信息安全~区块链

王润川

2021.11.17

比特币

发展背景

- 2008年，全球金融危机
- 2009年，冰岛、希腊债务危机
- 2010年，欧洲五国债务危机
- 2013年，塞浦路斯债务危机
- ……

- 一个靠信任和权威运行的世界已经崩塌

发展过程

- 2008年：想法的提出
- 2009年：比特币的发行
- 2010年：建立首个加密货币交易所
- 2011年：比特币=美金
- 2014年：微软接受比特币的使用
- 2017年(始)：比特币价格飙升

发展优势

- 总量恒定：
  - 比特币唯一序列，共2100万枚
- 坚不可摧：
  - 比特币网络，存在共识
  - 挖矿算法：哈希碰撞(SHA256算法，公认最安全先进的算法之一)
- 交易隐蔽：
  - 记录全网公开，但背后真人隐匿

区块链

区块链VS比特币

- 比特币≠区块链
- 比特币：
  - 区块链技术的一种应用产品
- 区块链：
  - 比特币应用的技术基础，广泛的应用包括加密数字货币

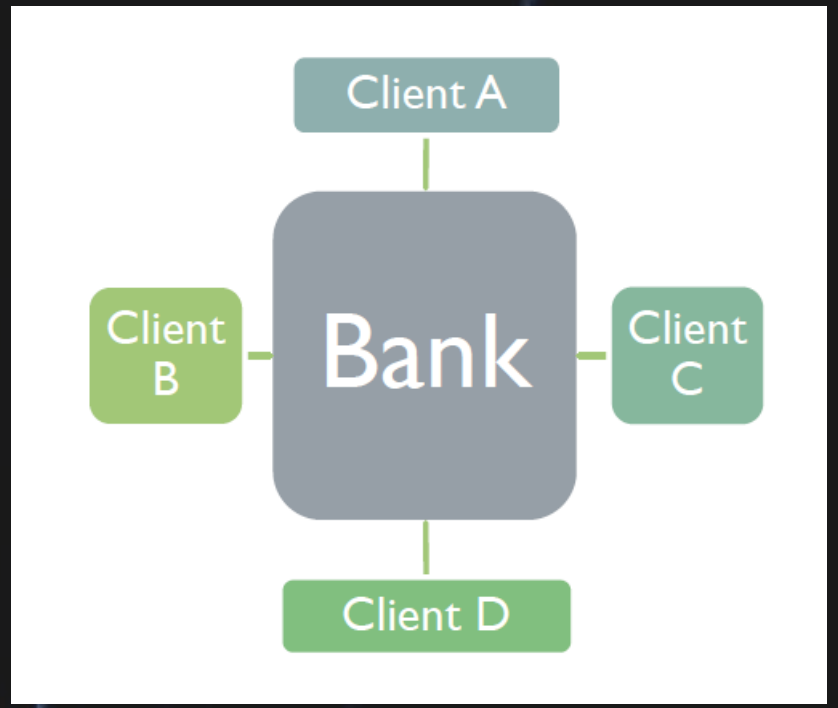
什么是区块链

- 本质上是在多个分布式节点间传递账本信息并通过一定的共识机制达成一致性，建立信任关系的技术

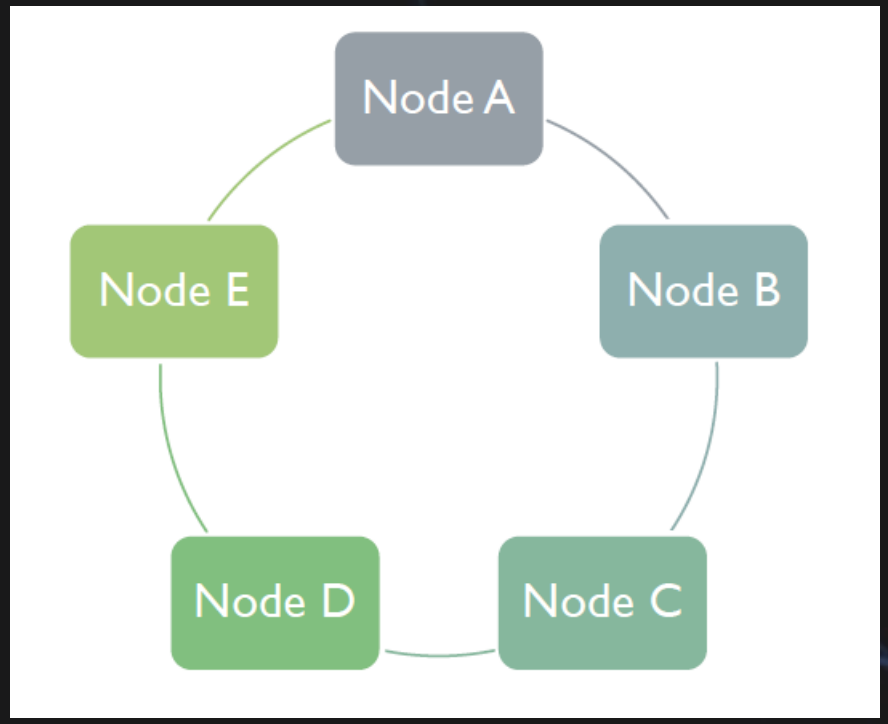
- 独特的遗传式链状数据结构
  - 数据块不会被以追溯方式更改以前的时间戳，即数据不被更改
- 去中心化
  - 公共账本记录事务，通过链上参与节点的多少人投票方式校验并达成一致共识
- 信任机器
  - 区块链技术是在彼此不信任的节点间建立信任关系的技术，即不需要权威中心授权

中心化VS去中心化

中心化

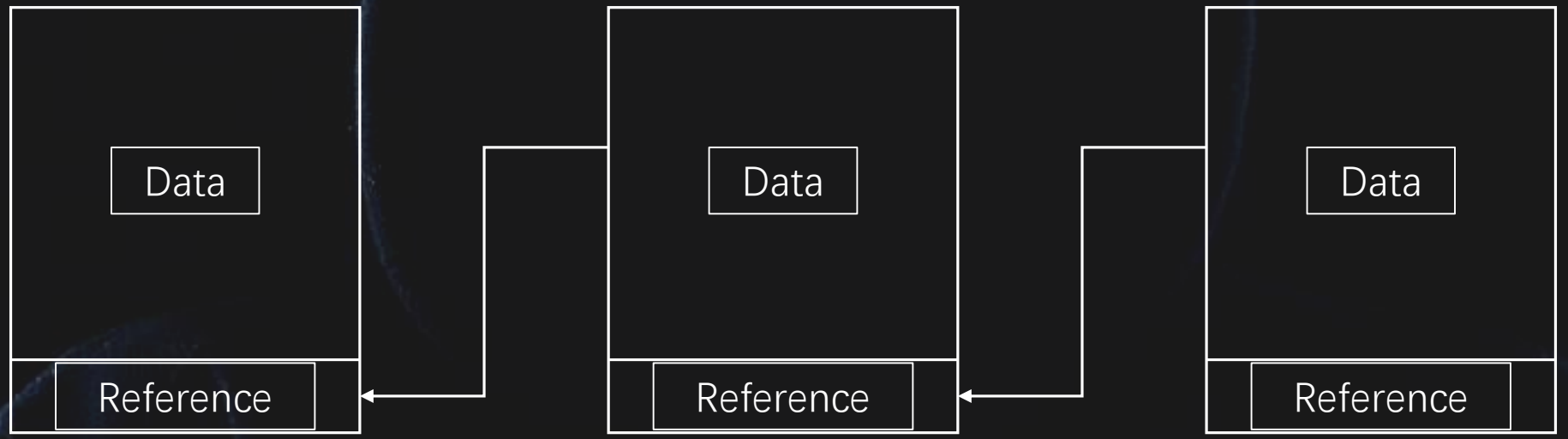


去中心化



区块链加密机制

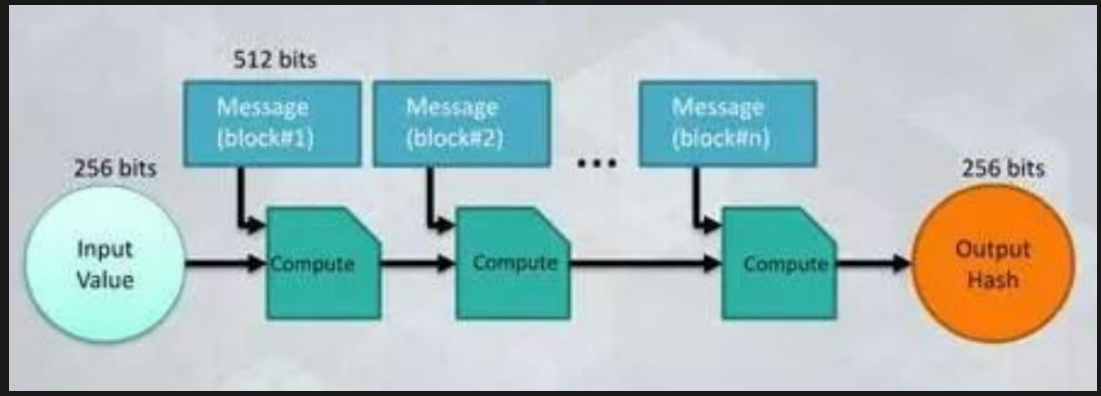
- 数字加密哈希函数+哈希指针数据结构+数字签名
  - 密码通过哈希函数加密后拿到的结果只可以校验，不能反向计算
  - 哈希指针的数据结构一旦产生便不可修改
  - 只可以自己签名，别人可以校验但是不能修改、移动或删除



