一个佛系安全从业者的区块链世界观

现状

等待下一个轮回

曾经“辉煌”的通证经济市场

1932亿美元

最高点8285亿美元

当前区块链数字货币的总市值

367万美元

大额账户的平均持有数字货币的价值

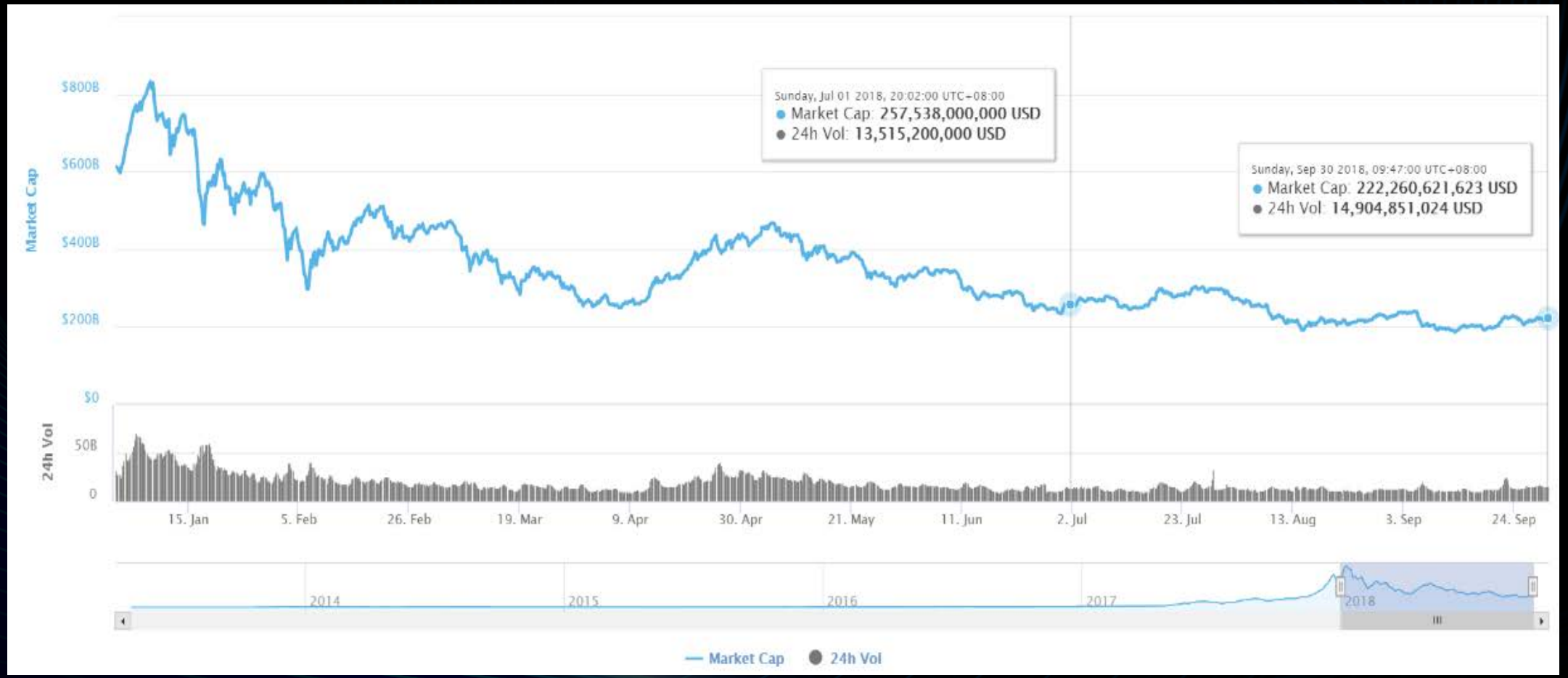
1/7

ICO占全球公开市场融资的比例

数据来源: coinmarketcap.com @ 2018.09.11

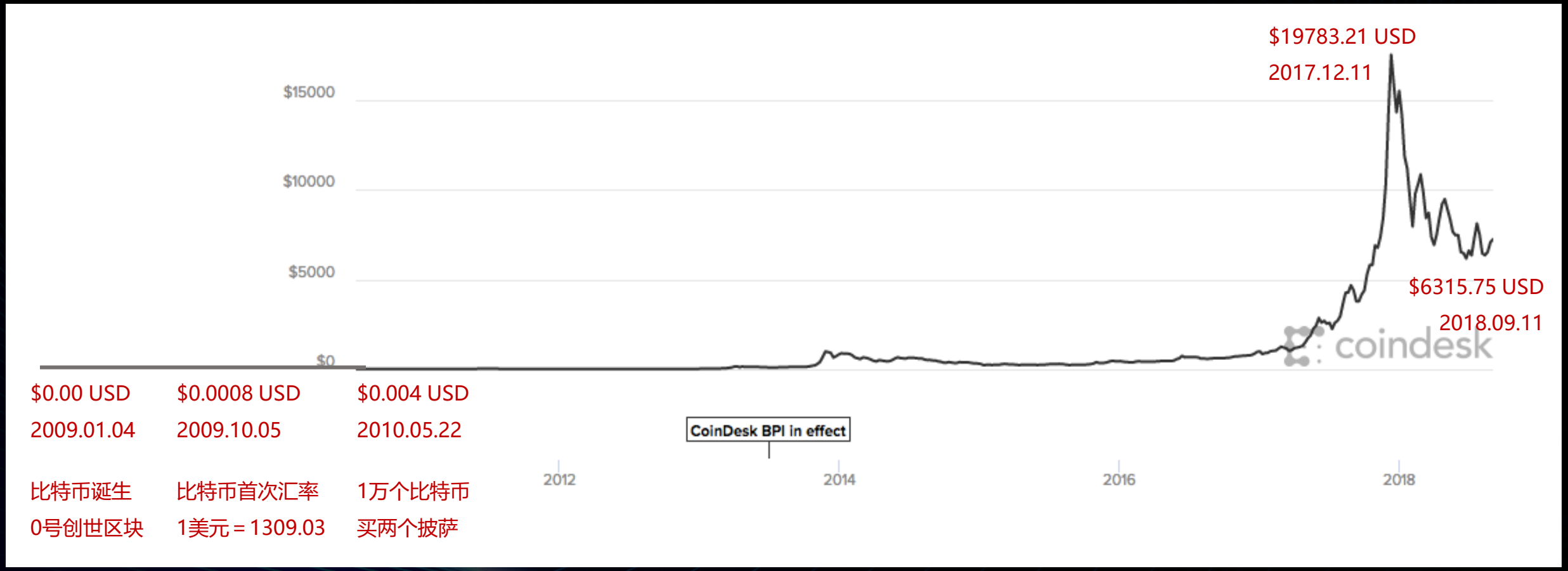
全球“虚拟货币”市值持续低迷

全球“虚拟货币”从今年初开始持续下跌，最近三个月（7-9月）市值下跌了14%，参与者多数保持观望。



（数据来源：coinmarketcap.com @2018.09.30）

比特币



$0.00 USD
2009.01.04
比特币诞生
0号创世区块

$0.0008 USD
2009.10.05
比特币首次汇率
1美元 = 1309.03

$0.004 USD
2010.05.22
1万个比特币
买两个披萨

佛系正见

佛学角度看区块链 - 八识

眼识 耳识 鼻识 舌识 身识 意识	区块链应用
末那识	智能合约
阿赖耶识	区块链公链

阿赖耶识与区块链的共同点：

不增不减、至高理性、由某种特定规则决定

佛学角度看区块链 - 六道

人道 —— “韭菜”
畜牲道 —— 币圈
修罗道 —— 链圈
饿鬼道 —— 黑客
地狱道 —— 归零
天道 —— 无币区块链

佛学角度看区块链 - 四圣谛

无常 —— 区块链的P2P网络

无我 —— 虚拟货币（UTXO）
区块链P2P网络

皆苦 —— 有漏洞；）

涅槃 —— 洞悉区块链的本源

区块链是什么

- 是加密数字货币
- 是全球化的分布式账本
- 也是世界计算机

比特币为什么会被创造出来

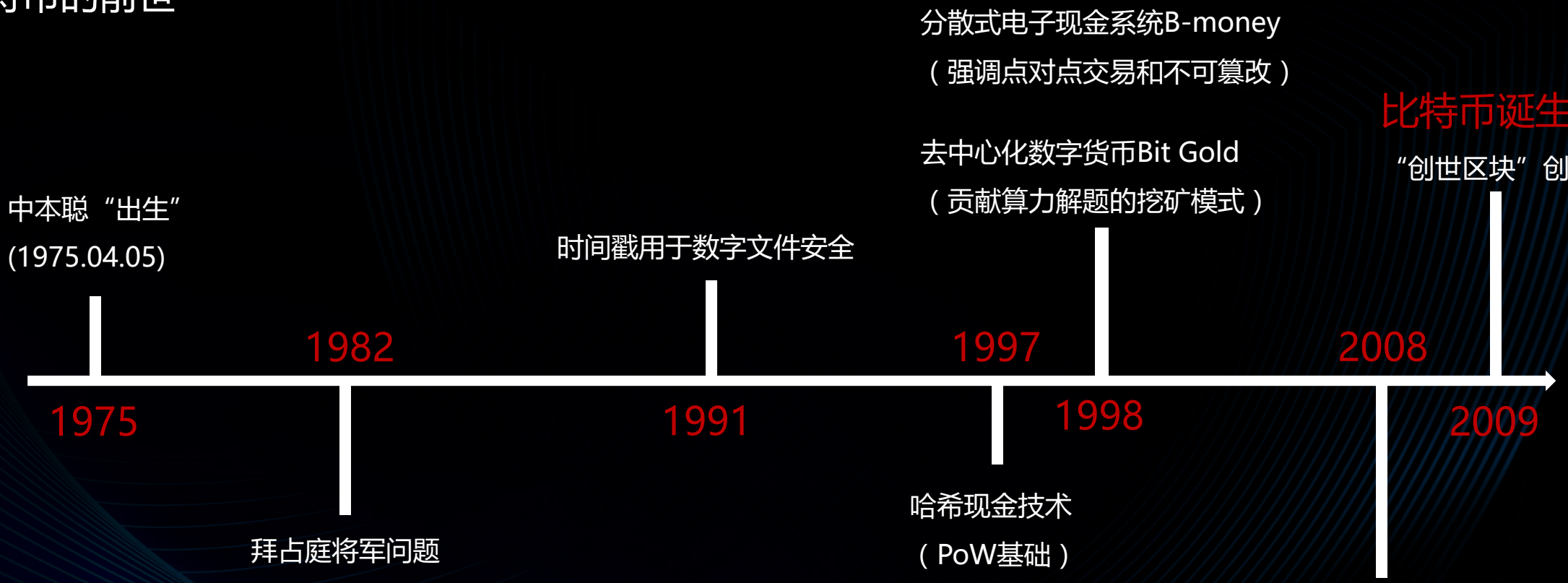
- 金融衍生品泛滥所导致的全球性金融危机
- 全球货币大放水所导致的通货膨胀
- 华尔街金融工作者的腐败现象

2008年来自金融领域的三大影响，催生了比特币

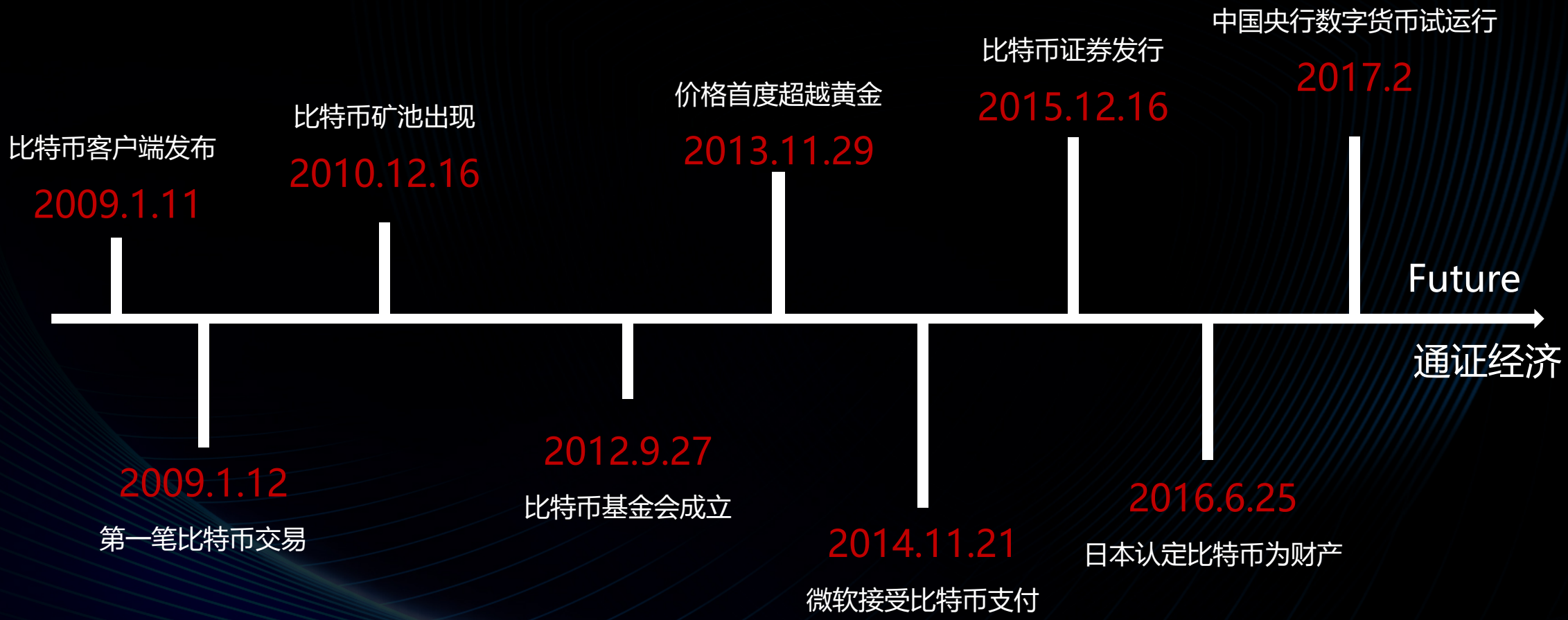
中本聪留在比特币创世区块中的一句话

“TheTimes 03/Jan/2009 Chancellor on brink of secondbailout for banks”

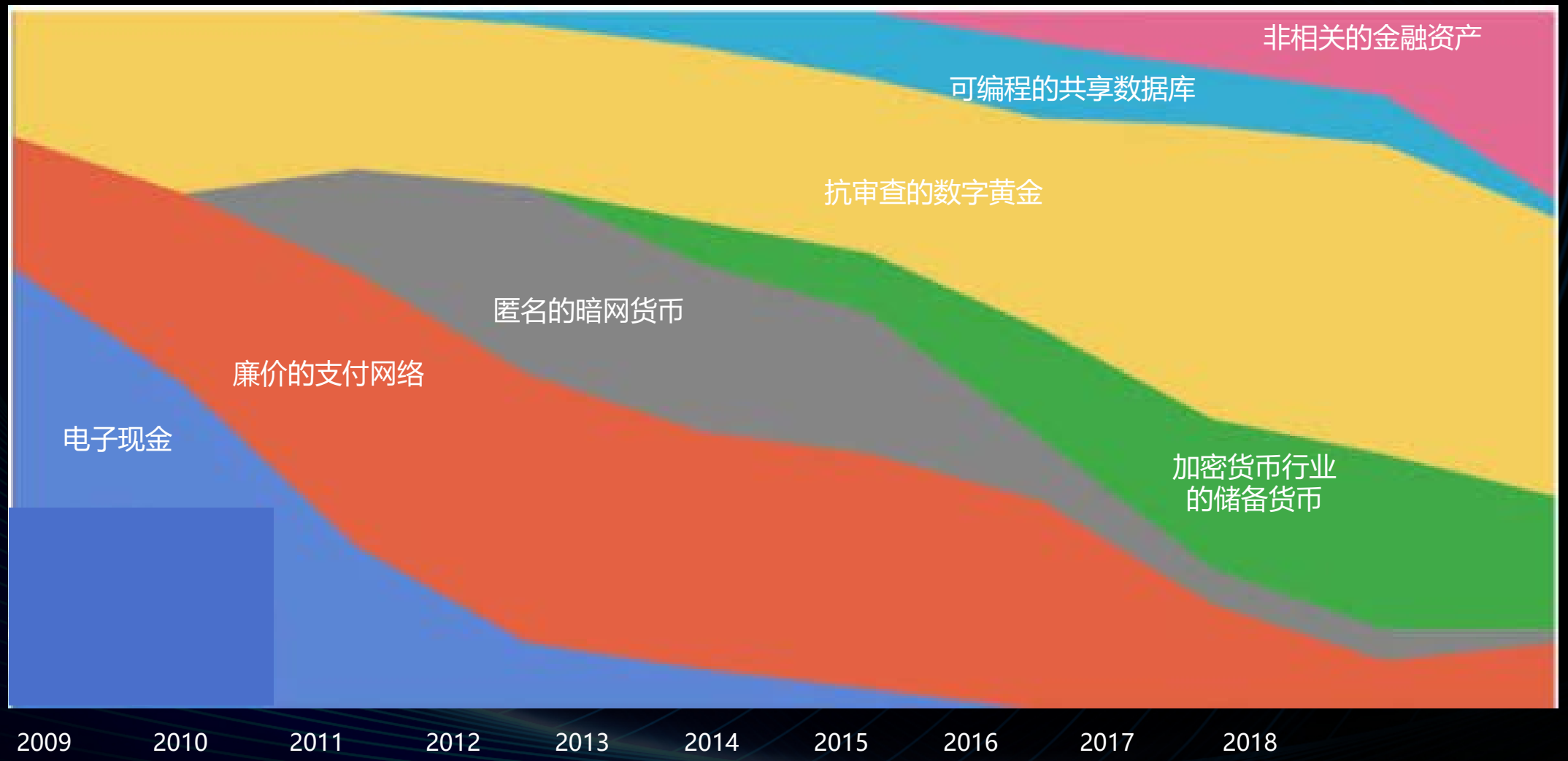
比特币的前世


中本聪“出生”
(1975.04.05)
1975
1982
拜占庭将军问题
（多数共识）
密码学网络支付系统
（为比特币提供隐私安全的雏形）
时间戳用于数字文件安全
1991
分散式电子现金系统B-money
（强调点对点交易和不可篡改）
1997
哈希现金技术
（PoW基础）
去中心化数字货币Bit Gold
（贡献算力解题的挖矿模式）
1998
2008
比特币白皮书发布
<Bitcoin: A Peer-to-Peer
ElectronicCash System>
比特币诞生
“创世区块”创建
2009

