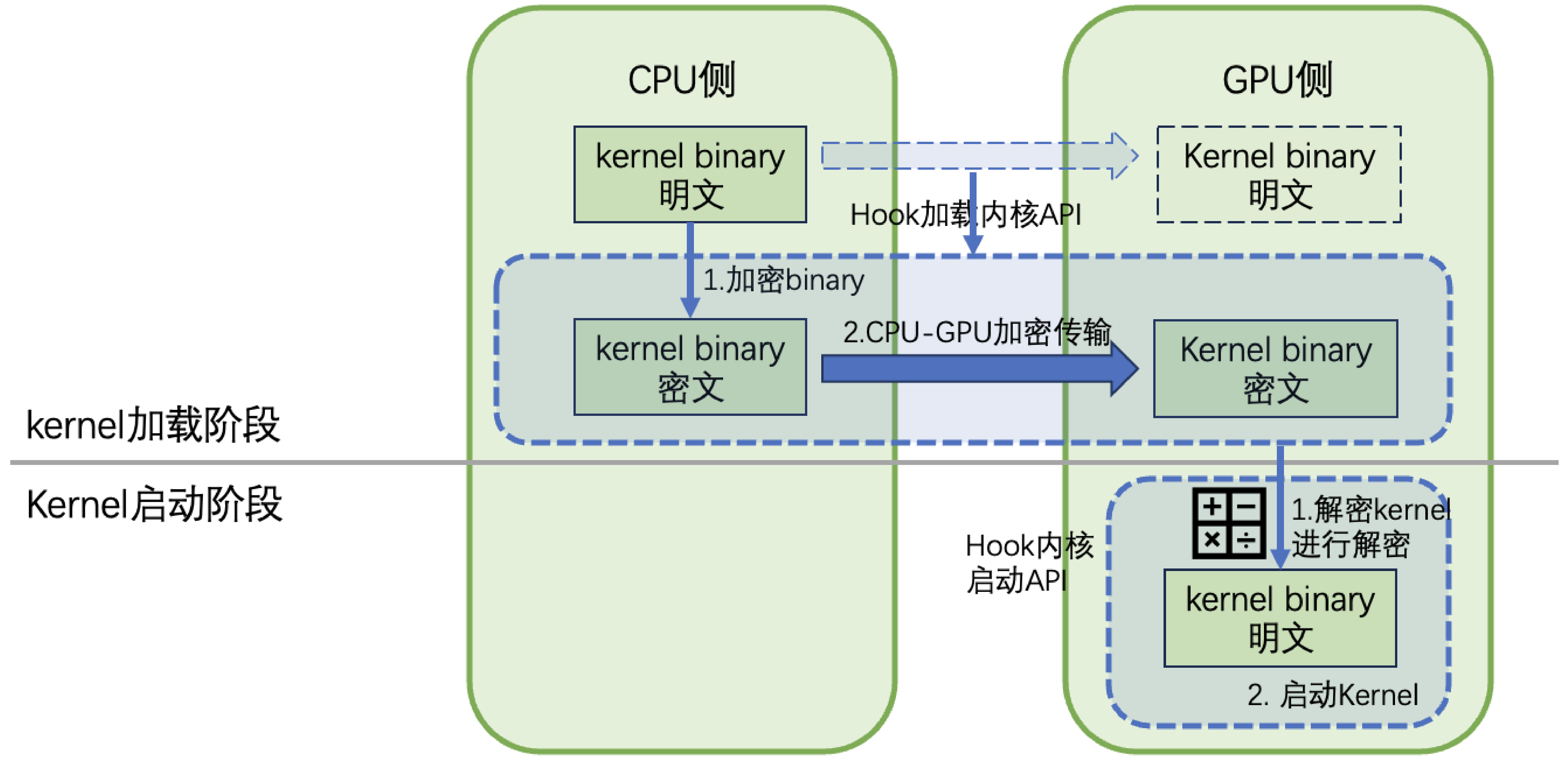
蚂蚁集团
ANT GROUP

密算团队CUDA kernel 加密工作分享

演讲人 吴新月

2025.10

CUDA Kernel加密总览



1. Hook加载内核API，获取明文kernel binary
2. 解析kernel binary布局，找到需要加密的SASS
3. 替换kernel binary中SASS为加密后二进制并加载
4. Hook kernel启动API，即为解密时机
5. 获取GPU侧内核所在位置，读取GPU侧kernel二进制码
6. GPU侧In-place解密kernel二进制

