一种新型的混合异构口令恢复系统

李 斌[1)] 周清雷[2)] 斯雪明[1)]

[1)](信息工程大学数学工程与先进计算国家重点实验室 郑州 450001)

[2)](郑州大学信息工程学院 郑州 450001)

摘 要 口令恢复作为电子取证、信息情报获取和犯罪记录审查的重要手段,是对互联网信息进行监控、维护网络信息安全的关键环节之一.传统的口令恢复系统主要以 CPU 和 GPU 为主,体系结构单一、破解效率低,无法满足用户的计算需求.由此,该文提出了一种新型的混合异构口令恢复系统.结合拟态计算的思想,通过 CPU、GPU 和拟态计算机搭建混合异构系统,建立多维可重构体系.并在拟态计算机上设计相关高速口令穷举算法、万兆网络字典传输协议和全流水可重构加密恢复算法,提高破解效率.同时,根据具体口令恢复应用的 PMC(Processing-Memory-Communication)特征,动态调整系统结构,均衡向下分配口令空间,使整个系统高效地完成加密恢复任务.实验分析和结果表明,与传统 CPU 系统相比,该文系统在破解速度上提高 18.84 倍~84.94 倍,在能效比上提高 3.07 倍~15.73 倍,与传统口令穷举和字典破解模式相比,其恢复效率有所提升,且能较好地支持异构系统.

关键词 拟态计算;拟态计算机;口令恢复;混合异构;口令穷举;字典传输;负载均衡

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2018.02804

A New Hybrid Heterogeneous Password Recovery System

LI Bin[1)] ZHOU Qing-Lei[2)] SI Xue-Ming[1)]

[1)](*State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, Zhengzhou* 450001)

[2)](*School of Information Engineering, Zhengzhou University, Zhengzhou* 450001)

Abstract Password recovery is an important means for electronic forensics, information acquisition and criminal record review. It is one of the key links to monitor and maintain the security of Internet information. The application of password recovery mainly includes password recovery of encrypted documents, operating systems, network protocols, disk encryption, database encryption, and network applications, covering all aspects of computer systems. The diversity of encryption algorithms, the timeliness of applications and the density of computation contribute to the high demand for computing capacity and solution of various encryption algorithms. However, traditional password recovery system is mainly based on CPU and GPU, and its architecture is simple, cracking efficiency is low, can't meet the user's computing needs. The CPU is limited by its processing architecture, and the computation speed of the encryption algorithm is low, while the GPU has a small local memory on the single core, and it is not very good for the encryption algorithm containing the S-box. Therefore, this paper presents a new hybrid heterogeneous password recovery system based on the mimic computing. Mimic computing aims at high-efficiency computing, with the features of resource allocation on-demand, variable structure, and flexible computing. Mimic computing can conveniently integrate various types of computing resources and storage resources into the architecture and form an organic entirety and use the aggregation of resources to deal

收稿日期:2017-09-06;在线出版日期:2018-03-27.本课题得到国家重点研发计划项目(2016YFB0800100,2016YFB0800101)、国家自然科学基金项目(61250007)、国家"八六三"高技术研究发展计划项目基金(2009AA012201)资助.**李 斌**,男,1986 年生,博士研究生,主要研究方向为高性能计算和信息安全.E-mail:cctvlibin@163.com.**周清雷**,男,1962 年生,博士,教授,博士生导师,主要研究领域为信息安全、自动机理论及计算复杂性理论.**斯雪明**,男,1966 年生,博士,副教授,主要研究方向为密码学、网络安全和高性能计算.

with the different computing combination requirements contained in the application to achieve application-oriented high-performance computing. Combined with the idea of mimic computing, this system is constructed by CPU, GPU and mimic computer, to establish multi-dimensional reconfigurable system. It works in a software-hardware collaborative manner, makes full use of the versatility of CPU to handle task scheduling and password distribution, and uses of the parallelism and easy programming of GPU to rapid deployment of password recovery tasks, and uses of the reconfigure ability and low power consumption of mimic computer to achieve high-efficiency password recovery computing. Secondly, taking the mimic computer as the core computing component, through in-depth analysis of the various types of encryption algorithm features and crack modes, reasonable layout of the on-chip logical structure, and design related high-speed password exhaustion algorithm, 10 Gigabit network dictionary transmission protocol and full pipeline reconfigurable encryption recovery algorithm to improve cracking efficiency. Meanwhile, according to the PMC (Processing-Memory-Communication) features of the specific password recovery application, the system structure is dynamically changed, and the password space is allocated in a balanced manner to complete the cooperation of the heterogeneous systems and improve the recovery efficiency. Thirdly, the system supports parallel computing of multi-tasking and multi-password attack modes. According to different cracking tasks and modes, the interconnection of computing components can be changed, and different strategies and structures can be used to improve the computational efficiency. Finally, a comprehensive evaluation and analysis of the system is carried out from various perspectives, including performance comparison, cracking efficiency comparison, load balancing analysis, structural transformation analysis and system composition analysis. The experimental analysis and results show that compared with the traditional CPU system, this system improves the cracking speed by 18.84 times—84.94 times and improves the energy efficiency ratio by 3.07 times—15.73 times. Compared with traditional password exhaustion and dictionary cracking mode, the recovery efficiency has also improved, and can support heterogeneous systems better.

Keywords　mimic computing; mimic computer; password recovery; hybrid heterogeneous; password exhaustion; dictionary transmission; load balance

1　引　言

随着社会信息化的快速发展，人们日常生活不断网络化，资产不断数字化，身份认证成为了保障用户信息安全的重要手段[1]. 虽然基于图像、视觉和指纹等的认证方法已经开始大量应用于认证用户身份，但人可记忆的文本口令认证方法以其方便的应用和极低的成本仍被广泛地应用着[2]. 同时，现今流行的口令加密算法多达 200 种，分析不同的密码技术和恢复算法，对加密信息系统的攻击和加密信息的还原至关重要. 加密算法的多样性、应用的时效性以及计算的密集性均对计算能力和解算种类提出了很高的要求.

口令恢复应用部署于互联网出口，是不良信息过滤系统应用层处理的关键环节之一，是电子取证的重要手段. 随着互联网文档数量的增多，涉及秘密信息的加密文档也随着人们信息安全意识的增强在不断增多. 但是，现有的计算结构和计算能力难以满足加密文档口令恢复的需求，基于传统 CPU 架构的口令恢复受限于其对密码算法的计算速度，只能破解口令复杂度较低的口令，且每秒能够验证的口令数比较少，计算效率远远不能满足网络电子取证的需求. 为此，很多学者利用 CUDA 编程，在 CPU+GPU[3-11] 上实现了高速口令恢复系统. 然而 GPU 虽然加速了口令恢复的计算，但其工作在较高的频率，造成了系统较大的功耗，并且 GPU 的发展更多是支持浮点运算，对于口令恢复问题，只能依靠核数和工作频率的提升来提高加速比. 由于 GPU 单核上的 Local Memory 比较少，对于 RC4、DES 和 AES

的加速,GPU 都不擅长.同时,互联网用户数量随时间周期变化十分明显,这就造成加密文档数量随时间明显波动.加密恢复任务计算量大、计算量周期变化明显是现有计算结构难以应对的,需要寻求更加高效能的计算结构.

拟态计算[12]依托可重构技术,以高效能计算为目标,从体系结构创新入手,对高性能计算在多个典型领域的应用、结构和效能关系进行了深入分析,提出了"应用决定结构,结构决定效能"的理念.拟态计算通过识别应用的需求、应用的变化,同时感知系统中可以利用的处理资源,依据尽可能高效的原则,构建出适合于应用需求的处理结构,并且该结构随着应用的变化,如计算进展阶段、处理负荷等的变化,而进行结构的主动变更,达到应用决定结构,结构决定效能的目的.拟态计算以拟态计算机为处理核心,通过高速网络将拟态计算机、CPU、GPU、DSP 和 CELL 等性能和功能各异的计算部件以一定的耦合方式连接形成并行计算环境[13],充分利用程序和计算部件的异构性,各尽潜能,协同完成计算任务.拟态计算面向领域应用,根据不同的算法特征,动态调用计算资源,可按需重构性能、结构各异的系统以满足不同领域的计算需求,非常适用于口令恢复计算任务.

为了有效地将拟态计算应用在口令恢复领域并构建灵活可变的体系结构,提高系统计算效率,需要解决以下几个问题:(1)如何对口令恢复领域进行算法特征分析,构建高性能的算法;(2)如何选取合适的计算部件,并充分利用计算资源,提高系统的效能;(3)如何满足口令恢复领域应用的灵活性和可拓展性.为此,本文提出了一种新型的混合异构口令恢复系统,结合 CPU、GPU 和拟态计算机等异构计算部件,以软硬件协同方式工作,充分利用 CPU 的通用性,处理复杂的事务管理和调度,利用 GPU 的并行性和易编程性,快速部署高性能算法,利用拟态计算机的可重构性和低功耗,实现高效能的计算.并通过分析各类加解密算法的 PMC(Processing-Memory-Communication)特征,合理分配系统资源,以合适的结构匹配应用.同时,以拟态计算机为核心计算部件,设计口令穷举算法和万兆网络字典传输协议,采用多流水线并行的方式实现口令恢复算法,进一步提高系统计算性能.其次,通过负载均衡策略,动态划分口令空间,向下合理地分配口令穷举规则和字典文件,完成异构系统的协作,提高破解效率.最后,根据不同的破解任务和破解模式,变换系统结构,高效地完成加密恢复任务.

本文第 1 节描述口令恢复应用的背景和主要的研究内容;第 2 节介绍拟态计算的概念和口令恢复的相关研究工作;第 3 节给出基于拟态计算的混合异构口令恢复系统设计流程;第 4 节对本文提出的方案进行测试和分析;最后,第 5 节对本文进行总结和展望,提出下一步的工作.

2 相关工作

2.1 拟态计算简介

包含软件和硬件变体的多维重构函数化体系结构称为拟态架构,它能根据动态参数选择生成多种功能等价的可计算实体,实现拟态变换.对于一个确定的可计算问题,在拟态架构中可以由多种功能等价、计算效能不同的硬件变体和软件变体来实现,动态地选择与使用这些变体,计算效能可以达到最优化,即为拟态计算[12].

拟态计算以实现高效能和高性能计算为目的,将计算结构作为高阶函数,在任务处理的全过程中通过感知自变量,动态地选择或生成最优效能解算结构集合.拟态计算模型可抽象为七元组 $MCM=(APP,MA,OA,SE,EF,KB,DS)$.其中,$APP$ 表示应用服务;MA 表示元结构;OA 表示目标结构;EF 表示评价函数;SE 表示系统状态;KB 表示决策知识库;DS 表示决策函数.DS 综合利用 APP、MA、SE、EF、KB 等要素,以 EF 趋向于最优为主要决策依据,得到某一应用在某时刻的高效能结构 OA.

按照拟态计算原理构造的计算机称为拟态计算机[14].拟态计算机以态变结构、软硬件结合实现基于效能的计算,能够针对不同应用需求,通过改变自身结构来提高效能.测试表明,拟态计算机典型应用的能效,比一般计算机可提升十几倍到上百倍,高效能特点显著.拟态计算机高效能、高性能的特点使它非常适合计算密集型的大数据量处理任务.2013 年 9 月,我国成功研制出世界首台结构动态可变的拟态计算机原理样机,其结构如图 1 所示.

拟态计算机原理样机主要由 4 块可重构 FPGA 构成,通过系统总线板级互联,并由控制单元进行控制,片上集成了千兆和万兆通讯网口、PCI-E 接口、内存、FLASH、USB 等多种接口,可直接访问盘阵,并与外部进行通信.FPGA 作为拟态计算机原理样机的主要组成部件之一,具有很强的并行计算和位运算能力[15],可定义任意宽度的寄存器,并可根据不同的应用进行重构,非常适合计算密集型的口令恢复任务.