比特币的今生

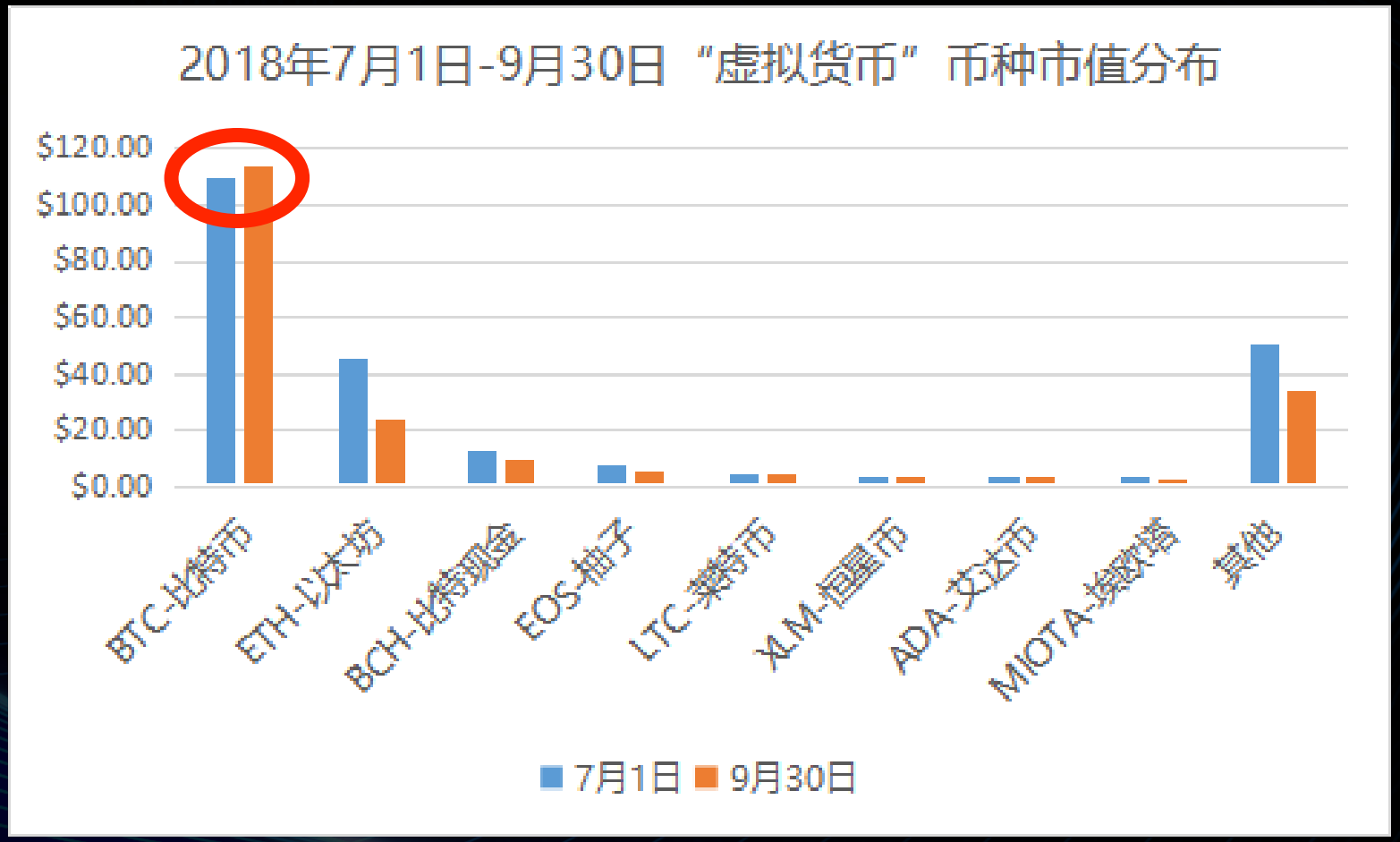

比特币客户端发布
2009.1.11
比特币矿池出现
2010.12.16
价格首度超越黄金
2013.11.29
比特币证券发行
2015.12.16
中国央行数字货币试运行
2017.2
Future
通证经济
2009.1.12
第一笔比特币交易
2012.9.27
比特币基金会成立
2014.11.21
微软接受比特币支付
2016.6.25
日本认定比特币为财产

比特币的七个愿景



图片来源：哈希派

全球持币者拥抱比特币过冬



（数据来源：coinmarketcap.com @2018.09.30）

区块链基础

区块链的基础技术架构


应用层
脚本代码
智能合约
DAPP
激励层
发行机制
分配机制
惩罚机制
共识层
POW
POS
POA
DPOS
BFT
Lottery
POW + POS
POS + BFT
DPOS + BFT
Lottery + POS + BFT
区块高度确认（最长链确认）
最重链接确认（最多引用确认）
网络层
对等网络
进出机制
传播机制
数据层
区块数据
链式结构/图结构
数字签名
哈希函数
Merkle证明
非对称加密

应用层

应用层

脚本代码 | 智能合约 | DAPP

ICO Token 智能合约

SOLIDITY CONTRACT SOURCE CODE | CONTRACT BYTE CODE

```
1  pragma solidity 0.4.8;
2  contract Token {
3    mapping (address => uint) public balancesOf;
4    address public owner;
5    function Token() {
6      owner = msg.sender;                    1. 合约代码
7      balancesOf[msg.sender] = 10000;
8    }
9
10   function transfer(address _to, uint _value) {
11     if (balancesOf[msg.sender] < _value) throw; //避免转移出去的代币超过
12     if (balancesOf[_to] + _value < balancesOf[_to]) throw; //避免自己调
13 用  balancesOf[msg.sender] -= _value;
14     balancesOf[_to] += _value;
15   }
16
17   function mint(uint _amount) {
18     balancesOf[owner] += _amount;
19   }
20 }
21
```

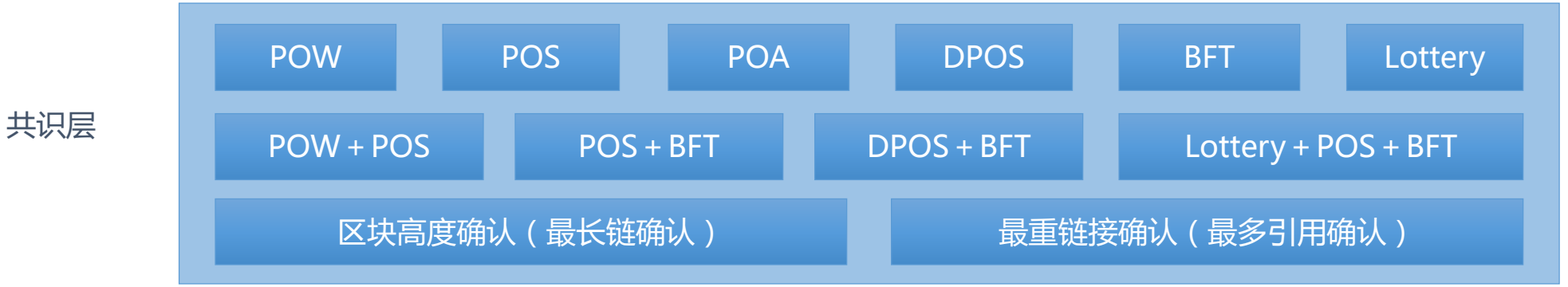
游戏 DAPP

2018年9月4日 600个ETH

激励层

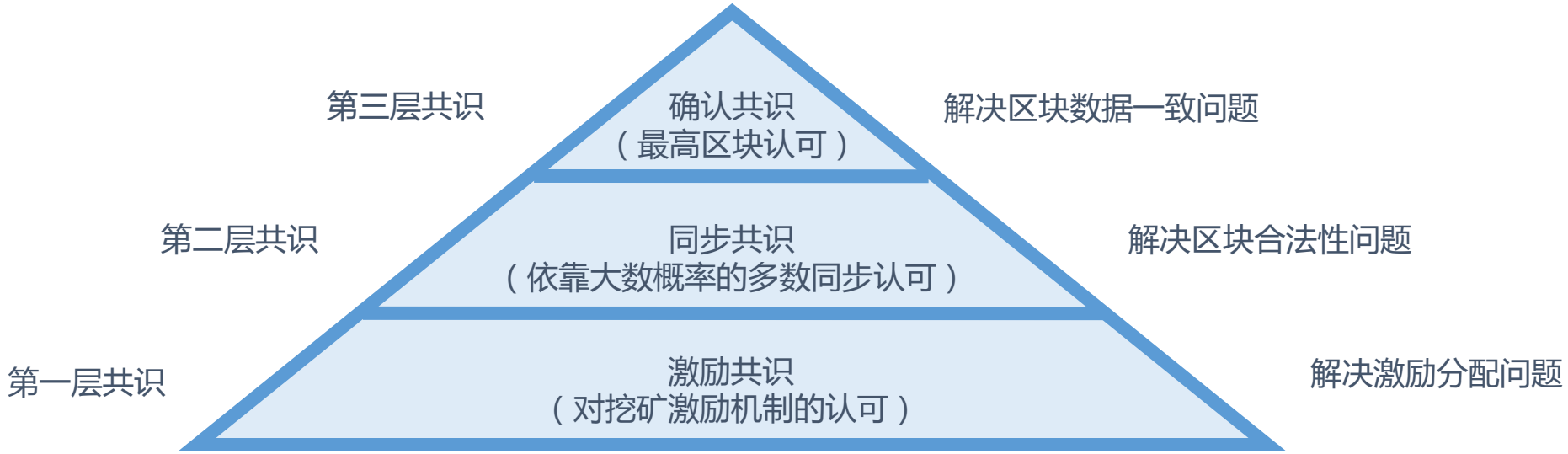
- 发行机制：通过制定数字货币或Token的发行及投放方式，来促进项目的推广、宣传和获取新用户
- 分配机制：通过制定挖矿奖励（POW）、抵押奖励（POS、DPOS）、交易手续费，来维持系统的持续运行
- 惩罚机制：通过制定惩罚机制，来提高作恶者或恶意节点的侵犯他人利益的成本

共识层

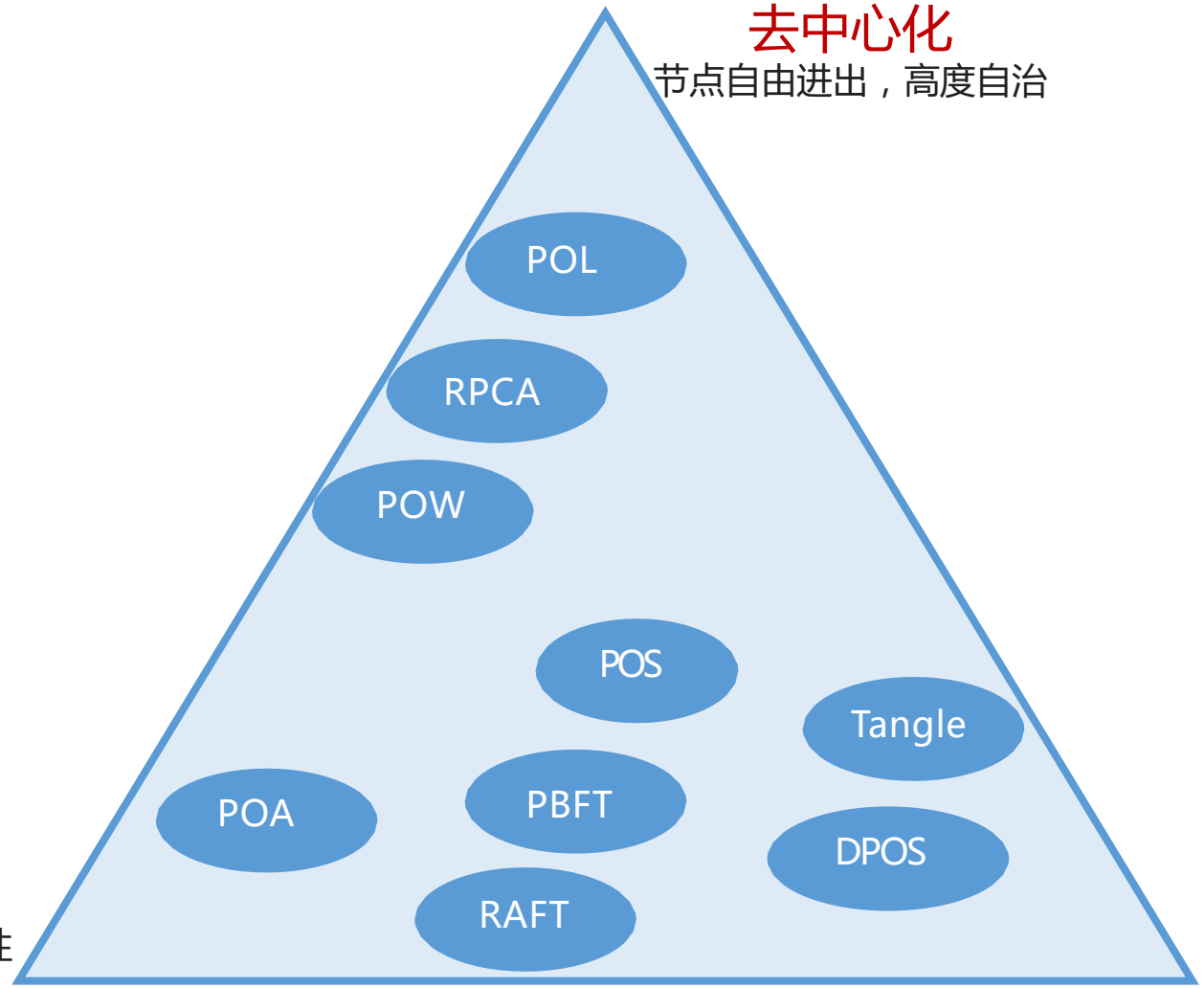


区块链的共识含有多个层次。比如：参与POW不仅仅是>50%的多数共识

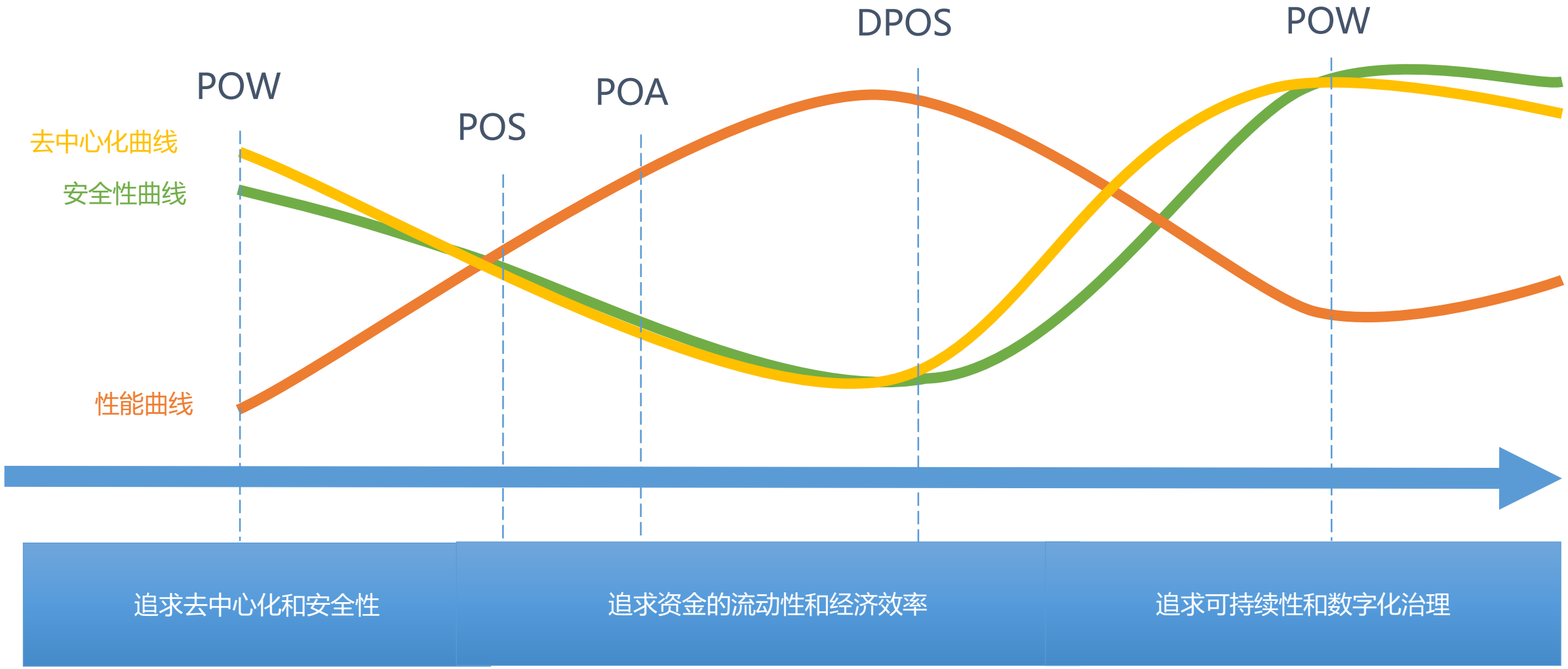
POW的三层共识



共识层面的不可能三角


去中心化
节点自由进出，高度自治
POL
RPCA
POW
POS
Tangle
POA
PBFT
DPOS
RAFT
安全性
机密性、完整性、可用性
可扩展性
高吞吐、高容量、低延迟

