计算机科学
COMPUTER SCIENCE

基于跳表的secGear性能优化方法

唐鑫, 狄农雨, 杨浩, 刘忻

引用本文

唐鑫, 狄农雨, 杨浩, 刘忻. 基于跳表的secGear性能优化方法[J]. 计算机科学, 2024, 51(6A): 230700030-5.

TANG Xin, DI Nongyu, YANG Hao, LIU Xin. Optimum Proposal to secGear Based on Skiplist[J]. Computer Science, 2024, 51(6A): 230700030-5.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230700030

基于跳表的 secGear 性能优化方法

唐　鑫[1]　狄农雨[1]　杨　浩[2]　刘　忻[1]
1 兰州大学信息科学与工程学院　兰州 730000
2 国网甘肃省电力公司数字化事业部　兰州 730000
(tangx1230@163.com)

摘　要　机密计算自提出以来,已成为云计算安全问题的重要解决方案。其凭借为云用户提供一个隔离的可信执行环境(TEE),来保证代码和数据的机密性和完整性。但目前主流的机密计算技术存在 I/O 较慢等性能瓶颈,因此,如何提高机密计算的性能成为了研究热点。现有研究未从数据本身出发进行优化,并不适用于大数据的真实环境。在 TEE 中设计并实现了一种能够高效组织管理数据的跳表数据结构,以优化机密计算的运行效率,降低 TEE 中数据处理的开销。最后,通过在国产机密计算框架 secGear 中进行了对比实验,证明所提方法相比红黑树在数据顺序插入、删除、查找的时间开销方面分别获得了 13.5%,10.5%以及 1.9%的提升,相比链表在随机插入时性能也得到了明显的提升,能有效提高机密计算的运行效率,具有更好的实际应用意义。

关键词:机密计算;跳表;secGear;云安全
中图分类号　TP309

Optimum Proposal to secGear Based on Skiplist

TANG Xin[1],DI Nongyu[1],YANG Hao[2] and LIU Xin[1]
1 School of Information Science & Engineering,Lanzhou University,Lanzhou 730000,China
2 State Grid Gansu Electric Power Company Digital Division,Lanzhou 730000,China

Abstract　Confidential computing has been an important method to protect the cloud computing security since it is proposed. It can provide an isolated trusted execution environment(TEE) for user space on computing platform to ensure the confidentiality and integrity of critical user code and data. However,the current mainstream confidential computing technology has performance bottlenecks such as slow I/O. Therefore,how to improve the performance of confidential computing has become a research hotspot. Existing researches haven't thought of data itself,thus can't work well in complex practical scenes. A skiplist data structure that can organize and manage data efficiently in TEE is proposed to optimize the operational efficiency of confidential computing and reduce overhead of processing data in TEE. Finally,comparison experiments are conducted using secGear to prove that comparing with red-black tree,the skiplist can improve the efficiency of confidential computing for 13.5%,10.5% and 1.9% when conducting insertion,deleting and searching respectively,and shows obvious improvement for random insertion when comparing with list. It shows that this proposal can improve the operational efficiency of confidential computing and has practicability.

Keywords　Confidential computing,Skiplist,secGear,Cloud computing

1　引言

根据 Gartner 的预计,截至 2024 年,中国终端用户在系统基础设施和基础设施软件上的支出将有近 40%转移到云服务支出[1]。随着云服务的普及,如何防止云服务商及其他攻击者窃取机密数据正在成为关注的重点,同时也是相关研究的热点与难点。针对敏感数据保护,传统做法是使用密码学技术为数据机密性和数据完整性提供保护,但这些技术主要被用于保护传输态和存储态的数据,目前对"使用中"的数据提供安全防护的技术仍旧属于新的前沿领域。为了解决数据在使用过程中的安全性问题,机密计算技术应运而生。

目前主流的机密计算是一种基于硬件可信执行环境(TEE)的隐私保护技术。其为在底层硬件构建最小信任依赖,将操作系统、Hypervisor、基础设施、系统管理员、服务提供商等都从信任实体列表中删除,视为未经授权的实体,从而减少潜在的风险,保护可信执行环境中数据的机密性、完整性[2]。因此,引入机密计算技术框架可以有效保障数据在运行态下的安全性,提供更加全面的数据安全防护。

