针对 AES 加密前两轮的访问驱动 Cache 攻击方法

赵新杰,米 东,王 韬,郑媛媛,陈财森,郑 伟
ZHAO Xin-jie,MI Dong,WANG Tao,ZHENG Yuan-yuan,CHEN Cai-sen,ZHENG Wei

军械工程学院 计算机工程系,石家庄 050003
Department of Computer Engineering,Ordnance Engineering College,Shijiazhuang 050003,China
E-mail:zhaoxinjieem@163.com

ZHAO Xin-jie,MI Dong,WANG Tao,et al.Access-driven Cache attack method against AES on the first two rounds.Computer Engineering and Applications,2009,45(16):130-133.

Abstract:The memory Cache has the features of data access time uncertainty and multi-process resource sharing.The AES software implementation uses lots of table lookup operations,while these indices will affect the Cache hit or miss,and these indices have a close connection with the secret key.According to the 128 bit AES,authors use a spy process to gather cache access patterns of the AES process,after analyzing the relationship among the table lookup indices,the plaintext and the initial key during the first two rounds encryption,64 bit partial key can be recovered through the first round analysis,the extra key can be found through the second round analysis,finally the full 128 bit AES key can be successfully got.

Key words:access-driven;Cache;timing attack;Advanced Encryption Standard(AES);table lookup indices

摘 要:高速缓存 Cache 具有数据访问时间不确定和多进程资源共享两大特征,AES 加密快速实现中使用了大量查表操作进行 Cache 访问,查表索引值会影响 Cache 命中与否,而查表的索引值和密钥存在密切关系。针对 128 位 AES 加密算法,利用间谍进程采集 AES 进程加密时 Cache 访问特征信息,通过对 AES 前两轮加密过程中查表索引值、明文和初始密钥之间关系进行分析,第一轮分析可获取 64 位密钥,第二轮分析可获取剩余密钥,最终成功获取 AES 全部密钥。

关键词:访问驱动;Cache;计时攻击;高级加密标准;查表索引
DOI:10.3778/j.issn.1002-8331.2009.16.038 **文章编号**:1002-8331(2009)16-0130-04 **文献标识码**:A **中图分类号**:TP309

1 引言

近来,密码界提出了一种新的解密方法,Cache 密码分析学。其基本原理是:现代加密算法大多都要进行查表操作访问存储器中数据信息,而相同的数据或指令访问存储器时,由于访问目标数据当前是否在 Cache 中同 Cache 的历史状态有关,此时是否发生 Cache 命中是不确定的,这将导致数据访问操作的不确定。这种不确定性行为可以通过计时差异信息泄露出来,通过采集此类计时差异信息,结合加密算法数学特性可推断出与密钥相关的秘密信息,甚至直接得到部分或全部密钥,AES 加密算法面临严重威胁。本文介绍了一种新的基于 Cache 行为的密码攻击方法——访问驱动 Cache 攻击[1],并结合 128 位 AES 加密算法针对其加密前两轮进行了攻击应用。

2 AES 访问驱动 Cache 攻击基本原理

2.1 AES 加密算法

AES[2]是一个基于有限域运算的迭代密码,其加密过程如图 1 所示,第 r 轮迭代根据 16 Byte 的输入状态 x_r 和一个 16 Byte 的子密钥 K_r,产生一个 16 Byte 的输出状态 x_{r+1}。除最后一轮之外,每一轮迭代运算都包括对 x_r 的以下 4 个代数运算:字节代换、行移位、列混淆,然后和轮子密钥 K_r 做异或运算。

为提高性能,现代常用的 AES 加密算法[3]在每一轮实现中,将轮密钥和状态变量 x 异或以外的加密操作合并为查表操作,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,T_4)中,这样查表操作就成为对 Cache 中查找表进行的 Cache 访问。

$$\begin{aligned}T_0[x]&=(2\cdot S(x),S(x),S(x),3\cdot S(x))\\T_1[x]&=(3\cdot S(x),2\cdot S(x),S(x),S(x))\\T_2[x]&=(S(x),3\cdot S(x),2\cdot S(x),S(x))\\T_3[x]&=(S(x),S(x),3\cdot S(x),2\cdot S(x))\\T_4[x]&=(S(x),S(x),S(x),S(x))\end{aligned}\tag{1}$$

$S(x)$ 代表对于每一个输入值 x,AES 进行查找 S 盒操作的结果,"$\cdot$"代表一个在有限域 GF(2^8) 的字节乘法运算。第 r 轮的加密计算就可以用公式(2)来表示:

