doi:10.11959/j.issn.1000-436x.2016275

密码芯片的多算法随机作业流调度方法

李莉[1,2]，史国振[3]，耿魁[4]，董秀则[2]，王璇[1]，李凤华[4]

（1. 西安电子科技大学通信工程学院，陕西 西安 710071；
2. 北京电子科技学院电子信息工程系，北京 100070；
3. 北京电子科技学院信息安全系，北京 100070；
4. 中国科学院信息工程研究所信息安全国家重点实验室，北京 100093）

摘 要：针对安全领域中海量业务安全需求多样性导致的多种密码算法运算随机交叉的现象，提出了具有关联判断控制的基于业务标识的分层硬件调度方法(HHS-ACDID)。第一级调度完成业务在不同算法簇上的分配，通过优化检索逻辑，实现数据的快速分配；第二级调度通过增设关联控制模块和关联队列的方式，完成上下文相关作业分组调度顺序的处理。采用中间状态存储模块，以业务号为索引完成串行密码算法工作模式下中间状态的存储，并通过预处理模块完成对后序关联作业分组输入数据的处理。实验验证所提调度方法有效解决了高速数据流下多对多通信中多密码算法、多数据流的随机交叉加解密问题。

关键词：交叉加解密；作业调度；多算法；密码处理芯片；硬件调度

中图分类号：TP393.2 **文献标识码**：A

Stochastic job stream scheduling method for cipher chip with multi-cryptography

LI Li[1, 2], SHI Guo-zhen[3], GENG Kui[4], DONG Xiu-ze[2], WANG Xuan[1], LI Feng-hua[4]

(1. College of Communication Engineering, Xidian University, Xi'an 710071, China;
2.Department of Electronic and Information Engineering, Beijing Electronics Science and Technology Institute, Beijing 100070,China;
3. Department of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China;
4.State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing 100093, China)

Abstract: Aiming at the rich of safety requirements of tasks which resulting in random cross access to multi cipher algorithms, a hierarchical hardware scheduling method was presented with associated control based on data identification. The first level was responsible for distributing tasks to different cipher clusters, and by optimizing the search logic to achieve rapid distribution of data. The second level was responsible for completing the context-related tasks in scheduling order by adding an association control module and association queues. Intermediate state storage module realized the saving of the intermediate state in serial cipher algorithm modes, which was indexed by task ID. Pre-processing module process data inputted by the succeeding tasks. It is proved that the proposed scheduling algorithm solves the problem of random cross encryption and decryption in many-to-many communication model of high-speed data stream.

Key words: cross encryption and decryption, job stream scheduling, multi-cryptography, cipher chip, hardware scheduling

---

收稿日期：2016-08-29；**修回日期**：2016-10-25
通信作者：李凤华，lfh@iie.ac.cn
基金项目：国家重点研发计划基金资助项目（No.2016YFB0800304）；北京市自然科学基金资助项目（No.4152048）；新闻出版重大科技工程基金资助项目（No.1681300000119）

Foundation Items: The National Key Research and Development Project (No.2016YFB0800304), The Natural Science Foundation of Beijing (No.4152048), The Major Science and Technology Project of Press and Publication (No.1681300000119)

1　引言

计算机网络的发展以及互联网应用、移动通信的普及，加快了全球经济一体化的进程，促使社会经济及网络经济快速发展。云服务、电子商务、网上银行、电子支付等新型服务模式的不断涌现，使云安全进入了高速发展期，云加密服务成为云安全市场新的增长热点。在数据大集中处理方式下，数据加解密以及数字签名、验签的业务请求数量非常巨大，加密服务终端必然存在着不同业务数据的高度融合和密码服务请求的交叉访问，即多个不同业务的不同密码服务请求间的交叉融合。这些业务请求的不同可能表现在处理算法上，如加密邮件请求的 DES 算法、签名请求的散列算法，以及数据库加密请求的 AES 算法等；也可能表现在密钥上，不同客户的业务请求具有不同的公、私钥；或表现在不同的密码算法工作模式上，如对于安全性要求不高或强调处理速度的 ECB、CTR 工作模式，或对安全性要求较高的 CBC、OFB、CFB 模式等。如何对这些随机交叉的安全服务请求进行正确且快速的处理，是海量数据安全服务请求必须要解决的问题。

目前，对于多算法交叉加解密的调度研究，多是在软件或算法层面解决，通过软件层面的调度实现业务数据在多算法核上的分配，且以非关联数据的处理为主。软件实现的调度方法具有处理灵活的优点，但是速度提升有限，不能很好地满足海量数据的快速、有效处理，影响用户体验。本文以云安全应用下高性能密码处理芯片的设计为研究背景，探讨了在多算法交叉加解密服务需求下，采用带有中间状态管理的基于数据标识的分层硬件调度方法（HHS-ACDID）实现交叉业务请求下多算法、多密钥、多 IP 核的随机交叉加解密服务，实现数据大集中处理场景下业务的快速处理。

