基于动态异构冗余架构的微控制器设计与实现

欧阳玲[1,2*],宋 克[1],兰巨龙[1]
(1. 战略支援部队信息工程大学信息技术研究所,河南郑州 450002;2. 中原工学院电子信息学院,河南郑州 451191)

摘 要: 针对物联网环境下微控制器面临的功能安全与信息安全的"两安融合"问题,提出了一种动态异构冗余微控制器架构. 该架构设计了一种软硬件协同的调度裁决器,将策略分发、策略裁决和策略调度一体化实现,兼顾调度裁决器自身的安全性及高效性要求;设计了一种基于分布式共识的异构执行体同步算法,解决多个异构微控制器运行状态同步和数据一致性问题;设计实现了适用于动态异构冗余微控制器的原型系统及测试平台,证明所提架构在保证功能正常、延迟提高约1%的情况下,白盒插桩测试下的抗攻击能力提升了2个数量级以上.

关键词: 两安融合;微控制器;动态异构冗余;调度裁决器;分布式共识
基金项目: 国家自然科学基金(No.61572520)
引用格式: 欧阳玲,宋克,兰巨龙. 基于动态异构冗余架构的微控制器设计与实现[J]. 电子学报,XXXX,XX(XX):1-11. DOI:10.12263/DZXB.20221292
中图分类号: TN47;TP368 **文献标识码:** A **文章编号:** 0372-2112(XXXX)XX-0001-11
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20221292

Design and Implementation of Microcontroller Based on Dynamic Heterogeneous Redundancy Architecture

OUYANG Ling[1,2*], SONG Ke[1], LAN Ju-long[1]
(1. *Institute of Information Technology, Information Engineering University, Zhengzhou, Henan* 450002, *China*;
2. *School of Electronic Information, Zhongyuan University of Technology, Zhengzhou, henan* 450002, *China*)

Abstract: In this paper, we propose a dynamic heterogeneous redundant microcontroller architecture to address the convergence problem of functional safety and information security faced by microcontrollers in the IoT environment. In this architecture, we design a hardware-software co-design scheduling adjudicator, which integrates policy distribution, policy adjudication, and policy scheduling, considering the security and efficiency requirements of the scheduling adjudicator. we design a distributed consensus-based synchronization algorithm for heterogeneous executors to solve the problems of the operation state synchronization and data consistency among multiple heterogeneous microcontroller cores. Also, we design and implement a prototype system and a testing platform that are applicable to dynamic and heterogeneous redundant microcontrollers. It is proved that under the condition that the function of the proposed architecture is normal and the delay is increased by about 1%, the anti-attack ability under the white box pile insertion test is improved by more than 2 orders of magnitude.

Key words: safety and security integration; microcontroller; dynamic heterogeneous redundancy; scheduling arbiter; distributed consensus
Foundation Item(s): National Natural Science Foundation of China (No.61572520)
Citation: OUYANG Ling, SONG Ke, LAN Ju-long. Design and Implementation of Microcontroller Based on Dynamic Heterogeneous Redundancy Architecture[J]. Acta Electronica Sinica, XXXX, XX(XX): 1-11. DOI: 10.12263/DZXB.20221292

收稿日期:2022-11-09;修回日期:2023-02-10;责任编辑:孙瑶
*通讯作者:欧阳玲

1 引言

微控制器(Microcontroller Unit,MCU)是将CPU频率与规格做适当缩减,并与内存、计数器、USB、A/D转换、通用异步收发器(Universal Asynchronous Receiver Transmitter,UART)、可编程逻辑控制器(Programmable Logic Controller,PLC)、直接存储器存取(Direct Memory Access,DMA)等周边接口都整合在同一芯片上的单芯片微型计算机[1]. 经过半个世纪的发展,MCU已广泛应用在消费电子、汽车电子和工业控制等领域,随着物联网时代的来临,工业互联网、智能网联汽车、智能家居等行业的迅猛发展进一步推动了MCU的发展,对MCU的需求也将迎来新的高峰[2]. 然而,物联网在为MCU带来巨大发展契机的同时,也同样给MCU带来了新的挑战,其中MCU的安全性就是最值得重点关注的一环. 例如工业互联网使原本处于封闭的工业控制系统面临日益严峻的网络安全风险,一旦工业控制系统遭到攻击,不仅可能引发故障停机,还会导致安全事故发生,甚至影响正常公共服务,给社会带来不可估量的损失. 因此,作为物联网终端控制器件,MCU的安全性成为其必备的功能特性.

在此背景下,国内外相关厂商均开始重视MCU安全技术的研究,并推出了一系列具有安全特性的MCU产品. 恩智浦公司在MPC574x系列MCU中设计了两个内核,通过一个与主内核锁步运行的检查内核来实现安全运行,检查内核执行与主内核运行相同的指令,通过检查单元将两个内核的地址和数据总线进行对比以检测运行偏差,并将检测到的错误报告给错误收集和应对模块[3]. 英飞凌公司在其AURIX系列MCU产品中,则集成了最多三个核,其中一个主核与启动ROM(Read-Only Memory)和AES128(Advanced Encryption Standard 128)加密模块以及其他一些重要模块,通过逻辑隔离的方式运行在硬件安全模块中,其余两个核作为校验核[4]. 国内MCU厂商芯驰科技2022年上半年发布了针对汽车安全相关应用设计的新一代高性能微控制器产品E3-控之芯系列车规MCU,采用双核锁步技术,最多可提供3对6个ARM Cortex R5内核[5]. 总体来说,采用冗余内核设计是目前国内外MCU厂商普遍采用的安全MCU设计技术. 然而,上述设计中的冗余结构均使用同构处理核心构建,即采用传统的冗余控制思想解决偶发性故障引起的功能安全问题,对于外部攻击而造成的信息安全问题基本上无能为力,而叠加加密、认证、可信等安全技术手段又受限于MCU自身成本、功耗、资源多种因素影响. 因此,将MCU功能安全和信息安全进行一体化考虑,从而设计具备"双安融合"的MCU意义重大.

动态异构冗余(Dynamic Heterogeneous Redundancy,DHR)结构是我国邬江兴院士团队提出的一种新型安全体系结构[6],基于策略裁决的闭环迭代式多维动态重构鲁棒控制结构,将因自身故障引起和因外界攻击造成的系统错误统一定义为"广义扰动",将功能安全与信息安全问题统一为"广义鲁棒性"问题[7],可以作为一种普适性的使能技术用于各种软硬件设备的基础构造设计和集成应用创新. 目前已成功应用于交换机[8]、路由器[9]、Web服务器[10]、DNS服务器[11]等网络设备,以及区块链[12]、数据库[13]、软件定义网络[14]、云计算[15]、自动控制[16]等领域. 本文将DHR结构与MCU设计相融合,提供一种集可靠性和安全性为一体的DHR-MCU架构,主要工作如下:

(1)提出了一种异构四冗余度的DHR-MCU架构,克服了短时安全降级时的输出中断及高错误率问题;

(2)设计了一种软硬件协同的DHR调度裁决器,兼顾了DHR架构自身的安全性及高效性要求;

(3)设计了一种基于分布式共识的异构执行体同步算法,解决多异构MCU的运行状态同步问题;

(4)实现了原型系统及其测试平台,通过测试证明DHR-MCU架构具备较高的广义鲁棒性.

