# 一种改进的针对滑动窗口模幂运算实现的密码数据Cache计时攻击

周 平 寇应展 王 韬 赵新杰 刘会英
(军械工程学院计算机工程系 石家庄 050003)

**摘 要** RSA、DSA等公钥密码大都基于"滑动窗口"算法实现模幂运算,其运算过程中进行的Cache访问会产生旁路信息泄漏并用于密钥破解,基于Cache访问泄漏的幂指数分析算法是提高攻击效率的关键。通过分析现有攻击的不足,进一步分析了预计算乘法因子到Cache的映射规律,提出了一种基于窗口值判定的幂指数分析改进算法;以基本模幂运算为例,通过实际攻击实验验证改进算法的效率,结果表明改进算法可恢复出60%的幂指数位,优于前人最好工作的47%;最后以RSA和DSA为例,给出了改进算法对密钥分析的影响。

**关键词** 滑动窗口,模幂运算,RSA,DSA,Cache计时攻击
**中图法分类号** TP393.08 **文献标识码** A

## Improved Data-Cache Timing Attack on Cryptography Adopting Sliding Window Method for Modular Exponentiation

ZHOU Ping KOU Ying-zhan WANG Tao ZHAO Xin-jie LIU Hui-ying
(Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang 050003, China)

**Abstract** Public-key cryptography, such as RSA and DSA, adopt sliding window method for modular exponentiation, from which side-channel information can be leaked while accessing Cache during execution, thus private key can be decrypted. Exponent analysis algorithm is the key point to improve the efficiency of the attack. By analyzing the shortcoming of previous work, this paper farther analyzed the relationship between Cache-access trace and precomputed multipliers and proposed an improved exponent analysis algorithm based on window-value identifying. Experiments were made to prove the efficiency the improved algorithm and results showed that the improved algorithm was able to recover 60% exponential bits, which is better than the previous result 47%. In the end, the application of the improved algorithm was showed on RSA and DSA.

**Keywords** Sliding window algorithm, Modular exponentiation, RSA, DSA, Cache timing attack

## 1 引言

1996年,Kocher[1]提出了针对密码实现的旁路攻击思想。旁路攻击是指利用密码系统在执行过程中泄漏的额外物理效应信息(如功耗[2]、电磁辐射[3]、执行时间[4]等)来进行密钥分析的一种方式。数据Cache计时攻击[5-23]是旁路攻击的一种,主要利用密码实现过程中CPU向Cache读取数据时发生的Cache命中和Cache失效导致时间差异,并结合算法设计推测密码内部执行过程,最终实现密钥破解。Cache计时攻击已经被广泛应用于分组密码[5-15]、公钥密码[16-20]和流密码[21-23]。

模幂运算是当前几种主流公钥密码中最核心、最重要的运算之一,其安全性直接关系到整个密码方案的安全性。对于RSA加密和签名方案,幂指数即为私钥;对于DSA、ElGamal等签名方案,可以通过幂指数间接分析出签名者私钥。而模幂运算执行时需大量消耗时间和内存资源,同其它运算步骤相比,泄露出的旁路信息更多。现有针对RSA、DSA、ElGamal等密码的Cache计时攻击本质上都是针对算法中模幂运算过程的旁路泄漏开展的,其通过分析旁路泄露推测部分幂指数信息,并结合算法结构分析获取密钥。

多篇文献从数学角度分析了RSA[24,25]、DSA[26,27]、ElGamal[28]在已知幂指数部分离散比特情况下恢复密钥的方法,为针对模幂运算实现的密码分析提供了理论支持。2005年,Percival[16]利用处理器同步多线程特性,针对滑动窗口算法模幂运算的RSA密码进行了数据Cache计时攻击,获取了512比特幂指数中的200位,并给出了恢复完整私钥的分析方法。2010年,Aciicmez等[17]针对基于滑动窗口算法实现模幂运算的RSA密码进行了指令Cache计时攻击,并利用格攻击计算出私钥。

本文以典型公钥密码中基于滑动窗口算法的模幂运算作

到稿日期:2012-05-23 返修日期:2012-12-12 本文受国家自然科学基金(61272491,60772082)资助。
周 平(1988-),男,硕士生,主要研究方向为密码旁路攻击,E-mail:Zhou.Ping_01@hotmail.com;寇应展(1964-),男,教授,主要研究方向为信息安全;王 韬(1964-),男,博士,教授,博士生导师,主要研究方向为信息安全和密码旁路分析;赵新杰(1986-),男,博士生,主要研究方向为分组密码旁路分析和故障分析;刘会英(1984-),男,博士生,主要研究方向为图像加密和密码旁路分析。