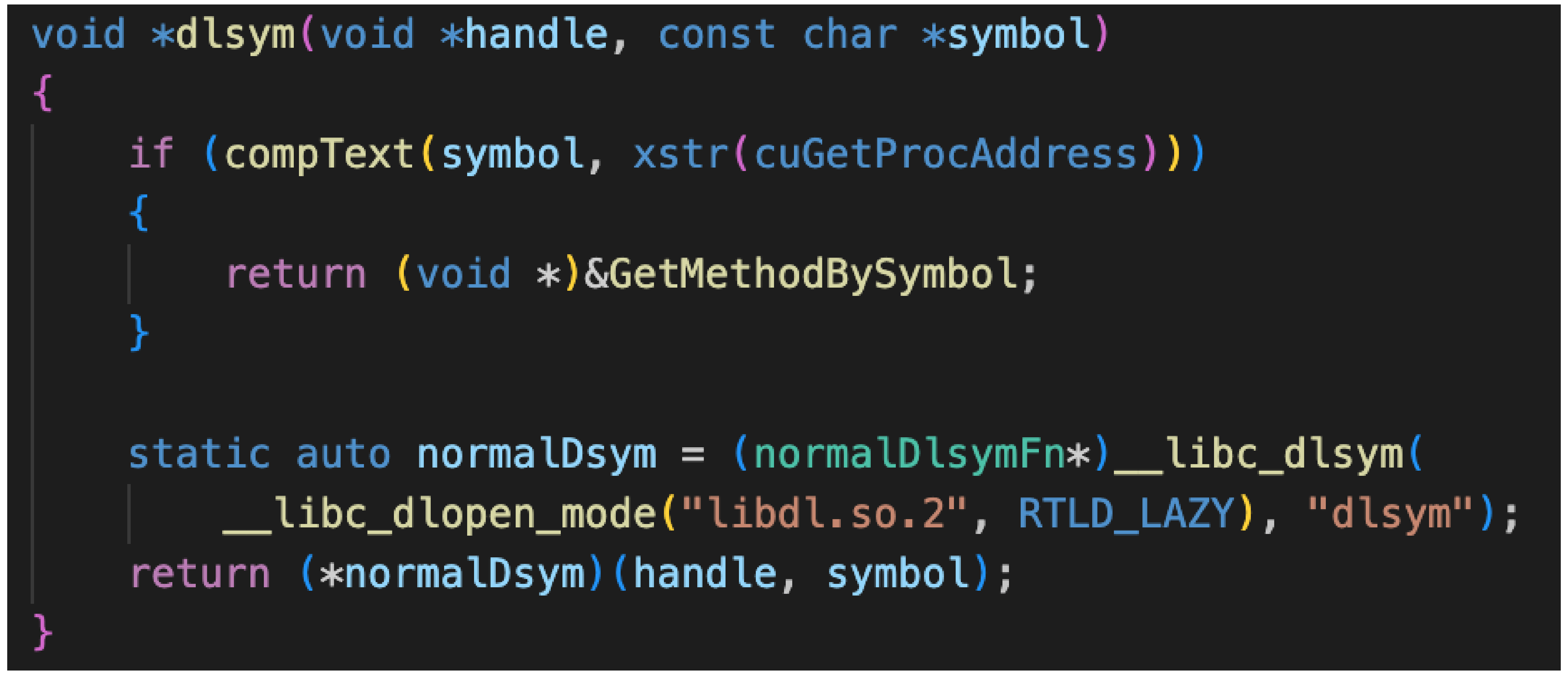
Hook加载内核API

CUDA runtime封装了对kernel加载API的调用，不需要用户显式调用。

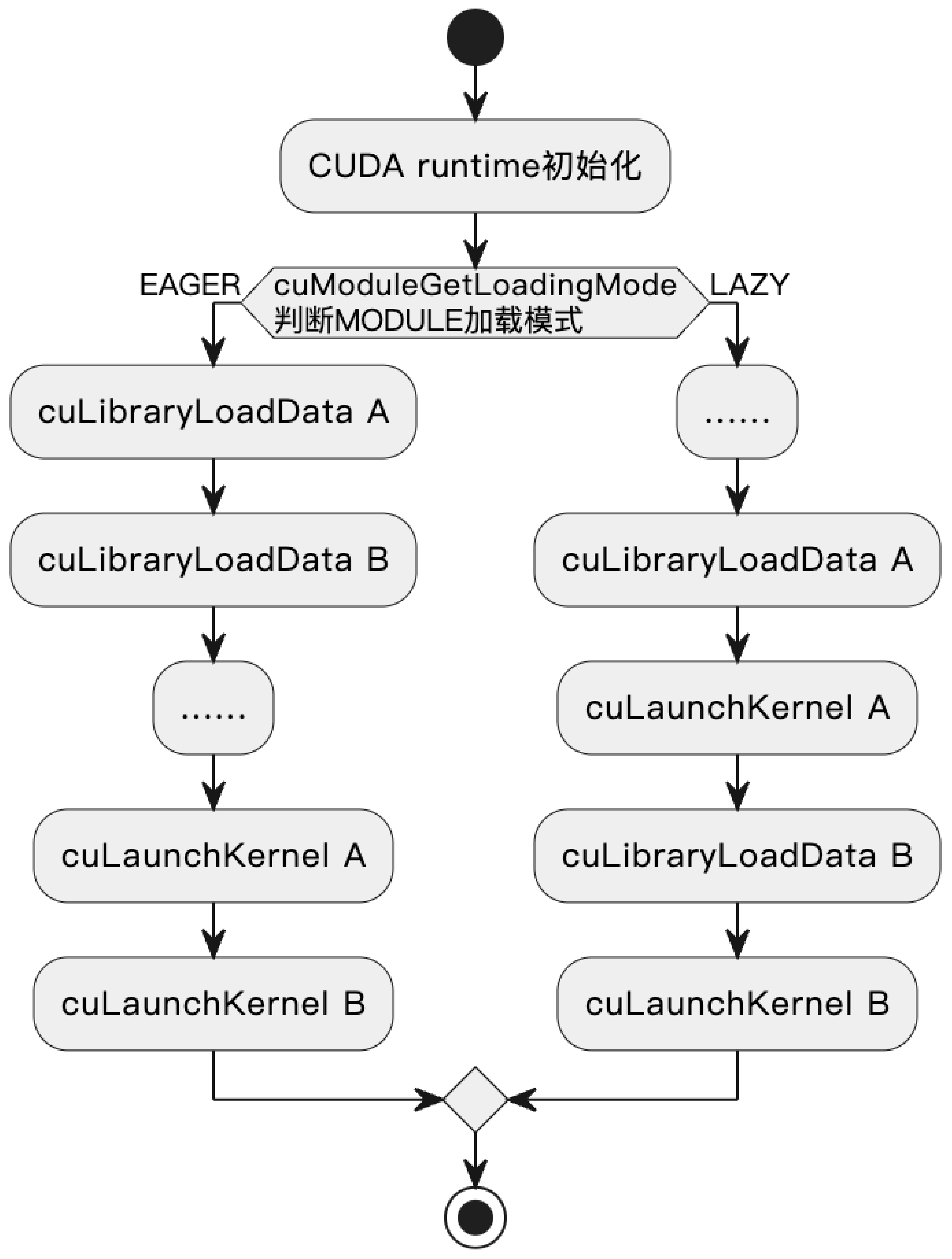
CUDA driver kernel加载系列API：
- cuModuleLoadData
- cuModuleLoad
- cuLibraryLoadData
- cuLibraryLoadFromFile