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60772082)。
作者简介:赵新杰(1986-),男,硕士研究生,主要研究领域为信息安全;米东,男,教授,博士生导师;王韬,男,教授,博士生导师。
收稿日期:2008-04-10 修回日期:2008-06-23

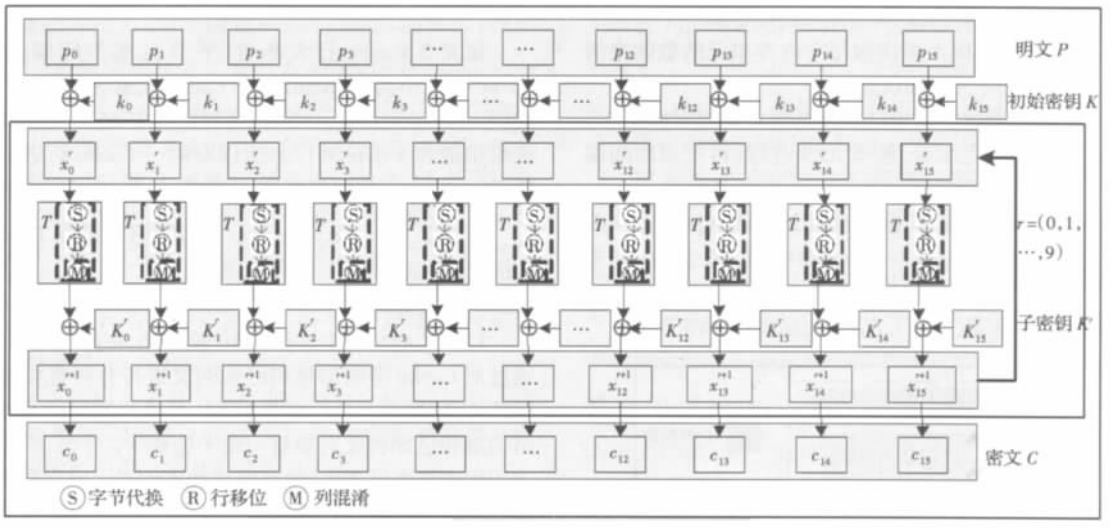

图 1　AES 加密流程图

$$\begin{cases}(x_0^{r+1},x_1^{r+1},x_2^{r+1},x_3^{r+1})=T_0[x_0^r]\oplus T_1[x_5^r]\oplus\\ \quad T_2[x_{10}^r]\oplus T_3[x_{15}^r]\oplus(K_0^r,K_1^r,K_2^r,K_3^r)\\ (x_4^{r+1},x_5^{r+1},x_6^{r+1},x_7^{r+1})=T_0[x_4^r]\oplus T_1[x_9^r]\oplus\\ \quad T_2[x_{14}^r]\oplus T_3[x_3^r]\oplus(K_4^r,K_5^r,K_6^r,K_7^r)\\ (x_8^{r+1},x_9^{r+1},x_{10}^{r+1},x_{11}^{r+1})=T_0[x_8^r]\oplus T_1[x_{13}^r]\oplus\\ \quad T_2[x_2^r]\oplus T_3[x_7^r]\oplus(K_8^r,K_9^r,K_{10}^r,K_{11}^r)\\ (x_{12}^{r+1},x_{13}^{r+1},x_{14}^{r+1},x_{15}^{r+1})=T_0[x_{12}^r]\oplus T_1[x_1^r]\oplus\\ \quad T_2[x_6^r]\oplus T_3[x_{11}^r]\oplus(K_{12}^r,K_{13}^r,K_{14}^r,K_{15}^r)\end{cases}\quad(2)$$

这样,整个加密过程就由 160 次查表操作和 176 次异或操作组成,执行效率非常高,但是因为查找表数据存储在 Cache 中,而查找表索引信息和密钥有着密切的关系,所以只要采集到足够多的 AES 加密查找表索引信息,就可以推算出某一轮的扩展密钥,由于 AES 密钥扩展结构的可逆性,攻击者只要恢复了中间任意轮 16 Byte 扩展密钥就相当于恢复了原始密钥,这些为针对 AES 的访问驱动 Cache 攻击提供了很好的切入点。

2.2　Cache 访问

现代微处理器和微型计算机中,尤其在 PC 机中,大都使用高速缓存 Cache 来解决 CPU 与主存之间速度不匹配的问题。Cache 和主存结构如图 2 所示,每一个小块为一个基本存储单元,在 Cache 中称之为一个 Cache 行,共有 B 字节,每个 Cache 行由 δ 个 Cache 元素组成;每一列为一个 Cache 组,共 S 个 Cache 组;每组由 W 个 Cache 行组成,这样整个 Cache 大小为 $S*W*B$ 字节。

Cache 映射机制[4-6]为:

(1) CPU 读取主存中一个字 A 时,首先将 A 内存地址传送到 Cache 和主存中,Cache 控制逻辑依据地址判断 A 当前是否在 Cache 中;如果是,产生"Cache 命中",A 立即被传送到 CPU;否则产生"Cache 失效",需把 A 从主存读出送到 CPU,与此同时,将包含 A 的整个数据块 B 字节从主存中读出送到 Cache 中;

(2) 地址为 a 的内存块只能被映射到 Cache 组[a/B]mod/S 中,多进程在主存中的私有数据可能被映射到同一 Cache 组中。

由(1)知同样一条访问存储器的指令在目标数据不在 Cache 内的时候就可能产生延迟,而这种延迟表现在程序的执行结果上是程序较长的运行时间或是较大的能量消耗,就典型处理器中数据访问时间差异来说,"Cache 命中"时直接访问 Cache 大致需要 0.3 ns,而"Cache 失效"时访问主存则需要 50~150 ns;由(2)知不同进程在对自身私有数据进行内存访问时,由于这些数据可以被映射到同一 Cache 组中,进而共享 Cache 存储空间,这样恶意进程可以通过对自身数据 Cache 访问的时间或者能量消耗差异来监测其他进程的 Cache 访问特征。

访问驱动 Cache 攻击正是根据 Cache 访问时间不确定性和 Cache 资源共享机制,利用间谍进程监视加密进程的 Cache 访问操作,通过计时方法采集本进程 Cache"命中"和"失效"特征信息,间接得到加密进程 Cache 访问特征信息,并结合其他旁路信息分析方法,缩小密钥搜索空间甚至得到完整密钥。

3　AES 访问驱动 Cache 攻击模型[7]

由第 2 章知道 Cache 具有数据访问时间不确定和多进程资源共享两大特性,而改进后的 AES 算法使用了大量的查表操作对 Cache 进行访问,因此通过编写间谍进程采集 AES 加密进程所访问的 Cache 行地址信息,结合其他分析方法,就可以获取到部分甚至全部密钥信息[7]。假设间谍进程所占内存空间为字节数组 $A[0,\cdots,S*W*B-1]$,其起始地址和 AES 查找表 T_l 起始地址被映射到 Cache 中同一 Cache 组,如图 2 所示。

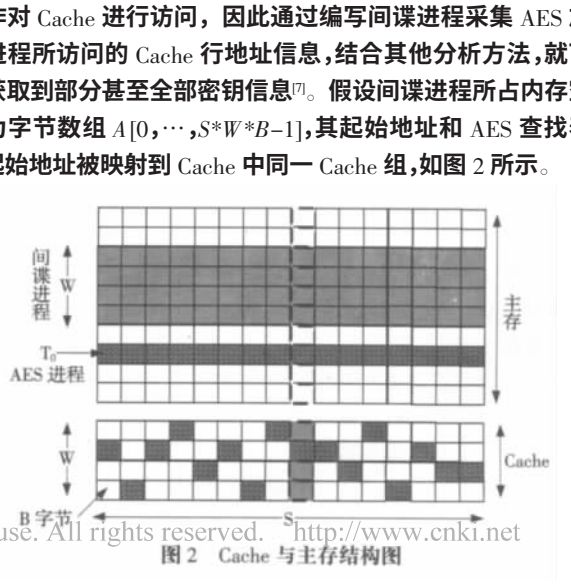

图 2　Cache 与主存结构图

攻击模型示意图如图 3 所示,具体攻击过程如下:

(1) 间谍进程启动,从 A 中读取每个内存单元的数据来清空 Cache,初始化 Cache 为已知状态。

(2) 触发 AES 进程执行基于已知随机明文 P 的加密操作。

(3) AES 加密操作完成后,悬挂 AES 进程;再次启动间谍进程,对于每一个 AES 查找表 T_l(l=0,1,2,3) 及其索引 y=0,δ,2δ,3δ,…,256−δ,再次读取 W 次内存地址单元 $A[1\ 024l+4y+tSB]$,t=0,1,…,W−1,通过 Cache 访问"命中"和"失效"时间差异信息推测 AES 进程查表操作所访问的 Cache 行 CL。