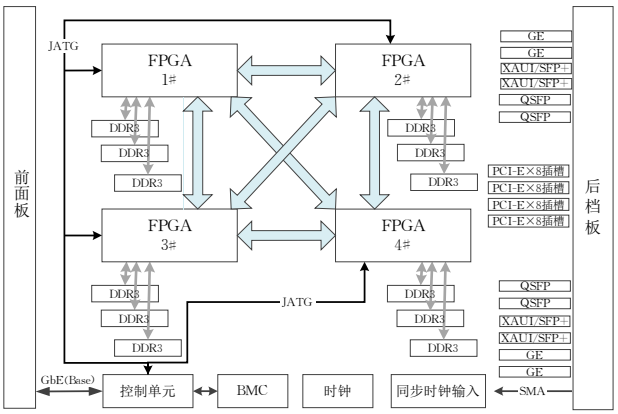

图 1　拟态计算机原理样机的结构图

2.2　口令恢复

在互联网世界，口令加密是保护个人信息安全的最主要方法之一[16]，有着不可撼动的地位. 时至今日，口令在越来越多的信息系统中得到加强. 基于口令验证的加密信息系统通常以 MD5、SHA1、DES、AES、RC4、ECC 等算法为核心. 加密信息的还原方法主要依靠穷举破解、字典攻击和时空折中攻击等口令还原方法[4]. 当前对口令恢复应用的研究主要集中在加密恢复算法的优化加速、口令猜测攻击的算法和计算平台的搭建.

在算法优化加速方面，许多学者在 GPU 和 FPGA 上开展了研究工作. 在 GPU 方面，文献[3]在 GPU 上实现了 MD5 和 SHA1 算法；文献[4-5]在 GPU 优化上实现了 MD5 Crypt 破解算法；文献[6-9]分别使用 GPU 实现了 Office 2003、PDF、ZIP 和 RAR 口令恢复算法；文献[10]分析了 PDF、DOC 和 ZIP 三类加密文档在 CPU 和 GPU 上的恢复效率；文献[11]在 GPU 上实现了 Office 2007 和 Office 2010 加密恢复算法. 在 FPGA 方面，文献[17]介绍了多种提高口令破解效率的方法，并提到使用 FPGA 异构部件可以获得较好的性能和较低的代价；文献[18]在 Xilinx XC2VP20-5 上实现了 RC4 算法；文献[19]在 Altera Stratix II GX EP2SGX90FF 上实现了 32 级流水线 MD5 算法；文献[20]在 Xilinx Virtex-6 LX240T 上实现了高吞吐量的 SHA1 算法；文献[21]分别在 Epiphany 微处理器和 Zynq 系列 FPGA 开发板上实现了 Bcrypt 破解算法，并与 CPU 在性能、价格方面进行了对比；文献[22]在 GPU 和 FPGA 上实现了 Bcrypt 和 Scrypt 两种哈希密码的破解；文献[23]在 Stratix III EP3SE50F484C4 上实现了 MD5 Crypt 算法；文献[24]结合 GPU 和 FPGA 异构框架实现了 PDF 恢复算法；文献[25]提出了 WPA/WPA2 的 FPGA 实现结构；文献[26]在 FPGA 上实现了低功耗的 WPA2 口令恢复算法. 但是，上述方案仅针对某单一应用，结构简单，且无法兼顾计算系统的性能、效能和灵活性. 同时 GPU 功耗高，能效比低，具有“功耗墙”的问题. 而 FPGA 实现的算法性能不一，在时钟频率和资源上还有提升空间.

在口令猜测攻击的算法方面，文献[27-28]提出了基于上下文无关文法的 SLD 模式口令破解方法，与传统的 John the Ripper 工具相比在破解速度和成功率上有较大的提升. 文献[29]提出了基于语义的口令破解方法，降低了对训练集的依赖性. 文献[30]提出了 TDT 口令破解模型，并结合不同攻击方式设计了混合口令破解平台. 文献[31]提出了一种基于结构划分概率的口令攻击方法，提高了高概率口令结构的产生. 文献[32]研究了中国拼音与国外英语在设置密码方面的差异，优化了基于上下文无关语法和马尔可夫模型两种密码猜测方法，在一定程度上提高了命中率. 文献[33]对比分析了不同的口令猜测模型，并指出马尔科夫模型要优于上下文无关文法模型，且提出口令概率阈值是评估口令集的一个重要指标. 文献[34]通过对大量真实口令集的统计分析，指出口令的频率分布符合 Zipf 分布，并给出了公式：$f_r = C/r^s$，其中 r 表示口令排名，f_r 表示排名为 r 的口令频次，s 和 C 为常数. Zipf 分布有效地刻画了口令集中口令使用频率和排名之间的相互关系，为评估口令猜测攻击模型提供了理论基础. 文献[35]提出了一种定向口令在线猜测攻击方法，通过利用用户的相关个人信息，包括姓名、生日、手机号码和用户已泄露的口令等，进行关联分析，减小口令搜索空间，有针对性地猜测出给定用户的口令. 该方法详细给出了四种定向口令猜测算法，以用户信息替换口令字符或规则地对口令字符进行添加、插入、删除和变形等，产生新的口令，其攻击效率有了明显提高. 文献[36]通过对大规模的真实口令数据进行分析挖掘，获得若干统计特征及口令设置规则，并将这些统计特征和规则成功应用于口令的脆弱性评估及恢复系统. 文献[37]提出了一种基于样本的模拟口令集生成算法，该算法利用小规模真实口令样本，借助上下文无关文法学习生成概率模型，采用基于扰动的方式产生大规模用户口令集合，在一定程度上提高了口令覆盖率，且口令结构更符合真实情况. 但是，上述部分方案随着候选口令数量的剧增，占用空间越来越大，导致口令覆盖率低，且训练过程较为复杂，匹配速度无法满足高性能计算的需求. 同时，口令破解的效率不仅依赖于高概率

口令本身,还需要高性能算法的支持.

在计算平台方面,现有的口令恢复平台[38]主要有 John the Ripper、Elcomsoft、Hashcat 和 Wrathion,这些平台主要运行在 CPU 和 GPU 上.同时,文献[38]使用 BOINC(Berkeley Open Infrastructure for Network Computing)架构搭建了基于 CPU 和 GPU 的分布式口令恢复系统.文献[39]通过 MPI 和 CUDA 混合编程框架,实现了一个口令恢复平台.但是这些平台都是基于 CPU 和 GPU 实现的,无法很好地支持 FPGA 等其他异构部件,可拓展性较差.

综上,本文结合口令恢复方面的研究,深入剖析口令恢复算法特征,结合 CPU、GPU 和拟态计算机搭建混合异构系统,构建动态可变的体系结构,实现口令恢复领域内应用重构的灵活性,进而提高口令破解的效率.

3 基于拟态计算的混合异构口令恢复系统设计

3.1 系统整体框架

3.1.1 系统组成

结合拟态计算的思想,采用(CPU+GPU+MC)(Mimic Computer)异构计算部件搭建混合异构口令恢复系统(Hybrid Heterogeneous Password Recovery System,HHPRS),并通过恢复加密的 MS Office 2010 文档、MS Office 2007 文档、WinZip 文档、PDF 文档及 Linux 操作系统 MD5 Crypt 算法等,针对不同应用算法,实现系统各异构部件的拟态变换,提高系统性能、效率和能效比.

如图 2 所示,HHPRS 由客户端、盘阵和 3 种计算部件构成:CPU 服务端、GPU 服务端和 MC 服务端.其中,客户端主要负责任务的分发及维护、负载均衡规则的下发、系统结构的决策变换、中间结果及正确结果的显示和入库.盘阵用于存放字典文件.CPU 服务端、GPU 服务端和 MC 服务端主要负责加密文档的恢复任务.

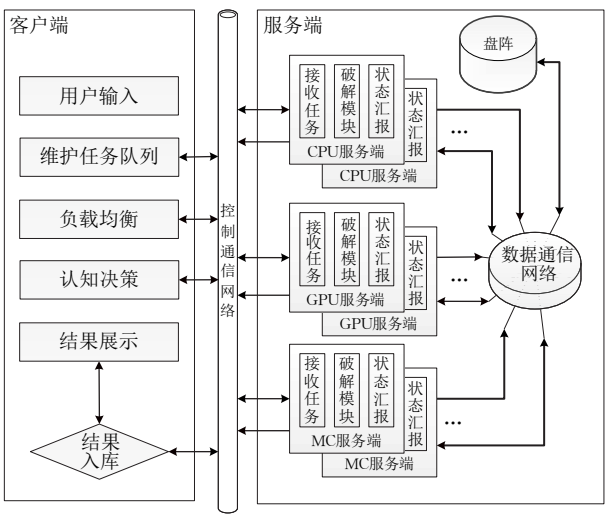

图 2 混合异构口令恢复系统结构图

3.1.2 设计流程

对拟态计算系统的设计主要分为三个部分,即计算部件的功能设计、存储资源的设计和互连结构的设计.用 P_{rs} 表示计算部件资源集合,M_{rs} 表示存储部件资源集合,C_{rs} 表示通信部件资源集合,由于计算、存储和通信部件资源有多种设计方案,因此 P_{rs}、M_{rs} 和 C_{rs} 可表示为

$$P_{rs}=\{p_1,p_2,\cdots,p_x\} \tag{1}$$

$$M_{rs}=\{m_1,m_2,\cdots,m_y\} \tag{2}$$

$$C_{rs}=\{c_1,c_2,\cdots,c_z\} \tag{3}$$

在式(1)、(2)和式(3)中,x、y 和 z 取有限值.那么,对于拟态计算系统结构的设计空间可以用 P_{rs}、M_{rs} 和 C_{rs} 组成的三元组表示.进一步,由 P_{rs}、M_{rs} 和 C_{rs} 的不同组合构成了可重构结构域 $TR=\{tr_1,tr_2,\cdots,tr_k\}$.对于应用领域 S,假设 S 中有 n 个业务 $s_i(1\leq i\leq n)$,每个业务 s_i 都要在 TR 上实现.那么实现目标系统的过程就转化为在一定约束条件下的可重构设计空间搜索问题,即在 TR 中搜索结构 tr_{opt}($tr_{opt}\in TR$),使 tr_{opt} 为业务 s_i 在 TR 上运行时的最优结构.

对于 HHPRS,有 $P_{rs}=\{\text{CPU},\text{GPU},\text{MC}\}$,$M_{rs}=\{\text{Local hard disk},\text{Disk array}\}$,$C_{rs}=\{\text{1000 M network},\text{10G network},\text{Ethernet switch}\}$,S=$\{\text{MS Office 2010},\text{MS Office 2007},\text{WinZip},\text{PDF},\text{MD5 Crypt}\}$,其中每个业务对应穷举攻击和字典攻击两种破解模式,以破解速度和时间为约束条件,那么对 HHPRS 的实现就转变为在破解具体任务时,搜索对应的最优系统结构 tr_{opt}.

由上,HHPRS 的具体执行步骤如下:

(1)用户在某一时刻提交一定数量的待破解文件,由任务队列程序获取后,形成任务链表 $S_{task}=\{task_1,task_2,\cdots,task_n\}$;

(2)根据用户输入的口令规则或字典文件,由系统的 P_{rs}、M_{rs} 和 C_{rs} 资源,依据 $EF(\text{HHPRS})=\text{Max}(Crack\ Speed)$ or $\text{Min}(Time)$,由认知决策,重构系统结构形成 tr_{opt},并将任务队列中的任务 $task_j$($1\leq j\leq n$)、口令规则或字典文件分配到各个计算部件中;

(3)各计算部件接收到任务后,开始产生口令

并进行计算恢复任务，并将中间处理状态 SE 实时入库，反馈给客户端；

(4) 客户端在收到中间状态 SE 后，根据各计算部件的任务执行情况，再由负载均衡和认知决策，继续向下分发口令规则或字典文件，或者通知某计算单元改变结构、破解模式等；

(5) 当某个计算单元找到正确口令后，客户端通知所有计算单元停止计算，并根据下一任务 $task_{j+1}$，由步骤(2)重新配置系统，进行下一轮计算；否则，继续进行破解当前任务.