2 DHR-MCU架构设计

2.1 四冗余度DHR-MCU架构

DHR的典型结构如图1所示,由功能等价的异构执行体以及策略分发、策略裁决和策略调度构成. DHR结构通过对执行体的"异构"和"冗余"设计,使得在绝大多数情况下,特定的攻击只会影响个别执行体并产生输出响应,并通过裁决被当成异常输出屏蔽;同时,结构的"动态"调度又解决了极少数情况下,因共性缺陷或协同攻击导致的多个执行体返回一致的攻击响应而造成的短期攻击逃逸的问题. 显然,如果用随机性差模故障事件替换已知或未知的攻击事件,DHR结构具有完全相同或相似的效果[17].

DHR的理论基础是建立在"多数正确"的逻辑表达上,在考虑系统复杂度的情况下,工程实现上多采用三模冗余(Triple Modular Redundancy,TMR). 前文提到的基于DHR构建的应用系统多是采用TMR结构,理论上可证明在常规扰动和恢复模式下具备足够的广义鲁棒性. 然而,TMR结构的最大问题在于,当清洗恢复的异构执行体还未完成时,如果剩余两个异构执行体又出现输出不一致,一般会采用暂停输出或者随机输出某个异构执行体的方式处理[18]. 而在MCU的应用环境中,比如工业控制领域,需要考虑系统工作的可靠性和稳定性,很多应用场景是不允许暂停输出的,而随机输出高达50%的错误率也很难接受. 因此,本文设计了一种四模冗余QMR(Quadruple Modular Redundancy,

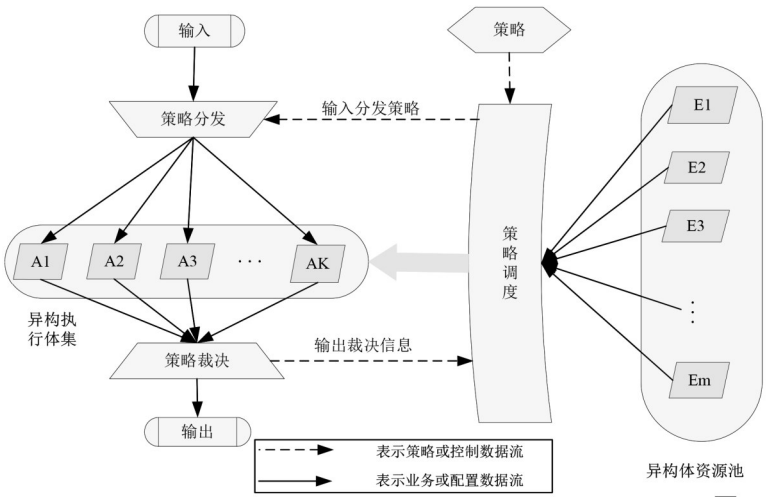


图1 动态异构冗余DHR典型结构

QMR)的DHR-MCU结构,在TMR结构的基础上,增加一个热备份异构MCU,并支持工作模式切换,可有效缓解上述问题. 图2所示为四冗余度DHR-MCU的系统架构,其中异构MCU组是完成系统运算的核心部件,由四个不同底层架构的MCU、运行异构操作系统(根据具体应用也可无操作系统)和经过多样化编译的控制程序组成. 架构采用一个统一的调度裁决器,将DHR结构中的策略分发、策略裁决以及策略调度等功能在一个组件上实现.

四个异构的MCU同时工作,共同处理相同的输入控制请求数据,处理后的输出控制结果数据经过调度裁决器进行比对后输出. 正常情况下,多个异构MCU的输出应该是一致的,经过比对后进行正常输出;但在故障或者受到攻击时,某个MCU的输出就会与其他MCU的输出不一致,经过比对裁决,就可以将这种不一致的输出进行排除,保障整个系统的稳定和安全. 为了便于比对裁决,本文提出的DHR-MCU由四个异构的MCU组成,正常状态下,其中三个参与输出比对,另一个处于热备份状态,随时替换出问题被清洗的MCU.

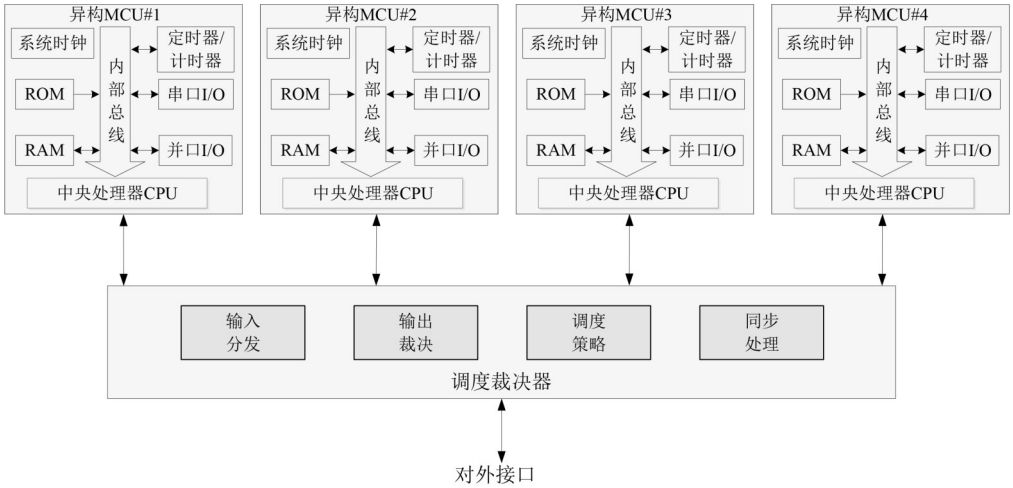


图2 DHR-MCU系统整体架构示意图

2.2 DHR-MCU架构安全性分析

四冗余度DHR-MCU架构共有五种工作状态,其状态定义与转移关系如图3所示. 其中q为时刻t单个MCU被攻击成功的概率,p为被清洗恢复成功的概率. 假定时间足够长的时候,单个MCU必然受到攻击,且肯定可以被清洗恢复成功,则有:

$$\begin{cases} q=\int_0^t q(t)\mathrm{d}t \\ p=\int_0^t p(t)\mathrm{d}t \end{cases} \tag{1}$$