2　相关研究

目前，对于海量数据密码应用的研究和处理主要集中在多核并行处理架构和密码算法本身的高速并行实现上。并行处理架构又分为以 GPU 为主的通用多核 CPU 架构[1, 2]和以密码算法处理 IP 核为主的专用多核架构。算法本身的并行高速实现主要集中在挖掘算法本身的并行性和算法的流水线实现 2 个方面，将算法的处理分为多个阶段[3]，部分功能采用并行化的实现方式[4, 5]，如模幂运算[6]或点乘算法[7]的并行化，以及对密钥扩展模块采用并行化的实现方式[8]。文献[9]建立了以 GPU 为中心的流水线架构模型，通过对 GPU 核的周期性调度，获得可预期的 GPU 处理时间，但是并未考虑对交叉作业分组的处理，每个 GPU 核流水线处理的是同一业务的顺序作业分组，其中，GPU 的调度周期受限于最大作业分组的处理时间，且没有涉及异构核和关联任务处理的问题。文献[10]基于数据密态检索采用的 Gentry 全同态加密，提出建立数据依赖图解决上下文相关业务并行处理的方法。文献[11]探讨了一种虚拟化的软件架构，通过 Dispatcher 将数据送至不同的处理器实现对全同态加密算法的并行化处理。这些处理都是软件层面的解决方案，调度的处理相对滞后，不能根据处理模块的运行情况进行业务调度的实时处理。文献[12]针对业务数据交叉加解密的思想，提出了在典型密码处理器体系结构上增设专用易失性存储器、专用易失性存储器控制器和索引寄存器的多线程密码芯片的体系结构，但并未对其进行实现与验证。以密码算法处理 IP 核为主的专用多核架构采用一个系统内部挂接多个密码算法处理 IP 核[13]，通过指令级和数据级并行，最大化数据叠加流水处理，提高系统的吞吐率，加速算法数据处理速度，实现单一链接下的不同任务在不同工作模式下的多种密码算法的并行处理[14~17]。这些研究成果虽然涉及了多密钥多数据加解密以及单一密钥多数据流加解密操作的技术，但是均未涉及不同业务作业流间存在随机交叉加解密请求状况的处理，且未考虑依赖数据的处理。

许多研究者在操作系统层面对异构多核处理器下的任务调度算法进行了研究[18,19]，文献[20]通过对嵌入式流媒体中数据相关任务建模，采用无环静态数据流（CSDF）模型，实现了周期性任务的硬件调度。还有一些研究采用多任务调度算法实现任务在 FPGA 平台上的分配，通过 FPGA 的可重构特性，实现对 FPGA 资源的有效利用，从而达到计算灵活性和高速性的目的[21]。文献[22]针对重构时出现的开销问题，提出混合映射的调度技术，利用应用程序之间的关系来减少重构产生的开销。由于在多处理器上进行任务的分配是一个典型的 NP 问题，无法在多项式时间内找到一个最优的任务分配方案，因此，一些研究者采用启发式算法或全局优化技术，来寻找近似最优解。文献[23]提出了一种可抢占式调度的硬件实现方法，通过扫描路径上的

寄存器结构，来暂停低优先级任务，启动高优先级任务，设计实现了不影响其他任务执行的低优先级任务上下文保存和恢复电路。这些调度机制都是停留在理论仿真阶段，并未提出与硬件架构相配合的调度算法。硬件调度算法不仅具有运算速度快的特点，而且可以大大降低软件调度算法设计的复杂度。

因此，根据高并发数据的处理特点，本文提出了基于数据标识的分层硬件调度机制，通过构造具有特定标识的作业分组，采用专门的调度解析模块进行作业分组的转发，实现多个算法模块间的并行运算，完成高并发多密钥随机交叉的密码运算操作，提高了加解密效率。

3 问题描述

本文所研究的密码系统包含有多种密码算法，这里将不同的算法按簇进行划分，每种算法对应一个或多个能并行处理的 IP 核，称为一个算法簇。若系统可以实现 p 种算法处理，ip_{ik} 表示实现算法 $s_j(1 \leqslant j \leqslant p)$ 的第 k 个处理单元，m_j 表示算法簇 s_j 中的 IP 核数，即

$$P = \{ip_{11}, ip_{12}, \cdots, ip_{1m_1}, \cdots, ip_{p1}, \cdots, ip_{pm_p}\}$$

$$m_1 + m_2 + \cdots + m_p = n$$

不同的算法核对分组数据的处理速度可能不同。业务以作业分组的形式进入此多核密码处理系统，密码处理系统需要按照不同业务的不同处理需求将作业分组快速地分配至能进行对应密码运算的处理节点上。由于来自不同应用请求的数据传输速度不同，因此，进入密码处理系统的作业分组的先后顺序不同，存在不同业务不同数据分组间的交叉现象。如图 1 所示，P_{ia} 表示业务 i 的第 a 个作业分组，同一业务的 2 个连续作业分组 P_{ia}、$P_{i(a+1)}$ 间可能穿插有其他业务的作业分组。因此，在进行业务处理时，需要解决 3 个方面的问题。

1) 作业分组 P_{ia} 的处理需求是什么，即要解决作业分组与处理节点间的映射关系。

2) 业务中间状态的管理。这里称同一业务最后一个作业分组之前的运算结果为中间状态。在关联业务的后序作业分组 $P_{i(a+1)}$ 到来之前，P_{ia} 的运算结果 C_{ia} 作为后续作业分组的中间状态如何保存以及在后序作业分组 $P_{i(a+1)}$ 到来之后，如何获取中间状态值 C_{ia}。