(4) 根据所采集到 $M(P_i(C_i),T_l,CI)$ 信息来预测密钥 K。

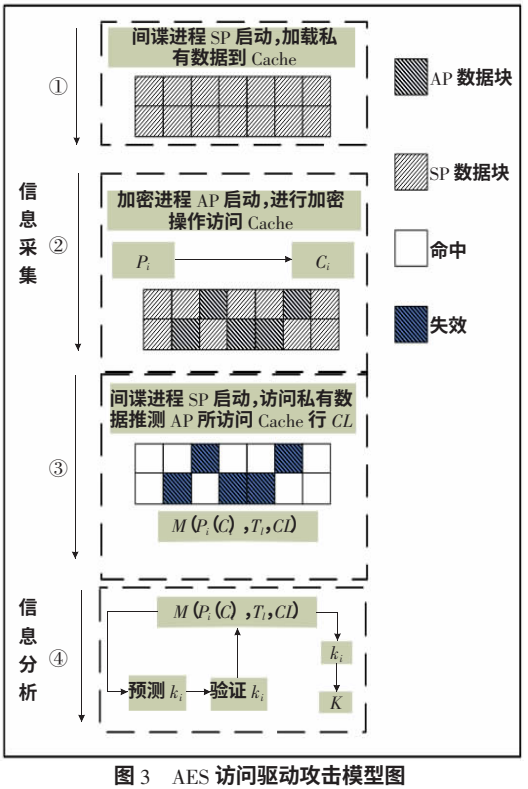

图 3 AES 访问驱动攻击模型图

4 针对 AES 加密前两轮的攻击

4.1 攻击条件及约定

针对 AES 进行访问驱动攻击需要具备以下攻击条件:

条件 1 已知明文或者密文信息。

条件 2 能够进行进程控制[8]。间谍进程可以和加密进程并发运行,能够自动触发 AES 加密,加密完成后可以悬挂AES 进程。

条件 3 加密前清空 Cache。在触发 AES 进程加密前,要先启动间谍进程访问私有数据进行 Cache 读写,实现 Cache 清空操作。

条件 4 高精度计时[9]。由于 Cache"命中"和"失效"时间差异精度很高,需能够根据 CPU 时间戳进行纳秒级高精度计时。

条件 5 查找表及 Cache 行定位[10]。能够定位 AES 进程加密过程中查找表 T_l 起始位置,能够准确定位 AES 进程加密过程中所访问的 Cache 行地址。

为了便于对所采集数据进行分析处理,有以下预先定义:

定义 1 $E_k(P,T_l,y)$:P 为已知随机明文,T_l 为 AES 查找表,y 为查找表 T_l 索引值。$E_k(P,T_l,y)=1$ 时表示十轮迭代中明文 P 在密钥为 K 情况下对查找表 T_l 索引为 y 的 Cache 行至少访问一次。

定义 2 Cache 行大小 B(字节):本节所描述的处理器 Cache 行大小为 64 Byte,每个 Cache 元素为 4 Byte,δ=16,因此对同一 Cache 行进行访问将会对应其中 16 个可能值。每次查表索引值为 1 Byte(8 位),所以对同一 Cache 行进行访问索引值最后 lb 16=4 位可能不同,但是前 8−lb 16=4 位相同。

定义 3 k_i,$\tilde{k}_i$,$<k_i>$:k_i 表示初始密钥 K 的第 i 个字节,$\tilde{k}_i$ 为 k_i 预测值,$<k_i>$表示 k_i 部分位值。

4.2 攻击原理

128 位 AES 加密进行了 10 轮迭代,在前两轮迭代时,可以通过对 Cache 旁路信息和已知明文进行分析直接推算出初始密钥 K;同样在最后一轮迭代时,可以通过对所采集 Cache 旁路信息和已知密文对最后一轮子密钥 K_{10} 分析,然后利用 AES 密钥生成算法可逆性,推算出初始密钥 K。本文的攻击主要针对前两轮迭代进行分析。

在第一轮迭代中,$x_i^{(0)}=p_i\oplus k_i$,x_0 为第一轮迭代中查表索引值,p_i 和 k_i 分别为明文和初始密钥的第 i 个字节,上式可变换为 $k_i=p_i\oplus x_i^{(0)}$,由于 $x_i^{(0)}$ 的部分值可以通过采集 $E_k(P,T_l,y)$ 信息获取到,那么便不难推测出子密钥 k_i 部分位值$<k_i>$,进而得到密钥 K 部分信息 K^H,通过对第二轮迭代中 Cache 旁路信息进一步分析可得到 K 剩余未知信息 K^L,最终获取完整密钥 K。

4.3 攻击实施

攻击分为样本采集、第一轮分析、第二轮分析三个阶段,样本采集阶段主要采集 AES 对已知明文加密过程中的 $E_k(P,T_l,y)$ 信息,然后分别对第一轮迭代、第二轮迭代进行离线分析,在离线分析中采用预测验证方法:首先预测子密钥 k_i 预测值$\tilde{k}_i$,使用$\tilde{k}_i$ 预测可能的 Cache 访问旁路信息,根据所预测信息和实际所采集 Cache 旁路信息进行比对验证,数据不一致则排除$\tilde{k}_i$,数据一致则保留$\tilde{k}_i$,多次验证得到 k_i,最终推测出 K。

