Ethereum & Smart Contracts

--Enabling a Decentralized Future

LING Zong, Ph. D.
Senior Software Engineer / Scientist
IBM Almaden Research Center
San Jose, California, U.S.A.


ethereum
HOMESTEAD RELEASE
BLOCKCHAIN APP PLATFORM

Ethereum

~~buzzwords~~

overview

BLOCKCHAIN

TRUSTLESS

DECENTRALIZED APPS

SMART CONTRACTS

Overview

What is Ethereum?

- Ethereum is a decentralized platform that runs smart contracts.
- Ethereum is an account-based blockchain.
- Ethereum is a distributed state machine that relies on transactions to move between states.

以太坊是一个运行智能合同的去中心化平台。
Ethereum是一个基于帐户的区块链。
Ethereum是一种分布式状态机，它依赖交易在状态之间移动。

Decentralized: no single point of control/failure; censorship resistant

Blockchain: state is built from a series of blocks, composed of transactions, created by *accounts*, + network consensus

Smart Contracts: more sophisticated scripting that allows for (nearly) arbitrary computation

Transaction-based: state transitions occur on new transactions, which transfer value and information between *accounts*

Account-based: state is made up of *accounts*, with each *account* having an address, some balance of ether, and optionally some contract code and storage

分散:没有单点控制/失败;抗审查的
区块链:状态由一系列块构建，由交易、账号创建+网络共识组成
智能合约:更复杂的脚本，允许(几乎)任意计算
基于交易:状态转换发生在新交易上，它在帐户之间传递值和信息
基于帐户的:状态由帐户组成，每个帐户有一个地址、一些ether的余额和可选的一些合约代码和存储

Some Differences from Bitcoin

- Though **Ethereum** and **Bitcoin** have similar features, both are billed quite differently:
  - Ethereum: **Smart Contract Platform**
  - Bitcoin: **Decentralized Asset**
- Account-based instead of UTXO-based
- Ethereum has a Turing complete scripting language that is significantly more powerful than Bitcoin Script. Enables smart contracts.
- The Ether asset is, in some ways, a side effect of having an incentive-aligned smart contract platform.
- Ethereum plans to move to Proof-of-stake in the near future.

Misc. Implementation details:

Block creation time: (~12 sec vs ~10 min)

Proof-of-work: (Ethash vs Sha256)

Ethash is (currently) ASIC resistant

Exchange Rate: (2016-10-19 00:37 PST)

ETH ⇒ USD : $12.45

BTC ⇒ USD : $635.38

尽管以太坊和比特币有相似的功能，但它们的收费方式却截然不同:
〇以太坊:智能合同平台
〇比特币:分散化资产
基于账户而不是utxo
Ethereum有一个图灵完整的脚本语言，明显比比特币脚本更强大。使聪明的合同。
以太资产，在某种程度上，是拥有一个激励对齐的智能合同平台的副作用。
Ethereum计划在不久的将来转向股权证明

In computability theory, a system of data-manipulation rules (such as a computer's instruction set, a programming language, or a cellular automaton) is said to be Turing complete or computationally universal if it can be used to simulate any Turing machine.

在可计算性理论中,数据操作规则系统(如计算机的指令集,一个编程语言,或细胞自动机)据说是图灵完全或计算通用如果它可以用来模拟任何图灵机

外部拥有的帐户：
· 通常由一些外部实体拥有
· 由地址识别
· 持有一些以太余额(以太货币单位)
· 可以发送交易(传送以太到其他账户，触发合同代码)
合同账户(合同)：
· 由地址识别
· 有一些以太余额
· 有关联的合同代码
· 代码执行由从其他合同接收到的交易或消息(函数调用)触发
· 合同具有持久性存储

Accounts vs. UTXOs

Recall: A Bitcoin user's available balance is the sum of unspent transaction outputs for which they own the private keys to the output addresses.

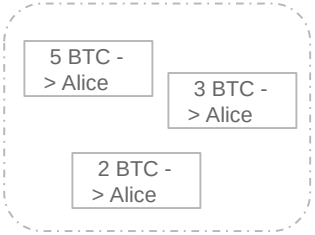
Instead Ethereum uses a different concept, called Accounts.

回想一下:一个比特币用户的可用余额是他们拥有输出地址私钥的未使用交易输出的总和。
相反，Ethereum使用了一个不同的概念，称为帐户

Bitcoin:

Bob拥有UTXOs集合的私钥

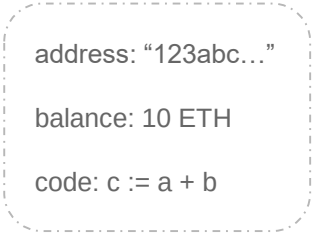
Bob owns private keys to set of UTXOs



Ethereum:

Evan拥有一个账户的私钥

Evan owns private keys to an account



Externally Owned Accounts (EOAs):

- Generally owned by some external entity
- Identified by an ***address***
- Holds some balance of ***ether*** (unit of Ethereum currency)
- Can send transactions (transfer ether to other accounts, trigger contract code)

Contract Accounts (Contracts):

- Identified by an *address*
- Has some ***ether*** balance
- Has associated contract code
- Code execution is triggered by transactions or messages (function calls) received from other contracts
- Contracts have persistent storage

所有帐户==网络状态

All Accounts == Network State

The state of all accounts is the state of the Ethereum network, i.e., the ***entire*** Ethereum network agrees on the current balance, storage state, contract code, etc... of ***every single account***.

