文章编号:1000-582X(2012)08-042-06

FPGA 的 Linux 口令密码高速破解模型设计

韩津生[1],林家骏[1],周文锦[2],叶建武[3]
(1.华东理工大学 信息科学与工程学院,上海 200237;2.天津市政府国际经济研究室,天津 300041;
3.东方通信股份有限公司,浙江 杭州 310053)

摘　要:为了提高 linux 口令密码的破解速度,提出了基于数据流的破解核设计思想。对 linux 核心破解模块 MD5 核算法结构进行深入分析,设计了 3 种破解模型,并对其 ALMs 资源消耗和处理速度进行理论分析。在全流水线结构下,按照基于数据流的设计思想,设计 linux 破解核,实现 linux 口令密码的高速破解。实验结果表明:在 EP3SE50F484C4 的 FPGA 芯片上破解 linux 口令,其破解速度达到 24.95×10^4 个/s。在全流水线架构下,基于数据流的设计思想使得流水线上的所有数据块处于高效工作状态,Linux 破解速度大幅提高。

关键词:Linux;口令破解;数据流
中图分类号:TN791　　**文献标志码**:A

Design of Linux password high-speed crack model on FPGA

HAN Jinsheng[1], *LIN Jiajun*[1], *ZHOU Wenjin*[2], *YE Jianwu*[3]
(1. School of Information Science & Engineering, East China University of Science and Technology, Shanghai 200237, P. R. China; 2. Tianjin Municipal Government, International Economic Research, Tianjin 300041, P. R. China; 3. Eastern Communications Company Limited, Hangzhou 310053, Zhejiang, P. R. China)

Abstract: In order to improve the crack speed of linux password, the crack core design idea based on dataflow is proposed. After analyzing core crack module MD5 of linux, three crack models are designed, and their ALMs consumption and run speed are analyzed. Under the structure of full pipeline, based on the idea of dataflow, the Linux crack core is designed to crack linux password fast. Experimental results show that while cracking linux password on FPGA of EP4CE40F29C8, the speed can reach up to 24.95 million/s. In the framework of fully pipeline, the idea based on dataflow makes all the pipeline data blocks in efficient working condition. Linux password crack speed increases greatly.

Key words: Linux; password crack; dataflow

Linux 自问世以来,由于其开放性、源代码公开、丰富的网络功能、良好的用户界面及可移植性等特点,它得到了世界各地数以万计的编程高手和计算机爱好者们的共同开发和维护。如今,Linux 已成为一个稳定可靠、功能完善、性能卓越的操作系统。Linux 可安装在各种计算机硬件设备中,从手机、平板电脑、路由器和视频游戏控制台,到台式计算机、大型机和超级计算机,都可见到 Linux 身影[1]。在服务器领域,约有 90% 的用户使用了 Linux/Unix 操作系统。作为发展成熟和比较成功的操作系统,Linux 将引领未来软件的发展发向。基于源代码开放这一特性,越来越多的大型企业及政府将投入更多的资源来开发 Linux。俄罗斯总理普京已经做出决定,强制政府在 2015 年前将办公平

收稿日期:2011-10-11
基金项目:国家自然科学基金资助项目(60903186)
作者简介:韩津生(1963-),男,高级工程师,主要从事计算机系统方向研究,(Tel)18602290903;(E-mail)daidaidou@163.com

台迁移到 Linux 和开源软件上。

然而，开源软件在带来诸多便利的时候，其安全性的研究也成为人们关注的焦点[2]。目前针对 Linux 口令破解主要采用软件，如 John the Ripper、Crack by Alex Muffett 和 Crack Jack 等。其中 John the Ripper 的破解速度最快，约能达到 7 000 个/s，但仍远不能满足实际应用的需要。

目前微处理器正从双核走向 4 核甚至 8 核，主频突破 2 GHz，位宽从 32 位扩展到了 64 位，内存工作频率也提升到了 800 MHz 以上，虽然微处理器性能大幅提高，但远不能满足破解需要。高速运算的研究仍在不断深入和扩展，或是采用 GPU 加速[3-5]，或是开发高性能引擎[6-7]。事实上，从现代密码诞生开始，高性能通用加密算法引擎的研究和开发就没有停止过，相继推出的密码算法专用芯片或有密码算法硬核的功能芯片，其处理性能都远高于同时代的微处理器，如 MD5 核[8-11] 和 AES 核等[12-15]。ASIC 或 FPGA 方式，能有效提高密码算法处理能力。

