**总计完成Project：1 2 3 4 5 8 9 11 15 16**

**重复完成的部分为独立实现而非合作实现**

**[方博韬完成部分]**

**Project 4**

使用官方文档中的精简算法，优化空间小，效率大约为25ms/1w次

**Project 8**

参考ARMv8的密码学拓展模块。但是由于是x86架构所以无法运行。

**Project 9**

由个人高度(软件)优化的AES算法，使用独特的uint32\_t数据存储模式以四个单位为一个数据单元,将访存转化为ALU基本操作，效率大约16ms/1w次。

**Project 11**

SM2数字签名和加解密算法，使用python实现。包含原始版本和RPF6979版本（RFC6979版本实现了k的安全非随机生成）。编写了 椭圆曲线基本运算、类型转换、HMAC等基本功能作为前置。采用分模块设计，详情见readme。

**Project 15**

基于socket的SM2在线数字签名，继承Project11，详情见readme

**Project 16**

基于socket的SM2在线解密，继承Project11，详情见readme

**[侯纪舟完成部分]**

**Project 4**

使用字符串实现大整数的各种运算，单线程下经过64轮迭代的效率约为64ms/次（不包括输出）

**Project 5**

使用字符型数组实现aes和sm4，单线程下aes的效率约为34ms/10000次，sm4的效率约为84ms/10000次（不包括输出）

**Project 9**

基于sha256算法哈希实现的merkletree，可以实现快速的对所求元素是否在树中的检查

**[余海军完成部分]**

**Project 1 & Project 2**

使用了类似查表攻击似的数据结构，一边存表一边查表（可以使用多线程进一步优化脚本性能），以便可以在较短时间内找到一个前16bit的hash弱碰撞。

**Project 4**

主要用了循环展开的方法对sm3在软件层面上优化

**Project 5**

以sm3的散列模块为基础实现了默克尔数，手动输入数据时默克尔树叶节点会生成相应的哈希值，非叶节点会生成相应的联结哈希值

**[李纯完成部分]**

**Project 3**

利用特殊的填充规则，对所截获的字符串先进行填充，再加入自己要伪造的字符串，之后再获取杂凑值。运行速度约为：0.54s

设备信息：

11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz