智能合约安全研究进展

何立宝

深圳市优权天成科技有限公司

2017 年 11 月 4 日

内容提要

智能合约概念及应用

- 基本概念
- 商业场景
- 实例分析

基本概念

- 智能合约是 1990s 年代 Nick Szabo 提出的概念

  a set of promises, specified in digital form, including protocols within which the parties perform on these promises.
- 数字化交易协议、自动适时执行、无法人工停止
- 智能合约 $\neq$ 法律合同
- Smart $\neq$ 聪明，Smart $=$ 灵活

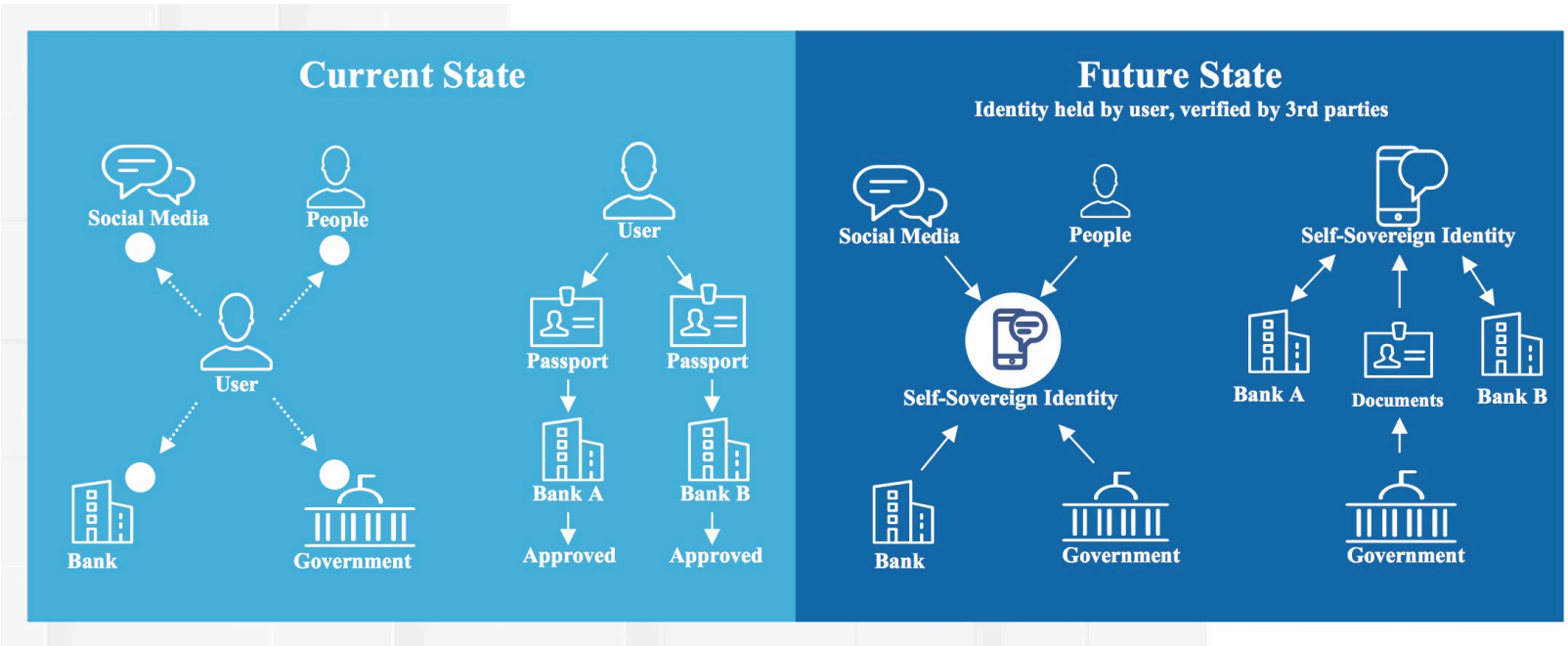
应用场景

- 智能合约重塑商业流程

1. 数字身份
2. 记录
3. 证券
4. 贸易金融
5. 衍生品
6. 金融数据记录
7. 抵押贷款
8. 土地所有权记录
9. 供应链
10. 汽车保险
11. 临床试验
12. 癌症研究

实例分析

- 数字身份：中心化多账户 => 自主控制单账户



实例分析

- 中心化多账户的挑战
  1. KYC 代价高且信息不完备
  2. 潜在信息泄露可能性高
  3. 企业负责保管用户数据，需要防范单点失败和黑客攻击风险
- 自主控制单账户的优势
  1. 个人拥有数据所有权，可选择性披露信息给交易方
  2. 企业不再保管用户数据，成本降低
  3. 提高了合规性、弹性、互操作性
- 智能合约实施要点
  1. 法律框架内接受数字证明，并建立对智能合约安全的信任
  2. 与认证提供商的技术集成
  3. 形成协议和标准，以提供相关方的互操作性

