研究计划书

曲直

一、研究背景

数据的**隐私性保护**是当前数据应用领域面临的关键挑战。传统的加解密技术需要固化受保护的目标对象，并会对数据访问（尤其是频繁的数据访问）效率产生明显的影响。针对该问题，我建立了一种**软硬件协同**的**弹性数据隐私保护系统**。该系统根据**代码执行逻辑**和**数据访问逻辑**，利用 **Intel MPK 技术**动态地对隐私数据进行保护。数据的访问权限会在**时间域**和**代码域**上**动态变化**，进而实现对敏感数据的实时保护。

二、主要研究内容

2.1 研究框架

主要研究内容框架如图 1 所示：

该系统分为两大模块：**域处理模块**和**密钥分配模块**，在域处理模块中，系统输入文件源码或二进制代码，在进行数据域捕捉和敏感数据标定后，将**敏感数据**（如用户自定义敏感数据、密钥数据、私有 Token）打上**标签**，送入下一个模块----**密钥分配模块：**

密钥分配模块支持 x86 和 ARM 两种架构，在 x86 架构下，使用 **Intel Memory Protection Keys（MPK）**技术，修改**页表**的 **49-52 位**，利用 PKRU 寄存器，实现页表的访问控制。在 ARM 架构下，使用 ARM Memory Tagging Extension 技术，启用 Top-byte Ignore，将页表的最高四位和密钥绑定，实现对页表的访问控制。