由于 FPGA 能够减少电子系统的开发风险和开发成本，缩短上市时间，通过在系统编程、远程在线重构等技术降低维护升级成本。因此在通信、控制、数据计算等领域得到了广泛的应用。2009 年，人们还翘首期盼 32 nmFPGA，2010 年 Xilinx 和 Altera 就先后推出了 28 nm 工艺节点的 FPGA 产品[7-8]，甚至还有一枝新秀的（Archronix）宣布将推出 22 nm 产品[10]。

基于 FPGA，针对操作系统口令密码破解的报道相对较少，但资料[9]表明，基于 FPGA 的 BEE2 架构，用于 UNIX 操作系统的口令密码破解，其破解速度较通用处理器而言，可以获得 10 000 倍以上的速度提升，基本接近应用的要求。较 UNIX 而言，Linux 系统口令加密算法的基本设计思想是一致的，对 Linux 口令加密算法安全性的研究必将成为密码应用领域的研究方向。

1 Linux 加密算法

Linux 操作系统口令加密算法的一种典型方式是基于 MD5 内核来实现。其基本过程为：在用户的登录过程中，系统将用户输入的明文密码提交给单向函数 crypt()，crypt()根据用户名检索用户相关的登录参数，其中较为核心的参数有 Salt 值，Salt 值的位宽为 64 Bt，在用户创建密码时由系统随机产生。Crypt()函数围绕用户密码和 Salt 值等相关参数展开 1 002 次以上的 MD5 运算，最后的哈希值即为口令密文。如果登录时 crypt()生成的口令密文和系统留存的口令密文相匹配，那么，本次登录成功，否则，登陆被拒绝。通过 MD5 哈希，单向函数 crypt()从数学原理上保证了口令密文对口令明文是不可逆的，从而保证口令明文是安全的。

该算法采用了 64 Bt 的 Salt 值，能有效抵御传统的查表方式攻击以及当代的彩虹链攻击；通过 1 002次以上的 MD5 运算，算法的正向加密运算将消耗大量的 CPU 计算资源，使得理论破解时间足以满足密码安全的要求。

使用 MD5 内核的 Linux 系统口令加密算法结构如图 1 所示。在图 1 中，Plaintext 的长度设定为 4，在这种情况下，操作系统在 crypt()中只实现 1002 次基本的 MD5 运算。算法结构基本上分成两部分：数据表生成模块和 MD5 内核模块。数据表生成模块直接为 MD5 内核提供 512 Bt 的数据表，MD5 内核将处理完成的散列值回送给数据表生成模块，数据表生成模块再生成下一组处理数据，如此循环，在完成 1 001 次处理之后，最后一次形成的散列值输出，即 128 Bt 的口令密文。

在数据表生成模块外部，还有 3 组输入数据：口令明文（Plaintext）、Salt 和特征码（$ 1 $）。

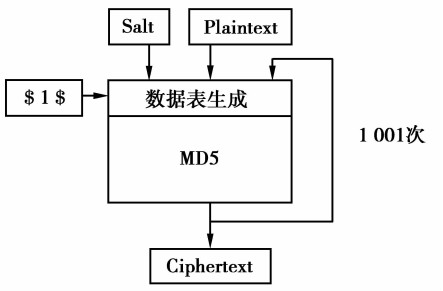

图 1 Linux 口令加密算法结构

以下内容从 Linux 加密算法特征及 FPGA 对通用算法实现方法两方面出发，完成 Linux 口令密码高速破解模型设计实现。

2 基于数据流的 MD5 核设计

MD5 算法的核心是轮变化，它将每一个 512 位消息分组划分为 16 组，对上述进行 4 轮，每轮 16 次操作，共 64 次操作。每轮操作对 a、b、c、d 变量中的后 3 个做一次非线性函数运算 function(b,c,d)，然后将所得结果加上第 4 个变量 a、一个分组 M 和一个常数 T，再将所得结果向左循环移动 S 位，并加上 b。

function(b,c,d)为组合逻辑运算，每轮操作的

表达式有差异：

第一轮为：$F(b,c,d)=(b\&c)|((!\ b)\&d)$；

第二轮为：$G(b,c,d)=(b\&d)|(c\&(!\ d))$；

第三轮为：$H(b,c,d)=b\hat{}c\hat{}d$；

第四轮为：$I(b,c,d)=c\hat{}(b|(!\ d))$。

为描述的方便，以第一轮为例，function$(b,c,d)=$F(b,c,d)表达式，得到公式(1)

$$b\leftarrow b+(a+(F(b,c,d)+M+T)\ll S,\quad (1)$$

在公式(1)中，表达式包含5个数的加法运算，由于StratixIII系列的FPGA在共享的算术模式下支持3输入1输出的加法运算，按照这一硬件特征构建的深度1节点模型如图2所示。

