MySQL密码认证算法的FPGA优化实现※

杨文勇　林捷　沈舒莉　黄鹭　张琪　陈艺慧　李晓潮

（1.厦门城市职业学院，福建省　厦门市　361008；2.厦门大学，福建省　厦门市　361005）

摘　要：MySQL数据库软件被广泛地应用于WEB项目中，因此它的安全性极为重要。本文针对MySQL密码认证算法的FPGA实现以及应用进行研究，利用并行化、流水线架构实现MySQL认证算法。提出一种基于BRAM的SHA-1算法流水线架构并对其流水线级数进行优化，从而实现高性能。同时利用多模块并行，多密码验证算法，以提高硬件运行速度和计算资源的利用效率。通过优化，高性能实现了基于FPGA硬件的MySQL认证算法，单块芯片的口令恢复速度为18亿个口令密码每秒，多个FPGA构成子板的认证速度为72亿/秒，比采用GPU GTX 690平台的速度提高了5倍。

关键词：现场可编程门阵列；认证算法；并行架构；流水线架构

中图分类号：TP311.13　　**文献标识码**：A　　**文章编号**：1671—1440(2017)06—0085—06

1　引言

互联网时代的发展逐步进入大数据时代，而数据库软件为我们提供了处理和存储数据的方式，其中最流行的关系型数据库管理系统就是MySQL数据库软件。在WEB应用方面MySQL是最好的关系数据库管理系统之一。全球许多行业巨头（包括Facebook、Google、Adobe、Alcatel Lucent和Zappos）都使用 MySQL 来支持其高流量网站、关键业务系统[1]。在这些大型网站架构中，MySQL扮演着最重要的角色。为了保障数据安全，MySQL数据库软件在用户需要获得更高数据处理权限的时候需要用户输入口令进行身份验证。

MySQL密码认证所用到的核心算法为SHA-1。基于FPGA的SHA-1算法实现主要是利用流水线和循环展开技术来提高吞吐率[2]，研究表明随着流水线级数的增加，尽管资源消耗更多，但是吞吐率也随之提高。由于单个FPGA内部的资源有限，至今尚未有人实现全流水线结构。Makkad等人综合使用了循环展开，预计算和流水线等优化手段，在Virtex-6器件上实现的SHA-1算法吞吐率达到了8.6 Gbps[3]。目前被广泛认可的优化方案主要有进位保存加法[4]，它将SHA-1算法中的关键路径中的加法分割为加和进位两种运算，从而减小了关键路径的延时；循环展开通过将SHA-1循环中的几轮运算合并来减少计算轮数，提高轮计算的效率[5]；以及利用寄存器来减少关键路径延时的流水线方案[6]等。

本文工作主要针对MySQL密码认证机制算法进行FPGA硬件优化实现，使用并行框架下全流水线的设计模式对其算法进行了资源和运行频率的优化，以取得高吞吐率的效果。其最终的性能将和目前通用众

※**基金项目**：本研究项目获得集成电路设计与测试分析福建省高校重点实验室建设资金，厦门市科技局科技专项经费以及2017年厦门大学教学改革研究项目资金的支持。

收稿日期：2017—11—20

作者简介：杨文勇（1971—），男，黑龙江省齐齐哈尔人，厦门城市职业技术学院电子与信息工程系讲师，研究方向：计算机软件、网络及自动控制。

李晓潮（1970—），男，福建省福州人，厦门大学电子工程系副教授，电路与系统方向首席研究员，博士生导师，IEEE SSCS会员，厦门市第七批领军型创业人才，研究方向：集成电路设计、信息安全及人工智能等。

核GPU平台的吞吐率进行比较,并最终应用到基于FPGA的密码恢复应用系统方案中。基于这个目的,本文将从系统架构,全流水线模块设计和优化,多密文匹配以及系统性能测试和分析几个方面详细阐述。

2 系统构成

MySQL的密码认证机制(也称为mysql_native_password)流程如图1所示:

