计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190300104

基于异构计算平台的规则处理器的设计与实现

陈孟东　郭东升　谢向辉　吴　东
数学工程与先进计算国家重点实验室　江苏 无锡 214125
(chen.mengdong@meac-skl.cn)

摘　要　对于身份认证机制中的安全字符串恢复,字典结合变换规则是一种常用的方法。通过变换规则的处理,可以快速生成大量具有针对性的新字符串供验证使用。但是,规则的处理过程复杂,对处理性能、系统功耗等有很高的要求,现有的工具和研究都是基于软件方式进行处理,难以满足实际恢复系统的需求。为此,文中提出了基于异构计算平台的规则处理器技术,首次使用可重构FPGA硬件加速规则的处理过程,同时使用ARM通用计算核心进行规则处理过程的配置、管理、监控等工作,并在Xilinx Zynq XC7Z030芯片上进行了具体实现。实验结果表明,在典型情况下,该混合架构的规则处理器相比于单纯使用ARM通用计算核心,性能提升了214倍,规则处理器的运行性能优于Intel i7-6700 CPU,性能功耗比相比NVIDIA GeForce GTX 1080 Ti GPU有1.4～2.1倍的提升,相比CPU有70倍的提升,有效提升了规则处理的速率和能效。实验数据充分说明,基于异构计算平台,采用硬件加速的规则处理器有效解决了规则处理中的速率和能效问题,可以满足实际工程需求,为整个安全字符串恢复系统的设计奠定了基础。

关键词:身份认证;异构;字符串;规则;处理器
中图法分类号　TP391

Design and Implementation of Rule Processor Based on Heterogeneous Computing Platform

CHEN Meng-dong,GUO Dong-sheng,XIE Xiang-hui and WU Dong
State Key Laboratory of Mathematical Engineering and Advanced Computing,Wuxi,Jiangsu 214125,China

Abstract　Using dictionaries and their transformation rules is a common method. In recovering the secure string in the identity authentication mechanism. Through the processing of the transformation rules,a large number of targeted new strings can be quickly generated for verification. The rule processing process is complex,and has high requirements on processing performance and system power consumption. The existing tools and research are processed based on software,which are difficult to meet the needs of the actual recovery system. To this end,a rule processor technology based on heterogeneous computing platform was proposed in this paper. For the first time,reconfigurable FPGA hardware is used to accelerate the process of rule processing. At the same time,the ARM universal computing core is used to configure,manage and monitor the process of rule processing. It is implemented on Xilinx Zynq XC7Z030 chip. The experimental results show that the performance of the rule processor based on the hybrid architecture is 214 times higher than that of the rule processor based on ARM only. Typically,the performance of rule processor is better than that of Intel i7-6700 CPU. Compared with NVIDIA GeForce GTX 1080 Ti GPU,the performance power ratio of rule processor is 1.4-2.1 times higher,70 times higher than that of CPU,which effectively improves the speed and efficiency of rule processing. The experimental data fully show that the speed and efficiency of rule processing can be effectively solved by using hardware-accelerated rule processor based on heterogeneous computing platform,which can meet the actual engineering requirements and provide a basis for the design of the whole secure string recovery system.

Keywords　Identity authentication,Heterogeneous,Character string,Rule,Processor

1　引言

在计算机与网络系统中,身份认证机制是一种重要的信息安全措施,作为身份识别与保护个人信息的有效手段而被广泛使用[1]。认证加密过程需要一个用户名和安全字符串组合的身份信息。

通常使用HASH算法来计算安全字符串的摘要,并将摘要值与用户凭据一同存储,如图1(a)所示。当用户进行身份认证时,认证系统接收用户输入的安全字符串,重新使用HASH算法把字符串转换成摘要,并将其与系统存储的摘要值进行对比,以区分合法用户和非法用户,完成认证过程[2],如图1(b)所示。

到稿日期:2019-03-21　返修日期:2019-07-15　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61732018)
This work was supported by the National Natural Science Foundation of China (61732018).
通信作者:谢向辉(xie.xianghui@meac-skl.cn)