The network state is updated with every block.

You can think of the block as the state transition function; it takes the previous state and produces a new network state, which every node has to agree upon.

Accounts interact with the network, other accounts, other contracts, and contract state through transactions.

帐户通过交易与网络、其他帐户、其他合同和合同状态交互

所有账户的状态就是以太网络的状态，即整个以太网络对每一个账户的当前余额、存储状态、合同代码等达成一致……。
每个块都会更新网络状态。
你可以把块看作是状态转换函数;它接受以前的状态并产生一个新的网络状态，每个节点都必须同意这个状态。

账户基本原理

Accounts Rationale

Space Savings: only need to update each account's balance instead of storing every UTXO

Most importantly, smart contracts are more intuitive to program when transferring between accounts with a balance vs. constantly updating a UTXO set to compute user's available balance.

One weakness of the account model is that in order to prevent **replay attacks**, every transaction must have a "nonce".

节省空间:只需要更新每个帐户的余额，而不是存储每个UTXO
最重要的是，与不断更新UTXO集来计算用户可用余额相比，智能合同在编程时更加直观。
帐户模型的一个弱点是为了防止重播攻击，每个交易必须有一个“现时”

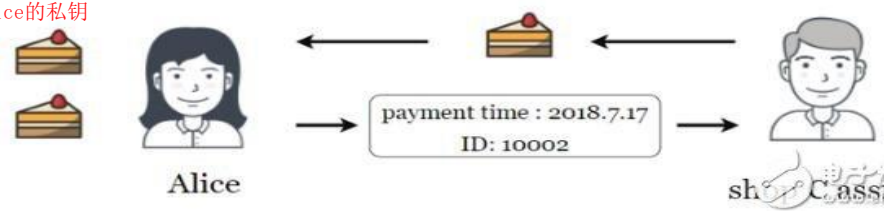
Every account has a nonce attached that increments every time it sends a new transaction. A transaction is valid if its nonce value is 1 + account's nonce.

This means that even no-longer-used accounts can never be pruned from the account state!

Example of a replay attack:

1. Bob receives 5 ETH from Alice.
2. Bob broadcasts the same exact txn…
3. Without a nonce, network sees another totally valid txn! With nonce, Bob can't modify new txn's nonce since he doesn't have access to Alice's private keys.

每个帐户都附加了一个nonce，它在每次发送新交易时递增。如果交易的nonce值为1 + 账户nonce，则该事务有效。
这意味着即使不再使用的帐户也永远不能从帐户状态中删除!
重播攻击的例子:
1。鲍勃从爱丽丝那里收到5个ETH。
2。鲍勃广播完全一样的txn
3。没有nonce，网络看到另一个完全有效的txn!使用nonce，Bob不能修改新的txn的 nonce，因为他不能访问Alice的私钥



Smart Contracts - Introduction

con·tract

(noun) /ˈkäntrakt/

一种书面或口头的协议，尤指有关雇佣、销售或租赁的，旨在由法律强制执行的协议

1. a written or spoken agreement, especially one concerning employment, sales, or tenancy, that is intended to be enforceable by law.

smart con·tract

(noun) /smärt ˈkäntrakt/

促进、验证或强制数字契约的协商或执行的代码

1. code that facilitates, verifies, or enforces the negotiation or execution of a digital contract.

In the case of Ethereum, a smart contract is just an account with code.

Contracts in Ethereum are like autonomous agents that live inside of the Ethereum execution environment, always executing a specific piece of code when “poked” by a transaction or message, and having direct control over their own ether balance and their own permanent state.

在Ethereum的情况下，智能合同只是一个带有代码的账户。

Ethereum中的合同就像生活在Ethereum执行环境中的自治代理一样，总是在被交易或消息戳到时执行特定的代码段，并且直接控制自己的以太余额和自己的永久状态

Smart Contracts in Ethereum

Ethereum Contracts generally serve four purposes:

- **Store and maintain data**, representing something useful to users or other contracts, e.g., a token currency or organization's membership.
- **Manage contract** or relationship between multiple, usually untrusting users, e.g., financial contracts, escrow, insurance.
- **Provide functions** to other contracts, serving as a software library.
- **Serve as an externally owned account** with a more complicated access policy, a.k.a. "forwarding contract". Usually the contract receives incoming messages and forwards them to a certain destination if certain conditions are met, e.g., a multisignature contract that only forwards the message if M-of-N of the key holders approve.

Or some combination of the above!

以太坊合同一般有四个目的：
· 存储和维护数据，表示一些对用户或其他合同有用的东西，例如，代用货币或组织的会员资格。
· 管理多个，通常是不信任的用户之间的合同或关系，例如，金融合同，托管，保险。
· 为其他合同提供功能，作为一个软件库。
· 作为一个外部拥有的帐户，有一个更复杂的访问策略，也就是转发合同。通常，合同接收传入的消息，并在满足某些条件时将其转发到某个目的地，例如，多签名合同仅在密钥持有者的M-of-N批准时才转发消息。
· 或者以上几种情况的组合

Ethereum Virtual Machine

The Ethereum contract code that actually gets executed on every node is so-called EVM code, a low-level, stack-based byte-code language.

Every Ethereum node runs the EVM as part of its block verification procedure.