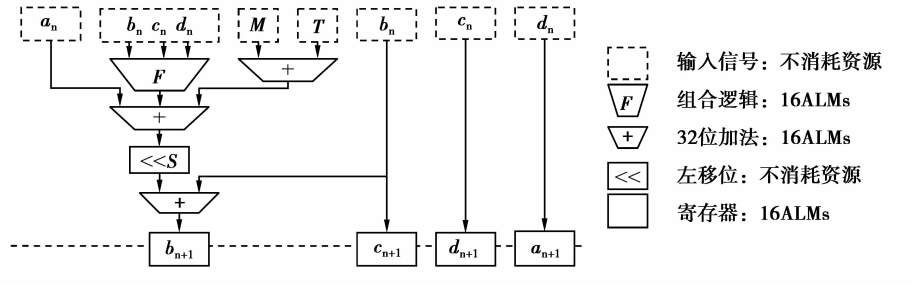

图2 深度1节点模型

图2可表示为公式(2)

$$b_{n+1}=b_n+(a_n+F(b_n,c_n,d_n)+M+T)\ll S,$$
$$c_{n+1}=b_n,d_{n+1}=c_n,a_{n+1}=d_n。\quad (2)$$

在图2中，存在3组加法运算，其中两组加法不带寄存器输出，共消耗32个ALMs，另外一组加法运算寄存器输出和b_{n+1}一起共消耗16个ALMs。F示意的是3输入的32位组合逻辑运算，共消耗16个ALMs，不带寄存器输出。ALMs在通常模式下支持ALUTS和寄存器的分离，c_{n+1}、d_{n+1}、a_{n+1}的输出理论上需要消耗48个ALMs，但前面的加法和组合逻辑部分消耗的ALMs中共有96个寄存器没有被占用。所以，在最好的情况下c_{n+1}、d_{n+1}、a_{n+1}会使用这些资源，从而不额外消耗ALMs。综上所述，图2模型最大ALMs消耗为112，最小ALMs消耗为64，实际消耗与设计实现有关。

为提高节点吞吐量，基于流水线设计思想技术完成对图2节点的分段，得到如图3所示的深度2节点模型。

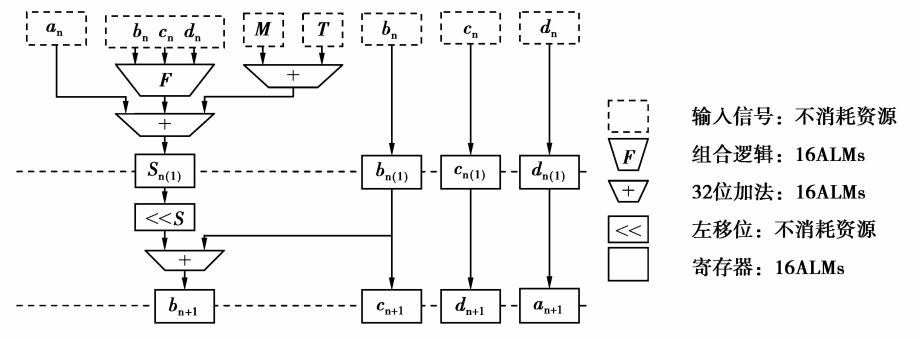

图3 深度2节点模型

图3中，b_{n+1}可表示为公式(3)

$$b_{n+1}=b_n+(a_n+F(b_n,c_n,d_n)+M+T)\ll S=$$
$$S_{n(1)}\ll S+b_{n(1)}。\quad (3)$$