智能合约运行机制

- 智能合约与区块链关系
- 区块链技术基础
- 以太坊智能合约

智能合约与区块链关系

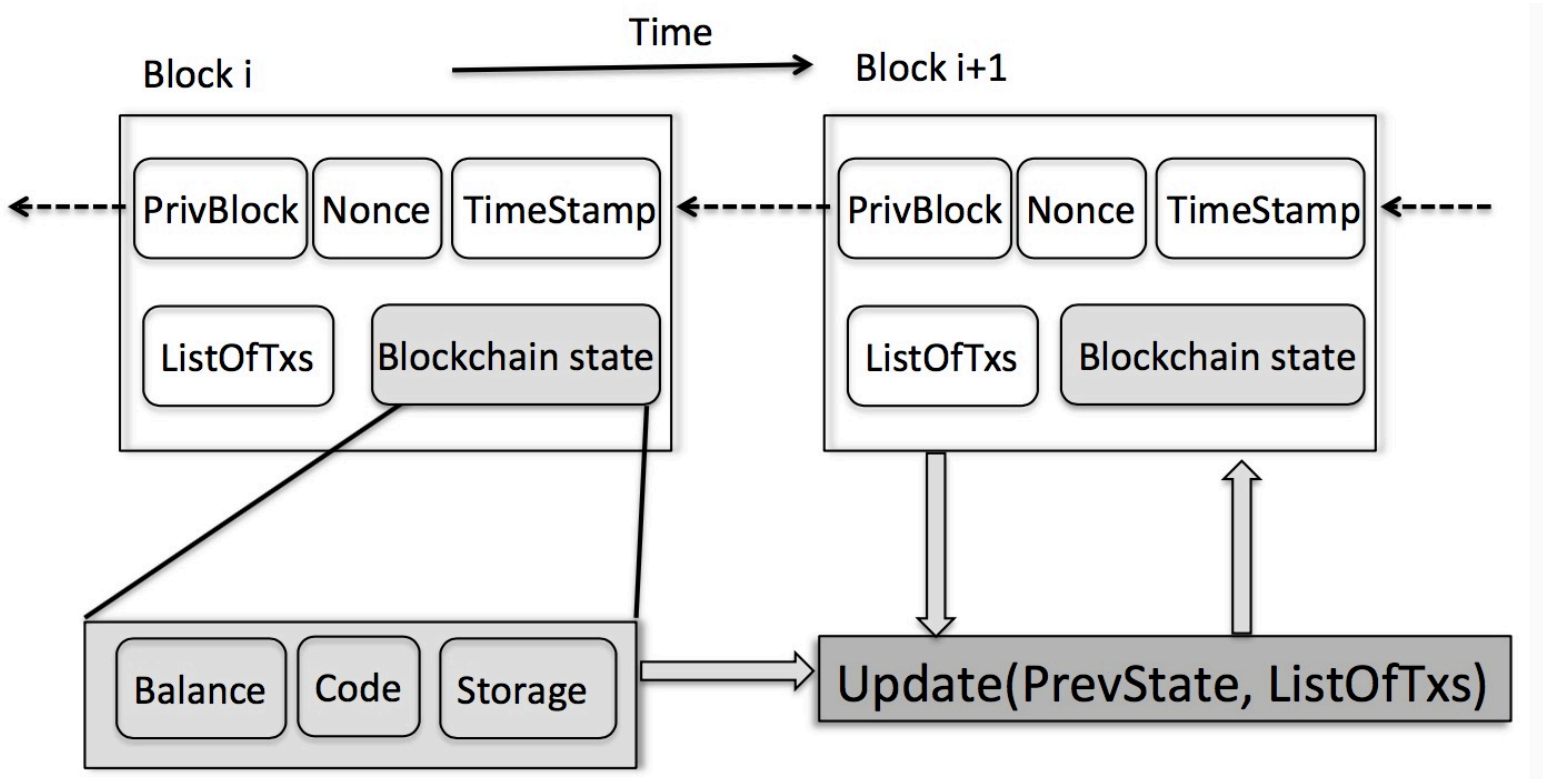
- 智能合约未被大规模应用，根因是可信执行环境长期缺失
- 智能合约运行环境基本需求
  1. 灵活的合约编程语言
  2. 合约部署后不可篡改
  3. 合约执行具有确定性并且不可撤销
  4. 合约状态公开可验证
- 区块链是运行智能合约的绝佳载体

基本概念

- 狭义来讲，区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构，并以密码学方式保证的不可篡改和不可伪造的分布式账本
- 涉及技术领域
  1. 密码学
  2. 共识机制
  3. P2P 技术
  4. 分布式数据存储

基本概念

- 共识达成的区块才能链接到分布式账本
- 交易促成区块链状态变化



比特币

- 比特币核心概念
  - POW 挖矿：计算满足 $proofhash < target$ 的 nonce
  - 区块奖励：新挖区块链接到公共账本获得奖励
  - 区块链交易：比特币从发送方转移到接收方
- 比特币与法币
  - 去中心化公共账本 $\Leftrightarrow$ 中心化私有账本
  - POW 挖矿获得区块奖励 $\Leftrightarrow$ 铸币新增货币发行，总量增加
  - 区块链交易 $\Leftrightarrow$ 货币流通，总量恒定

