计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(1):392−398 [doi: 10.15888/j.cnki.csa.008896]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 RISC-V 的数据安全指令①

刘　阳, 汪　丹, 方林伟, 王利明

(中国科学院 信息工程研究所, 北京 100093)
通信作者: 汪　丹, E-mail: wangdan@iie.ac.cn

摘　要: RISC-V 是基于精简指令集原理建立的免费开放指令集架构, 具有完全开源、架构简单、易于移植、模块化设计等特点. 随着网络高速发展, 安全风险无处不在, 利用 RISC-V 的可扩展特性是一种非常有效地提升 RISC-V 设备安全的方式. 因此, 本文针对 RISC-V 自定义指令的安全能力, 结合可信计算、流密码技术, 设计了简单高效的 RISC-V 自定义指令, 实现基于可信基的数据安全存储功能, 并依托 GNU 编译工具链实现对自定义指令的编译支持, 在模拟器上测试应用程序对自定义指令的调用执行. 该指令充分结合可信计算与流密码的安全特性, 可实现较强的安全性.

关键词: RISC-V; 自定义指令; 数据安全存储; 可信计算; 密码技术; 处理器; 云存储; 隐私保护

引用格式: 刘阳,汪丹,方林伟,王利明.基于 RISC-V 的数据安全指令.计算机系统应用,2023,32(1):392–398. http://www.c-s-a.org.cn/1003-3254/8896.html

Data Security Instruction Based on RISC-V

LIU Yang, WANG Dan, FANG Lin-Wei, WANG Li-Ming

(Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)

Abstract: RISC-V is a free and open instruction set architecture built by the principle of reduced instruction sets, which features complete open source, simple architecture, easy portability, and modular design. With the rapid development of networks, security risks are ubiquitous. The extensibility feature of RISC-V can be utilized to effectively improve the security of RISC-V devices. Therefore, this study designs a simple and efficient RISC-V custom instruction considering the security capabilities of RISC-V custom instructions and by use of trusted computing and stream cipher technology to realize the function of data security storage based on the trusted computing base. Moreover, the compilation support for the custom instruction is achieved with the GNU compilation toolchain. The calling and execution of the custom instruction by an application are tested on a simulator. This instruction fully combines the security features of trusted computing and stream ciphers, and hence, it can achieve strong security.

Key words: RISC-V; custom instruction; data security storage; trusted computing; cryptographic technology; processor; cloud storage; privacy protection

RISC-V 是基于精简指令集 (reduced instruction set computer, RISC) 原理建立的免费开放指令集架构 (instruction set architecture, ISA), 是第 5 代 RISC. RISC-V 作为一种指令集规范, 具备完全开源、架构简单、易于移植、模块化设计、完整工具链等特点[1], 在物联网领域、嵌入式市场等具有广泛的应用前景. 在 RISC-V 基金会和 SiFive 公司的带动下, 国内外芯片厂商、研究机构纷纷加入其研究热潮. 中国工程院院士

① 基金项目: 国家重点研发计划 (2019YFB1005200)
收稿时间: 2022-05-09; 修改时间: 2022-06-15; 采用时间: 2022-07-01; csa 在线出版时间: 2022-10-28
CNKI 网络首发时间: 2022-11-15

倪光南在 2021 RISC-V 中国峰会上表示, 未来 RISC-V 很可能发展成为世界主流 CPU 之一, 从而在 CPU 领域形成 x86、ARM、RISC-V 三分天下的格局.

如今网络高速发展, 安全风险无处不在, 提升 RISC-V 设备的安全性无疑是当下最重要的课题之一[2]. 基于 RISC-V 可扩展的能力, 当前实现 RISC-V 设备安全性最有效的方式之一就是扩展自定义安全指令, 以实现指令层面的安全性[3]. 当前, 在网络安全、人工智能, 边缘计算等大趋势下, 通用指令及通用型处理器有越来越多的局限性, 越来越多的芯片厂商开始打造更为适合自己需求的, 更具能效的自主可控的专用型芯片, 用以代替传统的通用型芯片, 获得更为出色的安全、性能、面积和成本等竞争优势. 由于 RISC-V 开源和可扩展的特性, 处理器设计人员可以根据运行在处理器的应用功能自主添加相应的自定义指令, 以实现更加自主可控的针对特定应用的专用性处理器, 如在一个主要运行图像处理应用的处理器中添加向量指令以实现大量的数据运算, 也可以在一个主要运行安全加密算法的处理器中添加自定义指令以实现密码运算加速[4−7]. 使用自定义扩展指令, 可以实现通用指令满足不了的功能、安全和性能需求. 因此基于 RISC-V 指令集支持自定义扩展的能力, 通过自定义指令的方式扩展 RISC-V 的安全能力, 可以根据需求实现特定的安全功能, 有效增强 RISC-V 芯片的安全性, 且在指令层面实现安全功能还能保证其运行性能.

目前有关 RISC-V 的指令安全扩展的研究有很多. Saarinen 等人[8] 在 RISC-V 上评估了 SNEIK 密码算法, 并进行了相应 RISC-V 指令扩展, 提升了 SNEIK 在 RISC-V 上的执行效率; Yu 等人[9] 在 RISC-V 上提出了基于数据遗忘的指令集扩展, 就安全性而言, 提出的扩展指令可以抵御相关的侧信道攻击, 就性能而言, 扩展指令可高效支持内存遗忘计算; Kim 等人[10] 提出了 RIMI, 一种新的指令扩展, 可在嵌入式系统中提供内存隔离. 因此, 对于提升 RISC-V 安全而言, 指令安全扩展是一种非常有效的方式, 受到研究人员的青睐.

为了防止 RISC-V 平台的敏感数据泄露问题, 迫切需要针对 RISC-V 平台设计自定义指令来实现数据的安全存储. 可信计算技术[11,12] 提出了一种保护数据安全的思路, 即通过可信安全模块来保护和验证软件运行环境是否安全和可信, 以此实现数据的安全存储. 其中密码技术是实现可信计算技术关键机制的基础, 是可信计算技术的核心. 流密码[13] 是对称加密算法的一种, 即加密和解密双方使用相同伪随机加密数据流 (pseudo-random stream) 作为密钥, 明文数据每次与密钥数据流顺次对应加密, 得到密文数据流, 实践中通常使用异或 (xor) 操作实现. 通过设计自定义安全指令, 借鉴流密码的实现逻辑, 将可信基信息与数据信息绑定, 实现加密存储机制的人、机可信信息的隔离, 是保障数据安全的有效方式.

因此, 本文对 RISC-V 自定义指令的安全能力进行研究, 以自定义指令的方式对 RISC-V 进行安全扩展. 结合可信计算、流密码技术设计简单高效的 RISC-V 自定义指令, 实现基于可信基的数据安全存储功能. 最后, 依托 GNU 编译工具链实现对自定义指令的编译支持, 并设计应用程序测试用例, 在模拟器上测试应用程序对自定义指令的调用执行. 该指令充分结合可信计算与流密码的安全特性, 可实现较强的安全性.

1 数据安全指令的设计

本文结合可信计算、内存加密等安全相关应用场景的需求, 提出基于可信基的数据安全存储功能, 根据该功能, 设计并实现了一条指令, 名称为 TSTORE. 第 1.1 节介绍该指令功能, 第 1.2 节介绍该指令的编码设计.

1.1 功能设计

该指令的功能是将需要安全存储的用户明文信息与设备可信基、用户自定义身份信息结合, 实现明文数据的混淆, 将其以密文的形式安全地存储. 其中, 用户自定义身份信息可以是 pin 码、随机数、用户程序 ID、自定义值等与用户身份绑定的信息, 可信基可以是来自外部可信芯片的密钥, 也可以是存储在内核态, 用户态不可随意访问的内核密钥信息, 具体运算如图 1 所示. 其中 y 为加密后数据, x 为明文数据, c 为用户自定义信息, k 为可信计算基信息. 加密后的数据与明文、可信基、用户信息均相关, 从而实现该加密存储机制的人、机、可信信息的隔离. 在保证 k 和 c 两个因子均正确的情况下, 可对所存储数据进行有效获取. 具体公式如下:

$$y = x \oplus k \oplus c \tag{1}$$

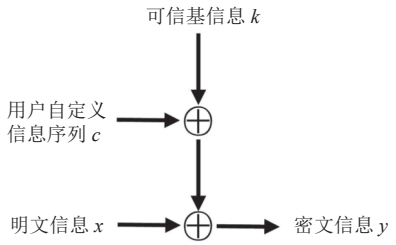

图 1 数据安全指令运算流程

基于上述功能, 可知在指令层面实现安全存储功能需要有效地从用户内存中获取明文数据、用户自定义信息, 从安全内存区域获取可信基信息, 并对这 3 个数据进行异或操作. 由于明文数据和用户自定义信息由用户定义, 因此作为输入被指令获取, 而可信基信息是由平台或内核安全定义与存储, 存储位置固定, 不随着运行的应用程序改变而改变, 因此指令自动去相应安全内存中获取可信基信息, 而非作为输入获取. 指令的功能逻辑如图 2 所示.

1.2 编码设计

本部分介绍自定义安全指令的编码设计. 根据上述功能, 定义一条扩展指令 TSTORE, 该指令隶属于新增的安全指令集模块, 以实现基于可信基的数据安全存储功能.

从 RISC-V 指令集手册上来说, RISC-V 的指令类型被分成了 R-type, I-type, S-type, B-type, U-type, J-type. 每一种类型的指令的格式都不相同, 按照特定的机器码编排的指令有着特殊的用途[14], 如图 3 所示. RISC-V 架构具有开放、标准的扩展方式, 预留了扩展指令空间, 可以用来分配自定义指令. RISC-V 架构的指令 opcode 格式如表 1 所示, 定义了 4 组自定义指令可用的操作码: custom0, custom1, custom2 和 custom3, 每组 custom 的 opcode 各不相同, 结合功能码能够扩展大量自定义指令.

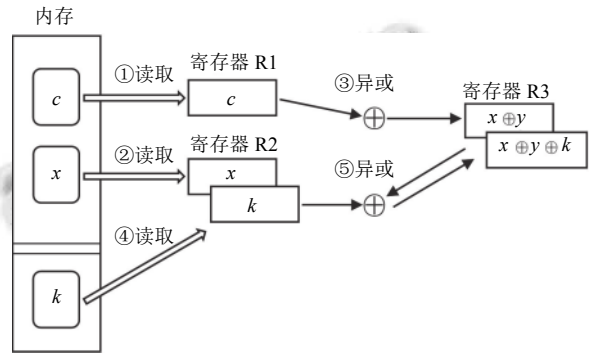

图 2 数据安全指令功能逻辑

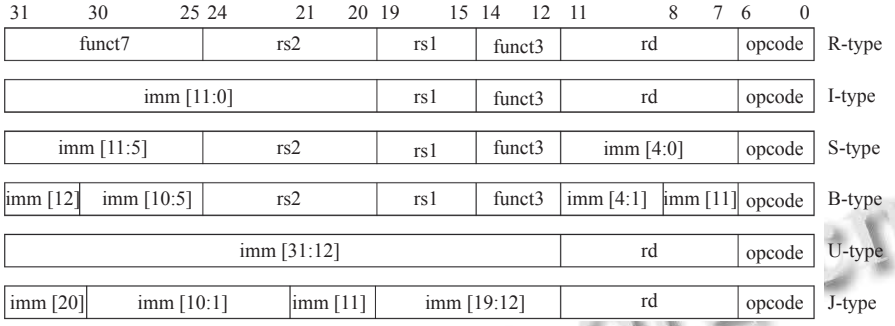

图 3 RISC-V 指令集类型

表 1 RISC-V 基本操作码映射表

Inst[6:5]	Inst[4:2]							
	000	001	010	011	100	101	110	111 (>32 b)
00	LOAD	LOAD-FP	custom-0	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48 b
01	STORE	STORE-FP	custom-1	AMO	OP	LUI	OP-32	64 b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	reserved	custom-2/rv128	48 b
11	BRANCH	JALR	reserved	JAL	SYSTEM	reserved	custom-3/rv128	≥80 b

对于本文自定义的 TSTORE 指令, 根据实际功能需求, 需要两个寄存器的值进行运算, 因此采用 R 型指令格式, 指令格式如下:

tstore rd, rs1, rs2

我们使用 RV64 指令集进行自定义安全 TSTORE 指令扩展, 根据 RISC-V 预留的操作码, 选择可用于 RV64 指令集的 custom-3 作为操作码, 指令的编码格式如图 4 所示. 根据 32 位 R 型指令格式, 25–31 位为 fun7 编码, 可自主设置为 0000110; 12–14 位为 fun3 编码, 可自主设置为 110; 0–6 位为操作码 opcode 编码, 根据 RISC-V 规定选择 custom-3, 为 1111011; 20–24 位为目的寄存器 rs2, 15–19 位, 7–11 位分别为两个源寄存器 rs1 和 rs2, 具体编码由编译器决定.