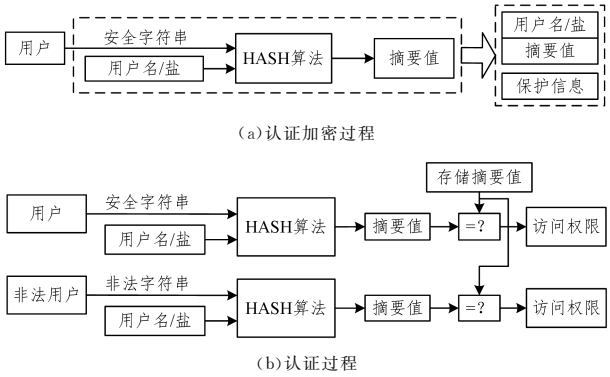

图 1 身份认证过程

Fig. 1 Process of identity authentication

安全字符串的恢复是一个逆向的过程，该过程从大量待测试字符串中尝试寻找正确的安全字符串，常被用来恢复被遗忘的安全字符串[1]。这个过程需要在短时间内快速生成大量的待测试字符串以供后续 HASH 算法计算摘要值，将计算的摘要值与存储摘要值进行比较，进而分析出正确的字符串。采用已有字符串并结合规则变换的方式是一种非常快速、有效的生成字符串的方式[3-4]，如图 2 所示。字典文件是搜集的已知常用安全字符串的集合，具有较高的命中率，将其根据变换规则进行变形，形成的新条目在保证准确率的基础上，扩大了原字符串的覆盖空间，同时也保证了一定的搜索规模，可以提高分析的成功率。字典文件中每一行是一个安全字符串，规则文件中每一行是一次变换，一次变换常常是几个规则的组合。实际中，任务规模大时下，字典文件可能多达几亿个条目，规则文件可能包含几十万次变换。规则的处理过程中，字典文件中的每一个条目都需要经过规则文件中所有变换的处理，生成的新字符串数量是字典条目和规则条目的乘积。

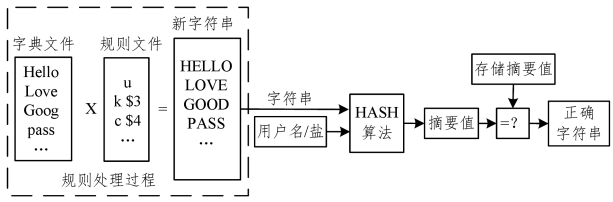

图 2 基于变换规则的安全字符串恢复过程

Fig. 2 Secure string recovery process based on transform rules

性能和功耗成本是变换规则处理乃至安全字符串恢复系统的重要因素。变换规则包括几十种，且有复杂的组合情况，本身是一项计算量大、对处理速度要求很高的任务。到目前为止，公开的实现方式都是基于 CPU 和 GPU[4-5] 进行处理，在处理速度、系统功耗等方面存在诸多不足。

本文针对安全字符串恢复中的规则处理，以及实际工程中使用的基于 Xilinx Zynq 7Z030 芯片的安全字符串恢复系统，提出了一种基于混合异构平台的适合于规则处理的架构，并首次将主要的规则处理功能通过 FPGA 硬件进行加速处理，实现了高能效、可重构的规则处理器。实验结果表明，该处理器能满足实际工程需求，在处理性能、系统能耗等方面表现良好。

2 相关工作

人们在设置安全字符串时，常常将一个基础简单变形成新的字符串，如判断并替换特定字符，将所有大写字母变换为小写，字符倒序，特定位置插入或删除字符等，这种变换被称为变换规则[4-5]。规则通常组合在一起共同进行一次变换，以生成一个新的字符串。如图 3 所示，字符替换、字母的大写变换、增加后缀 3 个规则一起使用，对初始字符串进行变换。

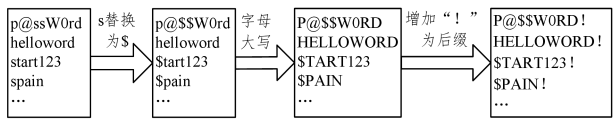

图 3 字符串变换规则

Fig. 3 String transformation rules

在安全字符串恢复中使用字典加规则的方式越来越受到重视，已有多个研究证明了规则的有效性[6-8]。文献[9-10]也有基于规则的分析研究，其规则处理过程都由软件实现；文献[11-12]的规则处理过程在 GPU 上实现。