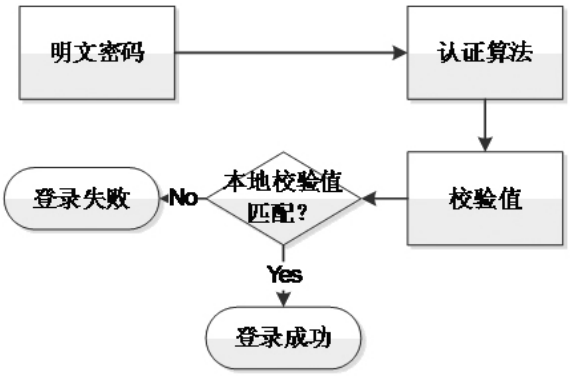

图1 MySQL认证机制流程

根据用户输入的用户名,通过密码认证算法对明文密码进行计算,计算后得出的校验值与该用户在MySQL本地中存储的校验值进行匹配比对,如果匹配则用户身份得到认证,登录成功,否则身份认证失败,无法获取权限。MySQL的密码支持最长64字节(512bit)的密码设置。

MySQL V4和V5版本的校验算法如下,先把密码明文Message根据SHA-1的要求进行扩充得到64字节,进行一次SHA-1算法计算,得到散列的20字节的哈希值,然后再把这哈希值作为第二次SHA-1操作的输入进行扩充和再一次的SHA-1计算,得到最后的20字节的校验值。V4版本的校验值为10个字节,V5版本的校验值为20个字节[6]。因此,MySQL密码认证机制中,SHA-1算法的实现和优化至关重要。

2.1 系统硬件架构

整个系统架构分成两个部分,一部分是处理器模块,系统利用ARM CPU对MySQL认证数据的进行解析和处理,而后通过自定义的AXI-4总线将认证数据和策略发送给认证算法模块(FPGA)进行认证和校验运算。另外一部分是由多个FPGA构成的并行认证算法兑现模块。为了提高计算性能,我们在每个芯片上尽可能多的实现多个并行运算的认证模块,并以片选信号CS[j]进行不同策略分配。数据和策略包括初始密码配置,密码字符集,密码最大长度,待匹配的校验值以及相应的命令等信息。如下图所示,

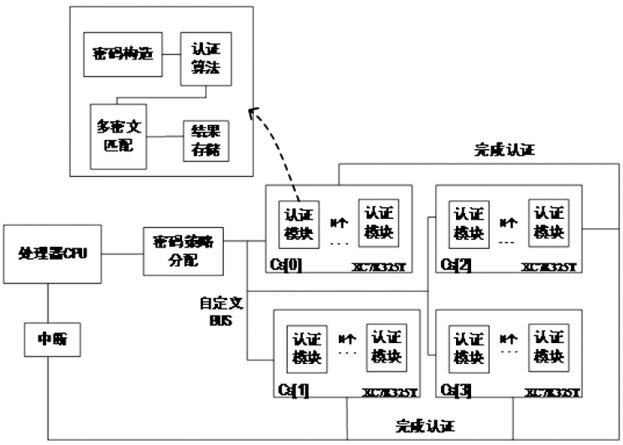

图2 系统整体架构框图

图中每个MySQL密码认证模块包括:密码构造、认证算法、多密文匹配模块。密码构造模块利用从处理器接收到的策略进行密码构造,生成的密码输出给认证算法模块进行认证。认证算法模块中以流水线的方式工作,计算生成校验值。生成的校验值通过多密文匹配模块进行多密文的同时匹配对比,若密文匹配则对该密文的明文密码索引进行保存。在完成全部待校验值匹配后或者完成密码遍历后,当所有并行运算的认证模块中断都有效时,触发处理器中断信号,通知系统已经完成该次策略的计算。

2.2 流水线SHA-1算法实现

流水线型算法实现方式的优点是效率高、吞吐率大,其缺点在于占用的硬件资源较多。从而限制了在单个FPGA内部实现多个并行认证模块的数量。为此我们提出了一种新的流水线型SHA-1算法实现方式,利用FPGA芯片中硬核BRAM模块资源来实现SHA-1算法中Wt数据的写入和读出,不但实现了全流水线架构,而且节省了大部分的寄存器资源。具体实现如下图所示:

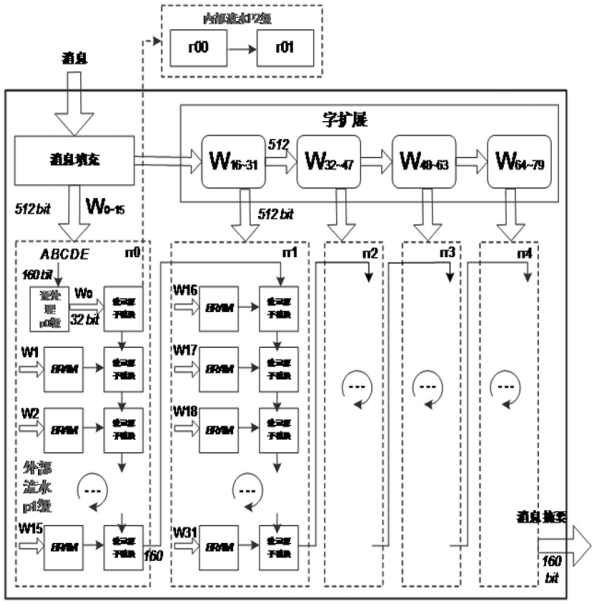

图3　全流水线型SHA-1算法实现框图

图中消息输入经过消息填充模块后，通过字扩展模块将512-bit数据转换为W_0-W_{79}。由公式1可知，从W_{16}开始，每间隔16个Wt是由前面的16个W_t计算成的新的16个W_t，因此，我们将SHA-1算法的80轮循环计算分为5组，分别对应框图中的rr0，rr1，rr2，rr3，rr4。每一组的16个W_t输入字扩展模块生成新的16个W_t输出，以提供给下一组循环进行使用。计算好的W_{16}-W_{79}存放在各自的BRAM中，通过读取各个出BRAM端口b_addr特定地址的数据值W_n，进行每一轮的计算。

全流水线架构的核心在于保证输入的每一时钟周期的消息块M都能处理到，并且最终的模块输出时每一时钟周期输出一个消息摘要值，而这个消息块M又是决定了SHA-1算法中W_0~W_{79}的值。为了保证每一个时钟输入的消息块M能够进行处理，需要不停的存储和转换新的数据Wt，其计算公式如下：

$$W_t=\begin{cases} M_t[32(t+1)-1:32], & (0\le t<16) \\ RotL^1(W_{t-3}\oplus W_{t-8}\oplus W_{t-14}\oplus W_{t-16}\oplus), & (16\le t<80) \end{cases} \quad (1)$$

式中RotL^x(y)表示对括号内的数进行循环左移x位。从式中我们可以看到$W_{16\text{-}18}$可以直接从$W_{0\text{-}15}$得到，而$W_{19\text{-}21}$三个数据需要等待$W_{16\text{-}18}$计算完成后才能计算获得结果，即每间隔三个数据可以开始计算新的三个数据，以此类推。根据这个计算规律，我们设计了Wt数据转换模块，如下图所示，

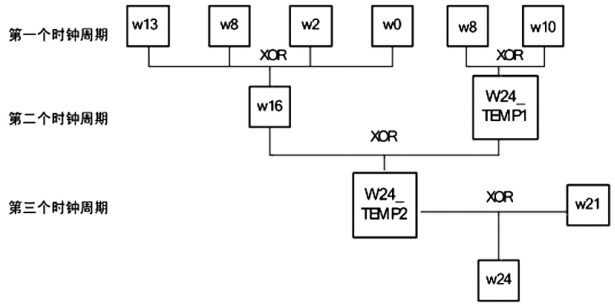

图4　Wt数据转换模块流程框图

利用FPGA芯片XC7K325T内部的六输入查表电路，在第一个时钟周期实现4个输入的异或计算W_{16}，并且仅存储还未计算完成的W_t_TEMP值，只需要三个时钟周期即可完成3个W_t的计算。每个SHA-1算法实现需要用到4个字扩展模块构成流水线架构，如图3所示。

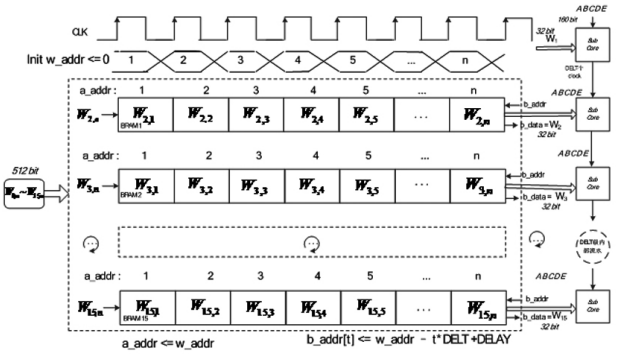

图5　基于BRAM模块实现流水线SHA-1算法框图

图5是基于BRAM模块实现流水线SHA-1算法框图。在全流水线架构中，每一个时钟就输入一个消息块M_n等待计算，rr0根据公式1将M_n分为W0,n~$W_{15,n}$，其中n表示不同时钟输入的消息块M，本设计中用一个计数器变量w_addr进行时序控制，初始化w_addr=0，在每个时钟延w_addr加1。在每一组$W_{0,n}$~$W_{15,n}$输入时，将$W_{2,n}$~$W_{15,n}$共14个数据写入对应的14个BRAM中，写入的端口地址a_addr为该时刻的w_addr的计数值。同时第一个循环子算法模块Sub_core开始计算该Mn的第一个循环，当流水线计算到$W_{t,n}$对应的第t个循环模块时，BRAMt的读取地址b_addr[t]为：w_addr-2*DELT+DELAY，这里DELT为每个循环的内部流水级数，DELAY为所需数据从BRAM读出到当前t模块需要的延时时钟个数。在进行完16轮的计算后将rr0的结果