4.3.1 样本采集

令 AES 加密进程 AP 和间谍进程 SP 并发运行在同一 PC 机上,SP 对 AP 的数据访问进行严密监视,反复不停地向 Cache 中加载和 Cache 空间同样大小的表 A,然后触发 AP 进行加密操作,AP 在加密过程中进行查表操作数据访问时不可避免地要将表 A 中的部分数据从 Cache 中驱逐出去,SP 检测到 AP 一次加密执行完毕后,立即对表 A 中所有数据进行再次访问,如果对某些数据访问需要占据更多的时钟周期,则推断这些数据以前所在的 Cache 行在加密过程中被加密进程访问过,进而可以获取到一次完整加密过程中对所有查找表访问的 Cache 行 CL,进而得到 $E_k(P,T_l,y)$ 信息。下面详细描述如何利用这些信息进行旁路分析。

4.3.2 第一轮分析

由公式(2)知,AES 加密第一轮迭代时有 $x_i^{(0)}=p_i\oplus k_i$,$i\in(0,\cdots,15)$,则对于已知明文 p_i,在获取到查表操作的索引值 $x_i^{(0)}$ 时,便不难得出 k_i 的候选值$\tilde{k}_i$,进而经过样本验证得到 k_i。具体分析方法如下:

上节样本采集阶段中,采集到了大量的 $E_k(P,T_l,y)$ 样本信息,$T_l[x_i^{(0)}]$ $(i=l\ (\mathrm{mod}\ 4))$ 表示 k_i 在第一轮迭代中和 p_i 异或后进行查 T_l 表操作结果,可以通过验证候选值$\tilde{k}_i$ 来推测 k_i 中部分

位值$<k_i>$。验证方法为：对于满足$<y>=<p_i \oplus \tilde{k}_i>$的样本，如果$<k_i>=<\tilde{k}_i>$，那么此时对查找表 T_l 索引为 y 所对应的 Cache 行会发生一次访问，由定义 1 可知 $E_k(p,l,y)=1$，相反，如果$<k_i> \neq <\tilde{k}_i>$，就不会访问查找表 T_l 索引为 y 所对应的 Cache 行，但是整个加密过程中对 T_l 还有 35 次其他的访问，这 35 次访问是受其他明文字节 p_j 所影响的，所以对于随机明文来说，在任何一轮迭代中不访问 T_l 索引为 y 所对应 Cache 行可能性 P 为 $(1-\delta/256)^{35}$，这也就是之前定义的 $E_k(P,T_l,y)=0$ 的情况，对于 $\delta=16$，$P=0.104$。这样，在对一定数量的样本进行分析后可以确认$<\tilde{k}_i>$值，此时满足 $E_k(P,T_l,y)=1$ 并且$<y>=<p_i \oplus \tilde{k}_i>$。

通过这种方法对 k 中每个字节进行独立分析，可得到 k 中每个字节 k_i 的高 $\mathrm{lb}(256/\delta)=4$ 位$<k_i>$，k_i 的一半值，这是第一轮分析能获取到的密钥每个字节最多信息。

4.3.3　第二轮分析

通过第一轮分析可推断出密钥 K 的 64 位信息，要想得到全部密钥，还需对第二轮迭代进行分析，具体分析过程如下：

由公式(1)和(2)，可以推导出第二轮加密中的查表索引值，如公式(3)所示。

$$
\begin{aligned}
x_2^{(1)} &= S(p_0 \oplus k_0) \oplus S(p_5 \oplus k_5) \oplus 2 \cdot S(p_{10} \oplus k_{10}) \oplus \\
&\quad 3 \cdot S(p_{15} \oplus k_{15}) \oplus S(k_{15}) \oplus k_2 \\
x_5^{(1)} &= S(p_4 \oplus k_4) \oplus 2 \cdot S(p_9 \oplus k_9) \oplus 3 \cdot S(p_{14} \oplus k_{14}) \oplus \\
&\quad S(p_3 \oplus k_3) \oplus S(k_{14}) \oplus k_1 \oplus k_5 \\
x_8^{(1)} &= 2 \cdot S(p_8 \oplus k_8) \oplus 3 \cdot S(p_{13} \oplus k_{13}) \oplus S(p_2 \oplus k_2) \oplus \\
&\quad S(p_7 \oplus k_7) \oplus S(k_{13}) \oplus k_0 \oplus k_4 \oplus k_8 \oplus 1 \\
x_{15}^{(1)} &= 3 \cdot S(p_{12} \oplus k_{12}) \oplus S(p_1 \oplus k_1) \oplus S(p_6 \oplus k_6) \oplus \\
&\quad 2 \cdot S(p_{11} \oplus k_{11}) \oplus S(k_{13}) \oplus k_{15} \oplus k_3 \oplus k_7 \oplus k_{11}
\end{aligned}
\tag{3}
$$

