

面向无线传感器网络的抗攻击低开销 AES 实现方法

罗新强 齐 悦 万亚东 王 沁

(北京科技大学计算机与通信工程学院 北京 100083)

摘 要 高级加密标准(AES)加密被众多无线传感器网络(WSN)标准定义为其数据链路层的核心密码算法。传统 AES 实现由于计算复杂度高,难以在资源受限的 WSN 节点上实现。通过查找表可以大幅提高 AES 软件实现的加密速度,但是传统的基于 4 张 1kB 查找表的 AES 实现(4-T)不仅存储开销大,而且面临着访问驱动 Cache 攻击的威胁。通过对查找表的结构进行优化,提出一种基于单张 512B 查找表的 AES 实现方法(1-T),在降低存储开销的同时,提高了 AES 实现抵抗访问驱动 Cache 攻击的能力。此外,通过对轮加密公式的优化,减小了对加密速度的影响。在 ARM 平台上的实验显示,1-T 实现的加密时间相比 4-T 增加 43.5%,但仅是采用硬件加速器 AES 实现的加密时间的 38.55%。

关键词 无线传感器网络, AES 加密, 查找表, 访问驱动 Cache 攻击, 低开销

中图法分类号 TP309.7 文献标识码 A

Attack-resistant and Low-cost AES Implementation for Wireless Sensor Network

LUO Xin-qiang QI Yue WAN Ya-dong WANG Qin

(School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract Advanced encryption standard(AES) is specified as the core cipher algorithm of the data link layer by many wireless sensor network(WSN) standards. But traditional AES implementation is hard to perform on the resource-constrained WSN nodes due to its high computation complexity. Look-up table(LUT) can improve the speed of AES software implementations significantly, but the traditional AES implement(4-T) based on 4 LUTs consumes high storage and faces the threat of access-driven cache attack. This paper proposed an AES implementation(1-T) based on one 512-Byte LUT, by optimizing the structure of the LUT, decreasing its storage consumption and increasing its ability of against access-driven cache attack significantly at the same time. In order to eliminate the encryption speed impact on 1-T, the round encryption function of 1-T was optimized as well. The experiment result on ARM shows that, the 1-T's encryption time is increased 43.5% comparing to 4-T's, but only 38.55% of the one of the AES implementation based on hardware accelerator.

Keywords Wireless sensor network, AES encryption, Look-up table, Access-driven cache attack, Low-cost

1 引言

无线传感器网络(WSN)^[1,2]由于其信道及部署环境开放,其安全性受到越来越多的关注,各种针对 WSN 安全研究层出不穷。虽然诸如网络诊断^[3]、隐私保护^[4]等方法有效地提高了 WSN 的防御能力,但是对于很多资源受限的 WSN 节点并不适用。数据传输安全提供了数据隐私性和完整性保护,为 WSN 提供了最基本的安全保障,是目前众多 WSN 普遍采用的安全措施。而密码算法作为数据安全传输的核心,也成为 WSN 安全领域研究的重点。虽然非对称算法能够提供比对称算法更高的安全性,针对 WSN 的低开销非对称算法研究^[5]也不断出现,但其开销相对对称算法而言仍旧给 WSN 节点带来沉重的负担。高级加密算法(AES)^[6]由于其相对高的安全性和灵活性,被许多 WSN 标准都定义为它们

数据链路层的核心密码算法。尽管 AES 可以满足很多 WSN 应用的需求,但 AES 加密时间仍然开销过大,对许多基于嵌入式环境的且对网络帧处理有实时性要求的 WSN 应用提出了很大挑战。究其原因, AES 加密计算中包含了多次 GF(28)乘法运算,而这些乘法运算不被处理器支持,致使完成这些运算需要耗费大量的时钟周期,从而导致 AES 加密时间开销过大。

自 AES 发布以来,如何提高 AES 的加解密速度一直是学术界和工业界的研究热点,主要有 3 类方法:硬件加速^[7-11]、指令集扩展^[12-15]和软件实现优化^[16-19]。硬件加速和指令集扩展利用硬件完成 AES 加密中的全部或部分操作,可以大幅度提高 AES 的计算速度,但由于需要额外硬件的支持,灵活性较差。软件实现优化是通过对软件实现进行优化,从而提高 AES 的执行速度。一种公认的软件实现优化方法

本文受国家高技术研究发展计划(863 计划)项目(2011AA040101-3, 2014AA041801-2), 国家自然科学基金项目(61003251, 61172049, 61173150)资助。

罗新强(1986—),男,博士生,主要研究方向为工业无线网络安全;齐悦(1975—),女,博士,讲师,主要研究方向为工业无线网络、嵌入式系统, E-mail: qiyue@ustb.edu.cn; 万亚东(1982—),男,博士,讲师,主要研究方向为工业无线网络;王沁(1961—),女,博士,教授,博士生导师,主要研究方向为工业无线网络、计算机体系结构。

是查找表方法^[20],即通过查表的方式取代 AES 加密中的部分操作,有效提高 AES 的计算速度。Gladman^[17]通过建立 4 张 1kB 的查找表,将每轮加密操作转换为 16 次查表操作和 16 次异或运算,从而将 AES 加密的时间开销降到 993 个周期。

虽然通过查表方式可以有效提高 AES 软件实现的计算速度,但是 4 张 1kB 的查找表对于基于资源有限嵌入式系统的 WSN 节点而言形成了不小的负担。与此同时,对于采用 Cache 的系统而言,查表过程中产生的 Cache “命中”和“缺失”所形成的时间和能量变化特征,为基于 Cache 访问特征的攻击提供了方便。

目前,基于 Cache 访问特征对基于查找表的 AES 实现进行攻击(简称 cache 攻击)也已成为 AES 研究的一大热点,主要分为 3 类:时序驱动攻击(Time driven)^[21-24]、踪迹驱动攻击(Trace driven)^[25-27]和访问驱动攻击(Access driven)^[28-30]。时序驱动攻击利用不同明文和密钥所引起 Cache 访问的时间差异,通过统计方法分析出密钥。由于诸如操作系统调度、处理器架构、温度等因素都会对 AES 加密时间产生影响,同时统计所需样本集也很大,时序驱动攻击并不易实现。踪迹驱动攻击则是通过跟踪 AES 加密过程中每次访问的地址,进而对密钥进行分析。由于需要对攻击目标植入监视进程,这种方法缺乏灵活性。访问驱动攻击是利用访问 Cache 的时间差异,找出 AES 特定轮加密所访问的内存块,结合轮加密公式分析出密钥信息。由于访问驱动较容易实现,目前针对 AES 的 Cache 攻击主要基于这种方法进行。

针对所述基于查找表的 AES 实现所存在的问题,本文将提出一种新的查找表结构,在维持原基于查找表的 AES 实现加密速度基本不变的前提下,减少查找表所带的存储开销,同时提高访问驱动 Cache 攻击成功的难度,从而提高基于查找表的 AES 实现的安全性。

本文第 2 节简要介绍 AES 加密机制及传统基于 4 张 1kB 查找表的 AES 实现方法;第 3 节讨论访问驱动 Cache 攻击的原理;第 4 节详细描述 1-T 查找表及其轮加密公式的优化;第 5 节和第 6 节分别对 1-T 实现进行抗攻击验证及开销评估,最后是总结。

2 AES 加密机制

2.1 AES 算法分析

AES 支持 128/192/256 位密钥长度,分别对应 128/192/256 位的块大小和 10/12/14 的加密轮数 N 。图 1 所示为 AES 加密过程。