输出,依次进入rr1,rr2,rr3,rr4进行计算。

2.3 流水线架构优化

为了提高MySQL认证算法FPGA实现的吞吐率(即单位时间内所处理的比特数),可以通过提高时钟频率,减少处理时延来实现。吞吐率的定义如下:

$$Throughput = \frac{block\ size \times frequency}{latency} \quad (2)$$

其中Throughput为吞吐率,block size是指每次处理的数据块比特数,SHA-1算法的数据块大小是512bit,frequency是指工作频率,latency是指处理所需时钟数。

全流水线SHA-1架构的设计可以分为外部和内部两级流水线。外部流水线的每一级从BRAM读取数据Wt,作为SHA-1轮运算子模块的输入。内部流水线设计主要针对SHA-1轮运算子模块,主要是通过循环展开、拆分关键时钟路径来进行优化。我们可以把一个全流水的SHA-1的流水线级数配置为(p0+p1*p2)级来进行优化,其中p0,p1和p2分别表示预计算、外部和内部流水线级数。

内部流水级数p2的设计,主要增加中间寄存器从而降低关键路径的延时,来提高工作频率。外部流水线p1的设计主要根据循环展开轮数来确定。循环展开是基于FPGA实现SHA-`1的常用优化方式,它把两轮或者更多的哈希轮计算展开出来并压缩在一轮来实现[8]。文章[8]通过比较不同展开级数的吞吐率,得出两轮展开的实现方式能够获得最大的吞吐率。经过2轮循环展开的SHA-1算法,其循环计算的路径延迟增长一倍。为此引入预计算的优化方法来减少计算延迟[6]。预计算把一个关键时钟路径的计算量分开成几个互不联系的部分,在流水线架构的前一级预先计算下一级流水线中关键时钟路径的中间值并通过寄存器在两级流水线模块中传递。通过这种方式,把关键路径分割成并行的几部分进行计算,从而大大的减少路径长度,提高算法的最大时钟频率。为此我们配置和设计了一组流水线架构进行性能比较,如下表所示。

根据上面的实验数据,当p0=2,p1=40,p2=2时,即架构配置采用外部40级流水线设计,每级轮运算流水线内部采用2级循环展开、预计算,并且以2级内部流水线,能够得到最好的SHA-1吞吐率和TPS指标。

2.4 多密文匹配模块的实现

在实际的口令恢复应用中,往往会有大量的待解析密钥值需要进行匹配和密码恢复。通常的密文匹配方式仅进行单一密文匹配,没有充分利用通过复杂计算得到的密文,为此,我们设计了的多密文匹配模块,使得应用系统能够同时查找多个密码明文,完成对多个待匹配密文的明文查找,有效地提高了密码恢复的速度。为了实现多密文匹配,我们设计了三个表格进行匹配遍历,通过低位匹配和间接寻址的方式来加快匹配查找的速度,具体流程如下图所示:

表1 不同流水线架构的SHA-1算法性能比较

架构设计 (p0+p1*p2)	时钟频率 (MHz)	时延	Slice Register	Slice Luts	BRAM	Slice	吞吐率 (Gbps)	TPS (Mbps)/slice
2+40*2(325T)	311.7	1	28457	11563	40(36k)	4803	159.590	33.23
1+40*1(325T)	256.1	1	20994	11966	35(36k)	4514	131.123	29.05
2+40*2 (V6)	345.2	1	25849	19303	79(18k)	4908	176.743	36.01
1+40*1 (V6)	236.0	1	20083	16131	58(18k)	4563	120.832	26.48
10*1[6] (V6)	110.668	4.1	8313	12087	–		13.820	
20*1[6] (V6)	107.365	2.05	16202	23523	–		26.815	
Virtex-6[7]	164.7	10				2807	8.4	2.99
Virtex-6[7]	124.7	5				4129	12.8	3.10