同时设清洗的空间是足够大的,也就是说在任意同一时刻可以保证多个MCU处于清洗恢复状态,并且清洗结束到恢复运行的时间可以忽略不计. 由于每个

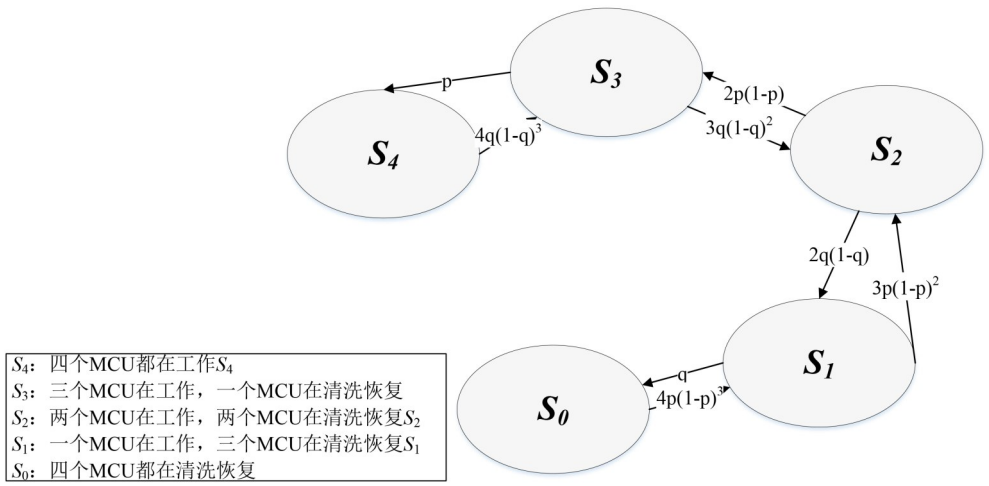


图3 DHR-MCU工作状态转移图

MCU都是独立运行的,且处于运行状态的每一个MCU都有可能被攻击成功(由于是异构MCU故可认为各MCU被攻击成功是相互独立的);处于清洗恢复状态的每一个MCU都能够被成功清洗恢复(同样各MCU被清洗恢复成功也是相互独立的). 因此,对于每一个MCU,下个时刻是处于运行状态还是清洗恢复状态,与系统中其他MCU被攻击成功或者被清洗恢复成功没有关系,满足马氏性的要求,因此其我们也可以用马尔可夫过程来求解MCU的状态概率,可得状态转移过程的矩阵为

$$Q=\begin{bmatrix} 1-4q(1-q)^3 & 4q(1-q)^3 & 0 & 0 & 0 \\ p & 1-p-3q(1-q)^2 & 3q(1-q)^2 & 0 & 0 \\ 0 & 2p(1-p) & 1-2p(1-p)-2q(1-q) & 2q(1-q) & 0 \\ 0 & 0 & 3p(1-p)^2 & 1-3p(1-p)^2-q & q \\ 0 & 0 & 0 & 4p(1-p)^3 & 1-4p(1-p)^3 \end{bmatrix} \tag{2}$$

从而可以得到各个状态的概率:

$$\begin{cases} s_0=\dfrac{q^4(1-q)^6}{\text{sum}} \\ s_1=\dfrac{4p(1-p)^3q^3(1-q)^6}{\text{sum}} \\ s_2=\dfrac{6p^2(1-p)^5q^2(1-q)^5}{\text{sum}} \\ s_3=\dfrac{4p^3(1-p)^6q(1-q)^3}{\text{sum}} \\ s_4=\dfrac{p^4(1-p)^6}{\text{sum}} \end{cases} \tag{3}$$

其中,

$$\begin{aligned} \text{sum}=&p^4(1-p)^6+4p^3(1-p)^6q(1-q)^3 \\ &+6p^2(1-p)^5q^2(1-q)^5 \\ &+4p(1-p)^3q^3(1-q)^6+q^4(1-q)^6 \end{aligned}$$

p/q的范围设置为(1:1 000). 仿真结果如图4所示. 从图中可以看出,当p/q较小时,DHR-MCU的状态的概率没有明显差别. 但是,随着p/q的逐渐增大,S_4逐渐趋于1,其他状态逐渐趋于0. 即当清洗恢复成功的概率(p)远大于攻击成功的概率(q)时,DHR-MCU在稳态下最终进入状态S_4,也就是说,DHR-MCU能够从攻击状态下并自行恢复.

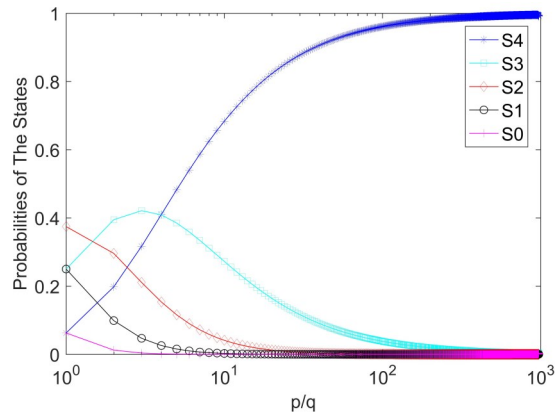


图4 DHR-MCU稳态状态迁移仿真图

3 调度裁决器设计

3.1 调度裁决器硬件设计

在DHR-MCU架构中,调度裁决器的功能涵盖了DHR典型结构中的策略分发、策略裁决以及策略调

度等核心功能,是整个体系结构的核心控制环节.作为DHR-MCU系统与外部唯一的数据传输接口,如果攻击者绕过DHR架构防御的异构MCU,直接针对调度裁决器发起攻击,当攻击成功后,便可获取DHR-MCU的完全控制权,因此调度裁决器的安全性是设计的首要原则;此外,相较于常规的MCU,除了冗余的异构MCU,调度裁决器是DHR-MCU架构中额外增加的部件,考虑MCU控制场景的实时性和多样性要求,调度裁决器在完成DHR-MCU相关的输入分发、输出裁决和策略调度、同步处理等功能时,在不影响运行效率的同时,还应具备一定的灵活性.

根据上述设计需求,本文设计了一种适用于DHR-MCU架构的软硬件协同的调度裁决器,整体功能结构示意图如图5所示,主要由上行通道、下行通道、调度控制等部分组成.