在图3中，组合逻辑F消耗16个ALMs，3组加法消耗48个ALMs。从b_n、c_n、d_n到b_{n+1}、c_{n+1}、d_{n+1}二级缓冲，消耗192个寄存器，相当于96个ALMs。在理论上，组合逻辑F和一个不带寄存器输出的加法可以分离出64个寄存器；当FPGA中ALMs的寄存器不够而ALUTs较为充裕时，工具软件会自动启用LUT寄存器模式，通过两ALUTs额外构建一个寄存器，这样，一个ALMs将实现3寄存器的功能，43个ALMs可以提供129个寄存器。所以深度2节点模型理论上ALMs的最大消耗为160，最小消耗为107。

图2所示的节点模型在关键路径上存在三级加法级联，按照流水线的设计思想，采用深度3节点模

型可以获得更高的处理速度，其模型如图 4 所示。

图 4 中，b_{n+1} 可表示为公式(4)

$$b_{n+1}=b_n+(a_n+F(b_n,c_n,d_n)+M+T)\ll S=(a_{n(1)}+FT_{n(1)}+MT_{n(1)})\ll S+b_{n(1)}=ST_{n(2)}+b_{n(2)}。\tag{4}$$

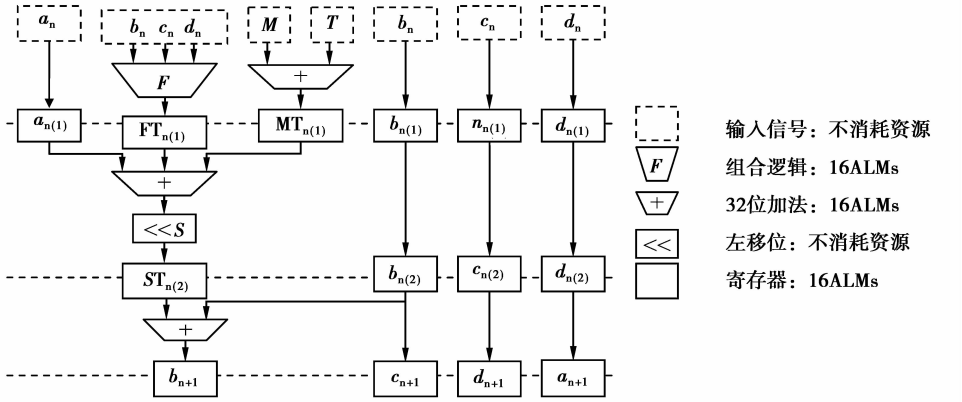

图 4　深度 3 节点模型

在图 4 中，3 组加法和组合逻辑均带寄存器输出，共消耗 64 个 ALMs。从 b_n、c_n、d_n 到 b_{n+1}、c_{n+1}、d_{n+1} 3 级缓冲，如果全部通过 ALMs 的寄存器来实现，那么 ALMs 的最大消耗为 144 个。如果一个 ALMs 按照 3 寄存器的功能实现，那么，ALMs 的资源消耗为 96 个；从 a_n 到 $a_{n(1)}$ 需要消耗 16 个 ALMs。图 4 模型理论上 ALMs 的最大消耗为 224，最小消耗为 176。

通过分段处理得到的深度 2 和 3 节点模型能有效提高节点的处理效率，提高节点数据处理的吞吐量，但由于数据缓冲会增加资源消耗；通过分段能提高节点的工作频率，但并不意味着单位资源处理能力单调增加。表 1 为三模型的核心指标理论数据分析。

表 1　三模型核心指标

参数	ALMs	节点理论处理速度	单位资源理论效率
深度 1 节点模型	[112,64]	1.0	[0.57,1]
深度 2 节点模型	[160,107]	1.5	[0.6,0.89]
深度 3 节点模型	[224,176]	3.0	[0.85,1.09]

在表 1 中，以基本加法运算处理时间为基准来衡量节点理论处理速度。在深度 1 节点模型中，关键路径上存在 3 次级联的加法运算，而经过分段后的深度 2 和 3 节点模型，各段内只包含 2 或 1 个加法运算，所以当深度 1 节点模型的理论处理速度为 1 时，深度 2 和 3 节点模型的理论处理速度为 1.5 和 3。这意味着较深度 1 节点模型而言，深度 2 和 3 节点模型的理论工作频率将提高 1.5 和 3 倍。

单位资源理论效率是以深度 1 节点模型在资源消耗为 64 为基准点来计算的，从数据来看，深度 2 和 3 节点模型的单位资源处理能力并没有显著的提高。这说明，通过对节点的分段处理能有效提高单核的工作频率，增加单核的吞吐量，但不一定能提高单位资源的处理能力，改善 FPGA 的整体性能。