注:TPS表示Throughput per slice, 325T表示基于XC7K325T芯片综合实现并且在板测试的数据,V6表示使用Virtex 6 ML605开发板使用ISE4进行综合和实现的数据。

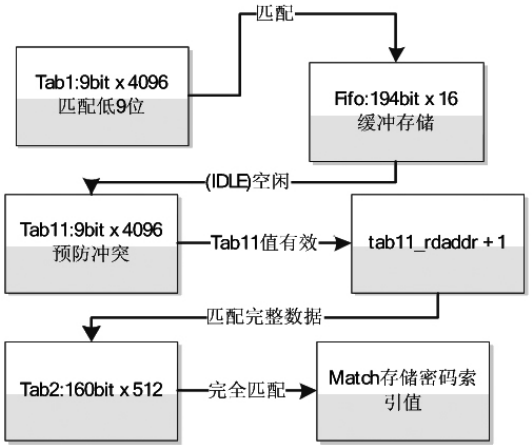

图6　多密文匹配模块的设计流程图

图中密文匹配是通过tab1和tab2，其中tab2存放待解析的160bit密文值，并将待解析密文值的低九位作为表tab1的地址来存放该密文在表tab2的存放地址，即tab1作为tab2的间接寻址索引。若不存在待解析的密文，则tab1上对应的数据设置为无效地址。每一个计算出来的密文先取其低九位作为tab1的地址查找其在表tab1中的数据是否为有效地址，有效则进一步读取tab2中存放的密文进行全字符匹配，若匹配结果相同，则表示密码恢复成功并返回对应明文密码的索引值。为了避免密文低九位冲突的情况，我们还构造了表格tab11。当遇到相同的低九位密文值时，则在tab11表中填写冲突密文所在tab2中的存放地址。发生冲突时，一次可能生成多个需要匹配的密文，所以引入FIFO队列来避免tab1的数据丢失。

为了实现tab1，tab11，tab2以及FIFO模块之间的同步，我们设置了cur_state状态字来表示匹配模块的工作状态，IDLE表示空闲，BUSY表示工作。状态转换如下图所示，

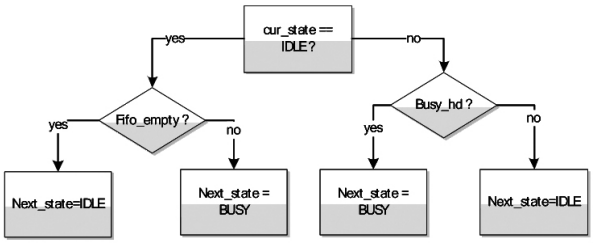

图7　多密文匹配模块的转态转换流程

图中当处于空闲状态时，若fifo_empty为真，则表示FIFO中没有等待匹配的密文值，后续状态为空闲状态；若fifo_empty不为真，则需要读取FIFO进行后续匹配，后续状态为工作状态。当处于工作状态时，需要判断Busy_hd标识，即判断是否完成tab11遍历。遍历完成时，busy_hd恢复为0，则后续状态变为空闲状态。

3　测试结果和分析

整个系统测试的硬件平台是自行设计的一块由1个Znyq-7020和4个XC7K325T　FPGA芯片所构成的子板，其中Znyq-7020主要负责策略分配以及多FPGA芯片之间的调度，XC7K325T主要实现和例化MySQL密码认证算法模块，如下图所示。

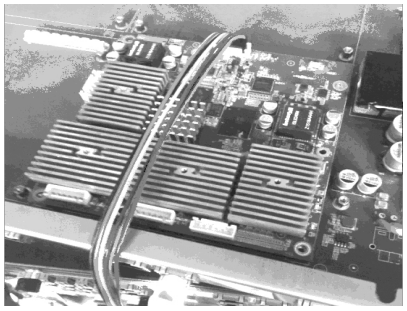

图8　系统测试的硬件平台

在2.3节优化设计的基础上，可以在一个FPGA芯片内部实现多个密码认证模块的并行架构，从而进一步提高密码恢复的速度，如下表所示。

表2　多个密码认证模块的并行架构性能比较

架构配置 (p0+p1*p2)*d	时钟频率 (MHz)	时延	Slice Register	Slice Luts	Block Ram	Slice	吞吐率 (Gbps)	TPS (Mbps)/slice
(2+40*2)*1	311.7	1	33003	13702	60	6328	159.590	25.220
(2+40*2)*6	297.3	1	196583	81676	363	37031	913.306	24.663
文章[6]2*5	113.345	20.5	9786	14562	–	–	14.154	–
文章[6]10*2	110.509	4.1	16612	24335	–	–	27.600	–
文章[6]5*4	113.675	8.2	17398	26042	–	–	28.391	–