区别于 CPU 和 GPU，拟态计算机具有很强的并行计算能力和较低的功耗，可以作为 HHPRS 的核心计算部件. 但由于 FPGA 开发周期长，复杂算法开发难度大，对算法结构、芯片资源、布局布线要求高. 同时，对于 CPU 和 GPU 的口令恢复算法已经有了较成熟的实现，如文献[3-11,38,39]. 因此，下面主要以口令恢复算法在拟态计算机上的实现展开来讲，包括算法特征分析和片内软硬件划分、破解模式和核心运算段的实现等，并拓展到 CPU 和 GPU.

3.2　片内结构

对于口令恢复，它属于计算密集型任务，期望可高速并行，如 MS Office 2010、MS Office 2007、WinZip、PDF、MD5 Crypt 等加密恢复算法，各步骤计算特征如表 1 所示，其中核心运算段占了整个运算量的 95%以上.

表 1　部分加密文档口令破解应用主要步骤计算特征

算法类型	主要步骤分析			
	文档解析	口令扩展	核心运算段	对比验证
MS Office 2010	提取 64 字节验证串	Unicode 转码，SHA1，字符串处理	100 000 次 SHA1	字符串拼接，AES 加解密，字符串比较
MS Office 2007	提取 64 字节验证串	Unicode 转码，SHA1，字符串处理	50 000 次 SHA1	字符串拼接，AES 加解密，字符串比较
WinZip	提取 24 字节验证串	HMAC-SHA1，字符串处理	4000 次 SHA1	一次验证：HMAC-SHA1 二次验证：Hash 整个文件，字符串比较
PDF	提取 88 字节验证串	字符串拼接，截断(32 字节)	3 次 MD5，2 次 RC4	异或，字符串比较
MD5 Crypt	提取 30 字节验证串	MD5，字符串拼接	1000 次 MD5	字符串拼接，字符串比较

在拟态计算机上，虽然 MS Office 2010、MS Office 2007、WinZip、PDF 和 MD5 Crypt 五类加密恢复算法的破解流程各不相同，但其主要过程是一致的，因此它们拟态计算内部的结构也基本一致. 针对口令恢复应用，对拟态计算机各 FPGA 进行细粒度重构，形成如图 3 所示的结构.

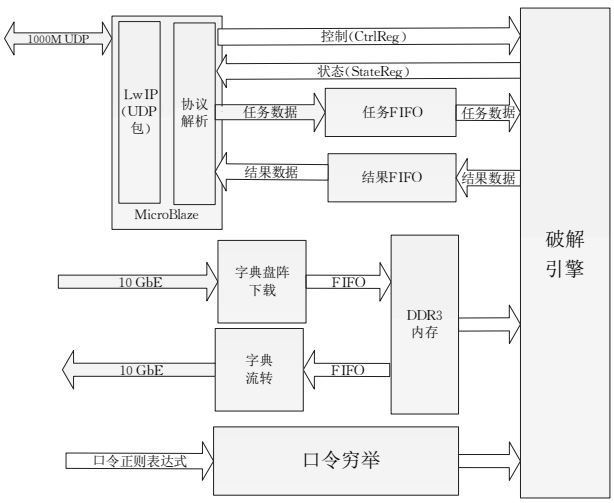

图 3　针对口令恢复应用的 FPGA 片内结构图

从图 3 中可以看出，FPGA 内部逻辑主要划分为 MicroBlaze 软件模块和硬件加速模块，硬件加速模块主要完成高速的口令穷举、盘阵字典下载和字典流转、内存字典缓存、破解引擎(高速口令恢复算法)等. MicroBlaze 软件模块上运行的有轻量级 TCP/IP 层 LwIP 协议，通过千兆网络与客户端进行通信，主要完成控制下载、任务下载、状态上报、结果上报等功能.

破解引擎，是算法的核心破解模块，主要实现五类加密文档的破解，可根据不同的算法特点及 FPGA资源占用情况，以不同的并行流水线方式实现，其基本结构如图 4 所示.

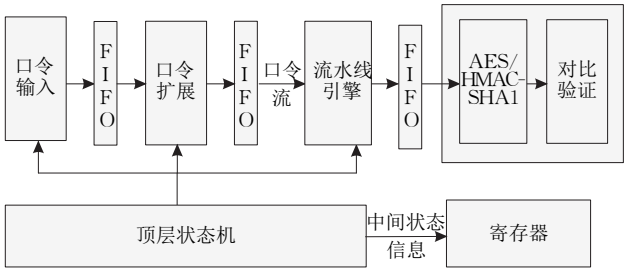

图 4　5 类加密恢复算法破解引擎

由于口令扩展和口令验证模块计算性能要求低，均采用低频串行编码实现. 而 Hash 迭代，计算量大，计算性能要求高，采用高频流水线算法实现，并根据不同算法及片上资源，放置多路流水线并行执行. 其次，为满足计算需求，流水线算法需要多个口令同时输入. 为此，在口令扩展和流水线之间放置

一个异步 FIFO,从而对扩展后的口令进行缓存,形成口令流,传输给流水线.最后,由流水线算法产生的临时验证串,也都缓存在 FIFO 中,并交由验证模块进行比较处理.

进一步,通过片上软核与硬核相结合,各核心运算模块之间的数据交互,由顶层状态机控制完成.计算过程中产生的状态信息、正确结果交由顶层状态机,并以寄存器的方式传递给 MicroBlaze 处理,随后发送给客户端.

最后,针对不同具体算法特点,粗粒度动态部分重构口令扩展、流水线引擎、AES/HMAC-SHA1、对比验证模块,使其灵活、快速地完成应用拟态变换.还可根据用户需求,实现字典模式和穷举模式的切换、任务的转换和字典的流转.

3.3 破解模式的实现

当前口令恢复典型的破解模式主要有口令穷举和字典攻击,口令穷举的规则和字典的口令内容,对口令恢复的效率有着重要的影响.对于口令穷举攻击,不仅要能够产生各种口令的组合,还要能产生高概率的口令.而对于字典攻击,不仅要具有一定的规模,包含各种常见口令,且对传输速率有着较高的要求.借助口令猜测攻击的研究,以提高口令破解率为目标,对口令穷举和字典攻击进行优化与改进.

3.3.1 口令穷举

口令穷举主要通过正则表达式产生口令,且为待穷举的口令每一位分配应一个字符集,这样能够以跳跃的方式产生口令,加快口令产生的速度,减少不必要口令的生成.同时,口令出现的频次服从 Zipf 分布,为了提高口令的命中率,以泄露的真实口令集为样本,通过口令的 SLD 模型统计分析所有口令集的口令结构组成,形成口令规则集合.如表 2 所示,给出了真实口令集 CSDN、Maopu、Renren、Rockyou、Yahoo、Hotmail 及综合这 6 种口令集的前 8 种口令结构的分布和比例,据此可按口令结构概率由高到低,依次添加到口令规则集中.此外,还可结合口令的定向攻击,以用户个人信息、键盘规律和常用语缩写等,取代部分口令结构,并扩展已有口令规则.例如国内用户常用口令结构为“姓名+数字”,数字可能是生日、身份证、手机号和键盘规律数字中的某几位.这样可以固定口令前几位为姓名全拼或缩写,穷举后面 6 到 8 位数字,以缩小口令搜索范围.

表 2 各口令集前 8 种口令结构的分布及比例

口令集	前 8 种口令结构分布及比例							
CSDN	D_8(21.51%)	D_9(11.11%)	L_8(4.92%)	D_{11}(4.31%)	D_{10}(3.75%)	L_3D_6(2.74%)	L_9(2.45%)	L_2D_6(2.02%)
Maopu	D_8(20.24%)	D_7(9.14%)	D_{11}(7.99%)	D_8(7.79%)	D_9(7.37%)	D_{10}(3.52%)	L_3D_6(1.70%)	L_8(1.27%)
Renren	D_8(16.86%)	D_7(10.17%)	D_8(9.15%)	D_9(5.43%)	L_6(5.21%)	L_8(4.52%)	D_{11}(4.08%)	L_7(3.72%)
Rockyou	L_8(4.80%)	L_6(4.19%)	L_7(4.08%)	L_9(3.60%)	D_7(3.40%)	D_{10}(3.33%)	D_8(2.99%)	L_6D_2(2.93%)
Yahoo	L_6(9.19%)	L_8(7.33%)	L_7(6.58%)	L_6D_2(4.59%)	L_9(3.65%)	L_8D_2(2.85%)	D_6(2.84%)	L_7D_1(2.40%)
Hotmail	L_6(9.67%)	L_8(7.88%)	L_7(7.23%)	D_6(7.14%)	L_9(5.21%)	D_8(5.02%)	L_{10}(3.46%)	D_7(2.43%)
综合口令集	D_8(7.69%)	D_6(6.70%)	D_7(4.59%)	L_8(4.42%)	D_9(3.66%)	L_6(3.41%)	D_{10}(3.24%)	L_7(2.91%)

另外,由于客户端可一次向下发送多个口令规则,因此需要将新产生的口令规则写入缓存中.基于拟态计算机的口令穷举模块具体结构如图 5 所示.

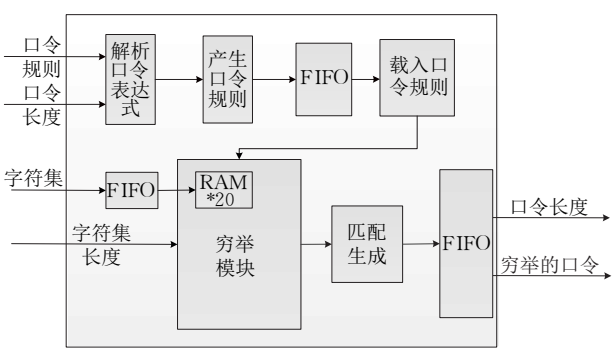

图 5 口令穷举模块结构图

通过客户端将口令规则 *pwd_regular* 下发到 FPGA 进行解析并产生新的口令规则.然后,穷举模块在收到字符集和新的口令规则后,进行口令的枚举生成.最后将口令和 *pwd_regular* 进行匹配,生成新的口令,形成口令流.具体描述如算法 1 所示.

算法 1. 基于正则表达式的口令穷举算法.

输入:口令规则 *pwd_regular*

输出:口令 *pwd″*

1. 解析口令规则 *pwd_regular* 中通配符‘?’的个数 n,并产生新的口令规则 r,r=“???…?”(n 个‘?’),将 r 写入 FIFO 堆栈中,形成规则集 R;
2. $r=POP(R)$;
3. FOR($i=0$; $i<n$; $i++$)
4. 初始化字符集 S_i 的指针 $Sp_i=0$;
5. *bnext*=TRUE;
6. WHILE(*bnext*)
7. {
8. FOR($i=0$; $i<n$; $i++$)

9.　　取字符集 S_i 的指针 Sp_i 对应的字符，组合产生 pwd' 并写入 FIFO 堆栈中；
10.　FOR(j=0；j<n；j++)
11.　{
12.　　判断字符集 S_j 的指针 Sp_j 是否需要进位或归 0；
13.　　如果 Sp_j 都为 0，$bnext$=FALSE；
14.　}
15.　l=0；将 pwd' 从 FIFO 中弹出；
16.　WHILE(k<$pwd\_regular$ 的长度)
17.　{
18.　　IF($pwd\_regular$ 的第 k 位==‘?’)
19.　　{
20.　　pwd'' 的第 k 位=pwd' 的第 l 位；
21.　　l++；
22.　　}
23.　　ELSE
24.　　pwd'' 的第 k 位=$pwd\_regular$ 的第 k 位；
25.　　k++；
26.　}
27.　将 pwd'' 写入 FIFO 中；
28. }
29. 跳转步 2，直至 R 为空；

算法 1 中，第 8～14 行代码段与第 15～27 行代码段在 FPGA 上可并行执行，口令生成峰值速度约为 180 M 个/秒，满足高口令输入的要求. 对于 CPU 和 GPU 的口令穷举模块，其算法流程和 FPGA 相同，可通过 CUDA 编程实现，并在 CPU 和 GPU 的每个线程中插入口令穷举模块，分配不同的口令前缀字符，完成口令的生成.

3.3.2　字典攻击

字典攻击通过预先定义好的常见或泄露的口令列表实施攻击，具有较好的攻击效率. 为满足不同服务的需求，设计三级口令库，提高覆盖率，一级库 M 级，二级库 G 级，三级库 T 级，一级库主要包含弱口令字典，二级库包含常规字典，三级库包含各种次常见口令组合和变形. 并进行分级管理，有针对性地实施字典攻击.