EVM as a state transition mechanism:

在每个节点上实际执行的Ethereum合同代码是所谓的EVM代码，这是一种低级的、基于堆栈的字节码语言。
每个Ethereum节点运行EVM作为其块验证过程的一部分。
EVM是一种状态转换机制

(block_state, gas, memory, transaction, message, code, stack, pc)

(block_state', gas')

其中block_state是全局状态，包含所有帐户，包括余额和长期存储

where block_state is the global state containing all accounts and includes balances and long-term storage

EVM Design Goals:

Simplicity: op-codes should be as low-level as possible. The number of op-codes should be minimized.

Determinism: The execution of EVM code should be deterministic; the same input state should always yield the same output state.

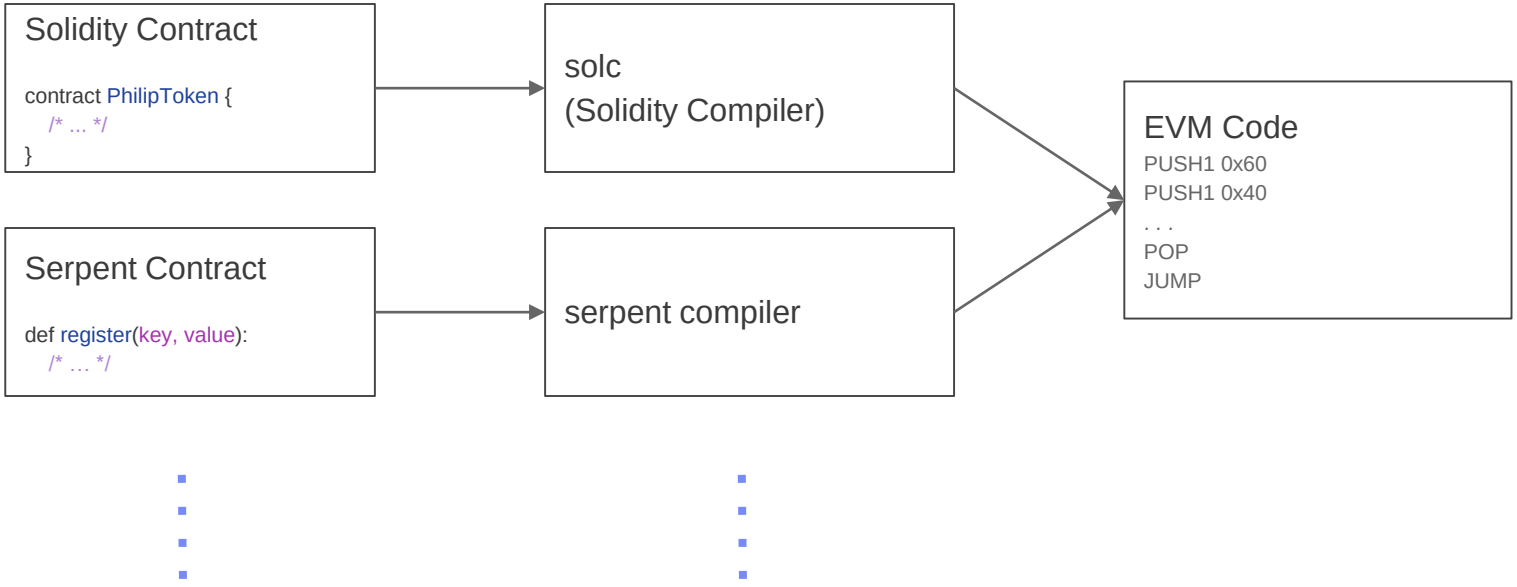
Space Efficiency: EVM assembly should be as compact as possible

Specialization: easily handle 20-byte addresses and custom cryptography with 32-byte values, modular arithmetic used in custom cryptography, read block and transaction data, interact with state, etc

Security: it should be easy to come up with a gas cost model for operations that makes the VM non-exploitable

· 简单性:操作码应该尽可能低层。操作码的数量应该最小化。
· 确定性:EVM代码的执行应该是确定性的;相同的输入状态应该总是产生相同的输出状态。
· 空间效率:EVM组装应尽可能紧凑
· 专门化:轻松处理20字节地址和自定义加密32字节值，自定义加密使用的模块化算术，读取块和交易数据，与状态交互，等等
· 安全性:为使VM不可利用的操作提供一个gas成本模型应该很容易

EVM Code Compilation



EVM Gas and Fees

直接的问题

Immediate Issue:

如果我们的合同有一个无限循环呢?

What if our contract has an infinite loop?

```
function foo()
{
   while (true) {
      /* Loop forever! */
   }
}
```

Every node on the network will get stuck executing the loop forever! By the halting problem, it is impossible to determine ahead of time whether the contract will ever terminate ⇒ Denial of Service Attack!

网络上的每个节点都将永远执行这个循环!由于中止问题，不可能提前确定合同是否会终止->拒绝服务攻击

Ethereum's Solution:

Every contract requires "gas", which "fuels" contract execution.

Specifically, every EVM op code requires a certain amount of gas in order to execute.

Every transaction specifies the startgas, or maximum quantity of gas it is willing to consume, and the gasprice, or the fee in ether it is willing to pay per unit gas.

每个合同都需要燃气，这是合同执行的燃料。
具体来说，每个EVM操作代码都需要一定量的燃气才能执行。
每笔交易都规定了起始汽油量，即它愿意消费的最大汽油量，以及汽油价格，或每单位汽油它愿意支付的费用

EVM Gas and Fees

At the start of the transaction, startgas * gasprice ether are subtracted from the sender's account.

If the contract successfully executes and uses less than the prespecified amount of gas, the remaining gas is refunded to the sender.

