MDx 差分攻击算法改进及 GPGPU 上的有效实现

周 林 韩文报 祝卫华 王 政

(解放军信息工程大学信息工程学院 郑州 450002)

摘 要 Hash 函数广泛应用于商业、安全等领域,其中 MDx 系列 Hash 算法应用最为广泛.因此对 MDx 系列 Hash 算法的攻击在理论上和实际应用上都有重要的意义.自王小云教授提出差分攻击算法并攻破 MD5、MD4 等 MDx 系列算法以来,对该算法的研究日益受到关注.文中以攻击 MD5 的差分攻击算法为例,改进了 Klima 提出的 MD5 隧道差分攻击算法,分析其在 GPGPU 上实现的可行性和技术要求并在 Visual studio 6.0 的环境下利用 CUDA 语言开发完成.算法的 CUDA 程序在 GeForce 9800 GX2 平台下运行,平均每 1.35s 能找到一对 MD5 碰撞.通过同 4 核 Core 2 Quad Q9000 (2.0GHz) PC 上的实现相比较,在 GeForce 9800 GX2 上的实现能达到 11.5 倍的性价比.

关键词 MD5; Hash 函数;差分攻击;隧道技术;多消息修正方法; CUDA; GPGPU; Mersenne Twister

中图法分类号 TP331 **DOI 号**: 10.3724/SP.J.1016.2010.01177

Improvement and Efficient Implementation of Differential Attack Algorithm to MDx on GPGPU

ZHOU Lin HAN Wen-Bao ZHU Wei-Hua WANG Zheng

(*Information Engineering Institute, PLA Information Engineering University, Zhengzhou* 450002)

Abstract Hash functions are widely used in business, military field etc. Among them, MD-family Hash functions are the most extensively utilized hash functions. Therefore, the attack to MD-family hash functions has important meaning in theory and in practical application. Since professor Wang proposed differential attack algorithm and succeeded to break MD5, MD4, etc MD-family algorithms, this algorithm has been paid more and more attention. This paper takes the tunnel differential attack algorithm to MD5 as example, improves the tunnel differential attack algorithm to MD5 proposed by Klima and analyzes the feasibility and technology request of differential attack algorithm's implementation on GPGPU. Its CUDA program run on the GeForce 9800 GX2 platform and can find a pair of MD5 collision in just 1.35 seconds average. Through comparing the software implementation on PC Core 2 Quad Q9000 (2.0GHz), we can get nearly 11.5 times more cost-efficient by the implementation on GeForce 9800 GX2.

Keywords MD5; Hash functions; differential attack; tunnel technique; multi-message modification method; CUDA; GPGPU; Mersenne Twister

---

收稿日期:2009-09-28;最终修改稿收到日期:2010-05-20.本课题得到国家自然科学基金(2007B74)、国家"八六三"高技术研究发展计划项目基金(2009AA012417)资助.**周 林**,男,1986 年生,硕士研究生,主要研究方向为网络密码分析、Hash 函数分析、GPGPU、CELL、FPGA 在密码学中的应用. E-mail: zhoulinpx@163.com. **韩文报**,男,1963 年生,教授,博士生导师,主要研究领域为密码学和信息安全. **祝卫华**,男,1977 年生,博士,讲师,主要研究方向为网络密码分析、Hash 函数分析、GPU 在密码学中的应用. **王 政**,男,1975 年生,博士,副教授,主要研究方向为网络密码分析、Hash 函数分析、网络协议分析.

1 引 言

Hash 函数是密码学的重要分支. 以 SHA-1、MD5 为代表的 MDx 系列 Hash 算法是最为典型的 Hash 函数,应用也最为广泛,例如 Hash 算法在密码协议中具有重要作用. 在密码协议设计中一般都使用随机预言模型,即假设所使用的 Hash 算法是安全的. Hash 算法的安全性直接关系到密码协议的安全性. 因此对 Hash 函数的安全性分析一直是密码界的热点.