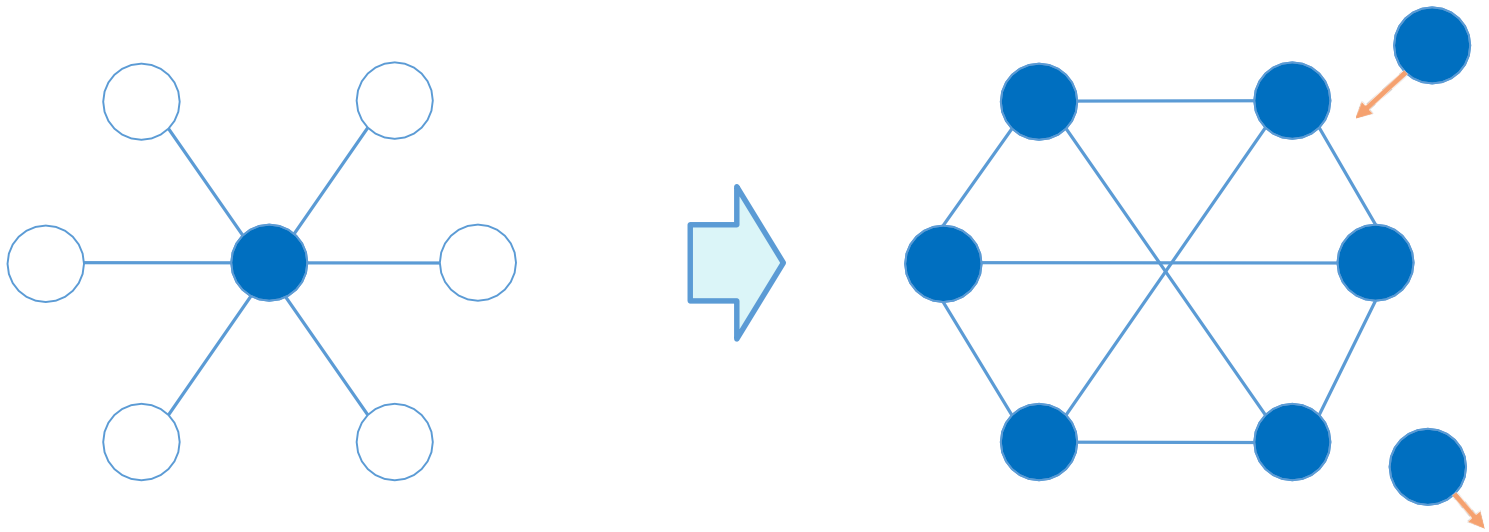
共识机制的更迭


POW
POS
POA
DPOS
POW
去中心化曲线
安全性曲线
性能曲线
追求去中心化和安全性
追求资金的流动性和经济效率
追求可持续性和数字化治理

网络层

网络层

对等网络 | 进出机制 | 传播机制

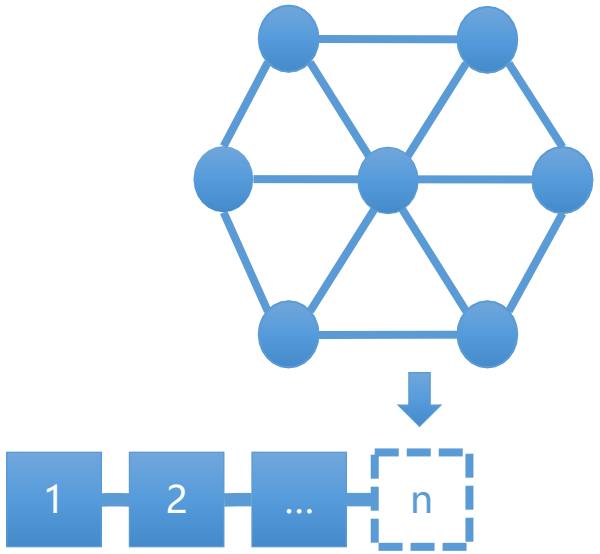


集中式　　对等式

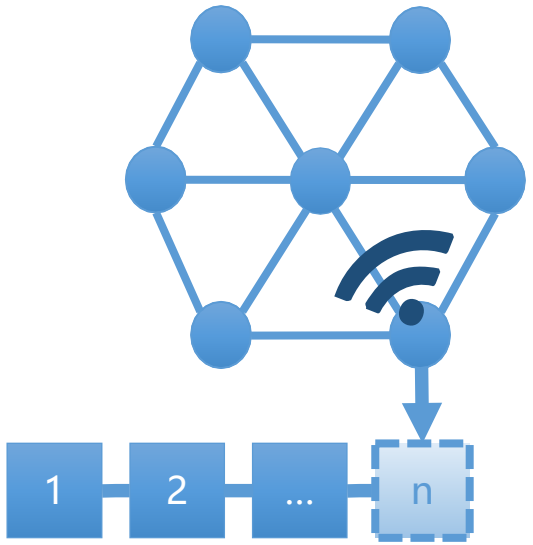
P2P网络：适合大规模节点组网，实现节点动态加入/退出，共同维护保持网络稳定

技术要点：节点发现、网络连接、交易广播、区块广播、下载区块数据等

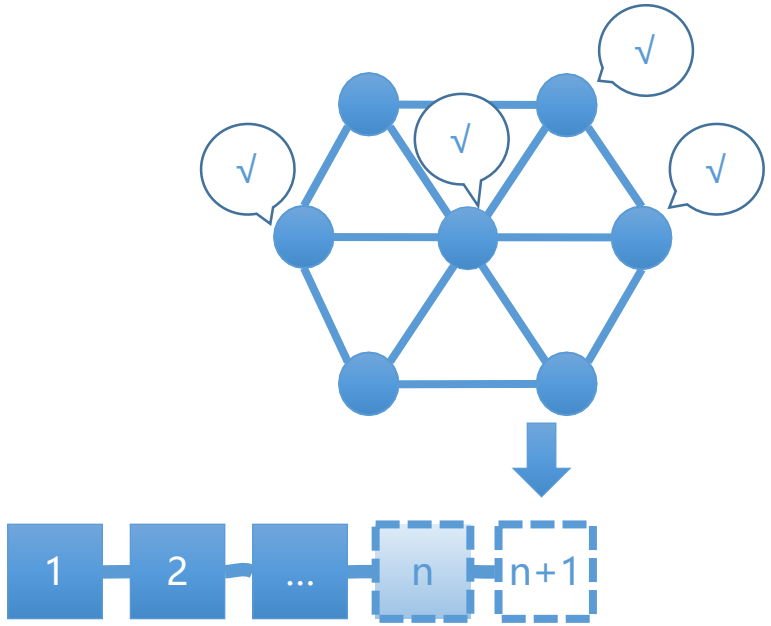
网络层 - 广播机制



1. 全网各节点挖掘区块高度N的区块



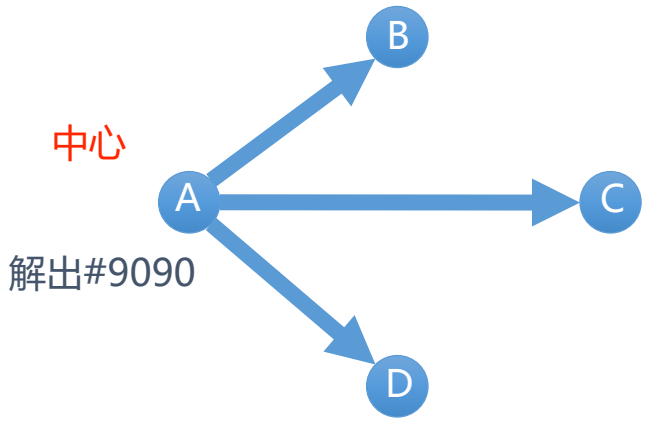
2. 某个节点挖出后打包区块广播（区块高度N）



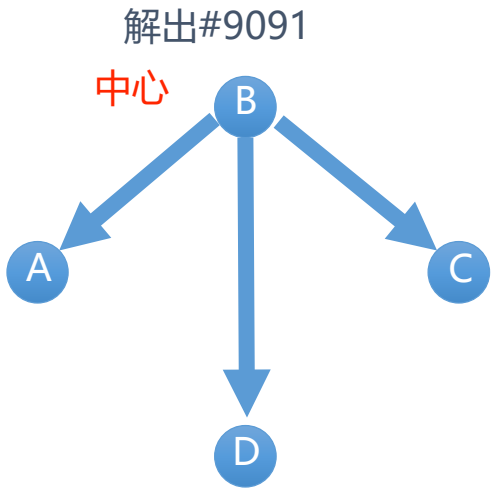
3. 全网节点收到区块高度为N的区块广播后，停止挖掘区块高度N的区块，并根据N号区块的信息，挖掘N+1号区块

网络层 - 关于去中心化

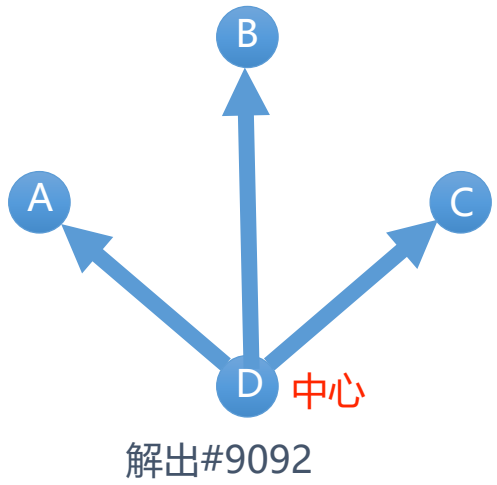
区块链的去中心化，事实上是 时分中心化



区块高度：#9090



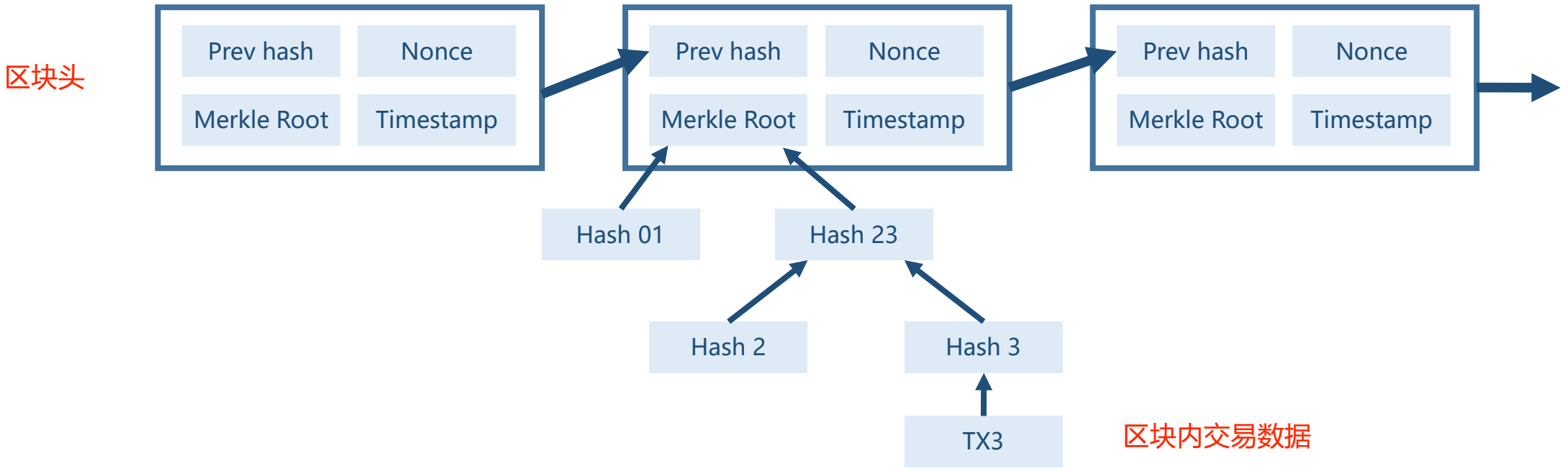
区块高度：#9091



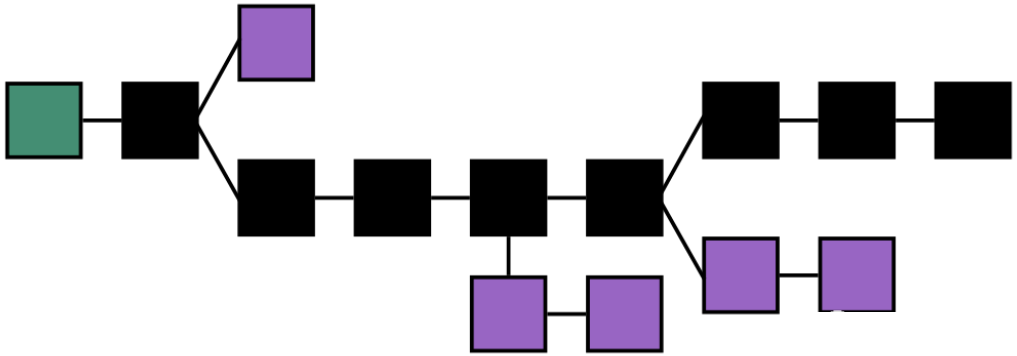
区块高度：#9092

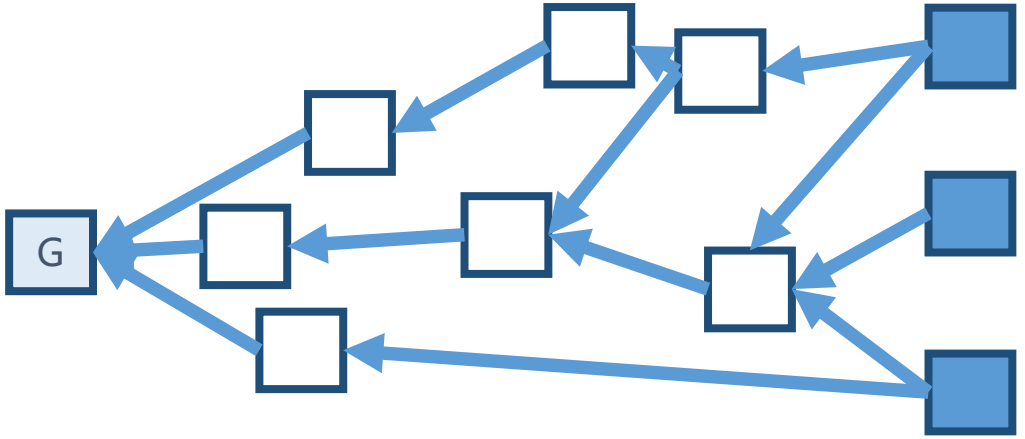
数据层

比特币的区块结构



数据层 - 数据结构的演进 - 从链向图





数据结构的变化

块链结构

图结构（有向无环图）DAG

区块确认机制的变化

根据时间确认：根据区块高度进行确认

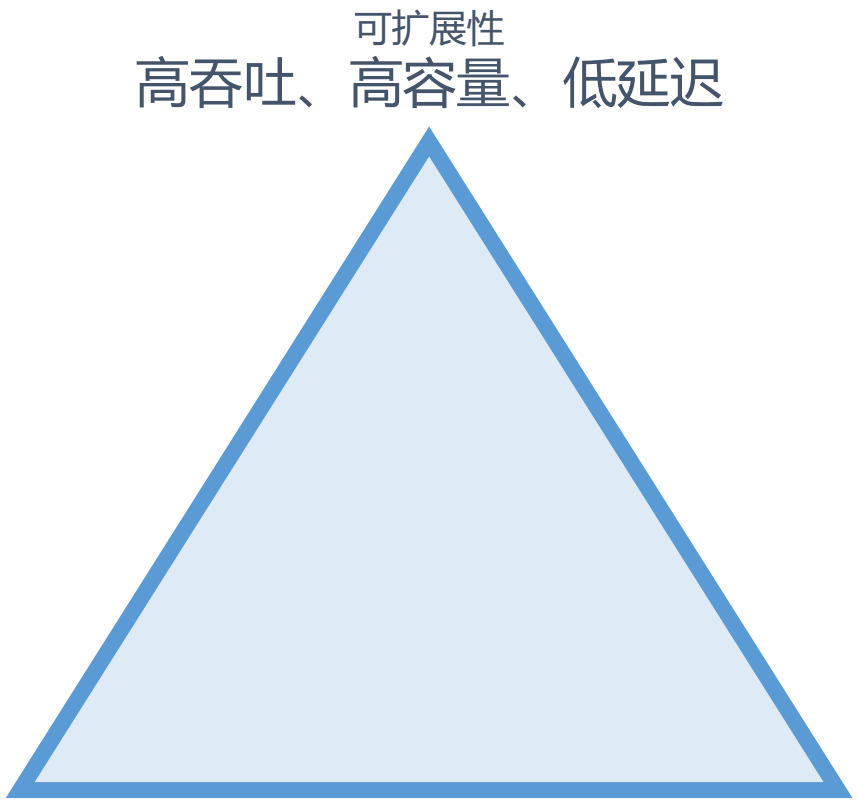
根据关联度确认：根据区块所被引用的次数确认

突破不可能三角

Algorand——突破区块链不可能三角

- Algorand是由图灵奖得主，来自MIT的Silvio Micali教授于2016年提出的一个区块链协议。

- Algorand通过以下方式解决不可能三角的问题：
  - 委员会共识，实现可扩展性
  - 参与者更换，解决去中心问题
  - 加权用户，解决安全性问题
  - 加密抽签，解决安全性问题