以太坊目标和定位

- 目标：解决比特币的四个局限性
  - 缺少图灵完备
  - Value-blindness：UTXO 脚本不能提供账户余额的精确控制
  - 缺少状态：UTXO 只有已花费或者未花费状态
  - Blockchain-blindness：UTXO 看不到区块链数据
- 定位：下一代智能合约和去中心化平台

以太坊账户模型

- 账户分类
  - 外部账户 => 对应业务参与方，通过私钥控制
  - 合约账户 => 对应业务逻辑，只能通过外部账户控制
- 账户状态
  - nonce: 账户发起的消息调用总数，防重放攻击
  - balance: 账户余额
  - storageRoot: 合约变量状态
  - codeHash: 合约账户关联 EVM 代码
- 区块链状态：由外部账户和合约账户的状态集合构成

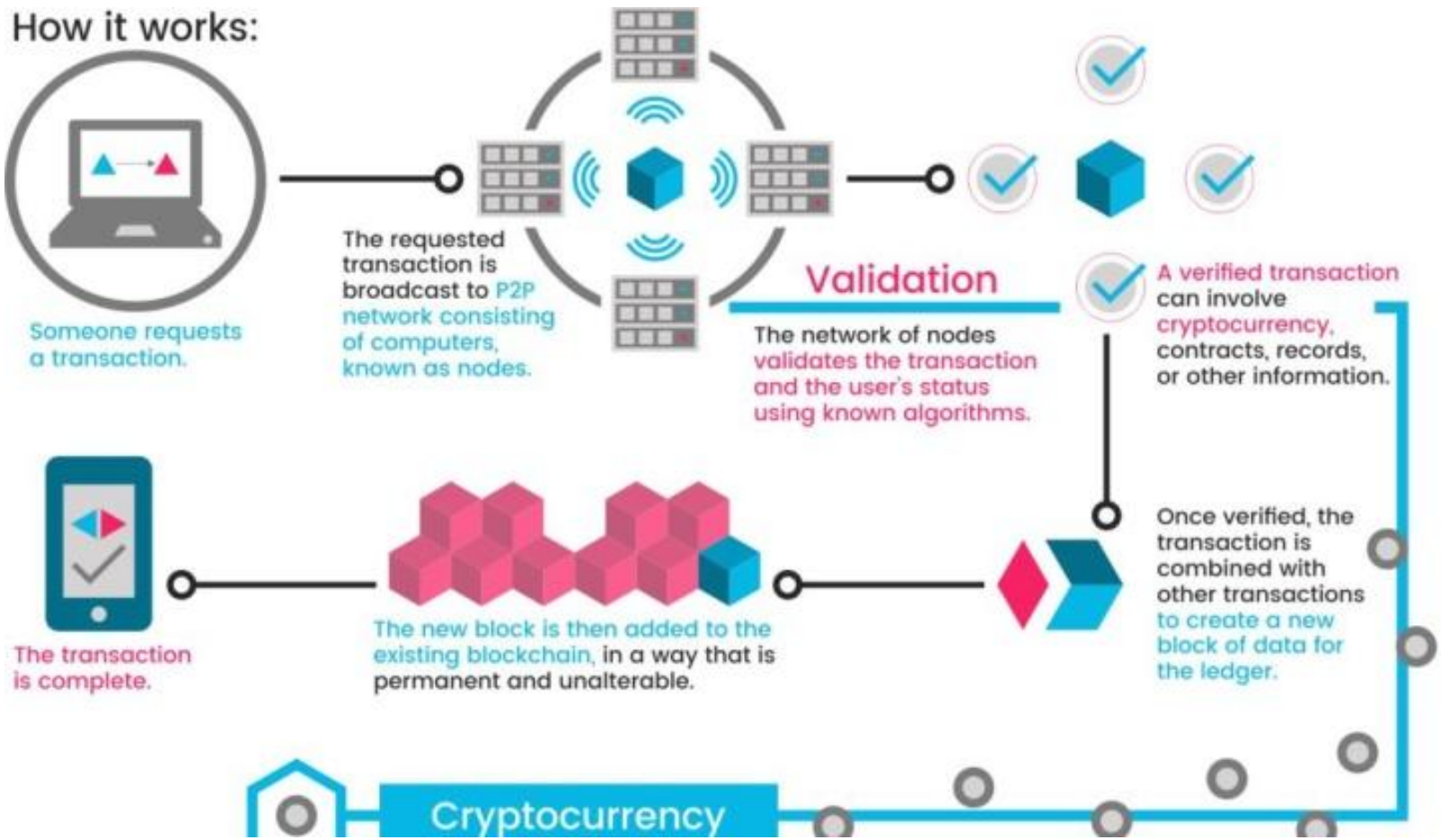
以太坊交易模型

- 符号约定：
  (1) T_n: nonce
  (2) T_p: gasPrice
  (3) T_g: gasLimit
  (4) T_t: 交易接收方
  (5) T_v: 发送金额
  (6) T_w, T_r, T_s：交易签名相关参数 v, r, s
  (7) init: 合约账户初始化时附带的 EVM 代码
  (8) data: 消息调用输入参数，ABI 编码
- 交易类型：
  (1) 合约创建：$L_T(T) = (T_n, T_p, T_g, T_t, T_v, T_\mathrm{i}, T_w, T_r, T_s)$, $T_t = \varnothing$
  (2) 消息调用：$L_T(T) = (T_n, T_p, T_g, T_t, T_v, T_\mathrm{d}, T_w, T_r, T_s)$, $T_t \neq \varnothing$