……

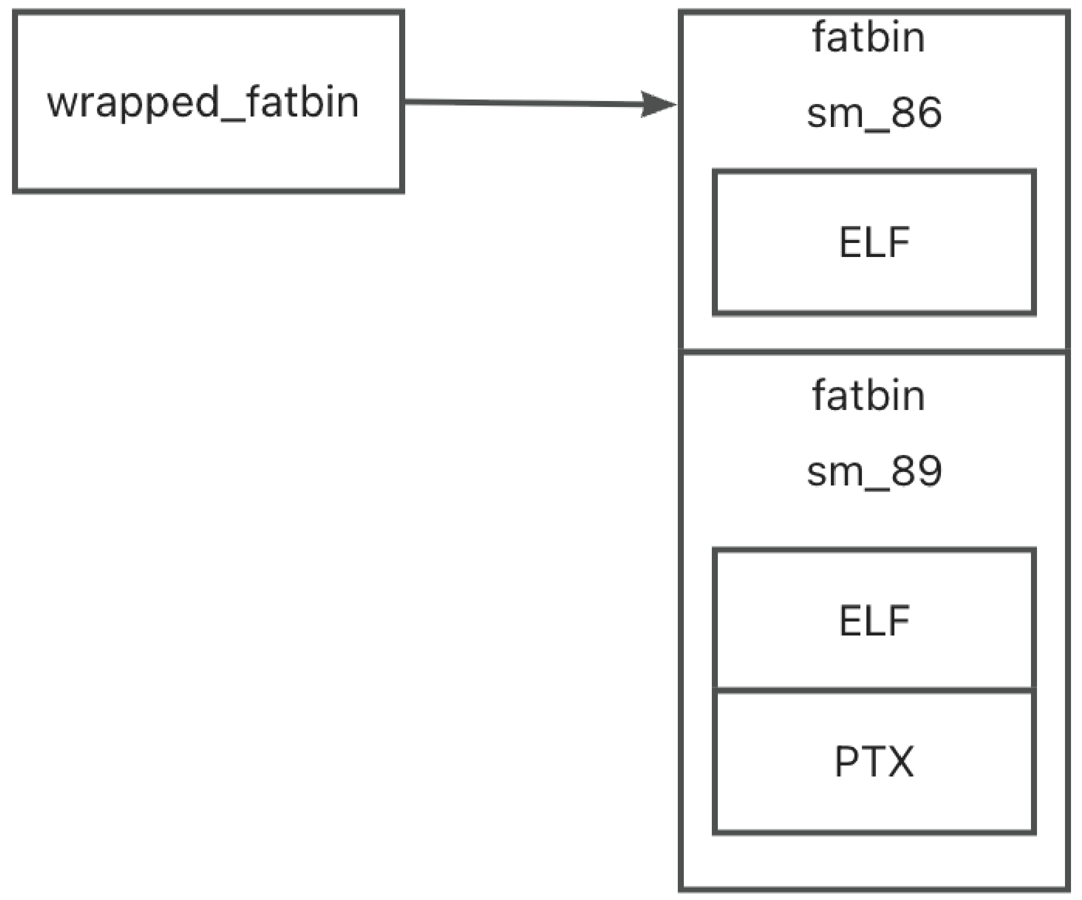
目前POC只实现了cuLibraryLoadData hook。
具体实现方法：CUDA runtime使用cuGetProcAddress方法定位driver API，通过hook该方法可以替换对应driver API。



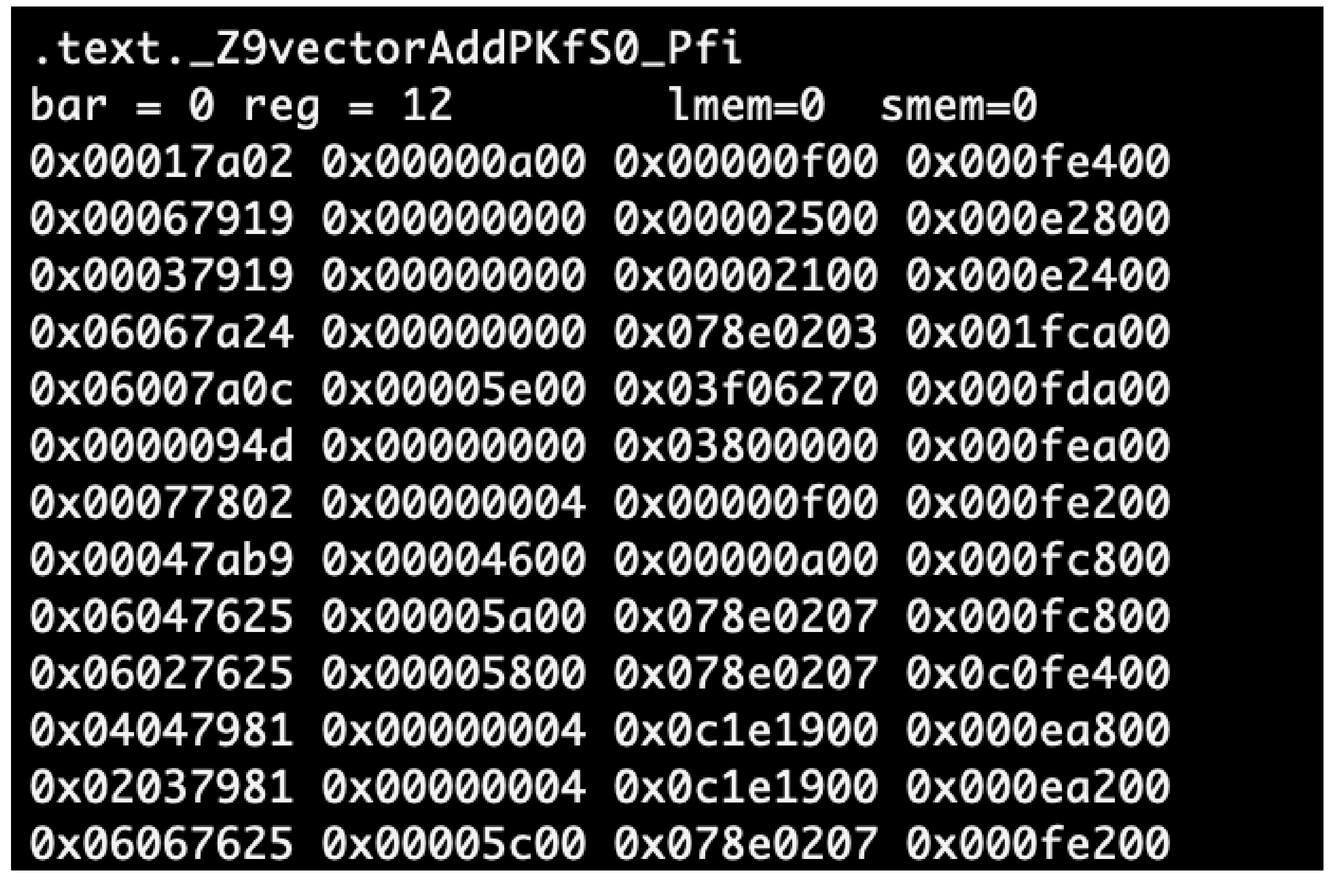
```
void *dlsym(void *handle, const char *symbol)
{
    if (compText(symbol, xstr(cuGetProcAddress)))
    {
        return (void *)&GetMethodBySymbol;
    }

    static auto normalDsym = (normalDlsymFn*)__libc_dlsym(
        __libc_dlopen_mode("libdl.so.2", RTLD_LAZY), "dlsym");
    return (*normalDsym)(handle, symbol);
}
```



CUDA Kernel binary布局



cuLoadModuleData API参数中获取的kernel binary布局。
每个fatbin可以有一个或多个ELF/PTX。



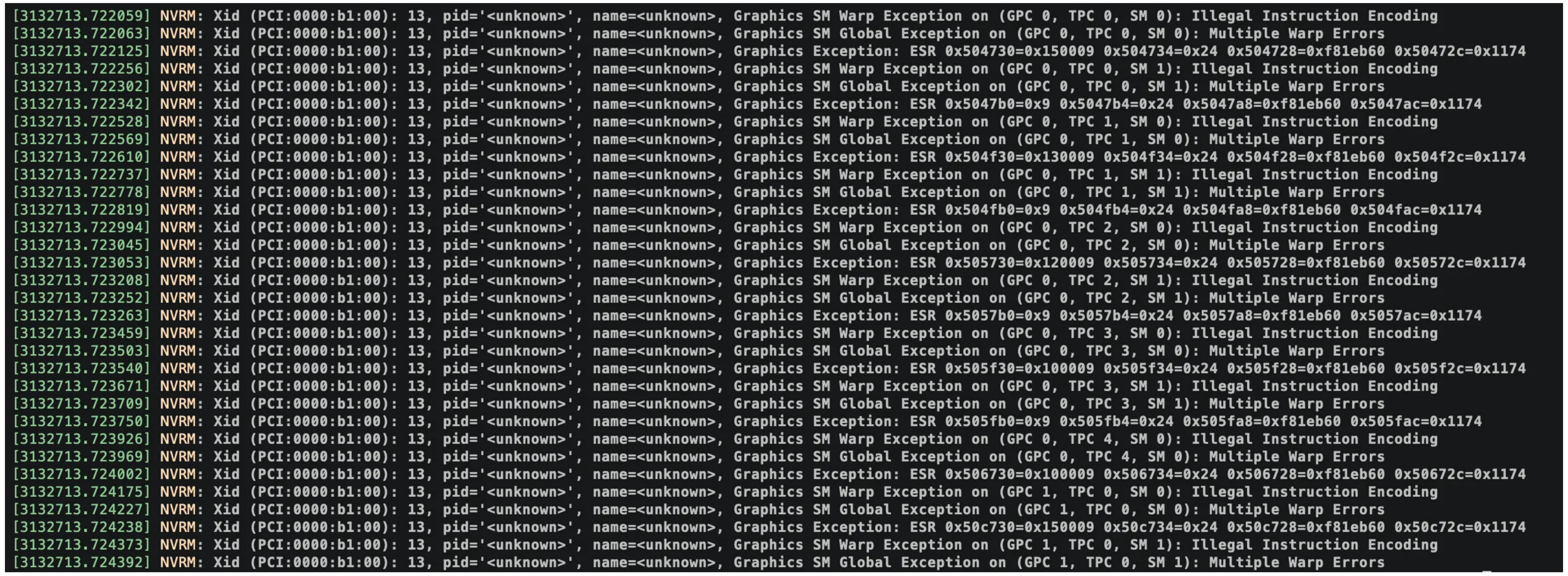
```
.text._Z9vectorAddPKfS0_Pfi
bar = 0 reg = 12          lmem=0   smem=0
0x00017a02 0x00000a00 0x00000f00 0x000fe400
0x00067919 0x00000000 0x00002500 0x000e2800
0x00037919 0x00000000 0x00002100 0x000e2400
0x06067a24 0x00000000 0x078e0203 0x001fca00
0x06007a0c 0x00005e00 0x03f06270 0x000fda00
0x0000094d 0x00000000 0x03800000 0x000fea00
0x00077802 0x00000004 0x00000f00 0x000fe200
0x00047ab9 0x00004600 0x00000a00 0x000fc800
0x06047625 0x00005a00 0x078e0207 0x000fc800
0x06027625 0x00005800 0x078e0207 0x0c0fe400
0x04047981 0x00000004 0x0c1e1900 0x000ea800
0x02037981 0x00000004 0x0c1e1900 0x000ea200
0x06067625 0x00005c00 0x078e0207 0x000fe200
```

SASS编码位于ELF中的.text段。

cuLoadModuleData API的参数指向kernel binary buffer。通过逐层解析header，可以最终定位到目标SASS编码进行加密。

限制：当前方案只能对编译成SASS的cuda kernel进行加密。对于编译成PTX类型的cuda kernel，大概率由CPU侧CUDA driver进行PTX->SASS的翻译，这意味着我们无法在hook driver API时加密PTX code。可能的解决方案为在hook函数中执行nvcc先把PTX翻译为ELF再进行加密。

CPU侧加密SASS后现象



```
[3132713.722059] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 0, SM 0): Illegal Instruction Encoding
[3132713.722063] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 0, SM 0): Multiple Warp Errors
[3132713.722125] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x504730=0x150009 0x504734=0x24 0x504728=0xf81eb60 0x50472c=0x1174
[3132713.722256] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 0, SM 1): Illegal Instruction Encoding
[3132713.722302] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 0, SM 1): Multiple Warp Errors
[3132713.722342] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x5047b0=0x9 0x5047b4=0x24 0x5047a8=0xf81eb60 0x5047ac=0x1174
[3132713.722528] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 1, SM 0): Illegal Instruction Encoding
[3132713.722569] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 1, SM 0): Multiple Warp Errors
[3132713.722610] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x504f30=0x130009 0x504f34=0x24 0x504f28=0xf81eb60 0x504f2c=0x1174
[3132713.722737] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 1, SM 1): Illegal Instruction Encoding
[3132713.722778] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 1, SM 1): Multiple Warp Errors
[3132713.722819] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x504fb0=0x9 0x504fb4=0x24 0x504fa8=0xf81eb60 0x504fac=0x1174
[3132713.722994] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 2, SM 0): Illegal Instruction Encoding
[3132713.723045] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 2, SM 0): Multiple Warp Errors
[3132713.723053] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x505730=0x120009 0x505734=0x24 0x505728=0xf81eb60 0x50572c=0x1174
[3132713.723208] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 2, SM 1): Illegal Instruction Encoding
[3132713.723252] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 2, SM 1): Multiple Warp Errors
[3132713.723263] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x5057b0=0x9 0x5057b4=0x24 0x5057a8=0xf81eb60 0x5057ac=0x1174
[3132713.723459] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 3, SM 0): Illegal Instruction Encoding
[3132713.723503] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 3, SM 0): Multiple Warp Errors
[3132713.723540] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x505f30=0x100009 0x505f34=0x24 0x505f28=0xf81eb60 0x505f2c=0x1174
[3132713.723671] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 3, SM 1): Illegal Instruction Encoding
[3132713.723709] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 3, SM 1): Multiple Warp Errors
[3132713.723750] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x505fb0=0x9 0x505fb4=0x24 0x505fa8=0xf81eb60 0x505fac=0x1174
[3132713.723926] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 0, TPC 4, SM 0): Illegal Instruction Encoding
[3132713.723969] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 0, TPC 4, SM 0): Multiple Warp Errors
[3132713.724002] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x506730=0x100009 0x506734=0x24 0x506728=0xf81eb60 0x50672c=0x1174
[3132713.724175] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 1, TPC 0, SM 0): Illegal Instruction Encoding
[3132713.724227] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 1, TPC 0, SM 0): Multiple Warp Errors
[3132713.724238] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics Exception: ESR 0x50c730=0x150009 0x50c734=0x24 0x50c728=0xf81eb60 0x50c72c=0x1174
[3132713.724373] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Warp Exception on (GPC 1, TPC 0, SM 1): Illegal Instruction Encoding
[3132713.724392] NVRM: Xid (PCI:0000:b1:00): 13, pid='<unknown>', name=<unknown>, Graphics SM Global Exception on (GPC 1, TPC 0, SM 1): Multiple Warp Errors
```

在CPU侧加密（掩码）、GPU侧无解密的情况下，launch kernel后可以看到GPU驱动有illegal instruction报错。说明加密生效。

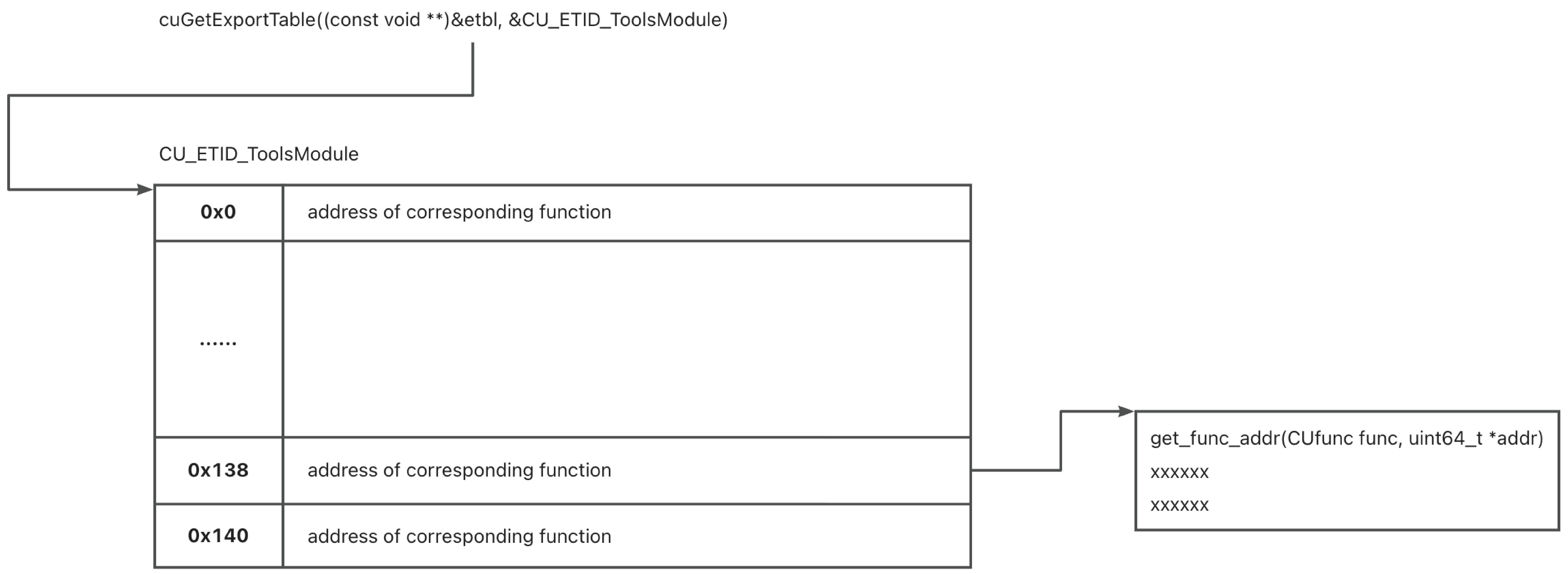
GPU侧解密触发时机

当前解密触发时机为应用程序launch kernel时。
加解密模块以hook内核加载API的相同方法hook了launch kernel API cuLaunchKernel。