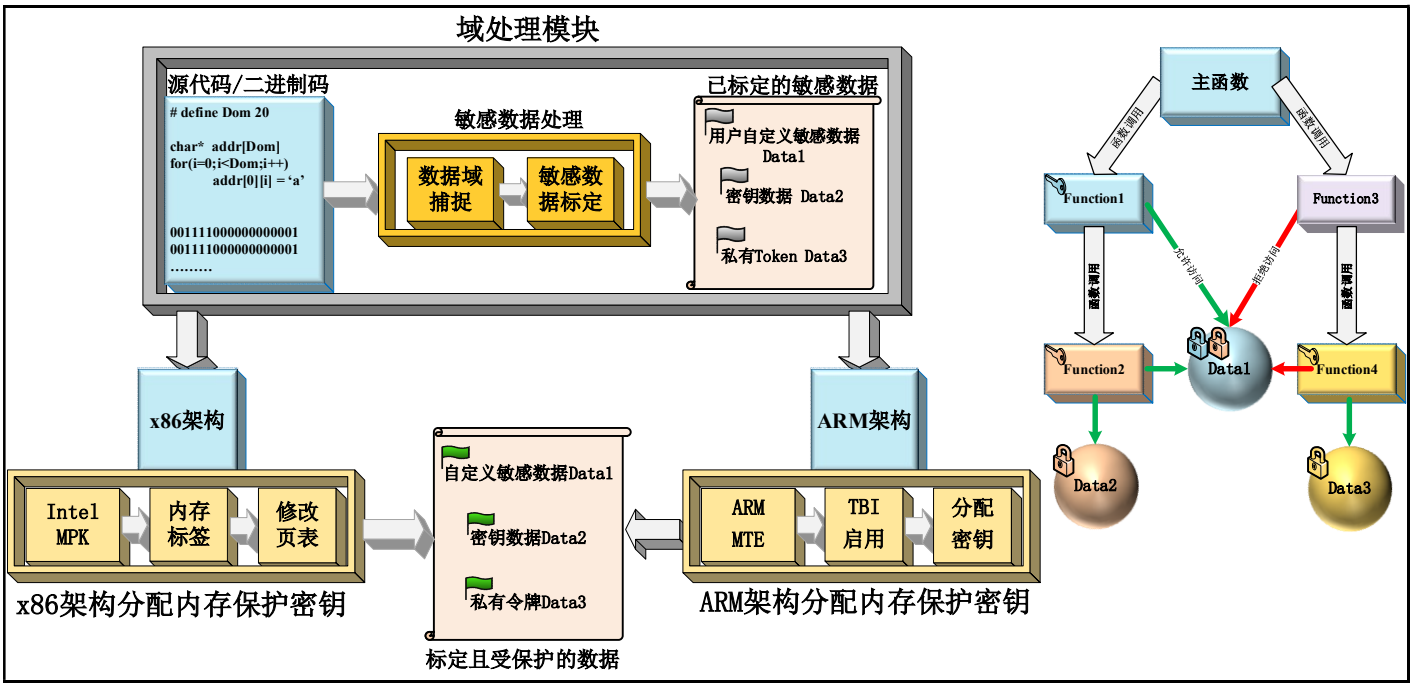

图 1 研究框架

在运行完上述两个模块后，如图 1 右侧函数调用图所示，只有**拥有内存数据密钥**的函数，才可以访问对应的数据。如 Function1 拥有蓝色密钥，则可以访问被蓝色锁加密的 Data1，同理拥有橙色密钥的 Function2，可以访问 Data1 和 Data2，但是，由于 Function3 缺少蓝色密钥，所以其对 Data1 的访问将会被拒绝。

2.2 访问权限动态改变

当源代码或二进制代码中的数据 B，映射到物理内存，将其所在的页表标记，记为 B 数据（T1，Domain1）。此时，由于经过了**敏感数据识别模块**和**密钥分配模块**的处理，该内存页（0xC9000）已经与 pkey=8 密钥绑定。此时，通过查阅 PKRU 寄存器，可知 pkey=8 对应的权限位**（拒绝写，拒绝读）**为（0，0），即可读、可写。此时，拥有 pkey=8 这把密钥的 Function_B()，对这块内存拥有读、写权限。

当系统状态发生**时间或代码域**上的**变化**时：

（1）当时间前移，代码域不变时：系统状态表示为（T2，Domain1）。此时，由于执行了 pkey_exit()函数，pkey=8 对应的 PKRU 寄存器权限为**（1，1）**，表示**拒绝读、拒绝写**。这时，即使 Function_B()拥有 B 数据所在内存的访问密钥，也无法对该块数据进行读写操作。

（2）当时间不变，代码域改变时：A 数据存放到内存中的另一个页中（0x3CA000）此时，拥有访问密钥 pkey=2 的 Function_A()可以正常访问这个内存页，但是 Function_B()由于**缺少密钥**，所以无法访问到这个内存页。