区块链加密机制——数字加密哈希函数

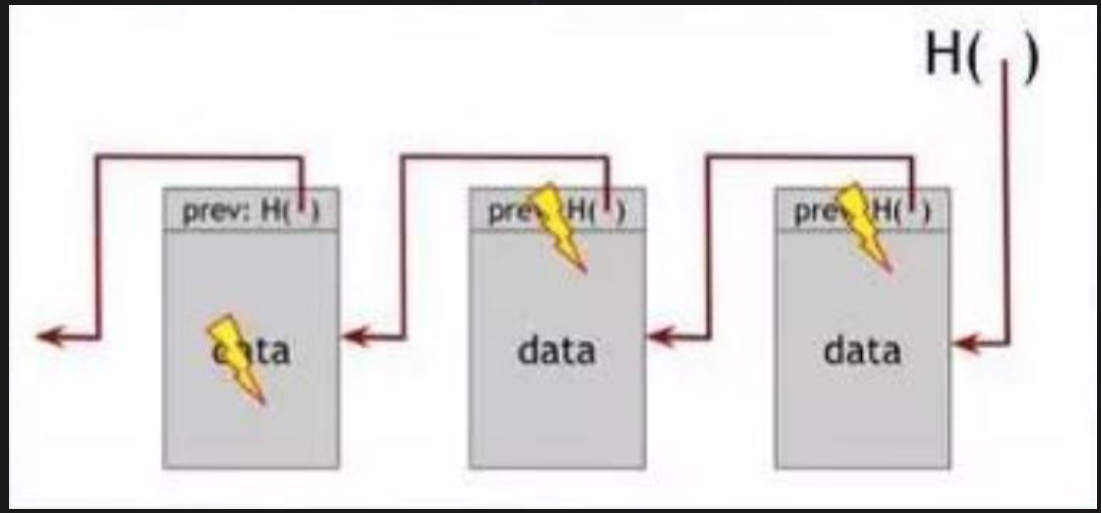
- 抗密码碰撞

- SHA-256哈希函数
  - Merkle-Damgarded变换
  - 任意长度的信息/字符转换成多个哈希函数的512个字节的输入值
  - 不断迭代加密
  - 产生256个字节的加密后的字符串



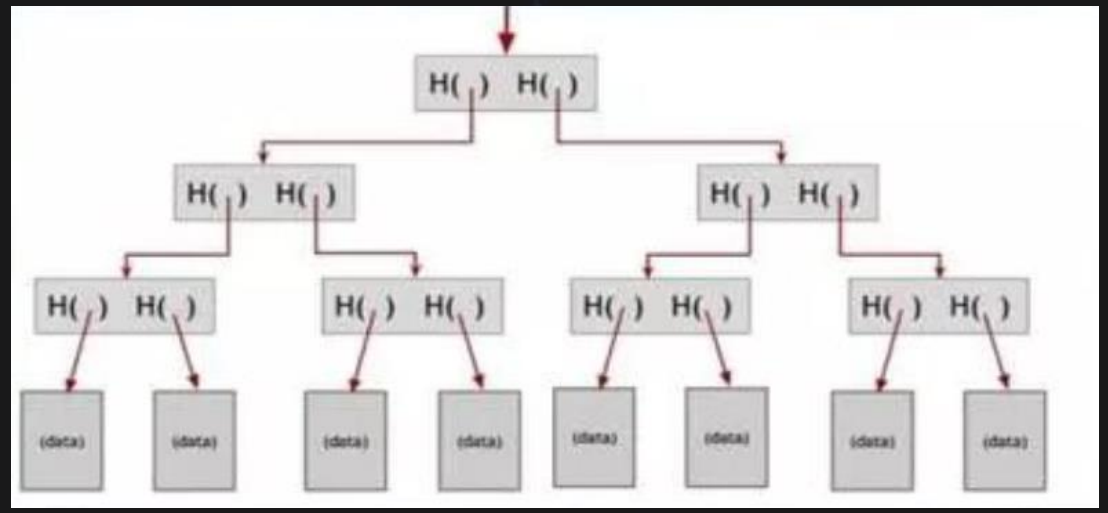
区块链加密机制——哈希指针

- 防止篡改
  - 篡改中间数据无法产生正确的(唯一的)指向下一个数据块的哈希指针
  - 篡改整个区块链指针和数据会因为不知道原始输入密码而无法产生第一个头指针，会被其它节点侦测到