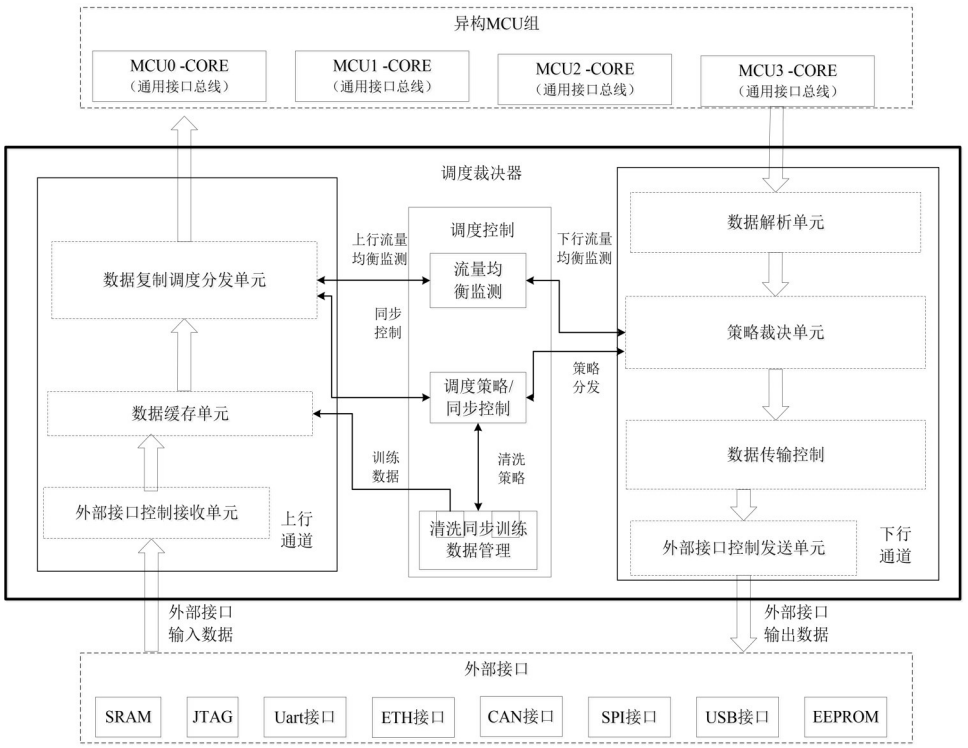


图5 DHR-MCU调度裁决器功能结构图

上行通道和下行通各单元由硬件逻辑设计实现.通过外部接口上送的用户数据在上行通道中仅进行简单的数据包头处理和复制分发,不对数据载荷进行深层解析;异构MCU输出的响应数据也仅通过描述符队列进行一致性比对,根据比对结果进行完整数据包的选择输出. 一方面,采用硬件逻辑实现大大减少了数据处理的延迟,另一方面,通过对业务数据的"透明传输",规避可能的攻击威胁. 调度控制模块由内嵌的处理器单元通过软件实现,完成流量均衡监测、裁决调度策略、进程同步控制以及清洗恢复管理等较为复杂的功能. 策略控制与上下行通道仅传输相关控制指令,而不进行任何上送和下发的业务数据处理,业务数据中可能隐含的攻击指令无法到达该模块,可以认为"攻击不可达". 因此,本文提出的调度裁决器采用软硬件协同方式,通过"透明传输"与"攻击不可达"机制,满足调度裁决器了安全、高效和灵活的要求.

3.2 运行状态控制设计

调度裁决器除了分发裁决功能外,另一个重要功能就是对于异构MCU的运行状态调度管理,DHR-MCU的运行状态分为A级运行/恢复、B级运行/恢复、C级运行/恢复六种,运行状态定义及切换方式如图6所示.

4 异构执行体程序状态同步算法

4.1 算法描述

MCU器件需要运行多种传输协议及控制程序,因此对于DHR-MCU的多个异构MCU之间的程序运行状态需要进行同步处理,以满足系统的裁决和调度需求.借鉴传统分布式共识算法思想[19,20],本文提出了一种改进的固定领导者分布式共识算法(Fixed Leader Distributed Consensus Algorithm,FLDCA). 与其他的强领导者分布式共识算法不同的是,FLDCA算法的领导者不是通过选举产生的,而是事先固定好的. 运行在调

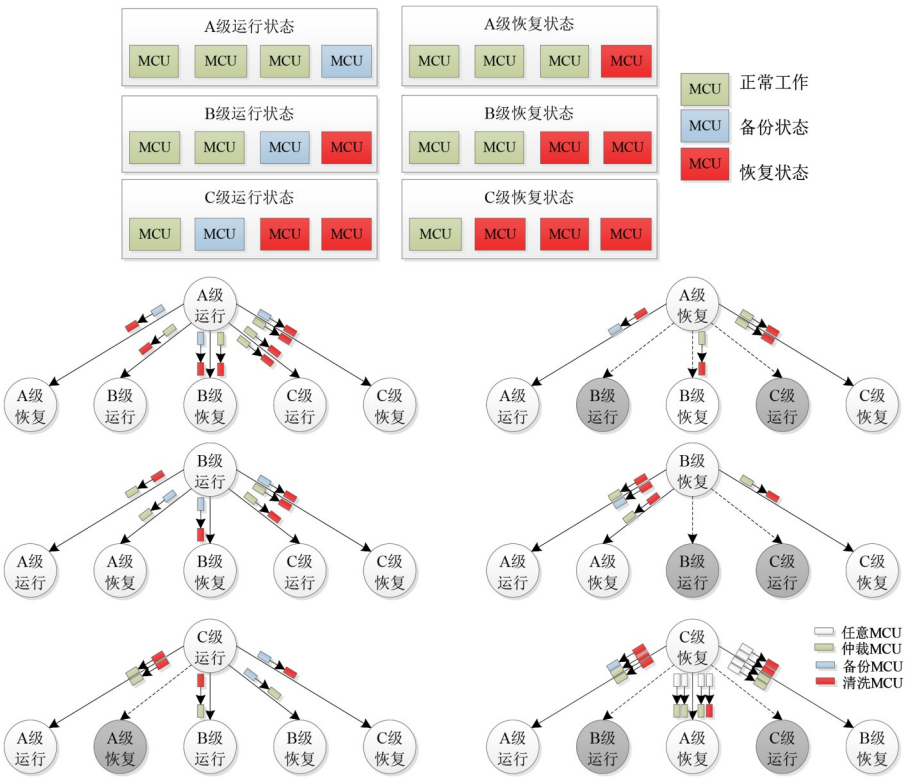


图6 异构MCU运行状态定义及切换

度裁决器的同步处理模块属于固定领导者，主要用于不同异构MCU之间进程与状态同步，不参与最终结果的输出和比对．采用FLDCA算法既保证了不同异构MCU的相互独立以避免可能存在的关联攻击，又可以在一个固定领导者的进程下实现同步和共识．

FLDCA算法的核心思想是执行基于相对时间的异构MCU程序状态同步[21]，即通过控制各异构MCU读事件和定时器事件顺序来保证异构MCU的运行状态，方便裁决器器进行有效判决并保证整体系统正常运行．同时，FLDCA算法设计充分考虑了异构MCU事件执行顺序的一致性、同步申请量、工程应用环境的丢包等问题．

为便于对算法功能进行描述，定义一组符号如表1所示，并据此定义进行形式化描述．

4.2 参数设定

4.2.1 定时器事件时间基准

为实现基于相对时间的异构MCU同步方法，FLDCA定时器事件添加和超时基于的是从同步模块申请的时间：

$$\mathrm{RT}=\mathrm{RT}_{\mathrm{end}}-\mathrm{RT}_{\mathrm{start}} \tag{4}$$

式(4)中，异构MCU定时器事件添加的时间表示为RT，超时时间表示为$\mathrm{RT}+P_i$．MCU每次判决线程是否超时不依据本地时间，而是依据同步模块反馈的相对时间RT_{r}．如果定时器事件执行顺序一致，在外部读事件的统一管控下即可实现异构MCU中定时器事件、读事件的一致性执行，进而保证程序运行状态的一致．

4.2.2 事件顺序控制

进程状态之间的变迁由读事件、定时器事件引起，对于这两种事件执行顺序的准确控制是保证异构MCU状态机一致的前提．因此，设定FLDCA中异构MCU程序同步方法为在每一次IO复用检测函数触发执行时向同步模块申请定时器事件相对时间及读事件同步．