注：d表示并行模块的个数；文章[6]5*4，5表示流水的级数，4表示并行模块的个数。

本文设计的运行频率比文章 [6] 高100MHz以上。(2 +40*2)*6 的架构配置吞吐率达到913.306Gbps,是文章[6]的913.306/28.391=32.17倍。此外,从表中可以看到并行架构配置会稍微降低系统的工作频率，但是整体的吞吐率性能提高很大。预处理、2级循环展开以及2级内部流水线的配置在并行实现6个模块后，系统的吞吐率从159.590 Gbps提高到913.306 Gbps,提升了472.2%,转换成认证密码次数为17.84亿/秒。

我们将MySQL密码认证算法的FPGA实现和目前常用的基于GPU众核平台的实现进行性能对比,如下表所示。

表3 基于GPU和FPGA实现认证算法的性能对比

解密平台 密码 认证机制	GTX 690 显卡单张[9] (亿/秒)	GTX 590 显卡单张[10] (亿/秒)	单个 XC7K325T (亿/秒)	单块子板 4 个 XC7K325T (亿/秒)
SHA-1	11.36	10.54	18	72

表中GPU性能数据是基于OpenCL库实现多线程认证算法得到。可以看出,单个XC7K325TX芯片的速度比GTX690的速度提高了6.64亿/秒，比GTX 590的速度提高了7.46亿/秒。如果四个XC7K325T芯片同时运行,系统的整体速率提升至72亿/秒,比GTX 690提高了5.34倍。同时FPGA的整体功耗要低于GPU显卡。

4 结论

本文针对MySQL密码认证算法的FPGA优化实现进行研究,利用并行化、流水线架构来实现高速的MySQL密码恢复应用。对关键的SHA-1、多密文匹配等模块进行全流水线架构设计和实现。首次实现了SHA-1算法全流水线架构,并对其进行优化设计。同时设计了多密文匹配模块,以提高硬件运行速度和计算资源的利用效率。通过优化,单块芯片的口令恢复速度达到18亿个密码每秒,单块子板的认证速度为72亿/秒。

参考文献

[1]The world's most popular open source database[OL],https://www.mysql.com/。

[2]Michail H E, Athanasiou G S, Kelefouras V I, et al. Area-Throughput Trade-Offs for SHA-1 and SHA-256 Hash Functions' Pipelined Designs [J]. Journal of Circuits, Systems and Computers, 2016, 25(04): 1650032.

[3]Makkad R K, Sahu A K. Novel design of fast and compact SHA-1 algorithm for security applications [C], IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2016: 921-925.

[4]Dadda L, Macchetti M, Owen J. The design of a high speed ASIC unit for the hash function SHA-256 (384, 512) [C], IEEE Design, Automation and Test in Europe Conference and Exhibition, 2004,3: 70-75.

[5]Crowe F, Daly A, Kerins T, et al. Single-chip FPGA implementation of a cryptographic co-processor[C], IEEE International Conference on Field-Programmable Technology, 2004: 279-285.

[6]Macchetti M, Dadda L. Quasi-pipelined hash circuits[C], 17th IEEE Symposium on Computer Arithmetic, IEEE, 2005: 222-229.

[7]David Litchfield, Chris Anley, John Heasman and Bill Gri, The Database Hacker's Handbook Defending Database,Wiley Dreamtech India, 2005,Chapter 17.

[8]Lee H, Lee S, Kim J, et al. Parallelizing SHA-1[J]. IEICE Electronics Express, 2015, 12(12): 20150371-20150371.

[9]Michail H E, Athanasiou G S, Theodoridis G, et al. On the development of high-throughput and area-efficient multi-mode cryptographic hash designs in FPGAs[J]. Integration, the VLSI Journal, 2014, 47(4): 387-407.

[10]Michail H, Kakarountas A, Milidonis A, et al. A top-down design methodology for ultrahigh-performance hashing cores [J]. IEEE Transactions on Dependable and Secure Computing, 2009, 6 (4): 255-268.

[11]乐德广,常晋义,刘祥南,等.基于GPU的MD5高速解密算法的实现[J].2010,36(11),P154-155.

[12]潘锟.Android备份文件口令认证机制安全性分析及其改进[D].厦门大学,2014.