根据公式(3)，假设对于查找表 T_2、查表索引值 y、已知随机明文 P，通过第一轮分析，可得到$<k_0>$、$<k_2>$、$<k_5>$、$<k_{10}>$，$<k_{15}>$，未知的 k_2 低 4 位字节仅仅影响到 $x_2^{(1)}$ 的低 4 位字节，并不改变 $T_2[x_2^{(1)}]$对应内 Cache 行，所以改变 $T_2[x_2^{(1)}]$对应 Cache 行的是 k_0、k_5、k_{10}、k_{15} 的低 $\mathrm{lb}\,\delta$ 字节，这就有了 2^{4*4} 个$\tilde{k}_0$、$\tilde{k}_5$、$\tilde{k}_{10}$、$\tilde{k}_{15}$候选值，通过对这些有限候选值进行验证，可以推测出这 4 个字节所有剩余未知位值，验证方法如下：

对于每一个候选值和对应样本，对于低 4 位为任意值的密钥字节 k_2，在公式(3)中由$\tilde{k}_0$、$\tilde{k}_5$、$\tilde{k}_{10}$、$\tilde{k}_{15}$得出预测索引值$\tilde{x}_2^{(1)}$，如果$<y>=<\tilde{x}_2^{(1)}>$，可得$<y>=<\tilde{x}_2^{(1)}>=<x_2^{(1)}>$，此时对 T_2 中索引为 y 所对应 Cache 行会发生一次访问，可得 $E_k(P,T_l,y)=1$，定义该样本为候选值对应的有效样本，否则对于 $k_i \neq \tilde{k}_i, i \in (0,5,10,15)$，有 $x_2^{(1)} \oplus \tilde{x}_2^{(1)} = d \cdot s(pi \oplus k) \oplus f \cdot s(pi \oplus \tilde{k}) \oplus \cdots, d,f \in (1,2,3)$，因为 P 为随机明文，则对 T_2 中索引为 y 对应 Cache 行 $T_2[x_2^{(1)}]$访问可能性至少为 $(1-\delta/256)^3$，那么其他 35 次对 T_2 中索引为 y 对应 Cache 行 $T_2[x_2^{(1)}]$访问可能性为 $\delta/256$，则 $E_k(P,T_l,y)=0$ 可能性要大于 $(1-\delta/256)^{3+35}$。于是可得出排除样本比例为 $(\delta/256)\cdot(1-\delta/256)^{38}$，为了将 2^{16} 样本中错误候选值排除出去，需要 $\log\delta^{-4}/\log(1-\delta/256\cdot(1-\delta/256)^{38})$ 个样本，大约 2 056 个样本。同样，根据公式(3)对应其他 3 个公式，可以得到其他密钥字节剩余低 4 位信息，进而得到完整密钥 K。

4.4　攻击实验

根据前面攻击原理和实施过程，可编写针对 AES 加密前两轮访问驱动攻击算法，其流程如图 4 所示。

实验样本分析时，通过对 $E_k(P,T_l,y)$ 和 k_i 部分位预测值$<k_i>$采样分析，可得到$<k_i>$值，这样，通过第一轮分析获取到密钥 K 中每一个字节高 4 位值 K^H，通过第二轮分析获取到密钥 K 中每一个字节低 4 位值 K^L，最终得到完整密钥 K，如图 5 所示。