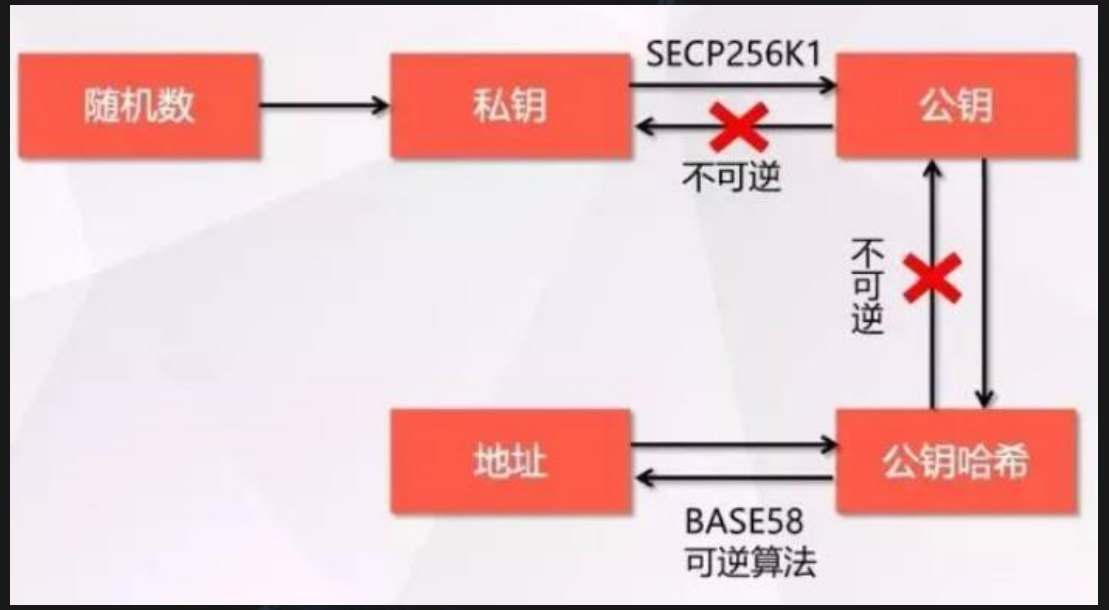


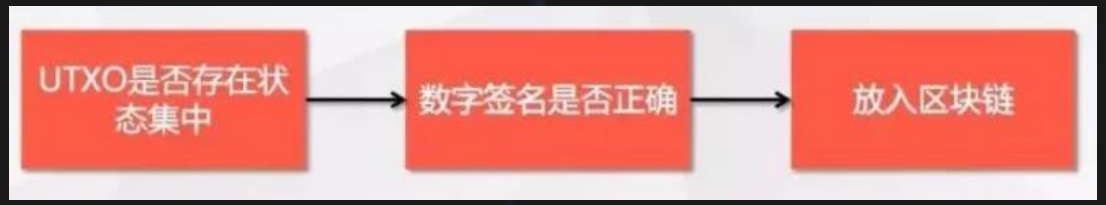
区块链加密机制——数据结构

- Merkle树
  - 所有数据块按配对存储
  - 哈希指针存储在树结构的父节点
  - 父节点产生新的父节点
  - 所有叶节点数据存储在树结构中



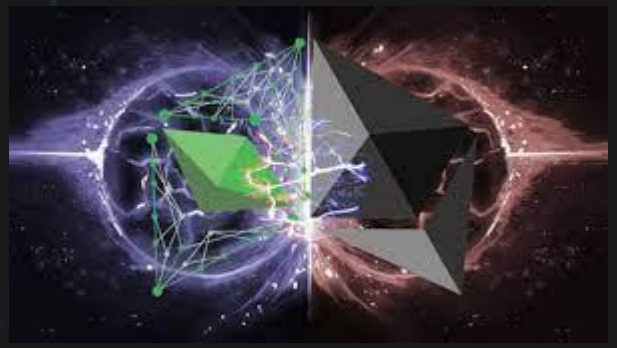
区块链加密机制——数字签名





区块链的共识机制

- 目的：权益划分、安全保障
- 股权证明机制(POS)
  - 持有越多，获得越多
  - 减少共识达成的时间，加快寻找随机数的速度
  - DAO攻击依旧存在(ETC、ETH)
- 授权股权证明机制(DPOS)
  - 股东代表，投票出来的前100名轮流分配时间来生产区块
  - 发布公钥，防止DDOS攻击
- 重要度证明机制(POI)
  - 持有越久，获得越多
  - 给予普通人机会，有意愿坚持持有便可以获得奖励

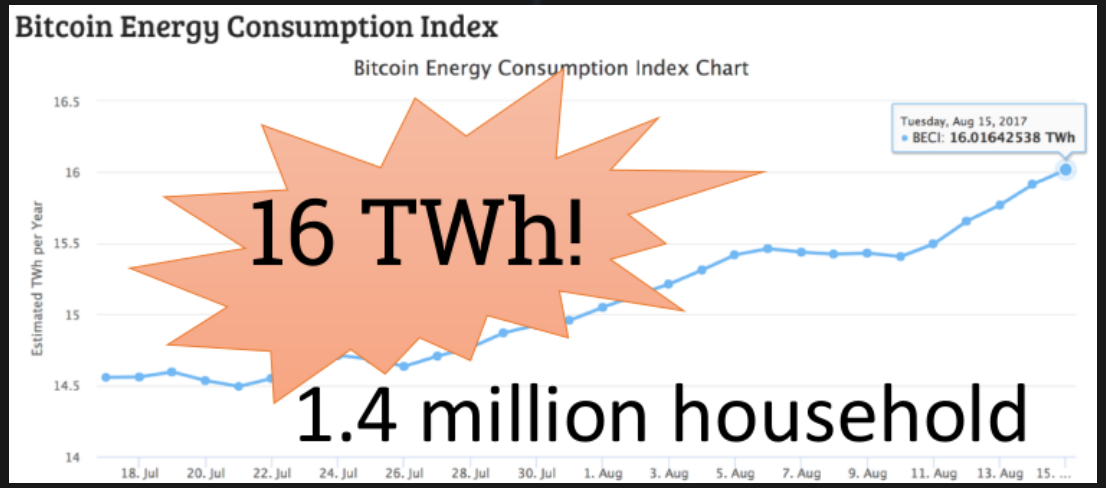


区块链的共识机制

- 困境：拜占庭将军问题
- 拜占庭共识机制(PBFT)
  - 保证活性和安全性，提供(n-1)/3的容错性
  - 1/3恶意会导致分叉，但会留下密码学证据
- 工作量证明机制(POW)
  - 干得越多，获得越多
  - 资源消耗相比其它共识机制更高，每次达成共识需要全网参与运算
  - 所有节点都平等计算一个数学难题，最先获得答案的节点将获得区块发布权
  - 容错性容许全网50%节点出错(全网算力作为保障)
  - “双花问题”、“51%攻击”
  - 区块链每隔10分钟发布一次，避免分叉
    - 每完成2018个块根据出块的平均时间调整一次难度

区块链的共识机制——POW的困境

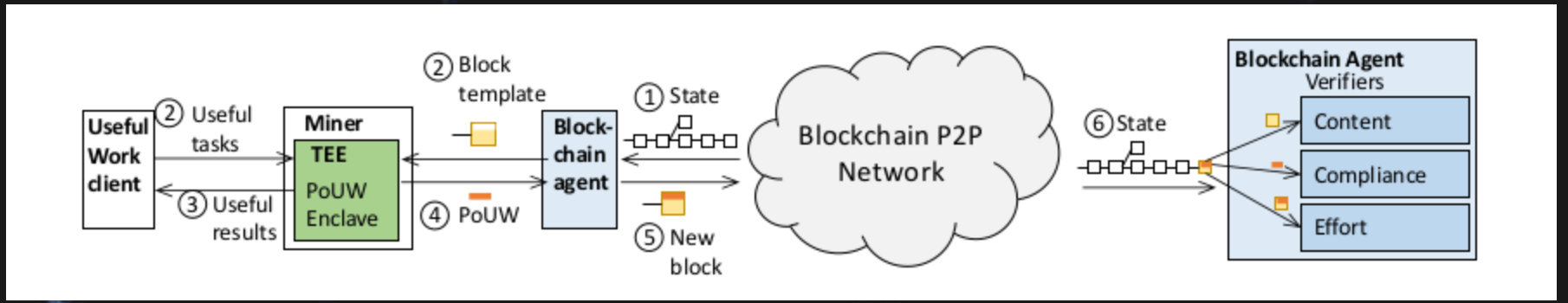
```
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/o
rg1.example.com/peers/peer0.org1.example.com/tls/server.key
CORE_PEER_LOCALMSPID=Org2MSP
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/
org1.example.com/peers/peer0.org1.example.com/tls/server.crt
CORE_PEER_TLS_ENABLED=true
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/
org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ID=cli
CORE_LOGGING_LEVEL=DEBUG
CORE_PEER_ADDRESS=peer1.org2.example.com:7051
Attempting to Query PEER3 ...3 secs

2018-08-09 22:57:40.905 UTC [msp] GetLocalMSP -> DEBU 001 Returning existing local MSP
2018-08-09 22:57:40.905 UTC [msp] GetDefaultSigningIdentity -> DEBU 002 Obtaining default signing id
entity
2018-08-09 22:57:40.905 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default escc
2018-08-09 22:57:40.905 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default vscc
2018-08-09 22:57:40.906 UTC [msp/identity] Sign -> DEBU 005 Sign: plaintext: 0A95070A6708031A0C08E48
DB3DB0510...6D7963631A0A0A0571756572790A0161
2018-08-09 22:57:40.906 UTC [msp/identity] Sign -> DEBU 006 Sign: digest: BC4097469A733188F7E4C2A846
C0B56CC1A24B832FEBEF29E6A6095412112 5AA
Query Result: 90
2018-08-09 22:58:01.248 UTC [main] main -> INFO 007 Exiting.....
==================== Query on PEER3 on channel 'mychannel' is successful ====================

==================== All GOOD, End-2-End execution completed ====================
```



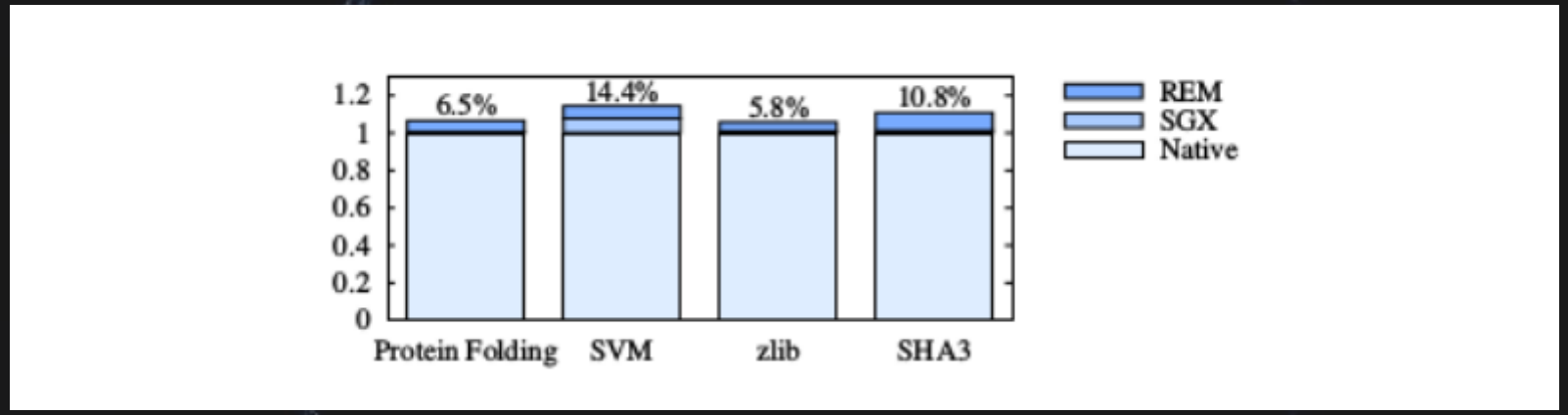
→ POUW & REM

区块链的共识机制——POUW

- 天下三分：
  - 代理端：分配任务
  - 矿工：接受任务
  - 客户端：提供负载，完成任务
- 结合静态的混合和动态的程序分析方法：
  - Intel的SGX保护区域
  - 工作程序是否恶意