Algorand的共识机制（BA⋆ 拜占庭协议）

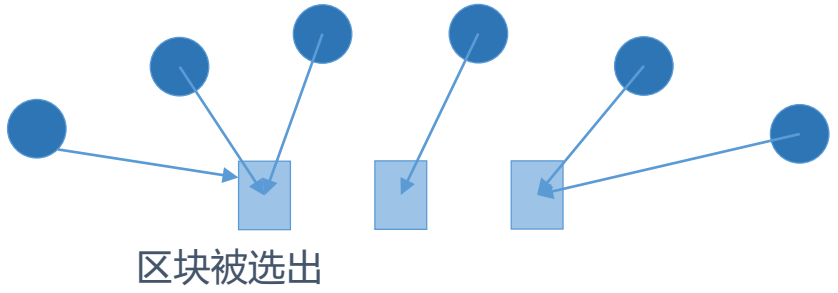
- 共识机制采用：三阶段共识，相当于一个随机选择投票人 + 两阶段投票机制。

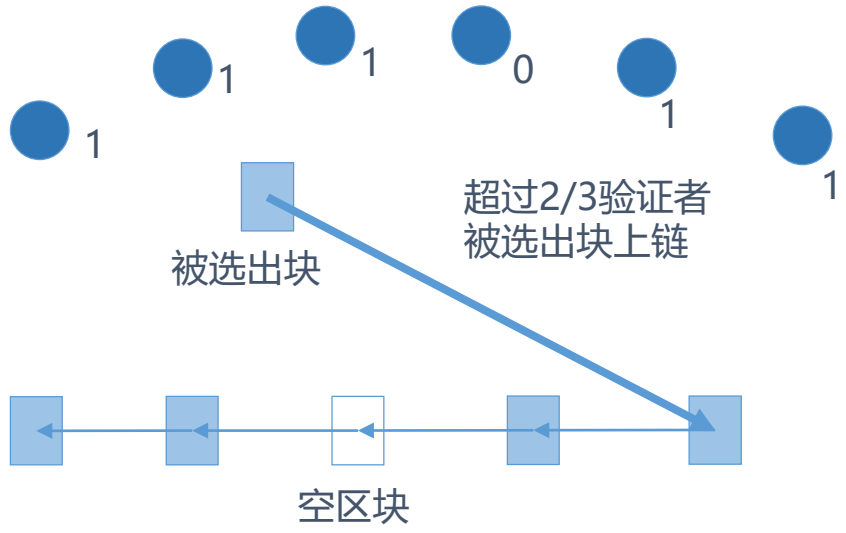
1. 角色确认阶段：对一个大规模网络，每个共识轮次开始时，每个节点先采用VRF（可验证的随机函数，verifiable random functions)生成一个凭证，用该凭证随机选出本轮参与共识的节点，被称为”验证者"，而其中凭证值最小的被选为“提议者”

2. 分级共识阶段：提议者负责组装本轮的候选区块，然后由投票者对本轮的领导者达成共识，也同时确认本轮收到的候选区块；

3. 二元拜占庭阶段：验证者对候选区块投票，即要么接受该候选区块（认为该区块没有问题），要么不接受该候选区块（认为该区块有错误，比如双花，不接受该区块，替换为空区块）

在每一阶段中的每一个子步骤，Algorand可能使用完全不同的“验证者”。每个步骤需要超过2/3的验证者才能通过。





区块链安全风险

从2011年至今，全球区块链安全事件损失总金额
损失超过28亿美元

流通环节

数字货币交易所安全事件损失

13.44亿 美元

单起事件数百万 - 数亿

造币发币环节

智能合约安全事件损失

12.4亿 美元

单起事件数千万 - 数十亿

流通环节

引发用户信任类安全事件损失

1.73亿 美元

单起事件百万 - 数千万

发币环节

矿工|节点安全事件损失

6328万 美元

单起事件数十万 - 数千万

流通环节

共识机制安全事件损失

3100万 美元

单起事件上千万

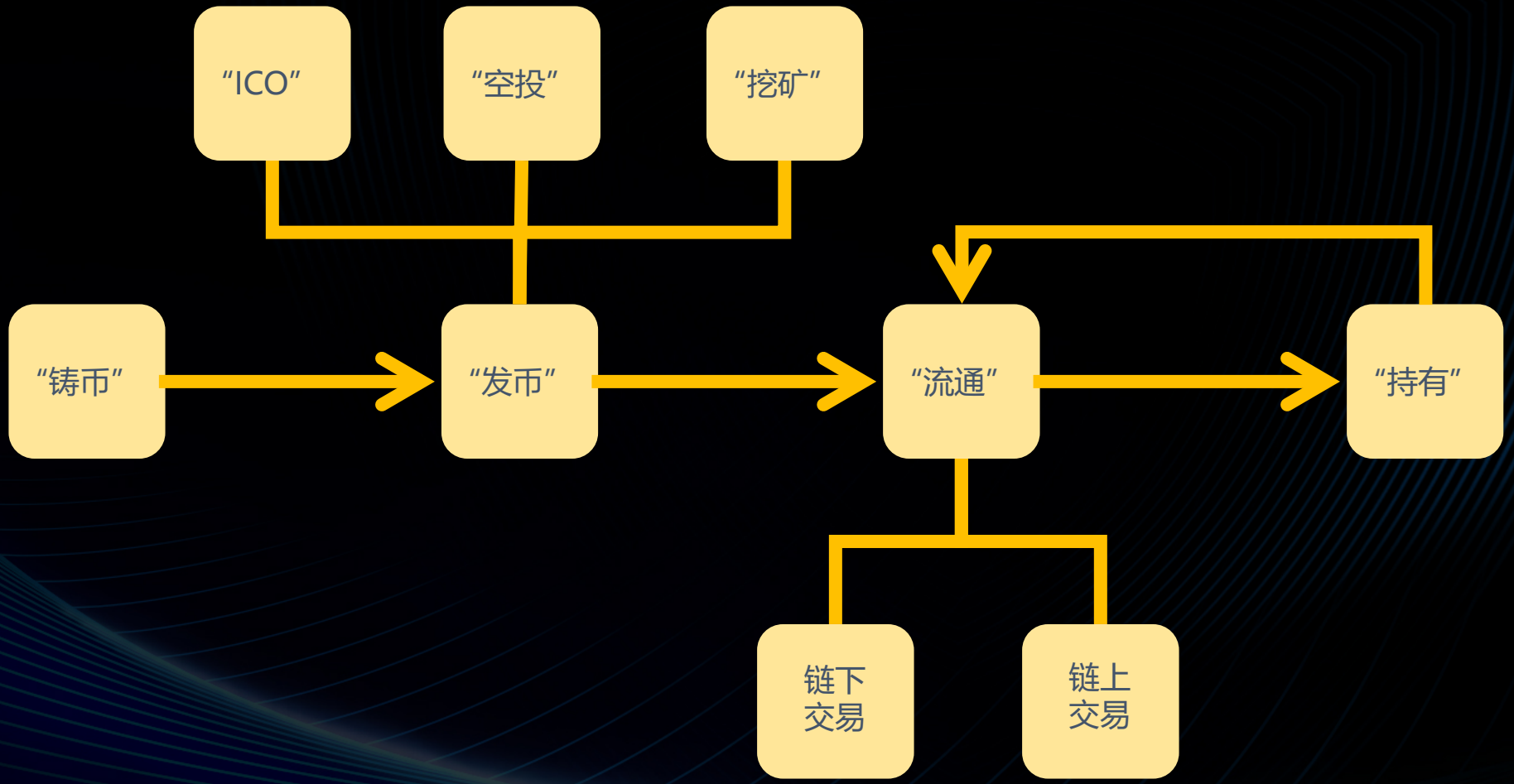
持有环节

货币存储类安全事件损失

1260万 美元

单起事件上百万

加密数字货币完整生命周期



区块链所面临的安全威胁

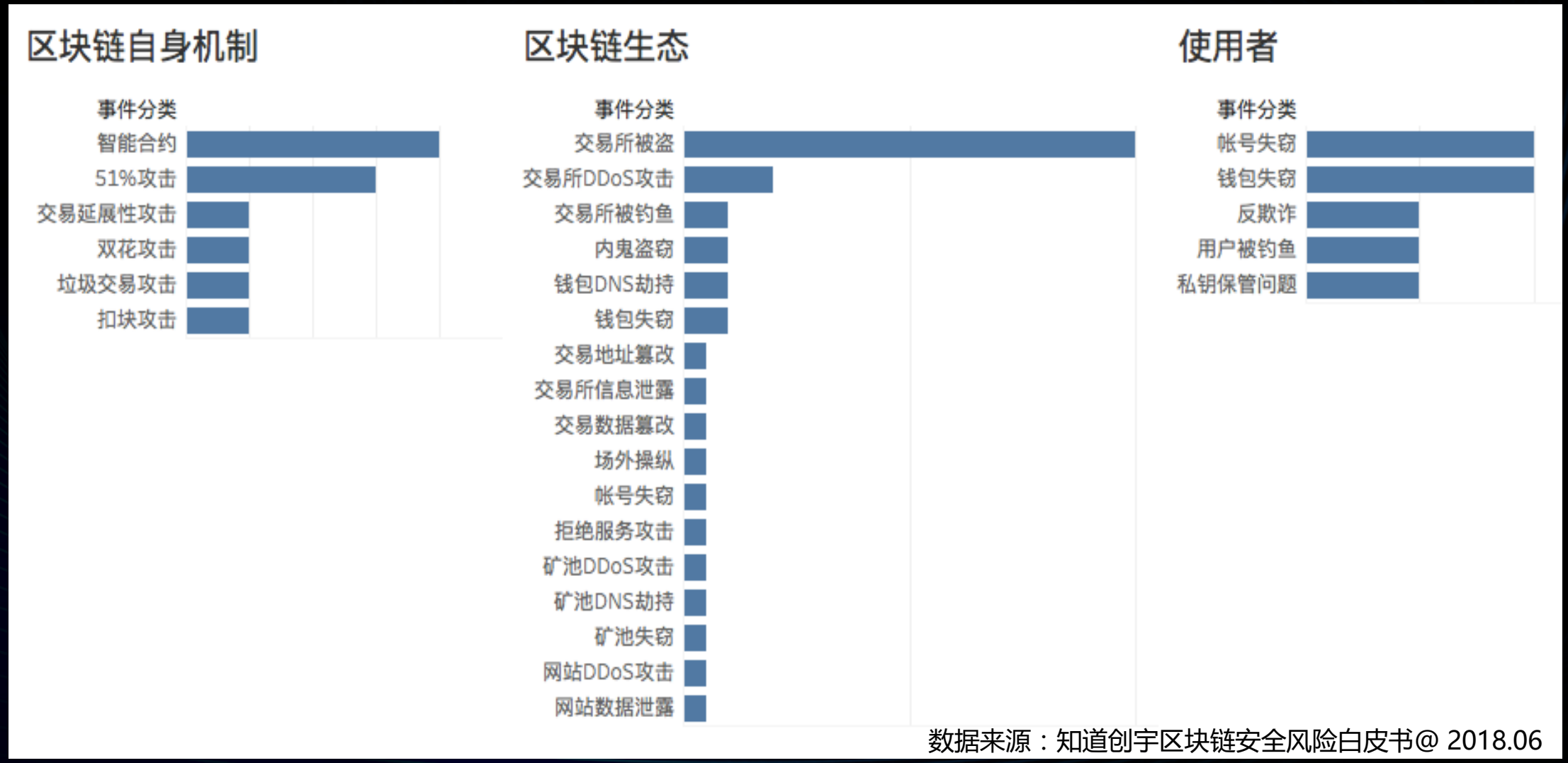
智能合约

基础协议

共识机制

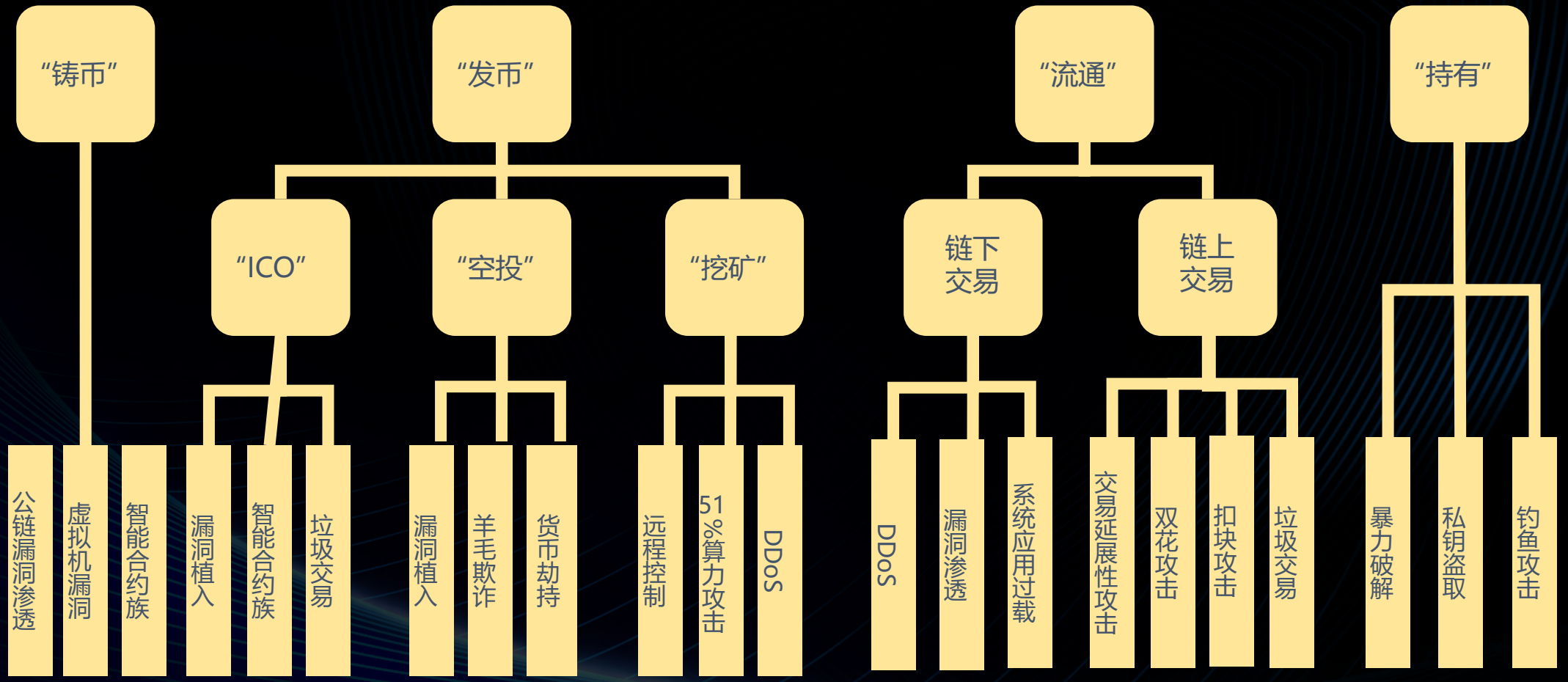
外部引用

区块链面临的安全威胁点



数据来源：知道创宇区块链安全风险白皮书@ 2018.06

区块链易被攻击的脆弱环节



智能合约的安全问题

开源项目 Decentralized Application Security Project (DASP) 统计并公布了智能合约的十大安全问题：

1. 递归调用漏洞
2. 访问控制
3. 整数溢出
4. 未检查的底层调用
5. 拒绝服务
6. 错误随机性
7. 事务顺序依赖
8. 时间戳依赖
9. 短地址攻击
10.(未知漏洞)