If the contract execution runs out of gas before it finishes, then execution reverts and startgas * gasprice are not refunded.

在交易开始时，startgas * gasprice以太将从发送方的帐户中扣除。
如果合同成功执行，使用的天然气少于预先规定的数量，剩余的天然气将退还给发送方。
如果合同执行在结束前耗尽了汽油，那么执行恢复和startgas * gasprice不退款

What about the infinite loop?

Ethereum still allows the infinite loop; however, whoever is attempting to DoS the network has to pay enough ether to fund the DoS.

Think of purchasing gas as purchasing distributed computational power.

那么无限循环呢?
以太坊仍然允许无限循环;然而，无论谁试图DoS网络都必须支付足够的ether来资助DoS。
可以将购买gas看作是购买分布式计算能力

Ethereum Smart Contracts

- Ethereum is ***not*** about optimising efficiency of computation

Ethereum不是关于优化计算效率的
它的并行处理采用冗余并行，提供了一种无需信任第三方即可对系统状态达成一致的有效方式。
由于合同执行是跨节点冗余复制的，因此开销很大，这通常会导致不使用区块链可以在链外完成的计算

- Its parallel processing is redundantly parallel to offer an efficient way to reach consensus on the system state without needing trusted third parties.

- Since contract executions are redundantly replicated across nodes, they are expensive, which generally creates an incentive not to use the blockchain for computation that can be done off chain.

Smart Contracts

Use Cases and Analysis

TOKENs

PUBLIC DATABASES

CROWDFUNDING

FILE STORAGE

MARKETs

BLOCKCHAIN IOT

AUTOMATED SHARING ECONOMY

DAOs

Basic Use Cases

Token Systems 令牌系统

- Very easy to implement in Ethereum
- **Database with one operation**
  - Ensure Alice has enough money and that she initiated the transaction
  - Subtract X from Alice, give X to Bob

在以太坊中很容易实现
一个操作的数据库
○确保爱丽丝有足够的钱，并由她发起交易
○从爱丽丝减去x，x给鲍勃

Example (from Ethereum white paper):

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

Token

```
contract PhilipToken {
    /* Maps account addresses to token balances */
    mapping (address => uint256) public balanceOf;
    /* Initializes contract with initial supply
       tokens to the creator of the contract */
    function PhilipToken(uint256 initialSupply)
    {
        // Give the creator all initial tokens
        balanceOf[msg.sender] = initialSupply;
    }
    /* Send tokens to a recipient address */
    function transfer(address to, uint256 value)
    {
        if (balanceOf[msg.sender] < value) throw;          // Check if the sender has enough
        if (balanceOf[to] + value < balanceOf[to]) throw;  // Check for overflows
        balanceOf[msg.sender] -= value;                    // Subtract from the sender
        balanceOf[to] += value;                            // Add the same to the recipient
    }
}
```

最小可行的令牌

Minimum Viable Token

PhilipToken is a stripped down version of a digital token contract, written in Solidity.

You can instantiate it with some initial supply of tokens which can be transferred between different accounts.

Remember, **the contract is an account!** It has its own ether balance, address, storage, etc...

PhilipToken是一个精简版的数字令牌合同，以可靠的方式书写。
您可以用一些可以在不同帐户之间传输的初始令牌来实例化它。
记住，合同就是账户!它有自己的以太余额，地址，存储，等等…

公共注册表/公共数据库

Public Registry / Public database

Example: Namecoin

- DNS system
  - Maps domain name to IP address
  - "maxfa.ng" => "69.69.69.69"
- Immutable
- Easy implementation in Ethereum