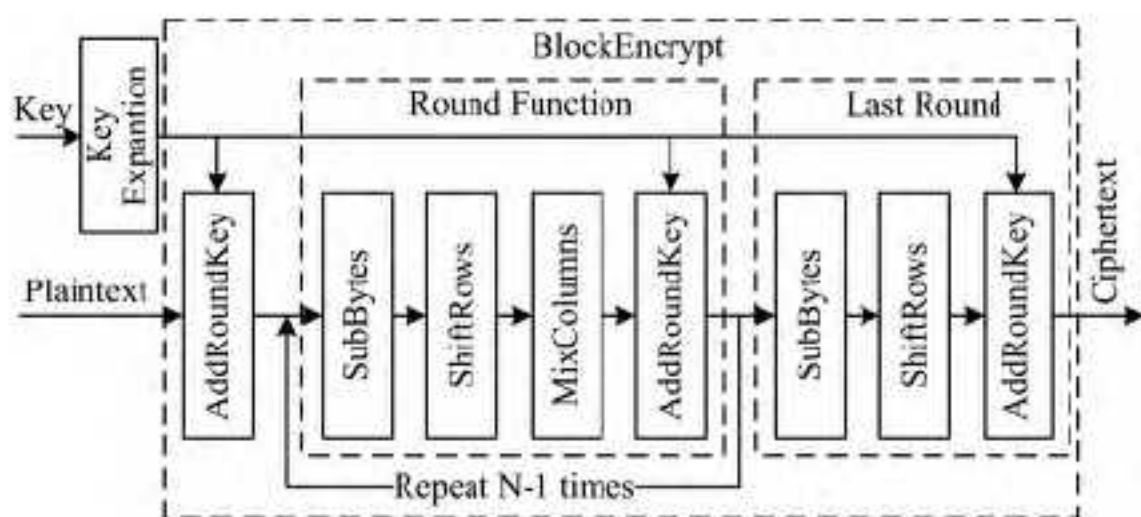


图 1 AES 加密

KeyExpansion 用于将输入密钥进行扩展得到扩展密钥 K_R ,为每一轮加密提供密钥。将明文块写成 4×4 的状态矩阵 (state) A , AES 加密操作在 A 上进行,过程如下:1) 将 A 与

密钥进行异或;2) 进行 $N-1$ 次轮加密,轮加密包含字节替换 (SubByte)、行位移 (ShiftRows)、列混合 (MixColumns),以及密钥相加 (AddRoundKey) 4 种操作;3) 最后一次轮加密,进行除列混合操作外的其他操作。

其中字节替换为非线性替换,将 A 中的字节依次替换为替换表 S_{box} 中对应的字节。行位移为线性变换,对 A 的每一列进行一定字节数的向左循环位移。列混合为线性变换,

将 A 与混淆变换矩阵 $C = \begin{bmatrix} 2 & 1 & 1 & 3 \\ 3 & 2 & 1 & 1 \\ 1 & 3 & 2 & 1 \\ 1 & 1 & 3 & 2 \end{bmatrix}$ 进行 $GF(2^8)$ 上的乘法。

密钥相加为线性变换,将 A 与 K_R 进行相加 (异或)。设

$A = \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}$, $K_R = \begin{bmatrix} kr_0 & kr_4 & kr_8 & kr_{12} \\ kr_1 & kr_5 & kr_9 & kr_{13} \\ kr_2 & kr_6 & kr_{10} & kr_{14} \\ kr_3 & kr_7 & kr_{11} & kr_{15} \end{bmatrix}$, 则轮加密函数如式(1)所示:

$$f_r(A, K_R) = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} S(a_0) & S(a_4) & S(a_8) & S(a_{12}) \\ S(a_5) & S(a_9) & S(a_{13}) & S(a_1) \\ S(a_{10}) & S(a_{14}) & S(a_2) & S(a_6) \\ S(a_{15}) & S(a_3) & S(a_7) & S(a_{11}) \end{bmatrix} \oplus \begin{bmatrix} kr_0 & kr_4 & kr_8 & kr_{12} \\ kr_1 & kr_5 & kr_9 & kr_{13} \\ kr_2 & kr_6 & kr_{10} & kr_{14} \\ kr_3 & kr_7 & kr_{11} & kr_{15} \end{bmatrix} \quad (1)$$

其中, $S(x)$ 表示字节替换操作,从 S_{box} 中查找 x 对应的字节; $\otimes$ 表示 $GF(2^8)$ 乘法操作, $\oplus$ 为异或运算。

2.2 基于查找表的 AES 加密实现

根据式(1), AES 每次轮加密的矩阵相乘中需要进行 64 次 $GF(2^8)$ 乘法操作,则整个加密过程中需要进行 $64 * (N-1)$ 次 $GF(2^8)$ 乘法操作。但由于目前处理器并不直接支持 $GF(2^8)$ 乘法,因此每次 $GF(2^8)$ 乘法需要多个时钟周期完成,而轮加密过程中的其它操作都是相对简单的访存、异或和字节移位等运算,经统计,以 $GF(2^8)$ 乘法运算为主的 MixColumns 占据了整个 AES 加密过程近 95% 的时钟周期。由于混淆矩阵中只包含 3 种元素 $\{1, 2, 3\}$, 因此可以建立 0 至 255 分别与这 3 个数字相乘的查找表,从而将每次 $GF(2^8)$ 乘法转换为查表运算,减少时钟周期开销。

设 B 为轮加密的结果,则根据式(1), B 中的每一列元素可表示如下:

$$\begin{aligned} b_0 b_1 b_2 b_3^T &= C_{2113} \otimes S(a_0) \oplus C_{3211} \otimes S(a_5) \oplus C_{1321} \otimes S(a_{10}) \\ &\quad \oplus C_{1132} \otimes S(a_{15}) \oplus K_{r0} K_{r1} K_{r2} K_{r3}^T \\ b_4 b_5 b_6 b_7^T &= C_{2113} \otimes S(a_4) \oplus C_{3211} \otimes S(a_9) \oplus C_{1321} \otimes S(a_{14}) \\ &\quad \oplus C_{1132} \otimes S(a_3) \oplus K_{r4} K_{r5} K_{r6} K_{r7}^T \\ b_8 b_9 b_{10} b_{11}^T &= C_{2113} \otimes S(a_8) \oplus C_{3211} \otimes S(a_{13}) \oplus C_{1321} \otimes S(a_2) \\ &\quad \oplus C_{1132} \otimes S(a_7) \oplus K_{r8} K_{r9} K_{r_{10}} K_{r_{11}}^T \\ b_{12} b_{13} b_{14} b_{15}^T &= C_{2113} \otimes S(a_{12}) \oplus C_{3211} \otimes S(a_1) \oplus C_{1321} \otimes S(a_6) \\ &\quad \oplus C_{1132} \otimes S(a_{11}) \oplus K_{r_{12}} K_{r_{13}} K_{r_{14}} K_{r_{15}}^T \end{aligned} \quad (2)$$

其中, C_{2113} 、 C_{3211} 、 C_{1321} 和 C_{1132} 分别表示混淆矩阵的每一行。从式(2)可以看到,每次轮加密实际上是将原状态矩阵 A 中

每个元素与固定数值相乘后再进行异或得到新状态矩阵。由此,文献[17]通过建立4张4*256字节的查找表 T_{3112} 、 T_{1123} 、 T_{1231} 和 T_{2311} (高字节在左边),分别存储与 C_{2113} 、 C_{3211} 、 C_{1321} 和 C_{1132} 进行GF(2⁸)乘法的结果,从而使得每次轮加密操作只包含16次查表操作和16次异或操作,大幅降低了轮加密的时间开销,提高了AES加密速度。本文中称该方法为4-T,其轮加密公式可以表示如下:

$$\begin{aligned} b_0 b_1 b_2 b_3^T &= S_{3112}(a_0) \oplus S_{1123}(a_5) \oplus S_{1231}(a_{10}) \oplus S_{2311}(a_{15}) \\ &\quad \oplus Kr_0 Kr_1 Kr_2 Kr_3^T \\ b_4 b_5 b_6 b_7^T &= S_{3112}(a_4) \oplus S_{1123}(a_9) \oplus S_{1231}(a_{14}) \oplus S_{2311}(a_3) \oplus \\ &\quad Kr_4 Kr_5 Kr_6 Kr_7^T \\ b_8 b_9 b_{10} b_{11}^T &= S_{3112}(a_8) \oplus S_{1123}(a_{13}) \oplus S_{1231}(a_2) \oplus S_{2311}(a_7) \\ &\quad \oplus Kr_8 Kr_9 Kr_{10} Kr_{11}^T \\ b_{12} b_{13} b_{14} b_{15}^T &= S_{3112}(a_{12}) \oplus S_{1123}(a_1) \oplus S_{1231}(a_6) \oplus \\ &\quad S_{2311}(a_{11}) \oplus Kr_{12} Kr_{13} Kr_{14} Kr_{15}^T \end{aligned} \quad (3)$$

其中, $S_{3112}(a)$ 、 $S_{1123}(a)$ 、 $S_{1231}(a)$ 和 $S_{2311}(a)$ 分别表示对 T_{3112} 、 T_{1123} 、 T_{1231} 和 T_{2311} 进行查找得到的字(4Byte)结果。

3 访问驱动 Cache 攻击

3.1 Cache 访问机制

为了提高处理器访问主存的速度,目前大部分处理器都带有高速缓存(Cache)。Cache的大小通常表示为 $Size=B \cdot W \cdot S$,其中 B 为Cache块大小, W 为Cache组的关联度, S 为 $CS_{2311}(a)$ ache组的个数。处理器访问数据 x 时,先访问Cache中 x 的地址 $\langle x \rangle$ 所对应的Cache组,即 $\langle x \rangle / B \bmod S$ 中是否包含存放该数据的数据块,如果存在,则返回数据,称为“Cache命中”;反之,则要从主存中将相应的数据块 $\langle x \rangle / B$ 读取到Cache组 $\langle x \rangle / B \bmod S$ 的某一块上,同时把数据 x 返回给处理器,称为“Cache缺失”。处理器访问存储时发生Cache命中或缺失所消耗的时钟周期数不同:在Cache命中时通常只需1—2个时钟周期;而在发生Cache缺失时,由于需要先访问主存再返回数据,通常消耗几十至上百个时钟周期。

3.2 攻击原理

如前所述,攻击者向系统输入明文,目的是获取系统中的密钥。根据AES加密流程,第一轮轮加密需要对状态 $A^0 = \{a_i \oplus k_i \mid i=0, \dots, 15\}$ 中的每个元素进行查表,其中 a_i 为明文中的元素, k_i 为输入密钥中的元素,因此第一轮加密的查表访问情况反映了 $a_i \oplus k_i$ 的信息。同时由于明文元素 a_i 已知,因此再通过简单异或计算,输入密钥 k_i 的信息也可以轻易获取。基于这种思想,文献[28]针对AES第一轮加密进行密钥攻击。通过分析AES加密过程中查找表所对应的Cache块 $\{Block_i \mid i=0, \dots, Size_{WT}/B-1\}$ (其中 $Size_{WT}$ 为查找表的大小)的访问情况,寻找未被访问的Cache块 $\{\neg Block_i\}$,根据 $\{\neg Block_i\}$ 排除候选 $a_i \oplus k_i$ 中导致访问 $\{\neg Block_i\}$ 的候选项。经过一定量的明文样本加密实验,密钥中每个字节 k_i 的高 $\log_2(256/\delta)$ (其中 δ 表示每个Cache块中包含的查找表项的项数)位即可确定,但无法确定剩下低 $\log_2 \delta$ 位。

为了确定每个字节的低 $\log_2 \delta$ 位,需要对AES的第二轮加密进行攻击。式(3)表示第一轮加密得到状态 A^1 中的 a_2^1 、 a_5^1 、 a_8^1 和 a_{15}^1 ,在第二轮加密中对这4个元素的查表访问反映了 k_i 的信息。文献[28]通过获得的 $\{\neg Block_i\}$,根据式(4)排

除候选 k_i 中导致访问 $\{\neg Block_i\}$ 的候选项。经过一定量的明文样本加密实验后,每个 k_i 的低 $\log_2 \delta$ 位即可确定。

$$\begin{aligned} a_2^1 &= S(a_0 \oplus Kr_0) \oplus S(a_5 \oplus Kr_5) \oplus 2 \otimes S(a_{10} \oplus Kr_{10}) \oplus \\ &\quad 3 \otimes S(a_{15} \oplus Kr_{15}) \oplus S(Kr_{15}) \oplus Kr_2 \\ a_5^1 &= S(a_4 \oplus Kr_4) \oplus 2 \otimes S(a_9 \oplus Kr_9) \oplus 3 \otimes S(a_{14} \oplus \\ &\quad Kr_{14}) \oplus S(a_3 \oplus Kr_3) \oplus S(Kr_{14}) \oplus Kr_1 \oplus Kr_5 \\ a_8^1 &= 2 \otimes S(a_8 \oplus Kr_8) \oplus 3 \otimes S(a_{13} \oplus Kr_{13}) \oplus S(a_2 \oplus \\ &\quad Kr_2) \oplus S(a_7 \oplus Kr_7) \oplus S(Kr_{13}) \oplus Kr_0 \oplus Kr_4 \oplus \\ &\quad Kr_8 \oplus 1 \\ a_{15}^1 &= 3 \otimes S(a_{12} \oplus Kr_{12}) \oplus S(a_1 \oplus Kr_1) \oplus S(a_6 \oplus Kr_6) \\ &\quad \oplus 2 \otimes S(a_{11} \oplus Kr_{11}) \oplus S(Kr_{12}) \oplus Kr_{15} \oplus Kr_3 \oplus \\ &\quad Kr_7 \oplus Kr_{11}^{[28]} \end{aligned} \quad (4)$$

3.3 攻击复杂度

对于密钥攻击而言,如果攻击所需的样本量越大,即实验的次数越多,则攻击所耗费的时间越长。因此提高攻击所需的样本数就意味着提高AES加密算法自身的安全性。

在一次AES加密中,假设每次查表相互独立,则 $\neg Block_i$ 出现的概率为:

$$\neg p = (1 - \delta/256)^{m \cdot N} \quad (5)$$

其中, m 为每轮加密该查找表被访问的次数。由于密钥长度确定条件下加密轮数 N 为定值,因此 $\neg p$ 只与 δ 和 m 相关, δ 和 m 越大, $\neg p$ 越小; $\neg p$ 越小,使 $\neg Block_i$ 出现所需的实验样本越多。式(6)表示了密钥中第 i 个字节的候选 k_i 的数量 $N_{k_i}^A$ 与样本量 N_s 之间的关系:

$$N_{k_i}^A = 256 \cdot (1 - \neg p)^{N_s} \quad (6)$$

从公式可见,随着 N_s 的增加, $N_{k_i}^A$ 最终将逼近1,即获得正确的 k_i 。换句话说,攻击密钥所需样本量 N_s 与 $\log(1 - \neg p)$ 成反比[28]。即 δ 和 m 越大,破解密钥所需的 N_s 越大。而 δ 与Cache的块大小 B 和查找表的结构有关, m 与查找表的结构有关,因此在给定Cache设置的条件下,改变查找表的结构使得 δ 和 m 增大可以提高 N_s ,亦即提高AES算法的安全性。

4 抗访问驱动 Cache 攻击的快速 AES 加密方法

4.1 抗访问驱动 Cache 攻击的查找表

首先,观察4-T中4张1kB查找表,我们发现它们相同下标的表项间可以通过相互循环移位获得。例如 T_{1123} 可以通过对 T_{3112} 向左循环移动一个字节得到。所以,只需要保存其中一张表(如 T_{3112})再通过循环移位的方式获得其它3张表。由此查找表的存储开销可以由4kB降为1kB,而代价是需要进行额外的循环位移以获得另外3张查找表的结果。与此同时,在每轮加密中对这张查找表的访问次数增加为16次,而在4-T中对每张表的访问次数为4次。由此, $\neg Block_i$ 出现的概率 $\neg p$ 由4-T的 $(1 - \delta/256)^{40}$ 降低为 $(1 - \delta/256)^{160}$ 。

其次,观察上述所保留的一张表 T_{3112} ,我们发现其每个表项均由 $S(a) \otimes 3$ 、 $S(a) \otimes 1$ 、 $S(a) \otimes 1$ 和 $S(a) \otimes 2$ 排列组成,其中 $S(a) \otimes 1$ 出现了两次,为冗余项。而且,在GF(28)乘法中,有: $S(a) \otimes 3 = S(a) \otimes 2 \oplus S(a) \otimes 1$,这意味着,只需要存储 $S(a) \otimes 1$ 和 $S(a) \otimes 2$,就可以通过简单异或和组合运算恢复出4-T中每张表的表项,且查找表所占的存储空间可以降为512B。4种表结构的存储大小和访问时 $\neg Block_i$ 出现的概

率 $-p$ 的对比如表1所列。

表1 不同表结构对比

	4 * 1kB Table	1 * 1kB Table	1 * 512B Table
Size(kB)	4	1	0.5
B=16B	0.5326	0.0805	6.221×10^{-3}
B=32B	0.2808	6.221×10^{-3}	3.276×10^{-5}
B=64B	0.0757	3.276×10^{-5}	5.264×10^{-10}
B=128B	4.790×10^{-3}	5.264×10^{-10}	1.023×10^{-20}

对比3种表结构,1*512B查找表不仅所占的存储空间比其他两种表结构要小,同时在相同的Cache块大小情况下,其 $-Block_i$ 出现的概率 $-p$ 也比其他两种表结构要小得多。在Cache块大小为16B时,1*512B的 $-p$ 为 6.221×10^{-3} ,为4*1kB的约1%。随着Cache块大小的增加,它们的比例急剧下降;在Cache块大小为128B时,1*512B的 $-p$ 为 1.023×10^{-20} ,仅仅为4*1kB的 2.14×10^{-16} 。这意味着在给定Cache块大小下,1*512B查找表需要更多测试样本才会出现 $-Block_i$,使得其访问驱动Cache攻击的能力远高于4*1kB,并且Cache块越大,1*512B查找表抗访问驱动Cache攻击的优势越明显。

4.2 1-T方法

本节通过对上述1*512B查找表结构做进一步优化,得到1-T方法。

对于1*512B查找表,其表项大小为2B,有两种排列方式: $S_1(a)S_2(a)$ 和 $S_2(a)S_1(a)$ 。不论采用哪种排列方式,在每轮轮加密中对状态中的每一个元素都只需进行一次查表,和4-T一致。由于查表得到的是半字结果(2B),为了获得式(3)中4种字结果: $S_{3112}(a)$ 、 $S_{1123}(a)$ 、 $S_{1231}(a)$ 和 $S_{2311}(a)$,需要先将半字结果拆分得到单字结果 $S_1(a)$ 和 $S_2(a)$,再通过移位操作以及或操作将单字结果合并为4-T中的字结果。但观察4-T中的4种字结果,它们在字中均包含了 $S_1(a)S_2(a)$ 或 $S_2(a)S_1(a)$ 相邻的情况,分别如图2和图3所示。对于出现 $S_1(a)S_2(a)$ 或 $S_2(a)S_1(a)$ 的地方,可以将查找到的半字结果直接通过1次移位放入字中,从而可以节省部分操作。

	Byte 3	Byte 2	Byte 1	Byte 0
$S_{3112}(a)$	$S_1(a) \oplus S_2(a)$	$S_1(a)$	$\ll S_1(a)$	$\gg S_2(a)$
$S_{1123}(a)$	$S_1(a)$	$\ll S_1(a)$	$\gg S_2(a)$	$S_1(a) \oplus S_2(a)$
$S_{1231}(a)$	$\ll S_1(a)$	$\gg S_2(a)$	$S_1(a) \oplus S_2(a)$	$S_1(a)$
$S_{2311}(a)$	$S_2(a)$	$S_1(a) \oplus S_2(a)$	$S_1(a)$	$S_1(a)$

图2 字中包含 $S_1(a)S_2(a)$ 排列的情况

	Byte 3	Byte 2	Byte 1	Byte 0
$S_{3112}(a)$	$S_1(a) \oplus S_2(a)$	$S_1(a)$	$S_1(a)$	$S_2(a)$
$S_{1123}(a)$	$S_1(a)$	$S_1(a)$	$\gg S_2(a)$	$S_1(a) \oplus S_2(a)$
$S_{1231}(a)$	$S_1(a)$	$\gg S_2(a)$	$S_1(a) \oplus S_2(a)$	$S_1(a)$
$S_{2311}(a)$	$\gg S_2(a)$	$S_1(a) \oplus S_2(a)$	$S_1(a)$	$S_1(a)$

图3 字中包含 $S_2(a)S_1(a)$ 排列的情况

直观上看,这些字中只出现了3处 $S_1(a)S_2(a)$,如图2中椭圆阴影所示,而没有出现 $S_2(a)S_1(a)$ 。但由于 $S_3(a) = S_2(a) \oplus S_1(a)$,因此 $S_3(a)$ 与它左边的 $S_2(a)$ 以及右边的 $S_1(a)$ 分别构成一个 $S_2(a)S_1(a)$ 。如图3的椭圆阴影所示,在这些字中出现了6处 $S_2(a)S_1(a)$ 。这意味着,查找表采用的 $S_2(a)S_1(a)$ 排列方式可以节省更多由组字产生的操作。

以 $S_{21}(a)$ 表示 $S_2(a)S_1(a)$ 的排列,构建存储 $S_{21}(a)$ 的查找表

T_{21} 。计算中所用的 $S_1(a)$ 和 $S_2(a)$ 可以通过以下几个公式得到:

$$S_1(a) = S_{21}(a) \& 0xff \quad (7)$$

$$S_2(a) = S_{21}(a) \gg 8$$

且利用变量表示范围溢出,有:

$$S_1(a) \ll 24 = S_{21}(a) \ll 24 \quad (8)$$

将此方法称为1-T,其轮加密公式如下所示:

$$b_3 b_2 b_1 b_0^T = ((S_{21}(a_0) \ll 24) \oplus (S_{21}(a_0) \ll 16) | ((S_{21}(a_0) \& 0xff) \ll 8) | (S_{21}(a_0) \gg 8)) \oplus ((S_{21}(a_5) \ll 24) | ((S_{21}(a_5) \& 0xff) \ll 16) | (S_{21}(a_5) \oplus (S_{21}(a_5) \gg 8))) \oplus ((S_{21}(a_{10}) \ll 24) | (S_{21}(a_{10}) \ll 8) \oplus S_{21}(a_{10})) \oplus ((S_{21}(a_{15}) \ll 16) \oplus (S_{21}(a_{15}) \ll 8) | (S_{21}(a_{15}) \& 0xff)) \oplus Kr_3 Kr_2 Kr_1 Kr_0^T \quad (9)$$

式(9)中只给出 $b_3 b_2 b_1 b_0$ 计算方法,其余列的计算可以此类推。对比4-T和1-T的轮加密公式,它们所包含的操作可以分为5类:查表(LUT)、将 $S_1(a)$ 和 $S_2(a)$ 通过异或($\oplus$)计算 $S_3(a)$ (Gen-S3)、通过与($\&$)和右移($\gg$)提取字节(Gen-Byte)、通过或($|$)和左移($\ll$)将字节合并成字(Gen-Word),及相加($\oplus$, ADD)。它们的统计如表2所列。

表2 轮加密操作对比

	LUT	Gen-S3	Gen-Byte	Gen-Word	ADD
4-T	16	0	0	0	16
1-T	16	16	20	60	16

从表中可以看到,4-T由于存储了 $S_{3112}(a)$ 、 $S_{1123}(a)$ 、 $S_{1231}(a)$ 和 $S_{2311}(a)$,其轮加密公式中只包含LUT和ADD两种操作。而1-T由于需要进行额外的Gen-S3、Gen-Byte以及Gen-Word等操作,其操作数量相比为4-T的4倍。