智能合约 - 递归调用问题

未对 call.value() 函数做严格限制，导致递归调用

call.value() 函数特性：

- 发起转账
- 发送所有可用 gas
- 可调用目标合约的函数

智能合约 - 递归调用问题案例 - DAO

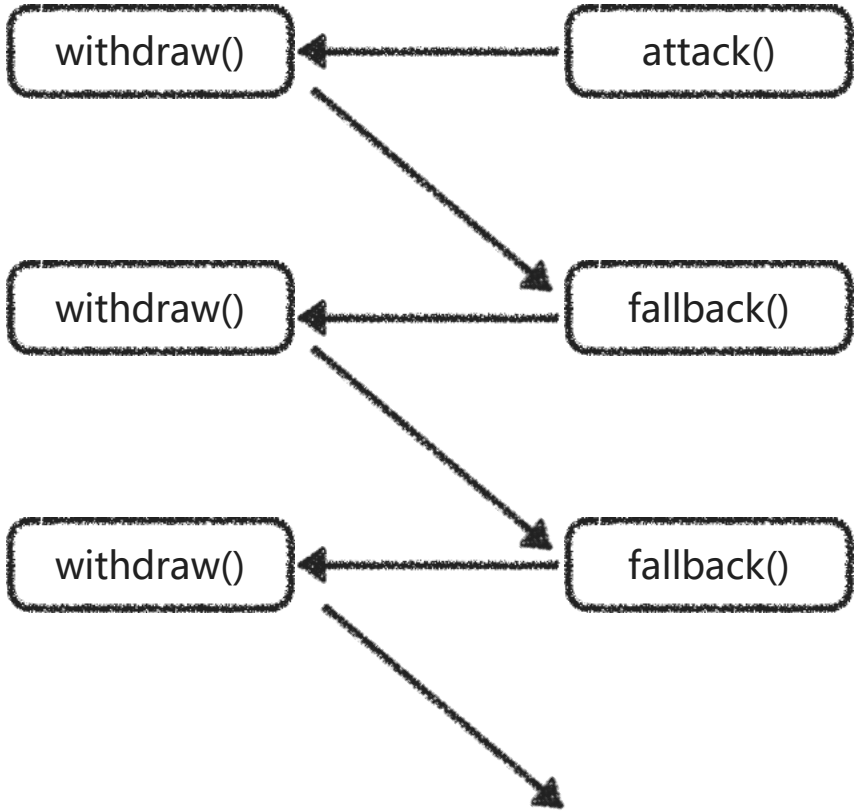
案例：The DAO - 超过 5000W 美元经济损失

```
function withdraw(address _to, uint256 _amount)
    public payable {
    //...
    _to.call.value(_amount)();
}
```

DAO demo 代码片段

```
function () public payable {
    addr.withdraw(this, 10000);
}
function attack() public payable {
    addr.withdraw(this, 10000);
}
```

攻击合约



递归调用

智能合约 - 访问控制问题

访问控制：某些函数只能由特定的用户才能调用。

初始化赋权函数没有检查是否已经被调用

```
function initContract() public {
    owner = msg.sender;
}
```

智能合约 - 访问控制问题案例

案例：Owner 构造函数书写错误

构造函数特性

- 1. 部署合约时调用
- 2. 不写入区块链中

普通函数特性

- 1. 任意调用
- 2. 代码写入区块链中

```
contract Owner {
    address own;

    // function Owner() {
    function owner() {
        own = msg.sender;
    }
}
```

书写错误导致构造函数变为普通公有函数

智能合约 - 整数溢出问题

```
> 255 + 2
> 1
```

```
> 10 - 11
> 115792089...39935
```

无符号整数上溢和下溢

```
function withdraw(uint _amount) {
    require(balances[msg.sender] - _amount > 0);
    msg.sender.transfer(_amount);
    balances[msg.sender] -= _amount;
}
```

未检查整数下溢，转账后余额可以变为一个极大的数

智能合约 - 整数溢出问题案例 - BEC

案例：BEC 代币整数溢出 - 造成 BEC 价值归零

From 0x09a34e01fbaa49f... To 0xb4d30cac5124b4... for
57,896,044,618,658,100,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.792003956564819968 ERC20 (BEC)

From 0x09a34e01fbaa49f... To 0x0e823ffe0187275... for
57,896,044,618,658,100,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.792003956564819968 ERC20 (BEC)

BEC 代币计划发放
7 000 000 000 枚

整数溢出，凭空造币
发放 578960..... 代币

智能合约 - 拒绝服务问题

造成合约意外停止工作的原因：

- 达到 gas 上限
- 超过 1024 次调用
- 意外抛出异常

可能造成

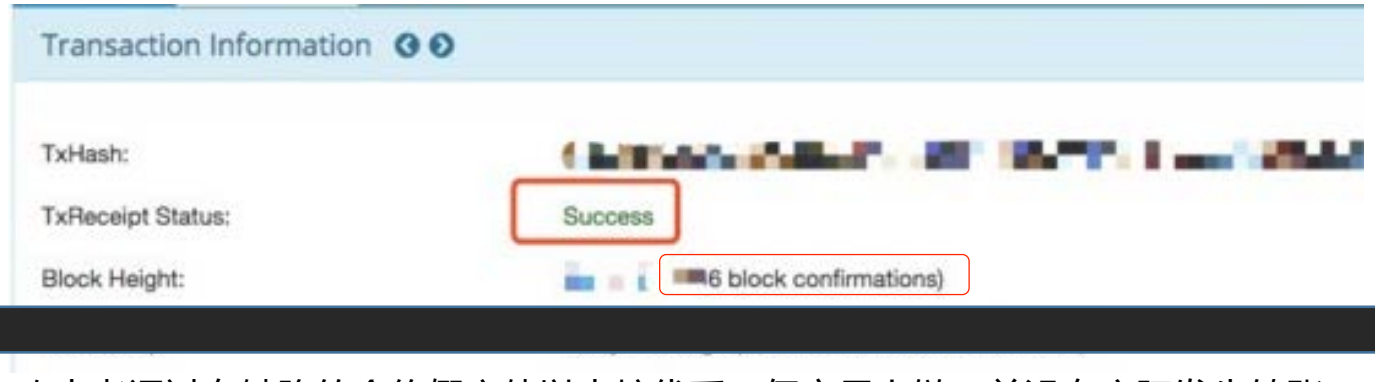
- gas 大量消耗
- 合约无法工作

```
function foo() public {
    for(uint256 i = 0; i < 100000, i++) {
        // heavy code
    }
}
```

10W次循环消耗大量 gas

以太坊代币“假充值”导致交易所被收割

以太坊代币“假充值”漏洞影响面非常之广，影响对象至少包括：中心化交易所、中心化钱包、代币合约等。据不完全统计，仅代币合约就有超过 3600 份存在“假充值”漏洞风险，其中不乏知名代币。



攻击者

有缺陷的合约

交易所

攻击者通过有缺陷的合约假充值以太坊代币，仅交易上链，并没有实际发生转账

```
function transfer(address _to, uint256 _value) returns (bool success) {
    if (balances[msg.sender] >= _value && _value > 0) {
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        Transfer(msg.sender, _to, _value);
        return true;
    } else { return false; }
}
```

某些代币合约的 transfer 函数对转账发起人(msg.sender)的余额检查，采用 if 判断方式，当 balances[msg.sender] < _value 时进入 else 逻辑部分并 return false，最终并没有抛出异常。

- 以太坊代币交易回执中 status 字段是 0x1(true) 还是 0x0(false)，取决于交易事务执行过程中是否抛出了异常（比如使用 require/assert/revert/throw 等机制）。
- 当用户调用某些代币合约的 transfer 函数进行转账时，如果 transfer 函数正常运行未抛出异常，该交易的 status 即是 0x1(true)。

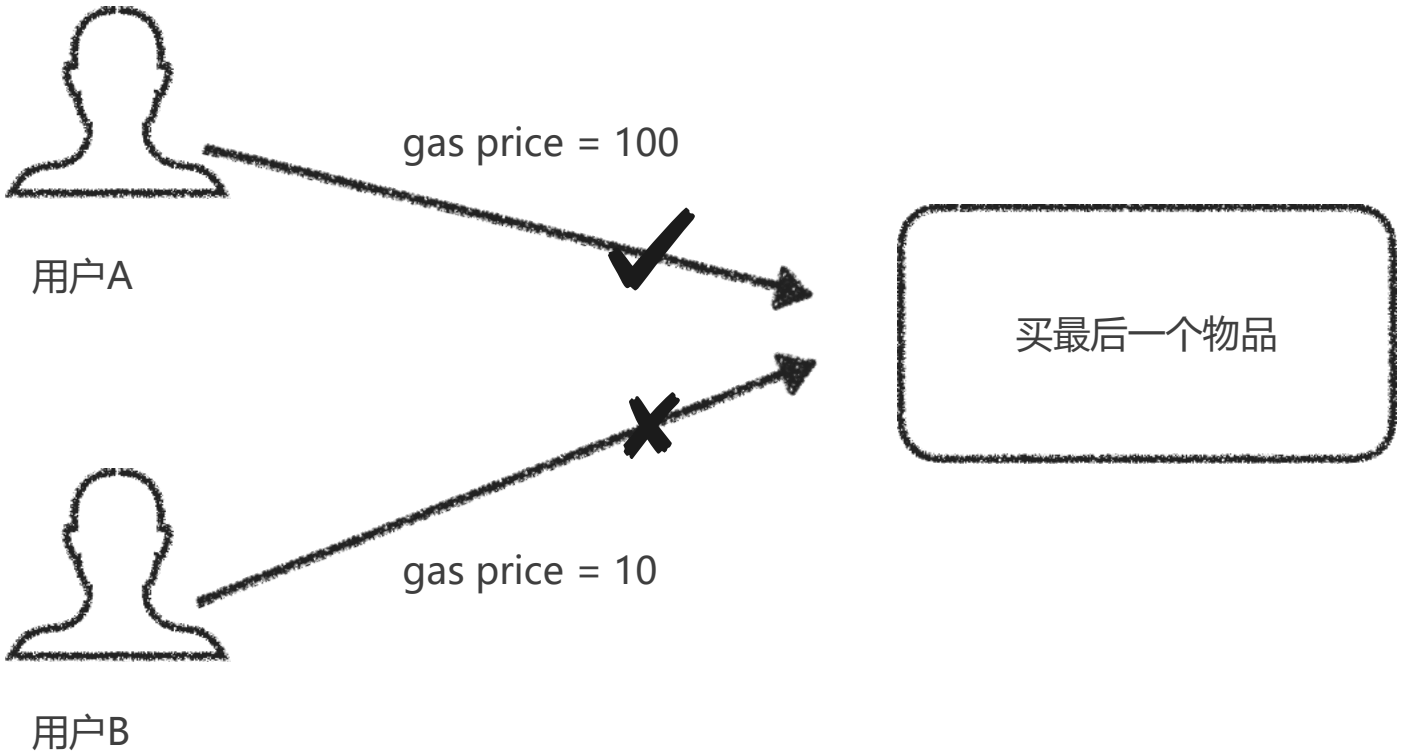
如果仅判断：
交易回执状态（TxReceipt Status）
和链上确认次数（Block Confirmation Number）

而忽视检查账户实际变化就会导致假充值的产生。

（实例图片来源：慢雾区）

智能合约 - 事务顺序依赖

「提前交易」，矿机优先处理 gas 价格更高的交易



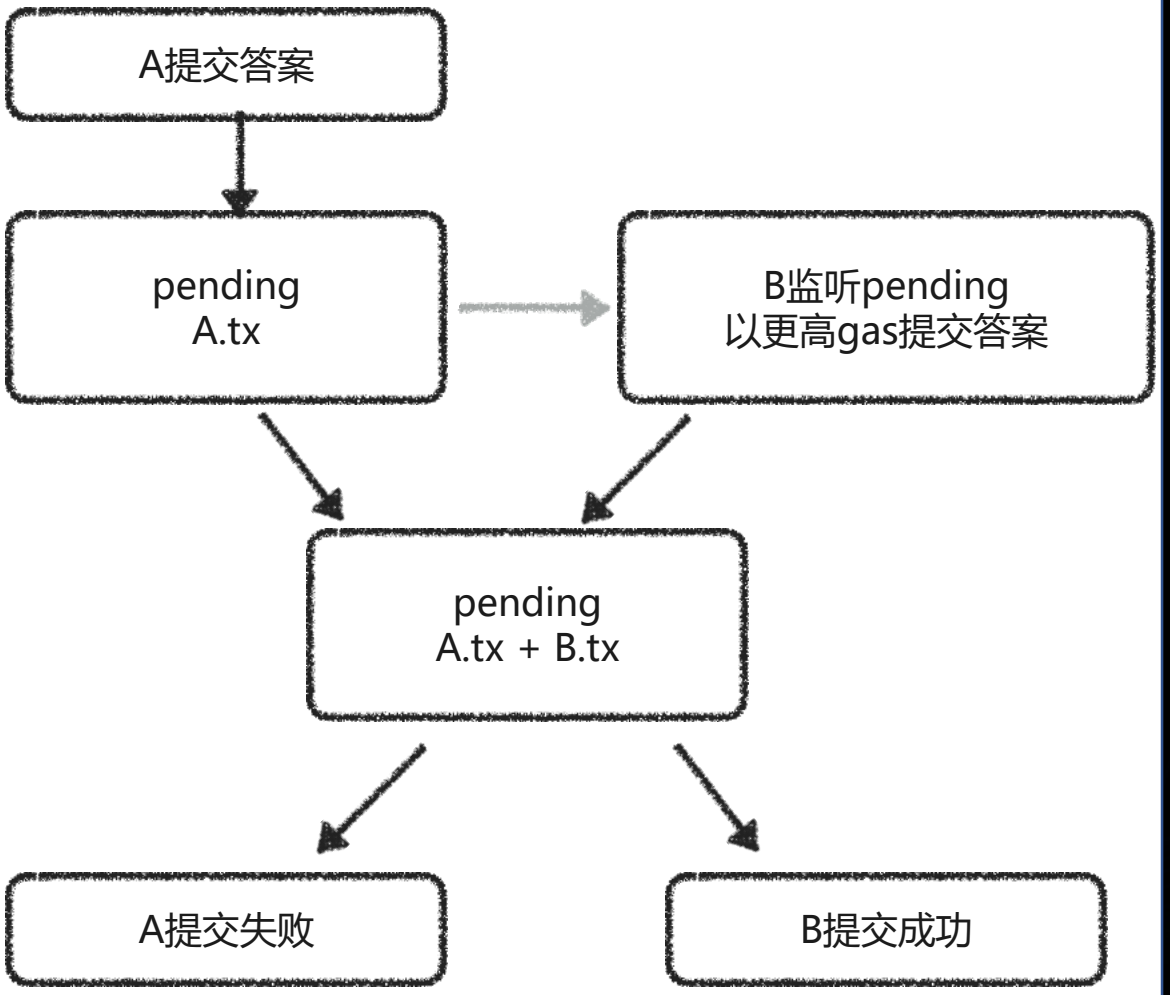
智能合约 - 事物顺序依赖案例

举例：猜数游戏

```
uint256 result = 0xabcd...;
function withdraw(uint256 _answer) {
    if (result == keccak256(_answer)) {
        msg.sender.send(1 ether);
        self_destory();
    }
}
```