为攻击对象，在分析现有工作不足的基础上，分析预计算表在Cache中的映射机制和规律，提出了一种基于窗口值判定的幂指数分析改进算法，进一步挖掘采集的Cache计时信息，成功获取更多幂指数位，并说明了改进算法应用于RSA、DSA等密码的效果。

## 2 相关知识

### 2.1 模幂运算与滑动窗口算法

模幂运算 $y=x^e \bmod m$ 是公钥密码中最重要的运算之一，是RSA、ElGamal、DSA等公钥密码方案的核心运算。为提高效率，通常使用平方乘算法[29]、m-ary算法[30]、滑动窗口算法[31]来实现模幂运算，其中滑动窗口算法效率最高，被当前主流密码实现如OpenSSL、LibTomMath等广泛采用，算法描述见表1。

表1 滑动窗口算法

| |
|---|
| 输入：窗口大小 w，底数 x，模数 m，幂指数 $e=(e_t e_{t-1}\cdots e_1 e_0)_2$，$e_t=1$，$w>1$ |
| 输出：$y=x^e \bmod m$ |
| 计算： |
| 1. 预计算 |
| 1.1 $x_1=x \bmod m$，$x_2=x^2 \bmod m$ |
| 1.2 对于 i 从 1 到 $(2^{w-1}-1)$，计算 $x_{2i+1}=x_{2i-1} * x_2 \bmod m$ |
| 2. $y=1$，$i=t$ |
| 3. 当 $i\geqslant 0$，计算： |
| 3.1 如果 $e_i=0$：$y=y^2 \bmod m$，$i=i-1$ |
| 3.2 否则 $(e_i\neq 0)$，找到最长的比特串 $e_i e_{i-1}\cdots e_l$，其中 $i-l+1\leqslant w$，$e_l=1$，并计算：$y=y^{2^{(i-l+1)}} \bmod m$；$y=y * x_{(e_i e_{i-1}\cdots e_l)_2} \bmod m$；$i=l-1$ |
| 4. 返回(y) |

### 2.2 数据Cache计时攻击

数据Cache计时攻击主要使用间谍进程在密码进程执行某个或者多个操作的前后分别进行Cache访问，通过二次访问执行时间差异采集到密码进程访问的Cache组地址，从而推测密码算法执行时的操作序列。

假定一个 $N$ 路组相联的L1 Cache共有 $S$ 个Cache组，间谍进程构造一个与Cache大小相同的数组，循环访问映射到第1至第 $S$ 个Cache组的数组元素并记录下访问时间。每一轮访问获取一次描述数据Cache状态的矢量 $\boldsymbol{C}_t=(c_{1t},c_{2t},c_{3t},\cdots,c_{st})^{\mathrm{T}}$，循环访问获得密码进程运行过程中每一步的状态矢量，构成的 $[\boldsymbol{C}_1,\boldsymbol{C}_2,\boldsymbol{C}_3,\cdots,\boldsymbol{C}_t]$ 即为加密进程在数据Cache上留下的“踪迹”。其中，$c_{it}$ 为第 $i$ 个Cache组的访问时间，如果访问时间较长，表示发生Cache失效，密码进程访问过该Cache组。

## 3 现有幂指数分析算法

现有针对模幂运算的数据Cache计时攻击仅有Percival攻击一例[16]，下面以此为例对现有幂指数攻击算法进行分析。

### 3.1 平方/乘法操作序列获取

Percival对RSA的攻击中，主要针对OpenSSL的RSA中计算 $x^{dp} \bmod p$ 和 $x^{dq} \bmod q$ 时使用的滑动窗口算法，算法中平方/乘法操作分别对应 *BN_sqr*()和 *BN_mul*()函数。这两个函数各自有不同的临时数据空间，数据会被映射到不同的Cache组，使得平方/乘法运算执行时会在Cache中留下不同的踪迹，经进一步分析可获取平方/乘法操作序列。