目前较为成熟的机密计算技术主要有 TrustZone 和 SGX。TrustZone 由 ARM 于 2002 年提出,SGX 则在 2013 年由 Intel 提出。一直以来,各大厂商不断完善各自产品,旨在保证安全性的前提下提升可用性。ARM 基于新一代 Cortex-

基金项目:基于零信任的工业互联网数据安全的研究(lzujbky-2022-04);基于电力物联网边缘计算的轻量化模型关键技术研究((22)0834)
This work was supported by the Research to Industrial Internet Data Security Based on Zero Trust(lzujbky-2022-04) and Research to Lightweight Model Key Technology Based on Power Internet of Things Edge Computing((22)0834).
通信作者:刘忻(xinl@lzu.edu.cn)

M 处理器对 TrustZone 进行了架构调整，相比 Cortex-A 处理器，调整后的 TrustZone 更加能适应低端设备[3]。2016 年，Intel 发布了 SGX2 代规范，但直到 2021 年才正式发布支持 SGX2 代的云服务器[4]。SGX2 代具有支持动态更新、有大量安全内存的特点，允许配置 1Tbyte 的加密内存，有效缓解了原先在 SGX1 代的性能瓶颈[5]。

对于现有的机密计算框架，如果飞地(enclave)中应用代码及数据大小超过页面大小，则会发生缺页异常，产生换页开销。同时，现有技术出于对安全性的考虑以及设计的局限，限制了 enclave 系统资源分配以及对操作系统资源的访问。Enclave 中无法使用系统调用(如 fread，fwrite，time 等系统调用函数)将导致频繁的模式切换，产生大量开销。此外，SGX 的内存单加密引擎会在数据及代码往返于处理器缓存与 enclave 之间进行加解密操作，产生大量数据加解密开销[6]。

为了提升机密计算的性能，众多研究者开展了深入的研究。为了减少 SGX 使用开销，部分研究通过修改 SGX 运行或内核模块，来提升基于系统级优化的 SGX 性能[7]。然而，系统级优化并不能准确针对程序特性(如线程模型)，因此对于传统 SGX 应用程序并不适用。SCONE[8]是一种基于容器(container)的 SGX 框架，其通过为 SGX 应用提供标准库接口，来避免系统级修改。同时，SCONE 支持基于用户级线程和异步系统调用机制，降低了 SGX Enclave 内线程同步和系统调用对性能的影响。Eleos[9]引入了一种新的安全用户管理虚拟内存(SUVM)抽象，在飞地内部实现应用程序级分页，解决了操作系统上下文转换的开销问题，但尚未解决数据转换和完整性树导航的问题。Taassori 等[10]评估了飞地内存页交换性能开销，分析了制约飞地内存大小的根本因素——用于保护飞地内存完整性的树结构的性能、飞地完整性验证所需的空间开销，并提出了 VAULT(Variable Arity Unified encrypted-Leaf Tree)数据结构，提高了树结构的性能，并通过 MAC 共享和压缩减少了完整性验证的空间开销，从而能够支持更大的飞地内存，避免飞地内存页交换。Yu 等基于 SGX Enclave 提出了 ELASTICLAVE 模型[11]，不同 enclave 通过获取和转移独占锁访问共享内存区域，实现跨 enclave 数据共享，具有更好的性能。

2020 年，华为 openEuler 推出机密计算统一开发框架 secGear，其也是目前国内最为成熟的机密计算框架。secGear 的零切换特性通过共享内存减少富执行环境(REE)与 TEE 上下文切换及数据拷贝次数，优化 REE 与 TEE 交互性能的技术，有效降低了机密计算在模式切换方面的开销。但是，由于 secGear 需要兼容多种设备以及技术，其不可避免地存在平台数据融合的问题，需要通过调度优化来进一步提高性能。同时，其仍然需要多次创建与销毁 enclave，仍然存在性能瓶颈。