在 CRYPTO98 上,Chabaud 和 Joux 提出一种对 SHA-0 的攻击方式. 在 2^{61} 的计算复杂度之内,就可以发现一次碰撞(即两个不同的消息对应到相同的消息摘要);这个数字小于生日攻击复杂度 2^{80},其安全性低于一个理想的杂凑函数抵抗攻击所应具备的计算复杂度.

2004 年,Biham 和 Chen 发现了 SHA-0 的近似碰撞,其中 160 位消息摘要中有 142 位相同. 同年 Joux、Carribault、Lemuet 和 Jalby 宣布找到 SHA-0 算法的完整碰撞,这是归纳 Chabaud 和 Joux 的攻击所完成的结果,发现一个完整碰撞只需要 2^{51} 的计算复杂度.

直到在 CRYPTO2004 上,王小云、冯登国、来学嘉和于红波利用差分攻击方法成功攻破 MD5、SHA-0 和其它杂凑函数. 他们攻击 SHA-0 的计算复杂度是 2^{40},攻击 MD5 的计算复杂度是 2^{42}. 在一台普通的 PC 上,15min 之内就可以找到 MD5 碰撞消息对.

2005 年,王小云和殷益群、于红波将差分攻击算法公开,并将差分攻击算法应用于对 SHA-1 的攻击,只需少于 2^{69} 的计算复杂度,就能找到一组碰撞. 在 CRYPTO2005 会议尾声中王小云、姚期智、姚储枫再度发表更有效率的 SHA-1 攻击法,能在 2^{63} 个计算复杂度内找到碰撞. 从此差分攻击算法被深入研究并得到了很大改进[1-2].

在密码学的学术理论中,任何攻击方式,其计算复杂度若少于暴力搜寻法所需要的计算复杂度,就能被视为针对该密码系统的一种破密法;但这并不表示该破密法已经可以进入实际应用的阶段.

在实际攻击中,理论上的攻击算法都要映射到合适硬件计算平台上的可执行程序. 硬件计算平台一般可分为标准计算平台(例如 PC)和专用计算平台(例如 FPGA、ASIC、GPU). 通常在专用计算平台上的开发难度要比在 PC 上难. 但往往可以获得更高的性价比,特别是对可以并行的算法.

差分攻击算法是一种有策略的随机搜索算法,每次搜索开始前,先产生一个随机数,所需的消息分组通过计算得到,由消息分组计算链变量并检查链变量是否符合条件,如果不符合则需要重新开始并产生新的随机数,因此每次搜索是独立的,可以并行操作,算法所需的存储空间仅仅是 64 个字的轮常量,适合在 GPGPU 上实现. 在公开文献中,还没有一篇是分析差分碰撞算法在 GPGPU 上实现的文章,本文希望能为同行抛砖引玉,进一步探索 GPGPU 在密码学中的应用.

本文第 2 节介绍 MDx 系列 Hash 算法的构造;第 3 节概括目前对 MDx 系列 Hash 算法的攻击算法;第 4 节改进文献[1]中的差分算法,通过分析差分攻击算法,得出在 GPGPU 上实现时所需的技术要求,介绍实现框架,给出实验结果并与普通 PC 上的实现进行比较;第 5 节总结全文.

2 MDx 系列 Hash 算法

将任意长度的数字串 m 映射成一个较短的定长的数字串的函数称为 Hash 函数,以 h 表示 Hash 函数:

$$h:\{0,1\}^t\rightarrow\{0,1\}^n,$$

则 $h(m)$ 为 m 的 Hash 值. $h(m)$ 应当易于计算且 $t>n$. Hash 函数还必须是单向的、弱无碰撞的和强无碰撞的.

定义 1. 给定一个 Hash 函数 h,y 为一个消息摘要,若要找出 x 使得 $y=h(x)$ 在计算上不可行,则称此 Hash 函数为单向的.

定义 2. 如果有两个消息 m_1,m_2;$m_1\neq m_2$ 使得 $h(m_1)=h(m_2)$,我们就说这两个消息 m_1 和 m_2 是碰撞(collision)的消息.

