# Meow(**未完成，存在问题**)

2022 年 7 月 31 日


学　　校: 山东大学

学　　院: 网络空间安全学院 (研究院)

姓　　名: 张起萌 202000460118

# 目录

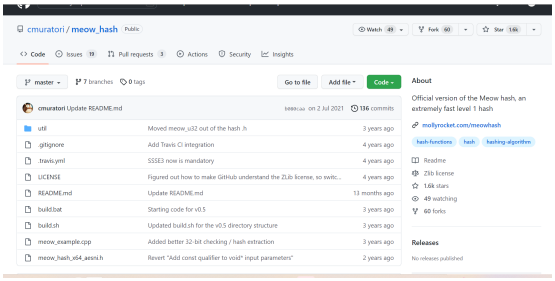
# 1 实验目的

hashA = sdu_cst_20220610，message = ZhangQimeng202000460118（ 0x5a ,0x68, 0x61, 0x6e, 0x67, 0x51, 0x69, 0x6d, 0x65, 0x6e, 0x67, 0x32, 0x30, 0x32, 0x30, 0x30, 0x30, 0x34, 0x36, 0x30, 0x31, 0x31, 0x38 ），构造一个 key 使得 message 的 hash 值等于 hashA。

# 2 实验过程

因为 Meow 的实现过程较为复杂，所以我选择尝试使用 github 上给出的源代码。

- 进入 github 界面，可以看到一些代码，其中我只调用了 meow_hash_x64_aesni.h 函数。




- 我的求解思路是尝试构建 key，利用 key 对 message 进行哈希得到 hashB 的值，如果 hashA=hashB，那么说明 key 构造成功。最开始的时候我简单对 Meow 的 hash 速度进行了估算，发现它非常快，于是我选择了这种方法。但很遗憾的是，我对数据的计算量估计错误，也有可能因为源代码中的 key 的生成并不完全随机，导致我并没有跑出正确的 key。

  代码实现：(其中 meow_hash_x64_aesni.h 为源文件，在参考文献中可以查阅到，不在报告中多加赘述。)

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include "meow_hash_x64_aesni.h"

static void PrintHash(meow_u128 Hash)
{
    printf("    %08X-%08X-%08X-%08X\n",
        MeowU32From(Hash, 3),
        MeowU32From(Hash, 2),
        MeowU32From(Hash, 1),
        MeowU32From(Hash, 0));
}

int main()
{
    size_t size = 128;
    meow_u128 HashA;
    HashA = _mm_loadu_si128((__m128i*)"sdu cst 20220610"); //sdu_cst_20220610
    char Message[] = { 0x5a ,0x68, 0x61, 0x6e, 0x67, 0x51, 0x69, 0x6d, 0x65, 0x6e, 0x67, 0x32, 0x30, 0x32,
    meow_u8* seed = (meow_u8*)malloc(size);
    for (int i = 0; i <= 1000; i++)
    {
        char input[] = "";
        MeowExpandSeed(128, input, seed);
```

```
26            int Size = 16;
27            meow_u128 HashB = MeowHash(seed, Size, Message);
28            int HashesMatch = MeowHashesAreEqual(HashA, HashB);
29            if (HashesMatch)
30            {
31                printf("seed:%p",(const char*)seed);
32                break;
33            }
34        }
35        free(seed);
36        return 0;
37    }
```

# 3 实验总结

这次实验失败的主要原因是我方法使用不当，事实上，我用该代码运行了将近 10 个小时也并没有得到结果，并且由于源代码定义了许多变量名，导致我最开始没有办法很好的一一对应每种变量名应赋什么值。在我重复查看 Meow 源代码的过程中，我发现该代码是可逆的，也就是说按理来讲我应该可以从 hash 值和 message 值通过一些代码的运行而直接得到密钥 key，但很遗憾的是我并没能实现它。

# 4 参考文献

Meow 源代码