第一个回答正确的人将获得奖励

提交交易会打破游戏公平性



来自网络协议和基础数据的安全威胁

网络协议

- 挖矿算法
- 加密协议
- P2P协议
- RPC服务

针对网络层的传输机制、验证制、P2P传输的数据包进行分析并进行漏洞挖掘。

基础数据

- 加密算法
- 数据存储
- 时间戳

针对数据层的加密算法、时间戳、数据存储方式进行分析和破解。

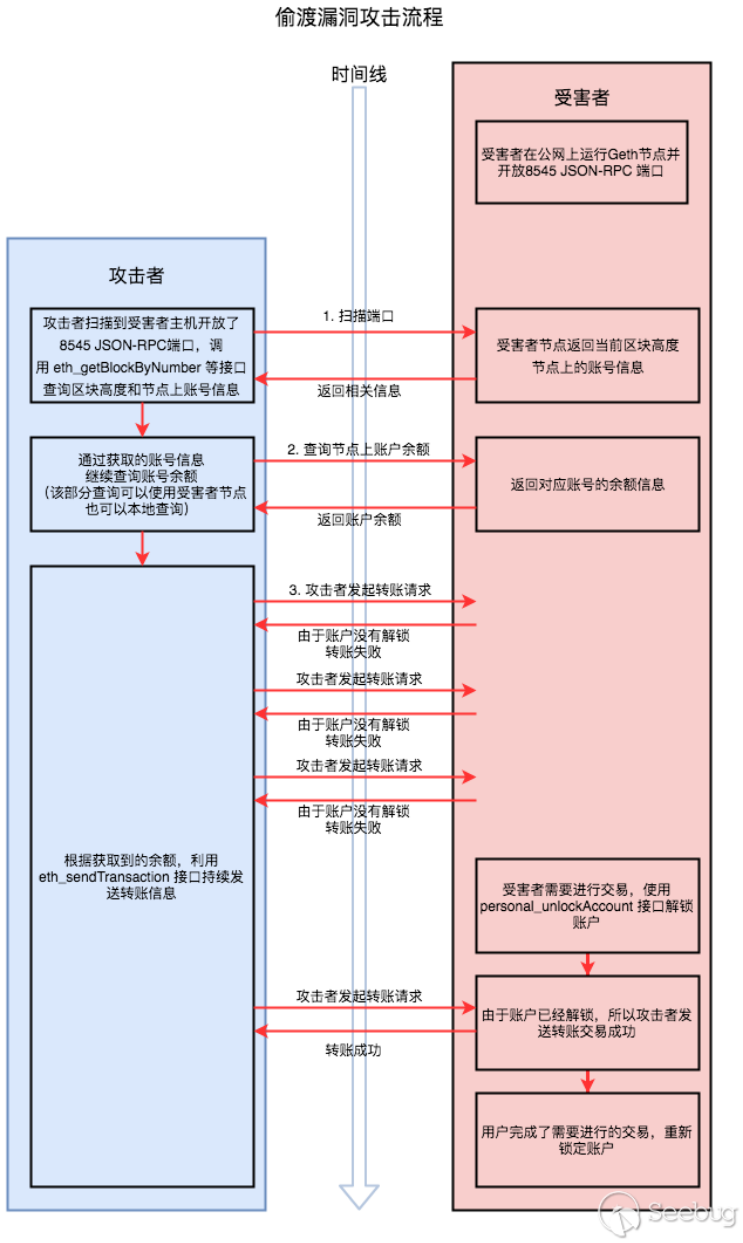
网络协议威胁案例—以太坊节点RPC漏洞

以太坊节点Geth/Parity RPC API 鉴权缺陷

被盗ETH	当前市值	被盗钱包数
47216	$ 15213939.52	3019

[Jun-29-2018 09:20:30 AM] 0x93fd2610eda7ea89860d4090f8789df63a635a7f被盗0.04380 Ether

—— “以太坊黑色情人节”




```
func (s *PrivateAccountAPI) UnlockAccount(addr common.Address, password string, durat
ion *uint64) (bool, error) {
    const max = uint64(time.Duration(math.MaxInt64) / time.Second)
    var d time.Duration
    if duration == nil {
        d = 300 * time.Second
    } else if *duration > max {
        return false, errors.New("unlock duration too large")
    } else {
        d = time.Duration(*duration) * time.Second
    }
    err := fetchKeystore(s.am).TimedUnlock(accounts.Account{Address: addr}, password,
 d)
    return err == nil, err
}
```

详情参考：金钱难寐，大盗独行——以太坊JSON-RPC接口多种盗币手法大揭秘（404区块链安全研究团队出品）
https://paper.seebug.org/656/

来自虚拟机的安全威胁案例—Redis - Lua

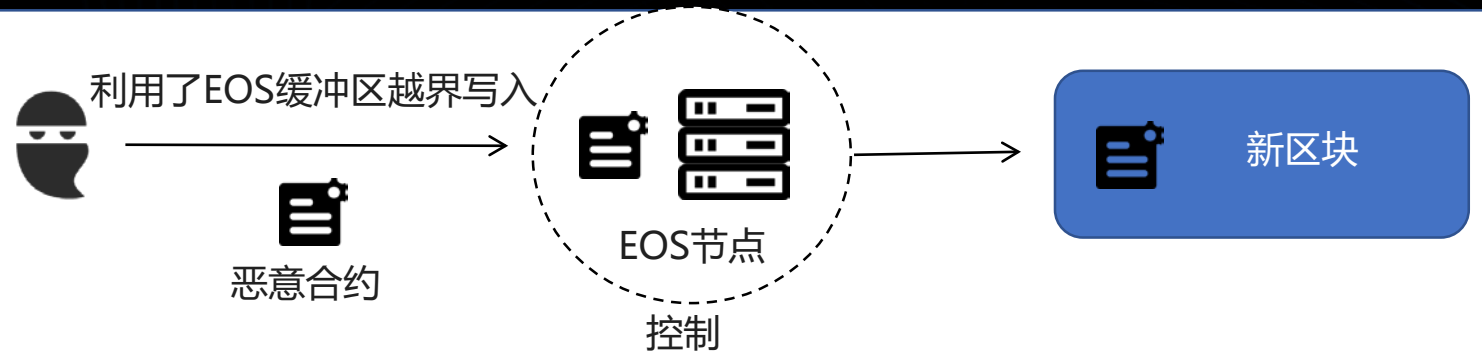
Redis—Lua虚拟机漏洞

00x01 Redis Lua沙盒逃逸漏洞导致可以执行任意字lua字节码，进而实现可执行任意代码。

00x02 Redis Lua子系统缓冲区错误漏洞，可造成拒绝服务或执行任意代码。

00x03 Redis Lua子系统数字错误漏洞，可造成拒绝服务或执行任意代码。

外部引用不安全代码案例——EOS引用WASM代码



2 libraries/chain/webassembly/binaryen.cpp View

```
@@ -73,7 +73,7 @@ std::unique_ptr<wasm_instantiated_module_interface> binaryen_runtime::instantiat
73  73        table.resize(module->table.initial);
74  74        for (auto& segment : module->table.segments) {
75  75           Address offset = ConstantExpressionRunner<TrivialGlobalManager>(globals).visit(segment.offset).value.geti32();
76      -        assert(offset + segment.data.size() <= module->table.initial);
    76  +        FC_ASSERT(offset + segment.data.size() <= module->table.initial);
77  77           for (size_t i = 0; i != segment.data.size(); ++i) {
78  78              table[offset + i] = segment.data[i];
79  79           }
```

1 comment on commit `ea89dce`

guhe120 commented on `ea89dce` 5 hours ago

Hi, there is still some problem with this patch. in 32-bits process, offset + segment.data.size() could overflow and bypass the FC_ASSERT check

来自共识算法的安全威胁

POW

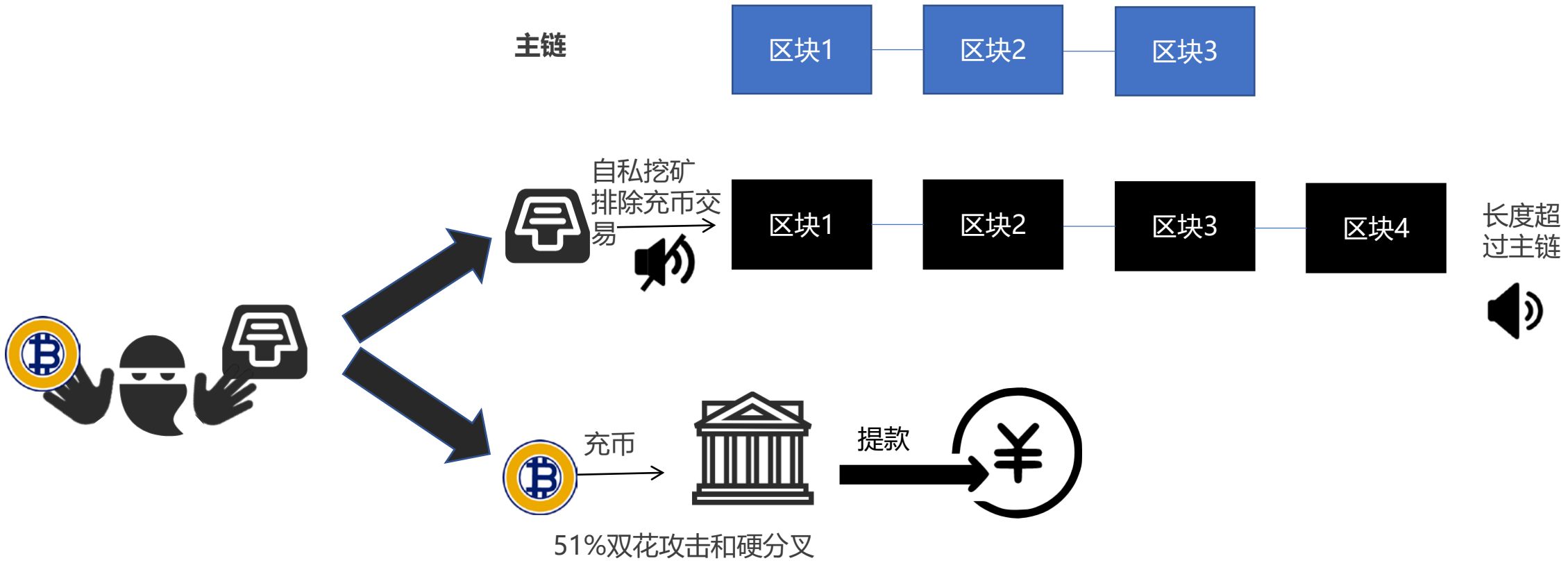
- 巨大能耗
- 算力集中
- 51%算力攻击

DPOS

- 腐败
- 中心化
- 黑客攻击
- 不可信
- 分叉

共识机制威胁案例—BTG双花攻击

背景：BTG区块链在2018年5月16日至19日遭受了连续的17次51%双花攻击，攻击者利用绝对大量的算力回滚交易，对BTG生态造成了严重破坏，从中获利1860万美元。



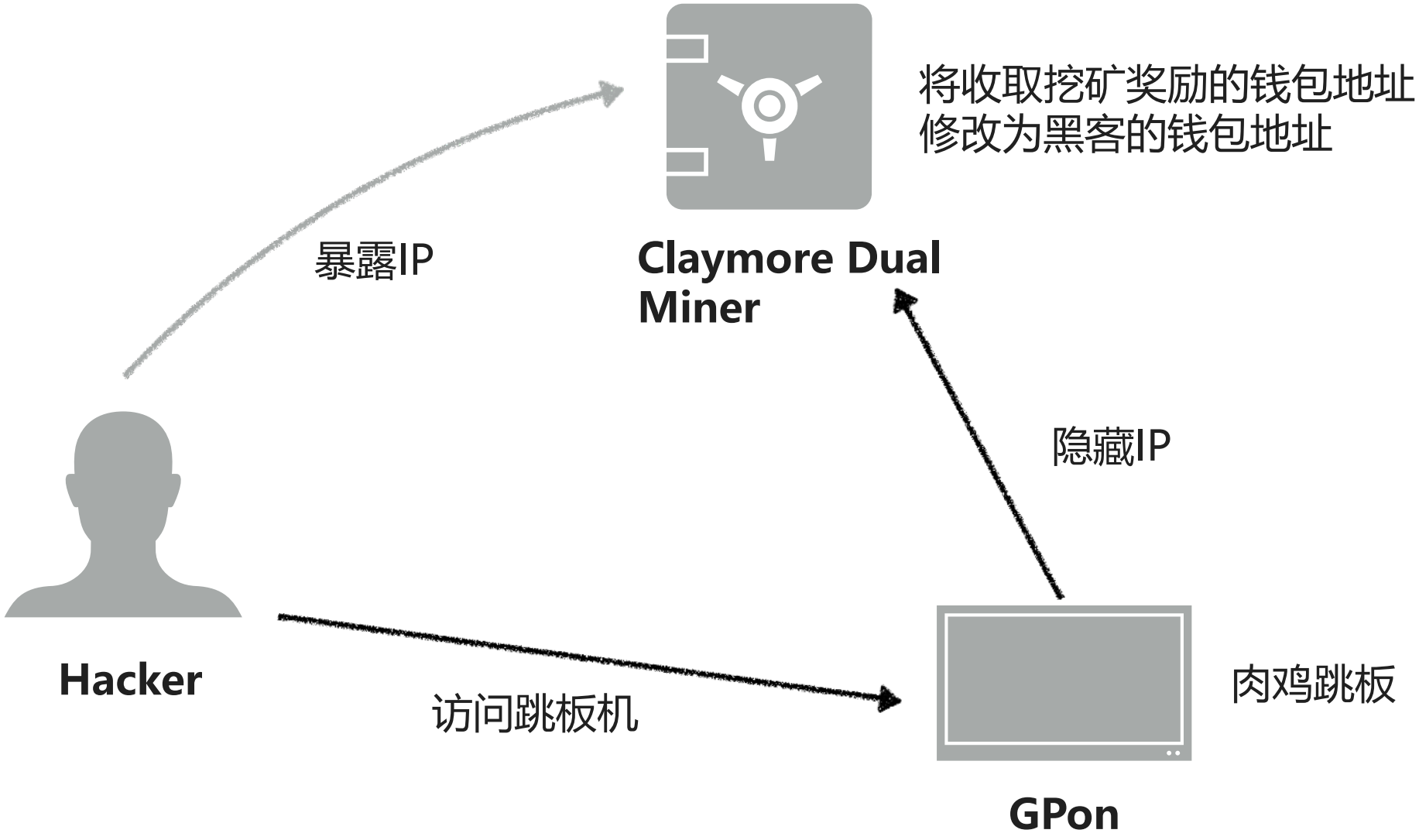
矿场通过多个漏洞的组合案例

GPon

- 认证绕过(CVE-2018-10561)
- 后台远程代码执行(CVE-2018-10562)
- 214W 存活设备

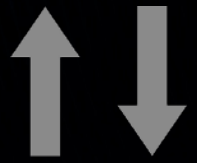
Claymore Dual Miner

- 远程命令执行漏洞(CVE-2018-1000049)
- 1500 台设备


将收取挖矿奖励的钱包地址
修改为黑客的钱包地址
暴露IP
Claymore Dual Miner
隐藏IP
Hacker
访问跳板机
肉鸡跳板
GPon

交易所面临的安全威胁

DDoS



恶意做空

漏洞渗透

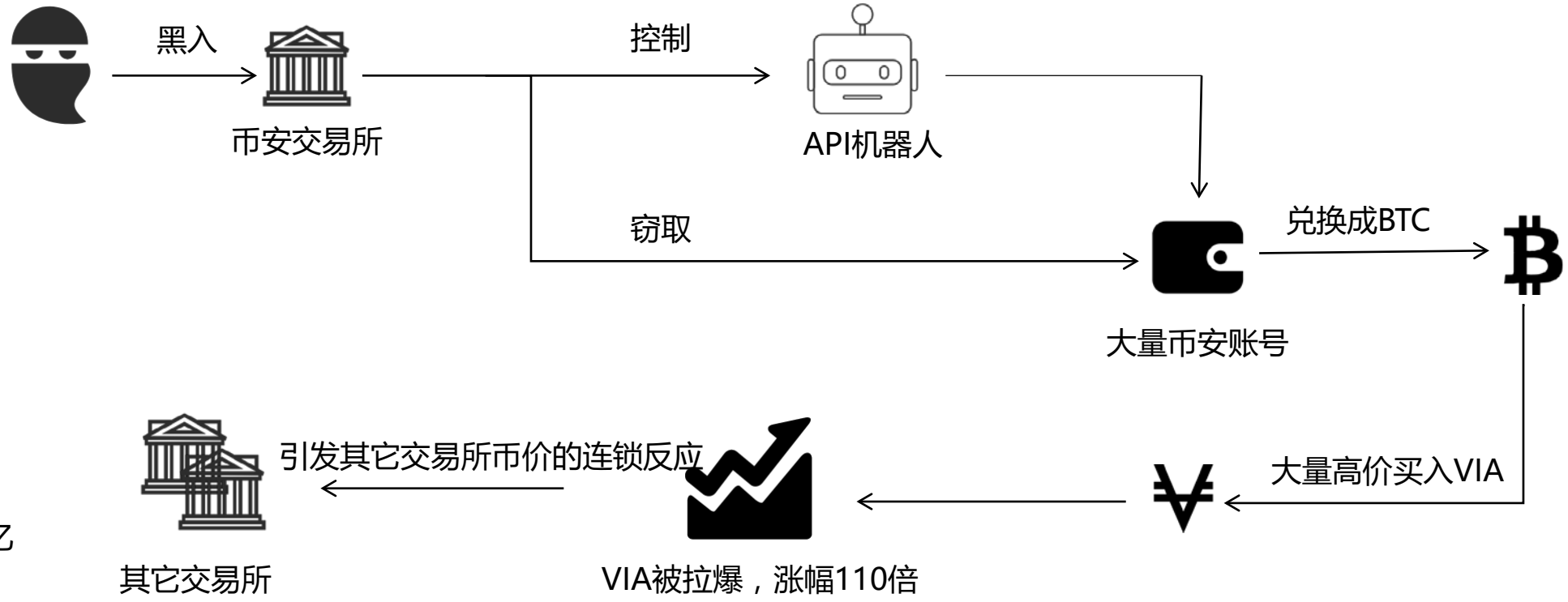
监守自盗

交易所遭遇DDoS闪电战

- 2015年8月，我们监测到一次针对某区块链交易平台的恶意UDP FLOOD攻击行为，受到的攻击流量和数据包峰值瞬间飙升到84517Mbps和30953746pps。攻击者在此次闪电突袭受挫后转为麻雀战术，各种间歇性小规模攻击一直持续了10天。

