Research Proposal

Zhi Qu

1 Research Background

Data privacy protection is a key challenge in the current data application field. Traditional encryption and decryption technologies require the protected target object to be fixed, which significantly impacts the efficiency of data access, especially with frequent data access. To address this problem, I aim to established a flexible data privacy protection system through software and hardware collaboration. The system utilizes Intel Memory Protection Keys(MPK) technology to dynamically protect private data based on code execution logic and data access logic. Data access rights dynamically change over time and within the code, thereby achieving real-time protection of sensitive data.

2 Main Research Content

2.1 Research Framework

The main research content framework is shown in Figure 1.

The system is divided into two major modules: the domain processing module and the key distribution module.In the domain processing module, the system inputs file source code or binary code and performs data domain capture and sensitive data calibration. Sensitive data (such as user-defined sensitive data, key data, and private tokens) is tagged and sent to the next module.

The key distribution module supports both x86 and ARM architectures. Under the x86 architecture, Intel Memory Protection Keys (MPK) technology is used to modify bits 49-52 of the page table, with the PKRU register implementing access control of the page table. Under the ARM architecture, ARM Memory Tagging Extension technology is used to enable Top-byte Ignore and bind the highest four bits of the page table to the key, achieving access control of the page table.

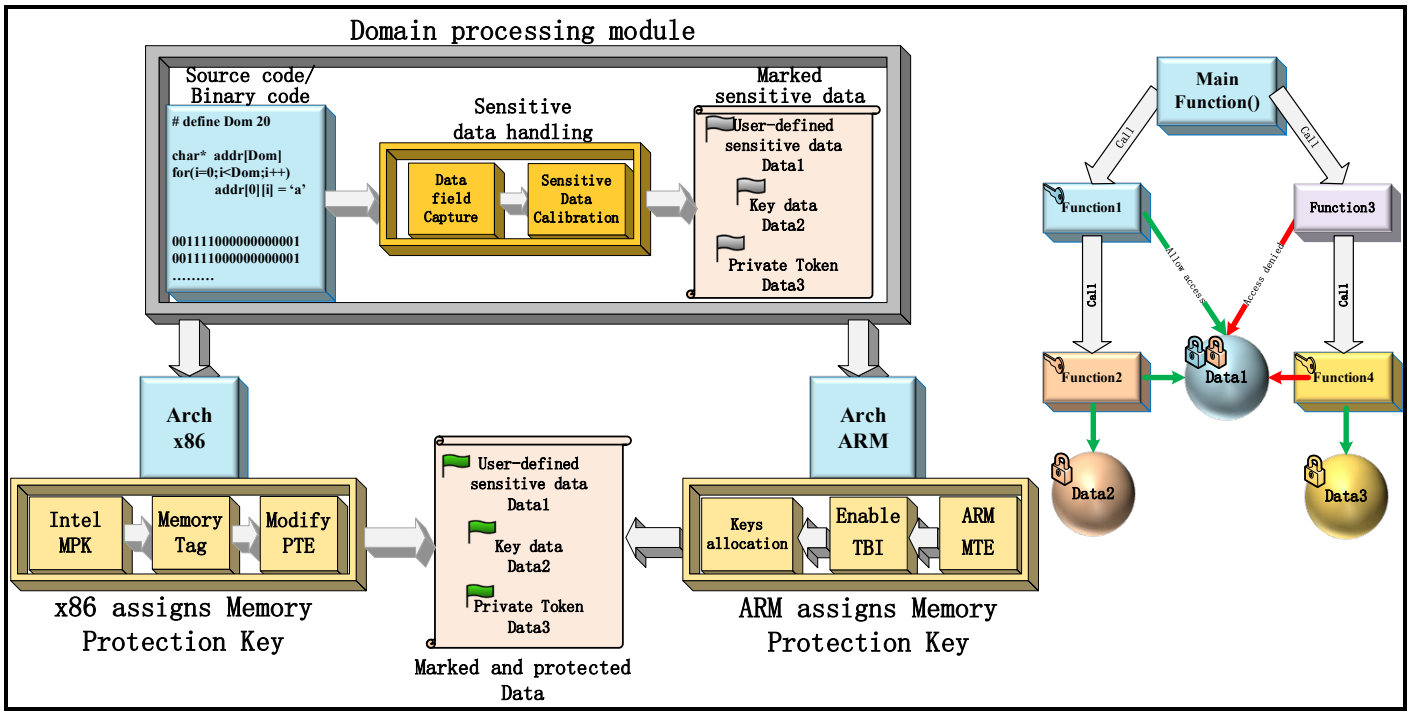

Figure1 Research Framework