### 3.2 幂指数分析

平方/乘法操作序列由幂指数决定，因此获得平方/乘法操作序列后便可推算出部分幂指数。

根据表1，滑动窗口算法将 $y=x^e \bmod m$ 分解为一系列的平方运算 $y=x^2 \bmod m$ 以及乘法运算 $y=y * x^{2k+1} \bmod m$。幂指数的每一位都对应一次平方操作，而乘法操作则发生在非零窗口的末比特位。以 $y=x^{11011010011100} \bmod m$ 为例，当窗口大小 $w=4$ 时，其窗口划分和运算序列如图1所示，其中 $S$ 表示平方操作，$M$ 表示乘法操作（为便于表述，本文主要分析 $w=4$ 的情况，$w=5$ 规律与其相同）。

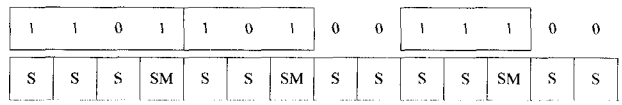


图1 窗口划分与操作序列

图1中，乘法操作中的乘法因子从预计算表 $\{x,x^3,x^5,\cdots,x^{2k+1}\} \bmod m$ 中选取，窗口值对应于预计算表中 $x$ 的指数。例如，窗口值为 $(101)_2$ 时选取 $x^5$ 作为乘法因子。

根据滑动窗口的算法特性，Percival给出了一种从操作序列推出部分幂指数位的方法：由于窗口值为奇数，因此每一个乘法操作可以确定一个值为1的比特位；由于窗口大小为4，因此只要出现连续4个以上的平方操作，就可以断定除最后4个连续的平方操作外，其它平方操作都对应0比特位。

仍然以序列 SSSSMSSSSMSSSSSMSS 为例，可恢复出幂指数 $(1xx1xx10xxx100)_2$，过程如图2所示。

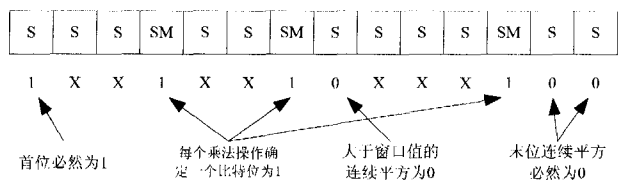


图2 Percival幂指数恢复

Percival攻击平均可恢复出512比特幂指数 $d_p$、$d_q$ 中的200比特，其中120比特来自于对连续平方判断得出的0，80比特来自对乘法操作判断得出的1。

### 3.3 不足分析

Percival算法的不足在于其仅仅利用了窗口大小和窗口末位为1的特性，而没有利用到窗口值这一信息，使得原分析算法仅能恢复出不到一半的幂指数位，导致后续数学分析的计算量过大，甚至在计算上不可行。实际上，乘法操作之间是不同的，当窗口值不同时，乘法操作会选取不同的预计算乘法因子，而不同的乘法因子可能会被加载到不同的Cache组（即每次乘法操作时发生失效的Cache组也会因窗口值的不同而不同）。分析计时信息时，在原分析算法的基础上，将乘法操作之间的不同区分出来可以进一步挖掘与每次乘法运算相对应的窗口值信息，从而缩小搜索范围，获取更多的幂指数位以大大降低后续分析的复杂度，提高攻击可行性。

## 4 基于窗口值判定的幂指数分析改进算法

### 4.1 改进思路与原理

针对现有算法的不足，改进的算法根据对应不同窗口值的乘法预计算因子被映射到不同Cache组的特性，分析窗口

值与乘法预计算因子在 Cache 中位置的映射关系，在计时信息处理时提取乘法操作中预计算乘法因子在 Cache 中的位置信息，找到可能的对应窗口值，从而在幂指数分析时缩小搜索范围，达到获取更多幂指数位的目的。

预计算表在 Cache 中的映射规律

图 3 表示数据虚拟地址的 Cache 地址映射方式[32]：低 $b$ 位表示数据在 B 字节大小的 Cache 块中的偏移；中间的 $s$ 位为组索引，标识该数据映射的 Cache 组位置；高 $t$ 位标识数据有效性。

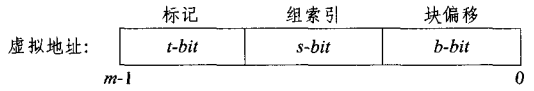


图 3 虚拟地址到 Cache 地址的映射方式

实验中，数据 Cache 采用 8 路组相联结构，共 32 个 Cache 组，每组 8 个 Cache 块，每块大小为 64 字节。32 位虚拟存储地址按如下方式被划为 3 部分：$b=6$，$s=5$，$t=21$。