智能合约部署和调用

- 智能合约部署
  - 用 Solidity 编码合约
  - 编译合约代码为 EVM bytecode
  - 外部账户发起交易部署合约代码，生成合约账户地址
- 智能合约调用
  - 外部账户发起交易调用合约函数

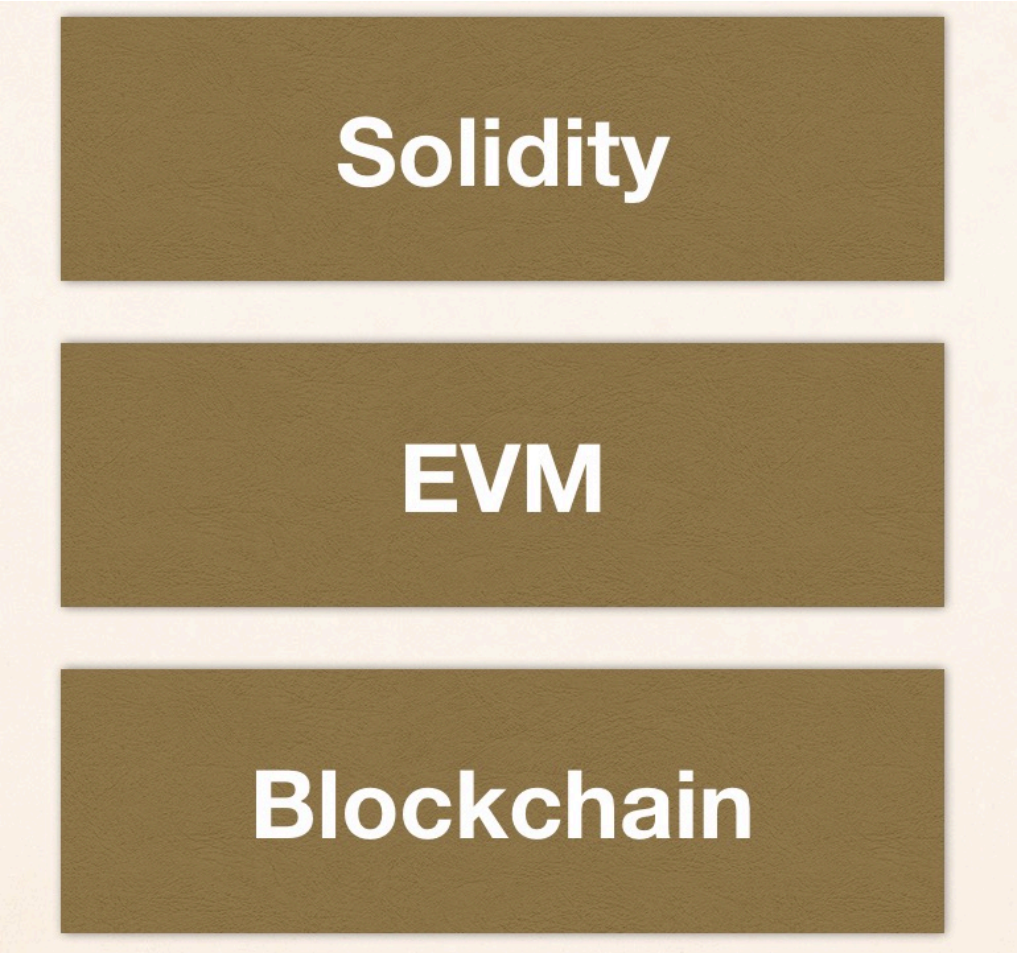
智能合约工作流程


How it works:
Someone requests a transaction.
The requested transaction is broadcast to P2P network consisting of computers, known as nodes.
Validation
The network of nodes validates the transaction and the user's status using known algorithms.
A verified transaction can involve cryptocurrency, contracts, records, or other information.
Once verified, the transaction is combined with other transactions to create a new block of data for the ledger.
The new block is then added to the existing blockchain, in a way that is permanent and unalterable.
The transaction is complete.
Cryptocurrency

智能合约安全编程

- 安全漏洞概览
- 安全事故剖析
- 最佳实践及安全工具

漏洞原因分层



SOLIDITY RELATED VULNERABILITY

1. 调用外部合约函数
   - call, send, delegatecall 等原语可能有调用 fallback 的副作用
   - 调用合约未知函数可能导致重入
2. 异常错乱
   - 合约函数代码通过 call/send 调用另一合约函数
   - 调用未检查返回值，可能导致事务性缺失
3. 交易 gas 限制
   - send 等同于 gas 上限为 2300 的 call 调用
   - 不支持支付到 fallback 函数昂贵的合约地址
4. 类型转换
   - 给定地址，通过类型检查无法确定其是否为特定类型合约
   - 转换类型不一致，调用错误合约的等价函数或 fallback