后续如果要支持kernel调用其他kernel的场景，以及kernel重复调用的场景，解密触发时机可能要调整。

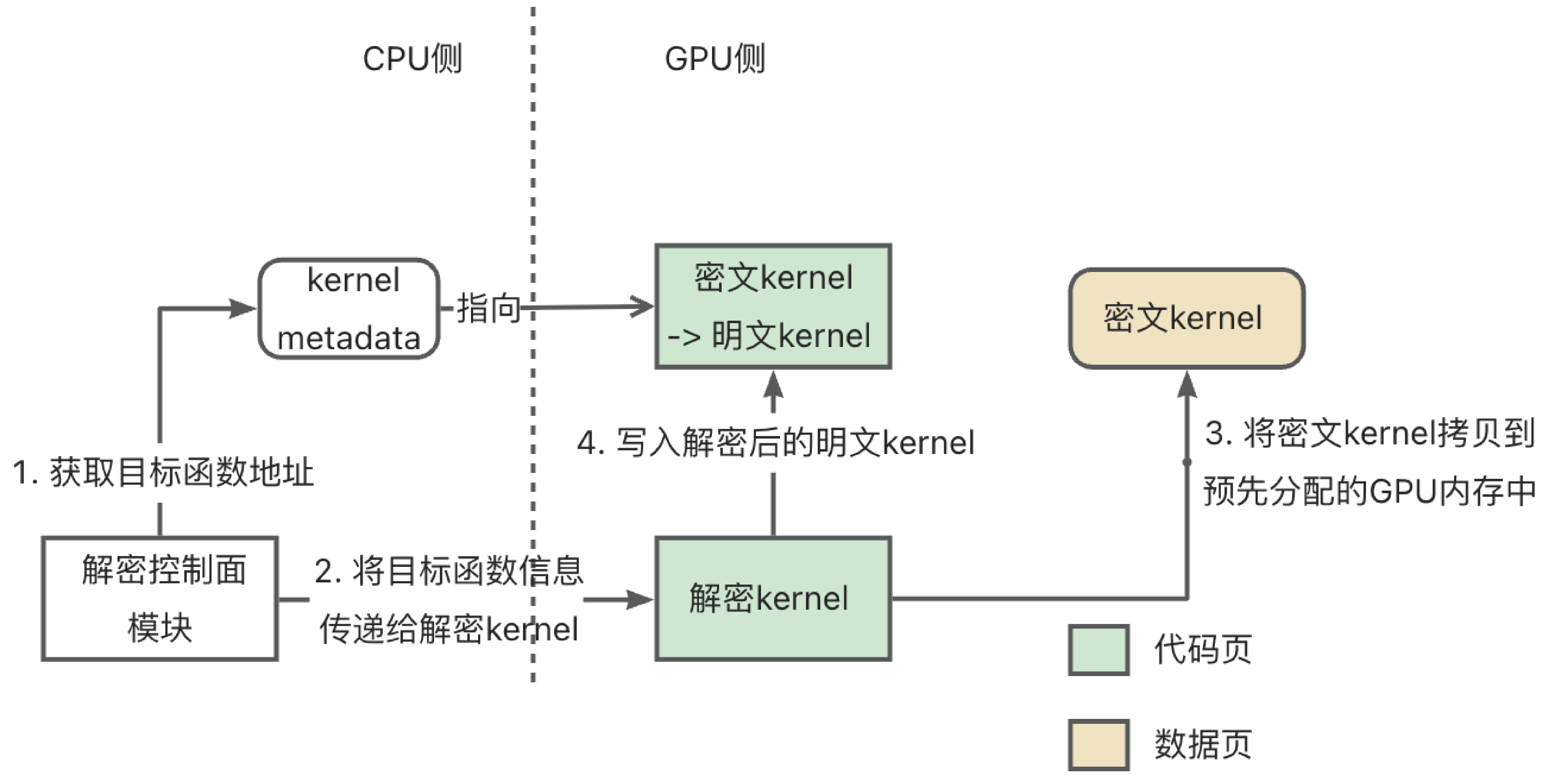
GPU侧SASS编码定位

Nvbit工具提供了获取GPU侧函数地址的方法nvbit_get_func_addr 。
通过反汇编NVbit（使用工具Binary Ninja）获取到了nvbit_get_func_addr的具体实现：该接口调用了一个CUDA driver internal接口，cuGetExportTable。该函数的返回值指向了一个函数指针表ExportTable，对应着一些没有公开的方法。
CU_ETID_ToolsModule对应的ExportTable，在偏移量0x138处指向了获取GPU函数地址的方法。
通过调用该方法，则可以定位到GPU侧的SASS编码位置。


cuGetExportTable((const void **)&etbl, &CU_ETID_ToolsModule)
CU_ETID_ToolsModule
0x0
address of corresponding function
......
0x138
address of corresponding function
0x140
address of corresponding function
get_func_addr(CUfunc func, uint64_t *addr)
xxxxxx
xxxxxx

GPU侧in-place解密

GPU侧代码段可被读、写，代码没有写保护。
通过在GPU侧launch解密kernel，可以in-place将CUDA kernel从密文修改为明文，解密后kernel能正常执行。



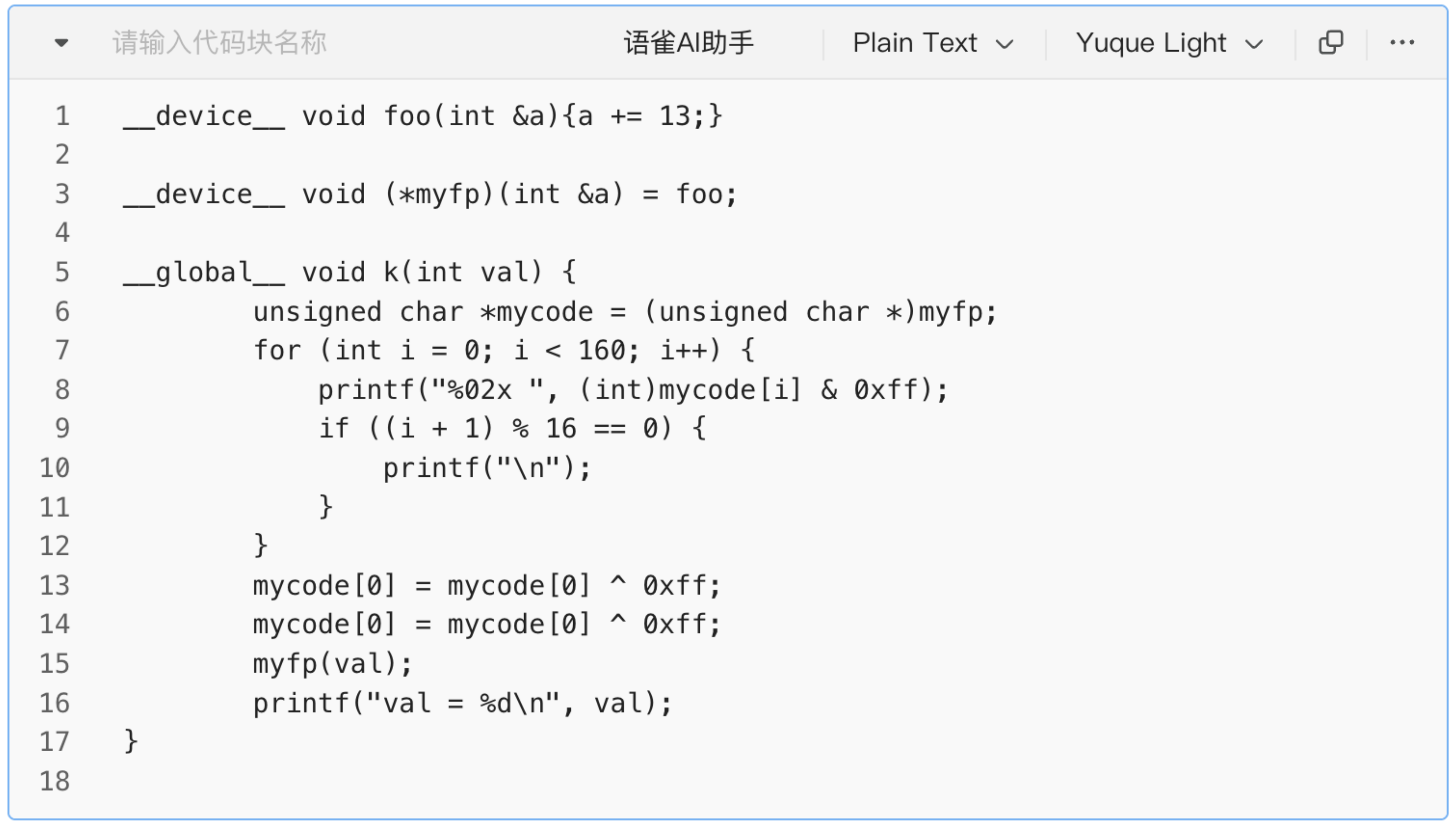
验证环境

目前已经在4090、L20、H20三个型号的GPU上跑通了加解密全流程。

关键发现

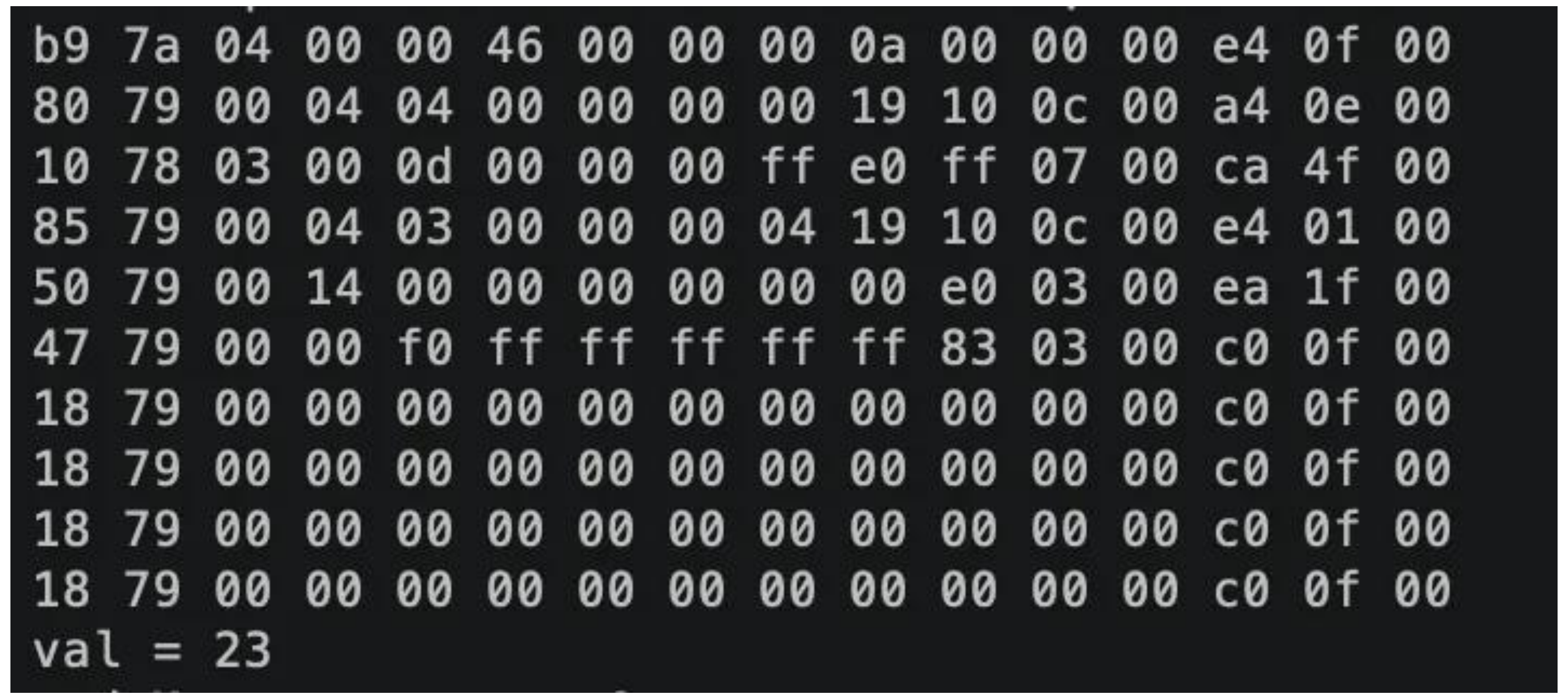
- CUDA driver通过一个特殊接口cuGetExportTable暴露了很多内部接口，目前利用该接口能获取到GPU侧函数地址，未来也许能利用该接口做更多事情。
- GPU侧函数的metadata（如GPU侧函数地址等）均由CPU侧CUDA driver维护。
- GPU侧的代码段没有写保护。