在 TSTORE 指令中, rs1 和 rs2 分别用来存储明文

信息和用户定义信息的内存地址, 可信基信息存储在固定位置, 该指令自主获取, rd 目的寄存器用来返回安全混淆后的密文信息, 如图 5 所示. 用户可输入明文信息 x 至 rs1, 输入自定义信息 c 至 rs2, 指令获取后可输出密文信息 y 至 rd.

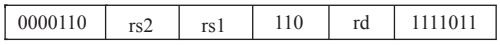

图 4 数据安全指令的编码格式

图 5 数据安全指令的编码格式

2 数据安全指令的实现

对于标准指令集扩展, RISC-V 社区提供了完整的工具链支持, 而对于非标准的自定义指令扩展, 则意味着需要用户自己实现工具链的支持[15]. 本部分首先介绍编译工具链对自定义安全指令的支持实现, 其次介绍在模拟器上对指令的功能实现, 最后对运行结果进行分析.

2.1 编译工具链的支持实现

GNU 工具链是一组用于开发应用程序和操作系统的编程工具的集合, 这些工具构成了一个完整的系统. GCC (GNU compiler collection) 是 GNU 在 1984 年创建的一个编译器系统, 其中包含编译器、调试器、链接器和汇编器等工具. GCC 支持高级编程语言, 如 C/C++、Java、Fortran 等, GCC 也可以为不同的处理器生成机器代码, 如 x86、ARM、MIPS、RISC-V, 并且可以针对特定的处理器进行修改和优化. GNU Binutils (GNU binary utilities) 是一组二进制工具集, 这些工具都是专门针对于二进制文件的[16].

为了满足 RISC-V 架构可添加自定义指令的特性, RISC-V GNU 工具链也提供了在汇编层面支持自定义指令的扩展方式. 从现有的解决办法来看, 主要通过修改 Binutils, 在 opcode/riscv-opc.h 头文件中声明和定义自定义指令, 添加自定义指令的编码, 使 GCC 能够识别新增指令.

示例代码1. 指令定义与声明

```
#define MATCH_TSTORE 0x0C00607B
#define MASK_ TSTORE 0xFE00707F
DECLARE_INSN (tstore, MATCH_ TSTORE, MASK_ TSTORE)
```

2.2 基于模拟器的功能实现

针对基于可信基的数据安全存储操作, 自定义安全指令功能的实现主要解决如下 3 个问题, 以保证可信计算基信息 k 的安全存储与访问及指令操作的机密性, 保证其不对用户可见.

(1) 可信计算基信息 k 的存储与加载

对于自定义安全指令 TSTROE, 可信基信息 k 是固定存放的, 无需将 k 作为参数传递, TSTORE 就可以自动获取 k 的地址并实现安全功能, 因此需要保证 k 的机密性与完整性. 实现时将 k 存储在内核态中, 只有内核态代码才能安全地访问, 保证 k 的值与地址不暴露在内核态以外的地方, 如图 6 所示, 实现可信计算基信息 k 的安全存储与访问.

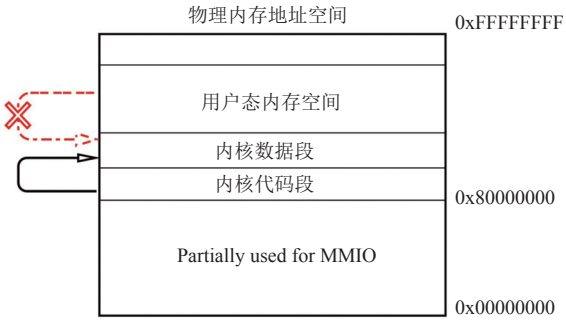

图 6 物理内存地址空间的访问

(2) 用户自定义信息 c 的生成

用户自定义信息 c 用来指示用户相关信息, 结合流密码密钥生成的随机性, 实现时将用户自定义信息种子与随机密钥生成函数相结合, 生成应用程序独有且随机的信息 c, 以此保证 c 的安全性与随机性.