定义 3. 给定 Hash 函数 h 和任意给定的消息 m,如果要找一个 m',$m'\neq m$,使得 $h(m')=h(m)$ 在计算上不可行,则称 h 是弱无碰撞的 Hash 函数(weak collision-free Hash function).

定义 4. 给定一个 Hash 函数 h,如果要找到任意一对消息 m_1,m_2;$m_1\neq m_2$ 使得 $h(m_1)=h(m_2)$ 在计算上不可行,则称 h 是强无碰撞的 Hash 函数(strong collision-free Hash function).

处理可变长消息是很困难的,所以 MDx 系列算法中先将消息 m 填充为若干固定长度的分组:

$$M_1, M_2, \cdots, M_r,$$

这里每个 $M_i(1 \leq i \leq r)$ 是长为 k bit 的串. 然后利用压缩函数 f 处理每个 M_i:

$$f: \{0,1\}^k \rightarrow \{0,1\}^n,$$

其中 $k > n$, f 的输出成为链变量 fv_i. 为了关联各个分组,使得 fv_i 的值依赖于已处理的分组. 处理每个分组 M_i 时,将 fv_{i-1} 作为初始值,再使用 f 将其更新得到 fv_i. 第一个分组的初始值 fv_0 是一个预先规定的常值 IV.

$$fv_0 = IV,$$
$$fv_i = f(fv_{i-1}, M_i),\ 1 \leq i \leq r,$$
$$h(m) = fv_r.$$

最后的消息摘要值为 fv_r. MDx 系列算法主要在 IV 的设置和 f 的构造上有所不同.

MDx 系列 Hash 算法以 Merkle-Damgard 理论为基础. Merkle-Damgard 理论表明:Hash 函数 h 的安全性取决于压缩函数 f 的安全性,如果 f 是单向的、弱无碰撞的和强无碰撞的,则 h 也是单向的、弱无碰撞的和强无碰撞的.

3　MDx 系列 Hash 函数的攻击算法

Hash 函数攻击算法大体可分为两类:通用算法(例如生日攻击、中途相遇攻击和穷举攻击)和特定算法(例如王小云的差分攻击、Dobbertin 的代数攻击). 通用算法一般适用于对所有 Hash 算法攻击,特定算法只能针对某一个或某一类 Hash 算法进行攻击. 通用算法攻击的复杂度一般都很大,例如假设 Hash 函数的输出的消息摘要长为 n,则利用生日攻击所需的运算量是 $O(2^{\frac{n}{2}})$[3]. MD5 的消息摘要长是 128,所以利用生日攻击所需的运算量是 $O(2^{64})$,而使用差分攻击算法,目前最好的结果是只需 $O(2^{10})$.

特定攻击算法是利用 Hash 函数的内在结构缺陷,找到 Hash 函数的弱点有针对性地进行攻击. 例如 Dobbertin 利用 MD5 中活动状态的高位不能尽快地充分混淆,通过构造两个不同的 512 消息分组和选择初始 IV 值得到半自由初始碰撞. 王小云的差分攻击算法也是利用了 MSB(最高比特位)不能尽快充分混淆,找到有效的差分和差分路径,成功攻击了 MD5、MD4 等算法[4]. 差分攻击算法一般分为 3 个步骤:

1. 构造可以产生高概率碰撞的差分 ΔM.

2. 推导出 ΔM 的差分路径 DP 和实现差分路径需要满足的充分条件 SC.

3. 利用单消息修正方法、多消息修正方法、隧道技术、“分而治之”技术设计和优化搜索满足充分条件的碰撞消息对.

步 1 中,将 $\Delta M \neq 0$ 加上一个消息 M 可得到 M',为了使得 $h(M) = h(M')$,差分 ΔM 在压缩函数 f 中的扩散和混淆必须按照预先设定的路径进行并最后消失. 步 2 就是要构造这样的差分路径 DP,并推出在什么充分条件下能得到这样的 DP,充分条件数决定整个攻击算法的运算复杂度. 步 3 利用消息中的自由比特位,使用单消息修正方法[4]、多消息修正方法[5]、隧道技术[2]、“分而治之”[6]技术提高所构造的消息满足所有充分条件的概率.