其他问题：部分型号GPU无法通过函数指针读kernel编码



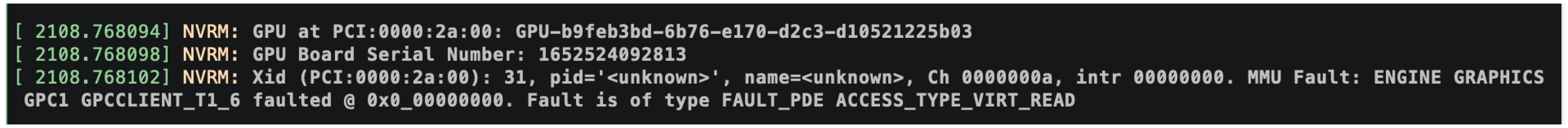
```
__device__ void foo(int &a){a += 13;}

__device__ void (*myfp)(int &a) = foo;

__global__ void k(int val) {
        unsigned char *mycode = (unsigned char *)myfp;
        for (int i = 0; i < 160; i++) {
            printf("%02x ", (int)mycode[i] & 0xff);
            if ((i + 1) % 16 == 0) {
                printf("\n");
            }
        }
        mycode[0] = mycode[0] ^ 0xff;
        mycode[0] = mycode[0] ^ 0xff;
        myfp(val);
        printf("val = %d\n", val);
}

```

测试代码。



```
b9 7a 04 00 00 46 00 00 00 0a 00 00 00 e4 0f 00
80 79 00 04 04 00 00 00 00 19 10 0c 00 a4 0e 00
10 78 03 00 0d 00 00 00 ff e0 ff 07 00 ca 4f 00
85 79 00 04 03 00 00 00 04 19 10 0c 00 e4 01 00
50 79 00 14 00 00 00 00 00 e0 03 00 ea 1f 00
47 79 00 00 f0 ff ff ff ff ff 83 03 00 c0 0f 00
18 79 00 00 00 00 00 00 00 00 00 00 c0 0f 00
18 79 00 00 00 00 00 00 00 00 00 00 c0 0f 00
18 79 00 00 00 00 00 00 00 00 00 00 c0 0f 00
18 79 00 00 00 00 00 00 00 00 00 00 c0 0f 00
val = 23
```

4090上执行k(10)能正确输出SASS编码。



```
[ 2108.768094] NVRM: GPU at PCI:0000:2a:00: GPU-b9feb3bd-6b76-e170-d2c3-d10521225b03
[ 2108.768098] NVRM: GPU Board Serial Number: 1652524092813
[ 2108.768102] NVRM: Xid (PCI:0000:2a:00): 31, pid='<unknown>', name=<unknown>, Ch 0000000a, intr 00000000. MMU Fault: ENGINE GRAPHICS
 GPC1 GPCCLIENT_T1_6 faulted @ 0x0_00000000. Fault is of type FAULT_PDE ACCESS_TYPE_VIRT_READ
```

L20和H20上执行测试代码触发PDE fault。

在4090上，可以通过函数指针获取函数地址并读写kernel编码。L20和H20上，无法通过函数指针地址读kernel编码，但仍然可以通过调用CUDA driver API获取的函数地址读写kernel编码。

GPU attacks related work

- **GPU memory exploit**

[Security’24] GPU Memory Exploitation for Fun and Profit
基于buffer overflow构造GPU ROP攻击。攻击约束条件较多：目标程序有可被利用的buffer overflow vulerabilities；能构造ROP gadgets;GPU无stack canary和ASLR。

- **GPU microarchitecture attacks**

[SEED’24] Beyond the Bridge: Contention-Based Covert and Side Channel Attacks on Multi-GPU Interconnect
[CCS’23] TunneLs for Bootlegging: Fully Reverse-Engineering GPU TLBs for Challenging Isolation Guarantees of NVIDIA MIG
[Security’25] GPUHammer: Rowhammer Attacks on GPU Memories are Practical
[Security’25] Not so Refreshing: Attacking GPUs using RFM Rowhammer Mitigation
[CCS’18] Rendered Insecure: GPU Side Channel Attacks are Practical

可能研究方向

- 基于LD_PRELOAD的GPU攻击。
- GPU**独占**场景下的有效LLM攻击。宿主机如何窃取直通给租户的GPU显存内大模型参数及提示词。
- 基于非公开接口exploit CUDA driver。

为世界带来微小而美好的改变

Bring small and beautiful changes to the world

蚂蚁集团
ANT GROUP