5　结束语

高速缓存 Cache 具有数据访问时间不确定和多进程资源

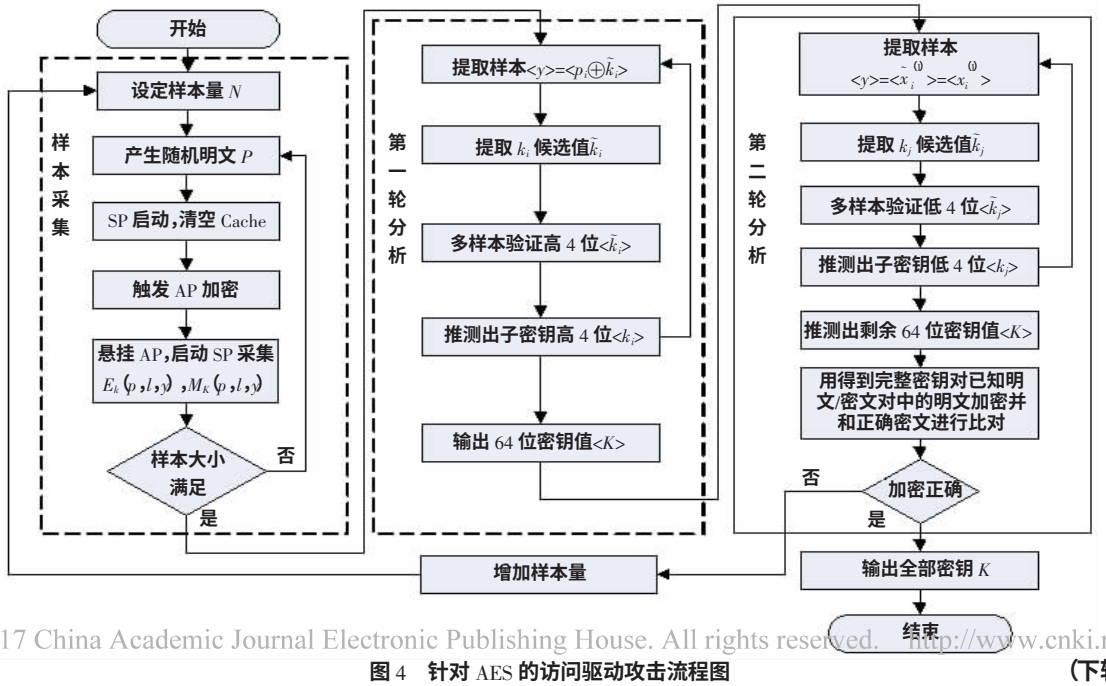

图 4　针对 AES 的访问驱动攻击流程图

(下转 210 页)

A 进行划分后得到 $A_1,A_2,\cdots,A_m$ 个子系统,利用离散时间系统给出每个子系统的切换关系 $x(k+1)^{(i)}=A_{\sigma(i)}x(k)^{(i)},i\leqslant m$,例如,若在图 1 中已经存在一条进路 $L_1=\{11,12,20\}$,令 $D=L_1$,显然 A_3 和 A_4 的切换关系受到影响,给出 A_1、A_4 的切换关系如下,其中 $x_i(k)$ 表示路段 i 在 $t(k)$ 时的状态。

$$\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}(k+1)=A_1X(k)=\begin{bmatrix}0&0&0&0\\0&0&0&0\\1&1&0&0\\1&1&0&0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}(k)$$

$$\begin{bmatrix}x_{12}\\x_{14}\\x_{16}\\x_{18}\\x_{17}\\x_{19}\\x_{20}\end{bmatrix}(k+1)=A_4X(k)=\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&1&1&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}X(k)$$

A_i 中输入状态对应的行全为 0,输出状态的列全为 0,转移的步数 k 表示输入到输出路段数。设 A_1 中进路初始路段为 1,则 $x_0=[1,0,0,0]^{\mathrm{T}}$,利用切换矩阵 A_1 得到 $x(1)=A_1x_0=[0,0,1,1]^{\mathrm{T}}$,说明经过一步转移后 1 可以切换到 3 或者 4。需要注意的是矩阵乘积运算使用的是逻辑和,经过多次转移后均可以得到每个子系统中输入到输出路段的转移关系。当一个子系统切换到另一个时,因前一个子系统的输出作为后一个的输入,所以将两个子系统的路径经过串联生成更长的路径,进路就是通过其切换序列各子系统路径串联而成。例如 1 到 19,需要进行 A_1、A_5 到 A_4 的切换,A_1 中经过一步切换从 1→4,A_5 中经一步 4→16,在 A_4 中经过二步切换 16→18→19,所以 1 到 19 的进路序列为{1,4,16,18,19}。如果选择 1 到 20 的进路也需要进行与 1 到 19 同样的切换,但因为敌对信号 D 的存在而使 A_4 中不存在到 20 的进路,所以 1 到 20 的进路目前不存在。