4　差分攻击算法改进及在 GPGPU 上的实现

差分攻击算法首先需要产生一个随机数,因此在实现中要构造一个伪随机数发生器. 由于寄存器和共享内存大小限制和内存寻址限制,迭代模式的伪随机数发生器(如线性同余法、Fibonecci 方法)在 GPGPU 上实现的效率比较低. 我们采用 Mersenne Twister 算法,Mersenne Twister 算法生成的随机数周期长、概率分布性好、所需内存少、速度快且适合在 GPGPU 上实现. 详细细节请参见 cuda sdk.

在计算链变量和消息值时还需要模加运算、模减运算、按位与、或、非运算和循环左移及循环右移算子. cuda 开发语言中含有模加运算、模减运算、按位与、或、非运算. 循环左移和循环右移算子可以通过左移和右移算子转换得到. 例如对数 x 循环左移 n 位,则可表示为 $(x \ll n) | (x \gg (32-n))$,对数 x 循环右移 n 位,则可表示为 $(x \gg n) | (x \ll (32-n))$.

在判断消息是否满足充分条件时还需用到控制流指令. 在开发语言 CUDA 中有 if,switch,do,for,while 控制流指令. 但是任何流控制指令(if,switch,do,for,while)都会导致同一 warp 块的线程分支,从而显著影响有效的指令吞吐量,也就是说,这些指令会导致线程采用不同的执行路径. 如果出现这种情况,就必须序列化不同的执行路径,因而增加了该 warp 块执行的指令总数. 完成所有不同的执行路径时,线程将重新汇聚到同一执行路径. 因此一方面我们尽量减少控制流指令,另一方面优化必要的减少控制流指令.

在差分攻击算法中,为了实现隧道技术,使用了

for 循环语句，可以通过展开 for 循环语句，消除控制流指令. 图 1 表示了展开和未展开 for 循环语句的效率差别.

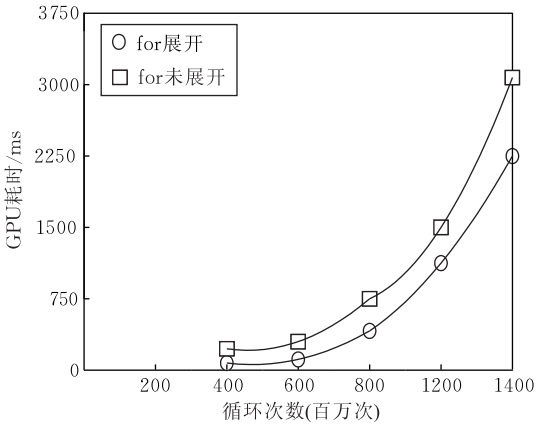

图 1 展开和未展开 for 循环语句的效率比较

从图 1 可以看出当循环的次数越大时，展开和未展开 for 循环语句的效率差别越大.

在实现多消息修正技术和判断充分条件是否成立时，需要使用 if 语句，但这些 if 语句都含空分支. if 指向空分支或者按 half warp 跳转都没有性能损失，所以在边界控制中的 if 不会造成性能损失，只是多了一次判断. 另外控制器可能用分支谓词来优化 if 指令. 在使用分支谓词时，依靠控制条件执行的任何指令都不会被跳过. 而是分别与一个每线程条件代码或根据控制条件设置为 true 或 false 的谓词相关联，尽管每一条指令都为执行而进行了调度，但只有谓词为 true 的指令才会被实际执行. 带有 false 谓词的指令不会写入结果，也不会计算地址或读取操作数. 只有在分支条件控制的指令数量小于或等于特定阈值时，编译器才会使用有谓词的指令替换分支指令：如果编译器确定出有可能产生大量分支 warp 块的条件，则此阈值为 7，否则为 4.