SOLIDITY RELATED VULNERABILITY

5. 数据安全
   - 修饰符 private 仅限制账户读取数据，不保证不可见性
   - 隐私性需要在应用层引入额外密码技术，如承诺机制等
6. 溢出
7. 拼写错误
   - 构造函数名拼写错误导致合约所有权丢失
8. 修饰符错误
   - 合约私有函数未指定访问修饰符默认为 public

EVM RELATED VULNERABILITY

1. 不可变更
   - 带 bug 的合约代码部署后无法修改
   - 需要在设计时考虑修复和升级机制
2. 转移 Ether 丢失
   - Ether 转移到孤儿地址，彻底丢失
3. 调用栈大小限制
   - 调用栈大小上限为 1024，溢出则调用失败
   - 可人为控制调用栈高度来控制 call 调用成功或失败

BLOCKCHAIN RELATED VULNERABILITY

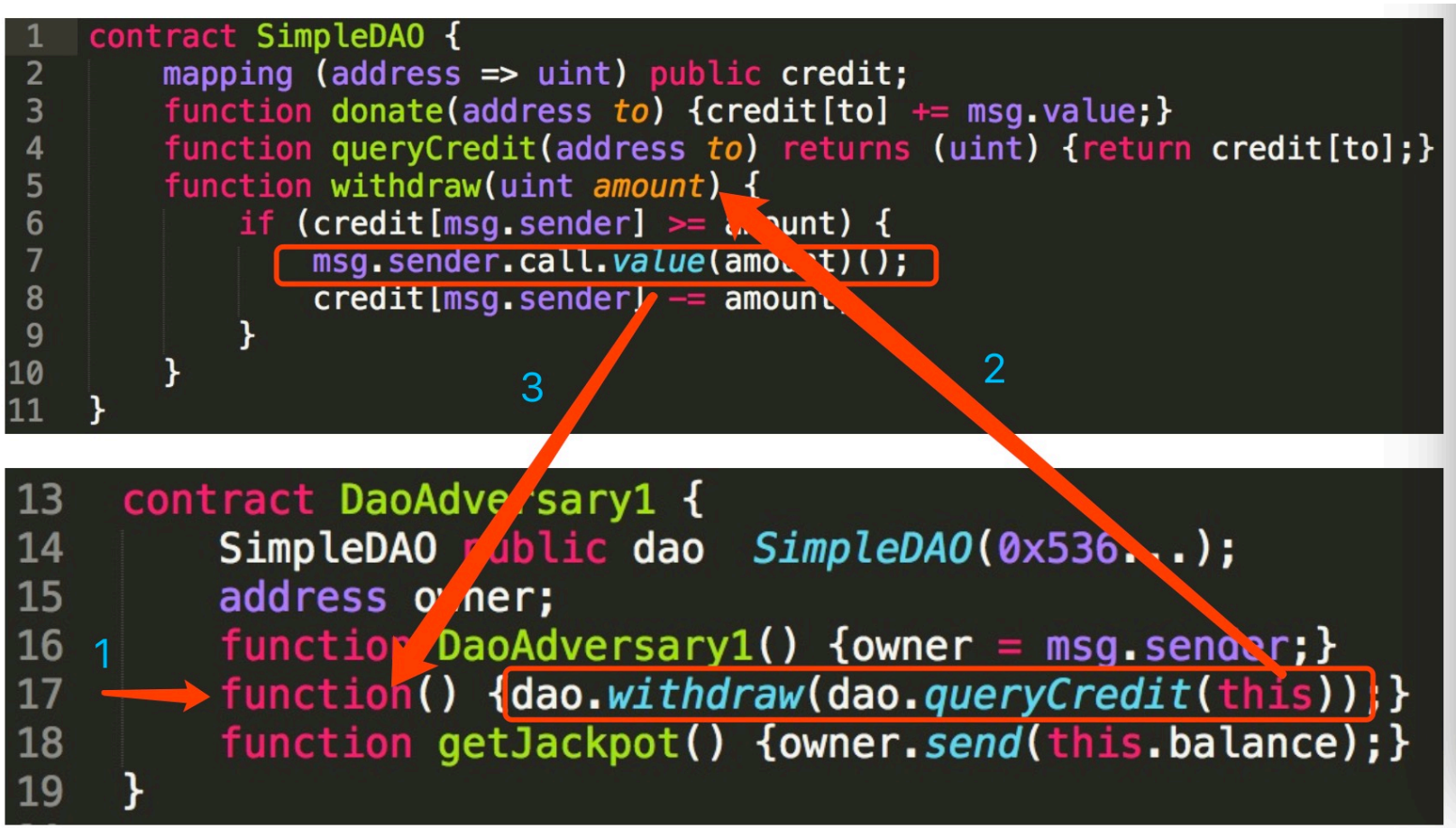
1. 状态不可预知
   - 交易执行时合约状态可能与交易发起时合约状态不一致
   - 矿工可操控交易打包顺序
2. 随机数依赖：矿工预先得知
3. 时间戳依赖：矿工预先得知

The DAO attack

- 分布式自治组织
- 众筹后根据代币数分配投票权，集体决定资金用途
- 智能合约漏洞导致筹集的 $60M（总共 $150M）被黑客转移
- 解决方案：以太坊硬分叉 ETH & ETC
- 项目报废