DNS系统
〇域名到IP地址的映射
〇”maxfa.ng”=>“69.69.69.69”
不可变的
在以太坊易实现

Example (from Ethereum white paper):

```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

集资和激励

Crowdfunding and Incentivization

Simple Example: "Ether-on-a-stick"

- Allows you to put a bounty on the completion of arbitrary tasks
- Contributors pool money into a smart contract that pays out to a specified recipient iff contributors vote that the task was indeed complete

Example use case

- A company is polluting a local river and nearby residents bear a negative externality
  - Local gov't slow/unresponsive but residents are willing to pay
- Residents pool money together to incentivize the company to clean it up

Implements a Dominant Assurance Contract: solves the free rider problem

- https://en.bitcoin.it/wiki/Dominant_Assurance_Contracts

简单的例子:“ Ether-on-a-stick ”
·允许你对完成任意任务的人给予奖励
·贡献者将资金汇集到一个智能合同中，如果贡献者投票认为任务确实完成了，该合同将支付给指定的接受者

示例用例
·一家公司污染了当地的一条河流，附近的居民就承担了负外部性
  ○当地政府反应不迟钝,但居民愿意支付
·居民们将资金集中起来，激励公司进行清理
实行显性保险合同:解决搭便车问题

Advanced Use Cases

Decentralized File Storage

“分散式Dropbox合同”:支付个人少量以太出租额外的硬盘空间

"Decentralized Dropbox Contract": Pay individuals small amounts of Ether to rent out extra hard drive space

Example contract specification:

- Split cat picture into blocks, encrypt each block for privacy
- Create Merkle tree from blocks, save Merkle root in contract
- Every N blocks, the contract will:
  - Using previous block header (source of randomness), pick a random block in Merkle tree
  - First entity to provide proof of storage of block (Merkle branch) receives small ETH reward

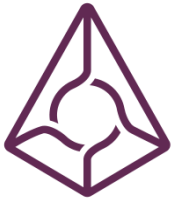
Recovering the file:

- Query node storing file and pay a small fee (via micropayment channels) to retrieve it
- >Decrypt data, obtain furry kitten
- >Profit

例如合同规范:
分割猫图片成块，加密每个块的隐私
从模块中创建Merkle树，在合同中保存Merkle根目录
每N块，合同将:
〇使用先前的块头(随机来源)，在Merkle树中选择一个随机块
〇第一个提供仓储证明的实体(Merkle分行)获得小额ETH奖励
恢复的文件:
· 查询节点存储文件，并支付少量费用(通过微支付渠道)来检索文件
· 解密数据，获得毛茸茸的小猫
· 获利

分散的市场预测

Decentralized Prediction Markets



augur

Prediction markets draws on the wisdom of the crowd to forecast the future

- Market makers create event
  - Ex: "Who will win the 2020 US Presidential election?"
  - Events must be public and easily verifiable, with set due date.
- Participants buy shares of Trump or Biden and pay a small fee
- On election day, random oracles on the network vote on who won.
  - Oracles who voted with the majority collect a fee, they are otherwise penalized
- Shareholders who voted correctly cash out on their bet

The share price for each market accurately represents the best predicted probability of event occurring

- Someone has extra information => arbitrage opportunity

预测市场利用大众的智慧来预测未来
造市商制造事件
· (谁将赢得2020年美国总统大选?)
· 事件必须是公开的，容易核实的，并设定截止日期。
参与者购买特朗普或拜登的股票，并支付少量费用
选举日那天，网络上的随机先知们投票决定谁赢了
· 和大多数人一起投票的神谕收费，否则就会受到惩罚
投票正确的股东兑现了他们的押注
每个市场的股价准确地反映了事件发生的最佳预测概率
某人有额外的信息=>套利机会

GNOSIS

Decentralized Prediction Markets

augur

Use cases

- **Cost efficient way to buy information on a future event**
  - Instead of hiring pundits and experts, create a market for your event
  - "Will this movie be a flop?"
  - Bet for and against your event to incentivize people who have information about this event (in this case, Hollywood insiders)
- **Hedging and insurance**
  - Fire insurance is a bet that your house will burn down
  - Create market "Will my house burn down?" and vote yes
  - => receive compensation if your house burns down
  - Possible to implement an entire insurance liquidity pool
  - Potential for extremely thin margins since no central intermediary is required

以经济有效的方式购买关于未来事件的信息
〇不要雇佣专家和专家，而要为你的活动创造市场
〇(这部电影会失败吗?)
〇支持和反对你的活动，以激励对该活动有了解的人
（这里是好莱坞内部人士)
套期保值和保险
〇火灾保险就是赌你的房子会被烧毁
〇创造市场“我的房子会被烧毁吗？”并投赞成票
〇如果你的房子被烧毁，你可以得到赔偿
〇有可能实施一个完整的保险流动性池
〇由于不需要中央中介，极薄的边际潜力

GNOSIS

Decentralized Prediction Markets

augur

Use cases

- **Set up a security bug bounty**
  - "Will my company be hacked?" Bet heavily against it to create a financial incentive
  - Someone who finds vulnerability will buy affirmative shares, then perform their hack
    - > Profit
  - Augur secures their own code this way