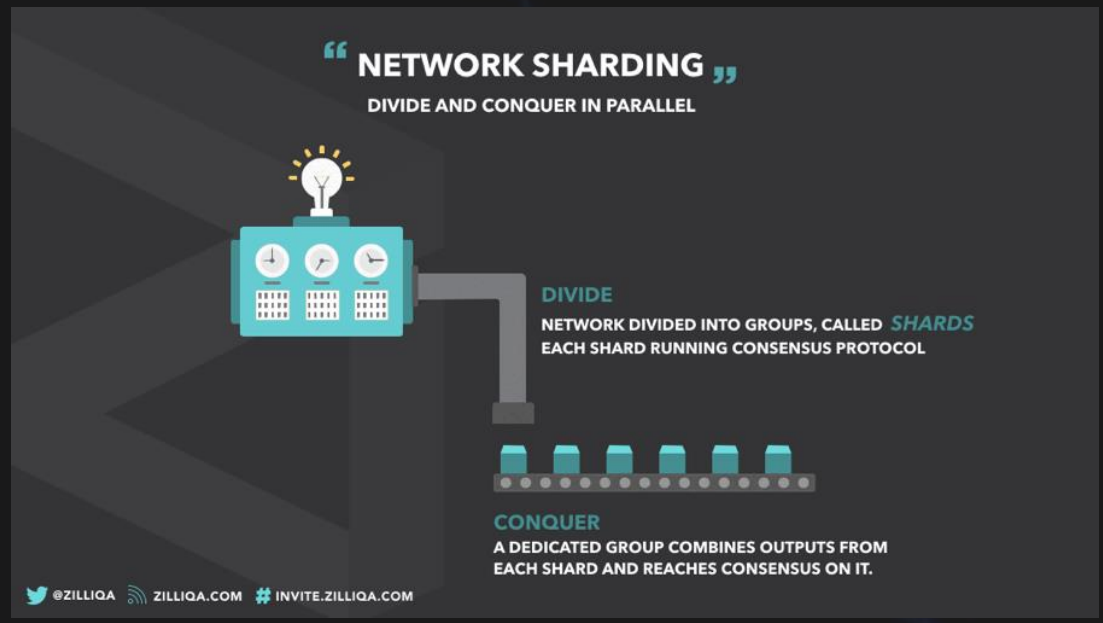
区块链的共识机制——POUW



效率还可以继续提高吗？

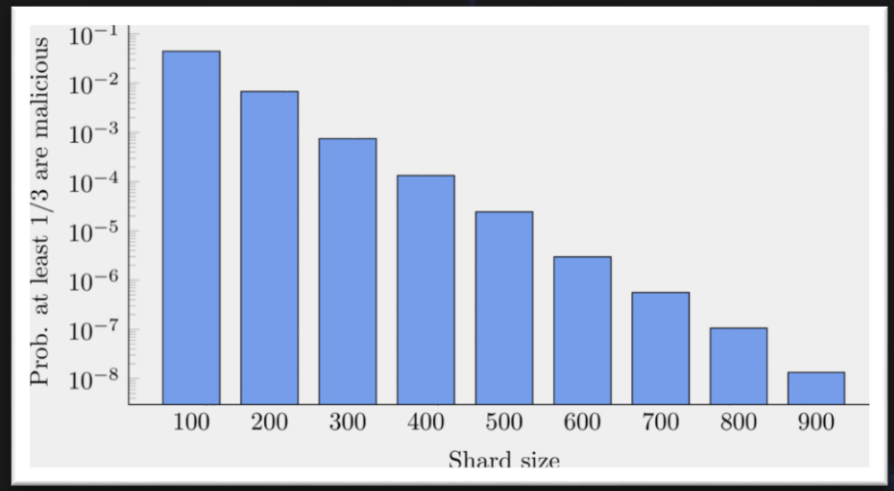
区块链的分片机制

- 传统方法：
  - 所有工作均有一个节点完成
  - 成本=共识机制执行
- 最新方法：
  - 工作被分配给不同的节点
  - 成本=共识机制执行+分片成本



区块链分片机制——问题和解决方法

- 分片攻击
  - Sybil攻击
  - 提高攻击成本，例如使用POW(POUW)，即50%容错
- 分片方法
  - 建立委员会
  - 组织节点竞争
  - 给节点随机分发任务
- 分片规模
  - 分片节点数目太少不安全，太多低效率
  - 兼顾效率和安全为600



安全研究

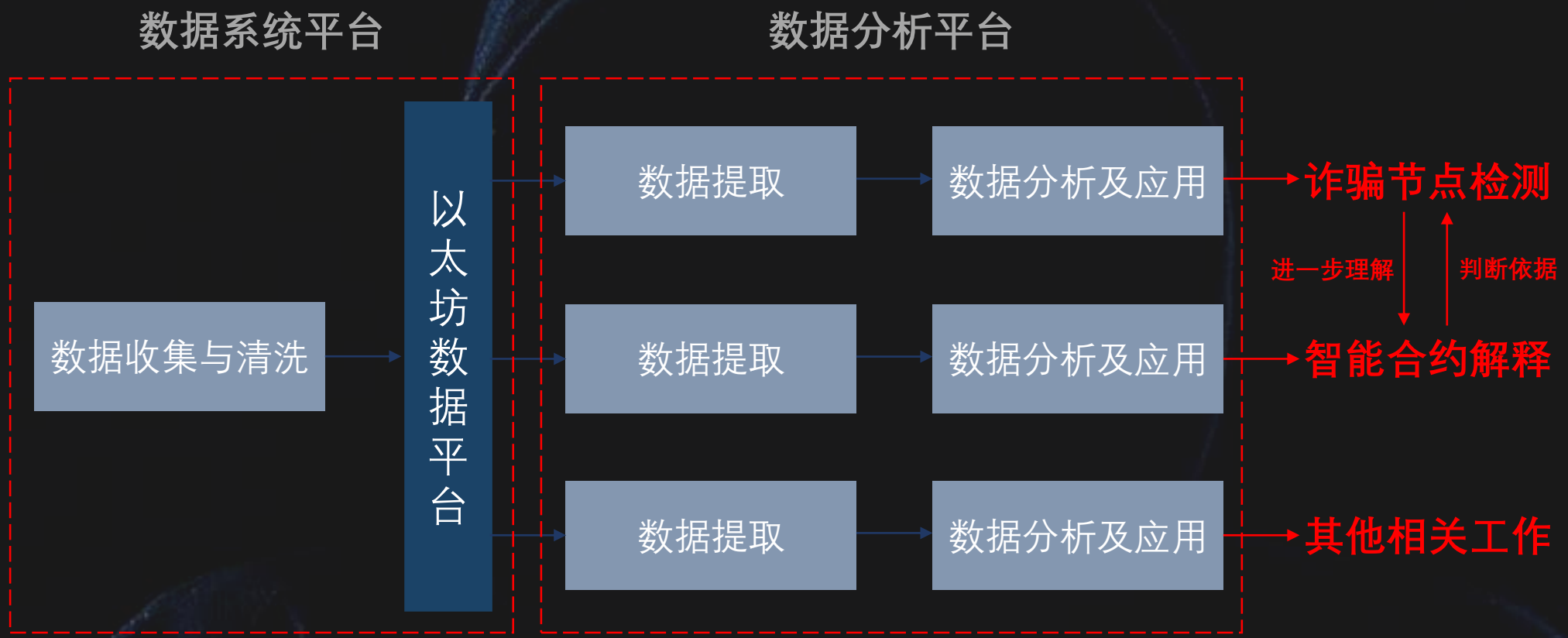
研究背景

- 以太坊(Ethereum)
  - 第二大公共区块链网络
  - 可编程的区块链平台，支持智能合约
    - 基于各种金融模块的开放式金融(DeFi)
    - 去中心化的金融产品
- 加密金融市场上占据重要地位