事件的添加顺序：

(1)将$\mathrm{RT}+P_i<\mathrm{RT}_{\mathrm{r}}$定时器加入准备队列$Q_{\mathrm{ready}}$；

(2)判断同步模块读时间响应结果RD_{r}是否有效来保证异构MCU均收到读事件；

$$\mathrm{RD}_{\mathrm{r}}=\sum_{i=1}^{4}\mathrm{RD}_i \tag{5}$$

(3)读事件加入Q_{ready}，保证了事件执行顺序一致性，从而保证程序运行状态一致．

4.2.3 申请方式

申请方式分为阻塞申请和非阻塞申请两种．对于阻塞申请，程序接收不到同步响应信息会一直阻塞不执行；而在非阻塞申请下，即使程序没有立刻接到响应信息也会继续执行．而在实际应用中，丢包是无法避免，因此采用阻塞申请方式会因同步信息丢包而导致程序阻塞等待无法继续执行，从而导致程序不能正常运行．

表1 算法中的符号定义

符号	含义
t_{ij}	异构MCUi的第j个定时器事件添加时间
t'_{ij}	异构MCUi的第j个定时器事件超时时间
RT	相对时间
RD_i	异构MCUi的读事件指示
EN	异构MCU个数
TV_i	定时器事件i
RD_r	同步模块的读时间响应结果
$NODE_{sn}$	同步次数为sn的同步信息节点
SN_s	同步次数
SN_r	有效执行体申请信息中的共有的SN的最大值
RT_{start}	同步模块中RT初始时间
RT_{end}	同步模块同步信息发送事件
R_{start}	同步模块中SN_s节点添加初始时间
R_{now}	同步模块当前时间
$INDEX_i$	$NOED_{sn}$中执行体的置位标识
$INDEX_{set}$	$NOED_{sn}$中执行体的置位个数
RT_r	同步模块返回的相对时间
T_i	定时器事件i的到期时间
P_i	定时器事件i周期
EV	$EV=\{ev_1,ev_2,\cdots,ev_m\}$,事件集合
T	$T=\{T_1,T_2,\cdots,T_n\}$,定时器事件到期时间集合
$T_{current}$	当前时间
T_{wait}	等待时间
Q_{ready}	ready队列
Q_{rest}	rest队列
M_s	同步消息
M_r	同步响应消息

因此,FLDCA中异构MCU同步方法采用非阻塞同步方式. 在每一次IO复用检测函数触发执行时,向同步模块发送同步信息中携带同步标识SN_s. 同步模块收到同步信息后,查找同步响应队列中包含所有有效MCU的最大标识SN_r的响应消息发送给该MCU. 异构MCU通过对比同步标识是否与SN_s相等来确定后续执行过程,避免由于丢包导致的程序不能正常运行.

4.2.4 同步信息等待时间阈值

在异构MCU程序中,每一次IO复用检测函数触发执行时均会向同步模块申请时间. 若某个异构MCU程序异常而申请中断时引起同步模块无限等待,则所有MCU程序均会等待. 因此,同步模块需要设定同步信息等待时间的阈值来避免此问题. 在FLDCA,该阈值由异构MCU程序运行时每秒钟IO复用检测函数触发执行的次数统计确定.

因此假设MCU程序根据交互量的不同,执行次数范围为10~300,设定1 s为同步信息等待时间的阈值. 同步模块在接收到同步标识为SN_s的同步信息后记录时间RT_{start},若在事件阈值以内未收到某异构MCU的同步申请(即$NODE_{sn}$中某些执行体的$INDEX_i=0$),则证明该异构MCU出现异常. 此时,将该MCU同步信息从同步队列中删除,转向处理其他MCU的同步信息并对异常MCU执行清洗.

4.3 算法流程

4.3.1 异构MCU向同步模块发送同步申请信息

首先异构MCU判断是否产生读事件. 若产生,则将读标识RD_s置位,发送包含读标识RD_s、申请次数SN_s的同步申请信息给同步模块. 同时,以定时器事件最小超时时间T为IO复用检测函数的等待时间.

4.3.2 同步模块返回同步申请响应

同步模块接收异构MCU同步申请信息后,若不存在$NODE_{sn}$节点则添加,并记录创建时间R_{start};若存在则将相应执行体标识$INDEX_i=0$置位,并判断$NODE_{sn}$的存在时间$R_{now}-R_{start}$是否大于T,若是则根据$NODE_{sn}$计算响应参数并发送M_r,清洗$NODE_{sn}$中$INDEX_i=0$的异构MCU.

$$SN_r=SN_s \tag{6}$$

$$RD_r=\sum_{i=1}^{4}RD_i \tag{7}$$

$$RT_r=R_{now}-RT_{start} \tag{8}$$

$$EN_r=EN-INDEX_{set} \tag{9}$$

4.3.3 异构MCU未收到同步响应信息的处理

异构MCU若在等待时间内未收到同步模块发送的同步响应信息,则以相同的申请次数SN_s主动发送同步申请信息给同步模块,以定时器事件最小超时时间T为IO复用检测函数的等待时间

4.3.4 异构执行体收到同步响应信息的处理

若异构MCU在等待时间内收到同步响应信息,则先判断SN_r与SN_s是否相等:

(1)若$SN_r=SN_s$,则根据收到的RT_r进行计算,将定时器事件的超时事件加入ready队列Q_{ready}中;同时判断RD_r与EN_r是否相等,若相等则将读事件加入Q_{ready}中,程序处理ready队列;