    - "Will someone be able to steal the money in this prediction market?"
- **Signaling: "Put your money where your mouth is"**
  - Demonstrate your commitment to something by showing you will take a large financial loss if you miss your commitment
  - Ex. Kickstarter campaign; investors are worried you will delay launch date
    - "Will my Kickstarter campaign launch on time?"
    - Bet heavily that you WILL launch your produce on time.

设置一个安全漏洞奖励
〇(我的公司会被黑吗?)大举做空它，以创造一种财务激励
〇发现漏洞的人会购买肯定的股票，然后进行攻击。获利
〇Augur通过这种方式确保他们自己代码的安全
“有人能在这个预测市场偷钱吗?”
信号:“言行一致”
〇展示你对某事的承诺，表明如果你错过承诺你将会承受巨大的经济损失
〇Kickstarter活动;投资者担心你会推迟上市日期
“我的Kickstarter活动会准时启动吗?”
打赌你会按时发布你的产品

GNOSIS

Decentralized Prediction Markets

augur

分散的好处
不限制市场创造
但也引发了道德问题
共享的流动性池
没有理由同一个市场应该存在于多个国家
考虑到更先进的市场;
如。组合预测市场
抗审查的档案
自动的,不可靠的支付

Benefits to being decentralized

- No restrictions on market creation
  - But raises ethical questions
- Shared liquidity pool
  - No reason why the same market should exist in multiple countries
  - Allows for more advanced markets;
    e.g. combinatorial prediction markets
- Censorship-resistant
- Automatic, trustless payments

GNOSIS

Decentralized IoT

FILAMENT

Filament

- "Blockchain-based decentralized Internet of Things"
- "Ad hoc mesh networks of smart sensors"
- Intended for industrial IoT applications

Product

- Sensors with 10 mile range
- battery lasts years
- no internet connection needed - uses mesh networking

灯丝
“基于区块链的分散物联网”
“智能传感器的特设网状网络”
用于工业物联网应用
产品
10英里范围的传感器
电池持续多年
没有互联网连接需要-使用网状网络

Technologies used:

- Telehash - end-to-end message encryption
- TMesh - self-forming radio mesh networks
- Blockname - private device discovery
  - Uses Bitcoin blockchain + public notaries to verify authenticity of name/address bindings
- Blocklet - smart contracts and microtransactions

Exchange

Value can be exchanged between devices in the form of data, network access, currencies such as Bitcoin, compute cycles, contracts for ongoing service, trusted introductions to other devices, and more.

Filament is a great application of decentralized tech especially because of its emphasis on resilience and dependability.

技术:
Telehash -端到端信息加密
TMesh-自形成的无线电网状网络
Blockname-私有设备发现
使用比特币区块链+公证员验证姓名/地址绑定的真实性
Blocklet——智能合同和微交易
灯丝是分散式技术的一个重要应用，尤其因为它强调弹性和可靠性。

Slock.it: 它是一种可以通过支付直接打开的锁
业主设定一个押金+价格
租户支付押金+价格进入连接到以太坊节点的锁
锁检测支付和解锁自身
用例(Slock.it):
全自动Airbnb公寓
不需要与所有者见面取钥匙
按需租用Wifi路由器
完全自动化的商店
购买商品时，把商品的价格送到锁上
自动自行车出租

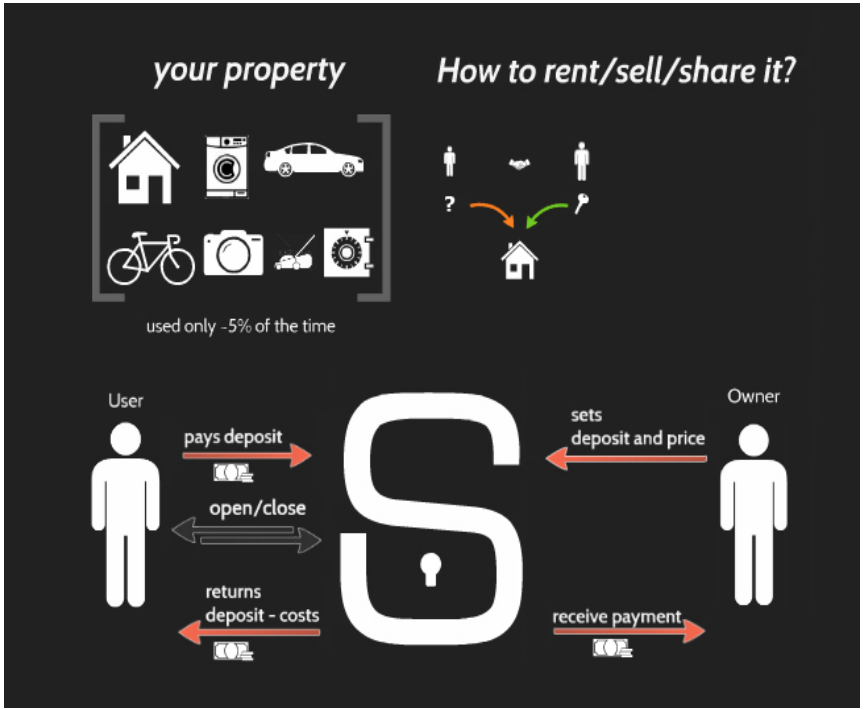
Decentralized Sharing Economy

Slock.it: A lock that can be directly opened by paying it

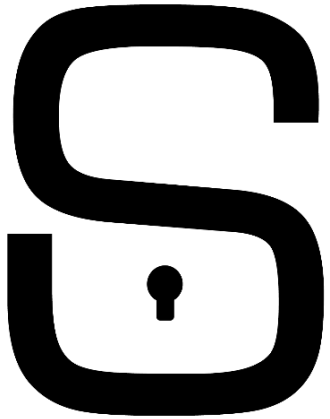
- Owner sets a deposit + price
- Renter pays deposit + price into lock connected to Ethereum node
- Lock detects payment and unlocks itself

Use Cases (Slock.it):

- Fully automated Airbnb apartments
  - no need to meet with owner for key
- Wifi routers rented on demand
- Fully automated shop
  - Purchase goods by sending the price of the good to the lock that holds it
- Automated bike rentals



Decentralized Sharing Economy



Benefits of being decentralized (Slock.it):

- ...Trustless?
  - Decentralized reputation is very hard to implement
  - Centralized solutions probably better
- Programmable money
  - Centralized solutions still programmable
  - Public blockchains (Bitcoin/Ethereum etc) have easier technological integration
  - No personal information needed to conduct transactions
- IoT: Device autonomy
  - Devices can act independently of a central management system
  - Modular
- ???

分散化的好处(slock.it):
不可靠的?
〇去中心化声誉很难实施
〇集中解决方案可能更好
可编程的钱
〇集中解决方案仍然可编程
〇公共区块链(比特币/以太坊等)更容易技术整合
〇交易不需要个人信息
物联网:设备自主
〇设备可以独立于中央管理系统
〇模块化

from slock.it

分散式自治组织

Decentralized Autonomous Organizations

A Decentralized Autonomous Organization, or **DAO**, is an organization governed entirely by code (smart contracts)

- Create and vote on proposals
- Businesses can theoretically exist entirely on a blockchain
  - Uncertain legal status
- "Code is law"

Issues:

- Hard to edit the governing laws (the code) once deployed
  - Possible solution: Child DAO
- Inactive participants: not enough people vote on proposals

分散式自治组织(或DAO)是完全由代码(智能合同)治理的组织
