集

可能值数量	密钥比特位	可能候选值
15	0	d8 dc da d9 df dd de db d2…
256	1	db 78 7b 90 c0 c6 cc 88 4a…
256	2	cc d8 d5 be d3 d1 d4 b3 c6…
256	3	d4 df db 9e bd b2 9d 53 c2…
8	4	fc fe fd ff fb f8 fa f9
256	5	5e d1 33 be a7 a5 a0 cc a6…
64	6	fa 4a ab 3a 9a 72 3b 42 b3…
18	7	28 2f 2e 2d 2b 2a 2c 29 35…
32	8	25 09 81 0b 68 3a 78 ba 08…
256	9	a8 aa ab af ad ae ac a9 f5…
256	10	54 59 41 55 e5 e6 8f ed 56…
9	11	6a 69 6c 6e 6d 68 6f 6b 0c…
229	12	8d 8c 6c 8e 6e 8f 82 8a 8d…
256	13	17 0e 70 15 b6 08 bf 25 48…
256	14	a6 a7 b4 e9 a4 a2 bf a0 ed…
8	15	90 97 95 92 93 91 94 96

通过实验发现，处理器缓存的大小以及 AES 实现方法上的不同决定了抵抗这种缓存攻击的能力。在缓存较小的处理器中，发生缓存未命中的概率更大，更容易产生加密时间上的不同。类似地，本文用 MIRACL 加密库中的 AES 实现工具，基本恢复出了密钥，见表 2。

表 2 数据集

可能值数量	密钥比特位	可能候选值
1	0	dc
1	1	00
70	2	cc d8 d5 be d3 d1 d4 b4…
1	3	49
2	4	fa fe
1	5	33
4	6	03 5f f3 6b
1	7	2e
30	8	a2 a4 a6 f8 fa f6 f1 a7…
1	9	6f
1	10	c6
2	11	6a 6e
9	12	8d 8c 6c 8e 6e 8f 82 8a…
1	13	70
20	14	c5 87 94 8d f1 e8 89 48…
1	15	93

4 抵抗缓存攻击

在研究密码安全性时，密码分析人员经常使用数学方法。传统的密码分析将注意力集中在攻击密码体系中的数学函数。能够抵抗差分分析、线性分析等攻击已经是现代分组密码设计中必需的工作。时间碰撞攻击不是针对算法本身，而是对其实现工具的一种攻击方法。

如果 AES 的 S 盒查找表在计算过程中可以完全装入缓存中，那么通过缓存泄漏的时间信息就可以完全消除。这可以通过使用小体积的 S 盒查找表来实现，但由于 CPU 的多任务分时处理，它们仍然可能从缓存中被踢出去，文献[4]建议通过联合其他方法来提高安全级别。AES 的实现算法也可以使用逻辑运算来代替查表，但是这样效率会低很多。

另一种方法是在返回加密数据时采用随机化时间，这样就不能通过缓存的存取过程进行时间分析[5]。但是这种方法无法抵抗能量攻击。

在硬件设计方面，可以通过设计新的结构来划分缓存单元，防止数据从缓存中被强制清除[6]。

5 时间碰撞攻击与同类攻击方式的比较

时间驱动的缓存攻击(time-driven cache-base attacks)通过分析密码设备已经执行的一组操作和操作所花费的时间引起的缓存遗漏来推导加密系统在计算中涉及的秘密参量。这里将时间碰撞攻击与其他 2 种具有代表性的攻击形式加以比较：

(1)能量分析攻击(power analysis attacks)通过分析密码设备进行操作的功率消耗来推导加密系统所进行的操作和在操作中涉及的秘密参量。

(2)电磁泄漏攻击(electromagnetic emanation attacks)通过在芯片周围放置线圈并且研究可测量的电磁场来实现。

在上述 3 种攻击形式中，缓存时间攻击的解密能力是最弱的。但是，缓存时间攻击所需要的设备和测量的简单性使得这种攻击形式应用起来比较简单。在许多情况下，由于条件的限制，进行能量和电磁辐射的测量比较困难，甚至不能完成。然而改进的缓存攻击由于所需设备少，易于测量，因此可以冲破客观条件的制约完成攻击任务。

参考文献

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

[2] OpenSSL. OpenSSL: the Open-source Toolkit for SSL/TLS[Z]. (1999-09-09). http://www.openssl.org/.

[3] 雷 航, 王 茜. 现代微处理器及总线技术[M]. 北京: 国防工业出版社, 2006.

[4] Osvik D A, Shamir A, Tromer E. Cache Attacks and Countermeasures: The Case of AES[Z]. (2005-05-05) . http://eprint.iacr.org/2005/271.

[5] Bertoni G, Zaccaria V. AES Power Attack Based on Induced Cache Miss and Countermeasure[C]//Proc. of International Conference on Information Technology: Coding and Computing. Milan, Italy: IEEE Press, 2005: 586-591.

[6] Daniel P. Partitioned Cache Architecture as a Side-channel Defence Mechanism[Z]. (2005-05-05). http://mirror.cr.yp.to/eprint.iacr.org/2005/280.ps.

（上接第 140 页）

(2)如果敌手在对 n 用户提出同样的请求，而这 n 用户披露了同样的策略，由于提取的随机性，在策略库中的策略数目趋近无穷大时，n 是有限的。在这种情况下，敌手不能分辨到底谁有该属性，谁没有该属性，因此，不能通过对方所披露的策略推理出敏感属性的拥有情况。

(3)由于策略库的内容是公开的，当用户披露的某一属性的策略在策略库中指定属性的策略中不存在，这有可能是策略库里没有满足用户的策略，但改进的策略库还保证了不存储一次提交的策略里的相同策略，用户可用已有的相同策略来保护与之存在层次关系的敏感属性而不会造成信息泄露。

综上所述，改进的策略库满足安全性定义。

4.3 可用性分析

改进策略库增加了一个契约中间件，假设有m个策略，$n_i(0<i<m+1)$为策略中包含的属性数目，其中$n=\max(n_1,n_2,\cdots)$，该件在检查是否存在包含关系和比较层次关系的时候，时间复杂度为$O(n\times m)$。

层次契约模型给拥有敏感属性的用户制定策略时提供了参考模型，更加有利于资源拥有敏感的保护。

参考文献

[1] Winsborough W H, Seamons K E, Jones V E. Automated Trust Negotiation[C]//Proc. of DARPA Information Survivability Conf. and Exposition. New York, USA: IEEE Press, 2000: 88-102.

[2] 李建欣. 自动信任协商研究[J]. 软件学报, 2006, 17(1): 124-133.

[3] Yu Ting, Winslett M. A Unified Scheme for Resource Protection in Automated Trust Negotiation[C]//Proc. of IEEE Symposium on Security and Privacy. [S. l.]: IEEE Press, 2003: 245-257.

[4] Seamons K, Winslett M, Yu Ting, et al. Protecting Privacy During On-line Trust Negotiation[C]//Proc. of the 2nd Workshop on Privacy Enhancing Technologies. San Francisco, CA, USA: [s. n.], 2002-04.

[5] Winsborough W. Safety in Automated Trust Negotiation[C]//Proc. of IEEE Symposium on Security and Privacy. Oakland, CA, USA: [s. n], 2004-05.

[6] Irwin K, Yu Ting. Preventing Attribute Information Leakage in Automated Trust Negotiation[C]//Proceedings of the 12th ACM Conference on Computer and Communications Security. [S. l.]: ACM Press, 2005-11: 41-51.