After running the above two modules, as shown in the function call diagram on the right side of Figure 1, only functions with the appropriate memory data keys can access the corresponding data. If Function1 has a blue key, it can access Data1 protected by the blue lock. Similarly, Function2, which has an orange key, can access both Data1 and Data2. However, because Function3 lacks the blue key, its access to Data1 will be denied.

2.2 Dynamically changing access permissions

When data B in the source code or binary code is mapped to physical memory, the page table entry for this data is marked as B data (T1, Domain1). Due to the processing by the sensitive data identification module and the key distribution module, the memory page (0xC9000) has been bound to the key pkey=8. By checking the PKRU register, we can see that the permission bits (deny writing, deny reading) corresponding to pkey=8 are (0, 0), indicating that the page is both readable and writable. Therefore, Function_B(), which possesses the key pkey=8, has read and write permissions for this memory.

When the system state changes in the time or code domain:

(1) When time moves forward and the code domain remains unchanged: The system state is expressed as (T2, Domain1). Due to the execution of the pkey_exit() function, the PKRU register permissions corresponding to pkey=8 become (1, 1), indicating that both reading and writing are denied. Even though Function_B() has the access key to the memory where the B data is located, it cannot read or write the data block.

(2) When the time remains unchanged and the code domain changes: Data A is stored in another memory page (0x3CA000). In this scenario, Function_A(), which has access key pkey=2, can access this memory page

normally. However, Function_B() cannot access this memory page due to the lack of the required key.