· 创建提案并投票
· 从理论上讲，企业可以完全依靠区块链生存
  不确定的法律地位
· “ 代码就是法律 ”
问题:
· 一旦部署，就很难编辑适用的法律(代码)
  可能的解决方案:子DAO
· 不活跃的参与者:没有足够的人对提案进行投票

DASH
· Privacy-centric cryptocurrency
· 采矿报酬
TheDAO
· 人群风险基金
· 历史最大的众筹项目
  在以太网上筹得1.5亿美元
· 2016年6月，价值6000万美元的以太被盗(占Ethereum市值的10%)
其他DAOs在以太坊:
· Slock.it
· Digix.io
  在以太坊里储存黄金

DASH

- Privacy-centric cryptocurrency
- % of mining reward

TheDAO

- Crowd venture fund
- Largest crowdfunded project in history
  - Raised >$150 million in Ether
- Hacked in June 2016, >$60 million worth of Ether stolen (10% of Ethereum market cap)

Other DAOs on Ethereum:

- Slock.it
- Digix.io
  - Store gold on the Ethereum blockchain

Generalizations

Limitations of Smart Contracts and Blockchain tech

没有不可靠的访问外部数据的方法
必须依靠oracles来提供区块链之外的信息
〇问题：Oracles必须可信
可能的解决方案:已证明的执行(不受信任的oracles)
〇Oraclize。它的执行很糟糕
TLSnotary -修改TLS协议，提供接收https页面的密码证明
可能的解决方案:Oracle网络对信息进行投票
〇缺点:共识协议之上的共识协议
〇激励/声誉很难对齐

No trustless way to access outside data

- Must rely on oracles to provide information from outside the blockchain
  - Problem… Oracles must be trusted
- Potential Solution: Proven execution (untrusted oracles)
  - Oraclize.it has a shoddy implementation
    - TLSnotary - modification of TLS protocol to provide cryptographic proof of receiving https page
- Potential Solution: Oracle network votes on information
  - Drawback: Consensus protocol on top of a consensus protocol
  - Hard to align incentives/reputation

No way to enforce on-chain payments

- Cannot implement financial products like loans and bonds
  - Money must be held on blockchain to ensure payment
- Intuition: We pay interest on loans partially because of risk of default

Contracts cannot manipulate confidential data

- Confidential data cannot be assembled on someone else's computer
- Very limited access control capabilities
- Can only store encrypted data and decrypt it locally
- Potential solution: Homomorphic encryption

没有办法强制执行链上支付
不能实施贷款、债券等金融产品
〇资金必须保存在区块链上以确保付款
直觉:我们支付贷款利息的部分原因是违约风险
合同不能操纵机密数据
机密数据不能在其他人的计算机上组装
非常有限的访问控制能力
只能存储加密的数据并在本地解密
可能的解决方案:同态加密

区块链杀手级应用程序的基本属性

Essential Properties of Blockchain Killer Apps

- Dapps可以看作是客户端软件——没有中央管理器
- 需要共识或协调的无信任环境(如智能电网、能源市场)
- 以隐私为中心的系统(比如社交网络?)
  虽然数据不应该存储在区块链本身
- 可编程货币与开放集成
  物联网、M2M支付(e.x. IBM ADEPT)
  容易发送和接收金钱-没有个人信息需要
  可能的小额支付(如Brave)

容错、弹性系统(如灯丝)
➢自治网络和设备
创造激励的新方法(例如灵知)
新的治理模型(例如DAOs、futarchy)
非中介化，抵制审查
信任数学和代码，而不是机构
集中对比：
深度集成，内聚的用户体验
➢效率——区块链整体缓慢
➢完全控制数据和读/写权限

Dapps can be thought of as client-side software - no central manager

Trustless environments that need consensus or coordination (ex. Smart grids, energy markets)

Privacy-centric systems (ex. social networks?)
- Although data shouldn't be stored on the blockchain itself

Programmable money with open integration
- IoT, M2M payments, (e.x. IBM ADEPT)
- Easy to send and receive money - no personal information required
- Micropayments possible (ex. Brave)

Fault-tolerant, resilient systems (ex. Filament)
- Autonomous networks and devices

News ways to creating incentives (ex. Gnosis)

New governance models (ex. DAOs, futarchy)

Disintermediation, censorship-resistance

Trust in math and code, not institutions

Contrast with centralization:

Deep integration, cohesive user experience
- Efficiency - blockchains are slow in general
- Full control over data and read/write permissions

经常问:为什么使用区块链比使用中央数据库更好?

ALWAYS ASK: Why is using a blockchain better than a central database?

Community, Regulation, and Controversy

Readings

- (Wiki) Know Your Customer
  - https://www.wikiwand.com/en/Know_your_customer
- (Wiki) BitLicense
  - https://www.wikiwand.com/en/BitLicense
- (Article) BitLicense 2.0
  - http://www.coindesk.com/bitlicense-2-0-latest-revisions-mean-bitcoin-businesses/
- (Article) Overview of the Blocksize Debate:
  - http://www.coindesk.com/making-sense-block-size-debate-bitcoin/

Optional reading:

- A current list of use cases for Ethereum (medium article)
  - https://medium.com/@AroundTheBlock_/a-current-list-of-use-cases-for-ethereum-b8caa5807553#.2epaf2jud
- Demystifying Incentives in the Consensus Computer: Investigates game-theoretic incentive problem in Ethereum similar to our last lecture