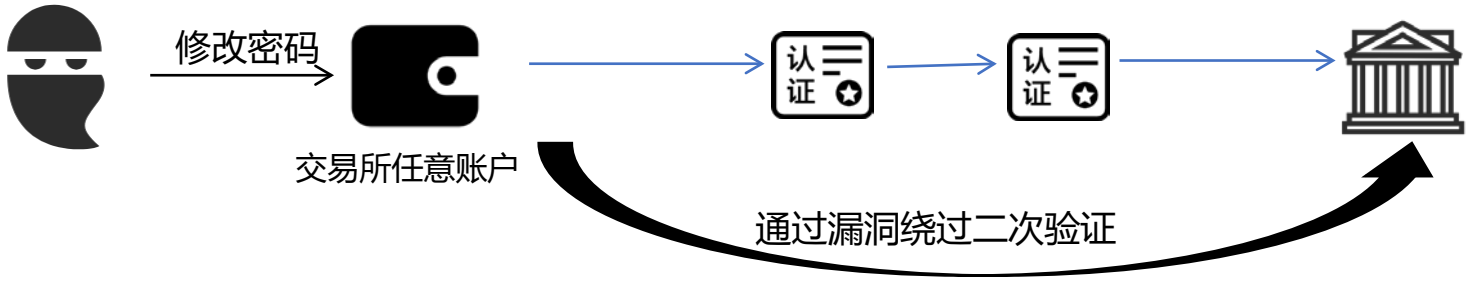
黑客操纵币安交易所账户获利

北京时间3月7日凌晨1：40，数字货币交易所币安（Binance）被爆出现故障，疑似被黑客攻击



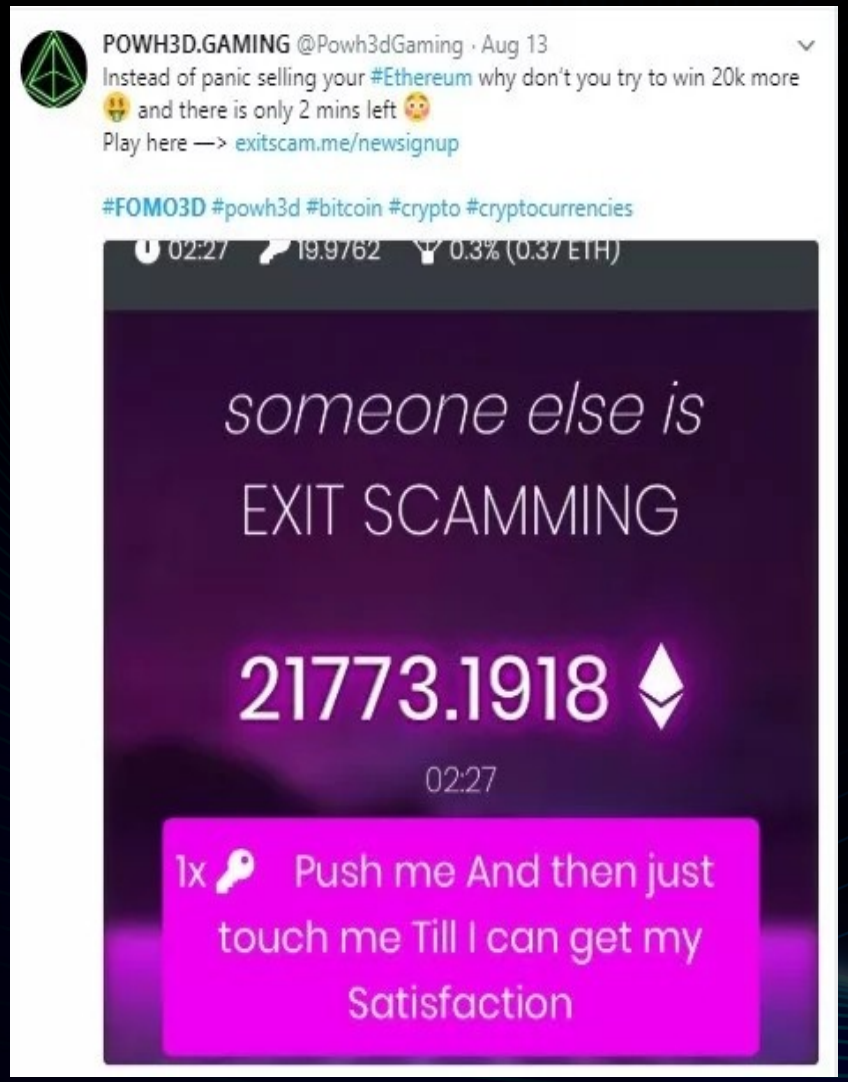
交易所使用有漏洞代码，导致认证被绕过

今年Punisher安全团队捕捉到了一枚交易所的0Day逻辑漏洞。黑客可以通过漏洞修改交易所任意账户的密码，并绕过Google安全验证码和短信验证码二次认证，直接进入交易所中获得权限。这些代码没有经过严格的审查发生漏洞。



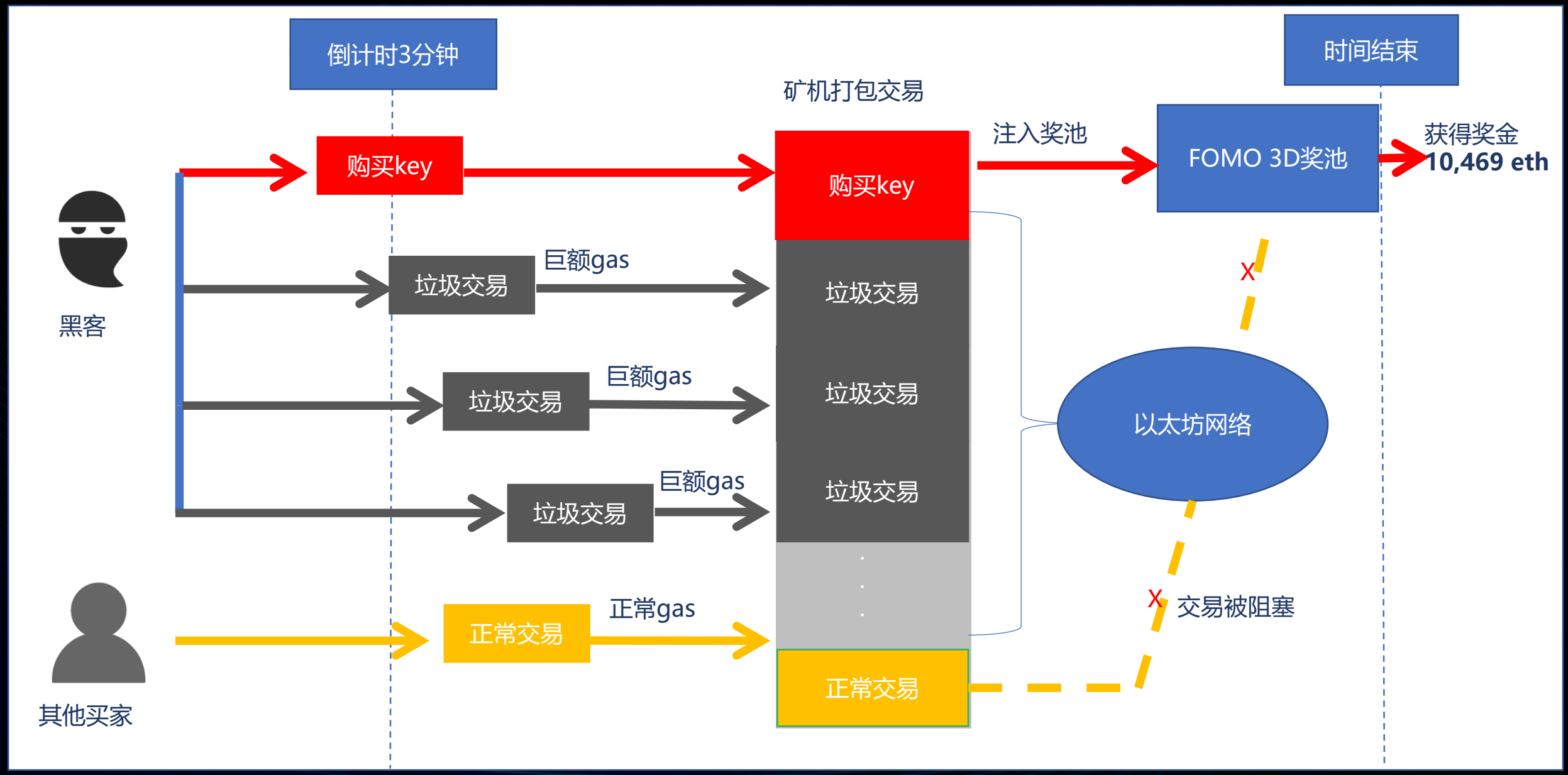
后经检查发现，使用此代码的交易所超过100家，目前存在漏洞的超过20家。Punisher安全团队深入调查，发现在半年前有更多使用此源码的交易所，有一部分在黑客的摧残下倒闭。

垃圾交易攻击案例 - FOMO 3D最后三分钟



2018年8月22日，以太坊上异常火爆的Fomo3D游戏第一轮正式结束，钱包开始为0xa169的用户最终拿走了这笔约**10,469 eth**的奖金，换算成人民币约2200万。

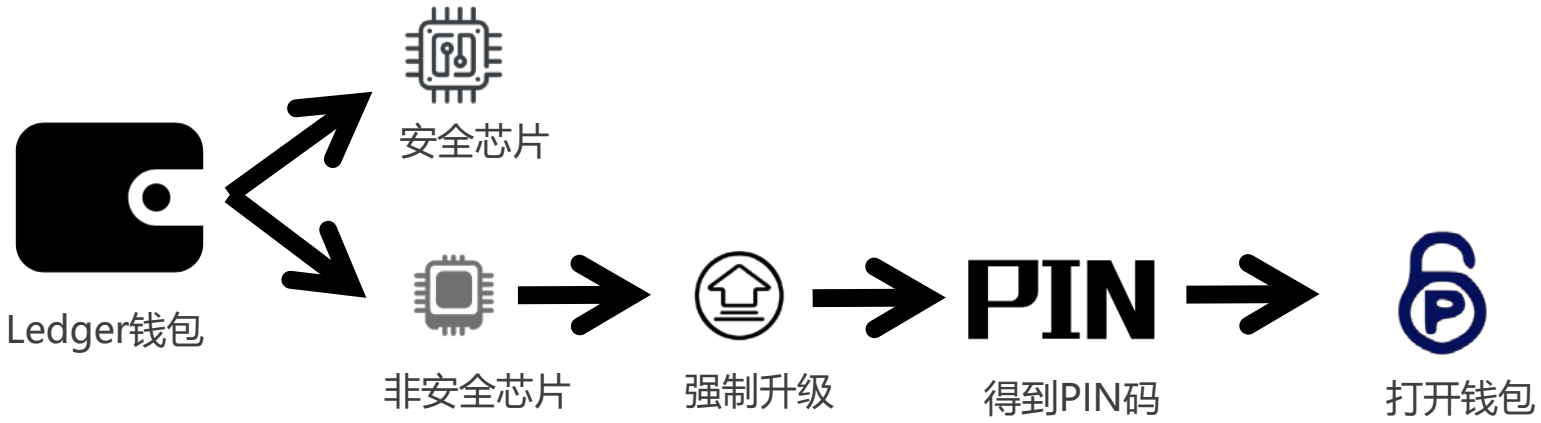
看上去只是一个好运的人买到了那张最大奖的“彩票”，可事实却是，攻击者凭借着对智能合约原理的熟悉，进行了一场精致的“攻击”！


倒计时3分钟
矿机打包交易
时间结束
购买key
注入奖池
FOMO 3D奖池
获得奖金
10,469 eth
黑客
垃圾交易
巨额gas
垃圾交易
巨额gas
垃圾交易
巨额gas
垃圾交易
垃圾交易
垃圾交易
以太坊网络
X
X
交易被阻塞
其他买家
正常交易
正常gas
正常交易

以太坊被阻塞的3分钟内区块Gas信息

中奖前区块所消耗GAS统计					
区块高度	黑客的7801合约	其他	黑客GAS占比	GAS价格	GAS费用（ETH）
6191898	7,000,000	1,000,000	88%	11	0.077
6191899	6,770,000	1,230,000	85%	14	0.095
6191900	7,700,000	300,000	96%	21	0.162
6191901	7,500,000	500,000	94%	21	0.158
6191902	7,000,000	1,000,000	88%	21	0.147
6191903	7,900,000	100,000	99%	190	1.501
6191904	8,000,000		100%	190	1.52
6191905	7,900,000	100,000	99%	501	3.958
6191906	8,000,000		100%	501	4.008
6191907	7,600,000		100%	501	3.808
6191908	7,600,000	379,000	95%	501	3.808
合计			95%		19.24

硬件钱包并不像我们想的那么安全

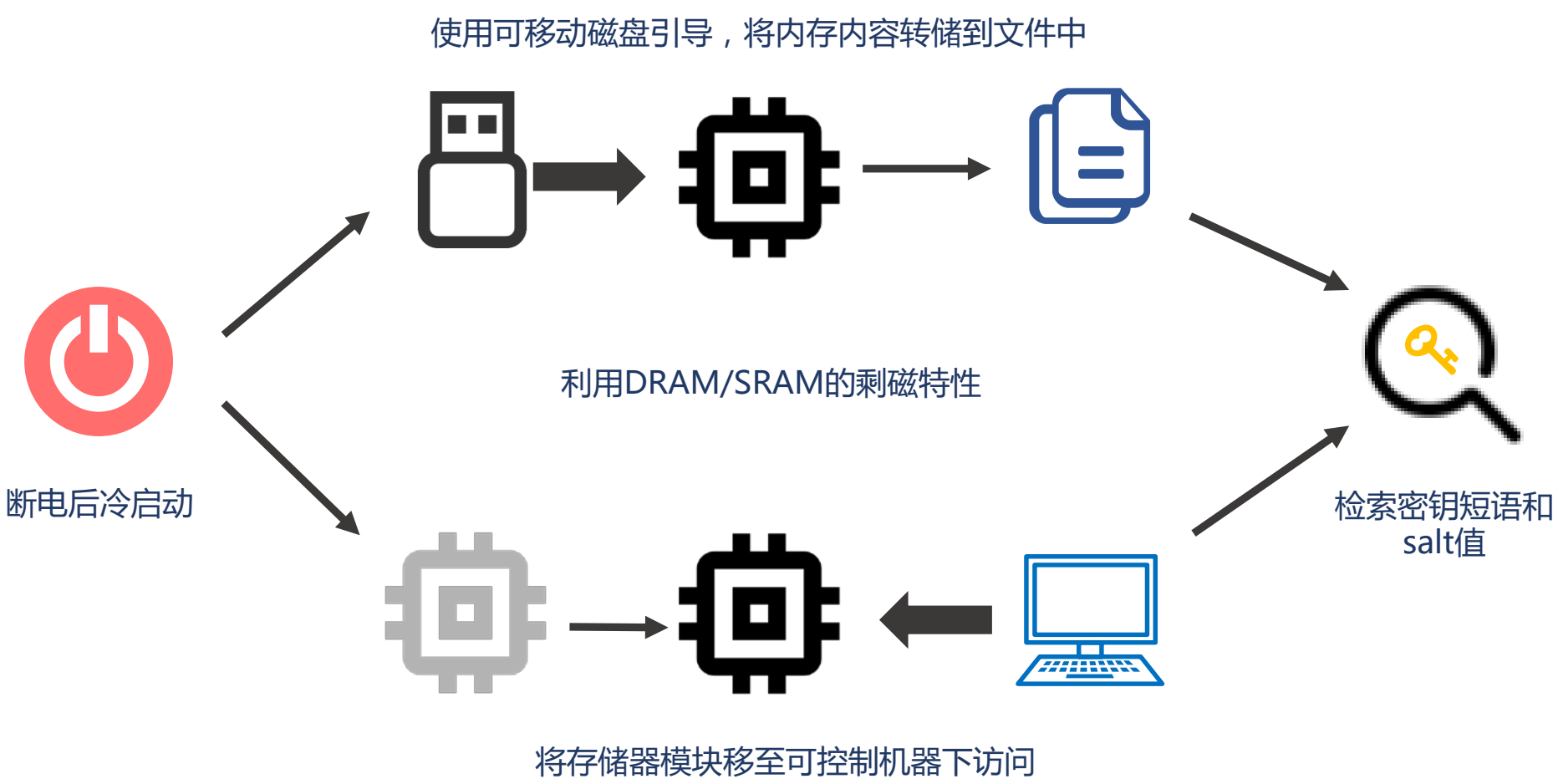


今年年初获得8000万美金融资的法国Ledger钱包。Ledger钱包在设计上有一个安全芯片和一个非安全芯片，我们通过强制升级非安全芯片的方式，在不拆除外壳的前提下，就能一步步取得钱包的PIN码。PIN码相当于钱包的密码，有了它，就可以打开钱包，把钱转走。