现有针对降低机密计算额外开销的解决方案主要基于以下方面：1)增加操作系统库/库函数，减少模式切换；2)无切换调用方法；3)共享内存技术。现有研究大多基于硬件、软件两方面展开，通过调整硬件结构、优化软件设计来提升机密计算框架的工作性能，未从数据组织的角度进行研究，且目前也很少有针对 secGear 开展的优化。硬件带来的性能堆叠可以一定程度上解决性能的瓶颈问题，对于小型、轻量级应用场景，提升效果较为明显。然而，面对海量数据，性能的堆叠会产生边际效应。只是为 TEE 分配大量系统资源已经不足以提升数据处理效率，并不适用于长远发展，亟须提出新的机密计算内存优化方法，以提高效率。

新硬件特性下影响可信任执行环境性能的主要因素已经发生转变，计算效率成为了比内存占用空间更值得关注的问题。如果提前对数据进行再组织，可以加速 enclave 中的计算，同时可以不用总销毁 enclave，从而提升性能。从数据本身出发，对数据的组织方式进行调整，使用特定的数据结构进行数据封装以便 TA 计算，可以有效降低 TA 处理数据过程中的性能开销。因此，需要考虑如何高效组织数据，从数据类型、数据用途及数据处理方式等角度出发对数据进行再分类，根据数据特征设计良好的数据结构，降低数据存储、传输、计算时 TEE 产生的额外开销，提升机密计算的可用性。

针对上述问题，本文提出在 TEE 中使用跳表数据结构对数据进行查找、插入和删除。该数据结构在链表的基础上增加了多层索引，性能近似于“二分”查找，且具有比红黑树等二叉树更小的内存占用，可以从数据组织层面优化机密计算的运行效率，以降低 TEE 的额外开销，提升机密计算性能。最后文章通过复杂度分析及在 secGear 中进行对比实验，来证明该数据结构能够有效降低 TEE 中进行操作的时间开销，相比普通有序链表以及以红黑树为代表的二叉树具有更高的实用性。

2 背景知识

2.1 secGear

secGear 是面向计算产业的机密计算安全应用开发套件，其包含 Base Layer，Middle Layer 和 Service Layer 三层架构。BaseLayer 可以提供丰富的 enclave 开发接口或工具，并在安全侧支持相应接口。MiddlewareLayer 提供常见的安全协议组件以及各种安全函数库，通过 TLS 安全地与 TEE 进行交互，开发者无需感知安全侧编程，提高了应用开发的安全性。ServiceLayer 可以提供通过 enclave 增强的服务，如 PCK11 组件等，其通过数据加密技术对 TA 的主密钥进行加密，并进行保存管理。secGear 目前支持 Intel SGX 硬件和 ARM Trustzone(安全 os 支持 iTrustee)，且很好地解决了两者兼容性差等问题。TrustZone，SGX 和 secGear 的性能对比如表 1 所列[13-15]。

表 1 Trustzone，SGX 和 secGear 的对比

Table 1 Comparison of Trustzone，SGX and secGear

	TrustZone	SGX	secGear
硬件实现方法	利用总线对 CPU 时间片切换	制定安全内存分区并辅以单独的 SGX 汇编	与芯片相关
部署应用方法	代码中定义	自行部署可信应用	集于多种接口自由开发
交互方式	SMC 指令通信	通过 EENTER、EEXIT 等指令集进行 enclave 的进出和内存操作	基于芯片交互方式及 TLS 安全协议
并发性	低并发	较低并发	高并发

应用可以运行在被称为飞地的可信执行环境中，其由用户态程序创建。运行在飞地中的程序是可信的，被称为可信代码；运行在飞地外的程序是不可信的，被称为不可信代码。两者通过用户定义的 Ecall 和 Ocall 代码进行互相调用，以保证机密计算和实现程序功能。