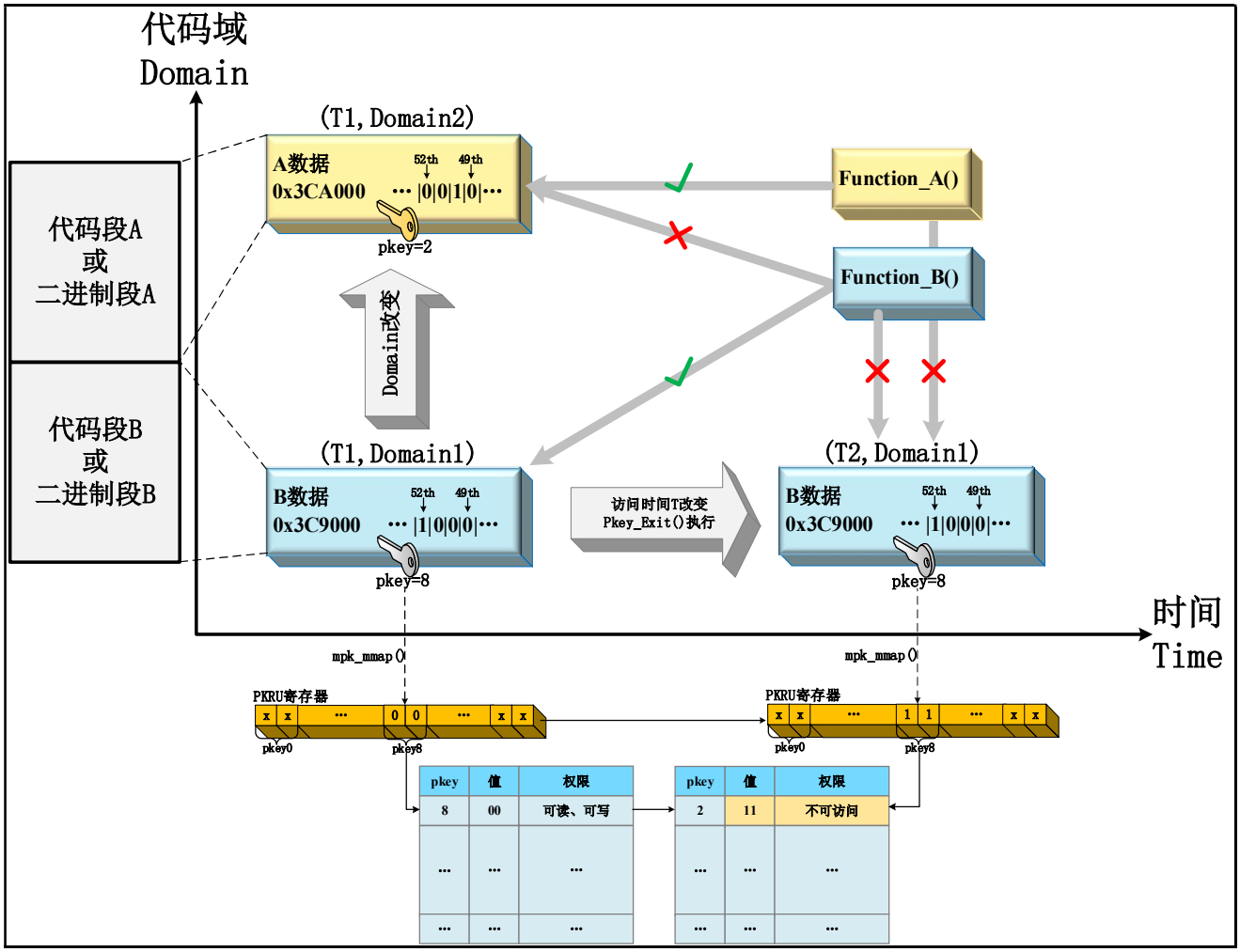

图 2 访问权限在时空二维改变

2.3 研究细节

本小节将更加细致的讲解如何利用 Intel MPK 技术实现对内存页的标记和保护。

首先，如图 3-1 所示，将源代码或二进制代码输入到敏感数据识别模块，识别出需要保护的敏感数据字段。随后经过虚拟地址映射，映射到物理页表。此时，**该内存页不受任何保护**，恶意程序可以任意访问这块未受保护的内存。

随后，在敏感数据对应的内存页中插入标签。由于 LINUX 页表只使用前 48 位，所以 Intel MPK 技术可以对闲置的 **49-52 位进行标记**，利用这 **4 个比特位**，与 PKRU 寄存器的 **16 把 pkey** 对应。