3) 若同一业务的作业分组间存在迭代关系，在前序作业分组未处理完之前，为保证系统处理的速度和系统入口作业流的不间断，流水到来的后序作业分组如何处理。

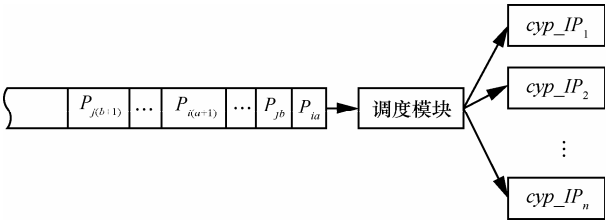

图 1 不同业务流对多种密码算法模块的随机交叉访问

因此，为便于系统的正确处理，需要对进入系统的作业分组进行归一化处理，使作业分组带有算法、运算类型、算法工作模式等数据标识，此处将作业分组设定为

$$P_{ia} = \{task_i, cmd_i, cyp_i, mode_i, No., long, d_{ia}\}$$

其中，$task_i$ 表示业务号，用来标识数据的来源，对数据的处理没有影响，但是决定着数据的正确返回；cmd_i 为处理命令，决定对数据的具体操作，如加密、解密、签名、验签等；cyp_i 为密码算法模块 ID 号，决定数据的处理节点；$mode_i$ 为密码算法工作模式，决定对密码算法处理节点入口数据的处理。对于 ECB 模式，可以将数据直接送入算法处理节点，而对于 CBC 模式，则需要将数据与同一个业务的前一组数据的运算结果进行异或后，再送入算法处理节点。$No.$ 为作业分组的序号，$long$ 为作业分组中运算数据的长度；d_{ia} 为运算数据。

4 分层调度机制

上述的 3 个问题，实际上都可以归结为调度问题，即作业分组在算法 IP 核的分配、分配的时机以及分配时数据分组的预处理。为了保证对业务流的快速处理，系统设计采用两级调度机制，第一级调度完成作业分组与算法模块处理节点间的映射，第二级调度根据作业分组间的依赖关系决定作业分组的调度顺序，并获取正确的密码算法模块入口数据。

调度模型如图 2 所示，其中，schedule1 为第一级调度模块，根据 cyp_i 完成作业分组数据送入对应的算法模块簇；schedule2 为第二级调度模块，根据 $mode_i$ 获取正确的算法入口数据，送入 IP 核入口队列。IP 为粗粒度实现的密码算法运算模块，feedback 为反馈模块。每个模块有自己的入口和出口队列，其中，pre_queue 为预处理队列，IP_queue 为 IP 核入口队列，post_queue 为处理后队列。系统总的入

口队列接收随机交叉流入的不同业务数据，入口队列由所有的 IP 核共享，出口队列的数据被分配至不同的 CPU 进程。此架构采用作业级并行 JLP（job-level parallelism）的粗粒度并行方式[8]，各模块间采用分段总线互连的方式。由于所有的数据处理模块均有自己的本地缓冲队列，可以独立运行，从而克服了存储器间的竞争，实现了同一业务多个作业分组的同时处理，避免资源约束造成的作业流处理时滞和效率问题；同时增加算法 IP 核不会对其他操作造成影响，便于进行系统的线性扩展。

在这种调度方式下，cyp_i 的设计如图 2 所示，其中，高 m 位表示簇号，低 n 位表示簇内的算法处理节点号。

4.1　第一级调度

第一级调度由 schedule1 模块完成，算法调度模块 schedule1 提取 input_queue 中的数据，通过对作业分组 cyp_i 的解析，获取算法类型，将数据分发至相应算法的预处理队列实现数据的分流，保证作业分组与算法间映射的正确性。若作业分组中表示运算数据长度的标识 *long* 以字节为单位，系统中的分段总线宽度为 w bit，则除分组头外，运算数据的读取至少需要 $\frac{long\times 8}{w}$ 个时钟。调度算法如下。

1) 若 input_queue 队列非空，则从 input_queue 中读取作业分组头数据。

2) 对分组头数据进行解析，获取处理节点簇号 *cluster_id* 和作业分组长度数据。

3) 根据 *cluster_id* 检索预处理队列索引表，并将分组头数据送对应的预处理队列；否则，*cluster_id* 为非法的算法簇号，数据送 ERROR_FIFO。

4) 若长度计数器中的值与长度寄存器中的值不同，则 input_queue 中的数据送 pre_queue；否则，进入步骤 6)。

5) 长度计数器加 $\frac{w}{8}$，返回步骤 4)。

6) 返回步骤 1)。

在此调度算法中，除步骤 3)外，其余的步骤都可在单个时钟完成。而步骤 3)根据抽取的 *cluster_id* 进行数据的分转，若系统中的算法类型共有 n 种，则预处理队列索引表的长度为 n，软件遍历的方式最多需要耗费 n 个时钟，大大影响数据传输的效率。基于散列的检索方式，由于存在冲突碰撞现象，也存在效率不高的问题。这里考虑硬件解决方案，由于作业分组对密码算法的请求具有唯一性，因此，*cluster_id* 至多只能与 pre_queue 索引表中的一个表项一致，多于一个表项相同的现象是不可能存在的，可以将其作为无关项对逻辑操作进行优化。假设 *cluster_id*=x，pre_queue 索引表中共有 3 个索引项 a、b、c，“^”表示异或操作，“|”表示逻辑或操作，y 表示 x 与哪个索引项匹配，则 x 与 a、b、c 之间的关系只能有如表 1 所示的 4 种情况。