同时，为加快字典攻击速度，在拟态计算机上，字典文件的传输和字典的流转主要通过万兆网络 IP 协议实现，其结构如图 6 所示.

发送端口包括三个 FIFO：发送数据 FIFO(ip_snd_fifo)、发送状态 FIFO(ip_snd_status_fifo)以及发送无 ARP 应答 FIFO(ip_snd_no_arp_response_fifo). 发送数据 FIFO，主要用来发送数据，支持数据长度为 1～1440 bytes，并采用流水线

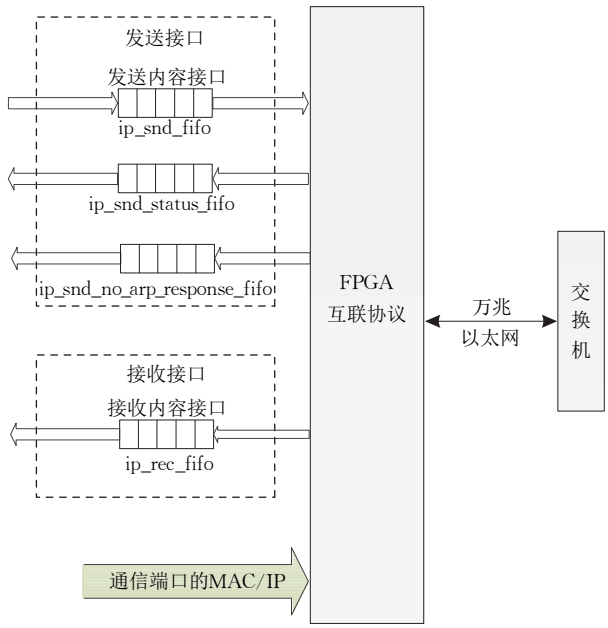

图 6　基于 IP 互联的 FPGA 接口示意图

的方式发送数据. 发送状态 FIFO 用来标识本次数据传输是否出现错误. 发送无 ARP 应答 FIFO 用来存储没有 ARP 应答的 IP 地址. 接收端口仅仅包括接收 FIFO，用来接收数据.

FPGA、盘阵之间通过可靠的 IP 协议进行字典传输，通信速率为 10 Gbps. 由 FPGA 自动发起 ARP 消息，建立连接，支持的路由表项为 2 千个. 盘阵上运行的有上位机服务端，在收到 FPGA 的请求后，会建立连接，并根据请求的字典名，向 FPGA 传输相应的文件. 当用户设置为字典模式时，拟态计算机各 FPGA 接收到客户端的配置指令后，从盘阵读取字典文件，并载入到内存中，然后再从内存中依次读取口令，完成计算任务. 对于 CPU 和 GPU 的字典模式，直接从本地硬盘获取字典文件，减少网络通信传输.

3.4　核心运算段的实现

由于 Hash 迭代属于核心运算段，这里以全流水的架构在拟态计算机上实现 SHA1 和 MD5 算法，提高系统性能. 而对于 CPU 采用汇编编程，GPU 采用 CUDA 编程，分别实现 SHA1 和 MD5 算法.

流水线结构通过把一个重复的过程分解为若干子过程，每个子过程可以和其他子过程同时进行，提高了系统的执行效率，具有较好的并行性. 如图 7 所示，具有 80 级流水线结构的 SHA1 算法，在其满负荷运行的情况下，可依次连续输入 80 组数据，80 级流水线并行执行，在经过 80 个时钟之后，连续输出 80 组 SHA1 运算的结果.

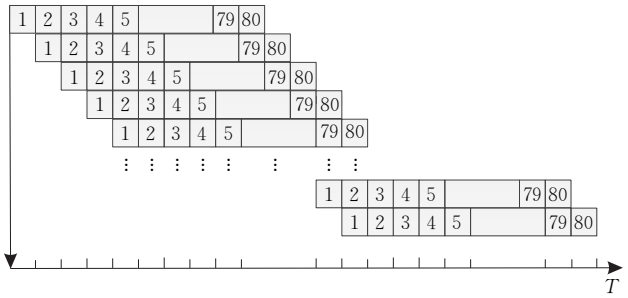

图 7 SHA1 全流水线结构示意图

3.4.1 SHA1 的改进与优化

SHA1 是 MS Office 2010、MS Office 2007 和 WinZip 的核心运算段，占整个运算量的 99%以上，这里以全流水架构实现对 SHA1 算法的硬件加速.

SHA1 将初始信息进行填充至 512 比特位，并初始化为 16 个 32 比特位分组 $w[15:0]$，以 H0=0x67452301，H1=0xEFCDAB89，H2=0x98BADCFE，H3=0x10325476，H4=0xC3D2E1F0 为初始链接变量，对五个中间变量 a、b、c、d、e 进行 80 轮迭代运算. 每轮迭代公式如下：

$$a=a\_next;\ b=a;\ c=c\_next;\ d=c;\ e=d;$$

$$a\_next=\{a[26:0],a[31:27]\}+f+e+kt+wt;$$

$$c\_next=\{b[1:0],b[31:2]\};$$

其中 f 为每轮迭代的非线性函数，kt 为每轮的常量，wt 为每轮的数据块.

最后，以 $a=a+H0$；$b=b+H1$；$c=c+H2$；$d=d+H3$；$e=e+H4$ 的级联输出 160 比特位散列值.

显然 b、c、d、e 可以直接通过传值得到，而 a 需要经过复杂运算得到，则延迟消耗都集中在关键路径 a 上. 对于 FPGA，加法比位运算延迟大得多，为了减少加法器的使用，定义进位保留加法器 CSA、$CSA4$、$CSA5$，如下所示：

$$CSA=(x\hat{}y\hat{}z)+(((x\&y)|(x\&z)|(y\&z))\ll 1);$$

$$CSA4=CSA((x\hat{}y\hat{}z),(((x\&y)|(x\&z)|(y\&z))\ll 1),u);$$

$$CSA5=CSA4((x\hat{}y\hat{}z),(((x\&y)|(x\&z)|(y\&z))\ll 1),u,v);$$

则 a 的计算可简化为

$$a=CSA5(\{a[26:0],a[31:27]\},f,e,kt,wt).$$

通过加入 CSA、$CSA4$、$CSA5$ 三个函数，最终使得五个模 32 加法运算简化为一个，对 a 值的运算过程来说，不仅缩短了关键路径降低了延迟，还减少了硬件面积. 优化后的 SHA1 单次迭代流程如图 8 所示.

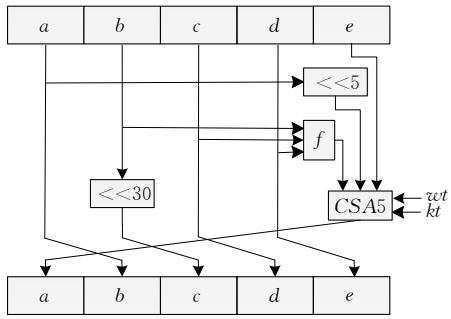

图 8 SHA1 单次迭代结构图

进一步，为了优化流水线的结构，将 a 值的计算和流水线工作独立开来. 首先，根据 SHA1 的 80 次迭代建立对应的 80 个子模块 Round0 到 Round79，每一个模块会单独计算出本次迭代下的 a 值，并以 a、b、c、d、e 的级联作为输出. 然后，在流水线的主结构中，定义 1 个 160 比特位宽的 reg 型数组 $R[0:79]$ 和 1 个 160 比特位宽的 wire 型数组 $Q[0:79]$，每个 $R[i](0\leq i\leq 79)$ 和 $Q[i]$ 与第 i 次 SHA1 操作的输入输出保持一致. 即 $Q[i]$ 作为本次 SHA1 计算的结果将值传递给 $R[i]$，$R[i]$ 作为下一次计算的输入. 最后，在主结构中定义 32 比特位宽的二维数组 $wt[0:79][0:15]$（从第 65 次迭代开始，部分 wt 由于没有参与计算，将会被优化掉），为每一次迭代计算提供参数. 而对于 w_next，采用 32 比特位宽的寄存器数组实现 $w\_next[63:0]$，计算公式如下所示：

$$\{w\_next[i][0],w\_next[i][31:1]\}=wt[i-1][1]\hat{}wt[i-1][3]\hat{}wt[i-1][9]\hat{}wt[i-1][14];$$

整个计算过程，如图 9 所示.

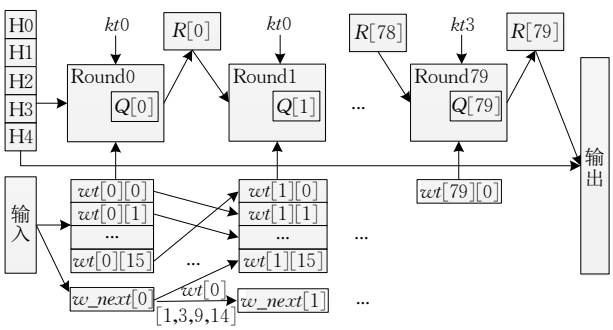

图 9 SHA1 流水线结构图

在拟态计算机上，SHA1 综合布局布线后占用资源为 7649 Slices，频率可达 303 MHz，吞吐量为 155 136 Mbps，具有较高的性能.

3.4.2　MD5 的改进与优化

MD5 是 PDF 和 MD5 Crypt 的核心运算段，下面通过对 MD5 算法进行深入分析，说明全流水架构 MD5 算法的硬件实现.

MD5 将初始信息进行填充至 512 比特位，并初始化为 16 个 32 比特位分组 $w[15:0]$，以 A＝0x01234567，B＝0x89ABCDEF，C＝0xFEDCBA98，D＝0x76543210 为初始链接变量，对四个中间变量 a、b、c、d 进行 16 轮 64 次迭代运算. 每轮迭代公式如下：

$$a=d;\ b=leftrotate((a+f+k+w),r)+b;$$

$$c=b;\ d=c;$$

其中 f 为每轮迭代的非线性函数，k 为初始参数，$leftrotate$ 左移位函数，r 为左移位参数.

最后，以 $a=a+$A；$b=b+$B；$c=c+$C；$d=d+$D 的级联输出 128 比特位散列值.

显然 a、c、d 可以通过简单的传值得到，而 b 需要经过复杂的运算，属于关键路径. 这里同样采用进位保留加法器 CSA、$CSA4$ 来完成 b 的计算，则 b 的计算可简化为

$$b=leftrotate(CSA4(a,f,k,w),r)+b;$$

通过加入 CSA 和 $CSA4$ 两个函数，最终使得四个模 32 加法运算简化为两个，缩短了 b 值运算的关键路径，减少了硬件面积占用. 优化后的 MD5 单次迭代流程如图 10 所示.

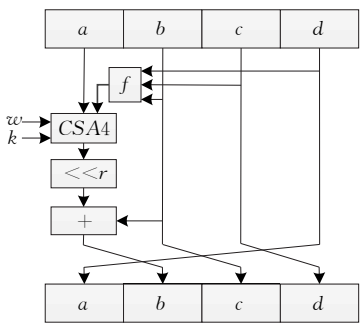

图 10　MD5 单次迭代结构图

进一步，为了优化流水线的结构，将 b 值的计算和流水线工作独立开来，具体实现方式和 SHA1 相同. 在拟态计算机上，MD5 综合布局布线后占用资源为 3963 Slices，最高频率可达 224 MHz，吞吐量为 114 688 Mbps，具有较高的性能.