2.2　跳表

跳表(Skiplist)[15],又称跳跃表、跳跃列表,是在线性链表的基础上进行改进的一种有序的概率型数据结构[16]。其通过在插入过程中随机地将一些元素提升为高度为 1～r 的索引以及使用多个指针域来对原有的有序链表添加多级索引,以支持快速的查找、插入和删除操作。跳表的每一层都是一个有序的链表,其默认是升序的。一般而言,层数越高的索引个数越少,最底层(Level 1)的链表包含所有元素。如果一个元素出现在 Level i 的链表中,则它在 Level 1 至 Level $i-1$ 的链表中也都会出现。跳表具有结构简单、易于实现、无锁并发、支持快速查找、插入以及删除等优点[17],因此被广泛应用于 LevelDB,MemSQL 以及 Redis 等数据库中,成为了数据库领域的一个重要索引技术。

图 1 为含有 10 个元素的跳表示例图,从水平方向来看,每一层都是一个单向链表,而且越往上层级的链表所包含的结点个数越少,从左至右元素的数值递增。跳表通过维护每个结点对应的层数,保证每层结点个数都比下一层大致减半。

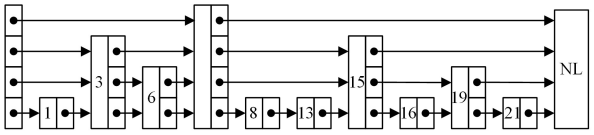

图 1　跳表示例图

Fig. 1　Example of skiplist

3　基于跳表的 secGear 性能优化方案

本文提出了一种基于跳表数据结构的机密计算技术优化方案。如图 2 所示,通过将跳表数据结构内置于 TEE 中对数据进行组织,以实现在 TEE 侧对数据进行高效的查找、插入以及删除操作,在保证安全的基础上提升效率。

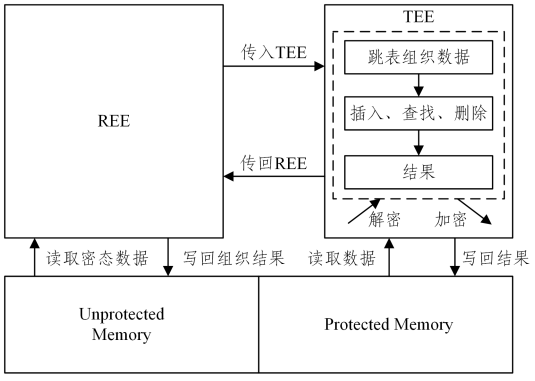

图 2　secGear 中的数据组织结构

Fig. 2　Data organization structure in secGear

3.1　查找元素

3.1.1　算法流程

跳表支持单个元素的查找和范围查找。在跳表中查找元素的过程如下:

1)从顶层链表的首元素开始,向该层链表的尾元素方向进行搜索,直至找到一个大于或等于目标的元素,或者到达当前层链表的尾部。

2)如果当下元素等于目标元素,则表明该元素已被找到,搜索结束,返回结果。

3)如果当下元素大于目标元素或已到达链表的尾部,则退回到当前层的前一个元素,然后转入下一层进行搜索。

算法 1 阐述了跳表中对于进行单个元素查找的算法。

算法 1　跳表元素查找算法

输入:跳表结构 skiplist,要查找的键 searchKey

输出:查找结果

```
1. Function Search(skiplist,searchKey)
2. x:=skiplist→header
3. -- loop invariant:x→key <searchKey
4. For i:=skiplist→level downto 1 do
5.      While x→forward[i]→key < searchKey do
6.          x:=x→forward[i]
7. --x→key=searchKey ≤ x→forward[1]→key
8. x:=x→forward[1]
9. If x→key=searchKey then return x→value
10.     Else return False
```

3.1.2　复杂度分析

在可支持超大容量 EPC 的情况下,时间开销成了提高机密计算性能的关键考虑因素。本文假设各数据结构所包含的元素个数均为 n,后续不再赘述。

有序链表查找元素时需要从头指针开始,逐个进行遍历直至找到目标结点,查询复杂度为 O(n)。跳表以及红黑树的查找性能均近似于“二分查找”,因此两者查找的平均时间复杂度均为 O(lbn)。