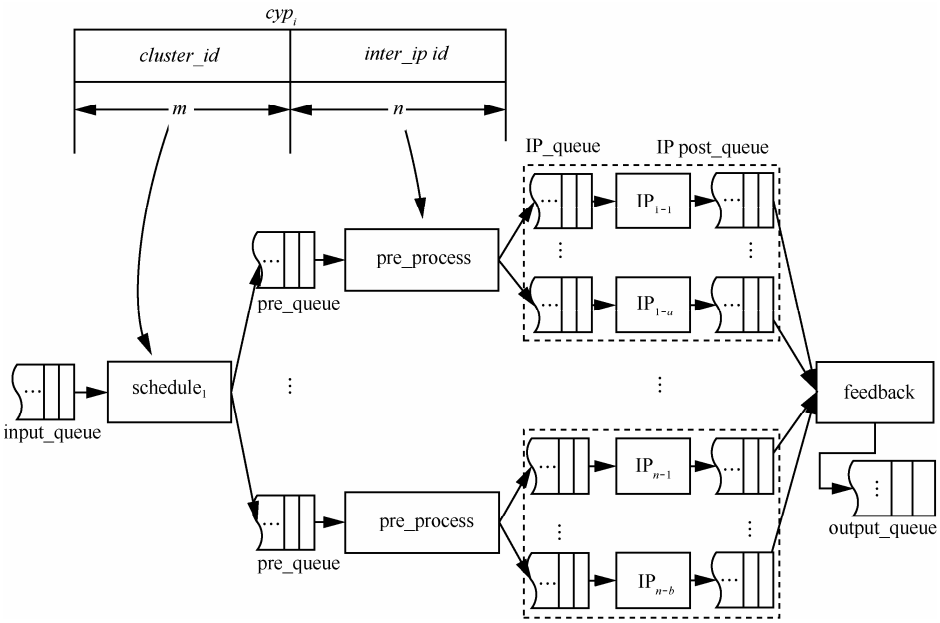

图 2　多核并行流水线密码处理调度模型

表 1　　簇索引查找真值

$\|(x\^a)$	$\|(x\^b)$	$\|(x\^c)$	y[1:0]	匹配
0	1	1	01	与 a 匹配
1	0	1	10	与 b 匹配
1	1	0	11	与 c 匹配
1	1	1	00	无匹配项

$$y(1)=(|(x\wedge b))'+(|(x\wedge c))'$$
$$y(0)=(|(x\wedge a))'+(|(x\wedge c))'$$

根据 y 值可以快速地将数据分转至不同的簇队列 pre_queue 中。分簇调度的方式，使调度时的对比位数有效减少，平均减少为原位数的$\frac{1}{2}$，若系统共实现 8 种密码算法，每种密码算法有 8 个处理节点，共 64 个处理节点，若采用一级调度的方式，则索引项的位宽为 6，而采用两级调度的方式，在处理节点的分配上，每级索引项的位宽为 3 bit，提高了以 4 输入查找表为单元的 FPGA 布局的灵活性，同时，此种硬件检索比较逻辑可以使索引表的时间复杂度降为 $O(1)$，没有碰撞冲突，且不影响系统的工作频率，很好地解决了数据分转调度效率低的问题。

4.2　第二级调度

第二级调度是保证交叉访问正确性的关键环节，由 pre_process 模块完成，通过对作业分组工作状态的判断，确定作业分组调度的时间顺序，并获得算法模块的输入数据。由于输入队列业务间的随机交叉性和 IP 核间的异构性，不能保证具有上下文相关的作业分组在进行处理时，其前序作业分组已处理完。假设 pre_queue 的作业分组序列为 $\{P_{11},P_{21},P_{31},P_{12},P_{22}\}$，如表 2 所示，IP 核内密码算法的处理采用流水线的方式。

表 2　　预处理队列作业分组序列

时间序列	作业分组	$P_{ia}=\{task_i,cyp_i,mode_i,No.\}$
1	P_{11}	{1,IP$_1$,CBC,1}
2	P_{21}	{2,IP$_2$,ECB,1}
3	P_{31}	{3,IP$_1$,ECB,1}
4	P_{12}	{1,IP$_1$,CBC,2}
5	P_{22}	{2,IP$_2$,ECB,2}

如图 3 所示，在图 3(a)中，由于不同业务的作业分组间相互独立，节点对数据的处理采用流水线的方式，P_{11}、P_{21}、P_{31} 这 3 个业务流的第一个作业分组依次从 pre_queue 中输出，进入对应的处理节点。由于 P_{11} 的工作模式为 CBC，作业分组间存在迭代关系，后序作业分组 P_{12} 的处理必须等到 P_{11} 运算结果输出的时刻才能进行，从而阻滞了 P_{22} 的读取。如果能将 P_{12} 旁路掉，在 P_{11} 运算完成后再取出，则可以在保证正确性的前提下，加速后序其他业务作业分组的处理，如图 3(b)所示。因此，本系统在架构中针对每一个算法簇在预处理模块前端增加关联控制模块和关联队列（relate_queue），如图 4 所示。

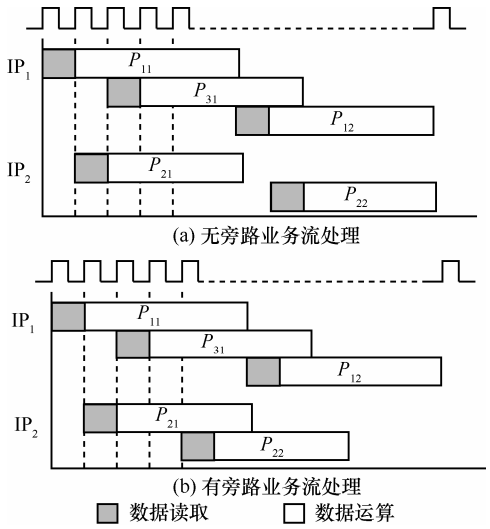

图 3　作业分组在处理节点上的分配

1) 关联控制模块

关联控制模块通过维护关联 ID 表，实现对关联作业分组调度次序的控制。关联 ID 表为正在进行处理的关联任务作业分组的索引表，这里以关联作业分组的 ID 号 $task_i$ 为索引项。

① 关联 ID 表项的追加：任何一个进入预处理模块（IP_CTRL）的关联作业分组，都要将其 ID 号存入关联 ID 表。

② 关联 ID 表项的删除：任何一个从算法运算模块输出的关联作业分组，都要通过 *req* 信号通知关联控制模块，关联控制模块通过提取此作业分组的 $task_i$，并与关联 ID 表中的 ID 号进行对比，若相同，则将此 ID 号从关联 ID 表中删除。