三节点模型可以按照几乎相同的结构构建高新能的 MD5 核，通过 64 次轮运算的节点前后级联，可以构建出全流水线的核结构。在节点 1 模型下，核的流水线深度为 65；在节点 2 模型下，深度为 129；在节点 3 模型下，深度为 193。流水线的最后 1 级为数据尾处理。

3　基于数据流的 Linux 破解核设计

高性能 Linux 破解核设计需要解决两个问题：单核处理能力和最小核资源消耗，并最终体现 FPGA 单位资源的处理能力，实现 FPGA 对 Linux 口令加密算法的处理性能。基于数据流的设计方法能有效解决这个问题。事实上，数据处理的过程就是数据在移动中不断变化的过程。在全流水线方式

下,口令数据需要跟随自身的时间隧道整体移动,这些数据量的大小及其移动速度是核设计需重点解决的问题。在设计前期,数据流的思想能完成对算法设计的整体评估,并构建出各微核的处理模型,通过对微核及接口的合理优化,实现 Linux 破解核资源消耗和工作频率的最优配置。

Linux 破解核核主要由 2 部分组成:Linux 动态表生成模块和高性能 MD5 处理引擎。Linux 动态表生成模块完成特征参数“$ 1 $”、Salt、Plaintext 及 MD5 的 128 位 HASH 值到 MD5 的 512 位消息分组的转换。高性能 MD5 核依赖于全流水线的结构,按照数据流的设计思想,动态表生成模块也必须采用全流水线结构,这样才能和 MD5 核的工作保持完全同步的关系。基于数据流的设计思想实现的全流水线 Linux 破解核,其结构如图 5 所示。

对于 4 位长度的口令密码,每次 MD5 运算的 ABCD 值都为初始值,只是在处理的过程中 512 位消息分组在变化,其变化来自于 2 个方面:前一次处理得到的 128 位 HASH 和口令密码的变换;Linux 动态表生成规则的变换。在全流水线工作方式下,流水线中存在和流水线级数等同的口令依次在核中循环处理,直到完成 1002 运算后才顺序输出,最终得到各自的 Ciphertext;与此同时,新的一组口令密码依次同步打入到加密核中。

对于某特定的口令,自身数据随自身的时间隧道同步流动,这意味着构成 Linux 破解核的 Linux 动态表生成模块和高性能 MD5 处理引擎 2 模块存在高度耦合,这种耦合关系导致了 MD5 核不再是通用的 MD5 结构,不能完成通用的 MD5 运算。为使 Linux 破解核能正常工作,还需要配置一些外围模块才能满足应用要求,其相关实现方法不是研究的内容。

4 实 验

按照图 5 结构,通过 StratixIII 系列中的 EP3SE50F484C4,分别按照深度 1、2、3 节点模型设计实现口令长度为 4 的 Linux 破解核,其核心指标如表 2 所示。

表 2 Linux 破解核指标

参数	ALMs (个)	M9K (个)	流水线深度	工作频率 (Mhz)	单核处理能力 (个/s)	单位 ALMs 处理能力 (个/s)	FPGA 处理能力 (个/s)
深度 1 节点模型	6 739	49	66	120	119 760	17.77	239 520
深度 2 节点模型	13 790	49	130	180	179 640	13.02	179 640
深度 3 节点模型	16 600	49	194	250	249 500	15.03	249 500

表 2 中,单核处理能力是指单位时间处理口令长度为 4 的密码个数;单位 ALMs 处理能力是指每个 ALMs 对口令长度为 4 的密码等效处理个数;FPGA 处理能力是在本 FPGA 下,可实现的密码处理个数。

在 EP3SE50F484C4 中,存在 19000 个 ALMs 和 400 个 M9K,M9K 在本次核设计中占比很低,不为关键资源。深度 1 节点模型对 ALMs 消耗占比为 35%;深度 2 节点模型对 ALMs 消耗占比为 73%;深度 3 节点模型对 ALMs 消耗占比为 87%。所以,在本 FPGA 中,理论上可以放置两个深度 1 节点模型的 Linux 破解核,对于深度 2 和 3 节点模型只能放置一个。