“不可破解的”Bitfi钱包被两次破解

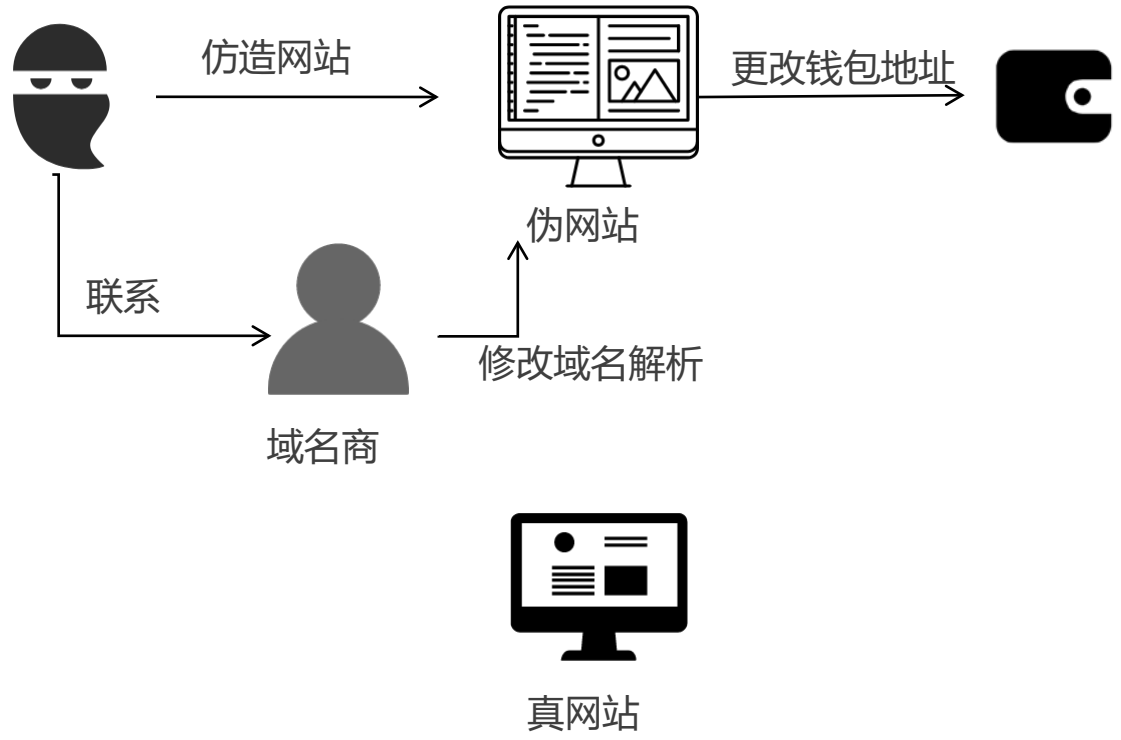
- 7月24日，Bitfi的执行主席John McAfee称Bitfi钱包为“世界上第一个不可破解（Un-hackable）的设备”
- 8月12日，一组研究人员声称入侵了“不可破解的”Bitfi钱包
- 9月2日，Bitfi钱包再次被两个安全研究人员入侵，两人都声称他们使用冷启动攻击入侵钱包
- Bitfi硬件钱包使用密钥短语和salt值计算私钥，未擦除干净RAM使攻击成为可能

Bitfi钱包遭遇冷启动攻击

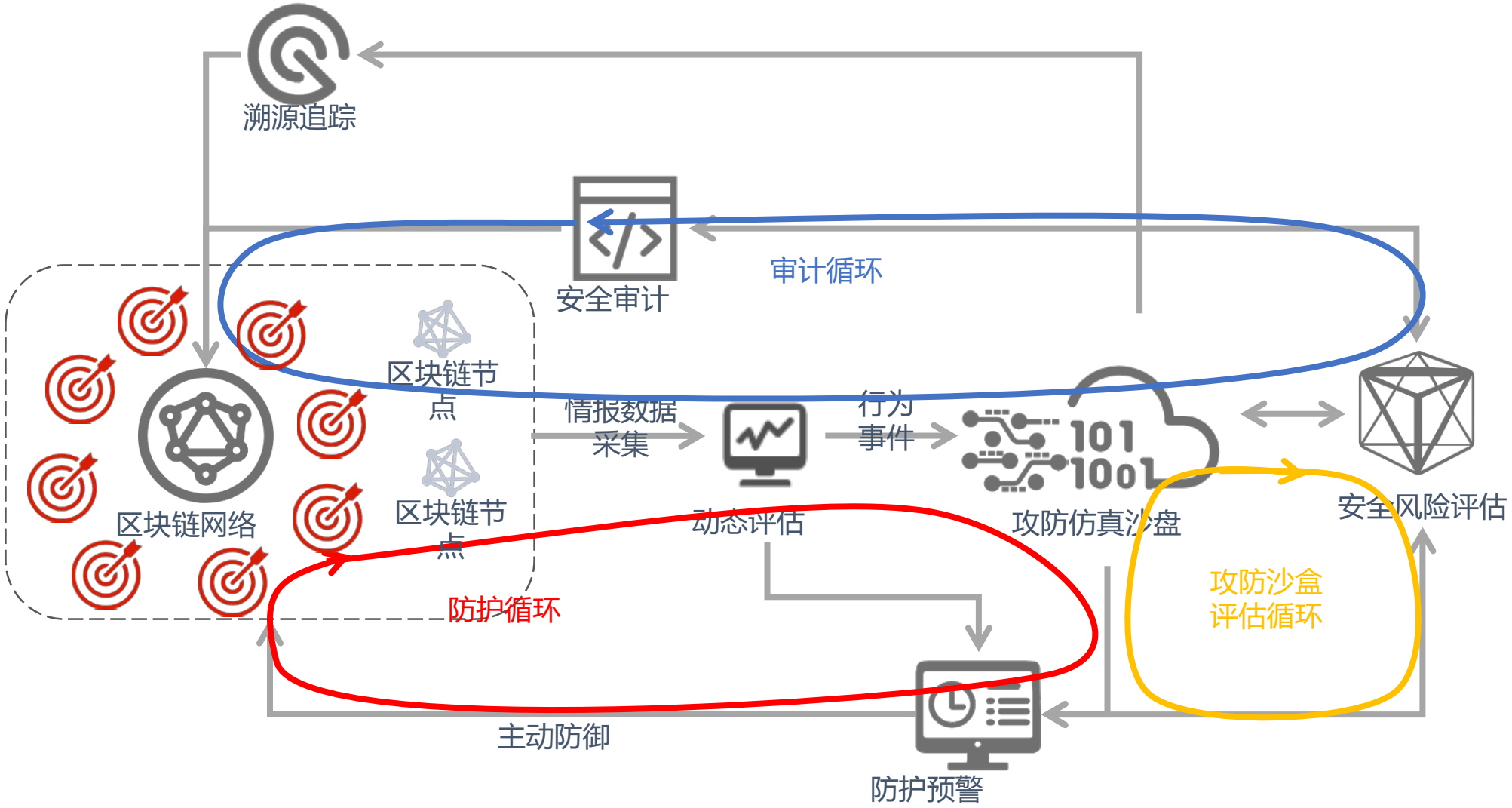


CoinDash事件:钓鱼网站+虚假的钱包地址欺诈

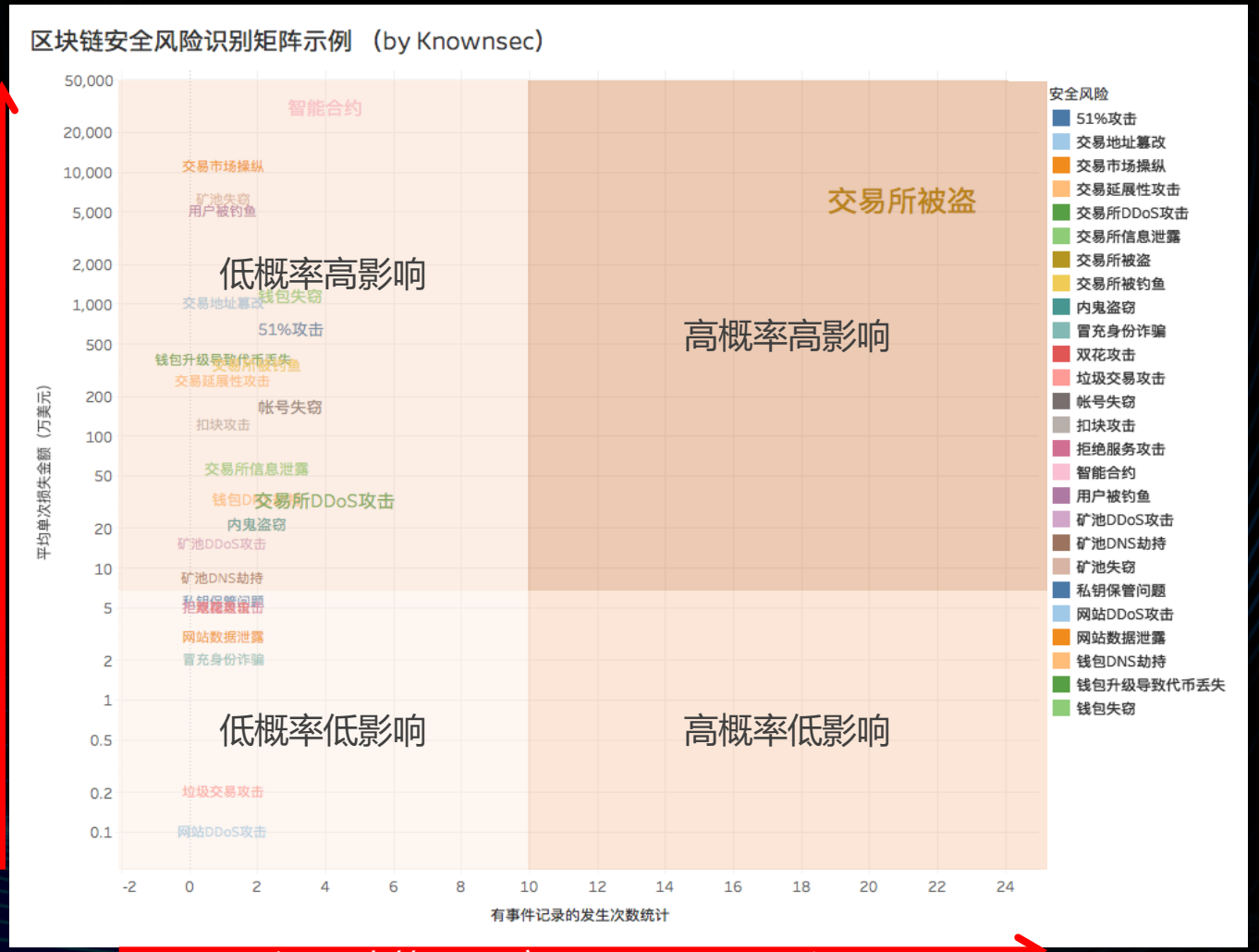
2017年7月17日，区块链初创公司CoinDash在发布通证众筹过程中突遭黑客袭击，钱包地址被篡改，导致尝试用以太币购买该公司通证的投资者们损失惨重，4万个以太坊被打入黑客地址不知所踪。



面向区块链的安全防护体系



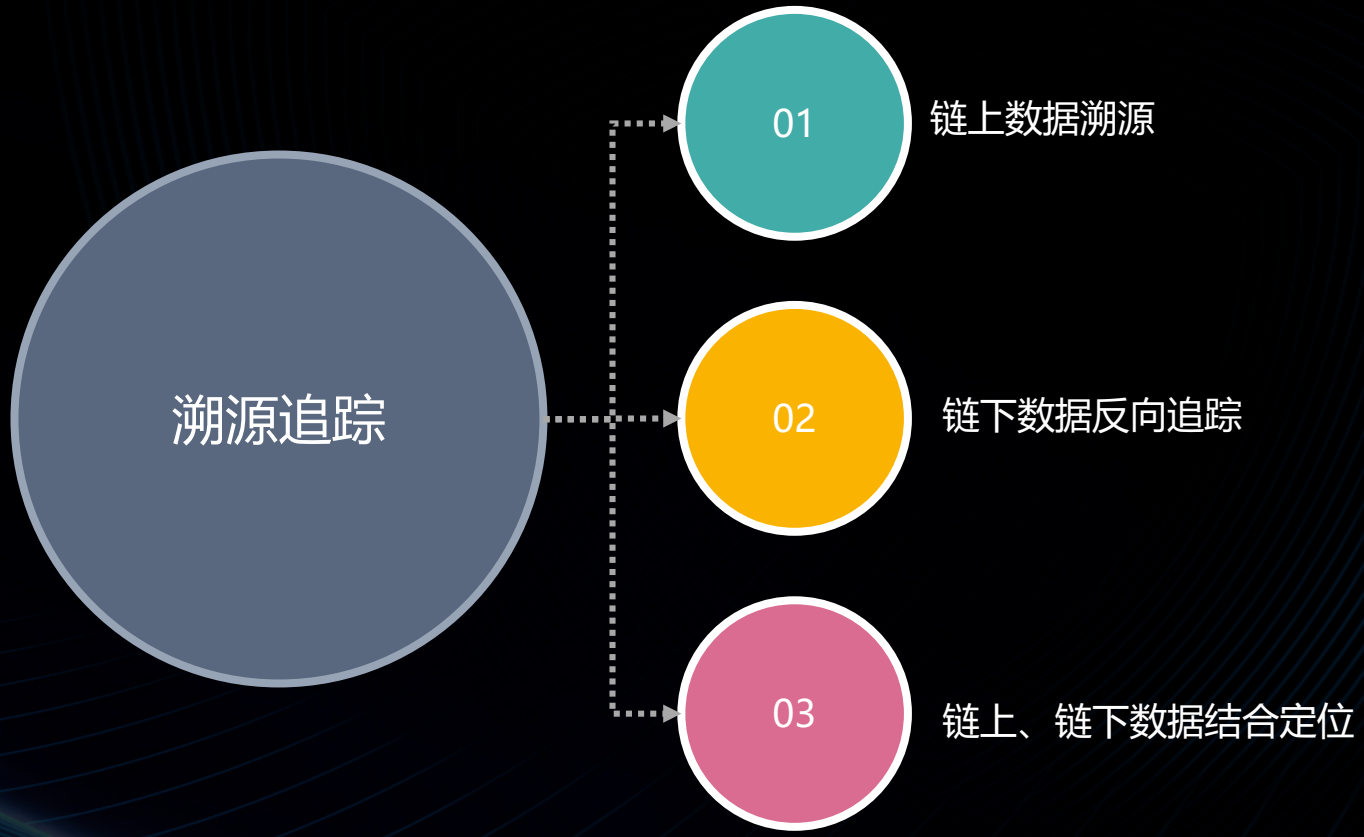
区块链安全风险识别预评估



区块链安全风险模拟大数据沙盘

通过从区块链节点的抓取数据信息建立沙盘并构建大数据分析模型，可以在沙盘上推演黑客的攻击手段和区块链可能承受的风险损失，并根据沙盘推演结果评估损失程度及发生概率。

区块链情报的溯源追踪



五个佛系认知

现实社会 vs 数字孪生

区块链是现实社会的“数字孪生”

- 数字孪生的概念源于现实世界中物体在数字世界的投影。
- 区块链系统所呈现的运行机制和模式就像是现实社会在数字世界数学化、算法化的孪生体。

传统经济 vs 通证经济

区块链更适合通证经济模式

- 相比集中化程度更高的传统经济而言，区块链系统更适合社群化特征明显的经济模式发展
- 通证经济的数字权益自由流通和社群众筹方式天生适合在区块链上运转

人治社会 vs 算法社会

区块链以数学算法和共识来治理数字社会

- 区块链以数学算法进行治理，所有的共识算法、一致性机制都是为了保证没有中间的腐败或差错。
- 区块链的去中介化，并非没有中介，而是无人化中介
- 整个区块链系统承担中介的角色，由算法和共识所形成的中介保证系统运行，过程中没有人参与。

实体化 vs 数字化

数字化程度越高的行业越适用区块链

- 实体产业数字化尚存在一定难度，过早的将区块链应用在实体产业上还为时尚早
- 与之相反金融、票证、数字资产权益、虚拟物品等已经被数字化的产业，能够更快速的应用区块链

刚性安全 vs 柔性安全

区块链系统的安全需要用柔性安全的视角来看待

- 区块链系统是现实治理方式在虚拟世界的影射，同时又涉及数字算法、经济模式、治理机制多个方面的知识，仅仅靠传统网络安全和信息安全的刚性安全是不足以完全覆盖区块链的安全保障范围。所以区块链系统的安全需要以一种柔性安全的视角去对待。

知己知彼，攻守有道

感谢大家的聆听

Thank you very much & best regards.

区块链技术的未来演进

——无所不用，无所不能！