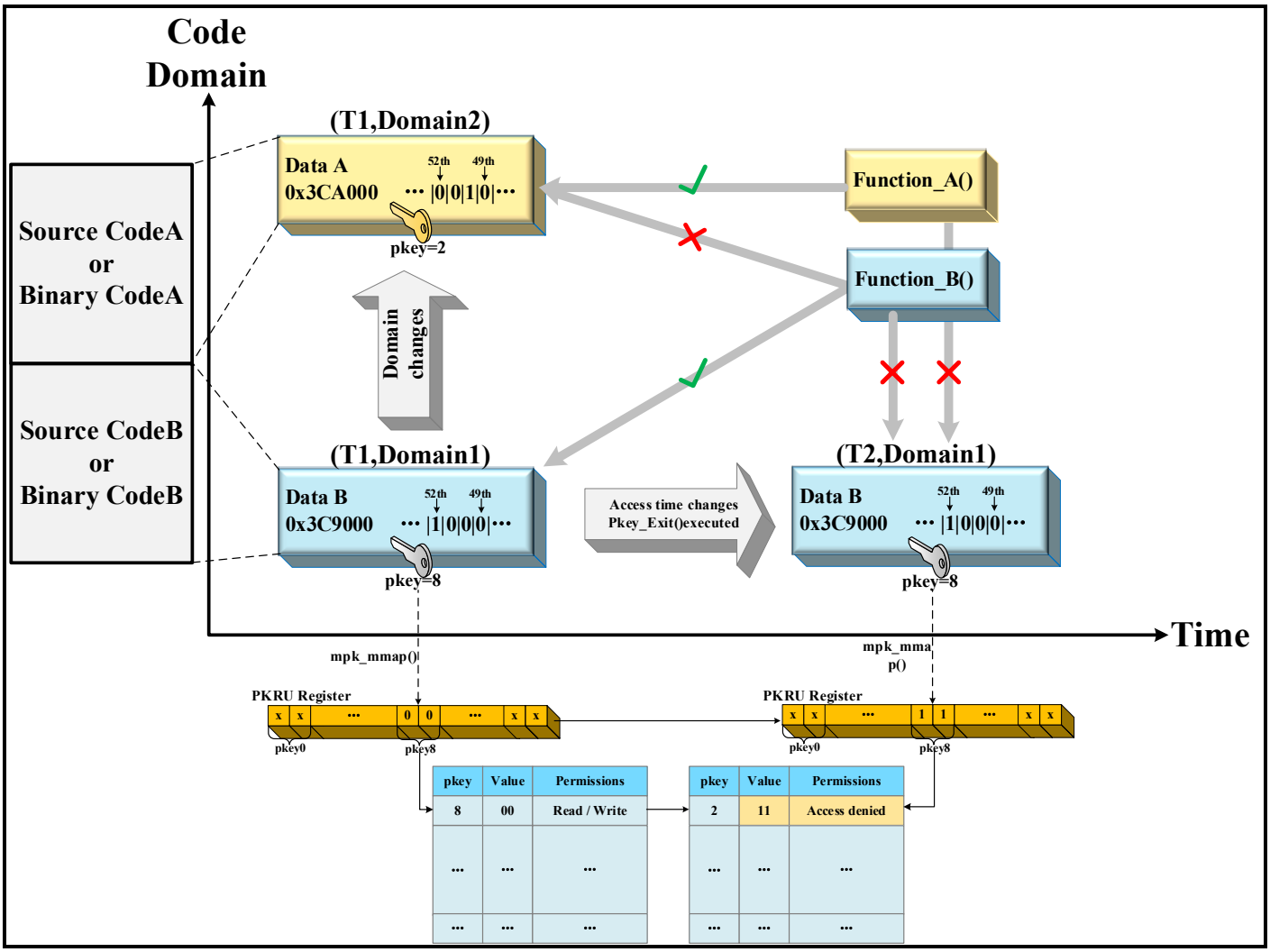

Figure 2 Access permissions change in two dimensions of space and time

2.3 Research Details

This section will explain in more detail how to use MPK technology to mark and protect memory pages.

First, as shown in Figure 3-1, the source code or binary code is entered into the sensitive data identification module to identify the sensitive data fields that need to be protected. These fields are then mapped to the physical page table through virtual address mapping. At this stage, the memory page is unprotected, allowing malicious programs to access it freely.

Next, tags are inserted into the memory pages corresponding to the sensitive data. Since the Linux page table uses only the first 48 bits, Intel MPK technology leverages the idle 49-52 bits, using these 4 bits to correspond to the 16 pkeys of the PKRU register. This marking process effectively protects the memory pages by assigning them to specific keys, which control access permissions.