本文实验采用的 OpenSSL 中，预计算表中的每个元素即预计算乘法因子都作为一个大数结构在内存中连续存储，大小为 20 字节。当窗口大小 $w=4$ 时，预计算表中有 $2^{w-1}=8$ 个乘法因子共 160 字节，会映射到 3～4 个 Cache 组。预计算表在 Cache 中的地址常是不对齐的[12]，根据其起始地址不同，8 个预计算值在 Cache 组中的所有可能分布见表 2。

表 2 预计算值在 Cache 组中的分布情况

| Cache 组相对位置 | i | i+1 | i+2 |
|---|---|---|---|
| 情况 1 | $x,x^3,x^5,x^7$ | $x^9,x^{11},x^{13}$ | $x^{15}$ |
| 情况 2 | $x,x^3,x5$ | $x^7,x^9,x^{11},x^{13}$ | $x^{15}$ |
| 情况 3 | $x,x^3,x^5$ | $x^7,x^9,x^{11}$ | $x^{13},x^{15}$ |
| 情况 4 | $x,x^3$ | $x^5,x^7,x^9,x^{11}$ | $x^{13},x^{15}$ |
| 情况 5 | $x,x^3$ | $x^5,x^7,x^9$ | $x^{11},x^{13},x^{15}$ |
| 情况 6 | $x$ | $x^3,x^5,x^7,x^9$ | $x^{11},x^{13},x^{15}$ |
| 情况 7 | $x$ | $x^3,x^5,x^7$ | $x^9,x^{11},x^{13},x^{15}$ |

将乘法操作根据其发生失效的 Cache 组的相对位置分别标记为 $M_1$、$M_2$、$M_3$，由表 2 可知，它们与预计算值有着特定的对应关系：

1）$M_1$ 对应 $\{x,x^3,x^5,x^7\}$ mod $m$，相应的窗口值为{1,11,101,111}；

2）$M_2$ 对应 $\{x^3,x^5,x^7,x^9,x^{11},x^{13}\}$ mod $m$，相应的窗口值为{11,101,111,1001,1011,1101}；

3）$M_3$ 对应 $\{x^9,x^{11},x^{13},x^{15}\}$ mod $m$，相应的窗口值为{1001,1011,1101,1111}。

图 4 给出一次实验获取的 Cache 踪迹图，能明显分辨出其平方/乘法操作序列为 $SSSSSSM_3SSSSSM_2SSSSM_1SSS$。在现有分析算法的基础上，幂指数恢复为“$0M_30M_2M_1$”，$M_1$、$M_2$、$M_3$ 分别表示{1,11,101,111}、{11,101,111,1001,1011,1101}、{1001,1011,1101,1111}，幂指数搜索空间为 $6*4*4=96$。

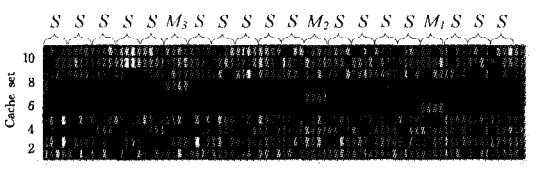


图 4 不同乘法因子映射到不同 Cache 组

作为比较，用 Percival 算法对图 4 序列进行恢复，可得幂指数位“0xxx10xxx1xxx1”，幂指数空间为 $2^9=512$。可以看出，Percival 算法本质上来说是认为乘法操作对应所有的窗口值{1,11,101,111,1001,1011,1101,1111}，而改进的算法则根据乘法操作发生的相对位置缩小了对应窗口值的可能范围，从而获得更多幂指数信息。

### 4.2 改进的幂指数分析算法及复杂度分析

在 Percival 基础上，结合 4.1 节的分析，提出了改进的幂指数分析算法，见表 3。

表 3 基于窗口值判断的幂指数分析算法

| |
|---|
| 输入：算法操作序列，乘法操作相对位置 |
| 输出：幂指数可能序列 |
| 参数：$x\in\{0,1\}$、$M_1\in\{1,11,101,111\}$、$M_2\in\{1001,1011,1101,1111\}$、$M_3\in\{1001,1011,1101,1111\}$ |
| 计算： |
| 1. 根据原始序列即乘法操作相对位置将 M 替换为 $M_1M_2M_3$ 表示 |
| 2. 判断连续的 S 操作个数 |
| 2.1 若小于 4，则按如下方式恢复：SM→1；SSM→x1；SSSM→xx1 |
| 2.2 若等于 4，则 SSSSM→$M_i$ |
| 2.3 若大于 4，则除最后 5 个操作 SSSSM→$M_i$，其余 S→0 |
| 3. 输出以 0，x，$M_i$ 表示的幂指数序列 |

设幂指数长度为 $l$，窗口大小 $w=4$，幂指数被分割为 $n$ 个非零窗口（即会发生 $n$ 次乘法操作）。幂指数信息损失完全来自幂指数到操作序列的多对一映射。非零窗口越多，损失幂指数位越多。

对于 Percival 算法，幂指数空间决定于非零窗口数，由于仅知道非零窗口末位为 1，剩下的 3 位未知，因此其幂指数搜索空间为 $2^3n$。对于改进幂指数分析算法，幂指数搜索空间取决于非零窗口数和窗口值分布。因为在改进的算法中，除由窗口末位确定的 1 以外，窗口值并非完全未知，而是根据相应的乘法操作位置判断该窗口值从{1,11,101,111}、{1001,1011,1101,1111}、{11,101,111,1001,1011,1101}的哪一个集合中选取。

本文按以下方法估算改进算法在 $w=4$ 时的平均幂指数搜索空间：由于各个窗口值的数量大致均匀分布，因此当预计算表按表 2 情况 1 映射时，分别会有平均 $4n/8$、$3n/8$、$n/8$ 次平方操作的乘法因子映射到第 $i$、$i+1$、$i+2$ 个 Cache 组，分别对应 4、6、4 种可能的窗口值，因此其平均幂指数搜索空间为 $4^{4n/8}*6^{3n/8}*4^{n/8}=2^{2.2194n}$。同理可得当预计算表按表 2 情况 2 至情况 7 映射时，其平均幂指数搜索空间分别为 $2^{2.2925n}$、$2^{2.2194n}$、$2^{2.2925n}$、$2^{2.2194n}$、$2^{2.2925n}$、$2^{2.2194n}$。由于预计算表的映射方式是均匀随机的，因此可算得总的平均幂指数搜索空间为 $2^{2.2246n}<2^{3n}$，优于 Percival 算法结果。上述估算中忽略了两次平方操作间隔小于窗口大小的情况，实际情况中改进算法的结果会更优。

## 5 实验

### 5.1 实验环境

为评估 Cache 计时攻击对 RSA、DSA 等公钥密码的安全性威胁，本文对攻击进行了仿真实验，实验环境见表 4。

表 4 仿真实验环境

| 操作系统 | 大数库 | CPU | L1 数据 Cache | | | |
|---|---|---|---|---|---|---|
| Linux Fedora 8 | OpenSSL v. 0.9.8a | Intel Pentium 4 | 容量 16kB | 相联度 8 路组相联 | 块大小 64B | 组数 32 |

### 5.2 Cache 计时信息获取及幂指数分析算法比较

调用 OpenSSL v. 0. 9. 8 大数库编写滑动窗口算法实现的模幂运算($w$=4)并对之进行了数据 Cache 计时信息采集，计时数据绘制成的灰度图如图 5 所示。通过图能够较明显地分辨出平方操作以及不同 Cache 组的乘法操作(第 11～13 Cache 组)。

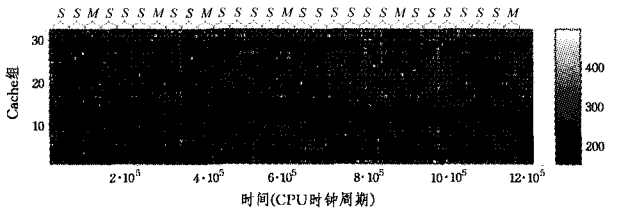


图 5　数据 Cache 计时信息伪色彩图

在已知模幂运算操作序列条件下，本文分别应用 Percival 算法和改进算法对 100000 个随机产生的 160 比特、512 比特和 1024 比特(三者为公钥密码幂指数的典型长度)幂指数进行分析，统计出能恢复出的等效比特位数以及占幂指数总位数的平均比，结果见表 5。