2) 关联队列

关联队列用于保存正在进行运算的关联作业分组的后续作业分组，保证后续作业分组的正常读取。

① 关联队列的输入：当预处理队列非空时，关联控制模块从预处理队列中提取作业分组头，首先根据 $mode_i$ 是否为 CBC|OFB|CFB，判断此作业是

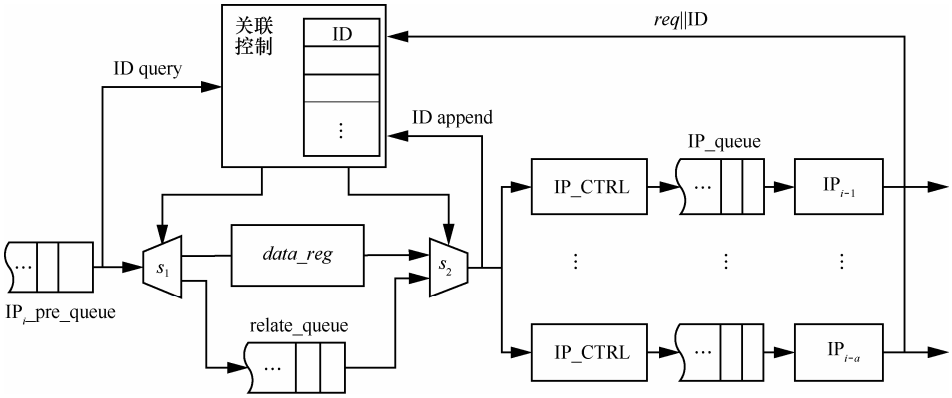

图 4　第二级调度模块内部逻辑

否为依赖作业；其次，提取依赖作业的 ID 号 $task_i$，与关联 ID 表中的 ID 号进行对比，若相同，则启动关联控制开关 s_1，将此作业分组放入关联队列中；否则，将此作业分组送入预处理模块。

② 关联队列的输出：每个 IP 模块在运算完一个关联作业分组后，都会向关联控制模块输出一个 *req* 信号以及作业分组的 ID 号。关联控制模块接收此 ID 号，并对关联 ID 表进行遍历查询，若存在相同的 ID 号，控制关联选择开关 s_2 将此 ID 号对应的作业分组从关联队列送入 cyp_i 指定的预处理模块；否则，丢弃此 ID 号。

第二级调度算法如下。

1) 若 pre_queue 非空，则关联控制模块从 pre_queue 中读取作业分组头数据。

2) 关联控制模块对分组头数据进行解析，获取 *inter_ip id* 号、作业分组长度和算法工作模式。若 $mode_i$=ECB，则同一业务不同作业分组间没有关联关系，控制选择开关 s_1 将数据送 *data_reg*；若 $mode_i$ 为 CBC|OFB|CFB，则此作业分组为关联作业分组，将 $task_i$ 与关联 ID 表进行对比，若存在相同的 ID 号，则控制选择开关 s_1 将数据送 relate_queue；否则，此作业分组为关联任务的第一个作业分组，数据送 *data_reg*，同时，将此作业分组的 $task_i$ 追加至关联 ID 表中。

3) 若关联控制模块没有收到 *req* 请求，则选择开关 s_2 根据请求的 *inter_ip id* 号将 *data_reg* 中的数据送入 IP_CTRL 模块进行运算前的准备；否则 s_2 模块根据请求的 *inter_ip id* 号，从 relate_queue 中选取具有相同 $task_i$ 的作业分组送入 IP_CTRL 模块。

4) 预处理模块为算法的运算做准备，用于获取 IP 核运算需要的数据，包括 *key* 和 *IV*，并将数据送密码算法入口队列。

两级调度算法保证了具有上下文相关的作业分组处理时的正确性，同时，其他业务的作业分组可以在某个关联业务作业分组的流水处理过程中，进入流水处理阶段，实现了数据处理的不断线。

5　中间状态管理

在不同业务流的作业分组随机交叉下传的过程中，不能保证同一业务作业分组的连续性，也不能保证关联业务的后序作业分组 $P_{i(a+1)}$恰好出现在 P_{ia} 运算完成时，因此，必须将其运算结果 C_{ia} 暂存起来，以备后序作业分组到来时使用。考虑到数据传输的可靠性，这里，对所有业务的中间状态均进行保存，在系统中增加密钥和中间状态寄存器 KSM，用于保存各作业分组的密钥以及不同作业分组运算过程中生成的中间状态数据。如图 5 所示，为了保证密钥、中间状态与作业分组间的对应关系，以业务唯一标识号 $task_i$ 为指针，进行密钥、中间状态的存储与获取。算法处理模块 IP 在输出运算结果的同时，将运算结果连同其密钥 *key* 一起以 $task_i$ 为存储地址存入 KSM 中，即每个业务流有唯一的 KSM 地址，从而保证中间状态数据提取的正确性。作业分组在进入算法处理队列 IP_queue 前，算法预处理模块首先根据作业分组头中的 $task_i$ 获取本作业分组需要的 *key* 和中间状态，并根据运算模式 $mode_i$ 获得正确的算法模块入口数据。

在这种处理方式下，由于数据的调度顺序，入口数据与初始向量间的运算都已在数据进入算法模块入口队列之前完成，所以算法模块的功能相对单一，仅仅专注于密码运算，所以可以方便地进行模块的线性扩展。同时，这种设置降低了传输过程中因意外或出错导致的上下文相关的数据运算的重传量。如若某业务的第 *i* 分组数据在传输时发生