5 抗访问驱动Cache攻击能力验证

根据式(6),为了将密钥的候选字节从256个降为1,4-T和1-T在不同Cache块大小时所需的攻击样本数量理论值如图4所示。从图中可以看到,在不同Cache块大小下,攻击1-T所需的样本量都比4-T所需的样本量要高很多。在Cache块大小为16字节时,攻击1-T所需的样本量为4-T的100倍以上。随着Cache块大小的增加,它们之间比例急剧上升,在Cache块大小为128字节时,攻击1-T需要1021量级的样本数,而攻击4-T则只需104量级的样本数。

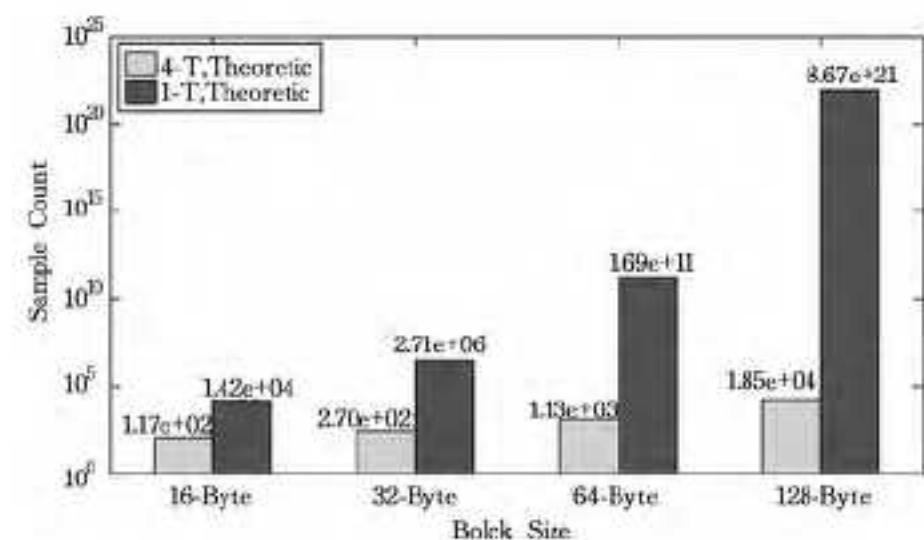


图4 攻击样本理论值对比

通过实验的方法评估1-T方法的抗访问驱动Cache攻击能力。评估流程如图5所示。其目的是获取每次AES加密时Cache的访问信息,从而得到查找表的访问情况。实际攻击时,可由文献[28,29]所提方法获取Cache块的访问信息。由于本文重点在于验证1-T的抗攻击能力,因此通过直接记录每次加密时所访问查找表的索引,直接获取Cache块的访问信息。

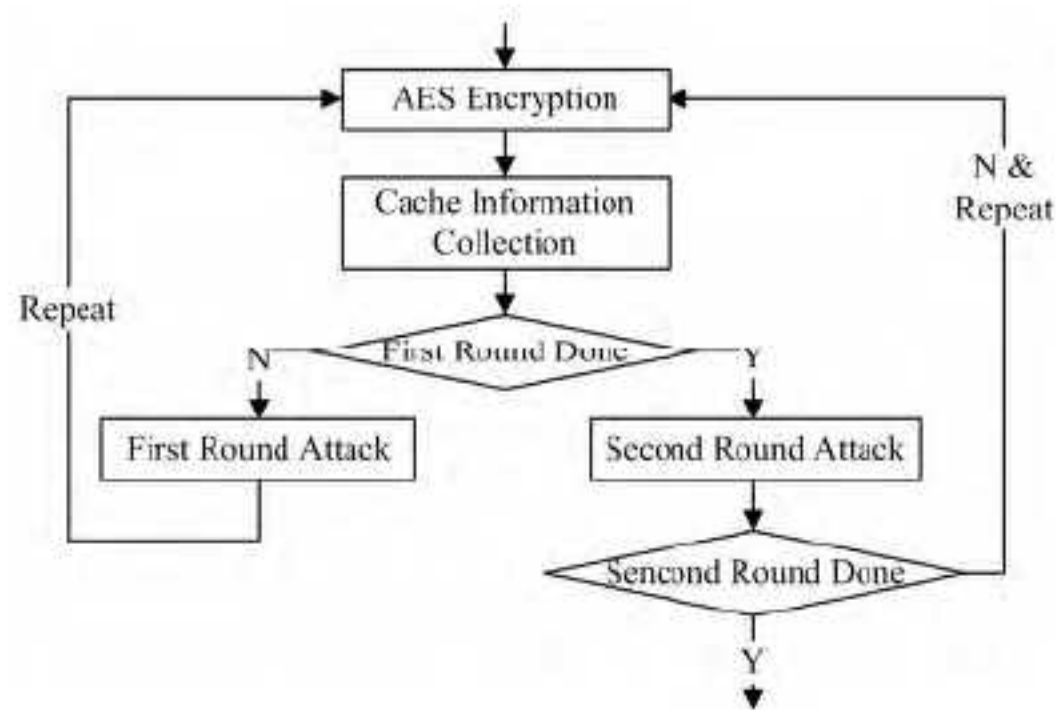
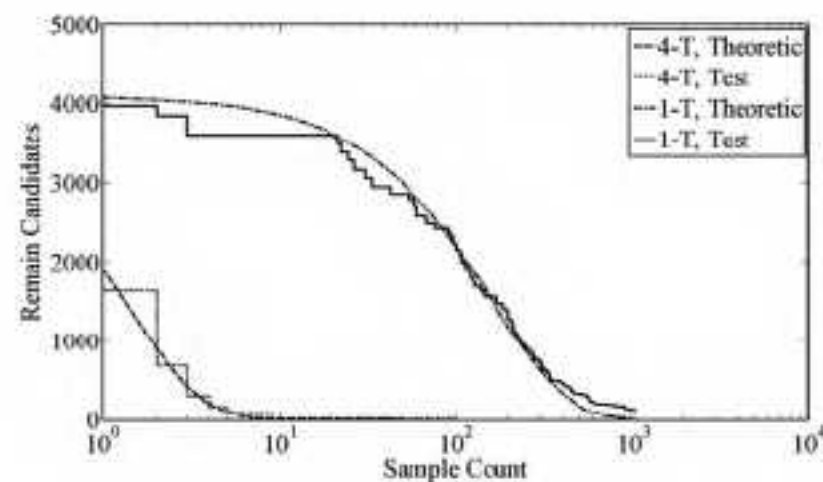
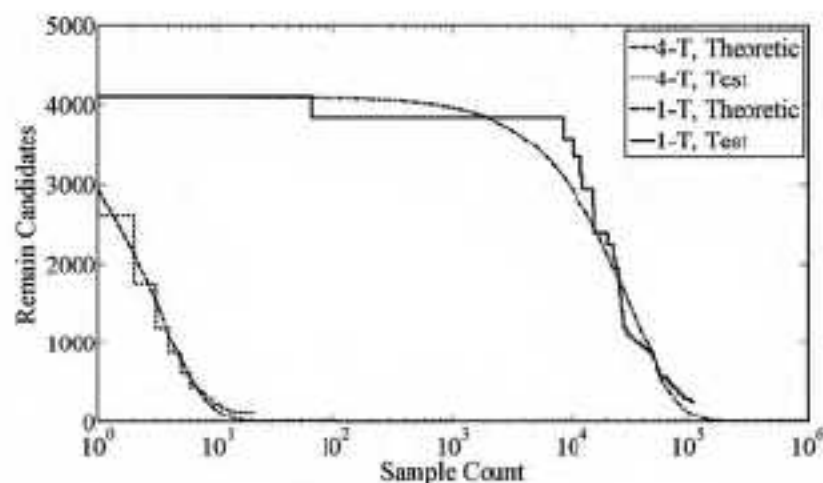