差分攻击算法中还需 continue 指令，但 CUDA 中并没有 continue 指令，我们可以去掉不能转换的 continue 指令. 例如可以去掉攻击 MD5 的隧道差分算法[1]中第 25 步到第 64 步中的 continue 指令，第 18 步～第 24 步中的 continue 指令可以利用多消息修正方法转换为 if 指令. 因为在算法结束时，会对链变量进行判断是否符合条件，且给定消息输入，连续 4 个链变量就可以决定下一步的链变量的值. 因此去掉 continue，并不会产生错误.

下面将分析对攻击 MD5 的隧道差分算法[1]的改进，定义 $Q_i(1\leq i\leq 64)$ 为 MD5 第 i 步计算出的链变量，$\{m_0,m_1,\cdots,m_{15}\}$ 为消息分组，其中 $m_j\{0\leq j\leq 15\}$ 为 32bits，详细符号定义和算法流程请参见文献[1]. 在文献[1]提出的差分攻击算法中计算和判断链变量 $Q_{18}\cdots Q_{24}$ 时都用到了 continue 控制流指令. 如果 $Q_{18}\cdots Q_{24}$ 不满足充分条件，则需要重新开始计算整个算法流程. 因此算法的效率不但降低了而且在 GPGPU 上不能执行 continue 指令.

因此我们利用多消息修正方法对此进行了改进. 本算法的具体改进方案为：如果 $Q_{18}\cdots Q_{24}$ 不满足充分条件，通过多消息修正方法调整消息值，使得 $Q_{18}\cdots Q_{24}$ 以一定概率满足充分条件，同时不影响以前充分条件的满足.

例如，在文献[4]中 Q_{21} 需要满足的充分条件是 $Q_{21}[31]=0$，假设 $Q_{21}[31]=1$，由于

$$Q_{21}=Q_{20}+(f_{20}(Q_{20},Q_{19},Q_{18})+Q_{17}+0xd62f105d+m_5)<<<5\bmod 2^{32}.$$

所以可以通过改变 m_5 的第 26 比特，即令 $m_5=m_5-2^{26}$ 使得 $Q_{21}[31]=0$. m_5 在第 1 轮的第 6 步中也用到了，

$$Q_6=Q_5+(f_5(Q_5,Q_4,Q_3)+Q_2+0x4787c62a+m_5)<<<12\bmod 2^{32}.$$

这导致 Q_6 的改变，为了不影响第 21 步之前充分条件的满足，我们还需要做一些相应的修改，详细情节请参见表 1.

表 1 利用多消息修正方法调整 $Q_{21}[31]$ 的修改方案

步数	修改方案
4	$m_3=m_3+2^4$
5	$m_4=((Q_5-Q_4^{\text{new}})>>>7-Q_1-f_4(Q_4,Q_3,Q_2)-K_4$
6	$m_5=m_5-2^{26}$
7	$m_6=m_6$
8	$m_7=m_7-2^{26}$

表 1 中不但 m_5 改变了，m_3,m_4,m_7 也改变了. 但是在前 21 步中，m_3,m_4,m_7 只在第 4,5,8 步中被用到，因此不会影响第 21 步之前充分条件的满足！

攻击算法主体实现流程图见图 2.

整个构架在 Visual studio 6.0 的环境下利用 CUDA 语言开发完成. 在 NVIDIA 公司 GeForce 9800 GX2 上运行. 得到的结果以及在目前高端 PC Core 2 Quad Q9000(2.0GHz)4 核电脑上的运行结果如表 2 所示，表中所给的市场价参照的是进行实验时的设备价格，把开发完成后的可执行文件用于实际攻击时，主要的问题是设备，因此以设备当时的市场价做为实现代价是比较合理的.