3.2　插入元素

3.2.1　算法流程

跳表结点的插入是一种随机算法,其通过随机函数(几何分布)控制每个结点的指针个数(层数),从而保证每一层的“链表”个数依次减半。跳表的插入操作主要包含以下 3 个步骤:

1)通过类似查找算法的算法找到待插入的元素,并记录查找过程中的所有下沉结点。

2)采用随机算法决定该结点层数,为该结点分配内存。

3)在每一层上修改插入结点的后继指针和下沉结点的后继指针。

算法 2 阐述了跳表中随机生成层数的算法,其原理相当于做一次伯努利实验,不断进行实验直至失败,统计首次失败的次数,用实验次数 K 作为结点的层数。本文设定随机变量 K 服从参数 $p=1/2$ 的几何分布,其中 p 为该结点在本层出现的前提下,在向上一层出现的概率。K 的期望值 $E[K]=1/p=2$。也就是说,跳表中元素的平均层数为 2 层,包含 N 个元素的跳表的指针域个数大致为 $2n$ 个。

算法 2　基于伯努利实验的跳表层数决策算法

输入:跳表最大层数 MaxLevel

输出:待插入 key 的层数

```
1. Functionrandom_level(MaxLevel)
2. level:=1
3. While(random(0,1)<0.5)
4.      level++
5. |v|:=min(level,MaxLevel)
6. Return |v|
```

算法 3 阐述了标准跳表中进行元素插入的算法。

算法 3　跳表元素插入算法

输入:跳表结构 skiplist,要插入的键 searchKey,要插入的值 newValue

输出:无

```
1. Function Insert(skiplist, searchKey, newValue)
2. Local update[1..MaxLevel]
3. x:=skiplist→header
4. For i:=skiplist→level downto 1 do
5. While x→forward[i]→key < searchKey do
6. x:=x→forward[i]
7. -- x→key=searchKey ≤ x→forward[1]→key
8. Update[i]:=x
9. x:=x→forward[1]
10. If x→key=searchKey then x→value:=newValue
11. Else
12. |v|:=random_level()
13.     If |v|>skiplist→level then
14.         Fori:=skiplist→level+1 to |v| do
15.             Update[i]:=skiplist→header
16. skiplist→level:=|v|
17.     x:=makeNode(|v|, searchKey, value)
18.     For i:=1 to level do
19.         x→forward[i]:=update[i]→forward[i]
20.         Update[i]→forward[i]:=x
```

3.2.2 复杂度分析

有序链表及跳表的插入过程在查找过程的基础上仅仅增加了删除结点相关结点的指针改变以及结点的删除，因此有序链表和跳表插入的时间复杂度分别为 $O(n)$和 $O(lbn)$。红黑树的插入在查找的 $O(lbn)$时间开销上可能涉及旋转以及结点颜色变换操作，这样的操作可能会涉及到整个树的其他部分，影响范围大于跳表结构，因此相比跳表具有更高的时间开销。

3.3 删除元素

3.3.1 算法流程

跳表的删除操作是类似于插入操作的逆操作，需要先查找到结点，然后修改指针域，最后回收该结点。算法 4 阐述了标准跳表中进行元素删除的算法。

算法 4 跳表元素删除算法

```
输入：跳表结构 skiplist，要插入的键 searchKey
输出：无
1. Function Delete(skiplist, searchKey)
2. Local update[1..MaxLevel]
3. x:=skiplist→header
4. For i:=skiplist→level downto 1 do
5.   While x→forward[i]→key < searchKey do
6.       x:=x→forward[i]
7.     Update[i]:=x
8. x:=x→forward[1]
9. If x→key=searchKey then
10.   For i:=1 to skiplist→level d
11.       If update[i]→forward[i] ≠x then break
12.       Update[i]→forward[i]:=x→forward[i]
13.   Free(x)
14.   hile skiplist→level > 1 andskiplist→header→forward
      [skiplist→level]=NIL do
15.       skiplist→level:=skiplist →level-1
```