  - https://eprint.iacr.org/2015/702.pdf

END !


धन्यवाद
Hindi
多謝
Traditional Chinese
ขอบคุณ
Thai
Спасибо
Russian
Gracias
Spanish
Thank You
English
شكراً
Arabic
Obrigado
Brazilian Portuguese
多谢
Simplified Chinese
Danke
German
Grazie
Italian
Merci
French
நன்றி
Tamil
ありがとうございました
Japanese
감사합니다
Korean

References (incomplete)

- Ethereum White Paper
- Ether-on-a-stick
  - https://github.com/phlip9/ether-on-a-stick
- Gnosis Use Cases presentation by Martin Koppelmann
  - http://www.slideshare.net/MartinKppelmann/gnosis-vision-and-crowdsale

Lemmas 1 & 2

计算能力需要电力，需要$$，如果矿商是收支平衡或盈利
引理1:采矿奖励=采矿成本
如果你的投资基本达到了收支平衡，那么获得更高的哈希速率几乎没有边际成本。你只是需要更多的资本来达到51%的目标
引理2:收购51%的成本<采矿成本

Computational power requires electricity, requires $$, reaches equilibrium if miners are breaking even or profitable

- Lemma 1: Mining Reward = Mining Cost

If you are roughly breaking even with the capital you invest, there is little to no marginal cost to getting more hashrate. You simply need more capital to attain 51%

- Lemma 2: Cost of acquiring 51% < Mining cost

Lemma 3

你能从拥有哈希率>51%的股份中获得什么利润
·崩溃的货币吗?没有问题。通过在交易所做空比特币，重获价值(然后部分重获价值)
你可以有效地获得100%的采矿奖励
只有在你自己的区块上挖掘
〇可以阻止任何人开采-你总是生产最长的PoW链
这将如何影响价格取决于门槛
q = 51% =>49%的块是孤立的
q = 80% =>20%的块是孤立的
〇普通比特币用户并未受到影响，但仍能进行交易
引理3: 51% attack的价值>挖掘奖励

What profits can you get from owning >51% of the hashrate

- Crash the currency? No problem. Regain value (and then some) by shorting Bitcoin on an exchange

You can effectively get 100% of the mining reward

- Only mine on your own blocks
  - Can prevent anyone else from mining - you always produce longest PoW chain

How this would affect the price depends on threshold

- q = 51% => 49% of blocks are orphaned
- q = 80% => 20% of blocks are orphaned
  - Average Bitcoin user not really affected, still able to make transactions

Lemma 3: Value of 51% attack > Mining Reward

Combining Lemmas

Lemmas:

- Lemma 1: Mining Reward = Mining Cost
- Lemma 2: Cost of acquiring 51% < Mining Cost
- Lemma 3: Value of 51% attack > Mining Reward

Therefore, Value of 51% attack > Cost of acquiring 51%

If math is correct, Game Theory says that 51% attacking Bitcoin is profitable

如果数学是正确的，博弈论认为51% 攻击比特币是有利可图的

(Originally presented by Martin Koppelmann at SF Bitcoin Devs Seminar)

Additional Ideas

Insurance Contracts

- A way to offset costs incurred by mining pools
- More orphaned blocks are a sign of pool wars
- Insurance contracts for bitcoin stakeholders based on number of orphaned blocks

保险合同
· 一种抵消采矿池成本的方法
· 更多的孤立块是泳池战争的标志
· 基于孤立块数量的比特币利益相关者的保险合同

泛化的漏洞

Generalization of Vulnerabilities

Bitcoin mining is zero sum

- In general, to increase earnings, someone else needs to be excluded

Members-only Mining

- Let hashrate join a collusion until 80% of the network is in, then exclude the rest
- No incentive not to join
  - Attack succeeds, get increased reward
  - Attack wouldn't fail: conduct attack in such a way that it wouldn't start until the threshold is reached

Naive Example

- 3 pools collude, own more than 51%
- Ignore every 10th block of another pool
- Undetectable

More profitable than honest strategy

比特币挖掘是零和的
一般来说，为了增加收入，需要把其他人排除在外
会员制矿业
让hashrate加入一个共谋，直到80%的网络都在内，然后排除其余
没有理由不加入
〇攻击成功,得到了奖励
〇攻击不会失败:以某种方式进行攻击，直到达到阈值才会开始
天真的例子
· 三池串通，拥有51%以上
· 忽略另一个池的第10个块
· 检测不到
比诚实的策略更有利可图

Post-block Reward Bitcoin

Assumption: average Bitcoin user holds $100,000 in Bitcoin, willing to pay $1000 in fees

- (This is when Bitcoin is near 0 block reward)
- Is mining based off transaction fees sustainable?
- Money must move, must be paid in transaction fees so that miners can collect it as mining reward
- Amount of hashpower going into Bitcoin dependent on mining reward

假设:比特币用户平均持有10万美元比特币，愿意支付1000美元费用
·(此时比特币块奖励接近0)
·基于交易费用的采矿是否可持续?
·资金必须流动，必须以交易费的形式支付，这样矿工就可以将其作为采矿报酬收取
·进入比特币的哈希力量的数量取决于采矿奖励
因此
·(平均支付费用)/ (avg控股)=(攻击成本)/(市值)
·在我们的例子中，攻击者只需支付比特币市值的1%就可以进行攻击
奖励后的比特币必须有一个高速度的钱是安全的

Therefore

- (average fees paid) / (avg holdings) = (cost of attacking) / (market cap)
- In our example, attacker only needs to pay 1% of the market cap of Bitcoin to attack

Post reward Bitcoin must have a high velocity of money to be secure