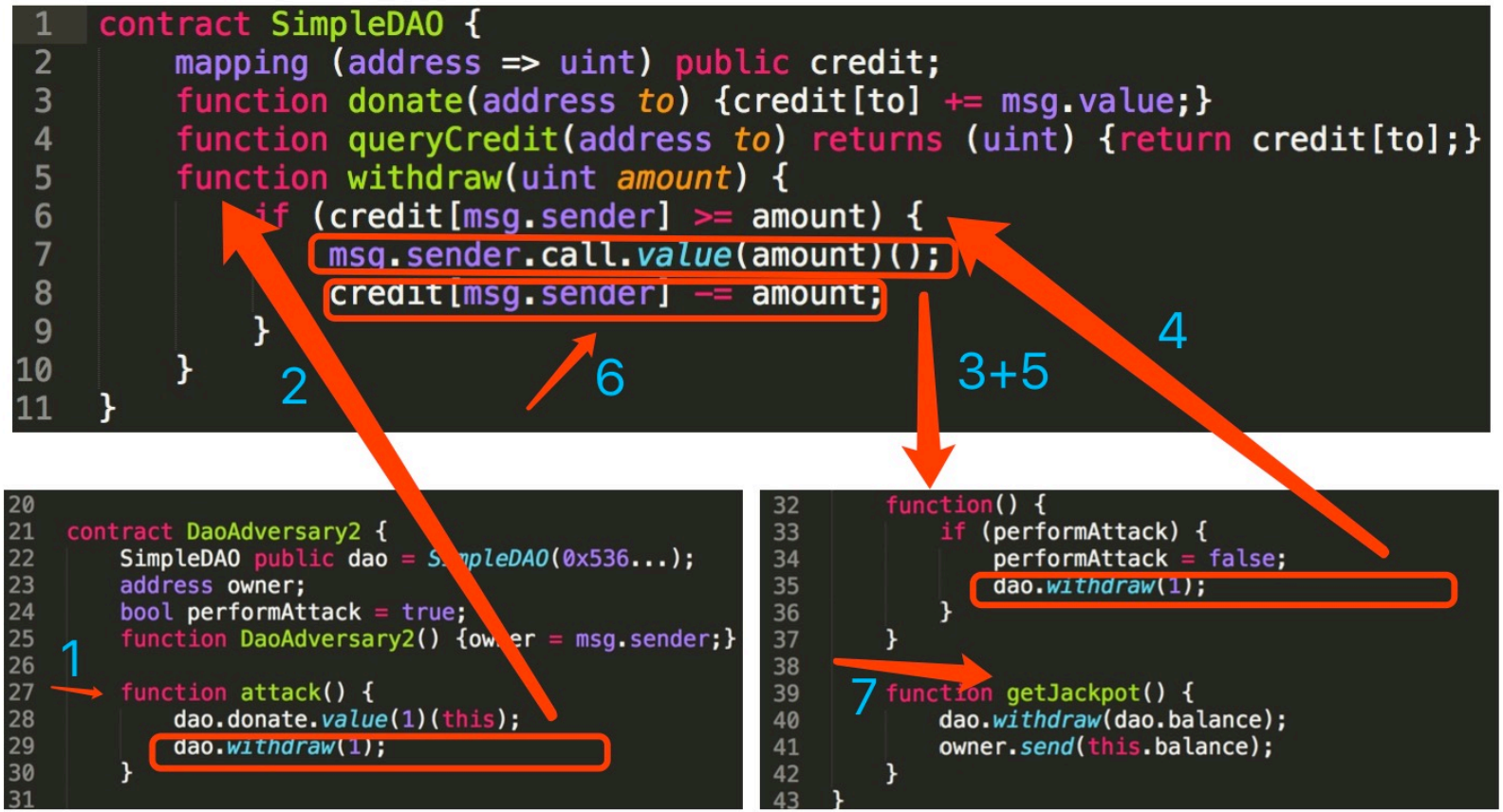
THE DAO ATTACK

- 攻击 #1：外部合约函数调用导致重入



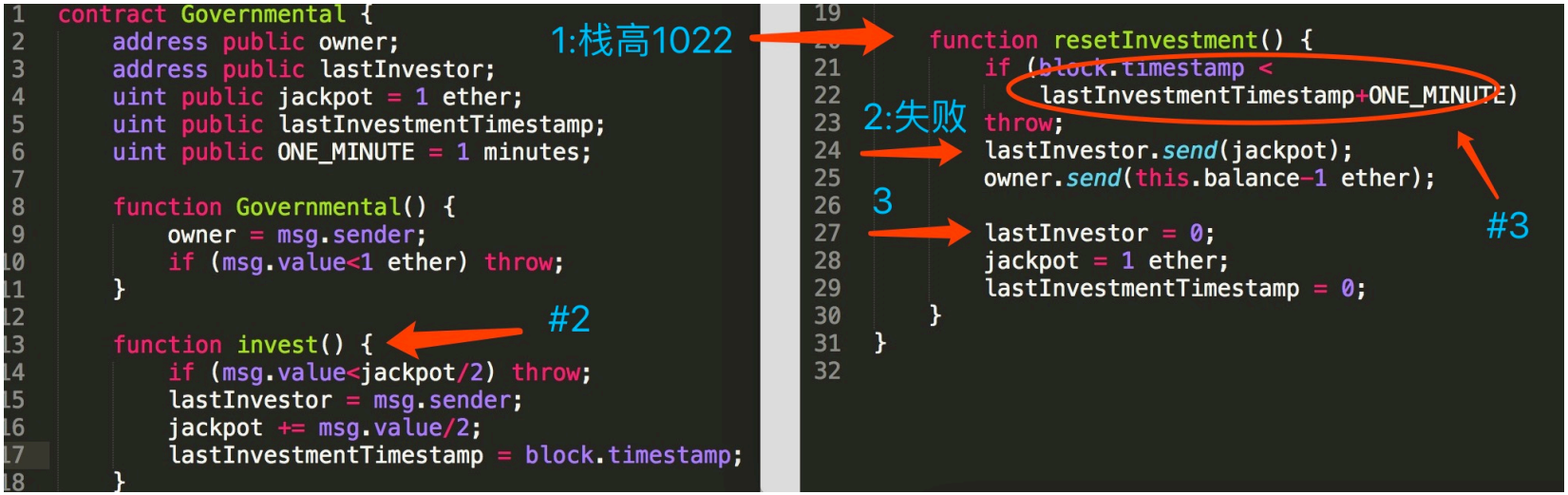
THE DAO ATTACK

- 攻击 #2：外部合约函数调用导致重入 + 下溢出



GovernMental Ponzi Attack

- 攻击 #1：调用栈大小限制
- 攻击 #2：不可预知状态
- 攻击 #3：时间戳依赖




```
contract Governmental {
    address public owner;
    address public lastInvestor;
    uint public jackpot = 1 ether;
    uint public lastInvestmentTimestamp;
    uint public ONE_MINUTE = 1 minutes;

    function Governmental() {
        owner = msg.sender;
        if (msg.value<1 ether) throw;
    }

    function invest() {
        if (msg.value<jackpot/2) throw;
        lastInvestor = msg.sender;
        jackpot += msg.value/2;
        lastInvestmentTimestamp = block.timestamp;
    }

    function resetInvestment() {
        if (block.timestamp <
            lastInvestmentTimestamp+ONE_MINUTE)
        throw;
        lastInvestor.send(jackpot);
        owner.send(this.balance-1 ether);

        lastInvestor = 0;
        jackpot = 1 ether;
        lastInvestmentTimestamp = 0;