3.3.2 复杂度分析

删除操作可理解为插入操作的逆过程。因此，链表删除操作的平均时间复杂度为 $O(n)$，而跳表和红黑树进行删除的平均时间复杂度均为 $O(lbn)$。但是同插入过程，红黑树在删除时可能需要做一些平衡及结点颜色变换操作，增加了时间开销。

4 实验及分析

下文将本文使用的跳表结构与红黑树以及有序链表在数据集上进行时间开销的对比。

4.1 实验环境

本次实验基于 secGear 进行性能评估，搭配 6 核 CPU，处理器为 i5-8500，搭载 9MB 三级缓存，运行系统为 Ubuntu 20.04 x86_64。对比实验所使用的 3 种数据结构均基于 C++实现，使用 g++9.4.0 进行编译。数据结构被写入 TEE 中，而数据集存储在 REE 侧，在 REE 侧通过调用接口以使用 enclave 中的数据结构对从非安全侧复制传入的数据进行插入、查找、删除。

4.2 测试数据集

因机密计算内存优化方面尚未存在标准数据集，故将针对数据顺序操作和数据随机操作采用自建数据集对 TEE 中不同数据结构的时间开销进行对比测试，数据集已发布在“码云”平台[1]。数据集的描述如下：

1）Order_10000：将 1～10000 的数据按照从小到大的顺序加入，生成对应的数据集。

2）Random_10000：设置随机种子，随机生成 1～10000 内的不重复整数加入，生成对应的数据集。

4.3 实验结果及分析

图 2 和图 3 分别给出了不同的数据结构在数据集 Random_10000 和 Order_10000 下的实验结果。实验结果表明：1）跳表及链表在时间开销方面存在巨大差异。当数据顺序随机时，链表在插入、查找和删除方面的时间开销分别是跳表的 38.68，6.87 及 6.56 倍。按照数据顺序进行操作时，链表在上述 3 种操作中所花费的时间分别为跳表的 13.02，2.05 以及 2.06 倍。相较之下链表明显体现出了较差的性能。2）跳表与红黑树在数据顺序随机时性能基本持平。但在按照数据大小顺序进行操作时，跳表相对于红黑树在插入、删除、查找的时间开销方面分别获得了 13.5%，10.5%以及 1.9%的提升。原因可能是红黑树会因为数据结构的退化而产生性能下降，因此产生的时间开销大于跳表。3）对比之下，跳表结构在数据随机插入以及顺序插入的情景下均适用。

综上，相对于有序链表以及以红黑树为代表的二叉树，跳表数据结构能够更加有效地降低 TEE 中查找、插入以及删除等操作的时间开销，具有更好的实用性，适用于更多场景。

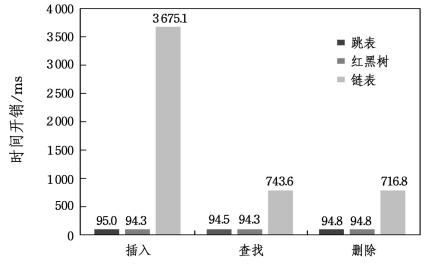

图 3 Random_10000 数据集上对应操作开销

Fig. 3 Time overhead on Random_10000 dataset

[1] https://gitee.com/ICC-NSG/lzu-icc-nsg/tree/mast-er/secGear+openGauss%20code/skliplist_enclave

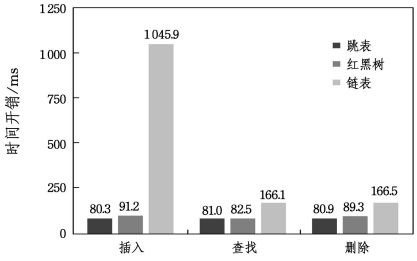

图 4　Order_10000 数据集上对应操作开销

Fig. 4　Time overhead on Order_10000 dataset

结束语　本文针对机密计算环境性能开销问题进行分析，提出了在 TEE 中使用跳表结构以降低对数据进行插入、查找和删除的时间开销。对跳表、红黑树以及有序链表进行了复杂度分析，并使用 secGear 在自建数据集上进行了对比实验。实验结果表明，跳表数据结构具有较低的时间开销且适用于更广泛的场景，确定了高效数据结构在降低机密计算时间开销方面的可行性，对于国产机密计算的普及应用具有积极意义。