业界已有许多总结积累而成的字符串变换规则，多个安全字符串恢复工具均有自己支持的规则，并提供字典加规则的恢复模式。

John the Ripper[5] 是一款开源且免费的密码分析软件，其主要被用来恢复弱的 Unix 口令，现已支持一百余种算法，提供对多种不同类型系统架构的支持，包括 Unix/Linux，Windows/DOS 模式和 OpenVMS 等。它支持字典恢复模式，并支持十大类共 40 余种字符串的变换规则及处理，其规则处理在 CPU 上进行。多篇文献[13-15] 使用 John the Ripper 及其自带的变换规则进行安全字符串的恢复，分析过程包括规则处理过程都是在 CPU 上通过软件实现。

hashcat[4] 是一款多平台的免费恢复套件，支持 CPU，GPU（NVIDIA GPU，AMD GPU），DSP，FPGA 等包含 OpenCL 运行时的各种平台；支持 Linux，Windows，MacOS 多种操作系统；支持分布式处理；支持近 200 种算法；支持多种恢复模式；支持字典与规则的处理，其规则的处理过程主要在 CPU 和 GPU 上进行。文献[16]使用 hashcat 及自带的规则在 Intel Xeon CPU 上进行实验，但没有统计单独的规则处理时间，且缺少与功耗相关的研究数据。文献[17-18]都采用 hashcat 中字典加规则的分析方式进行实验，但都是基于 GPU 平台。

通过以上分析可以发现，不管是学术研究还是开源软件、商业工具，对规则的处理都是基于软件在 CPU 或 GPU 平台上实现，实现过程关注规则的有效性，没有考虑实际恢复系统时大规模规则情况下的处理速率和能耗需求。

本文以开源工具 hashcat 所用的规则为基础，选取 41 种常用的基本变换规则及其自带的 30 余万个条目的规则文件进行研究，实现其处理器。表 1 列出了几种典型的变换规则，每一个规则各自以一个可见字符为其助记符，部分规则需要携带参数，参数的个数从 0～3 不等。表 1 中以字符串“p@ssW0rd”为输入，举例说明了基本规则的变换结果。在实际使用中，单个规则常常是组合在一起共同进行一次变换。例如，“lD3ss$”为 3 个规则组合的情况，首先将字符串中的所有大写字母变为小写，然后删除第 3 个位置的字符，最后将字符串中的“s”替换为“$”，所有规则处理结束后生成 1 个新的字符串。

表 1 典型规则的示例

Table 1 Examples of typical rules

助记符	定义	实例	变换结果
l	所有字母小写	u	p@ssw0rd
r	倒序	r	dr0Wss@p
pN	整体重复 N 遍作为后缀	p2	p@ssW0rdp@ssW0rdp@ssW0rd
{	循环左移	{	@ssW0rdp
DN	去除 N 位置的字符	D3	p@sW0rd
iNX	在 N 位置插入字符 X	i4!	p@ss!W0rd
sXY	所有的 X 替换为 Y	ss$	p@$$W0rd
*XY	交换 X、Y 位置的字符	*34	p@sWs0rd
+N	N 位置字符在 ASCII 表序中增加 1 位	+2	p@tsW0rd

3 基于异构混合架构的规则处理器的设计

为了追求更高的处理速率和更低的处理成本，从而满足安全字符串恢复的需求，规则处理面临一系列的挑战。首先，变换规则种类繁多、规模大，单个规则处理过程复杂，并且存在复杂的规则组合情况。其次，规则处理的速率需求高。一次任务需要处理的规则变换数量多，需要处理的初始字符串数量大，在实际恢复系统中，新字符串的生成速度须满足认证算法模块的需求；而认证算法模块多以流水线实现，且高度并行化，对规则的处理速度提出了很高的要求。最后，规则处理面临着资源、功耗等限制，尤其是在基于硬件的恢复系统中，认证算法和规则处理都需要占用较多的逻辑资源，都会产生较大的功耗，规则处理面临更苛刻的限制。

3.1 异构平台结构

本文的规则处理器基于一个混合异构的计算平台，该平台的核心部件是一个混合核心处理器 Xilinx Zynq XC7Z030[19]芯片，其包含了两个主频为 1GHz 的 ARM 通用嵌入式计算核心和一个可重构计算核心。两种异构计算资源通过高速的互连总线紧密耦合，可以支持通用计算任务和加速计算任务的并行协同执行。混合核心处理平台外围集成了 1GB 的低功耗 DDR 内存、32GB 的 Flash 存储器、千兆以太网接口、高速环形网接口等，其结构图如图 4 所示。