研究挑战

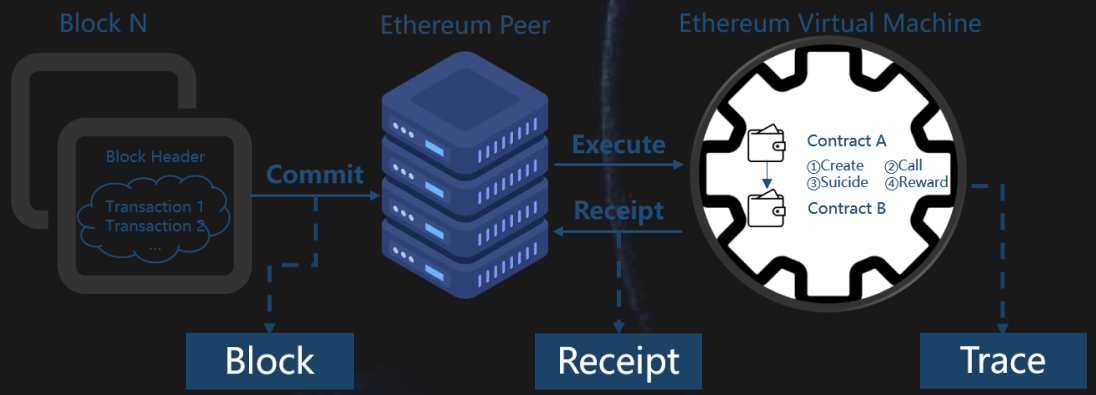
- 数据收集只能通过创建节点同步，同步需要大量的时间成本、设备成本
- 源数据是二进制或加密形式，而且数据属性复杂，处理困难
- 缺少通用的以太坊数据提取工具
- 缺少针对以太坊基本数据分析的工具

研究目标


数据系统平台
数据分析平台
数据收集与清洗
以太坊数据平台
数据提取
数据分析及应用
数据提取
数据分析及应用
数据提取
数据分析及应用
诈骗节点检测
进一步理解
判断依据
智能合约解释
其他相关工作

数据收集

- 数据的结构



- 数据的规模

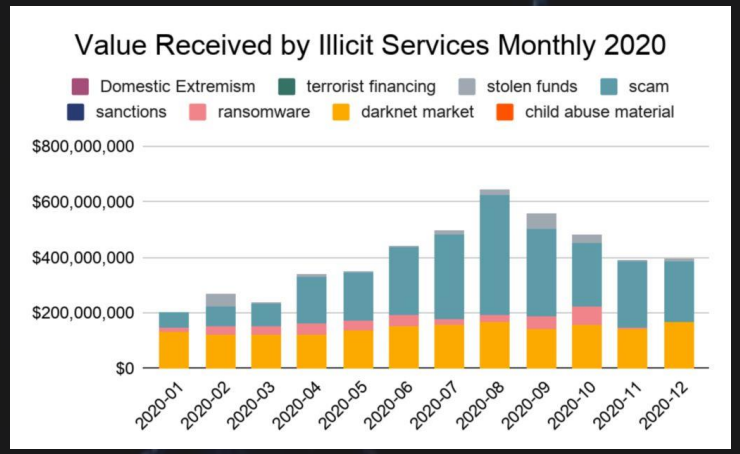
数据属性	Values
区块高度	13378614
地址数目	>1999999
日活地址数目	584646
交易数目	>1311325118
平均区块生成时间	13.46秒
平均区块大小	81744字节
Archive模式同步时长	41天9小时
Archive模式同步空间大小	8619.71945Gb

安全研究

工作一：以太坊诈骗节点检测

工作背景

- 形势
  - 黑客行为持续猖獗
  - 诈骗行为比重高、危害大



- 挑战
  - 带标记数据极少
  - 数据规模大

- 进展
  - 提出一种基于自监督图增量学习模型的以太坊诈骗节点检测方法

相关工作

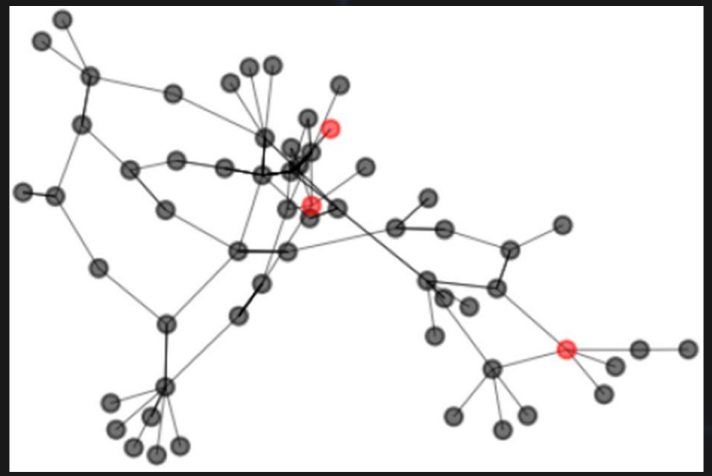
- 传统训练方式：基于特征工程
- 新兴训练方式：基于GCN方法

- 所使用数据集规模较小
  - 以太坊规模很大

- 采用带偏置的采样方法
  - 采样图和实际图分布差别大

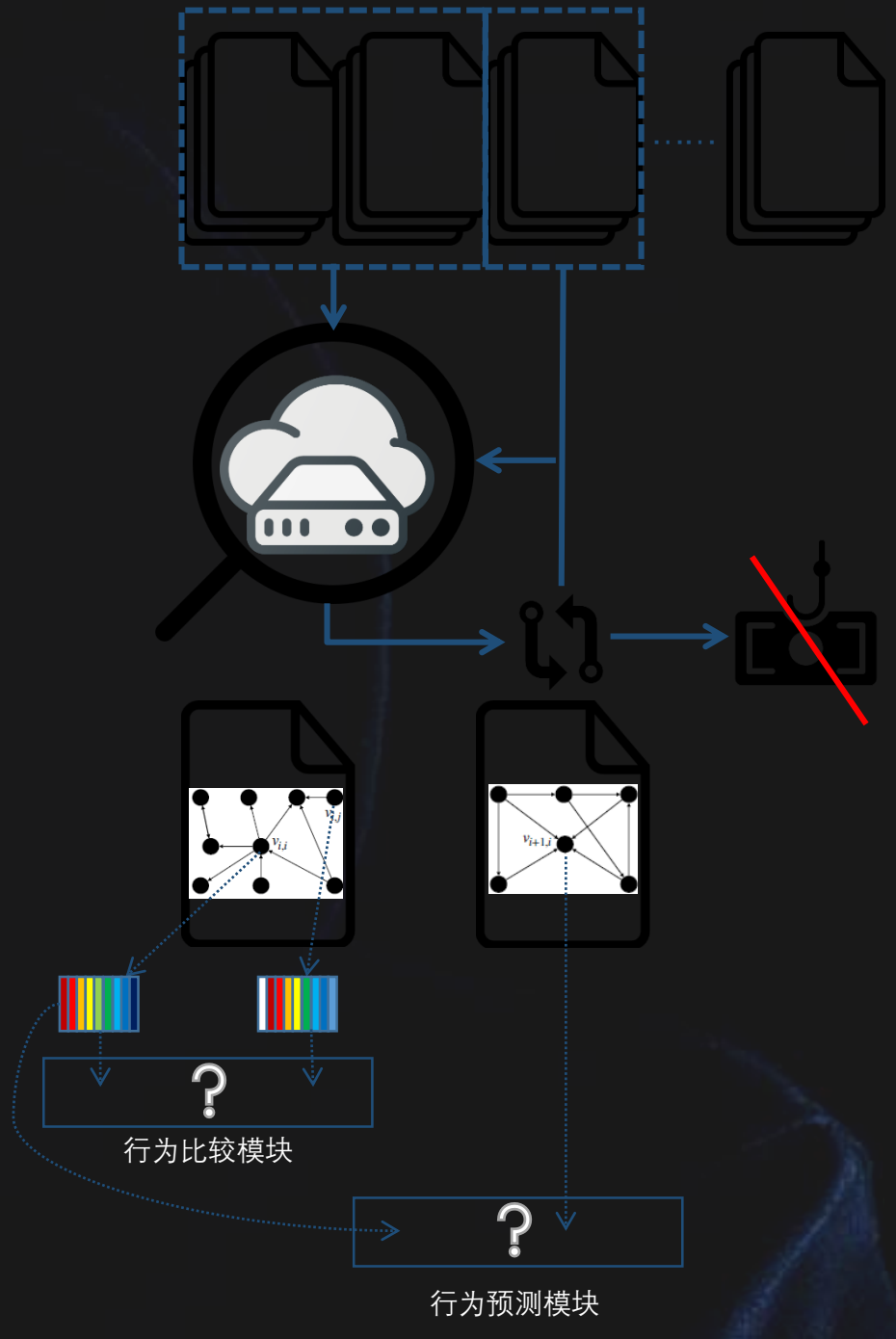
- 直推式方法
  - 不适合不断有新节点和新子图进入的场景

Subgraph	#Labeled nodes	#Edges	#Average degree
30,000	113	1,140,091	76.0061
40,000	134	1,292,279	64.6140
50,000	172	1,418,893	56.7557