图5 Cache 攻击流程

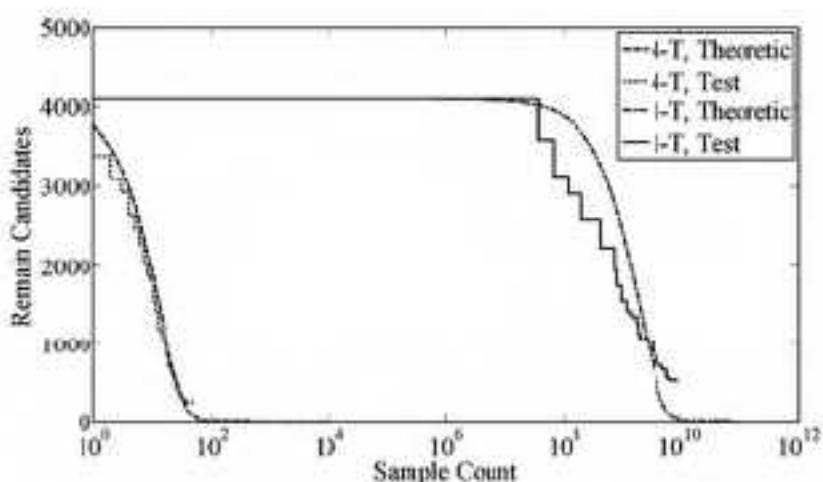
图6是对4-T和1-T进行第一轮实际攻击测试时候选密钥字节数量随攻击样本量的变化图。从图中可以看到,实际攻击测试的曲线基本上与式(6)的理论曲线拟合,为阶梯状变化。这是因为 $-Block_i$ 每测试一定数量的攻击样本后才出现一次,而每次 $-Block_i$ 的出现可以排除掉 δ 整数倍的密钥候选值。同时,可以观察到在实际攻击测试中,剩余的密钥候选字节数量在随样本量下降到一定量 $(16 \cdot \delta)$ 之后,不再发生变化,与前文的攻击分析一致。



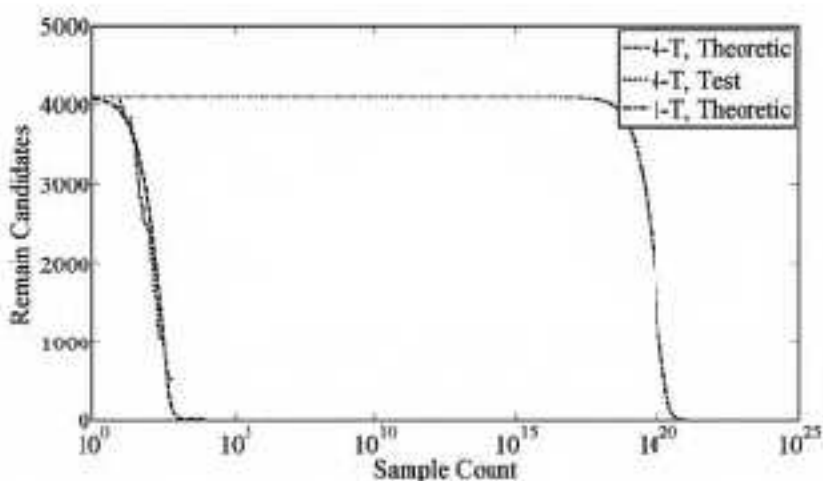
(a) $B=16B$



(b) $B=32B$



(c) $B=64B$



(d) $B=128B$

图6 第一轮攻击密钥候选字节数量随攻击样本变化

由于在Cache块大小为128字节时,攻击1-T出现

的 $-Block_i$ 概率 p 已经达到 10^{-20} 量级,即需要 10^{20} 量级的攻击样本才有可能出现一个 $-Block_i$,这样的量级已经难以在普通计算机上进行测试,因此不再对1-T在此Cache设置上进行抗攻击验证。

根据前文分析,为了确定密钥字节的低 δ 位,对4-T和1-T进行第二轮攻击测试。图7显示了对4-T和1-T完成第二轮实际攻击测试所需攻击样本数量。和图4对比,各个Cache配置下对4-T和1-T完成第二轮攻击所需的攻击样本数量与公式所给出的攻击所需样本理论值小约一个数量级。其原因主要有:1)式(6)给出的理论值仅考虑第一轮攻击的情况,其排除的是密钥中每个字节的候选值,而第二轮攻击则是对整个候选密钥(16个字节组合)进行排除,因此式(6)对其不适用;2)为了减少搜索 $-Block_i$ 的时间,本测试在第二轮攻击中复用了第一轮攻击出现 $-Block_i$ 的样本。尽管和理论样本量有差距,测试的结果仍然显示了在不同Cache配置条件下,攻击1-T比攻击4-T所需样本数至少要多100倍。

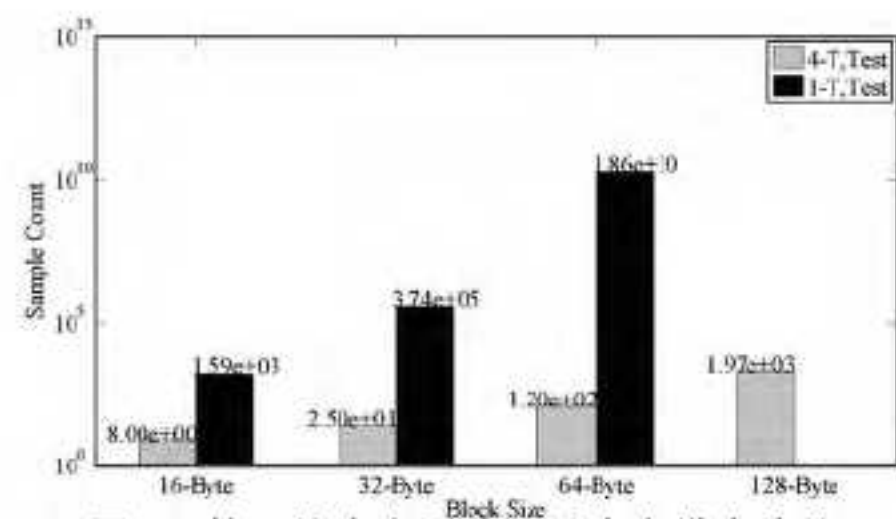


图7 第二轮攻击测试所需攻击样本比较

6 开销评估

本节将对1-T进行实现开销的评估。评估环境为目前常用于无线传感器网络环境的ARM CortexM3处理器STM32F103RE,工作主频为72MHz。为了得到更客观的评估,评估将与GF实现、4-T实现,以及采用射频芯片中AES硬件加速器的实现(HW)进行横向对比。HW实验中射频芯片为AT86RF231ZU,处理器芯片为STM32F103RE。评估指标包含存储开销和时间开销。

6.1 存储开销

对比4种AES实现的存储开销(见图8),HW由于将大部分计算都放在硬件上实现,其代码开销主要在SPI通信上,因此其代码开销和数据开销均最小。由于1-T的轮加密函数较为复杂,因此其代码开销在4种实现中最大,但是也仅比最小的HW大29.3%。而在数据存储上,4-T的4张1kB查找表所占数据存储达到4540字节,GF仅存储一张256字节的S-box,数据存储只有444字节;1-T存储的是一张512字节的查找表,所以其数据存储开销也较小,比GF大57.6%,但只有4-T的15.4%。

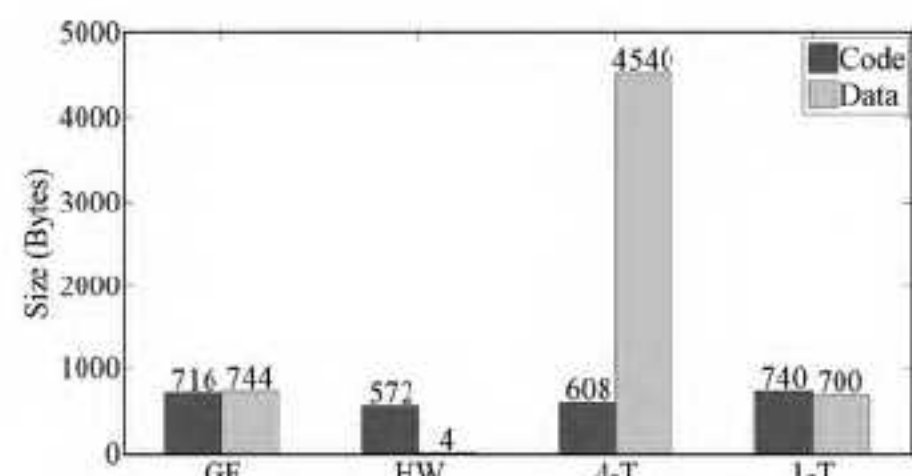


图8 存储开销对比

6.2 时间开销

图9为4种实现方式对一个明文块进行加密所需计算时

间的对比。从图中可以看到,在 ARM 平台上,虽然 HW 采用 AES 加速器进行加密,但是其加密时间却比 4-T 和 1-T 长两倍以上。这是由于射频的加速器与处理器间需要通过 SPI 进行数据传输,额外的通信时间延长了 HW 的加密时间。而 GF 由于需要进行大量复杂的 GF(28) 乘法,因此不论在哪个平台上,其加密时间远远超出其它 3 种实现方法。4-T 由于轮加密函数较为简单,因此其加密时间最短,而 1-T 由于需要进行字节提取和字合并操作,轮加密相比 4-T 复杂,从而导致其加密时间比 4-T 长 43.5%。