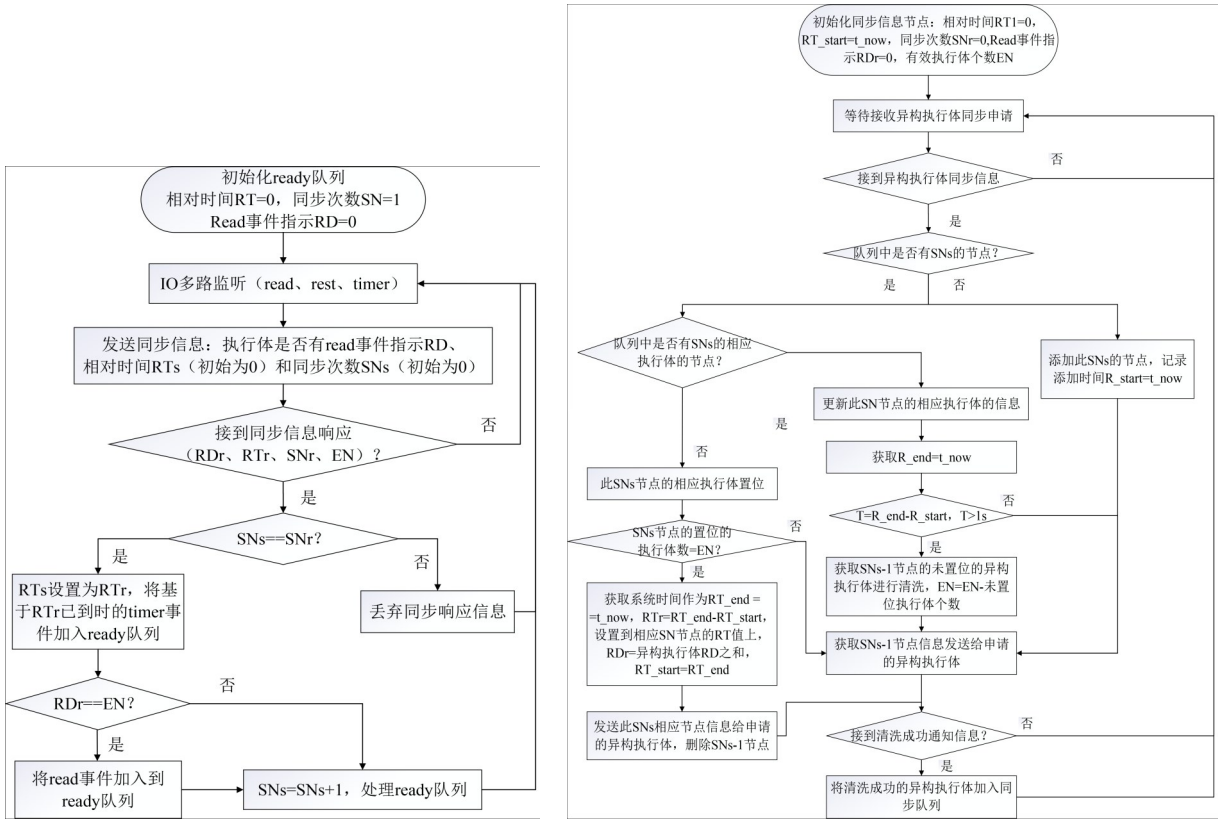
(2)若$SN_r\neq SN_s$,将同步响应消息丢弃.

FLDCA算法的异构MCU端和同步模块的流程图如图7所示.

5 DHR-MCU原型系统及测试

5.1 DHR-MCU原型系统

图8(a)所示为DHR-MCU原型系统实物,由母板和DHR-MCU核心板组成. 母板主要板载电源和外设接口,物理上采用核心板与母板扣板的形式. DHR-MCU



(*a*) 异构MCU端处理流程　　(*b*) 同步模块端处理流程

图7　FLDCA算法处理流图

核心板是整个原型系统的主要部件，主要由4种异构MCU和调度裁决器组成，每个异构MCU和调度裁决器通过2路UART（数据传输）和1路GPIO（清洗恢复命令）联接．DHR-MCU系统对外还包括RS232、RS485、Etherent、USB、WIFI和IIC等多种接口．4种异构MCU分别选用基于ARM、MIPS、AVR和ColdfireV2架构的MCU，以保证异构性；调度裁决器由内置嵌入式处理器核的FPGA构成，嵌入式处理器核运行FLDCA算法及状态控制程序，调度裁决器其余功能由逻辑实现．

为全面对DHR-MCU原型系统进行测试，构建了如图8(*b*)所示的测试平台，由DHR-MCU原型系统及多台测试电脑终端组成．

5.2　DHR-MCU常规功能性能测试

5.2.1　功能测试

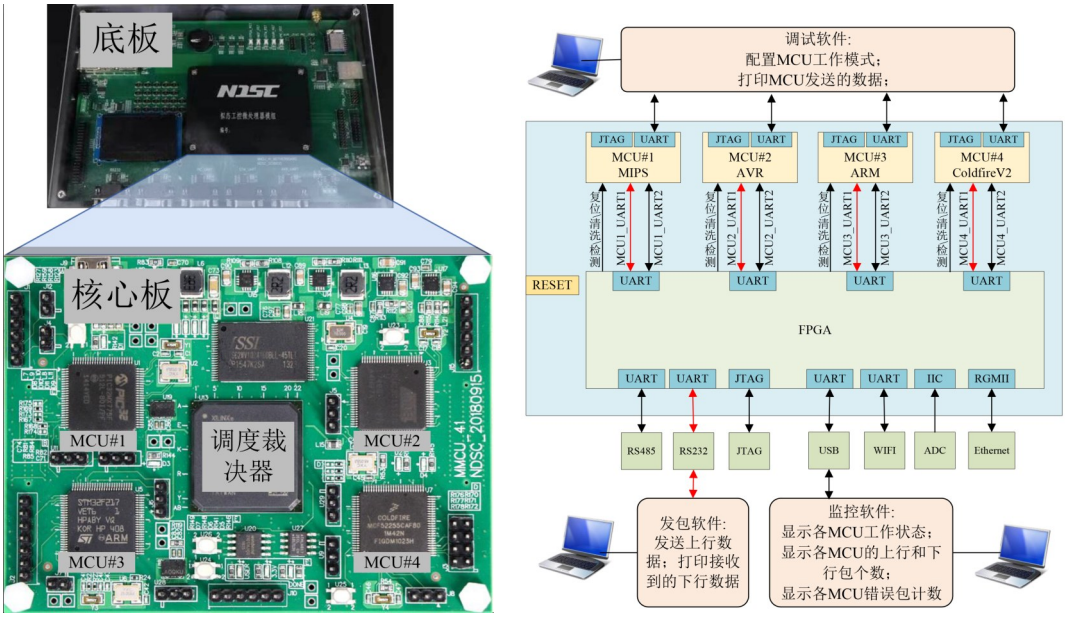
功能测试方面主要分为2类，一类是基础功能接口测试，另一类是调度机理测试．在调度机理测试中，使用“白盒插桩”的方式模拟各个通路的异常情况从而触发调度机制．“白盒插桩”的思想来源于ISO 26262流程模型右侧测试环节的故障注入测试（Fault Injection Test，FIT），错误输出不是由于实际的扰动，而是通过配置软件来人为产生的[22]．采用“白盒插桩”的方式，一方面降低了测试过程中产生错误结果的难度，另一方面更为贴切地模拟了未知攻击或未知故障造成的未知扰动，更加符合DHR结构的防御初衷．功能测试结果如表2所示．从测试结果可知，MCU各项基本功能均可通过测试用例，DHR-MCU在异常发生时可顺利地恢复到正常的通信状态．

5.2.2　性能测试

调度器的引入是否会对整个DHR-MCU的控制反馈速度造成明显延迟，是衡量DHR-MCU工作性能的核心指标．因此，在性能测试方面，本文仅考虑时延方面的性能对比．测试方法为通过设置计数器，在测试终端发送数据的同时开始计数，收到返回数据停止计数，通过计数时钟频率即可换算出数据收发所用的时间．为便于对比，可先依次测定四个异构MCU的延迟然后再测定整个DHR-MCU的延迟．通过RS232串口向FPGA发送数据，并在FPGA中设置抓取计数器信号，通过多次测试，获取平均值．测试结果如图9所示，与具有最高延迟的MCU3相比，DHR-MCU的延迟仅提高了约1%，完全可以满足正常工作需求．