(3) 数据安全存储操作过程的安全性.

RISC-V 架构有一个名为 ecall 的系统指令, 可以发出系统调用请求, 以进入特权模式获取更高的访问权限. 执行 ecall 后, 相关的陷阱处理函数会处理该请求, 执行相关操作, 处理结束后, 恢复执行 ecall 前的上下文并回到最初的系统模式. 因此, 实现时为了保证指令操作过程的安全性, 将该过程封装为系统调用, 由用户应用程序发出请求后, 在内核态执行, 可以实现指令操作的机密性, 保证其不对用户可见.

针对上述解决方案, 实现自定义安全指令功能主要通过扩展 Spike 模拟器和代理内核 PK.

Spike 是一个指令集架构 (ISA) 模拟器, 在主机环境中执行 RISC-V 可执行和可链接的二进制文件 (ELF).

它通过模拟 RISC-V 内核的功能来实现指令的获取、译码和执行. RISC-V 程序的编译及模拟执行过程如图 7 所示. Spike 可扩展性很强, 对于实现自定义指令的扩展较为方便. Spike 实现自定义指令的支持需要修改 riscv-tools/riscv-isa-sim 目录下 disasm.cc, encoding.h 和 riscv.mk.in 文件. disasm.cc 文件中定义了新指令; enchoding.h 自定义指令的 MATCH, MASK 和 DECLARE_INSN 宏, 这与修改的 RISC-V Binutils 的头文件信息相同; 指令名称添加到 riscv.mk.in 文件中的 riscv_insn_ext_i 列表中, 以生成自定义指令功能行为的源代码. 并且需要将自定义 TSTORE 指令的功能行为添加到 riscv/insns/tstore.h 中, 示例代码 2 简单描述功能实现.

示例代码2. Spike中TSTORE指令功能的实现

```
require_rv64;
uint64_t src1 = MMU.load_int64 (rs1);
uint64_t src2 = MMU.load_int64 (rs2);
uint64_t xor1 = src1 ^ src2;
uint64_t extRS = MMU.load_int64 (0x3ffffffb18);
uint64_t xor2 = xor1 ^ extRS;
WRITE_RD (sext_xlen(xor2));
```

图 7 RISC-V 程序的编译及模拟执行过程

PK (proxy kernel and boot loader) 是 RISC-V 代理内核, 是一个轻量级应用程序执行环境, 可以承载静态链接的 RISC-V 可执行二进制文件, 可以进行内存管理、处理中断与异常等, 可以看作可在 Spike 上运行的小型操作系统. 因此, 本文在 PK 中封装系统调用, 实现数据安全存储操作, 示例代码 3 描述 PK 中系统调用的实现.

完整的执行流程如图 8 所示. 在用户态程序中执行 ecall 指令, 发出系统调用请求并传入明文信息 x 和用户自定义信息种子 c, 进入内核态后获取到更高的访问权限, 内核态的陷阱处理函数 trap_handler() 根据系统调用号查询相应的处理函数 do_tstore(c, x). 该处理函数中先将用户自定义信息种子 c 传入随机密钥生成函数 random() 中, 生成用户自定义信息, 将随机密钥生成函数封装在系统调用中, 既可以保证 c 的值仅由自身程序信息随机生成, 又可保证该值不被窃取. 接着以内敛汇编的方式执行 TSTORE 指令, 最终将密文信息 y 返回给用户态程序. 该过程能够保证可信基信息 k 的安全性以及保证操作过程不对用户可见.

示例代码3. PK中系统调用的实现

```
uint64 sys_user_tstore(uint64 a, uint64 b)
{
uint64 c;
   b=rand(b);
   asm volatile (
         "tstore %[z], %[y], %[x]\n\t"
               : [z] "=r" (c)
         : [x] "r" (&a), [y] "r" (&b)
           );
  return c;
}
```

2.3 实验结果分析

基于上述编译器支持和模拟器功能实现, 应用程序可在用户态通过系统调用执行数据安全存储操作, 应用程序与输出结果如示例代码 4 所示. 从运行结果可知, 在应用程序中, 只需将需要安全存储的明文信息与用户自定义信息种子传入封装好的函数, 即可返回经混淆的密文信息, 该函数完成权限切换, 发起系统调用. 若将密文信息和相同的用户自定义信息种子重新传入该函数, 则可完成解密操作, 返回明文信息.

示例代码4. 数据安全存储应用程序

```
long long x = 28; //明文信息
long long c = 35; //用户自定义信息种子
long long y = get_tstore(x, c); //加密操作
printu("The \"tstore\" encryption is: %lld \n", y);
x = get_tstore(y, c); //解密操作
printu("The \"tstore\" decryption is: %lld \n", x);
输出: The tstore encryption is: 19
The tstore decryption is: 28
```

3 安全性分析

基于可信基的数据安全存储功能, 是借鉴流密码的实现逻辑, 其关键是如何构建密钥. 考虑到流密码机制中固定不变密钥对安全性的有限性, 所引入的用户信息的变化可以构造复合型的流密码密钥 $k \oplus c$, 从而在一定程度上保证了数据混淆加密的安全性, 实现目的地址中存储的数据都是安全混淆后的数据, 而非明文直接存储. 且混淆后的数据与明文、可信基、用户信息均相关, 从而实现该加密存储机制的人、机可信信息的隔离, 只有保证 k 和 c 两个因子均正确的情况下, 才能对所存储数据含义进行有效获取. 如示例代码

4 的解密操作所示, 只有同一个用户在同一平台下, 使用相同的 k 和 c, 才能有效解密. 若非同一用户 (不能保证 c 相同) 或非同一平台 (不能保证 k 相同), 便不能有效解密, 如示例代码 5 所示.

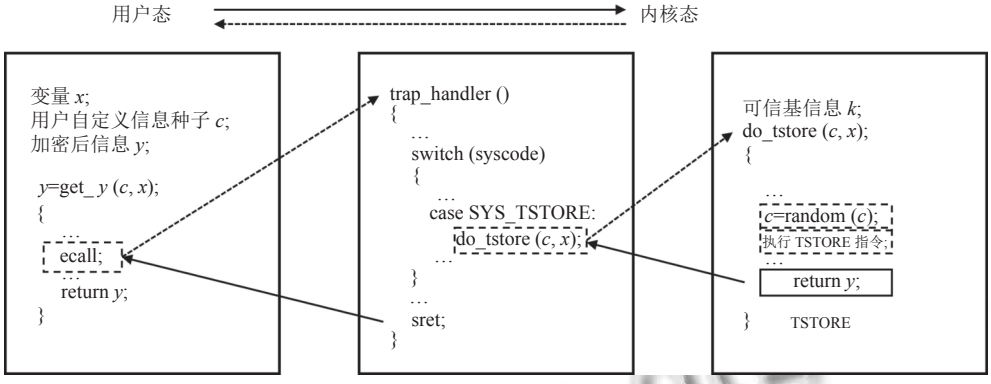

图 8 数据安全存储的完整执行流程

示例代码5. 错误解密示例

```
long long c = 78; //用户自定义信息种子
long long x;
x=get_tstore(y,c);//y为已加密存储的数据,c为该用户自定义信息种子
printu("The \"tstore\" result is: %lld \n", c);
输出: The tstore result is 113 (解密错误)
```

在指令实现层面, 对 k 的安全存储及指令的隐式获取进一步增加了安全性. k 安全地存放在内核或其他可信存储区域中, 用户态应用程序不可轻易获取, 指令隐式地获取 k, 不需要作为参数传递, 避免了 k 地址的暴露. 同时, 该指令的操作过程在内核中实现, 不对用户可见. 为了证明数据安全存储指令操作的安全性, 直接在应用程序中通过内联汇编方式调用 TSTORE 指令, 应用程序与输出结果如示例代码 6 所示, 应用程序运行出错, 不可执行. 其原因是可信基信息 k 存储在内核中, 不可直接在应用程序中通过 TSTORE 指令访问. 由此可证明, TSTORE 安全指令功能的实现可以保证可信基信息 k 的安全存储与访问, 也可保证指令操作的机密性, 不对用户态程序可见.

示例代码6. 数据安全存储应用程序错误示例

```
long long a = 28; //明文信息
long long b = 35; //用户自定义信息种子
long long c;
asm volatile (
"tstore %[z], %[y], %[x]\n\t"
      : [z] "=r" (c)
      : [x] "r" ( a), [y] "r" ( b)
 );
printu("The \"tstore\" result is: %lld \n", c);
输出: page fault: can't get trusted base information
```

另外, 进一步结合对输入明文数据长度的分组化处理, 并结合高层应用或协议中对用户信息的处理, 从而进一步完善该存储机制的安全性.