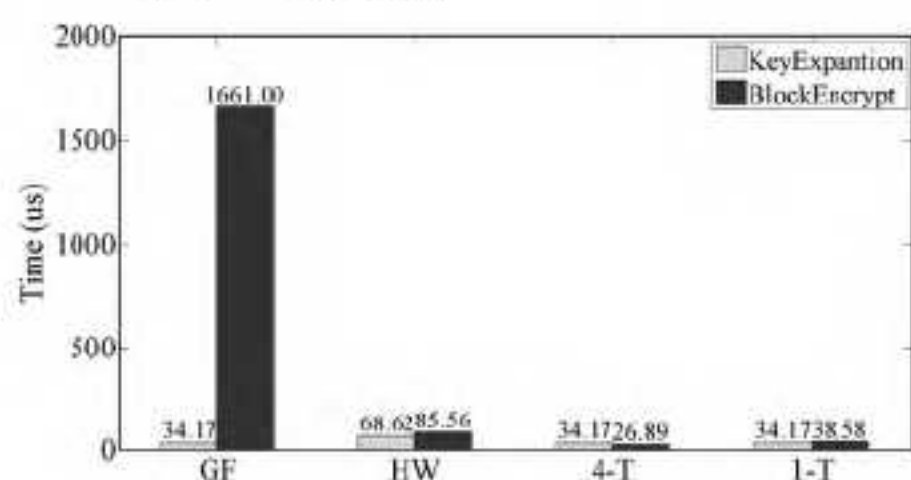


图 9 执行时间开销对比

注意到,1-T 与 4-T 之间的差距在实际测试中与表 2 所示的轮加密公式对应的操作数总和不成等比例关系。原因主要有以下几点:

- 1) 表 2 所列的操作数只考虑了轮加密公式中的操作,而对在每轮加密中的状态和密钥的读取、新状态的写回,以及第一轮加密和最后一轮加密操作,都没有进行统计;
- 2) 代码采用函数进行封装,函数调用所产生的开销也没有进行考虑;
- 3) 访存指令比其它算数指令的时钟开销大,并且在不同指令集上也不一致;
- 4) 移位指令在 ARM 指令集中可以作为算数指令的后缀,使得移位不占用单独指令周期,而 1-T 的轮加密公式包含了大量移位操作。

结束语 面向资源受限的无线传感器网络,本文提出一种优化的 AES 加密实现方法 1-T。1-T 基于一张 512 字节的查找表,并对查找表结构和轮加密公式进行了优化。实验显示,相比采用传统 4 张 1kB 查找表的方法 4-T,1-T 的存储开销下降了 72%,并且抗访问驱动 Cache 攻击的能力也得到了大幅提高。而在时间开销上,1-T 加密一个明文块所需的时间仅比 4-T 高 43.47%,但仅是采用射频上 AES 加速器方法的 38.55%,很好地满足了无线传感器网络中低开销和高实时性需求。

参考文献

- [1] LAN/MAN Standards Committee. Part 15. 4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)[S]. IEEE Computer Society, 2007
- [2] An ISA Standard Wireless systems for Industrial automation [C]// Process Control and Related Applications. ISA Std. ISA-100. 11a-2009, 2009
- [3] 莫路锋,毛方杰,等. 基于感知数据的无线传感网被动诊断方法[J]. 北京邮电大学学报, 2013, 36(1): 101-104
- [4] 王涛春,秦小麟,等. 两层无线传感器网络中隐私保护的范围查询[J]. 北京邮电大学学报, 2014, 37(2): 104-108
- [5] 周才学. 基于证书的签名方案的分析与改进[J]. 北京邮电大学学报, 2013, 36(6): 98-101

- [6] Daemen J, Rijmen V. AES proposal: Rijndael[C] // First Advanced Encryption Standard(AES) Conference. 1998
- [7] Schaumont P R, Kuo H, Verbauwhede I M. Unlocking the design secrets of a 2.29 Gb/s Rijndael processor[C] // Proceedings 39th Design Automation Conference, 2002. IEEE, 2002: 634-639
- [8] Rahimunnisa K, Karthigaikumar P, Kirubavathy J, et al. A 0.13- μ m implementation of 5 Gb/s and 3-mW folded parallel architecture for AES algorithm[J]. International Journal of Electronics, 2013(ahead-of-print): 1-12
- [9] Morioka S, Satoh A. A 10-Gbps full-AES crypto design with a twisted BDD S-Box architecture[J]. IEEE Transactions on VLSI Systems, 2004, 12(7): 686-691
- [10] Chang J K T, Liu C, Gaudiot J L. Hardware Acceleration for Cryptography Algorithms by Hotspot Detection[M] // Grid and Pervasive Computing. Springer Berlin Heidelberg, 2013: 472-481
- [11] Nguyen K, Lanante L, Nagao Y, et al. Implementation of 2.6 Gbps super-high speed AES-CCM security protocol for IEEE 802.11i[C] // 2013 13th International Symposium on Communications and Information Technologies(ISCIT). IEEE, 2013: 669-673
- [12] Xu Leslie. Secure the Enterprise with Intel[®] AES-NI: White Paper[OL]. <http://www.intel.cn/content/www/cn/zh/enterprise-security/enterprise-security-aes-ni-white-paper.html>
- [13] Lee R B, Chen Y Y. Processor accelerator for AES[C] // 2010 IEEE 8th Symposium on Application Specific Processors (SASP). IEEE, 2010: 16-21
- [14] Daemen J, Rijmen V. Resistance against implementation attacks: A comparative study of the AES proposals[C] // The Second AES Candidate Conference. 1999: 122-132
- [15] Yumbul K, Sava s E, Kocaba s Ö, et al. Design and implementation of a versatile cryptographic unit for risc processors[J]. Security and Communication Networks, 2014, 7(1): 36-52
- [16] Bertoni G, Breveglieri L, Fragneto P, et al. Efficient software implementation of AES on 32-bit platforms[M] // Cryptographic Hardware and Embedded Systems-CHES 2002. Springer Berlin Heidelberg, 2003: 159-171
- [17] Gladman B. A Specification for Rijndael, the AES Algorithm [OL]. <http://fp.gladman.plus.com>, May 2002
- [18] Atasu K, Breveglieri L, Macchetti M. Efficient AES implementations for ARM based platforms[C] // Proceedings of the 2004 ACM symposium on Applied computing. ACM, 2004: 841-845
- [19] Liu B, Baas B M. Parallel AES encryption engines for many-core processor arrays[J]. IEEE Transactions on Computers, 2013, 62(3): 536-547
- [20] Viega J, Messier M, Chandra P. Network Security with OpenSSL: Cryptography for Secure Communications [M]. O'Reilly Media, Inc., 2002
- [21] Bernstein D J. Cache-timing attacks on AES[OL]. 2005. <http://cr.yp.to/papers.html\#cachetiming>
- [22] Bonneau J, Mironov I. Cache-collision timing attacks against AES[M] // Cryptographic Hardware and Embedded Systems-CHES 2006. Springer Berlin Heidelberg, 2006: 201-215
- [23] Aclımez O, Schindler W, Koç Ç K. Cache based remote timing attack on the AES[M] // Topics in Cryptology-CT-RSA 2007. Springer Berlin Heidelberg, 2006: 271-286
- [24] 王韬,赵新杰,郭世泽,等. 针对 AES 的 Cache 计时模板攻击研究[J]. 计算机学报, 2012, 35(2): 325-341