了错误，由于第 i–1 分组数据已正确传输，且运算结果保存在 KSM 中，所以不需要对整个业务作业流进行重传，只需要从第 i 分组开始重传即可。每个作业分组根据其 ID 号，即可获知运算的密钥，并自动切换至其前序作业分组的加解密状态，保证了上下文相关的作业分组间数据的正确性，避免数据传输延迟，实现数据、密钥、中间状态间的同步，提高了业务处理的快速性。

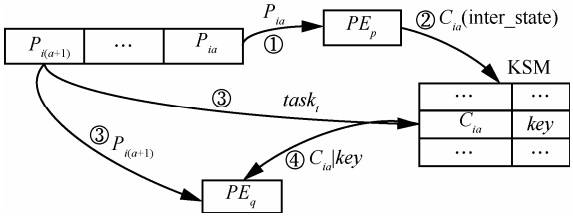

图 5　算法中间状态的处理

6　性能分析及原型测试

6.1　性能分析

第一级调度通过 *cluster_id* 与簇索引表的对比获得作业分组与处理节点间的映射关系，采用硬件的实现方式，可以实现时间复杂度为 $O(1)$的检索速度，用 t_1 表示第一级的调度时间。第二级调度的预处理包括 2 个环节，关联判断、入口数据获取，其处理时间分别用 t_{21}、t_{22} 表示。若系统中共包含 4 种密码算法，共 8 个算法 IP 核，其中，每种密码算法实现有 2 个 IP 核，每种算法的处理时间依次用 t_{31}、t_{32}、t_{33}、t_{34} 表示，反馈时间用 t_4 表示。若 t_1=1, t_{21}=1, t_{22}=2, t_{31}=18, t_{32}=32, t_{33}=16, t_{34}=64, t_4=3，请求密码处理的业务固定为 100 个，随机生成分属于 100 个业务的不同个数的作业分组，对作业流的调度时间与轮询调度(RR, round-robin)进行对比分析，结果如图 6 所示。

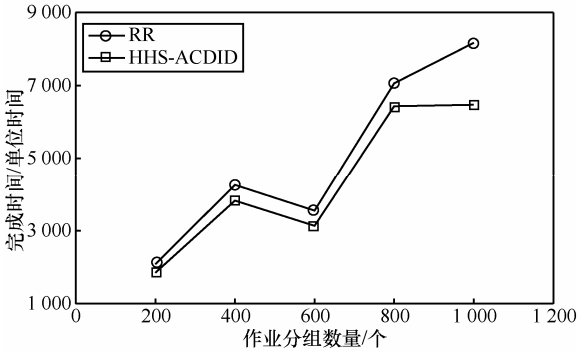

图 6　4×2 算法 IP 核下调度算法性能对比

由图 6 可知，随着随机交叉进入系统的作业分组的数量增多，调度时间的减少呈增大趋势，图 7 所示为 4 种共 16 个算法 IP 核在同样的作业流条件下与 8 个算法性能比较的仿真测试结果，可见随着处理节点的增多，总的调度时间呈倍数减少。

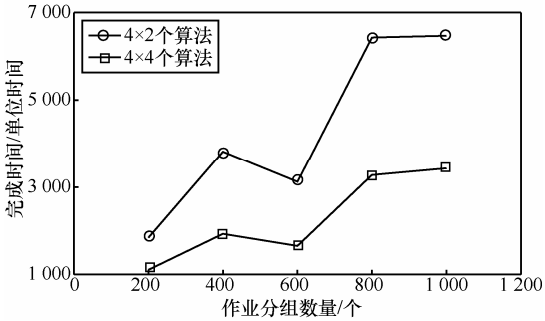

图 7　4×4 与 4×2 算法 IP 核下调度算法性能对比

6.2　实现及测试

为验证调度方法的正确性，本文采用 Verilog 语言在 Xilinx XC7K325t FPGA 上对上述调度算法进行了设计实现，并通过在 FPGA 上实现不同数量的算法 IP 核，对处理速度进行了测试。图 8 给出了 2 个 SM_2、2 个 SM_3 和 1 个 SM_4 算法核情况下，业务数据以随机交叉的方式进入系统时的两级调度

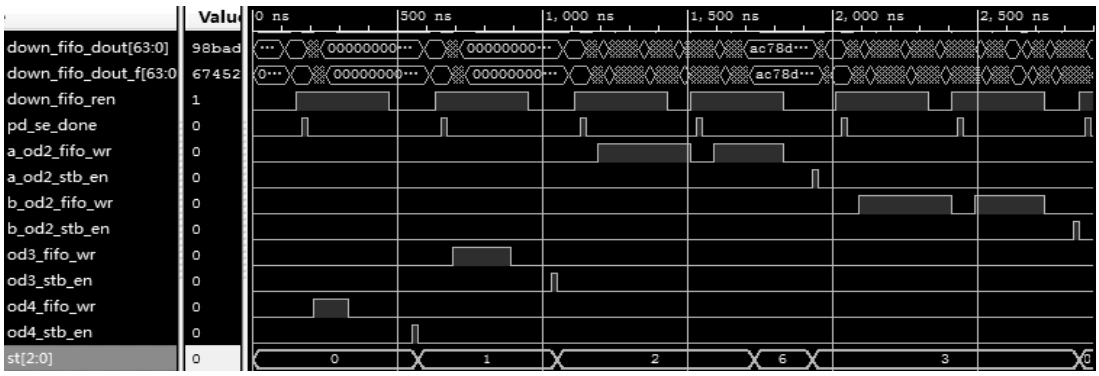

图 8　两级调度仿真结果

仿真结果，其中，*a_od2_fifo_wr*、*b_od2_fifo_wr* 为 2 个 SM_2 算法模块的入口 FIFO 写信号，*a_od2_stb_en*、*b_od2_stb_en* 为作业分组写完成信号，*od3_fifo_wr*、*od4_fifo_wr* 分别为 SM_3、SM_4 算法模块的入口 FIFO 写信号，*od3_stb_en*、*od4_stb_en* 为对应算法模块作业分组写完成信号，从图 8 可以看出此方案实现了对不同业务、不同长度作业分组的正确调度。