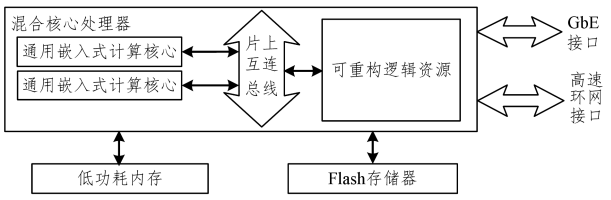

图 4 混合异构计算平台结构图

Fig. 4 Structural diagram of hybrid heterogeneous computing platform

混合核心计算平台的主要计算能力来自可重构计算核心，即 FPGA，适合进行计算量大、易于并行化、适于硬件处理的工作；通用计算核心适合进行灵活性较强的配置管理工作。因此在设计规则处理器时，将主要的规则处理功能放在 FPGA 中进行，通用计算核心负责对可重构计算核心的工作进行配置与管理，通过软件与硬件的有效协同达到最优性能。

3.2 规则处理器的设计

本文按照自顶向下的设计方法，首先给出规则处理器的总体架构，然后介绍详细的设计细节。

3.2.1 硬件部分总体结构的设计

硬件部分的总体结构如图 5 所示，其主体为 41 个规则执行单元(Rule Execute Unit，REU)。

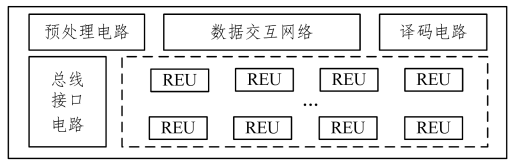

图 5 硬件规则处理器结构图

Fig. 5 Structural diagram of hardware rule processor

REU 负责各个规则进行处理功能；规则译码电路负责对变换规则进行译码，对各个 REU 进行调度；总线接口电路包含 AXI 总线接口，并负责与片外数据交互；预处理电路负责将连续存储的规则和字典数据处理成规则变换条目和字典条目；数据交互网络负责处理节点内部数据的交互，并将最终生成的安全字符串写入 FIFO 中供 HASH 模块使用。

3.2.2 REU

REU 负责实现每个规则的变换功能。在实际工程中，HASH 算法模块多以流水线实现，每个时钟周期都可以对一个安全字符串进行验证，对于资源占用少的认证算法，一个片内甚至可以放置多个算法流水线，这对规则执行的速率提出了很高的要求。为了尽量提升规则执行的速率，将单个规则执行的过程进行优化实现，使其在一个时钟周期内完成，在译码电路的调度下，做到一个时钟周期处理完一个基本规则。当规则组合在一起进行一次变换时，一次变换的执行过程如图 6 所示，图中以 3 个规则组合使用为例进行说明，3 个时钟周期执行完毕。

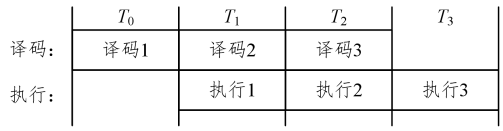

图 6 一次规则变换的执行过程示意图

Fig. 6 Schematic diagram of the execution process of a rule transformation

对于大量存在的规则组合使用的情况，即便基本规则在 1 个时钟周期之内执行完成，完整的规则变换过程处理完毕，生成一个新的字符串仍需要多个周期。此种情况下，每个时刻只有一个 REU 在工作，处理速率较低。因此，需要在译码电路的控制下，尽量将整个变换过程做到全流水实现。

3.2.3 译码电路

译码电路负责对规则进行译码，判断具体执行哪种变换，并根据译码结果对 REU 进行调度。调度分为两种方式：全流水的方式和串行的方式。

在只有一套规则处理单元的情况下，只有同一个时刻中没有处理单元发生冲突时，才可以做到全流水处理。规则处理过程需要对所有的字典条目和规则条目进行遍历组合，并对处理的流程进行调整优化。首先，固定一个规则变换，对所有的字典条目进行循环；然后，更换规则，对规则进行循环。这种方式在一段时间内处理的是同一个规则变换，这为规则