5.3　DHR-MCU抗攻击测试

对于DHR-MCU，外部攻击所造成的影响将直接反



(*a*) DHR-MCU 原型系统　　(*b*) DHR-MCU 测试平台

图 8　四余度 DHR-MCU 原型系统与测试平台

表 2　常规功能测试

测试项	测试例	测试结果
基础接口功能测试	异构 MCU RS232 上/下行通路测试	通过
	异构 MCU RS485 上/下行通路测试	通过
	异构 MCU Ethernet 上/下行通路测试	通过
	异构 MCU WIFI 上/下行通路测试	通过
	异构 MCU IIC 访问上/下行通路测试	通过
调度机理测试	调度裁决器上行分发功能测试	通过
	上/下行通路添加接口标签功能测试	通过
	异构 MCU 清洗功能测试	通过
	主通道自动/手动切换测试	通过
	主/从/备通道 MCU 替换机制测试	通过
	主/从/备通道异常反向验证机制测试	通过

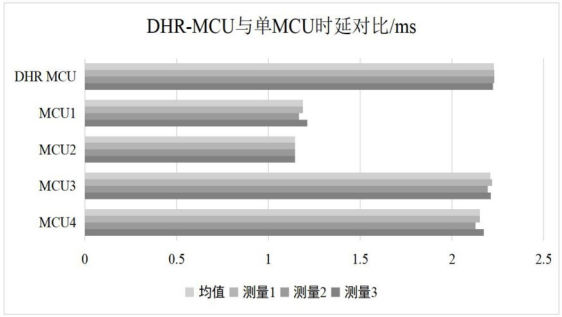
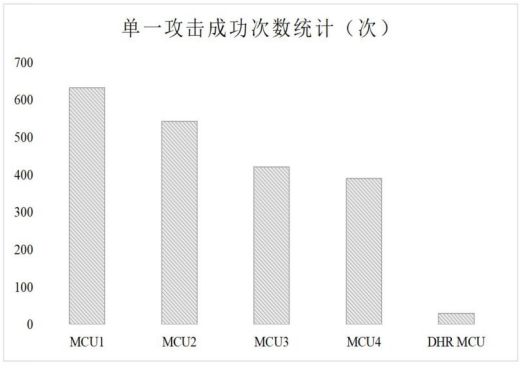


图 9　DHR-MCU 数据收发延迟对比

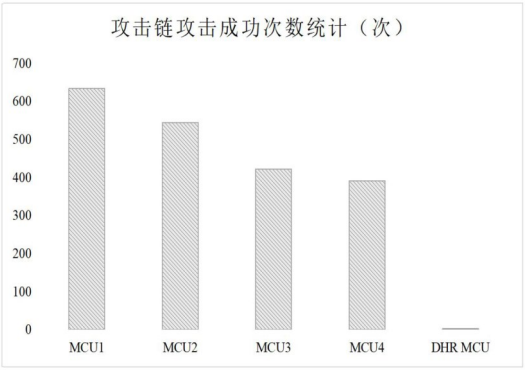
应在接口输出的数据中. 当单个 MCU 被攻击输出异常时,称为系统受到差模攻击;当 2 个或 3 个 MCU 被攻击输出异常时,称之为多模或 $N-1$ 模攻击;当所有 MCU 被攻击输出都是异常时,称之为共模攻击. 本文所提 DHR-MCU 使用的是大数表决算法进行判决,当发现某异构 MCU 输出异常超过阈值时进行切换清洗;同时在 DHR-MCU 工作正常时,采用动态调度的策略进行定时切换以排除潜伏式植入攻击. 参考文献[23]设定各类攻击为 1 000∶200∶40∶2,本文同样按照该比例进行白盒插桩测试,基于参考文献[24]设定异构执行体的调度策略参数,重复测试场景 1 000 次.

图 10(*a*)是一次攻击即可成功的单个 MCU 以及 DHR-MCU 系统被攻击成功的测试结果,可以看出,本文所提架构可显著降低攻击成功次数. 此外,由于大多数攻击都是以攻击链的形式进行的,即攻击者需要发动诸如系统探测、漏洞发现、系统突破以及系统控制等攻击阶段,因此需要多阶段的数据交互. 在整个攻击过程中,一次成功的攻击,在攻击链情况下是无效的. 使用与一次攻击相同的场景,在攻击链场景下的测试结果如图 10(*b*)所示. 单个 MCU 被攻击成功的次数与在一次攻击场景下基本一致,动态异构冗余 MCU 被成功攻击的次数锐减,几乎为 0,主要原因是动态异构冗余 MCU 具有裁决功能和异常感知功能,可以动态的进行清洗恢复以切断攻击行为.

需要说明的是,本文使用的白盒插桩测试方法将攻击场景理想化,即直接反映到各 MCU 的输出上来. 相比之下,实际网络攻击要比本实验中使用的理想化攻击场景更加困难. 因此,本文所提的 DHR-MCU 架构在实际的抗攻击性方面将会有更大的提升.



(*a*) 单一攻击成功数对比



(*b*) 攻击链攻击成功数对比

图10 DHR-MCU白盒插桩攻击结果

6 结束语

本文针对物联网环境下，现有同构多核MCU无法应对网络攻击带来的安全问题，构建了基于动态异构冗余结构的微控制器DHR-MCU架构，在利用传统的冗余设计增强功能安全的同时，通过引入“动态”和“异构”的设计理念，使DHR-MCU具备抵御外来攻击的信息安全特性. 提出的带有热备份的四冗余度结构可以很好地解决稳定性问题，设计的软硬协同的调度裁决器在保证自身安全性的同时确保了灵活和高效，提出了解决异构MCU状态一致性的FLDCA算法. 通过理论分析和原型系统实测，证明DHR-MCU架构在保证功能和性能正常的前提下，具备良好的抗攻击及自我恢复能力.

从工程实用角度，本文构建的DHR-MCU架构原型存在功耗大、成本高的问题. 后续可采用系统级封装或者堆叠封装方式将整个DHR-MCU封装成单一的MCU芯片，或者基于本文提出的原型系统进行芯片架构设计，并针对数据传输机制、异构MCU核配置、调度裁决器、外围接口进行优化，设计实现具备完整DHR结构的高安全MCU芯片，在提供高可用、高可靠、高安全的同时有效控制成本和功耗.

参考文献

[1] 左亚旻, 顾卫. 单片机实用技术[M]. 北京: 北京理工大学大学出版社, 2021.
ZHUO Y W, GU W. MCU Practical Technology[M]. Beijing: Beijing Institute of Technology University Press, 2021. (in Chinese)

[2] SALE20. 物联网时代下，MCU应用的新生态[EB/OL]. (2022-06-14) [2022-11-09]. https://www. elecfans. com/d/1848515.html.

[3] 汽车安全研究中兴. MCU如何实现功能安全[EB/OL]. (2022-04-09) [2022-11-09]. https://www. sohu. com/a/354116152_560178.

[4] Infineon. AURIX Security Solutions [EB/OL]. (2022-10-01) [2022-11-09]. https://www. infineon. com/cms/cn/product/microcontroller.