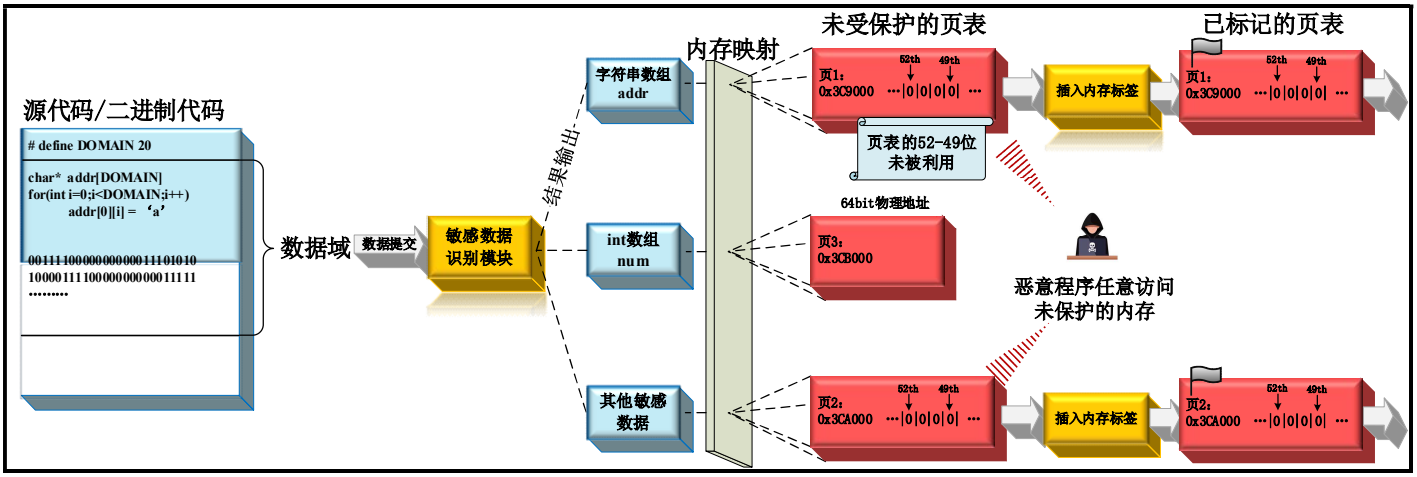

图 3-1 技术细节图-1

在对敏感数据进行标记后，如图 3-2 所示，由 Intel MPK 技术，利用页表的 **52-49 位**，将一把 pkey 分配给待保护的页表。这把 pkey 可以是**只读密钥**（灰色密钥，pkey=1），也可以是**读、写密钥**（黄色密钥）。密钥分配结束后，就得到了一张被标记且受保护的页表（浅绿色方块，pkey=15）。