4 总结与展望

本文针对 RISC-V 自定义指令的安全能力, 研究了 RISC-V 自定义指令安全扩展方案, 设计了简单高效的 TSTORE 指令, 实现了基于可信基的数据安全存储功能, 在密钥构建与密钥存储方面保证操作的安全性. 在 RISC-V GNU 工具链实现自定义安全指令编译层面的支持, 并成功在模拟器上测试应用对自定义指令的调用执行. 未来的工作将继续完善 TSTORE 安全指令的功能, 结合内存保护机制以实现更安全的数据存储, 并考虑在真实硬件环境中测试 TSTORE 指令, 测试其在真实硬件环境中的性能.

参考文献

1 刘畅, 武延军, 吴敬征, 等. RISC-V 指令集架构研究综述. 软件学报, 2021, 32(12): 3992–4024. [doi: 10.13328/j.cnki.jos.006490]

2 Bolat A, Cassano L, Reviriego P, *et al*. A microprocessor protection architecture against hardware Trojans in memories. Proceedings of the 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS). Marrakech: IEEE, 2020. 1–6.

3 Lu T. A survey on RISC-V security: Hardware and architecture. arXiv:2107.04175, 2021.

4 陈锐, 李冰, 刘向东. 基于 RISC-V 指令扩展的低开销 SM4 算法设计与实现. 电子器件, 2021, 44(1): 108–113.

[doi: 10.3969/j.issn.1005-9490.2021.01.021]

5 Altınay Ö, Örs B. Instruction extension of RV32I and GCC back end for ascon lightweight cryptography algorithm. Proceedings of 2021 IEEE International Conference on Omni-layer Intelligent Systems (COINS). Barcelona: IEEE, 2021. 1–6.

6 Marshall B, Newell GR, Page D, *et al*. The design of scalar AES instruction set extensions for RISC-V. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2021(1): 109–136.

7 Hepp A, Sigl G. Tapeout of a RISC-V crypto chip with hardware Trojans: A case-study on Trojan design and pre-silicon detectability. Proceedings of the 18th ACM International Conference on Computing Frontiers. Online: ACM, 2021. 213–220.

8 Saarinen MJO. SNEIK on microcontrollers: AVR, ARMv7-M, and RISC-V with custom instructions. IACR Cryptology ePrint Archive, 2019: 936.

9 Yu JY, Hsiung L, El Hajj M, *et al*. Data oblivious ISA extensions for side channel-resistant and high performance computing. Cryptology ePrint Archive, 2018.

10 Kim H, Lee J, Pratama D, *et al*. RIMI: Instruction-level memory isolation for embedded systems on RISC-V. Proceedings of 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD). San Diego: IEEE, 2020. 1–9.

11 邓良, 曾庆凯. 引入内可信基的应用程序保护方法. 软件学报, 2016, 27(4): 1042–1058. [doi: 10.13328/j.cnki.jos.005016]

12 Lee D, Kohlbrenner D, Shinde S, *et al*. Keystone: An open framework for architecting trusted execution environments. Proceedings of the 15th European Conference on Computer Systems. Heraklion: ACM, 2020. 38.

13 赵石磊, 刘玲, 黄海, 等. 流密码算法、架构与硬件实现研究. 密码学报, 2021, 8(6): 1039–1057. [doi: 10.13868/j.cnki.jcr.000495]

14 Waterman A, Lee Y, Avizienis R, *et al*. The RISC-V instruction set. Proceedings of 2013 IEEE Hot Chips 25 Symposium (HCS). Stanford: IEEE, 2013. 1.

15 王鹏, 陈影, 邢明杰. 基于 LLVM 的 RISC-V 自定义扩展指令支持方法. 计算机系统应用, 2021, 30(11): 20–26. [doi: 10.15888/j.cnki.csa.008347]

16 唐俊龙, 禹智文, 刘远治, 等. 面向 RISC-V 处理器的 GCC 移植与优化. 计算机应用与软件, 2021, 38(9): 262–267, 285. [doi: 10.3969/j.issn.1000-386x.2021.09.041]

(校对责编: 孙君艳)