    }
}
```

PARITY WALLET ATTACK

- 攻击：修饰符错误导致私有函数可公共访问

```
1  // constructor - just pass on the owner array to the multiowned and
2  // the limit to daylimit
3  function initWallet(address[] _owners, uint _required, uint _daylimit) {
4      initDaylimit(_daylimit);
5      initMultiowned(_owners, _required);
6  }
7
8  function() payable {
9      if (msg.value > 0)
10         Deposit(msg.sender, msg.value);
11         else if (msg.data.length > 0)
12             _walletLibrary.delegatecall(msg.data);
13 }
```

默认为public

安全编码策略

1. 失败准备
   - 出错时暂停合约，如提供断路器
   - 管理风险金额，如限速、限额
   - 为修复和改进提供有效的升级路径
2. 谨慎上线：彻底测试合约，发现新攻击增加测试
3. 保持简单
   - 合约模块化、逻辑简单
   - 不重复发明轮子
4. 保持最新：代码、文档、工具、bug 修复等
5. 注意区块链属性
   - 小心调用外部合约，可能执行恶意代码改变控制流程
   - 保持 gas 开销和 gas 限制
   - 区块链数据公开可获得
   - 矿工是潜在攻击者

安全相关工具

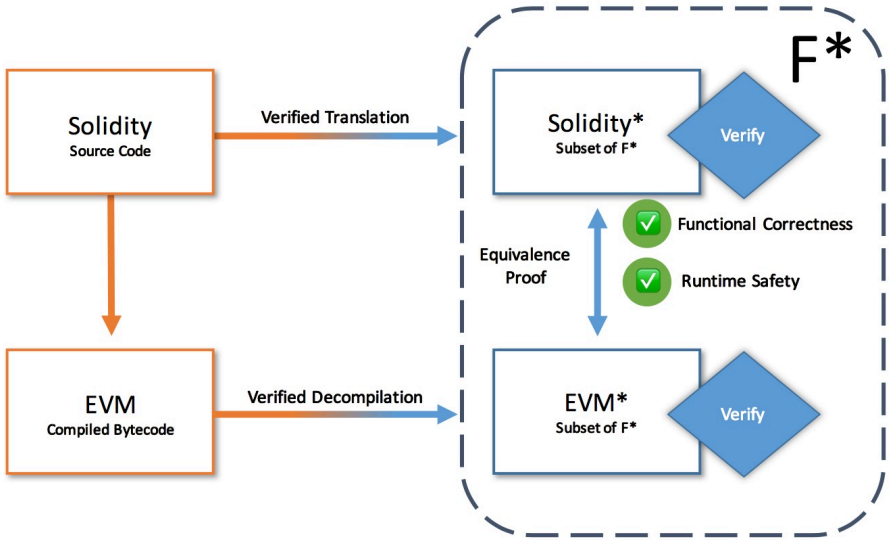
- Zeppelin: 安全代码组件，可以配合 truffle 使用
- Oyente：漏洞扫描工具，可以检测部分已经安全问题，如时间戳依赖、随机数生成等
- Solgraph: 可视化函数调用，高亮风险代码
- Solidity-coverage: Solidity 代码覆盖测试
- Solcheck、Solint：检查代码风格或错误，提高代码质量