3.5　负载均衡算法

HHPRS 是异构系统，对每一个具体的口令恢复应用 app，可以看成一个四元组 $app=\{F,G,Q,CS\}$，其中 F 表示是否找到正确口令，G 是随时间变化的负载曲线函数，Q 是随时间变化的 QoS(Quality of Service)函数，包括破解速度、优先级等，CS 函数以穷举破解、字典破解的配置参数作为函数参数，得到一个所有口令的集合 PSW_set. 即

$$PSW\_set=CS(argexh,argdic)$$

$$=\{psw_1,psw_2,psw_3,\cdots,psw_n\}.$$

显然 PSW_set 的大小和分配，影响着 F、G 和 Q 的变化. 因此，需要对异构口令恢复系统添加口令分配算法，动态调度各计算部件，使系统均衡工作，达到最优能效比. 如果采用静态分配策略，在系统初始化时，配置好每个计算单元的计算任务，采用固定的方式分发口令，将造成某些计算部件空闲，导致系统资源浪费. 对于口令空间的动态分配，这里根据各计算部件的计算能力，以 CPU 计算能力为基准，对口令空间进行动态划分. 如图 11 所示，将口令按字符位分割成动态分配单元和基准分配单元，动态分配单元主要用来产生口令前缀，基准分配单元用来穷举产生口令，以 CPU 处理能力计算而来，动态可调，是最小的分配单元.

图 11　口令空间的分配

对于穷举模式，动态口令负载均衡算法的具体流程如下：

(1) 根据具体的加密恢复任务，统计各部件的处理能力，CPU 服务端处理能力为 p_1、GPU 服务端处理能力为 p_2、FPGA 服务端处理能力为 p_3，则 $m=p_2/p_1$，$n=p_3/p_1$；

(2) 根据用户输入口令正则表达式和字符集，以 CPU 处理能力 p_1 为基准，划分基准分配单元包含的口令字符位数，其所能穷举的口令数目可以是 p_1 的整数倍(可根据需求进行设置)，其余口令字符位组成动态分配单元；

(3) 采用贪心策略，对动态分配单元按照 FPGA、GPU、CPU，优先分配 FPGA，再分配 GPU，最后分配 CPU，分配比例按 $n:m:1$，即向 CPU 分配 1 组口令前缀，向 GPU 和 FPGA 分别分配 m 和 n 组；

(4) 如果动态分配单元的所产生的口令前缀不足 n 组，则直接分配给 FPGA，如果大于 n 组且小于 $n+m$ 组，则分别分配给 FPGA 和 GPU，否则按比

例依次分配给 FPGA、GPU 和 CPU.

对于字典模式,可通过预处理,将所有字典按统一大小和格式存放.根据用户输入字典名,客户端将字典文件按 CPU、GPU 和 MC 服务端处理能力进行切分,并将切分后的结果分发到各计算部件,CPU 和 GPU 依次从本地读取字典,MC 从盘阵读取字典.

3.6 系统结构的变换

HHPRS 根据用户提交的应用参数,创建相关应用任务,在认知决策支持下,感知当前任务和口令集负载情况,动态分配 CPU、GPU 和 MC 计算组件并重构 FPGA 片内结构,改变系统互连方式,以合适的结构去处理当前的任务.针对口令恢复应用的 PMC 特征,在计算需求、通信需求、存储需求和服务需求方面,结合拟态计算的模型,对系统重构,可以形成胖树结构、环型结构、树环型结构、星型结构、金字塔型结构或蝶型结构.下面讨论在不同应用场景下,HHPRS 结构的变换.

(1)胖树结构主要应用于同时处理多个不同类型的加密文档,通过客户端管理各计算组件,然后由认知决策算法依次分配相同的口令集,进而调动整个系统的资源;

(2)环型结构主要用于解决低互连通信带宽、高口令输入的问题,通过客户端进行资源分配,将盘阵、FPGA 结构互连,形成流转链.例如在 PDF 字典破解方式中,对于长度为 20 字节的口令,每秒需要 500 MB 左右的传输带宽.为了解决盘阵"一对多"的瓶颈,由盘阵、FPGA 构成流转链,第一块 FPGA 从盘阵获取字典,第二块 FPGA 从第一块 FPGA 获取字典,第三块 FPGA 从第二块 FPGA 获取字典,以此类推直至最后一块,如图 12 所示.字典文件从盘阵依次流转到每片 FPGA,不仅解决了通信带宽可变的需求,也大大降低了盘阵资源的消耗;

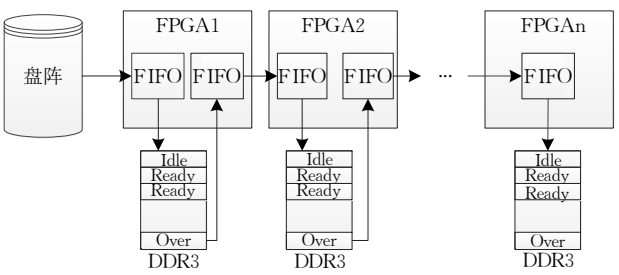

图 12 FPGA 字典流转链

(3)树环形结合了树型结构和环型结构的特点,是一种混合型结构.树环形结构便于客户端监控管理,降低了网络带宽带来的影响;

(4)对于口令输入要求低的应用,可以采取星型结构,由客户端决策,各计算部件分别从盘阵获取口令字典.例如 WinZip 的二次验证可以采用该结构,如图 13 所示.由于 WinZip 的二次验证需要用到整个加密文件,若将整个文档都加载到各计算部件中,会增加相当一部分的通信负担,进而影响破解速度.其次,二次验证的频率为每 65 536 个口令中有一个口令需要进行二次验证,在 WinZip 文件比较小的情况下(不超过 1 MB),一台服务器就能满足几十个计算部件的二次验证需求;

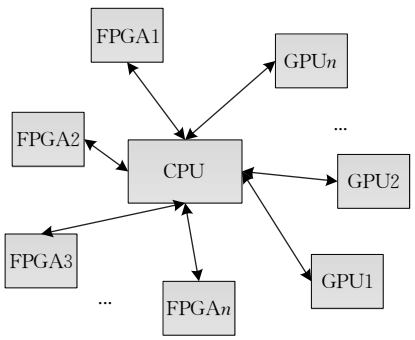

图 13 基于星型结构的 WinZip 二次验证

(5)金字塔型结构严谨,分工明确,较为稳定,便于客户端监控管理.同时,处于底层的计算部件,通过搏动式互连,可以更好地完成相互之间的通信,从而获知邻居单元的状态,或通过搏动式互连,对字典文件进行组播,以此提高互连通信带宽;

(6)蝶型结构主要用于各 CPU 之间的相互通信.通过蝶型互连,形成 CPU 点对点连接,而每个 CPU 又管理着其他计算部件,如 GPU 和 MC,从而使各 CPU 可以立刻获取对方的状态,减少系统决策的过程,使结构灵活可变.

另外,在口令空间较小的情况下,可直接使用 CPU+GPU 进行计算.由于 FPGA 需要一定的时间加载 bit 文件,直接使用 CPU+GPU 进行破解,减少了系统的重构过程.

4 实验比较及结果分析

本文搭建系统所用的客户端、CPU 服务端、GPU 服务端、MC 服务端、盘阵和万兆交换机,基本信息如表 3 所示.实验通过对比 HHPRS 的性能、分析穷举攻击和字典攻击的破解效率、系统的负载均衡、结构的变换及系统的组成,说明 HHPRS 的优势.

表 3　各计算部件配置信息

计算部件	名称	配置信息
CPU 客户端/服务端	IBM X3650 M3	1 颗 6 核 CPU；型号：X5650 2.66 GHz；内存：24 GB
GPU 服务端	Nvidia Tesla M2075	4 块 GPU 卡；核心频率：1.15 GHz；核心数：448；显存：6 GB
MC 服务端	Xilinx Virtex-6 LX550T	4 块 FPGA 卡；片内资源 Slices：85920；内存：24 GB
盘阵	Intel SSDSA2CW300G3	4 块 SSD；300 GB
万兆交换机	华三 24 口万兆交换机	24 个 1/10 G SFP+端口；4 个 10/100/1000 M 电口

对于 CPU 服务端，由于还要负责通信，并管理 GPU 的运行，故设定 CPU 最大破解线程数为 4. 而 GPU 服务端，设定 Block 数为 28，每个 Block 的线程数为 256. 对于 MC 服务端，各模块和算法实现情况如表 4 所示.

表 4　MC 服务端各模块和算法实现情况

模块	说明	频率	资源占用
MicroBlaze	负责与客户端通信	125 MHz	7292
DDR3	存储字典文件	核心 300 MHz 接口 100 MHz	5166
口令穷举	由正则表达式穷举口令	100 MHz	1212
万兆网络 IP 协议	传输字典	156.25 MHz	7328
MS Office 2010	10 路 SHA1 流水线并行	235 MHz	72487
MS Office 2007	5 路 SHA1 流水线并行	200 MHz	46300
WinZip	8 路 SHA1 流水线并行	200 MHz	67878
PDF	5 路 MD5 流水线并行	80 MHz	36886
MD5 Crypt	8 路 MD5 流水线并行	150 MHz	63670

注：表中各恢复算法为穷举模式下占用资源的情况，对于字典模式，会减少 MS Office 2010、WinZip 和 MD5 Crypt 的流水线个数到 5 路，使其满足布线要求.

4.1　性能对比

对于 MS Office 2010、MS Office 2007、WinZip、PDF、MD5 Crypt 五类加密文档，HHPRS 各 CPU、GPU、MC 计算部件口令破解速度如表 5 所示.

表 5　CPU、GPU、MC 口令破解速度

算法	CPU 速度（个/秒）	GPU 速度（个/秒）	MC 速度（个/秒）
MS Office 2010	43	14978	92000
MS Office 2007	91	29960	79200
WinZip	3199	467013	3165180
PDF	249877	28891400	21280000
MD5 Crypt	1009	936125	4742000

从表 5 中可以看出，CPU 的破解速度相对较低，而 GPU 和 MC 的破解速度有明显的数量级提升，且除 PDF 外，MC 较 GPU 的破解速度更高. 这主要由于 CPU 和 GPU 遵循冯诺依曼架构，需要对程序进行存储，且 GPU 主要依靠核心数和频率来提升加速比. 而 MC 主要由逻辑门电路组成，通过使用触发器（FF）和查找表（LUT）来实现计算，可以将应用算法的实现直接映射到硬件中，具有非指令执行的特性，明显区别于冯诺依曼架构，因此具有更高的执行效率.

“Elcomsoft Distributed Password Recovery”① 是一款高性能的分布式密码恢复工具，支持 Office 2010、Office 2007、WinZip 和 PDF 等加密文档的恢复. 这里采用 IBM X3650 M4 服务器进行对比测试，其中 M4 的配置信息为：8 核 Intel Xeon E5-2650 v2 CPU；频率 2.6 GHz；内存：24 GB. 可以看出 M4 较 M3 性能上有所提升. HHPRS 与专业 Elcomsoft 加密文档恢复软件在破解速度（Speed）和能效比（Energy Efficiency Ratio，EER）方面的对比如图 14、图 15 所示，其中能效比计算公式为

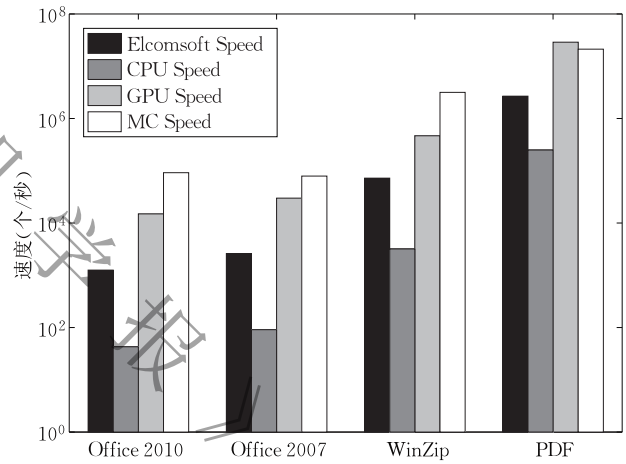

图 14　HHRPS 各计算部件与 Elcomsoft 口令破解速度对比

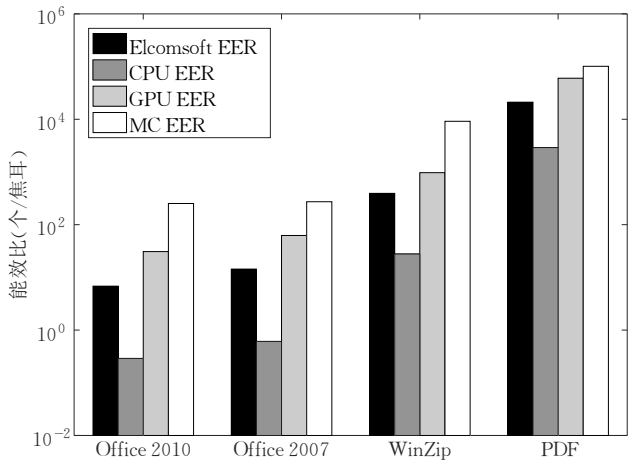

图 15　HHRPS 各计算部件与 Elcomsoft 能效比对比

① https://www.elcomsoft.com/

能效比$=\frac{\text{口令破解平均速度}}{\text{平均功耗}}$,单位是“个/焦耳”.

从图 14、图 15 中我们可以看出,虽然本系统中 CPU 的破解速度和能效比低于 Elcomsoft,但 GPU 的破解速度是 Elcomsoft 软件的 6.45 倍～11.89 倍,能效比是 Elcomsoft 软件的 2.47 倍～4.53 倍,MC 的破解速度是 Elcomsoft 软件的 7.95 倍～73.02 倍,能效比是 Elcomsoft 软件的 4.8 倍～36.96 倍.那么,由单个 CPU、GPU 和 MC 服务端构成的 HHPRS,对应各算法,其总的破解速度是 Elcomsoft 软件的 18.84 倍～84.94 倍,能效比是 Elcomsoft 软件 3.07 倍～15.73 倍,具有较高的加速比和能效比.虽然 Elcomsoft 软件经过专业的 CPU 优化加速,但由于 CPU 频繁的读写内存、硬盘,调度线程与切换,造成操作系统的效率低下,从而无法胜任高性能计算.而本系统中的 CPU 还要负责调度 GPU 工作及通信,破解效率较 Elcomsoft 更低.

HHPRS 与其他文献方案对比情况如表 6 所示.

表 6 HHPRS 与其他方案性能对比

其它方案				HHPRS
参考文献	算法	计算部件	速度(个/秒)	速度(个/秒)
文献[11]	MS Office 2010	Nvidia Tesla M2090	1474	107021
文献[11]	MS Office 2007	Nvidia Tesla M2090	2901	109251
文献[8]	WinZip	NVIDIA GTX 295 ×4	6144	3635392
文献[10]		AMD Tri-X R9 290x ×2	262000	
文献[10]	PDF	AMD Tri-X R9 290x ×2	4522000	3549526 *
文献[24]		Nvidia GTX 680 ×1 & Xilinx ML605 FPGA ×14	7900000	
文献[4]		NVIDIA GTX 295	884870	
文献[5]	MD5 Crypt	Tesla M2050	326000	5679134
文献[23]		Stratix III EP3SE50F484C4	24950	

注:文献[10]、[24]使用的 PDF 恢复算法版本为 Acrobat 5-8,本文使用的版本为 Acrobat 2-4,这里“*”的破解速度可由本文 PDF 实现的速度及 RC4 迭代次数换算得到.

从表 6 中可以看出,除 PDF 外,本文搭建的 HHPRS 较其他方案在性能上有较大的提升.一方面,由于本文采用 CPU+GPU+MC 拟态异构计算,并对算法特征进行了深入地分析,将程序段划分到合适的计算部件上,并对核心算法进行了优化处理.另一个方面,由于采用 MC 计算部件,并对 FPGA片内结构进行了合理布局,及采用多流水线并行的架构,提高了计算效率.而对于 PDF,由于 RC4 属于流式加密算法,无法采用流水线结构实现,只能利用多模块并行的方式进行加速.这也造成拟态计算机布局布线难度增加,算法频率降低,性能低于其他方案.而对于文献[10],主要依靠 GPU 多核心(2560 个核心)计算,从而提高了计算速度,但 RC4 仍是计算瓶颈.而文献[24]采用 GPU+FPGA 的方式,使用 GPU 完成 MD5 计算,多个 FPGA 完成 RC4 运算,提高了整体计算速度.这也说明了异构计算的优势.

4.2 破解效率分析

4.2.1 穷举攻击

对于传统的口令穷举算法,假设长度为 n 的口令 p,已知其中 $m(m\leq n)$ 位,字符集 S 长度为 l,则其最大搜索空间为 l^{n-m}.对于具体的口令 $p=c_1c_2\cdots c_n$,假设 c_i 映射到字符集的位置为 $d_i(1\leq i\leq n)$,即 $S(d_i)=c_i$,则搜索空间为

$$\sum_{i=1}^{n-1}(d_i-1)l^{n-i}+d_n.$$

对于本文的口令穷举算法,假设长度为 n 的口令 p',已知其中 $m(m\leq n)$ 位,每一位对应的字符集 S'_i 长度为 l'_i,则其最大搜索空间为 $\prod_{i=1}^{n-m}l'_i$,由于 $l'_i\leq l$,显然 $\prod_{i=1}^{n-m}l'_i\leq l^{n-m}$.对于具体的口令 $p'=c'_1c'_2\cdots c'_n$,假设 c'_i 映射到字符集 S'_i 的位置为 $d'_i(1\leq i\leq n)$,即 $S'_i(d'_i)=c'_i$,则搜索空间为

$$\sum_{i=1}^{n-1}(d'_i-1)\prod_{j=i+1}^{n}l'_j+d'_n.$$

由于 $l'_i\leq l$ 且 $d'_i\leq d_i$,显然

$$\sum_{i=1}^{n-1}(d'_i-1)\prod_{j=i+1}^{n}l'_j+d'_n\leq\sum_{i=1}^{n-1}(d_i-1)l^{n-i}+d_n.$$

以 6 位长度口令“hprs1$”为例.其中,口令的字符空间集为小写字母(a～z)、数字(0～9)、特殊字符集(′～!@#$ 等).假设已知口令前 4 位口令为小写字符,第 5 位为数字,第 6 位为特殊字符.若按传统口令穷举算法,穷举次序为 26 个小写字母、10 个数字、33 个特殊字符,共 69 个字符.由公式,则需计算 $7\times69^5+15\times69^4+17\times69^3+18\times69^2+27\times69^1+42=11293898514$ 次才能生成该口令.根据本文提出的口令穷举方法,设置前 4 位口令为小

写字母，中间 1 位为数字，最后 1 位为特殊字符. 由公式，只需计算 $7\times26^3\times10\times33+15\times26^2\times10\times33+17\times26^1\times10\times33+18\times10\times33+1\times33+6=44\,098\,599$ 次就能生成该口令. 此例中，本算法较传统口令生成算法少生成了 11 249 799 915 个口令. 由此可见，本文的口令穷举算法能减少大量不必要的口令生成，节省口令恢复的总体时间，提高效率.

4.2.2　字典攻击

对于字典模式，各 CPU 和 GPU 计算部件可直接从本地获取字典文件，以减少网络传输. 而对于 MC 计算部件，需要从盘阵获取字典文件. 获取方式有 2 种：(1) 各 MC 计算部件独立从盘阵获取字典，互不影响；(2) 通过认知决策进行配置，形成流转链，字典文件从盘阵依次流转到 MC 的每片 FPGA. 在口令输入要求低的应用中可以采用第 1 种方案，如 MS Office 2010、MS Office 2007. 而在口令输入要求高的应用中可以采用第 2 种方案，如 WinZip、PDF 和 MD5 Crypt.

MC 组件各 FPGA 以万兆网络互连，通过 IP 协议传输字典，每秒从盘阵获取字典文件的峰值速度约为 500 MB/s，具有很高的传输速度. 对于字典流转方案，前一个 FPGA 在获取到字典文件后，随即从内存中读出，并发送给后面的 FPGA，然后再开始破解任务. 两个 FPGA 之间的文件传输峰值速度也约为 500 MB/s.

对于 HHPRS，各算法在字典模式下，破解含有 308 915 800 个口令，2.3 GB 的大小的字典，所需要的时间如图 16 所示，其中 MS Office 2010、MS Office 2007 各自从盘阵读取字典，WinZip、PDF 和 MD5 Crypt 采用字典流转方案.

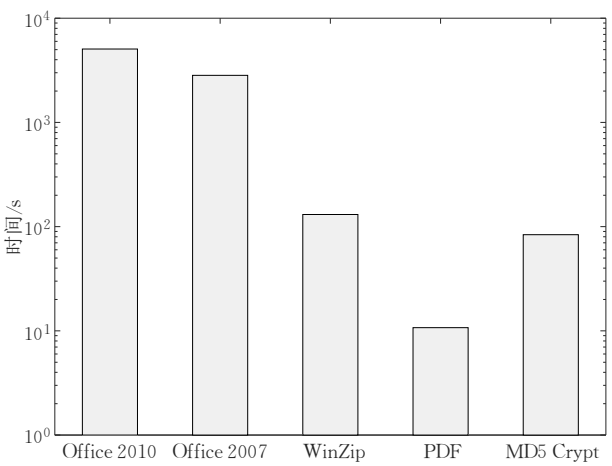

图 16　各算法在字典模式下的破解时间

另外，对于 PDF，使用 MC 计算部件，4 块 FPGA 分别采用各自从盘阵获取字典和字典流转 2 种方案，读取 2.3 GB、4.66 GB 和 6.06 GB 大小的字典，其所消耗的时间对比如图 17 所示，其中 Dic1-2-3 表示依次读取字典文件 Dic1、Dic2 和 Dic3.

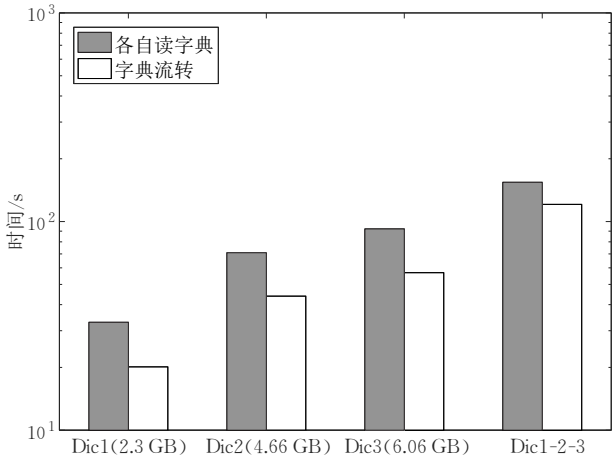

图 17　PDF 字典模式下 2 种方案破解时间对比

从图 16、图 17 中可以看出，口令破解速度越快，破解时间越短，对字典传输带宽需求越大. 而字典流转方案可以很好地解决字典传输瓶颈的问题，降低读取盘阵字典的次数、字典传输的时间和“一对多”排队等待的时间，满足系统的计算要求.

4.3　负载均衡分析

为了验证系统负载均衡决策算法的有效性，针对穷举攻击，分别采用静态分发策略和动态分发策略进行对比分析.

对于具体的恢复任务，采用静态分发时，需要根据计算部件的数量，将口令首字符按照 CPU、GPU、FPGA 依次进行分发. 由于正确口令的分布具有随机性，假设找到正确口令时，各计算部件运行时长分别为 t_{CPU}、t_{GPU}、t_{FPGA}，则整个系统运行时长为 $t_s=\mathrm{Max}(t_{\mathrm{CPU}},t_{\mathrm{GPU}},t_{\mathrm{FPGA}})$. 而对于具体的恢复任务，调用决策算法，采用动态分发策略，其系统运行时间，即为找到正确口令所用时间，记为 t_d.

假设穷尽空间大小为 Psw_{total} 的口令，采用静态策略分发，由于各处理单元计算能力不一，总有一个计算单元最后完成，假定 t_1 为各处理单元同时工作的时间，t_2 为剩余最后一个处理单元的工作时间，则总时间为

$$t_s=t_1+t_2.$$

又

$$t_1=\frac{Psw_1}{Perf_{\mathrm{CPU}}+Perf_{\mathrm{GPU}}+Perf_{\mathrm{FPGA}}},$$

$$t_2=\left(\frac{Psw_2}{Perf_{\mathrm{CPU}}}\text{或}\frac{Psw_2}{Perf_{\mathrm{GPU}}}\text{或}\frac{Psw_2}{Perf_{\mathrm{FPGA}}}\right)$$

且 $Psw_1+Psw_2=Psw_{\mathrm{total}}$.