表 5　改进算法与 Percival 算法效果比较

| 幂指数长度 | Percival 算法 | 改进的算法 | 提高比率 |
|---|---|---|---|
| 160-bit | 74. 56/46. 6% | 94. 5048/59. 1% | 26. 75% |
| 512-bit | 239. 01/46. 7% | 302. 03/59. 0% | 26. 37% |
| 1024-bit | 473. 19/46. 21% | 593. 27/57. 94% | 25. 38% |

由表 5 可知，改进算法可进一步利用数据 Cache 计时信息，将 Percival 算法性能提升 26%左右。

### 5.3 应用实例

1) RSA 攻击应用

Percival 给出了一种中国剩余定理实现 RSA 的密钥获取方法。先利用 Cache 计时攻击获取两个作为幂指数的中间变量 $d_p$、$d_q$ 的离散位，再从高位到低位依次对比以确定某个同余方程组的范围，直至找到 $N$ 的因子 $p$、$q$。令 $|S_i|$ 表示比较到第 $i$ 位时须判断的同余方程组数，$|S_{i+1}|$ 大小取决于 $d_p$、$d_q$ 第 $i+1$ 位值：当 $d_p$、$d_q$ 的 $i+1$ 位均未知时，$|S_{i+1}|=2|S_i|$；当仅已知 $d_p$、$d_q$ 的 $i+1$ 位的其中 1 位时，$|S_{i+1}|=|S_i|$；当 $d_p$、$d_q$ 的 $i+1$ 位均已知时，$|S_{i+1}|=|S_i|/2$。则对于 $n$-bit 的 $d_p$、$d_q$，需计算同余方程总次数为：

$$F=\sum_{i=0}^{n}|S_i| \tag{1}$$

在 RSA 公钥参数 $e$=3，$d_p$、$d_q$ 为 512 比特，$N$ 为 1024 比特的条件下，图 6 给出了 $F$ 随获取的幂指数位的变化情况。

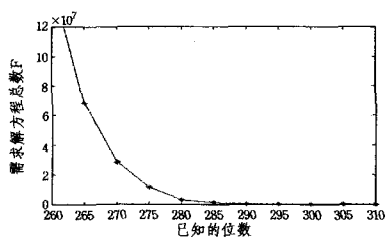


图 6　$d_p$、$d_q$ 已知位数与求解方程总数关系

图 6 表明，同余方程总次数随着 $d_p$、$d_q$ 已知位数的增加而呈指数级减少。根据实验，在 Percival 算法仅能获取不到 47%幂指数位的情况下，后续分析中需要计算的同余方程数超过 1019 个，计算量超过 3. 3 * $10^5$MIPS 年。在改进算法将 $d_p$、$d_q$ 恢复出 300 比特左右的情况下，需要计算同余方程的总次数约为 $10^4$ 个，普通计算机 1 小时内即可完成计算，大大提高了攻击的效率和可行性。

2) DSA 攻击应用

DSA 对消息 $m$ 的签名为$(r,s)$，通过如下公式产生，其中$(p,q,g)$为公钥参数，$x$ 为签名者私钥：

$$r=(g^k \bmod p) \bmod q \tag{2}$$

$$s=k^{-1}\{\mathrm{SHA}(m)+xr\} \bmod p \tag{3}$$

易知，若幂指数 $k$ 已知，则很容易通过等式(3)计算出签名者私钥 $x$。在只能得到 $k$ 的部分位情况下，格攻击[17,26,27]是当前主流密钥破解方法。然而格攻击要求攻击者有条件采集大量签名样本，在某些条件下可能需要采集上万次签名。

当 DSA 签名采用 160 比特 $k$ 时，本文改进算法可将 $k$ 的搜索空间降至约 $2^{66}$，其大大低于公认的 80 比特最小安全强度[33]。因此，在不能采集大量签名样本的情况下，本文的攻击仍然会对签名产生严重的安全威胁。而原算法仅能将 $k$ 的搜索空间降至约 $2^{86}$，仍具有一定的安全强度。

ElGamal 签名及其变体由于在计算 $r$、$s$ 时与 DSA 并无本质区别，因此同样会受到上述安全性威胁。

**结束语**　本文对基于滑动窗口实现的模幂运算数据 Cache 计时攻击进行了研究，在 Percival 工作的基础上，通过分析预计算表的映射规律，提出了一种基于窗口值判断的幂指数分析改进算法，它将幂指数获取能力提高了 26%左右；同时分析了改进算法对 RSA、DSA、ElGamal 公钥密码的现实应用。未来研究中，将分析能否通过统计映射到各个 Cache 组的窗口数量来确定预计算表是按照表 2 中哪一种具体情况映射的，从而进一步提高幂指数分析效率，降低其搜索空间。