智能合约安全课题

- 形式化验证
- 随机数生成
- 隐私保护的智能合约
- 其他安全课题

形式化验证

- 数学上证明程序没有 bug
- 仅支持 Solidity 语法子集，不够完美

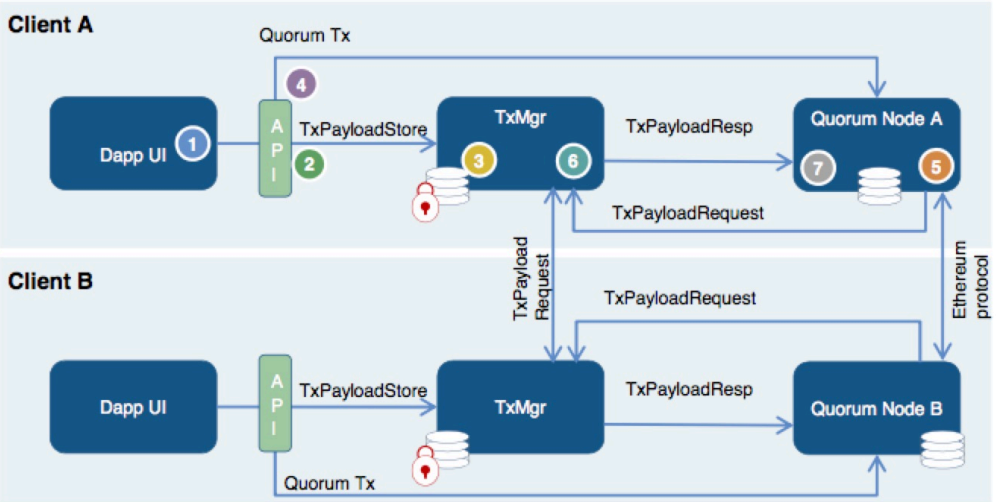


随机数生成

- 智能合约实现随机数发生器
  - randao
  - randao++
  - maker-darts

隐私保护的智能合约

- Quorum：以太坊 + 许可机制 + 数据隐私
- 特定节点解密加密合约调用请求获得明文并执行
  - 隐私保护力度过粗
  - 普通合约与加密合约无法互操作



隐私保护的智能合约

- 安全多方计算
  - 起源：图灵奖得主姚期智于 1980s 提出的百万富翁协议
  - 定义：在一个互相不信任的分布式网络中，两个或多个参与者能够在不泄露各自隐私数据的前提下合作计算某个约定函数并获得计算结果
- 安全多方计算 + 区块链
  - 智能合约处理混淆过的合约调用输入值
  - 特定问题设计特殊协议，依赖零知识证明、承诺等密码机制
  - Ex. 匿名投票协议

隐私保护的智能合约

- 同态加密
  - 1978 年提出，2009 年证明第一个全同态算法
  - $E(a \oplus b) = E(a) \otimes E(b)$
  - 明文运算后加密结果 = 明文加密后运算结果
  - 同态加密尤其全同态加密极其耗时，未达到大规模应用水平
- 同态加密 + 区块链：隐私保护的利器
  - 客户端加密合约调用输入值
  - 智能合约进行密文运算
  - 客户端解密获得运算结果

其他安全课题

- 性能问题加剧 DoS 攻击
  - 以太坊 POW => POS
  - 分片
  - EVM 升级
- 区块链共识算法安全
- 抗量子攻击的区块链
  - 椭圆曲线签名算法可被量子算法攻破，从而伪造签名
  - 采纳后量子密码如格密码构造抗量子攻击的签名方案

主要参考文献

- Nick Szabo. Smart Contracts: Building Blocks for Digital Markets, 1996
- Smart Contracts Alliance. Smart Contract: 12 Use Cases for Business & Beyond, 2016
- Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J], 2008
- Vitalik Buterin. Ethereum White Paper - A Next Generation Smart Contract & Decentralized Application Platform, 2013
- Dr. Gavin Wood. Ethereum Yellow Paper - Ethereum: A secure decentralised generalised transaction ledger, 2014

主要参考文献

- Atzei N, Bartoletti M, Cimoli T. A Survey of Attacks on Ethereum Smart Contracts (SoK)[C]//International Conference on Principles of Security and Trust. Springer, Berlin, Heidelberg, 2017: 164-186
- Luu L, Chu D H, Olickel H, et al. Making smart contracts smarter[C], Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 254-269
- Bhargavan K, Delignat-Lavaud A, Fournet C, et al. Formal verification of smart contracts[C], Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security-PLAS'16. 2016: 91-96

优权招聘

- 招聘岗位： 区块链开发工程师
- 联系邮箱： info@yqtc.co