[5] 芯驰科技. 芯驰科技高性能高可靠车规MCU E3控之芯[EB/OL]. (2022-07-05)[2022-11-09]. https://www.elecfans.com/d/1857374.html.

[6] 邬江兴. 网络空间拟态防御导论-上册[M]. 北京: 科学出版社, 2017.
WU J X. Introduction to Cyberspace Mimic Defense[M]. Beijing: Science Press, 2017. (in Chinese)

[7] 邬江兴. 网络空间内生安全-下册: 拟态防御与广义鲁棒控制[M]. 北京: 科学出版社, 2020.
WU J X. Endogenous Security in Cyberspace-Part II: Mimicry Defense and Generalized Robust Control[M]. Beijing: Science Press, 2020. (in Chinese)

[8] 宋克, 刘勤让, 魏帅, 等. 基于拟态防御的以太网交换机内生安全体系结构[J]. 通信学报, 2020, 41(5): 18-26.
SONG K, LIU Q R, WEI S, et al. Endogenous security architecture of Ethernet switch based on mimic defense[J]. Journal on Communications, 2020, 41(5): 18-26. (in Chinese)

[9] 马海龙, 江逸茗, 白冰, 等. 路由器拟态防御能力测试与分析[J]. 信息安全学报, 2017, 2(1): 43-53.
MA H L, JIANG Y M, BAI B, et al. Tests and analyses for mimic defense ability of routers[J]. Journal of Cyber Security, 2017, 2(1): 43-53. (in Chinese)

[10] 仝青, 张铮, 张为华, 等. 拟态防御Web服务器设计与实现[J]. 软件学报, 2017, 28(4): 883-897.
TONG Q, ZHANG Z, ZHANG W H, et al. Design and implementation of mimic defense web server[J]. Journal of Software, 2017, 28(4): 883-897. (in Chinese)

[11] 王禛鹏, 扈红超, 程国振. 一种基于拟态安全防御的DNS框架设计[J]. 电子学报, 2017, 45(11): 2705-2714.
WANG Z P, HU H C, CHENG G Z. A DNS architecture

based on mimic security defense[J]. Acta Electronica Sinica, 2017, 45(11): 2705-2714. (in Chinese)

[12] 徐蜜雪, 苑超, 王永娟, 等. 拟态区块链: 区块链安全解决方案[J]. 软件学报, 2019, 30(6): 1681-1691.
XU M X, YUAN C, WANG Y J, et al. Mimic blockchain—Solution to the security of blockchain[J]. Journal of Software, 2019, 30(6): 1681-1691. (in Chinese)

[13] 刘彩霞, 季新生, 邬江兴. 一种基于MSISDN虚拟化的移动通信用户数据拟态防御机制[J]. 计算机学报, 2018, 41(2): 275-287.
LIU C X, JI X S, WU J X. A mimic defense mechanism for mobile communication user data based on MSISDN virtualization[J]. Chinese Journal of Computers, 2018, 41(2): 275-287. (in Chinese)

[14] 李传煌, 任云方, 汤中运, 等. SDN中服务部署的拟态防御方法[J]. 通信学报, 2018, 39(S2): 121-130.
LI C H, REN Y F, TANG Z Y, et al. Mimicry defense method for service deployment in SDN[J]. Journal on Communications, 2018, 39(S2): 121-130. (in Chinese)

[15] 陈福才, 周梦丽, 刘文彦, 等. 云环境下面向拟态防御的反馈控制方法[J]. 信息网络安全, 2021, 21(1): 49-56.
CHEN F C, ZHOU M L, LIU W Y, et al. Feedback control method for mimic defense in cloud environment[J]. Netinfo Security, 2021, 21(1): 49-56. (in Chinese)

[16] 朱维军, 郭渊博, 黄伯虎. 动态异构冗余结构的拟态防御自动机模型[J]. 电子学报, 2019, 47(10): 2025-2031.
ZHU W J, GUO Y B, HUANG B H. A mimic defense automaton model of dynamic heterogeneous redundancy structures[J]. Acta Electronica Sinica, 2019, 47(10): 2025-2031. (in Chinese)

[17] 邬江兴. 网络空间拟态防御原理(下册): 广义鲁棒控制与内生安全[M]. 2版. 北京: 科学出版社, 2018.
WU J X. Principles of Mimicry Defense in Cyberspace-Volume II: Generalized Robust Control and Endogenous Security[M]. 2nd ed. Beijing: Science Press, 2018. (in Chinese)

[18] 贾洪勇, 潘云飞, 刘文贺, 等. 基于高阶异构度的执行体动态调度算法[J]. 通信学报, 2022, 43(3): 233-245.
JIA H Y, PAN Y F, LIU W H, et al. Executive dynamic scheduling algorithm based on high-order heterogeneity [J]. Journal on Communications, 2022, 43(3): 233-245. (in Chinese)

[19] 易星辰, 魏恒峰, 黄宇, 等. PaxosStore中共识协议TPaxos的推导、规约与精化[J]. 软件学报, 2020, 31(8): 2336-2361.
YI X C, WEI H F, HUANG Y, et al. TPaxos consensus protocol in PaxosStore: Derivation, specification, and refinement[J]. Journal of Software, 2020, 31(8): 2336-2361. (in Chinese)

[20] 王日宏, 周航, 徐泉清, 等. 用于联盟链的非拜占庭容错共识算法[J]. 计算机科学, 2021, 48(9): 317-323.
WANG R H, ZHOU H, XU Q Q, et al. Non-byzantine fault tolerance consensus algorithm for consortium blockchain[J]. Computer Science, 2021, 48(9): 317-323. (in Chinese)

[21] 宋克, 欧阳玲, 魏帅, 等. 面向拟态架构的差分超时参数预测算法[J]. 信息工程大学学报, 2020, 104(4): 90-95.
SONG K, OUYANG L, WEI S, et al. Prediction algorithm for differential time-out parameter on mimic architecture[J]. Journal of Information Engineering University, 2020, 104(4): 90-95.

[22] 邓鹏杰, 刘强. 典型故障注入攻击分析方法性能评估[J]. 南开大学学报(自然科学版), 2020, 53(6): 13-17.
DENG P J, LIU Q. Performance evaluation of typical fault analysis methods[J]. Acta Scientiarum Naturalium Universitatis Nankaiensis, 2020, 53(6): 13-17. (in Chinese)

[23] ZHANG W J, WEI S, TIAN L, et al. Scheduling algorithm based on heterogeneity and confidence for mimic defense[J]. Journal of Web Engineering, 2020, 19(7-8): 971-998.

[24] ZHANG W J, ZHU Z B, SONG K, et al. A quantification method for the heterogeneity of mimic control plane in SDN[J]. Electronics, 2022, 11(23): 3864.

作者简介

欧阳玲 女,1978年出生,安徽池州人. 信息工程大学在读博士生、中原工学院副教授. 主要研究方向为自动控制理论、工业控制系统与安全.
E-mail: 3937@zut.edu.cn

宋 克 男,1976年出生,河南许昌人. 信息工程大学副研究员、博士生. 主要研究方向为网络空间安全、集成电路设计技术.
E-mail: skoyl@163.com