对于动态分发，假设口令进行了均衡分配：

$$t_d=\frac{Psw_{\text{total}}}{Perf_{\text{CPU}}+Perf_{\text{GPU}}+Perf_{\text{FPGA}}}.$$

显然：$t_d<t_1+t_2=t_s$.

所以，采用动态分发策略，较静态分发策略更能提高系统利用率，减少系统资源浪费，提高效率.

以恢复 MS Office 2010 加密文档为例，对系统的负载均衡进行测试，设定口令长度为 6 位，字符集为{0123456789}，正确口令为 261234. 采用静态分发策略，以 0 为首字符的口令会优先分配给 MC，1 切分给 GPU，2 切分给 CPU，剩下的依据各计算部件破解情况，计算完当前口令空间，再次申请分配. 而采用动态分发策略，CPU、GPU、MC 各计算部件平均处理能力依次为：43 个/秒、14 978 个/秒、92 000 个/秒. 由系统决策将口令低 2 位作为基准分配单元，高 4 位作为动态分配单元. 向 CPU 分配 1 组口令前缀，分别向 GPU 和 MC 分配 350 组和 2140 组，依次切分给 MC、GPU 和 CPU. 最后，静态分发和动态分发各计算部件穷尽的口令空间和破解时间对比情况如图 18、图 19 所示.

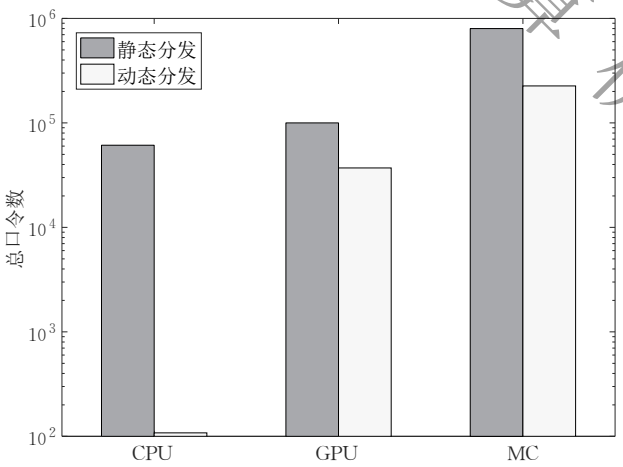

图 18 静态分发和动态分发各计算部件穷尽的口令数对比

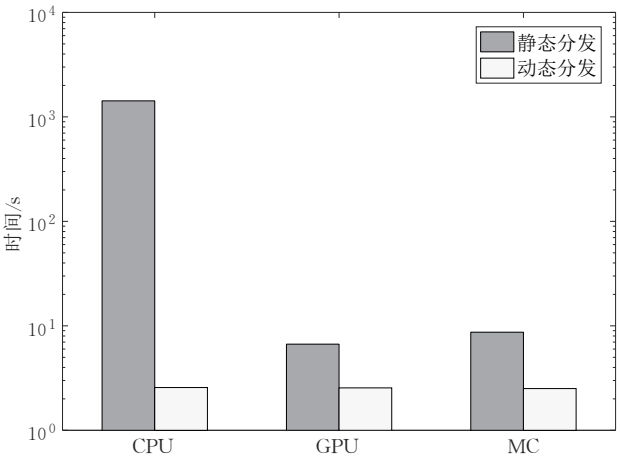

图 19 静态分发和动态分发各计算部件破解时间对比

从图 18 和图 19 中可以看出，在静态分发策略中，CPU 最终破解出正确口令，其他计算部件已穷尽所有口令，处于空闲等待状态，系统最终运行时间为 1225.5 s. 在动态分发策略中，MC 最终破解出正确口令，其他计算部件随后停止运行，系统平均运行时间为 2.57 s. 除运行时间较短外，各计算部件在动态分发中穷尽的口令数也要远低于静态分发，进而提高了系统破解效率和资源利用率.

4.4 结构变换分析

传统口令恢复系统在进行破解任务时，硬件平台固定不变，只是根据不同的业务采用不同的算法，从而得到最佳的性能或效能. 同时，这些系统只能提交并破解一个任务，无法支持多任务、多口令攻击模式并行计算. 而本文提出的口令恢复系统在处理破解任务时，不仅可以根据具体业务选择最佳算法，还可以为具体业务选择不同的硬件平台或者计算结构进行处理，从而趋于最佳的性能或效能.

当用户提交一个加密恢复任务时，HHPRS 会根据任务的类型和数量，及系统中的计算资源，改变计算部件的互连方式，以合适的结构匹配当前任务. 用 CPU、GPU 和 MC 服务端各 3 台，搭建 HHPRS，对于不同的加密算法，当用户提交单个加密文档和多个加密文档时，对应的系统互连结构，如表 7 所示.

表 7 不同算法和任务对应的互连结构

算法	单个加密文档对应的互连结构	多个加密文档对应的互连结构
MS Office 2010	星型	星型
MS Office 2007	星型	星型
WinZip	当文件大小≤1 MB 时：星型 否则：树环型	当文件大小≤1 MB 时：星环型 否则：树环型
PDF	星型	星环型
MD5 Crypt	星型	星环型

从表 7 中可以看出，由于 MS Office 2010 和 MS Office 2007 破解速度较低，对通信需求较低，可以选择结构简单的星型结构. 对于 WinZip，由于需要进行二次验证，当文件较小时，1 个 CPU 即可满足计算需求，采用星型和星环型结构更便于管理. 当文件较大时，二次验证计算时间增加，会成为整个 WinZip 计算的瓶颈. 例如，对于 10 MB 左右大小的 WinZip 文件，二次验证需要约 0.012 s 左右，即每秒可验证口令数为 83 个，而整个系统一次验证通过口令数为 3 635 392/65 536×3≈166 个，当只有 1 个 CPU 客户端进行二次验证时，各计算部件需要依次

排队等待. 因此，可以采用树环型结构，通过 1 个客户端管理 3 个 CPU 服务端，而 3 个 CPU 服务又各自管理 1 个 GPU 和 MC 服务端，从而提高二次验证计算效率. 对于 PDF 和 MD5 Crypt，由于破解速度较高，对于单个文档的破解，可以采用星型结构，并在字典模式下各自独立读取字典. 而对于多个文档的破解，可以采用星环型结构，依次下发多个任务，并在字典模式下配置为字典流转模式，使多个任务并行计算. 这样，由 Zipf 定律，可以在多个加密文档中优先恢复出一部分口令，不会因为某个加密文档口令过于复杂，造成系统一直在计算该加密文档，而其他任务一直在等待.

如图 20 所示，在恢复 50 个 PDF 加密文档的情况下，分别采用星型结构顺次下发 1 个任务和星环型结构依次下发多个任务(6 个)，在相应时间内破解出的口令数对比.

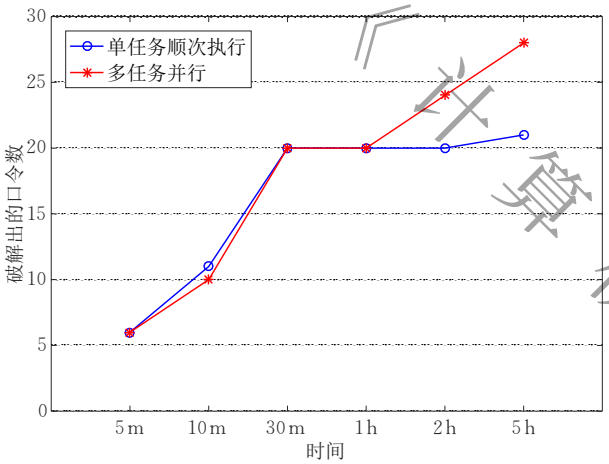

图 20　2 种策略和结构下 PDF 破解出的口令数对比

从图 20 中可以看出，采用多任务并行计算的星环型结构比采用单任务顺次计算的星型结构破解出的口令数多. 这也说明了不同的任务需要不同的策略和结构来提高计算效率. 随后，在多任务并行计算模式下，可将长时间未破解出的 PDF 加密文档进行任务迁移，对系统资源进行回收，以改变系统结构，使用更多的计算资源，加速当前口令复杂度高的任务的破解.

最后，对于字典＋穷举的混合攻击模式，由于 CPU＋GPU 读取字典更为方便，且 MC 为了支持字典模式牺牲了部分计算性能，因此可使用 CPU＋GPU 完成字典攻击，MC 完成规则穷举攻击，来进一步提高破解效率. 此时，可采用树星型结构，使用 1 个 CPU 管理 GPU 服务端，1 个 CPU 管理 MC 服务端，并将字典攻击下 GPU 未破解出的任务迁移到 MC，利用 MC 的高效能计算完成穷举攻击.

4.5　系统组成分析

口令恢复应用属于整型计算，计算密度高、算法逻辑结构简单，要求可高度并行，同时，字典模式下，不同算法对字典规模和通信带宽要求不一，非常适合 CPU＋GPU＋MC 异构系统实现，并可获得较好的计算性能. 下面，从多个角度综合对比分析了 HHPRS 的 CPU、GPU 和 MC 三种异构计算部件在口令恢复领域的应用情况，包括支持的算法种类、口令攻击策略、加速技术、计算瓶颈、互连方式和变换响应速度，如表 8 所示.

表 8　HHPRS 各计算部件综合对比分析

计算部件	支持算法种类	口令攻击策略	加速技术	计算瓶颈	互连方式	变换响应速度
CPU 服务端	多	字典、穷举、彩虹表	多线程	计算受限	千兆网络	立刻，灵活
GPU 服务端	较多	字典、穷举	多核心并行	访存受限	PCI-E 总线	立刻，灵活
MC 服务端	少	字典、穷举	流水线、多模块并行	布线受限	千兆、万兆网络	需重新配置 FPGA

从表 8 中可以看出，CPU 和 GPU 支持算法种类多于 MC，这主要得益于 CPU 和 GPU 的通用性和算法的易用性. 在口令攻击策略方面，除了字典和穷举攻击，CPU 还支持彩虹表攻击，即把已破解出的加密文档的验证串和口令入库，并通过查表法和将要计算的任务进行比对，如果验证串相同，可立即找到对应的口令. 在口令恢复算法的加速方面，CPU 主要利用多线程技术，以提高 CPU 利用率来提高计算速度，GPU 主要利用多核心并行技术，以核心数量和频率来提高加速比，MC 主要利用流水线和多模并行技术，将核心运算段以全流水架构实现，并配合状态机控制，实现多模块并行计算. 在计算瓶颈方面，CPU 主要受限于计算，GPU 受限于访存，MC 受限于布线资源，这也是由于不同计算部件的内部架构所决定的. 例如，对于 PDF 的加速，GPU 受限于 RC4 计算速度，MC 受限于 1 对多模块并行(1 个 MD5 模块对应 16 个 RC4 模块)，布线难度增加，较其他算法提高的倍数较低. 在互连方式方面，针对不同计算部件和通信需求，CPU 采用千兆网络，GPU 采用 PCI-E 总线与 CPU 互连，MC 采用千兆和万兆网络，多种互连方式并存，以适配多种攻击策略. 在系统结构和任务变换响应速度方面，CPU 和 GPU 可立即完成配置，较为灵活，而 MC 则需要重新加载 bit 文件.

由上,通过 HHPRS 可针对不同口令恢复应用和负载,结合 CPU、GPU 和 MC 不同的计算特点,组建合适的口令攻击策略和系统结构.例如,对某一算法,采用星型结构,使用 CPU 进行彩虹表攻击,使用 GPU 进行字典组合攻击,使用 MC 进行规则穷举攻击,协作完成任务.另外,也可根据不同口令恢复算法的性能,设置合适的口令攻击策略,以取得较好的收益.例如,对于千个/秒级,破解速度较慢的算法,可考虑只使用字典攻击;对于万个/秒和兆个/秒级,破解速度较快的算法,可使用字典+穷举攻击;对于全口令空间较小的算法,如 VPN(全口令空间为 2^{56})、LM(全口令空间为 69^7)算法等,可考虑只使用穷举攻击.显然,HHPRS 可以很高效地支持口令恢复领域应用的各种算法和攻击策略组合,并获得较好的恢复效率.