的全流水执行提供了基础。通过对自带规则进行统计分析发现，有 85％的变换中一次变换使用的基本规则是不同的，不会造成硬件执行单元的冲突，可以进行流水处理。

本文设计的全流水的执行结构如图 7 所示。

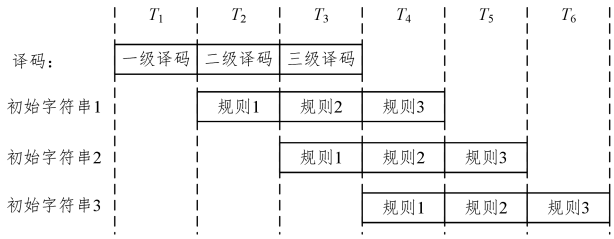

图 7　支持全流水的规则执行结构

Fig. 7　Rule execution architecture supporting full flow

每个时钟周期都有一个安全字符串处理完毕，且有一个新的字符串生成，每个时刻都有多个规则处理单元在同时工作，无须插入气泡和等待，最大限度地提升了规则处理的效率。对于另外少部分的情况，当一次变换中存在相同的基本规则时，通过译码电路的有效控制，数据严格按照串行方式进行处理，即一个安全字符串被完全处理完毕后才开始下一个处理工作，此时生成一个新的字符串需要多个时钟周期。

3.2.4　数据交互网络

任何规则之间都存在组合的可能性，对于 41 个 REU，由于两两之间都需要数据交互的通路，因此需要设计一个支持全互联的数据交互网络。在实际资源受限的情况下，数据交互网络的设计以低复杂度、低资源占用为目标。对规则组合情况进行统计分析发现，每次变换都是由几个基本规则组合而成，得到的结果如图 8 所示。

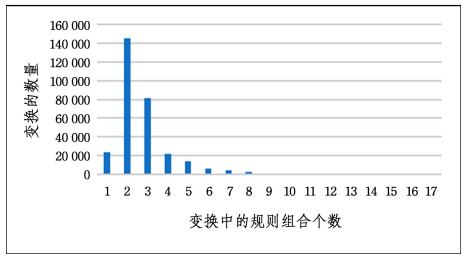

图 8　变换中的规则组合数量分布

Fig. 8　Distribution of rule combination numbers in transformations

可以看出，30 多万次变换中，规则的组合数分布于 1 到 17 之间，但是绝大多数的变换中，规则的组合数量小于 4。因此本文所设计的硬件规则处理器对规则组合数小于 4 的情况进行了重点优化，并据此设计了数据交互网络，以满足任意 REU 间的数据传递。

根据以时间换取空间的思想，通过将数据交互的时间延长来缓解硬件连线的复杂度，据此本文设计了基于三级公共寄存器的分层的数据交互网络，如图 9 所示。

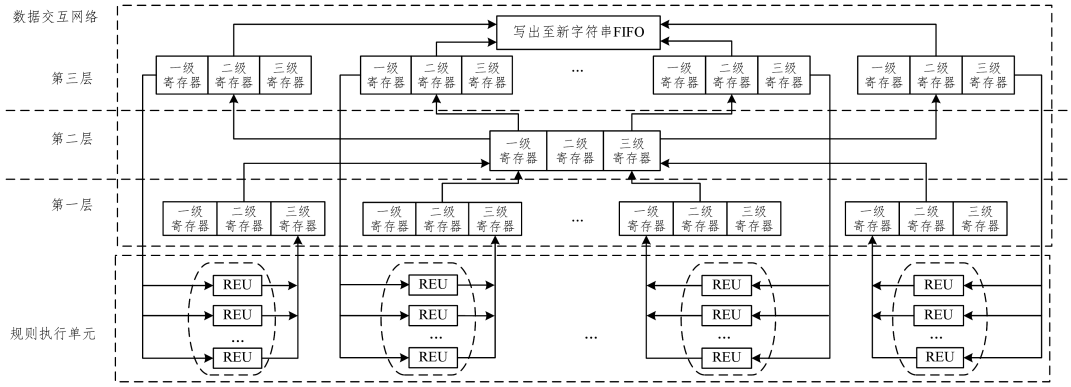

图 9　数据交互网络

Fig. 9　Data interactive network

处理过程中，每一个 REU 的结果不是直接输入到后续的的 REU 中，而是将结果存入公共寄存器中，后续的 REU 工作时再从公共寄存器中取数据。仍以变换“lD3ss＄”为例，将第一个规则“l”的处理结果存入一级寄存器中；处理第二个规则“D”的 REU 从一级寄存器中取数据，处理结果存入二级寄存器中；处理第三个规则“s”的 REU 从二级寄存器中取数据，处理结果存入三级寄存器中；译码电路会给出目的标记信息，表示该基本规则在组合中处于第几级。鉴于 83％以上的变换中规则组合的数量小于 4，处理器对这种大多数的情况进行重点支持，将公共寄存器的级数设置为 3，以保证组合数量小于 4 的变换可以进行流水处理。