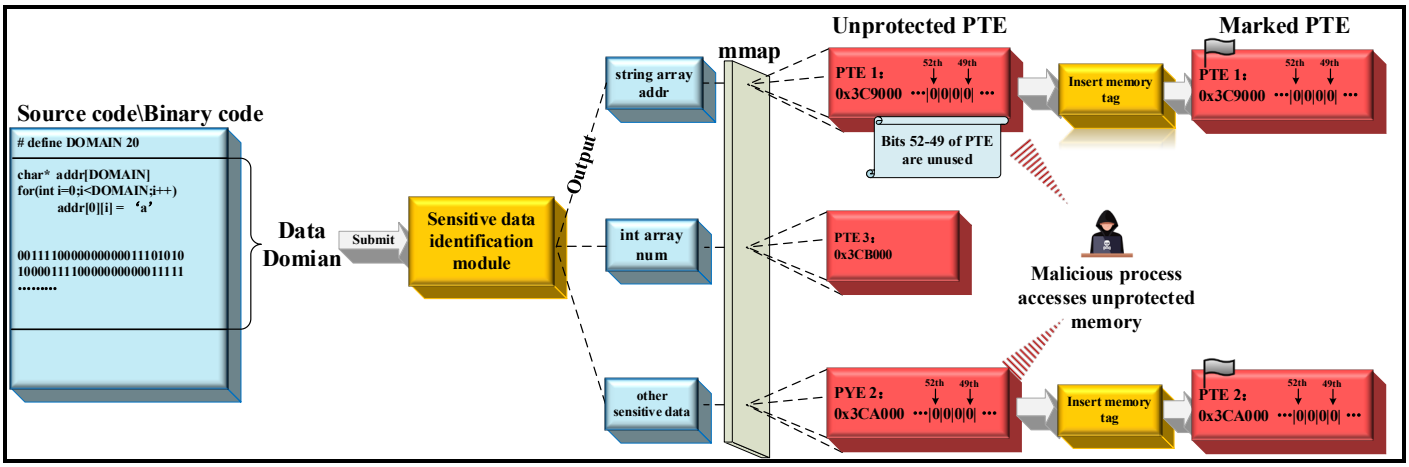

Figure 3-1 technical details-1

After the sensitive data is marked, as shown in Figure 3-2, Intel MPK technology uses bits 49-52 of the page table to allocate a pkey to the page table that needs protection. This pkey can represent different access levels, such as a read-only key (gray key, pkey=1) or a read-write key (yellow key). Once the key distribution is completed, the result is a marked and protected page table (light green square, pkey=15), ensuring that only functions with the appropriate keys can access the sensitive data.

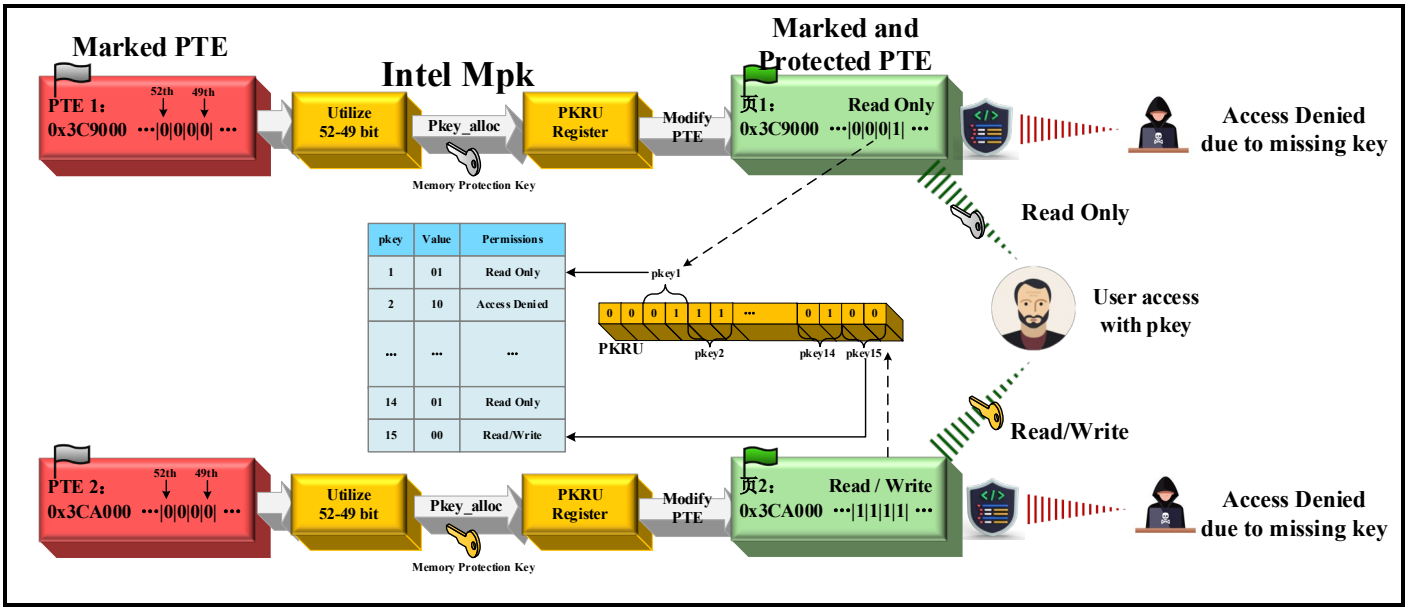

Figure 3-2 technical details-2

At this point, assuming the user has a gray key (pkey=1) and a yellow key (pkey=15), they can access memory page 1 with the gray key. The system queries the PKRU register value corresponding to pkey=1 and retrieves the flag bit (deny reading, deny writing) as (0, 1), indicating that the user has read permission but cannot write to this memory page. This means the user can read memory page 1 (0x3C9000) but cannot write to it.