5 结束语

本文提出的一种新型的混合异构口令恢复系统 HHPRS,结合拟态计算的思想,以 CPU、GPU 和拟态计算机搭建异构系统,通过变结构实现系统的高效计算.并在拟态计算机上实现了穷举攻击、字典攻击及高速口令恢复算法,提高了系统性能.同时,调用负载均衡算法合理分配口令空间,并由认知决策针对不同应用场景变换系统结构,实现异构部件间的协作.实验结果表明该系统不仅具有较高的计算性能,且具有较高的破解效率.同时,在拟态计算机上对口令恢复算法的研究,也为生产口令恢复 ASIC 芯片提供了理论基础和指导意义.

但是,由于口令恢复算法众多,部分算法并没有得到完全优化,仍有提升空间,如 PDF 中的 RC4 算法,需要进一步提升性能.同时,如何有针对性地进一步提高口令攻击命中率,也亟待研究.此外,如何实现 HHPRS 各计算部件间的任务无缝切换和迁移,使其在运行中变换系统结构,并使得 GPU 和 MC 更紧密的协作,也是下一步研究的方向.

参 考 文 献

[1] Wang Ping, Wang Ding, Huang Xin-Yi. Advances in password security. Journal of Computer Research and Development, 2016, 53(10): 2173-2188(in Chinese)
(王平,汪定,黄欣沂. 口令安全研究进展. 计算机研究与发展, 2016, 53(10): 2173-2188)

[2] Zou Jing, Lin Dong-Dai, Hao Chun-Hui. A password cracking method based on structure division probability. Chinese Journal of Computers, 2014, 37(5): 1206-1215(in Chinese)
(邹静,林东岱,郝春辉. 一种基于结构划分概率的口令攻击方法. 计算机学报, 2014, 37(5): 1206-1215)

[3] Qiu W, Gong Z, Guo Y, et al. GPU-based high performance password recovery technique for hash functions. Journal of Information Science & Engineering, 2016, 32(1): 97-112

[4] Sprengers M, Batina L. Speeding up GPU-based password cracking//Proceedings of the Workshop Record of SHARCS. Washington, USA, 2012: 35-54

[5] Wang F, Yang C, Wu Q, et al. Constant memory optimizations in MD5 Crypt cracking algorithm on GPU-accelerated supercomputer using CUDA//Proceedings of the International Conference on Computer Science & Education. Melbourne, Australia, 2012: 638-642

[6] Zhan X, Hong J. Study on GPU-based password recovery for MS Office2003 document//Proceedings of the International Conference on Computer Science & Education. Melbourne, Australia, 2012: 517-520

[7] Kim K, Lee S, Hong D, et al. GPU-accelerated password cracking of PDF files. KSII Transactions on Internet & Information Systems, 2011, 5(11): 2235-2253

[8] Tan N D, Pham P H, Nguyen D H, et al. Decryption-decompression of AES protected ZIP files on GPUs//Proceedings of the International Conference on Graphic and Image Processing. Cairo, Egypt, 2011: 170-177

[9] An X, Jia H, Zhang Y. Optimized password recovery for encrypted RAR on GPUs//Proceedings of the International Conference on High Performance Computing and Communications. New York, USA, 2015: 591-598

[10] Hranický R, Matoušek P, Rysavy O, et al. Experimental evaluation of password recovery in encrypted documents//Proceedings of the International Conference on Information Systems Security & Privacy. Rome, Italy, 2016: 299-306

[11] Lv Hui. Research on Crack Office File Password Based on CUDA Architecture[M. S. dissertation]. PLA Information Engineering University, Zhengzhou, 2014(in Chinese)
(吕辉. 基于 CUDA 架构的 Office 密文档破解技术研究[硕士学位论文]. 解放军信息工程大学, 郑州, 2014)

[12] Wu Jiang-Xing. Meaning and vision of mimic computing and mimic security defense. Telecommunications Science, 2014, 30(7): 1-7(in Chinese)
(邬江兴. 拟态计算与拟态安全防御的原意和愿景. 电信科学, 2014, 30(7): 1-7)

[13] Li Bo-Nan. Research on Key Technologies of Domain-Oriented Reconfigurable Systems[Ph. D. dissertation]. PLA Information Engineering University, Zhengzhou, 2015(in Chinese)
(李柏楠. 面向领域应用的可重构系统关键技术研究[博士学位论文]. 解放军信息工程大学, 郑州, 2015)

[14] Wu Jiang-Xing, Luo Xing-Guo, Si Xue-Ming. "Adaptable" mimic computing village. IEEE Spectrum, 2014, (5): 54-58 (in Chinese)
(邬江兴，罗兴国，斯雪明. "随机应变"的拟态计算村. 科技纵览，2014，(5)：54-58)

[15] Herbordt M C, VanCourt T, Gu Y, et al. Achieving high performance with FPGA-based computing. Computer, 2007, 40(3): 50-57

[16] Bonneau J, Herley C, Van Oorschot P C, et al. Passwords and the evolution of imperfect authentication. Communications of the ACM, 2015, 58(7): 78-87

[17] Marechal S. Advances in password cracking. Journal in Computer Virology, 2008, 4(1): 73-81

[18] Kwok S, Lam E. Effective uses of FPGAs for brute-force attack on RC4 ciphers. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16(8): 1096-1100

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

[20] Kim J W, Lee H U, Won Y. Design for high throughput SHA-1 hash function on FPGA//Proceedings of the International Conference on Ubiquitous & Future Networks. Phuket, Thailand, 2012: 403-404

[21] Malvoni K, Designer S, Knezovic J. Are your passwords safe: Energy-efficient bcrypt cracking with low-cost parallel hardware//Proceedings of the Woot'14 USENIX Workshop on Offensive Technologies. San Diego, USA, 2014: 10-10

[22] Dürmuth M, Kranz T. Technology and Practice of Passwords: On Password Guessing with GPUs and FPGAs. Cham, Switzerland: Springer International Publishing, 2015

[23] Han Jin-Sheng, Lin Jia-Jun, Zhou Wen-Jin, et al. Design of Linux password high-speed crack model on FPGA. Journal of Chongqing University, 2012, 35(8): 42-47(in Chinese)
(韩津生，林家骏，周文锦等. FPGA 的 Linux 口令密码高速破解模型设计. 重庆大学学报:自然科学版，2012，35(8)：42-47)

[24] Danczul B, Fuss J, Gradinger S, et al. Cuteforce analyzer: A distributed bruteforce attack on PDF encryption with GPUs and FPGAs//Proceedings of the International Conference on Availability, Reliability and Security. Regensburg, Germany, 2013: 720-725

[25] Johnson T, Roggow D, Jones P H, et al. An FPGA architecture for the recovery of WPA/WPA2 keys. Journal of Circuits, Systems and Computers, 2015, 24(7): 1550105

[26] Kammerstetter M, Muellner M, Burian D, et al. Efficient high-speed WPA2 brute force attacks using scalable low-cost FPGA clustering//Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems. Santa Barbara, USA, 2016: 559-577

[27] Weir M, Aggarwal S, De Medeiros B, et al. Password cracking using probabilistic context-free grammars//Proceedings of the IEEE Symposium on Security and Privacy. Berkeley, USA, 2009: 391-405

[28] Weir M, Aggarwal S, Collins M, et al. Testing metrics for password creation policies by attacking large sets of revealed passwords//Proceedings of the 17th ACM Conference on Computer and Communications Security. Chicago, USA, 2010: 162-175

[29] Veras R, Collins C, Thorpe J. On the semantic patterns of passwords and their security impact//Proceedings of the Network and Distributed System Security Symposium (NDSS'14). San Diego, USA, 2014: 1-16

[30] Chou H C, Lee H C, Yu H J, et al. Password cracking based on learned patterns from disclosed passwords. International Journal of Innovative Computing Information & Control, 2013, 9(2): 821-839

[31] Zou J, Lin D, Hao C, et al. Making a higher hit ratio cryptanalytic time-memory trade-off attack on passwords. Chinese Journal of Electronics, 2013, 22(4): 671-676

[32] Han W, Li Z, Yuan L, et al. Regional patterns and vulnerability analysis of Chinese web passwords. IEEE Transactions on Information Forensics & Security, 2015, 11(2): 258-272

[33] Ma J, Yang W, Luo M, et al. A study of probabilistic password models//Proceedings of the IEEE Symposium on Security & Privacy. San Jose, USA, 2014: 689-704

[34] Wang D, Cheng H, Wang P, et al. Zipf's law in passwords. IEEE Transactions on Information Forensics and Security, 2017, 12(11): 2776-2791

[35] Wang D, Zhang Z, Wang P, et al. Targeted online password guessing: an underestimated threat//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria, 2016: 1242-1254

[36] Liu Gong-Shen, Qiu Wei-Dong, Meng Kui, et al. Password vulnerability assessment and recovery based on rules mined from large-scale real data. Chinese Journal of Computers, 2016, 39(3): 454-467(in Chinese)
(刘功申，邱卫东，孟魁等. 基于真实数据挖掘的口令脆弱性评估及恢复. 计算机学报，2016，39(3)：454-467)

[37] Han Wei-Li, Yuan Lang, Li Si-Si, et al. An efficient algorithm to generate password sets based on samples. Chinese Journal of Computers, 2017, 40(5): 1151-1167(in Chinese)
(韩伟力，袁琅，李思斯等. 一种基于样本的模拟口令集生成算法. 计算机学报，2017，40(5)：1151-1167)

[38] Hranický R, Holkovič M, Matoušek P, et al. On efficiency of distributed password recovery. The Journal of Digital Forensics, Security and Law, 2016, 11(2): 79-96

[39] Apostal D, Foerster K, Chatterjee A, et al. Password recovery using MPI and CUDA//Proceedings of the International Conference on High Performance Computing. Pune, India, 2012: 1-9

LI Bin, born in 1986, Ph. D. candidate. His current research interests include high-performance computing and information security.

ZHOU Qing-Lei, born in 1962, Ph. D., professor, Ph. D. supervisor. His major research interests include information security, automaton theory and computational complexity theory.

SI Xue-Ming, born in 1966, Ph. D., associate professor. His research interests include cryptography, network security and high-performance computing.

Background

Password recovery is the key technology to maintain the security of network information. Many efforts have been devoted to password recovery related researches, such as CPU, GPU and FPGA password cracking platform, Elcomsoft, John the Ripper, Hashcat and Wrathion password cracking tools, password guessing, password strength evaluation and so on. However, there is few works to study CPU, GPU and MC(Mimic Computer) three kinds of hybrid heterogeneous password recovery system and provide a unified algorithm framework for multiple password recovery algorithms.

In this paper, we analyze the PMC (Processing-Memory-Communication) features of multiple applications in password recovery field and put different code fragments on the appropriate computing components to fully mobilize the entire computing resources and achieve efficient computing. On mimic computer, we implemented the reconfigurable password recovery algorithms, and combined with CPU and GPU to form a system-level, multi-dimensional reconfigurable architecture, for different applications to select the appropriate system structure. We also designed the password exhaustive module based on regular expression and 10 Gigabit network dictionary transmission modules, to provide a complete cracking mode for password recovery. Compared with Elcomsoft commercial encryption recovery software and other reference programs, the cracking speed and energy efficiency ratio has improved significantly.

This subject belongs to 863 Program of China under Grant No. 2009AA012201, which focuses on mimic computing for innovation on computer architecture. This work is also supported by the National Key Research and Development Plan of China under Grant Nos. 2016YFB0800100 and 2016YFB0800101, which studies MD (Mimic Defense) for cyber space, and the National Natural Science Foundation of China under Grant No. 61250007, which research on software watermark technology for software security.

Our group has been involved in the research of password cracking since 2009, over eight years. We built a distributed password recovery platform, created a password recovery accelerator card, implemented a variety of high-performance recovery algorithms, and prepare for ASIC chip development.

Our study as a sub-project of mimic computing (2009AA012201), aims to achieve high efficiency computing for password recovery, while providing high performance algorithms, reduce system power consumption, through system collaboration to improve password hit ratio. It will help with cyber space defense and software security, to improve their methods in some way.