如果将 41 个 REU 都连接到一组公共寄存器中，仍然存在连线的复杂性问题。对此，本文采用分层的方式，通过 3 层网络实现数据的汇聚与发散。将 REU 分为 6 组，每组内部 8 个 REU，同一组内的 REU 连接到一组公共寄存器中，这些公共寄存器组成第一层交互网络。REU 运行的结果数据经过第一层交互网络后，再通过第二层交互网络汇聚到一组公共寄存器中，接着通过第三层网络分散到 6 组公共寄存器中，最终到达下一级的每个 REU 中。数据的流动通过译码电路进行有效控制。通过分级和分层的数据交互网络，可以大大减小 REU 间互联的连线复杂度，降低工程实现的难度，提高硬件实现所能达到的频率。

对于变换中不存在相同基本规则且组合数小于 4 的情况，任一时刻每一级公共寄存器只会被一个 REU 占用，且交互花费的时间是相同且固定的，该数据交互网络可以保证全流水执行。对于一次变换中存在相同基本规则的情况，按照串行执行，通过译码电路的控制，仅使用部分公共寄存器，该数据交互网络可以做到有效支持。对于少数规则组合数大于 3 的情况，通过译码电路的控制，也按照串行方式进行处理，该网络结构也可以兼容。当规则按照串行方式进行处理时，

规则处理本身占用一个时钟周期，每级数据交互占用一个时钟周期，所以完成1个基本规则的处理共需2个时钟周期。比如，若变换由4个基本规则组合而成，则处理生成1个新的字符串需要8个时钟周期。

3.2.5 存储架构

整个规则处理器的数据存储架构共分为三级。

第一级存储：片外DDR。整个规则文件和字典文件都存储在DDR中，通用计算核心将规则文件和字典文件在DDR中的起始地址以及大小信息配置到可重构FPGA，硬件自动进行规则和字典的获取。同时，如果需要将规则处理器生成的新字符串传到片外供其他应用使用，则生成的新字符串也是由规则处理器自动传输到DDR内存中。

第二级存储：片内RAM。FPGA内部的规则处理器通过高级可扩展接口AXI总线将规则与字典分批预取入FPGA内部，然后将其缓存于片内RAM中。AXI总线的4个高性能接口AXI_HP工作于150MHz时，总带宽达到4.8GB/s，可以保证对硬件规则处理器的高速供数。随着处理逻辑不断消耗RAM中的数据，规则处理器会不断从片外获取数据，以保证处理逻辑的需求。

第三级存储：片内FIFO。处理逻辑从片内RAM获取字典数据，并对其进行预处理，然后将字典存入片内FIFO缓存中供核心处理逻辑进行高速处理。

3.2.6 通用计算核心功能的设计

通用计算核心主要完成对硬件规则处理器的配置管理工作。在硬件规则处理器工作前，通用计算核心须将规则和字典文件移入内存中，并将字典和规则在内存中存放的起始地址、大小信息、初始字符串长度信息等内容配置到规则处理器，同时发出启动控制信号。硬件规则处理器自己读取字典和规则内容，自动进行变换处理。在工作过程中，通用处理器负责监视硬件的工作状态。

4 实验与结果

本文基于实际的安全字符串恢复系统，在Zynq XC7Z030芯片上对所设计的规则处理器进行开发实现，并与其他平台的运行结果进行对比和分析。

4.1 规则处理器性能实验

首先，对本文的规则处理器进行实验；然后，基于相同的规则和字典文件，单纯使用ARM通用处理器进行处理，并对前后两者的处理性能进行对比。

在ARM通用计算核心上开发配置管理程序，通过Vivado工具，对FPGA设计部分进行综合与实现。结果显示，基于通用计算核心和可重构FPGA的规则处理器的运行结果正确，硬件处理部分可以达到的最高工作频率为150 MHz，硬件资源占用率为70%，FPGA内部仍有足够的资源来放置认证算法模块以进行片内的验证，也可将处理生成的字符串通过AXI总线传至片外供其他应用使用。在典型情况下，规则处理以全流水方式进行，每秒处理150 M次规则变换，生成150 M个新字符串。在少数的规则处理串行执行的情况下，处理性能与规则的组合情况有关，当4个规则组合时，8个时钟周期生成1个新的字符串，所以每秒钟生成的新字符串个数为18.75 M。