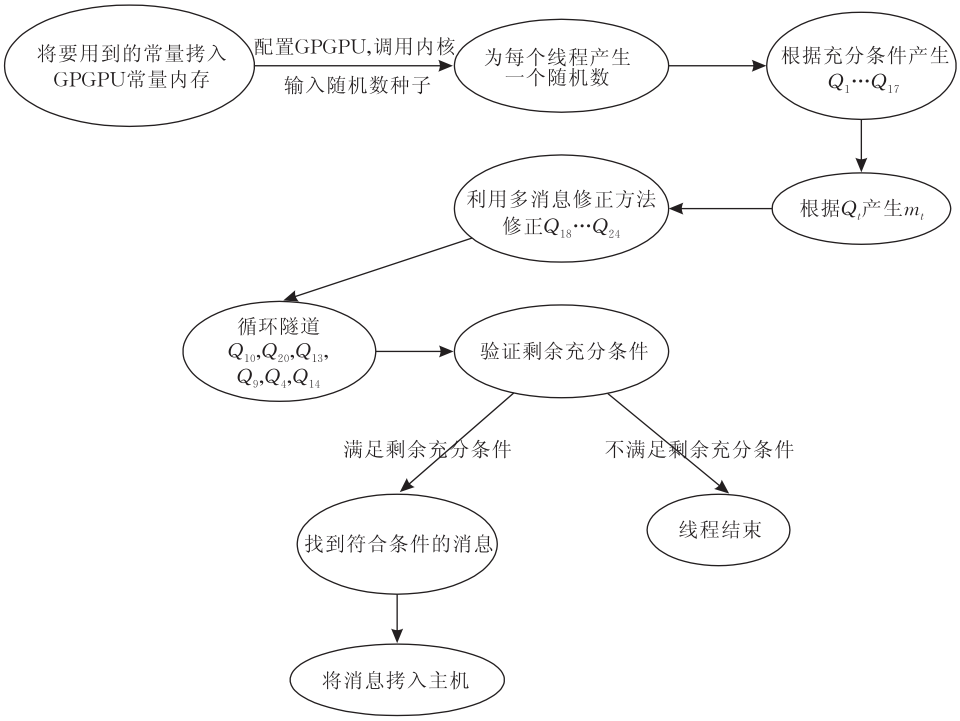

图 2　差分攻击算法在 GPGPU 上的实现流程图

表 2　在 GeForce 9800 GX2 和 Core 2 Quad Q9000(2.0GHz) 4 核 PC 上运行差分攻击算法的实验结果对比

计算平台	耗时/s	市场价格/¥	性价比
GeForce 9800 GX2	1.35	1388	11.5
Core 2 Quad Q9000	8.0	2700	0.08

5　结束语

MDx 系列 Hash 算法的软硬件实现效率高,同时安全性也比较高,被广泛应用于保密通信、数字签名、协议认证等安全领域. MDx 系列 Hash 算法的安全性成为了这些安全领域的安全基石. 然而对 MDx 系列 Hash 算法的安全分析直到王小云提出差分攻击算法后才有了重大突破. 本文认证了差分攻击算法不但可以在 GPGPU 上实现而且还能达到 11.5 倍的性价比. GPGPU 的功率太大,GeForce 9800 GX2 的功率就达到 210W,从而限制了多 GPGPU 使用,温度太高也增加了系统的不稳定性,然而 GPGPU 的强大计算能力,便宜的价格使得它的应用越来越广泛.

差分攻击思想具体应用到攻击某个 Hash 函数时,所得到的具体算法是不同的. Klima[1] 提出的隧道差分攻击算法被公认为经典差分攻击算法,虽然还有复杂度更低的差分攻击算法,但是关键的技术细节并没有透露. 因此我们采用隧道差分攻击算法作为改进和实现的对象. 另外 SHA-1 算法在理论上虽然已被攻破[4],但是实际上并没有攻破,目前并没有找到 SHA-1 的碰撞. 差分攻击算法在 GPGPU 上实现的优势将对寻找 SHA-1 的碰撞带来帮助.