在原型系统下对系统处理数据的速度进行测试，图 9 所示为采用 2 个和 3 个 SM_2 核，线程数分别为 32、64、128 和 256 时签名、验签的速度。

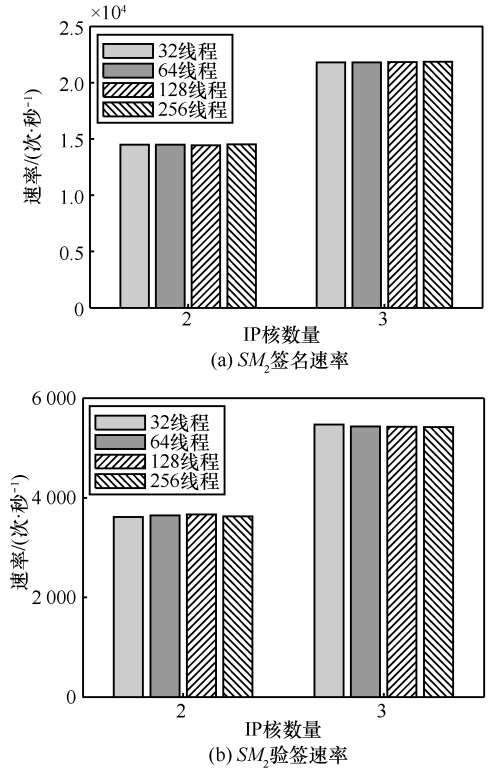

图 9　算法核和线程数量不同时 SM_2 签名、验签速率

测试证明当采用 2 个 SM_2 核时，SM_2 的签名速度可以到达每秒 14 000 次以上。Safenet 公司的高性能硬件安全模块（HSM）：Luna PCI-E 7 000 每秒可执行 7 000 次 RSA 1 024 位交易，本设计下的签名达到其 2 倍以上的运算速度，并且在某一算法处理节点数量不变的情况下，改变处理线程数，作业流的处理速度不受影响。当增加算法的处理节点数量时，业务流的处理速度近似线性增加。

图 10 为采用不同的线程数下发相同数据量的作业流进行 SM_4 加密处理时吞吐率，可以看出随着线程数的增加，SM_4 算法的处理速度有所增长，由于在相同数据量的前提下，线程数的增多，意味着相互独立的任务数增多，关联检索操作相应减少，从而提高了数据处理的吞吐率。文献[18]采用 6 个 IP 核在 FPGA 上实现 AES 算法，时钟频率为 177.9 MHz，处理速度为 5.6 Gbit/s。本文在作业流随机交叉加解密的情况下 SM_4 算法的实现速度是其 1.2 倍以上。

当在 FPGA 上实现的算法核的种类和个数分别为 $3SM_2+2SM_3+SM_4$ 时，LUT 的占用率为 43%，Registers 的占用率为 24%，Logic Distribution 的占用率为 66%，RAM 的占用率为 78%，系统能够实现的最高工作频率为 175 MHz，总功耗为 4.927 W。

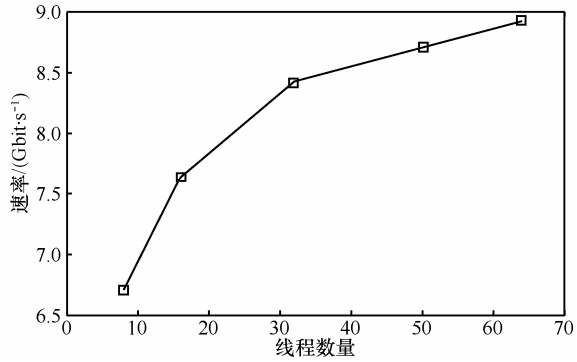

图 10　相同数据量、不同线程数下 SM_4 加密处理速度

7　结束语

本文设计的多密码算法随机作业流硬件调度方法相对于软件调度算法具有调度速度快的优势，能够实时地根据底层密码模块的运算速度进行作业流的调度，保证了密码模块服务的高效性，同时简化了上层应用程序的处理。根据特定的作业分组业务标识，两级调度机制以流水线的方式保证了密码服务设备入口数据流的高速不阻塞输入。关联控制和中间状态管理模块的引入保证了入口数据流高速接收前提下不同工作模式数据的正确调度，从而满足了多算法、多 IP 核、多作业流随机交叉情况下的密码服务需求，解决了单一芯片支持多算法、多 IP 核的并行运算的高并发处理问题。采用此调度方法实现面向云计算的高性能综合密码系统，其中，多核加解密的实现采用 Xilinx XC7K325t FPGA，在单芯片的情况下，可以支持 4 900 万个密码线程和 35 亿个并发应用，SM_4 加解密性能达到 6.4 Gbit/s、SM_3 杂凑性能达到 1.1 Gbit/s、SM_2 签名速率达到 10 000 次/秒以上、SM_2 验签速率达到 2 700 次/秒。

参考文献：

[1] 王蕾, 崔慧敏, 陈莉, 等. 任务并行编程模型研究与进展[J]. 软件学报,2013, 24(1): 77-90.
WANG L, CUI H M, CHEN L, et al. Research on task parallel programming model[J]. Journal of Software, 2013, 24(1):77-90.

[2] SEOG C S, JUNG H P, DONG H L, et al. An efficient implementation of KCDSA on graphic processing units[C]//2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering. 2011: 167-172.