基于上述相同的规则和字典文件，单纯使用芯片内的通用计算核心进行处理工作，实验结果如表2所列。可见，在典型情况下，相比单纯使用ARM进行处理，加入FPGA硬件处理后，处理器性能提升了214倍；对于少数需要串行处理的情况，仍有37.5倍的性能提升。实验结果证明，采用ARM和FPGA构成的异构规则处理器具有性能优势。

表2 规则处理器的性能

Table 2 Performance of rule processor

实现方式	典型情况性能	典型情况加速比	串行性能	串行情况加速比
ARM	700 k	1	500 k	1
ARM+FPGA	150 M	214	18.75 M	37.5

4.2 与其他平台的对比

本文首次以FPGA硬件加速的方式进行规则的解析处理工作，将其与intel CPU和NVIDIA GPU上的软件实现进行性能和功耗的对比。对比时，将相同的规则文件和字典文件分别在3种平台上运行，得出实验数据。

软件实现采用最新的hashcat 5.1.0。hashcat号称是业界最快的恢复工具，支持CPU和GPU平台[4]。软件的结果与其运行平台有很大关系，如NVIDA GPU中桌面产品和专门用于高性能计算的产品之间计算能力差距非常巨大。本文选取主流偏上的产品平台进行实验，采用的CPU平台为Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz，32 GB内存；GPU平台为NVIDIA GeForce GTX 1080 Ti。

对于规则组合情况，在每种平台上分别测试了1个规则、2个规则、3个规则组合成1次变换的情况。以字典中长8个字符的字符串为例进行实验，处理性能以每秒生成新字符串的数量(单位为M)来进行衡量，结果如表3所列。

表3 字典长度为8时的性能对比

Table 3 Performance comparison when dictionary length is 8

(单位：M/s)

平台	规则1组合	规则2组合	规则3组合
本文规则处理器	150	150	150
CPU	220	100	70
GPU	1700	12050	8000

通过分析可以发现：因为是全流水实现方式，规则的组合情况对本文处理器的处理性能没有影响，但对CPU和GPU平台的影响较大，规则组合越多，其处理性能越差。2个或者3个基本规则组合在一起进行一次变换占据绝大多数情况，此时本文规则处理器每秒生成150 M个新字符串，处理性能优于CPU平台，差于GPU；然而，当使用该规则处理器来构建大规模、分布式的规则处理系统时，其计算能力会显著增加。

改变字典中字符串的长度为12个字符，重新进行实验，结果如表4所列。分析可见，在不同字典长度的情况下，3个平台的处理性能都基本不受影响。

表4 字典长度为12时的性能对比

Table 4 Performance comparison when dictionary length is 12

(单位：M/s)

平台	规则1组合	规则2组合	规则3组合
本文规则处理器	150	150	150
CPU	220	100	70
GPU	17000	12000	7800

在实际运行过程中，对规则引擎和 GPU 平台的运行功耗进行实时观测，CPU 的功耗以 65 W 计算，性能功耗比（每瓦特每秒可以处理的规则变换数量）的计算结果如图 10 所示。对于 2 个或 3 个基本规则组合在一起的典型情况，本文规则处理器的性能功耗比相比 GPU 有 1.4～2.1 倍的提升，相比 CPU 有 70 倍的提升。可以看出，本文的规则处理器在能效方面具有优势，其运行速度快、功耗成本低，适合于构建大规模的规则处理系统。

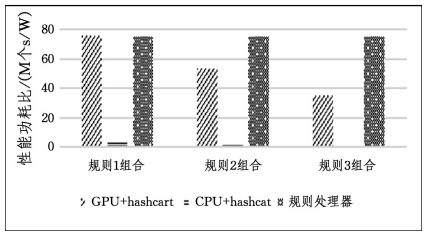

图 10 不同平台上的性能功耗比较

Fig. 10 Comparison of energy efficiency between different platforms

结束语 字符串变换规则的处理是安全字符串恢复的一个重要部分，其处理过程复杂，对处理性能、系统功耗有很高的要求。本文针对这些需求，首次提出了基于异构平台以硬件进行加速的规则处理架构，有效利用 FPGA 高并行、低功耗以及通用计算核心灵活性的特点，构建了规则处理器，并进行了设计实现。实验结果表明，典型情况下，规则处理器在 Zynq XC7Z030 FPGA 上的运行性能优于 Intel i7-6700 CPU，性能功耗比相比 NVIDIA GeForce GTX 1080 Ti GPU 有 1.4～2.1 倍的提升，相比 CPU 有 70 倍的提升，有效提升了规则处理的速率和能效。本文设计的规则处理器处理性能高，系统成本低，运行功耗低，特别适合后续构建大规模、分布式、可重构的规则处理系统，进而为整个安全字符串恢复系统的设计与实现奠定基础。

参 考 文 献

[1] GUO X. The Scale-free Network of Passwords: Visualization and Estimation of Empirical Passwords [EB/OL]. [2019-03-18]. http://arxiv.org/pdf/1511.08324v1.

[2] WU Z Y, CHIANG D L, LIN T C, et al. A Reliable Dynamic User-Remote Password Authentication Scheme over Insecure Network [C]// International Conference on Advanced Information Networking & Applications Workshops. IEEE, 2012.

[3] LIU J. Research and application of password cracking method based on special dictionary [D]. Harbin: Harbin Institute of Technology, 2015.

[4] STEUBE J. hashcat advanved password recovery [EB/OL]. [2019-03-18]. https://hashcat.net/hashcat/.

[5] Openwall. John the Ripper password cracker [EB/OL]. [2019-03-18]. http://www.openwall.com/john/doc/.

[6] DAS A, BONNEAU J, CAESAR M, et al. The Tangled Web of Password Reuse[C]// Network and Distributed System Security Symposium. 2014.

[7] TATLI E I. Cracking More Password Hashes With Patterns [J]. IEEE Transactions on Information Forensics & Security, 2015, 10(8): 1656-1665.

[8] NSAKEY. Password cracking rules and masks for hashcat that i generated from cracked passwords [EB/OL]. https://github.com/NSAKEY/nsa-rules.

[9] GOODIN, D. Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331" [EB/OL]. http://arstechnica.com/security/2013/05/how-crackers-make-minced-meat-out-of-yourpasswords/.

[10] UR B, SEGRETI S M, BAUER L, et al. Measuring real - world accuracies and biases in modeling password guessability [C]// Usenix Conference on Security Symposium. 2015.

[11] VERAS R, THORPE J, COLLINS C. Visualizing semantics in passwords: The role of dates[C]// Proceedings of the Ninth International Symposium on Visualization for Cyber Security (VizSec '12). New York, NY, USA: ACM, 2012: 88-95.

[12] MELICHER W, UR B, SEGRETI S M, et al. Fast, lean, and accurate: Modeling password guessability using neural networks [C]// 25th USENIX Security Symposium (USENIX Security 16). Austin, TX: USENIX Association, 2016: 175-191.

[13] CHOU H C, LEE H C, YU H J, et al. Password cracking based on learned patterns from disclosed passwords [C]// IJICIC. 2013.

[14] FAHL S, HARBACH M, ACAR Y, et al. On the ecological validity of a password study[C]// Symposium on Usable Privacy and Security. 2013: 1-13.

[15] DE CAMAVALET X C, MANNAN M. From very weak to very strong: Analyzing password—strength meters[C]// Proceedings of the Network and Distributed System Security Symposium. 2014.

[16] HUH J H, OH S, KIM H, et al. Surpass: System-initiated User-replaceable Passwords[C]// The 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015: 170-181.

[17] MILO F, BERNASCHI M, BISSON M. A fast, GPU based, dictionary attack to OpenPGP secret keyrings[J]. Journal of Systems & Software, 2011, 84(12): 2088-2096.

[18] XU L, GE C, QIU W, et al. Password Guessing Based on LSTM Recurrent Neural Networks[C]// IEEE International Conference on Computational Science and Engineering. IEEE, 2017: 785-788.

[19] Xilinx. Zynq-7000 All Programmable SoC [EB/OL]. [2019-03-18]. https://www.xilinx.com/support/documentation/product-briefs/zynq-7000-product-brief.pdf.

CHEN Meng-dong, born in 1984, postgraduate, engineer. His main research interests include computer architecture.

XIE Xiang-hui, born in 1958, Ph.D, senior engineer, Ph. D supervisor. His main research interests include computer architecture.