## 参 考 文 献


[1] Kocher P C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems [A]// CRYPTO 96[C]. Santa Barbara, USA, 1996: 104-113

[2] 刘会英，王韬，赵新杰. PRESENT 相关功耗分析攻击研究[J]. 计算机科学，2011，38(11)：40-42

[3] 张鹏，邓高明，邹程. 电磁分析环境下密码设备面向实际的安全性度量[J]. 计算机科学，2010，37(3)：61-63

[4] 王寅龙，赵强，林克成. 面向计时攻击的形式化分析[J]. 计算机科学，2011，38(10)：100-103

[5] Page D. Theoretical use of Cache memory as a cryptanalytic side-channel[R]. CSTR-02-003. Department of Computer Science, University of Bristol, 2002

[6] Tsunoo Y, Saito T, Suzaki T, et al. Cryptanalysis of DES implemented on computers with Cache[A]// Workshop on Cryptographic Hardware and Embedded Systems-CHES 2003[C]. LNCS 2779, 2003: 62-76

[7] Bernstein D J. Cache-timing attacks on AES [EB/OL]. http://cr. yp. to/papers. html\#Cachetiming, 2004

[8] Osvik D A, Shamir A, Tromer E. Other people's Cache: Hyper Attacks on HyperThreaded processors[EB/OL]. Fast Software Encryption (FSE) 2005 rump session, Feb. 2005

[9] 赵新杰，王韬，郭世泽，等. 分组密码 Cache 攻击技术研究[J]. 计算机研究与发展，2012，49(3)：453-468

[10] 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

[11] 赵新杰，王韬，郑媛媛. Camellia 访问驱动 Cache 计时攻击研究

[J]. 计算机学报,2010,33(7):1153-1164
[12] 赵新杰,王韬,郭世泽. AES 访问驱动 Cache 计时攻击[J]. 软件学报,2011,22(3):572-591
[13] 赵新杰,王韬,郑媛媛. 针对 SMS4 密码算法的 Cache 计时攻击[J]. 通信学报,2010,31(6):89-98
[14] 赵新杰,郭世泽,王韬,等. 针对 AES 和 CLEFIA 的改进 Cache 踪迹驱动攻击[J]. 通信学报,2011,32(8):101-110
[15] Zhao X J,Zhang F,Guo S Z,et al. MDASCA:An Enhanced Algebraic Side-Channel Attack for Error Tolerance and New Leakage Model Exploitation[C]// Schindler W,Huss S A,eds. COSADE 2012. LNCS 7275,2012:231-248
[16] Percival C. Cache missing for fun and profit[EB/OL]. http://www. daemonology. net/papers/htt. pdf,2005
[17] Aciiçmez O,Brumley B B,Grabher P. New Results on Instruction Cache Attacks[A]// CHES2010[C]. Santa Barbara,USA,2010:110-124
[18] Brumley B B,Hakala R M. Cache-timing template attacks[A]// ASIACRYPT 2009[C]. Tokyo,Japan,2009:667-684
[19] 陈财森,王韬,陈建泗,等. 基于 Cache Missing 的 RSA 计时攻击[J]. 微电子学与计算机,2009,26(5):180-182,186
[20] 郑媛媛,王韬,赵新杰,等. 针对 RSA 密码算法的指令 Cache 攻击方法[J]. 微电子学与计算机,2009,26(2):197-200
[21] Zenner E. A Cache timing analysis of HC-256[A]// SAC 2008[C]. LNCS 5381,2009:199-213
[22] Brumley B B,Hakala R M,Nyberg K,et al. Consecutive S-box Lookups:A Timing Attack on SNOW 3G[A]// ICICS 2010[C]. LNCS 6476,2010:171-185
[23] Leresteux D,Fouque P A,Chardin T. Cache Timing Analysis of RC4[A]// ACNS 2011[C]. LNCS 6715,2011:110-129
[24] Coppersmith D. Finding a small root of a bivariate integer equation;factoring with high bits known[A]// Cryptology EUROCRYPT 96[C]. Saragossa,Spain,1996:178-189
[25] May A. New RSA vulnerabilities using lattice reduction methods[D]. Paderborn,Germany,University of Paderborn,2003
[26] Nguyen P Q,Shparlinski I E. The insecurity of the Digital Signature Algorithm with partially known nonces[J]. Cryptology,2002,15:151-176
[27] Leadbitter P J,Page D,Smart N P. Attacking DSA Under a Repeated Bits Assumption[A]// CHES2004[C]. Boston,USA,2004:141-190
[28] Kuwakado H,Tanaka T. On the security of the ElGamal-Type signature scheme with small paramenters[EB/OL]. http://www. lib. kobe-u. ac. jp/handle_kernel/90001314,1999
[29] Knuth D E. The Art of Computer Programming[M]. Massachusetts:Addison-Wesley,1981
[30] Menezes A J,van Oorscht P C,Vanstome S A. 应用密码学手册[M]. 北京:电子工业出版社,2005
[31] Koc C. Analysis of sliding window techniques for exponentiation[J]. Computers and Mathematics with Application,1995,30(10):17-24
[32] Bryant R E,Hallaron D. 深入理解计算机系统[M]. 北京:机械工业出版社,2011
[33] Babbage S,Catalano D,Cid C. Yearly Report on Algorithms and Keysizes(2011)[R]. http://www. ecrypt. eu. org/documents/D. SPA. 17. pdf,2011