[3] QIU C, LU Y Q, GAO P D, et al. A parallel bulk loading algorithm for m-tree on multi-core CPU[C]//2010 Third International Joint Conference on Computational Science and Optimization. 2010: 300-303.

[4] SATOH A. High-speed parallel hardware architecture for galois counter mode[C]//2007 IEEE International Symposium on Circuits and Systems. 2007: 1863-1866.

[5] WU F, WANG L, WAN J G. A low cost and inner-round pipelined design of ECB-AES-256 crypto engine for solid state disk[C]//2010 IEEE Fifth International Conference on Networking, Architecture, and Storage. 2010: 485-491.

[6] LARA P, BORGES F, PORTUGAL R, et al. Parallel modular exponentiation using load balancing without precomputation[J]. Journal of Computer & System Sciences, 2012, 78(2):575-582.

[7] AMINI E, JEDDI Z, BAYAYOUMI M. A high-throughput ECC architecture[C]//2012 19th IEEE International Conference on Electronics, Circuits, and Systems. 2012: 901-904.

[8] MOZAFFARI K M, REYHANI M A. Efficient and high-performance parallel hardware architectures for the AES-GCM[J]. IEEE Transactions on Computers, 2012, 61(8): 1165-1178.

[9] ZHANG K, HU J Y, HUA B. A holistic approach to build real-time stream processing system with GPU[J]. Journal of Parallel and Distributed Computing, 2015, 83: 44-57.

[10] HAYWARD R, CHIANG C C. Parallelizing fully homomorphic encryption[C]//2014 International Symposium on Computer, Consumer and Control. 2014: 721-724.

[11] HAYWARD R, CHIANG C C. An architecture for parallelizing fully homomorphic cryptography on cloud[C]//2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems. 2013: 72-77.

[12] 李凤华. 分布式信息系统安全的理论与关键技术研究[D]. 西安电子科技大学, 2009.
LI F H. Theory and key technologies for the security in distributed information systems[D]. Xidian University,2009.

[13] HUANG W, HAN J, WANG S A, et al. A low-complexity heterogeneous multi-core platform for security soc[C]//2010 IEEE Asian Solid-State Circuits Conference. 2010: 1-4.

[14] LIU Q, XU Z, YUAN Y. High throughput and secure advanced encryption standard on field programmable gate array with fine pipelining and enhanced key expansion[J]. IET Computers & Digital Techniques, 2015, 9(3): 175-184.

[15] WANG M Y, SU C P, HORNG C L, et al. Single- and multi-core configurable AES architectures for flexible security[J]. IEEE Transactions on Very Large Scale Integration Systems, 2010, 18(4): 541-552.

[16] XIAO L, LI Y, RUAN L, et al. High performance implementation of aria encryption algorithm on graphics processing units[C]//IEEE International Conference on High Performance Computing & Communications & IEEE International Conference on Embedded & Ubiquitous Computing. 2013: 504 - 510.

[17] 方跃坚, 沈晴霓, 吴中海. 一种超椭圆曲线密码处理器并行结构设计[J]. 计算机研究与发展, 2013, 11: 2383-2388.
FANG Y J, SHEN Q N, WU Z H. A parallel architecture for FPGA based hyperelliptic curve cryptoprocessor[J]. Journal of Computer Research and Development, 2013, 50(11): 2383-2388.

[18] XUAN K D, LOUISE S, COHEN A. Managing the latency of data-dependent tasks in embedded streaming applications[C]//2015 IEEE 9th International Symposium on Embedded Multi-core/Many-core Systems-on-Chip. 2015: 9-16.

[19] CHEN W, FEKETE K, YOUNG C L. Exploiting deadline flexibility in Grid workflow rescheduling[C]//2010 11th IEEE/ACM International Conference on Grid Computing. 2010: 105-112.

[20] SPASIC J, LIU D, CANELLA E, et al. Improved hard real-time scheduling of CSDF-modeled streaming applications[C]//2015 International Conference on Hardware/Software Codesign and System Synthesis. 2015: 65-74.

[21] BAMAKHRAMA M, STEFANOY T. Hard-real-time scheduling of data-dependent tasks in embedded streaming applications[C]//The Ninth ACM International Conference on Embedded Software. 2011: 195-204.

[22] CLEMENTE J A, RANA V, SCIUTO D, et al. A hybrid mapping-scheduling technique for dynamically reconfigurable hardware[C]//2011 21st International Conference on Field Programmable Logic and Applications. 2011: 177-180.

[23] JOVANOVIC S, TANOUGAST C, WEBER S. A hardware preemptive multitasking mechanism based on scan-path register structure for FPGA-based reconfigurable systems[C]//Second NASA/ESA Conference on Adaptive Hardware and Systems. 2007: 358-364.

作者简介：

李莉（1974-），女，山东青岛人，西安电子科技大学博士生，北京电子科技学院副教授、硕士生导师，主要研究方向为网络与系统安全、嵌入式系统安全应用。

史国振（1974-），男，河南济源人，博士，北京电子科技学院副教授、硕士生导师，主要研究方向为网络与系统安全、嵌入式安全。

耿魁（1989-），男，湖北红安人，博士，中国科学院信息工程研究所助理研究员，主要研究方向为网络安全。

董秀刚（1976-），男，山东莒县人，北京电子科技学院讲师，主要研究方向为信息安全、密码工程实现。

王璇（1991-），女，山东菏泽人，西安电子科技大学硕士生，主要研究方向为多核调度。

李凤华（1966-），男，湖北浠水人，博士，中国科学院信息工程研究所副总工程师、研究员、博士生导师，主要研究方向为网络与系统安全、隐私计算、可信计算。