在暴力破解条件下,通过表 2 展示的单 Linux 破解核每 s 能完成 24.95 万个 4 位口令的加密处理。

5 结 论

以 MD5 为内核的 Linux 口令加密算法,通过 1 002次的 MD5 运算,加大了密码破解的难度,增强了密码的安全性。在算法设计初期,其计算机处理水平和相关技术还不足以对这种口令的安全性产生威胁,但随着计算机处理能力的提高以及微电子技术的飞速发展,这种设计上的口令安全性将趋于弱化。目前,在通用计算机上,软件破解速度的评估值还只有 7 000 个/s。

基于数据流设计思想实现的全流水线 Linux 破解核,通过在 EP3SE50F484C4 上的硬件实现,其深度 3 节点模型的工作频率可达 250 MHz,其单核破解速度可达 24.95×10^4 个/s,较通用计算机而言,破解处理速度提高 35 倍以上。比较结果显示,本 Linux 破解核有效地提高了单核处理能力、FPGA

的硬件资源利用效率及整体处理能力。

参考文献：

[1] Ge W, Luo M. The present situation and future of linux [J]. Computer Knowledge and Technology, 2010, 6(8): 2027-2028.

[2] SANS Institute. Securing Linux/unix [EB/OL]. (2011-05-06) [2012-02-01]. https://www.sans.org/security-training/securing-linux-unix-76-mid.

[3] Jarvinen K, Tommiska M, Skytta J. Hardware implementation analysis of the MD5 hash algorithm[C] // Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Jan, 3-6, 2005, Big Island, HI, USA. [S.l.]: IEEE, 2005: 297-306.

[4] Wang Y L, Zhao Q X, Jiang L H, et al. Ultra high throughput implementations for MD5 hash algorithm on FPGA [J]. Lecture Notes in Computer Science, 2010, 5938: 433-441.

[5] Trang H, Van loi N. An efficient FPGA implementation of advanced encryption standard algorithm [C] // Proceedings of the 2012 IEEE International Symposium on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, Feb. 27-March 1, 2012, Ho Chi Minh City, Vietnam. [S.l.]: IEEE, 2004: 1-4.

[6] Rais M H, Qasim S M. A novel FPGA implementation of AES-128 using reduced residue of prime numbers based S-box [J]. International Journal of Computer Science and Network Security, 2009, 9(9): 305-309.

[7] http://china.xilinx.com/publications/prod_mktg/7-Series-Product-Brief.pdf, 2011.

[8] Altera International Limited. Stratix V FPGA [EB/OL]. [2011-02-01]. http://www.altera.com.cn/products/devices/stratix-fpgas/stratix-v/stxv-index.jsp.

[9] Mentens N, Batina L, Preneel B, et al. Time-memory trade-off attack on FPGA platforms: UNIX password cracking[J]. ARC, 2006, 3985: 323-334.

[10] 郭艳杰. 借光英特尔 22nm 工艺 Archronix 打造业界最高性能 FPGA[EB/OL]. (2010-11-11)[2012-02-01]. http://news.eefocus.com/article/10-11/531131289461411.html?sort=1771_1775_1877_0.

[11] Helion Technology. High performance MD5 hash core for Xilinx FPGA [EB/OL]. [2012-02-01]. http://www.heliontech.com/downloads/md5_xilinx_helioncore.pdf.

[12] Kris G, Pawel C. Comparison of the hardware performance of the AES candidates using reconfigurable hardware [EB/OL]. [2012-02-01]. http://teal.gmu.edu/crypto/AES_gaj.PDF.

[13] 李涛，成晓雄，王文华，等. 一种 GPON-AES 的 FPGA 优化实现[J]. 电信科学，2009，25(12)：53-57.
Li Tao, Cheng Xixiong, WANG Wenhua, et al. An optimized FPGA implementation of GPON-AES[J]. Telecommunications Science, 2009, 25(12): 53-57.

[14] Hu G, Ma J H, Huang B X. High throughput implementation of MD5 algorithm on GPU [C] // Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications, Dec. 20-22, 2009, Fukuoka, Japan. [S.l.]: IEEE, 2009: 1-5.

[15] 刘凯，车明，秦存秀. 一种高吞吐量 MD5 算法的 FPGA 实现[J]. 微处理机，2008，29(1)：188-191.
Liu Kai, Che Ming, Qin Cunxiu. A high throughput FPGA implementation of MD5 algorithm [J]. Microprocessors, 2008, 29(1): 188-191.

（编辑 侯 湘）