参　考　文　献

[1] Klima V. Tunnels in Hash functions: MD5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105, 2006. http://eprint.iacr.org/

[2] Wang Xiaoyun, Yin Yiqun Lisa, Yu Hongbo. Finding collisions in the full SHA-1//Advances in Cryptology — Crypto05. LNCS 3621. Springer, 2005: 1-6

[3] NVIDIA Corporation. NVIDIA CUDA Programming Guide. 2007. http://www.nvidia.com/

[4] Wang Xiaoyun, Yu Hongbo. How to break MD5 and other Hash functions//Advances in Cryptology — Eurocrypt05. LNCS 3494. Springer, 2005: 1-18

[5] Zhang Meng, Sasaki Y, Naito Y, Kunihiro N, Ohta K. Improved Collision Attack on MD5, Cryptology ePrint Archive, Report 2005/400, November 2005. http://eprint.iacr.org/

[6] Tao Xie, Liu Fanbao, Feng Dengguo. Could the 1-MSB input difference Be the Fastest Collision Attack For MD5? Cryptology ePrint Archive, Report 2008/391, 2008. http://eprint.iacr.org/

[7] Biham E, Chen R. Near collisions of SHA-0//Advances in Cryptology — Crypto'04. Springer-Verlag, 2004: 290-305

[8] Wang Xiaoyun, Feng Dengguo, Lai Xuejia, Yu Hongbo. Collisions for Hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. http://eprint.iacr.org/

[9] Wang Xiaoyun, Yu Hongbo. How to break MD5 and other Hash functions//Ronald Cramer ed. Advances in Cryptology — EUROCRYPT 2005. Lecture Notes in Computer Science 3494. Springer, 2005: 19-35

[10] Cameron McDonald, Philip Hawkes, Josef Pieprzyk. Differential path for SHA-1 with complexity $O(2^{52})$. Cryptology ePrint Archive, Report 2009/328, March 2009. http://eprint.iacr.org/

ZHOU Lin, born in 1986, M. S. candidate. His research interests include network cryptograph analysis, Hash analysis and the application of GPGPU, CELL, FPGA in cryptograph.

HAN Wen-Bao, born in 1963, professor, Ph. D. supervisor. His research interests include cryptograph and information security.

ZHU Wei-Hua, born in 1977, Ph. D., lecturer. His research interests include network cryptograph analysis, Hash analysis and GPU's application in cryptograph.

WANG Zheng, born in 1975, Ph. D., associate professor. His research interests include network cryptograph analysis, Hash analysis and network protocol analysis.

Background

Data integrity assurance and data origin authentication are essential security services in financial, transactions, electronic commerce, electronic mail, software distribution, data storage and so on. Cryptographic hash functions are utilized to achieve these security services. The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. A hash value h is generated by a function H of the form $h=H(M)$, where M is a variable-length message and $H(M)$ is the fixed-length hash value. In a cryptographic hash function, a message of arbitrary length padded and broken into blocks is input sequentially to a compression function which converts a fixed-length input (current message block) to a fixed-length output (hash value).

MD5 is one of most widely used hash function. However, in August 2004 Wang et al. published their collisions for MD4, MD5, HAVAL-128 and RIPEMD. Later, Xiaoyun-Wang and Hongbo Yu presented the underlying method to construct collisions using differential paths, which are a precise description how differences propagate through the MD5 compression function. Later there are many improvements.

In recent years, there has been significant interest from both academy and industry in applying commodity graphics processing units (GPUs) toward general computing problems. This trend toward general-purpose computation on GPUs (GPGPU) is spurred by the large number of arithmetic units and the high memory bandwidth available in today's GPUs. The computational power of GPUs is growing at a faster rate than what Moore's Law predicts for CPUs. With the introduction of native integer and binary operations in the latest generation of GPUs, we believe many cryptography algorithm are ideally suited to the GPGPU programming model. But efficient parallelization of MD5 collision algorithms is particularly challenging on a graphics processor unit (GPU) due to the need for massive control flow instructions such as those generated from if and for statements which can significantly impact the instruction throughput. In this paper, the authors avoid this problem efficiently by unrolling the for instruction, utilizing branch predication technology and transforming continue instruction. The CUDA implementation improves the efficiency of MD5 collision algorithms by a factor of 11.5 compared to a standard CPU based implementation and can be used in MD5 collision algorithms which is based on Wang's. This research is supported by the National Natural Science Foundation of China under grant No. 2007B74 and the National High Technology Research and Development Program (863 Program) of China under grant No. 2009AA012417. The purpose of our research is to analyze, utilize the GPGPU in the cryptography field and get good performace.