(下转第 434 页)

7.3 多模式 BF-CAM 分析

由于网络流量中恶意流量比例 p_t 决定了系统工作的检测速度及吞吐性能,因此,实验测试了不同 p_t 下得到系统的吞吐性能。如图 13 所示,其中匹配模式最大长度即为 $L_{\max} - L_{\min}$ 。

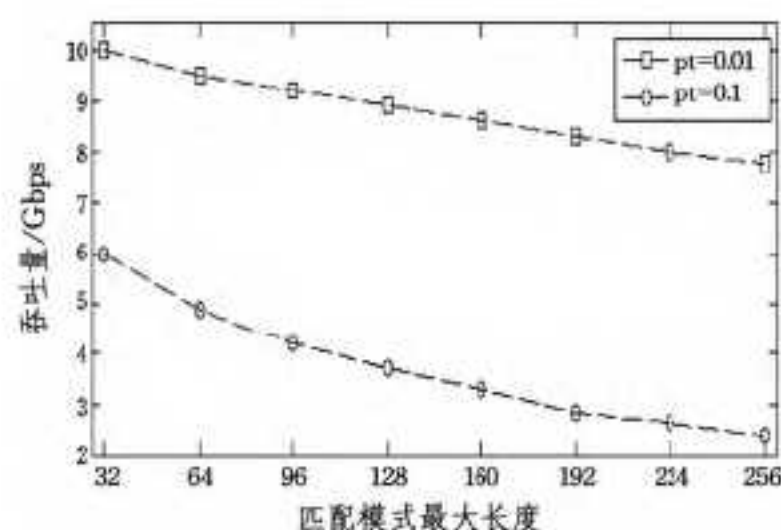


图 13 多模式的 BF-CAM 性能分析

通过实验分析得到, BF-CAM 在 p_t 为 1% 时, 能够达到 5Gbps 以上的检测速率, 当 p_t 为 10% 时, 系统能够达到 1Gbps 以上的检测速率。另外, BF-CAM 的硬件资源开销较小, 可以通过增加并行硬件资源进一步提高系统的性能。

结束语 随着计算机网络的高速发展, 网络和信息安全对国家安全、社会稳定的影响越来越大, 而当前网络中的安全问题日趋严重, 各种网络攻击已经给网络安全造成了极大的威胁。本文在研究基于 Bloom Filter 模式匹配引擎的基础上, 提出了采用 Bloom Filter 和 CAM 相结合的设计思想, 排除了 Bloom Filter 的假阳性, 实现精确匹配以及易于元素的删除。由于 Bloom Filter 匹配速度极快, 因此本算法能够减轻系统的负载, 大大提高了系统处理速度。在 Xilinx 系列 Virtex5 型号的 FPGA 器件上实验性能分析可得, 系统的吞吐率能够达到 Gbps, 达到了较高的吞吐性能。且相对于单级的 CAM 查找, 本文提出的 BF-CAM 算法能够表达庞大的数据集以及提高查找效率。结论表明, 在假阳率为 0.01 时, 整个系统的资源占用减少 5 倍以上, 功耗可以降低 10 倍以上。下一步工作将根据本文的分析, 按照可并行点做进一步的优化, 使系统达到更高的性能, 满足高速网络的需求。

参考文献

- [1] Ruijie Network. DPI Technical Papers[OL]. <http://wenku.baidu.com/view/aa73eac66137ee06ef91879.html>. 2010. 3
- [2] Tuck N, Sherwood T, Calder B, et al. Deterministic memory-efficient string matching algorithms for intrusion detection[C]// Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004). IEEE, 2004, 4: 2628-2639
- [3] Introduction to Snort[OL]. <http://www.snort.org/docs/>

- [4] Song H, Dharmapurikar S, Turner J, et al. Fast hash table look-up using extended bloom filter: an aid to network processing[J]. ACM SIGCOMM Computer Communication Review, ACM, 2005, 35(4): 181-192
- [5] Mitzenmacher. Compressed bloom filters [J]. IEEE/ACM Transactions on Networking (TON), 2002, 10(5): 604-612
- [6] Peir J K, Lai S C, Lu S L, et al. Bloom filtering cache misses for accurate data speculation and prefetching[C]// Proceedings of the 16th International Conference on Supercomputing. ACM, 2002: 189-198
- [7] Bloom B H. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426
- [8] Mitzenmacher M. Compressed bloom filters [J]. IEEE/ACM Transactions on Networking, 2002, 10(5): 604-612
- [9] 谢鲲. 布鲁姆过滤器查询算法及其应用研究 [D]. 长沙: 湖南大学, 2007
- [10] Tan J S, Kuang Z, Yang G. Bloom filter based frequent patterns mining over data streams [C]// 2012 International Conference on Graphic and Image Processing. International Society for Optics and Photonics, 2013: 87685V-87685V-7
- [11] 徐欣, 李宗华, 卢启中, 等. 基于 FPGA 的内容可寻址存储器研究设计与应用[J]. 国防科技大学学报, 2001, 23(5): 69-73
- [12] Guo R, Delgado-Frias J G. IP Routing table compaction and sampling schemes to enhance TCAM cache performance[J]. Journal of Systems Architecture, 2009, 55(1): 61-69
- [13] Kim Y D, Ahn H S, Kim S, et al. A high-speed range-matching TCAM for storage-efficient packet classification [J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2009, 56(6): 1221-1230
- [14] Chen Y, Oguntuyinbo O. Power efficient packet classification using cascaded bloom filter and off-the-shelf ternary CAM for WDM networks[J]. Computer Communications, 2009, 32(2): 349-356
- [15] Kanizo Y, Hay D, Keslassy I. Access-efficient balanced Bloom filters[J]. Computer Communications, 2013, 36(4): 373-385
- [16] AbuHmed T, Mohaisen A, Nyang D H. A survey on deep packet inspection for intrusion detection systems[J]. Magazine of Korea Telecommunication Society, 2007, 24(11): 25-36
- [17] McLaughlin K, O'Connor N, Sezer S. Exploring CAM design for network processing using FPGA technology [C]// International Conference on Internet and Web Applications and Services/ Advanced International Conference on Telecommunications, 2006 (AICT-ICIW'06). IEEE, 2006: 84-84
- [18] 田耘, 徐文波. Xilinx FPGA 开发实用教程[M]. 北京: 清华大学出版社, 2008

(上接第 407 页)

- [25] Bertoni G, Zaccaria V, Breveglieri L, et al. AES power attack based on induced cache miss and countermeasure[C]// International Conference on Information Technology, Coding and Computing, 2005 (ITCC 2005). IEEE, 2005, 1: 586-591
- [26] Acliçmez O, Koç Ç K. Trace-driven cache attacks on AES (short paper)[M]// Information and Communications Security. Springer Berlin Heidelberg, 2006: 112-121
- [27] Gallais J F, Kizhvatov I, Tunstall M. Improved trace-driven cache-collision attacks against embedded AES implementations

[M]// Information Security Applications. Springer Berlin Heidelberg, 2011: 243-257

- [28] Osvik D A, Shamir A, Tromer E. Cache attacks and countermeasures: the case of AES[M]// Topics in Cryptology-CT-RSA 2006. Springer Berlin Heidelberg, 2006: 1-20
- [29] Tromer E, Osvik D A, Shamir A. Efficient cache attacks on AES, and countermeasures[J]. Journal of Cryptology, 2010, 23(1): 37-71
- [30] 赵新杰, 王韬, 郭世泽, 等. AES 访问驱动 Cache 计时攻击[J]. 软件学报, 2011, 22(3): 572-591