Similarly, when the user uses the yellow key (pkey=15) to access memory page 2, the system queries the PKRU register value corresponding to pkey=15 and retrieves the flag bit (deny reading, deny writing) as (0, 0), indicating that the user has both read and write permissions. Thus, the user can read and write to memory

page 2 (0x3CA000).

However, when a malicious user attempts to access memory page 1 or memory page 2, they lack the necessary key. Consequently, the system cannot query the PKRU register with the appropriate key, resulting in denied access to both memory pages.

3 Application

Blockchain technology, a distributed database technology, links data in blocks and employs encryption to ensure security and transparency. However, due to its public nature, attackers can exploit vulnerabilities like code injection or memory leaks to compromise sensitive data and code within a victim program's address space. Thus, safeguarding key privacy data at the memory level becomes crucial.

This research addresses this challenge by leveraging MPK technology to protect privacy data before uploading it to the blockchain. Furthermore, it aims to implement access control in decentralized domains and clusters. By integrating MPK technology, the research seeks to fortify data protection mechanisms, enhancing security in blockchain environments.

In the deployment process of common smart contracts typically written in JavaScript, I utilized embedded C code to enhance security by protecting user keys. This integration of embedded C code also incorporates MPK technology, which effectively addresses the privacy concerns associated with user data before it is uploaded to the blockchain. By combining these technologies, the deployment process ensures a higher level of privacy and security for user data within smart contracts.

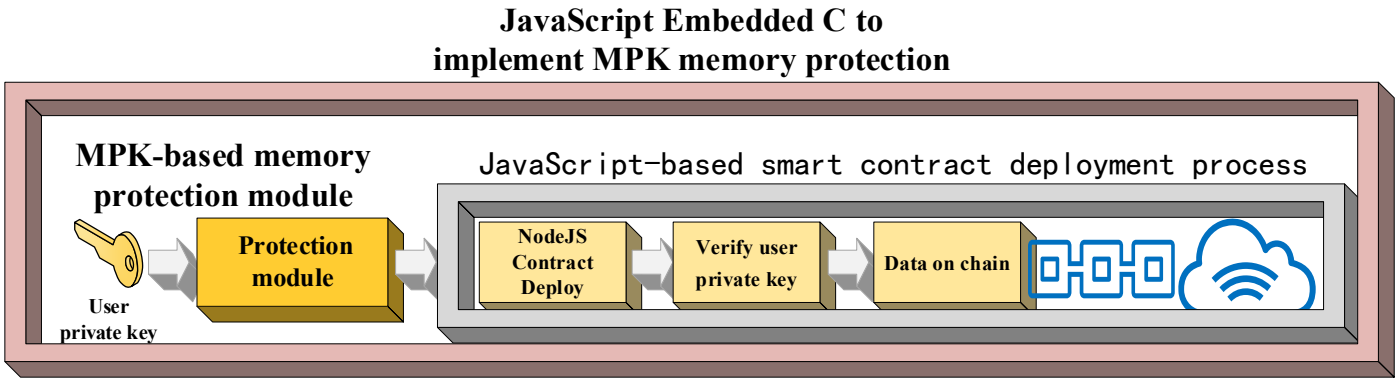

Figure 4 Blockchain privacy protection solution with JavaScript embedded in C language

4 Future Plan

1. The system's flexibility allows for the dynamic adjustment of the open access space domain of data as code is executed. This means that the protected area can expand or shrink in real-time, depending on the code's execution. By integrating Intel MPK technology with this dynamic approach, the system can effectively thwart file ransomware attacks and ensure the security of sensitive data.

2. To prevent file ransomware attacks, MPK technology can be leveraged to protect the virtual address space of files when they are opened by the operating system. Establishing a flexible data privacy protection system with software and hardware collaboration will enhance MPK's capabilities. This system dynamically encrypts and decrypts private data based on code execution and data access logic, providing real-time protection for sensitive data.