但是本文所对比的数据结构有限，且尚未结合实际数据特征及可信执行环境存储内容对跳表结构进行优化。未来工作中，可以考虑探索更多数据结构在 TEE 中的应用，将高效的数据结构与其他机密计算内存优化方法结合，以达到更显著地降低开销的效果，并且基于实际情况对结构进行优化。

参 考 文 献

[1] ZENG E，TIAN U，JI K. Market Guide for Cloud Infrastructure and Platform Service，China[OL]. (2021-03-24)[2023-06-12]. https://www.gartner.com/en/documents/3999770.

[2] Confidential Computing Consortium. A Technical Analysis of Confidential Computing v1. 2[OL]. (2021-09-28) [2023-06-12]. https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/04/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.2_updated_2022-11-02.pdf.

[3] ArmLtd. Trustzone technology for the armv8-m architecture version2. 0[OL]. (2017). [2023-06-12]. https://developer.arm.com/documentation/100690/0200/ARM- TrustZone- technology?lang=en，2017.

[4] LI M Y，XIA Y B，CHEN H B. Memory optimi-zation system for SGXv2 trusted execution environ-ment[J]. Journal of Software，2022，33(6)：20122029.

[5] WANG J W，JIANG Y，LI Q，et al. Survey of research on SGX technology application[J]. Network New Media Techonolgy，2017，6(5)：3-9.

[6] KIM S. An Optimization Methodology for Adapting Legacy SGX Applications to Use Switchless Calls[J]. Applied Sciences，2021，11(18)：8379.

[7] AUBLIN P L，KELBERT F，O'KEEFFE D，et al. Talos：Secure and Transparent TLS Termination inside SGX Enclaves[OL]. http://www.doc.ic.ac.uk/research/technicalreports/2017/DTRS17-5.pdf.

[8] PIETZUCH P R，ARNAUTOV S，TRACH B，et al. SCONE：secure Linux containers with Intel SGX[C]// USENIX. 2016.

[9] ORENBACH M，LIFSHITS P，MINKIN M，et al. Eleos：Exit-Less OS Services for SGX Enclaves[C]// EuroSys. 2017：238-253.

[10] TAASSORI M，SHAFIEE A，BALASUBRAM-ONIAN R. Vault：Reducing pag-ing overheads in sgx with efficient integrity verification structures[C]// Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems. 2018：665-678.

[11] YU J Z，SHINDE S，CARLSON T E，et al. Elasticlave：An efficient memory model for enclaves[C]// 31st USENIX Security Symposium(USENIX Security 22). 2022：4111-4128.

[12] Huawei. secGear[EB/OL]. https://gitee.com/src-openeuler/secGear#introduction，2021-05-11.

[13] WANG X Y. Secure Isolation Based on ARM TrustZone Research and Application[D]. Chengdu：University of Electronic Science and Technology of China，2013.

[14] LIU X，WANG J Y，YANG H R，et al. An Internet of vehicles authentication protocol based on blockchain and secGear framework[J]. Netinfo Security，2022，22(1)：27-36.

[15] PUGH W. Skip Lists：a Probabilistic Alternative to Balanced Trees[J]. Commun. ACM，1990，33(6)：668-676.

[16] YANG Z. Cloud storage of key-value data using trusted execution environments[D]. Chengdu：University of Science and Technology of China，2021.

[17] LI L，WU G，WANG G R. In-memory skiplist optimization technologies based on data feature[J]. Journal of Software，2020，31(3)：663-679.

TANG Xin，born in 2001，undergraduate. Her main research interests include zero trust，confidential computing and artificial intelligence.

LIU Xin，born in 1988，Ph.D，associate professor. His main research interests include confidential computing，zero trust and identity authentication.