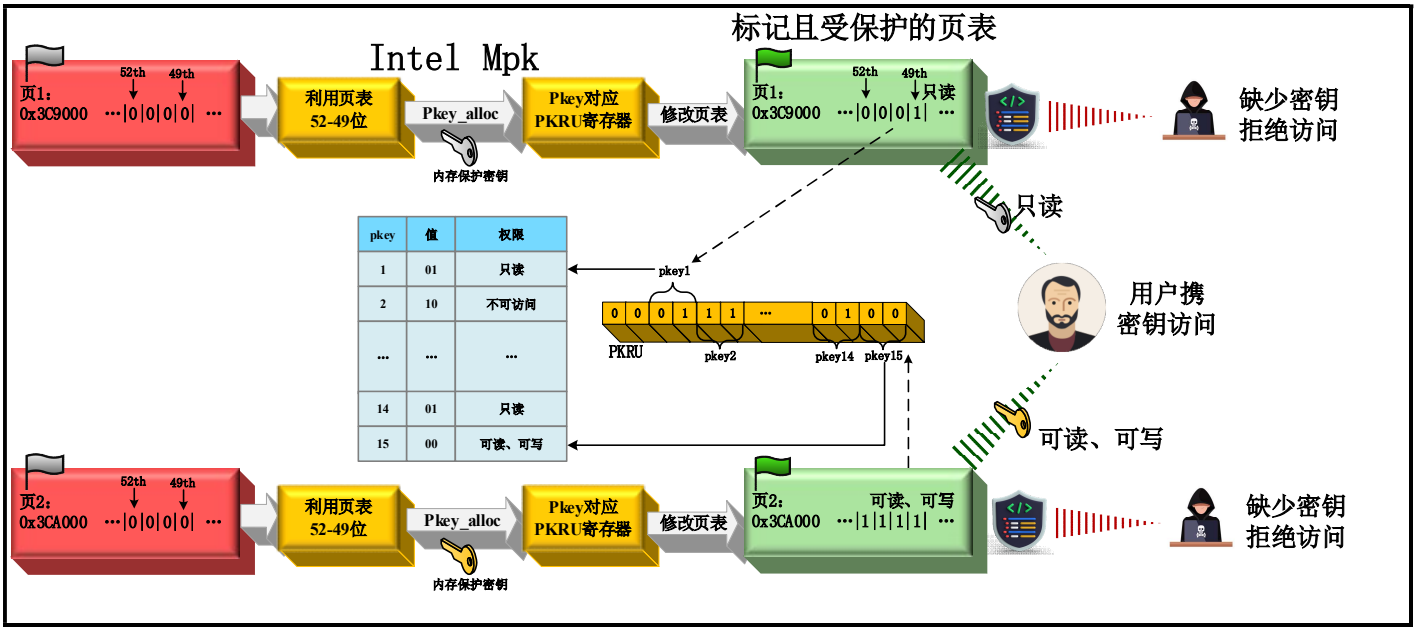

图 3-2 技术细节图-2

此时，假设用户拥有灰色密钥（pkey=1）和黄色密钥（pkey=15），其可以携第一把密钥访问内存页 1，随后系统查询与 pkey=1 对应的 PKRU 寄存器值，得到**标识位（拒绝读，拒绝写）**为**（0，1）**，即**只拥有读**权限。这说明用户对内存页 1（0x3C9000）拥有**读权限**，但**不可写**。同理，当用户携带第二把密钥（黄色密钥，pkey=15）访问内

存页 2 时，系统查询与 pkey=15 对应的 PKRU 寄存器的值，得到**标识位（拒绝读，拒绝写）**为**（0，0）**，即拥有**读、写权限**。这说明用户可以对内存页 2（0x3CA000）**进行读、写操作**。

但是，当恶意用户试图访问内存页 1 或内存页 2 时，由于其**缺少密钥**，系统无法根据其携带的密钥查询 PKRU 寄存器，所以系统会**拒绝**其对内存页 1 和 2 的**访问**。

三、应用

区块链技术是一种**分布式**数据库技术，它以区块的形式将数据链接在一起，并使用加密技术确保安全性和透明性。由于其公开的特点，一旦攻击者将代码注入受害者程序的地址空间，或发现内存泄露漏洞，该地址空间内的所有敏感数据和代码都会遭到盗窃或操纵。所以，在内存级别上对关键隐私数据的保护就显得尤为重要。研究通过 MPK 技术，解决了**数据上链前**的**隐私保护**，将进一步实现**分权域**、**分集群**的**访问控制**。

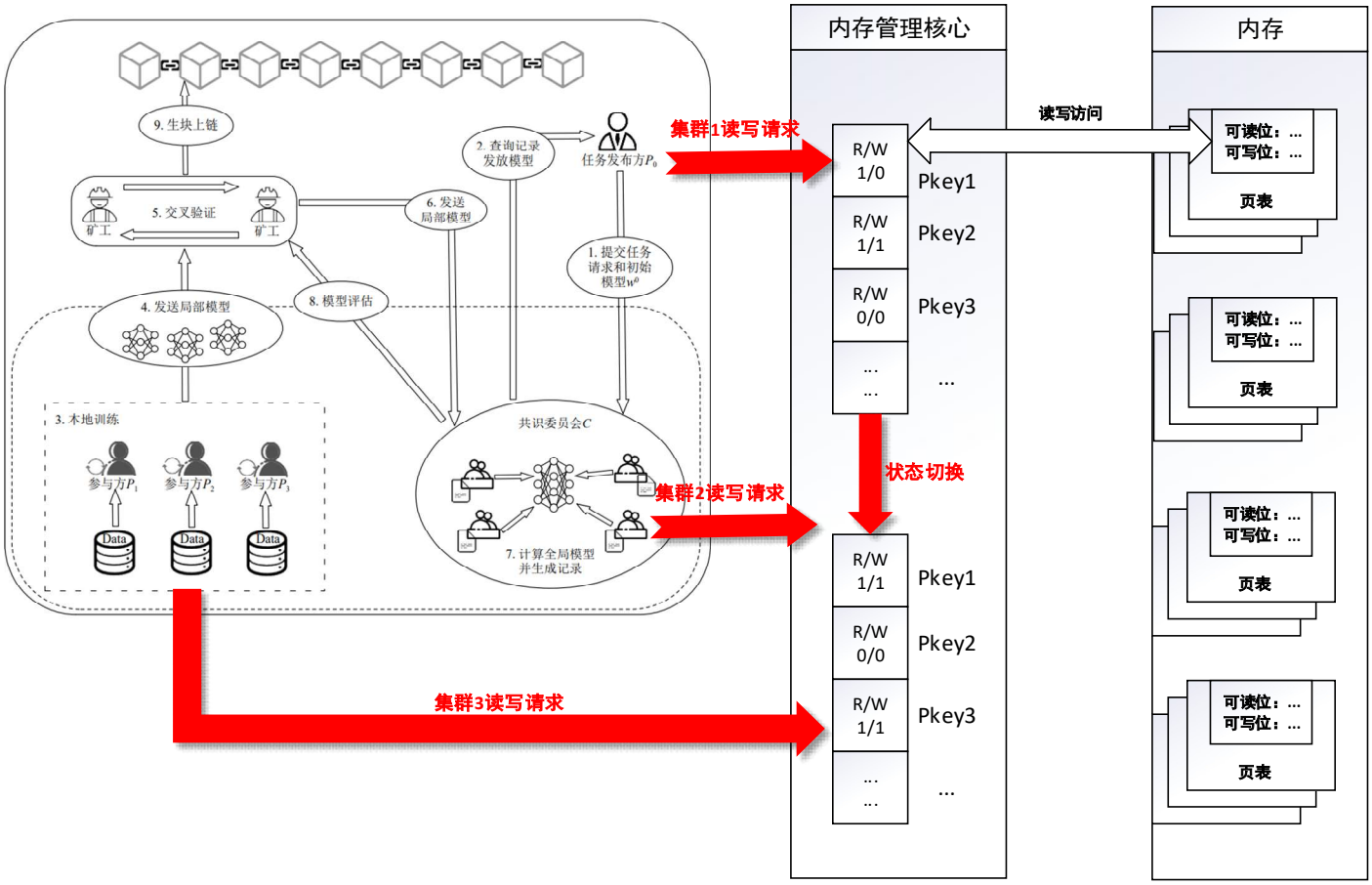

图 4 MPK 保护区块链上链前数据安全

常见的智能合约采用 **JavaScript** 编写，我采用了**内嵌 C 代码**的方式，将智能合约部署过程中的**用户密钥**，进行保护。该过程同样引入了 MPK 技术，解决了用户数据**上链前**的**隐私安全**。

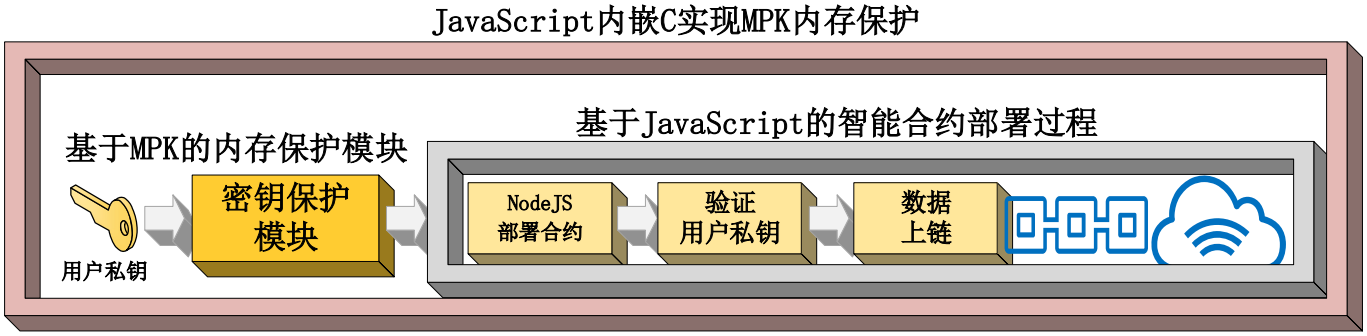

图 5 JavaScript 内嵌 C 语言的区块链隐私保护方案

四、未来计划

未来计划将 MPK 应用于**文件勒索的预防**。当操作系统打开一个文件时，操作系统会赋予文件一个虚拟地址空间，计划使用 MPK 技术，**对文件的虚拟地址空间进行保护**，达到预防文件勒索的效果。