(上接第 200 页)

这样,通过利用引理 interim 和 p_nr_equiv 来证明目标命题,其证明如图 14 所示。

```
lemma "s,deal_intr ⊢EF (Atom (λw. (reg_s w=reg_s s) ∧
    p_nr (now_proc s)=p_nr (now_proc w)))"
apply (insert p_nr_equiv [of s])
apply (insert interim [of s])
by (erule ssubst)
```

图 14 目标命题的证明

在 Isabelle 中最后的证明结果截图如图 15 所示。

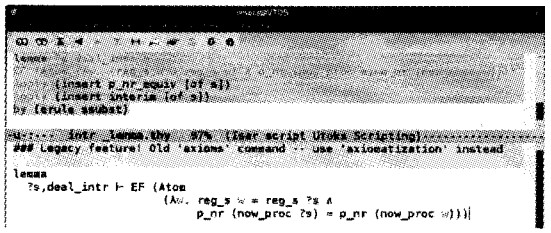

图 15 证明结果

**结束语** 本文采用形式化方法对 VTOS 的中断机制进行了设计,并采用 Isabelle/HOL 对设计过程进行了描述,对 VTOS 中断机制的完整性问题进行了验证。本文对操作系统形式化设计和验证工作起到了一定的借鉴意义。在具体的设计和验证过程中,不得不承认其工作量非常大,Isabelle/HOL 完整的形式化描述和验证工作在代码量与具体的实现代码上大概为 10∶1 左右。在未来的研究工作中,将考虑采用类型论来实现系统代码的自动构造。

## 参 考 文 献

[1] Elkaduwe D,Klein G,Elphinstone K. Verified protection model of the seL4 microkernel[R/OL]. http://ertos. nicta. com. au/publications/papers/Elkaduwe_GE_07. pdf,2012-05-25
[2] Yale Flink Project. Website[OL]. http://flink. cs. yale. edu/,2012
[3] Stampoulis A,Shao Z. VeriML:Typed Computation of Logical Terms Inside a Language with Effects[C]// Proc. of the 15th ACM SIGPLAN International Conference on Functional Programming. New York,USA:ACM Press,2010
[4] Shao Z,Ford B. Advanced Development of Certified OS Kernels[D]. New Haven,USA:Yale University,2010
[5] 陈意云,李兆鹏,王志芳. 一种用于指针程序验证的指针逻辑[J]. 软件学报,2010,21(3):414-426
[6] 王振明,陈意云,王志芳. 一个用于指针程序验证的自动定理证明器的设计与实现[J]. 小型微型计算机系统,2010,5:801-806
[7] Nipkow T,Paulson L C. Isabelle/HOL:A Proof Assistant for Higher-order Logic[M]. Berlin,Germany:Springer,2002
[8] 钱振江,刘苇,黄皓. OS 对象语义模型(OSOSM)及形式化验证[J]. 计算机研究与发展,2012(12):2702-2712