4 算法仿真结果与讨论

切换算法将系统进行划分,实际上是将大系统分为若干个小系统,若将小系统视为节点,通过图的遍历容易寻找到进路起始点 x_0 所在节点到进路终止端 s_f 所在节点的路径,将每条路径定义为一个切换序列关系,设$\{\sigma(0),\sigma(1),\cdots,\sigma(k-1)\}$为一经过 k 步切换序列,则有 $s_f\in A_{\sigma(k-1)}\cdots A_{\sigma(1)}A_{\sigma(0)}x_0$。当然也可以通过直接切换来实现上述过程,因为前一个切换子系统的输出与后一个切换子系统输入的交集必然非空,设 $x_0\in A_{\sigma(0)}$,计算 $x(1)=A_{\sigma(0)}x_0$,若存在 $x(k)^{(\sigma(1))}\cap x(1)$非空,则切换到 $A_{\sigma(1)}$,计算 $x(2)=A_{\sigma(1)}A_{\sigma(0)}x_0$,进行类似的比较和切换,直到 $A_{\sigma(k-1)}\cdots A_{\sigma(1)}A_{\sigma(0)}\cdot x_0$ 中包含 s_f 结束。

算法的重点是寻找到进路始终端之间的切换序列,而子系统内部输入到输出的切换关系因为维数的减少而使计算量减少,子系统 A_i 的 k 步输入到输出的计算公式为 $x(k)=A_i^kx(0)$。传统进路搜索算法的数量级为 $O(n)$,而切换系统为 $O(n/m)$,n 为系统维数,m 为子系统个数。切换控制方法因为系统的划分不同其切换序列会不同,但其最终求解的进路则是相同的。切换控制方法在复杂的大型的站场中将有更好的应用价值,借助计算机程序可实现进路的自动生成。

参考文献:

[1] 祝庚.K 步故障扩散定位算法的 Matlab 实现[J].计算机工程与设计,2008,29(8):1900-1903.

[2] Sun Z.Switched linear systems-control and design[M].London,Springer,2005.

[3] 吕兴寿.基于图形结构的列车/调车进路连锁表的自动生成算法研究[J].计算机应用研究,2006(5):77-79.

[4] Sun Zhen-dong.Reachability analysis of constrained switched linear systems[J].Automatica,2006,9:164-167.

[5] 杨运贵,王慈光,薛锋.树枝形铁路专用线取送车问题的遗传算法研究[J].计算机工程与应用,2008,44(12):210-211.

(上接 133 页)

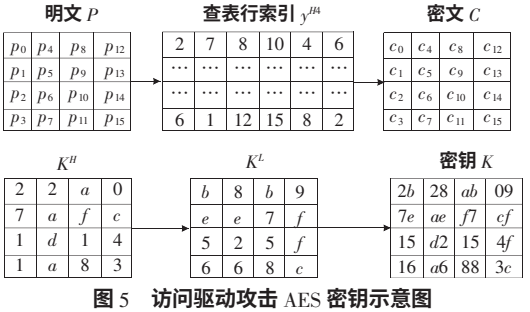

图 5 访问驱动攻击 AES 密钥示意图

共享两大特征,而 AES 算法加密时需要进行大量查表操作访问 Cache,此时表现出来的和查表索引相关的“命中”和“失效”特征为密码分析技术提供了一条新的途径,本文通过对 AES 加密算法前两轮迭代过程中的 Cache 访问特征旁路信息进行分析可获取到完整 128 位加密密钥。

以下几个方面将很值得人们研究和关注,如针对 AES 加密最后一轮迭代、针对其他分组加密算法、基于远程的访问驱动 Cache 攻击以及攻击有效防御措施等。

致谢 特别感谢矫文成教授以及杨杰、伦朝阳的指导和帮助。

参考文献:

[1] Neve M,Seifert J-P.Advances on access-driven cache attacks on AES[C]//Proceedings of Selected Areas in Cryptography 2006.

[2] Advanced Encryption Standard (AES).Federal Information Processing Standards Publication 197,2001.

[3] Daemen J,Rijmen V.The design of Rijndael:AES—the advanced encryption standard[M].[S.l.]:Springer-Verlag,2002.

[4] Handy,Jim.The cache memory book[M].2nd ed.[S.l.]:Academic Press Inc,1998:8-21,64-66,89-94.

[5] 胡向东,魏琴芳.应用密码学[M].北京:电子工业出版社,2006:83-105.

[6] 计算机组织与体系结构性能设计[M].张昆藏,译.北京:清华大学出版社,2006,78-110.

[7] Tromer E.Hardware-based cryptanalysis[D].Weizmann Institute of Science,Rehovot Israel,2007:133-151.

[8] Neve M.Cache-based vulnerabilities and SPAM analysis[D].UCL,2006.

[9] Yuan Feng.Windows graphics programming Win32 GDI and DirectDraw[M].[S.l.]:Prentice Hall PTR,2000.

[10] Steven A.Cache and memory hierarchy design:A performance directed approach[M].[S.l.]:Przybylski,1990.