我们的工作

- 自监督学习任务一
  - 空间前置训练任务
  - 保持常交易(K跳邻居)行为相似性
- 自监督学习任务二
  - 时间前置训练任务
  - 行为时间维度上连续性，弥补标签不平衡
- 大规模图
  - 实验根据区块划分演化图
  - 训练采用增量训练

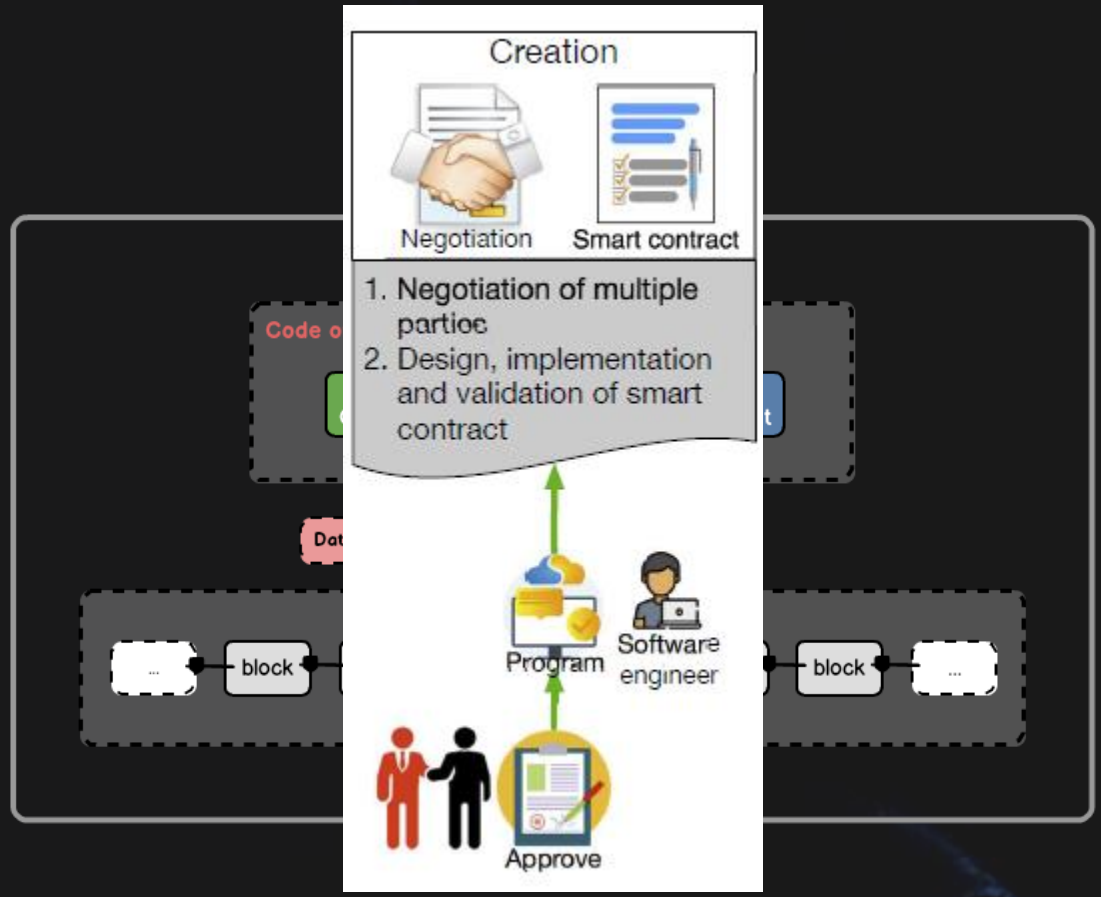


安全研究

工作二：以太坊智能合约解释

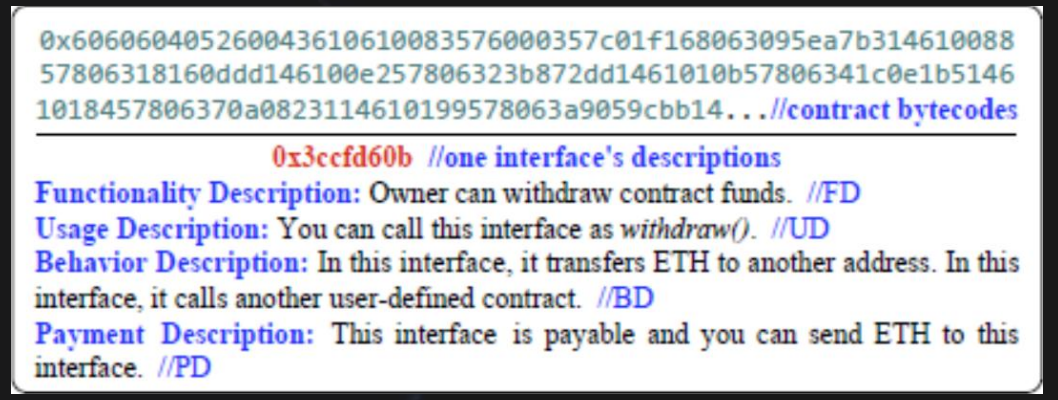
工作背景

- 智能合约
  - 自动执行的程序语言
  - 多方协商、指定业务逻辑、开发编写
- 工作重要性
  - 内部交易量Boom！智能合约量Boom！
  - 不开源的智能合约占比约77%
- 工作进展
  - 项目进行中



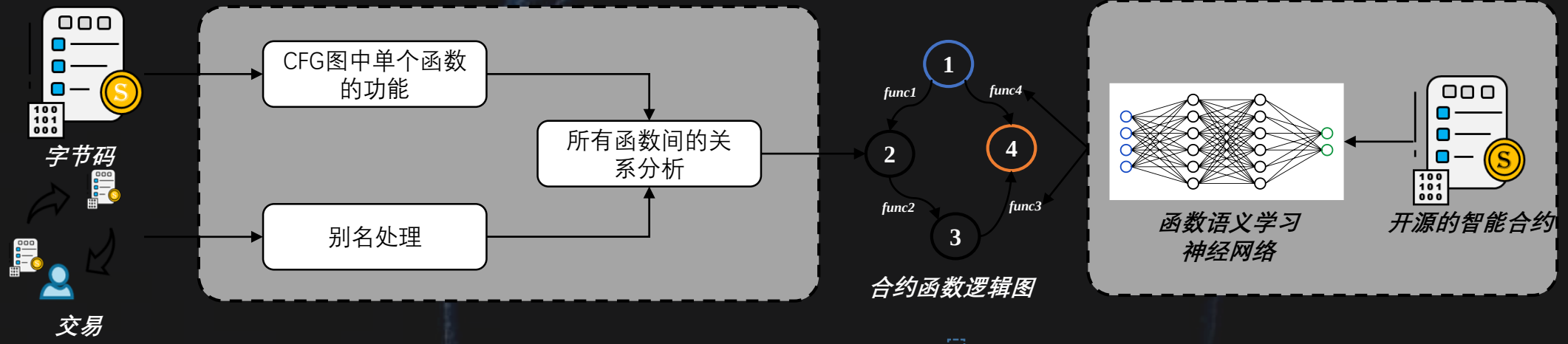
相关工作

- 二进制反编译为类源码
  - Erays、Gigahorse、IELE等
- 识别简单的函数行为
  - Stan等
  - 函数基本块识别特定的行为
- 工作的需求
  - 自动化、合约级别、面向字节码、面向所有人
  - 易于理解：主体、功能、顺序



	自动化式	合约级别	面向字节码	面向所有人
反编译	✔	✔	✔	
函数功能	✔		✔	✔
工作	✔	✔	✔	✔

我们的工作



时序信息

1. User A can create a new ballot. *(func1)*
2. User A then give User B the right to vote on this ballot. *(func2)*
3. User B can delegate vote to others or gives his vote. *(func3)*
4. User A can compute the wining proposal. *(func4)*

用户信息　　控制信息

课后作业

选做二

- 可选的课题：
  - ~~区块链发展的前世今生~~
  - parity和geth的比较
  - 区块链的安全策略
  - 区块链的安全场景
  - NFT token应用
  - ……
- 要求：
  - 独立完成，禁止抄袭
  - 提交报告，不少于1000字
  - 需要有2-3篇的论文支撑(最好是英文论文，这样你的分数会更高)
  - 截至时间：12月15前
  - 发送邮箱：3055242489@qq.com

信息安全~杂项

王润川

2021.11.17


图片隐写

图片隐写——图片格式


HEX
6A3F2C


About
010 Editor
SWEETSCAPE
SOFTWARE
010 Editor
v3.1.2
The Professional Text/Hex Editor
Modified by Playboysen
All Rights Reserved.
Close

图片隐写——图片格式

jpeg、jpg、png、gif…

png格式：

89 50 4E 47 0D 0A 1A 0A + 数据块 + 数据块 + 数据块…

标准数据块：IHDR(文件头数据块)、PLTE(调色板数据块)、IDAT(图像数据块)、IEND(图像结束数据块)

辅助数据块：…

特别的！！

IHDR:

Chunk Data部分13字节

1~4字节表示图像的宽度；

5~8字节表示图像的高度。

IEND:

一般为00 00 00 00 49 45 4E 44 AE 42 60 82

如果后面还有数据【滑稽】~

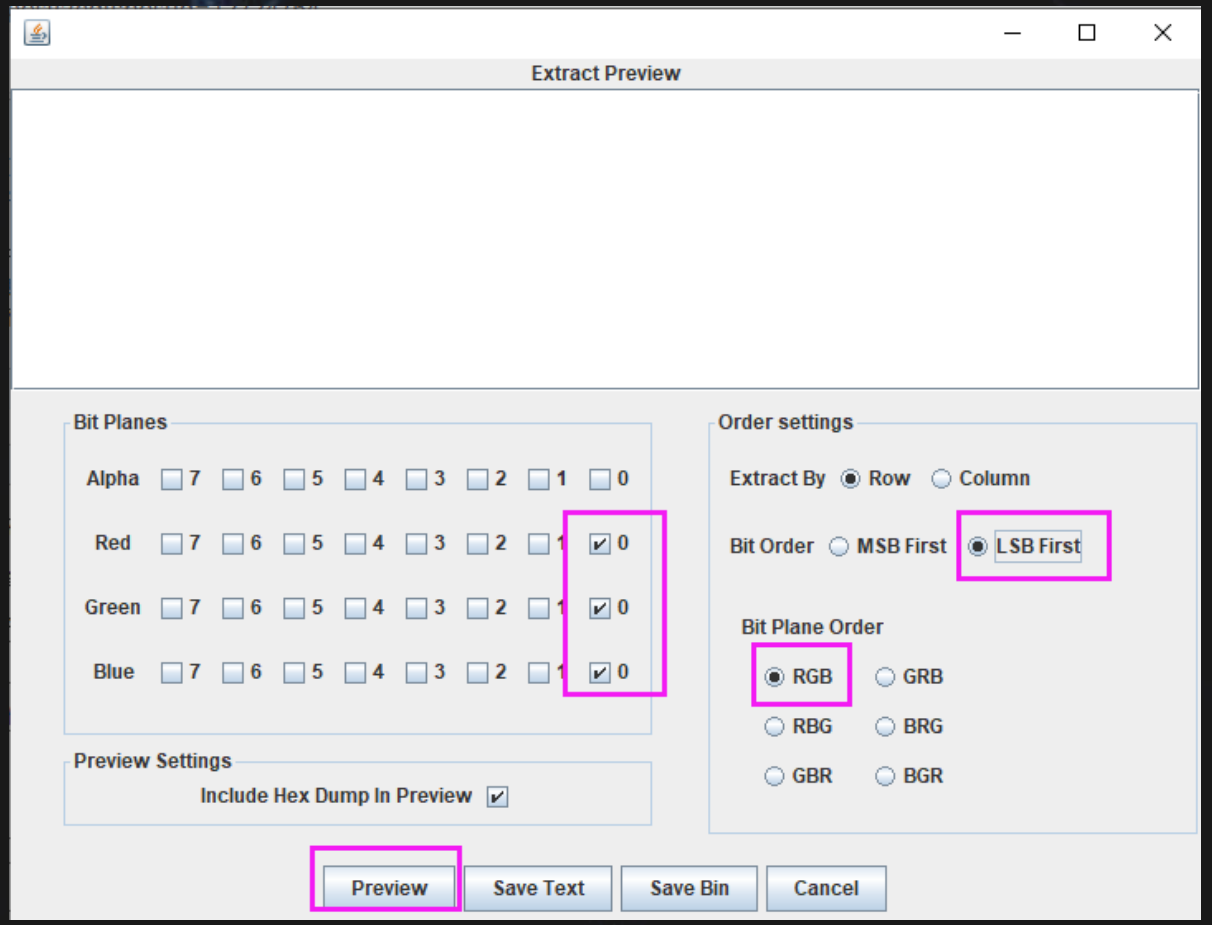
名称	字节数	含义
Length	4字节	数据域长度
Chunk Type Code	4字节	数据块类型码
Chunk Data	可变	数据
CRC	4字节	循环冗余码

图片隐写——LSB

- Least Significant Bit
- 最低有效位
- PNG の RGB三原色
- 每种颜色8位表示(0x00~0xFF)
- 1个像素携带3位数据

- python的Image模块
- Stegsolve
- 图片通道查看器、图片隐写必备

图片隐写——LSB


Extract Preview
Bit Planes
Alpha 7 6 5 4 3 2 1 0
Red 7 6 5 4 3 2 1 0
Green 7 6 5 4 3 2 1 0
Blue 7 6 5 4 3 2 1 0
Preview Settings
Include Hex Dump In Preview
Order settings
Extract By Row Column
Bit Order MSB First LSB First
Bit Plane Order
RGB GRB
RBG BRG
GBR BGR
Preview
Save Text
Save Bin
Cancel

压缩包加密

压缩包加密——压缩包格式

zip、rar、7z…

zip格式：
压缩源文件数据区 + 核心目录区 + 目录结束标识

zip格式的重要标志：
文件头标识由固定值50 4B 03 04开头

压缩包加密——伪加密

Offset	Bytes	Description	译
0	4	Central directory file header signature = 0x02014b50	核心目录文件 header 标识 =（0x02014b50）
4	2	Version made by	压缩所用的 pkware 版本
6	2	Version needed to extract (minimum)	解压所需 pkware 的最低版本
8	2	General purpose bit flag	通用位标记伪加密
10	2	Compression method	压缩方法
12	2	File last modification time	文件最后修改时间
14	2	File last modification date	文件最后修改日期
16	4	CRC-32	CRC-32 校验码

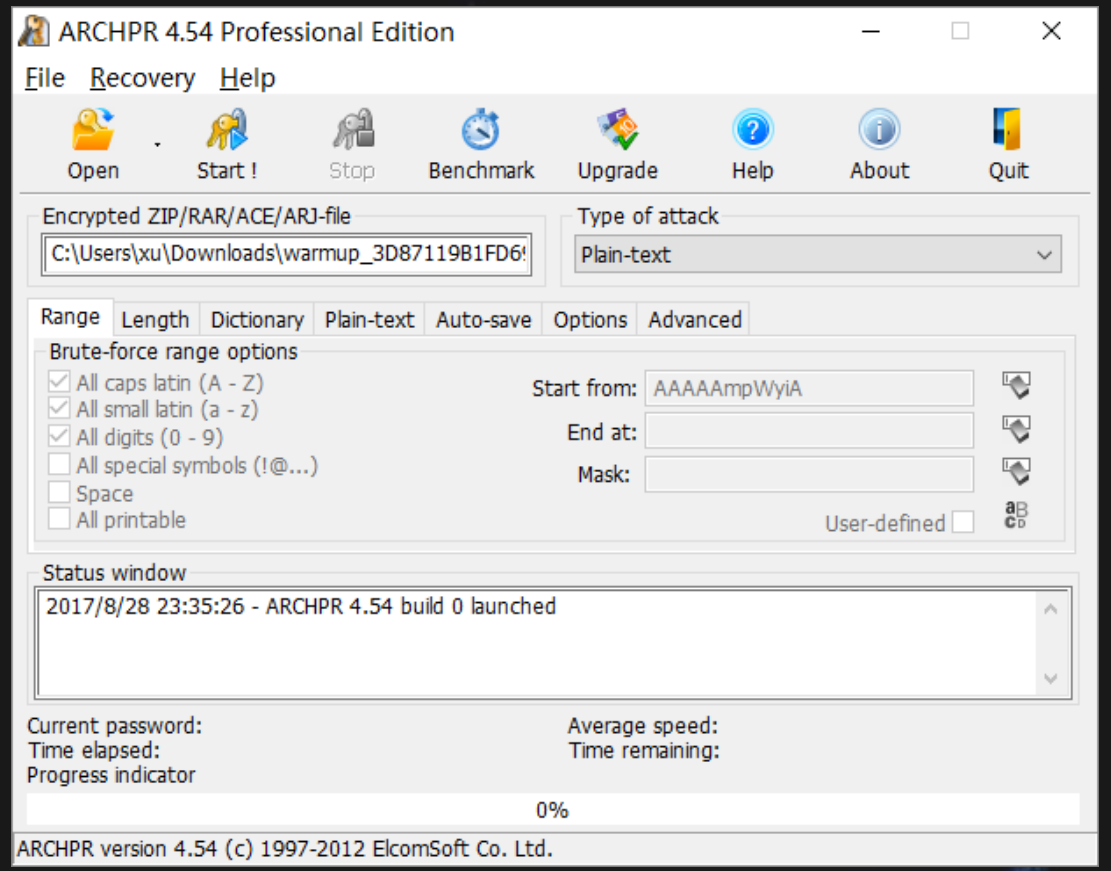
压缩包加密——爆破工具

密码较为简单

已知密码的格式或范围

Windows工具：ARCHPR

Linux工具：命令行fcrackzip



流量分析

流量分析——流量文件&流量工具

tcpdump工具
网络设备
PCAP格式流量文件

文件修复：pcapfix
协议分析：wireshark
数据提取：tshark

流量分析——特别的协议

No.	Time	Source	Destination	Protocol	Length	Bluetooth HCI Event	Info
1	0.000000	2.1.1	host	USB	35		URB_INTERRUPT in
2	0.137131	2.1.1	host	USB	35		URB_INTERRUPT in
3	0.299751	2.1.1	host	USB	35		URB_INTERRUPT in
4	0.399781	2.1.1	host	USB	35		URB_INTERRUPT in
5	0.838075	2.1.1	host	USB	35		URB_INTERRUPT in
6	0.968796	2.1.1	host	USB	35		URB_INTERRUPT in
7	1.184415	2.1.1	host	USB	35		URB_INTERRUPT in
8	1.316126	2.1.1	host	USB	35		URB_INTERRUPT in
9	1.599310	2.1.1	host	USB	35		URB_INTERRUPT in
10	1.934871	2.1.1	host	USB	35		URB_INTERRUPT in
11	2.054854	2.1.1	host	USB	35		URB_INTERRUPT in
12	2.067291	2.1.1	host	USB	35		URB_INTERRUPT in
13	2.384149	2.1.1	host	USB	35		URB_INTERRUPT in
14	2.484050	2.1.1	host	USB	35		URB_INTERRUPT in
15	3.000238	2.1.1	host	USB	35		URB_INTERRUPT in
16	3.116183	2.1.1	host	USB	35		URB_INTERRUPT in

```
> Frame 1: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
> USB URB
  Leftover Capture Data: 0000090000000000
```

```
tshark -r filename.pcap -T fields -e usb.capdata
```

```
0000  1b 00 40 39 2d ac 89 b6  ff ff 00 00 00 00 09 00   ..@9-... ........
0010  01 02 00 01 00 81 01 08  00 00 00 00 00 09 00 00   ........ ........
0020  00 00 00                                          ...
```

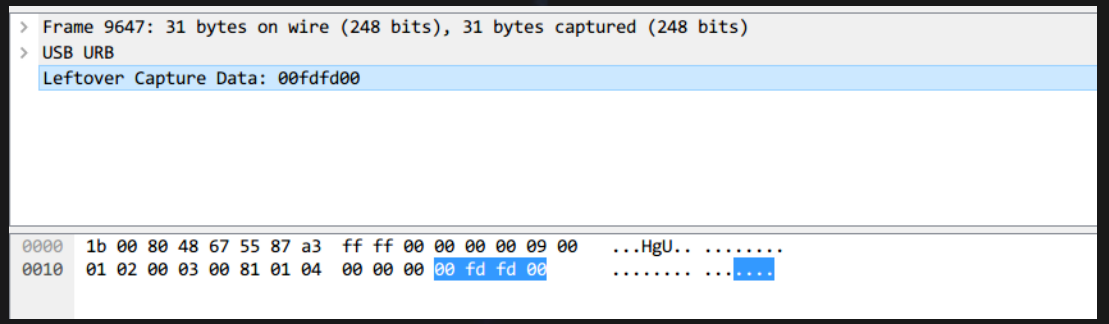
流量分析——特别的协议

鼠标协议：4个字节

第1个字节表示按键——0x00表示没有按键，0x01表示按左键，0x02表示按右键；

第2个字节表示X轴移动长度；

第3个字节表示Y轴移动长度；



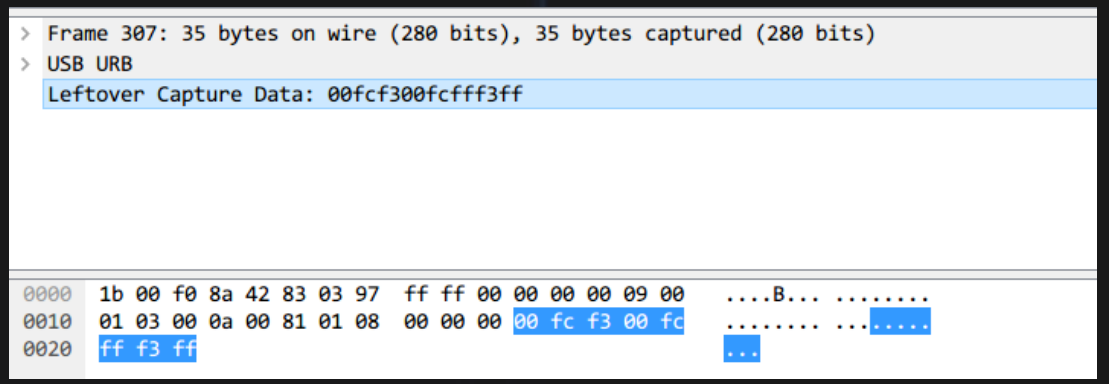
流量分析——特别的协议

键盘协议：8个字节

第1个字节：组合键

第2个字节：OEM保留

第3~8个字节：按键码
(击键信息集中在第3个字节)



流量分析——特别的协议

https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
0	00	Reserved (no event indicated)[9]	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver[9]	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail[9]	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined[9]	N/A	√	√	√	4/101/104
4	04	Keyboard a and A[4]	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C[4]	48	√	√	√	4/101/104
7	07	Keyboard d and D	33	√	√	√	4/101/104
8	08	Keyboard e and E	19	√	√	√	4/101/104
9	09	Keyboard f and F	34	√	√	√	4/101/104
10	0A	Keyboard g and G	35	√	√	√	4/101/104
11	0B	Keyboard h and H	36	√	√	√	4/101/104
12	0C	Keyboard i and I	24	√	√	√	4/101/104
13	0D	Keyboard j and J	37	√	√	√	4/101/104
14	0E	Keyboard k and K	38	√	√	√	4/101/104
15	0F	Keyboard l and L	39	√	√	√	4/101/104
16	10	Keyboard m and M[4]	52	√	√	√	4/101/104
17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O[4]	25	√	√	√	4/101/104
19	13	Keyboard p and P[4]	26	√	√	√	4/101/104
20	14	Keyboard q and Q[4]	17	√	√	√	4/101/104