# ETHEREUM AND SMART CONTRACTS: ENABLING A DECENTRALIZED FUTURE

Gloria Zhao
Aparna Krishnan

# LECTURE OUTLINE

## BLOCKCHAIN FUNDAMENTALS

1. SMART CONTRACTS
2. ETHEREUM
3. EVM
4. USE CASES
5. GENERALIZATIONS

# 1 SMART CONTRACTS

...but first, a question:

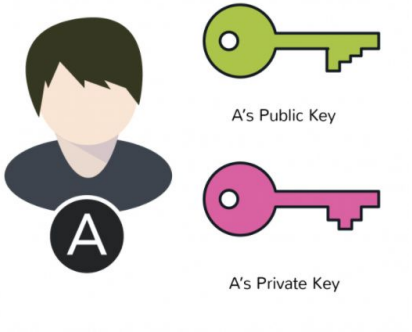
# What makes Bitcoin so special?





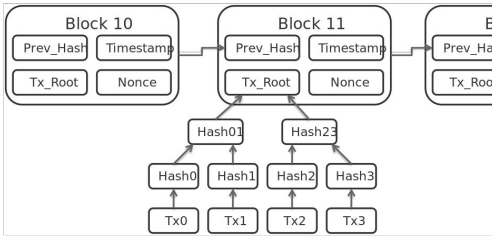
# A DISTRIBUTED NETWORK

## BITCOIN'S BARE BONES




Cryptographic Identities

Consensus Protocol

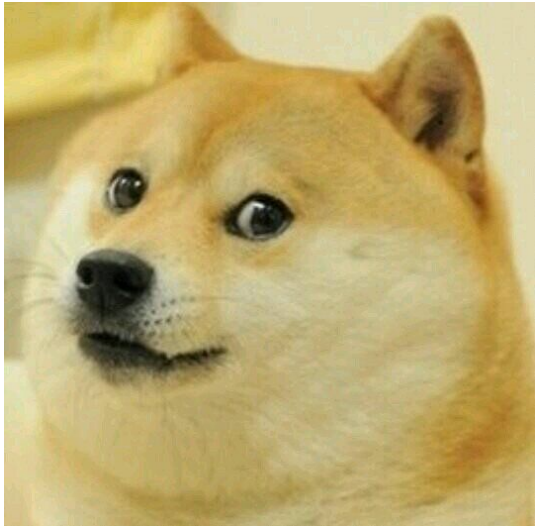



Blockchain

# A DISTRIBUTED NETWORK

## TRANSFERRABLE BENEFITS OF BITCOIN

- Pseudonymous, cryptographic identities allow for accountability
- Democratic decisions made through consensus protocol that **doesn't require trust**
- Immutable ledger of truth
- Uncensorable, cannot be controlled by any one party
- Distributed: no central point of failure



# SMART CONTRACTS

## CONTRACTS

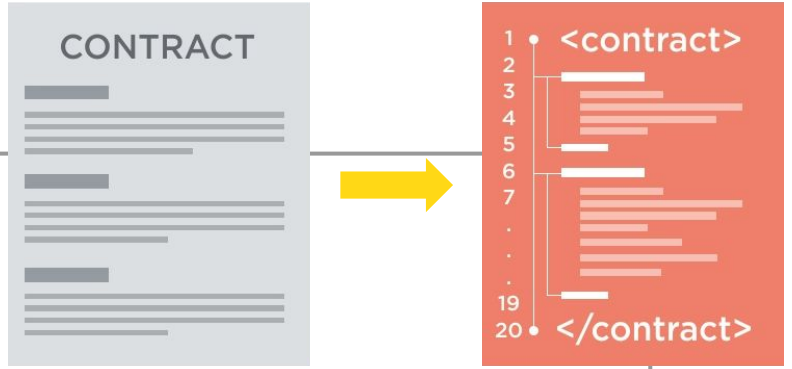
**con·tract**

(noun) /ˈkäntrakt/

1. a written or spoken agreement ... that is intended to be enforceable by law.

# SMART CONTRACTS

## CODE IS LAW




**smart con·tract**

(noun) /smärt ˈkäntrakt/

1. code that **facilitates**, **verifies**, or **enforces** the negotiation or execution of a digital contract.
   a. **Trusted entity** must run this code

# 2 ETHEREUM


ethereum
HOMESTEAD RELEASE
BLOCKCHAIN APP PLATFORM

# WHAT IS ETHEREUM?

## HIGH-LEVEL OVERVIEW

- Ethereum is a **decentralized** platform designed to run **smart contracts**
  - Like a distributed computer to execute code
  - Account-based blockchain
  - **Distributed state machine - transactions change global state**
    - transactions == state transition function
- Ethereum has a native asset called **ether**
  - basis of value in the Ethereum ecosystem
  - needed to align incentives, given as mining rewards

# WHO WOULD WIN?

## Bitcoin

- First successful cryptocurrency
- Trustless
- Immutable
- Uncensorable
- Pseudonymous
- No central point of failure
- One-CPU-One-Vote

1 turing-complete boi

# WHAT IS ETHEREUM?

## COMPARISON WITH BITCOIN

### Bitcoin

- The "Gold Standard" of blockchains
- Asset: bitcoins
  - Primary purpose of the Bitcoin blockchain
- Simple and robust
- Stack-based, primitive scripting language, not Turing-complete
- UTXO-based
- Will likely remain Proof-of-Work

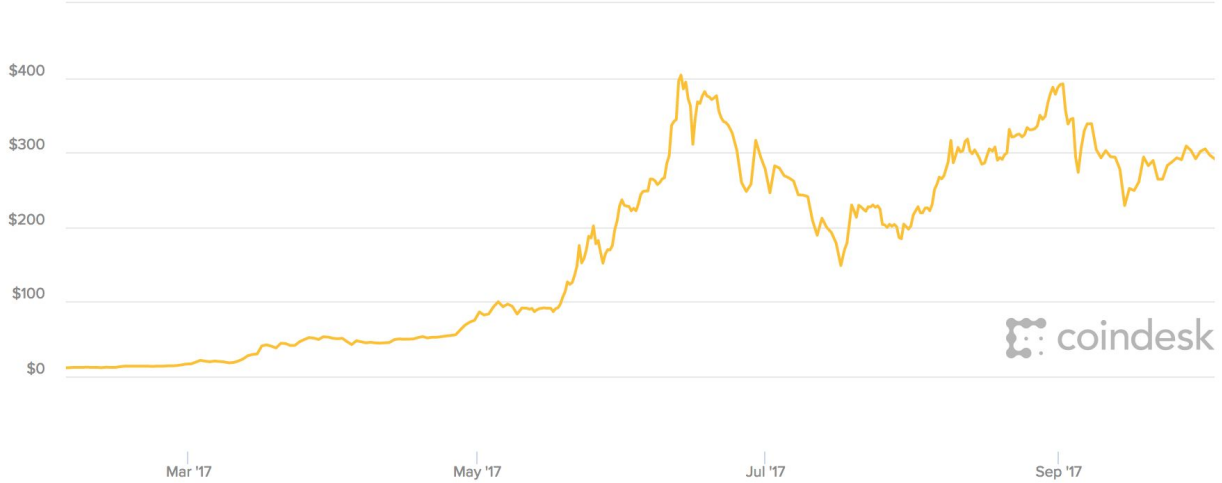
### Ethereum

- Smart Contract Blockchain Platform
- Asset: ether
  - Secondary purpose, used to align incentives
- Complex and feature-rich
- Turing-complete scripting language
- Account-based
- Planning to move to Proof-of-Stake

# WHAT IS ETHEREUM?

## COMPARISON WITH BITCOIN

- **Misc. Implementation Details**
  - Block creation time: ~12 sec vs ~10 min
  - Proof-of-Work: Ethash vs SHA-256 (currently ASIC resistant)
  - Exchange Rate: $291.92 (2017-10-03  16:18 PST)



# ETHEREUM ACCOUNTS

## ACCOUNTS VS UTXO MODEL

Easy to make transactions and prevent double spending

**Bitcoin:**

Bob owns private keys to set of UTXOs

- 5 BTC ⇒ Bob
- 3 BTC ⇒ Bob
- 2 BTC ⇒ Bob

**Ethereum:**

Alice owns private keys to an account

address: “0xfa38b...”

balance: 10 ETH

code: c := a + b

# ETHEREUM ACCOUNTS

## ACCOUNTS RATIONALE

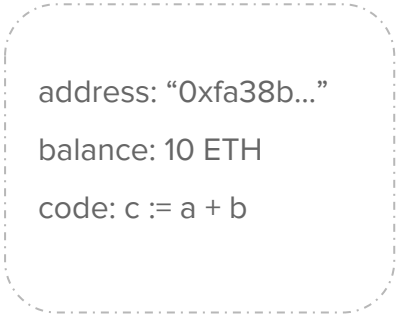
Easy to make transactions and prevent double spending

**Bitcoin:**

Bob owns private keys to set of UTXOs


**Ethereum:**

Alice owns private keys to an account

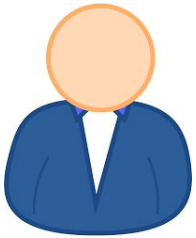


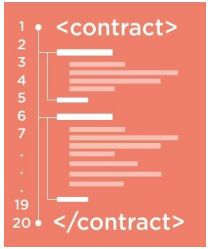

Space-efficient to update balances instead of storing UTXOs

Easier to look up balance and transfer between accounts when programming

# ETHEREUM ACCOUNTS

## ACCOUNT TYPES






### Externally Owned Accounts

- Generally owned by some external entity (person, corporation, etc.)
- Can send transactions to transfer ether or trigger contract code
- Contains:
  - Address
  - Ether Balance

### Contract Accounts

- “Owned” by contract
- Code execution triggered by transactions or function calls (msg)
- Contains:
  - Address
  - Associated contract code
  - Persistent storage

# ETHEREUM SMART CONTRACTS

## CONTROL

Contracts in Ethereum are like autonomous agents that live inside of Ethereum network

- React to external world when "poked" by transactions (which call functions)
- Have direct control over:
  - **internal ether balance**
  - **internal contract state**
  - **permanent storage**

# ETHEREUM SMART CONTRACTS

## SMART CONTRACTS IN ETHEREUM

- Ethereum Contracts generally serve four purposes:
  - **Store and maintain data**
    - Data represents something useful to users or other contracts
    - ex: a token currency or organization's membership
  - **Manage contract or relationship between untrusting users**
    - ex: financial contracts, escrow, insurance
  - **Provide functions to other contracts**
    - serving as a software library
  - **Complex Authentication**
    - ex: M-of-N multisignature access
- Some combination of the above!

# SMART CONTRACTS

## SAMPLE BETTING CONTRACT (PARTIAL)

```
contract Betting {

    address public owner;
    address public gamblerA, gamblerB, oracle;
    uint[] outcomes;

    struct Bet {                          /* Defines a Bet */
        uint outcome;
        uint amount;
        bool initialized;
    }

    mapping (address => Bet) bets;        /* Keep track of every gambler's bet */
    mapping (address => uint) winnings;   /* Keep track of every player's winnings */
    ...

    function makeBet(uint _outcome) payable returns (bool) { … }
    function makeDecision(uint _outcome) oracleOnly() { … }
    function withdraw(uint withdrawAmount) returns (uint remainingBal) { … }
    ...
}
```

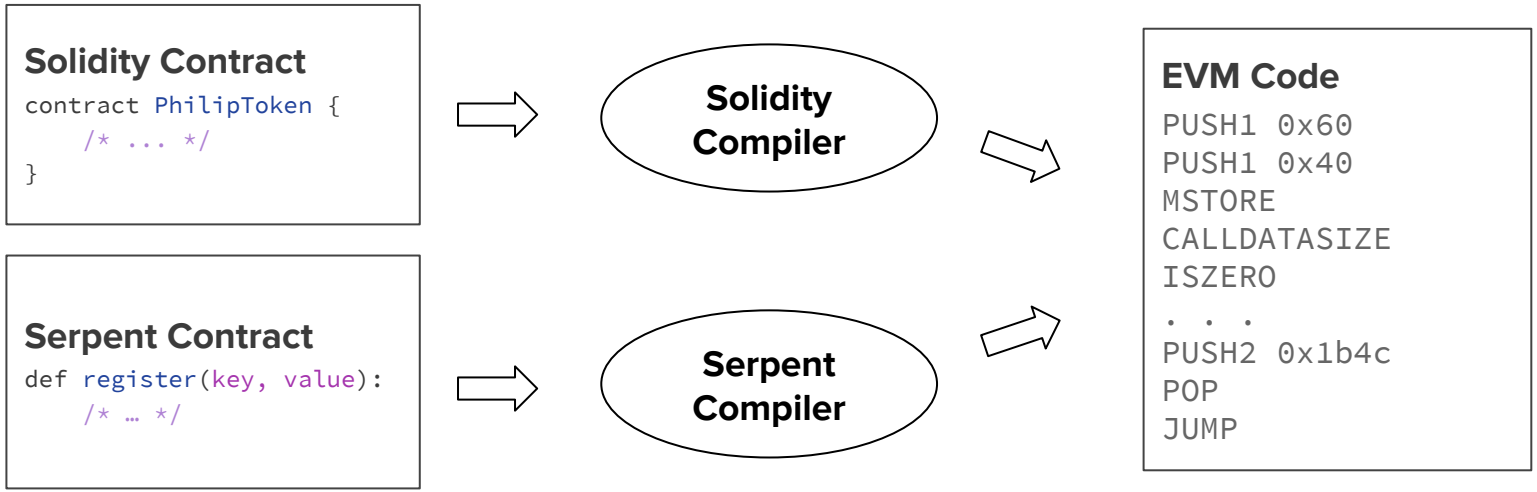


AUTHOR: GLORIA ZHAO & NICK ZOGHB

# 3 ETHEREUM VIRTUAL MACHINE

# ETHEREUM VIRTUAL MACHINE

## COMPILATION AND PROCESS



# ETHEREUM VIRTUAL MACHINE

## HIGH-LEVEL OVERVIEW

- Every Ethereum node runs the EVM as part of its block verification procedure
- Network consensus removes need for Trusted Third Party
  - Violation of contracts requires subverting the entire network
- Secure Peer-to-Peer agreements that live on the blockchain forever
- The **EVM (Ethereum Virtual Machine)** runs contract code
- Contract code that actually gets executed on every node is EVM code
  - Our complex features are made possible by the fact that we can compile contract code into something more simple
  - **EVM code** is a low-level, stack-based bytecode language, kind of like JVM's bytecode

# EVM GAS AND FEES

## HIGH-LEVEL OVERVIEW

**Immediate Issue:**

What if our contract has an infinite loop?

**Every node on the network will get stuck executing the loop forever!**

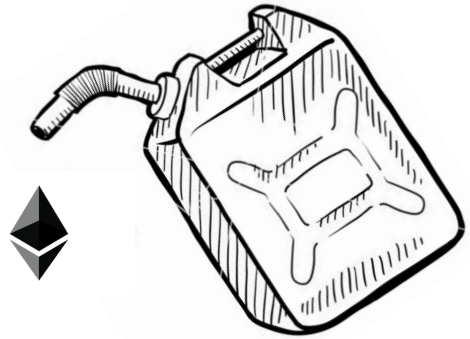
- By the *halting problem*, it is impossible to determine ahead of time whether the contract will ever terminate
- **⇒ Denial of Service Attack!**

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

# EVM GAS AND FEES

## HIGH-LEVEL OVERVIEW

- **Ethereum's Solution:**
  - Every contract requires "gas", which "fuels" contract execution
  - Every EVM op-code requires some gas in order to execute
  - Every transaction specifies
    - the `startgas`, or the maximum quantity of gas it is willing to consume
    - the `gasprice`, or the fee in ether it is willing to pay per unit gas



# EVM GAS AND FEES

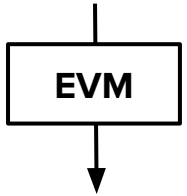
## HIGH-LEVEL OVERVIEW

- At the start of the transaction
  - `startgas * gasprice` ether are subtracted from the sender's account (the one "poking" the contract)
- If the contract **successfully executes**...
  - the remaining gas is refunded to the sender
- If the contract execution **runs out of gas** before it finishes...
  - execution reverts
  - `startgas * gasprice` are not refunded
- Purchasing gas == purchasing distributed, trustless computational power
- An attacker looking to launch a DoS attack will need to supply enough ether to fund the attack

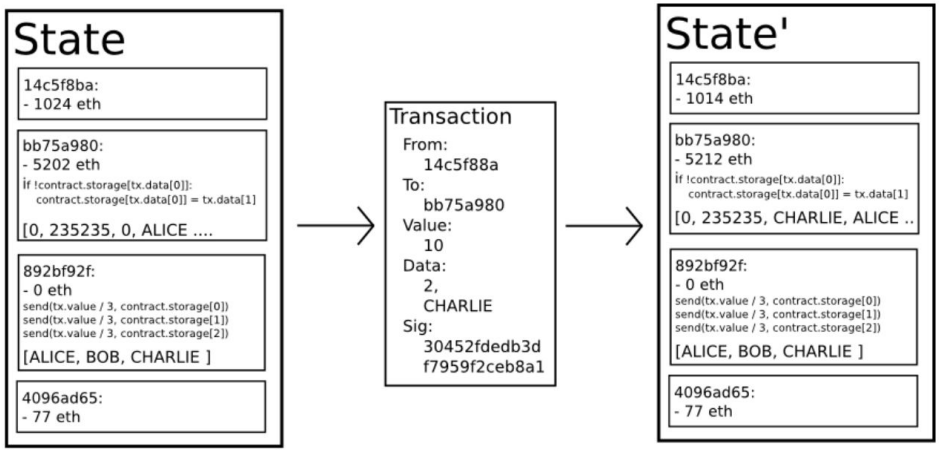
# ETHEREUM NETWORK STATE

## STATE TRANSITION FUNCTION

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




(block_state', gas')




Ethereum Whitepaper

# ETHEREUM CONCLUSIONS

## IT'S NOT FOR EVERYTHING

- **Ethereum is not about optimising efficiency of computation**
- Its parallel processing is **redundantly parallel**
  - **efficient way to reach consensus** on the system state without needing trusted third parties
- Contract executions are redundantly replicated across nodes
  - ⇒ expensive
  - creates an **incentive not to use the blockchain** for computation that can be done off chain

# 4 USE CASES

# 4.1 BASIC USE CASES

AUTHOR: MAX FANG

# BASIC USE CASES

## SMART ASSETS

- A token system is 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

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
```

# 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

Example (from Ethereum white paper):

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

# DOCUMENT OWNERSHIP

## BLOCKCHAIN FUNDAMENTALS

Simple Example: "**Proof-of-Existence**"

- A simple way to prove ownership of a certain document without revealing the data
- Document timestamps allow user to verify ownership at later points of time
- Since a hash of the document is stored on the ethereum blockchain, can verify integrity

Use cases:

- You rent your server space to store documents for people
  - Proof that you still have the document completely unmodified
    - User asks you to hash the document with a certain random number of their choice. If your hash value matches theirs, they are convinced.
    - Integrity guaranteed

# 4.2 ADVANCED USE CASES

# DECENTRALIZED LAND TITLES

## BLOCKCHAIN FUNDAMENTALS

**Problem:**

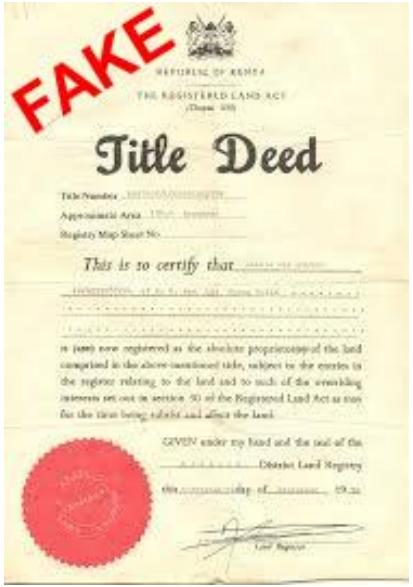
- Flawed paperwork, forged signatures, unclear mortgage documents
  - Hard to trace ownership of land
- Natural disasters, forced evacuations, dictatorships worsen the situation

**Possible Solution:**

Trusted central government authority in charge of keeping the records

**Pitfalls**:

- Corrupt officials may accept bribes and tamper with records
- Government may not have enough resources to support a land record authority
  - Citizens do not have a reason to trust an NGO, multiple NGOs could spring up



# DECENTRALIZED LAND TITLES

## BLOCKCHAIN FUNDAMENTALS

**The Blockchain Solution:**

- Use hashes, digital signatures to store land titles on public blockchain
  - Actual documents can still be stored in a central database
- Transparency of blockchain prevents confusion, fraudulent claims
- Immutability of the blockchain prevents forgery of signatures, corrupt information overwrites
- No need of government offices
  - Third party intervention for conflict is minimal
  - *Simple mechanism to transfer ownership, like making a transaction on bitcoin*

# DECENTRALIZED LAND TITLES

## BLOCKCHAIN FUNDAMENTALS

**Caveat:**

- Need to trust that the information fed into the blockchain is correct
- Tricky to associate virtual identities with land
  - Satellite measurements

**Countries adopting the solution:**
**Georgia, Ukraine, Sweden**

# DECENTRALIZED PREDICTION MARKETS

BLOCKCHAIN FUNDAMENTALS

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 Zuckerberg 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
- Shares resolve to $1 if true, $0 otherwise
- Someone has extra information => arbitrage opportunity

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

GNOSIS

# DECENTRALIZED PREDICTION MARKETS

BLOCKCHAIN FUNDAMENTALS

augur

Use cases

- Set up a security bug bounty
  - "Will my company be hacked, with the hacker revealing their exploit?"
    - 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 product on time.

GNOSIS

AUTHOR: MAX FANG

BLOCKCHAIN AT BERKELEY

# DECENTRALIZED PREDICTION MARKETS

BLOCKCHAIN FUNDAMENTALS

augur

Benefits to being decentralized

- No restrictions on market creation
  - Arbitrary markets allow for much wider range of applicability
  - **Caveat**: Need a defense against unethical/illegal markets
    - Ex Augur: Oracles vote on "unethical" or "undecidable"
- 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

# SUPPLY CHAIN AND PROVENANCE

## BLOCKCHAIN FUNDAMENTALS

Problem: **Conflict diamonds or "Blood diamonds"**

"Did someone die for that diamond?"

- The **Kimberley Process** is a governmental effort requiring participants to certify the origin of their diamond
- Has failed because corrupt officials in diamond producing countries take bribes to sign certifications
- Complex supply chains mask the actual trail of goods

# SUPPLY CHAIN AND PROVENANCE

## BLOCKCHAIN FUNDAMENTALS

**Everledger** uses a blockchain to prove the origin (provenance) of these diamonds.

Global ledger that tracks each diamond on the blockchain

- Record a irrefutable digital footprint of the diamond, puts this on the blockchain
  - Tokenizes the diamond
- As diamond moves along the supply chain, transfer the asset on the blockchain
  - Maintains a complete history of the diamond
- Ledger history is immutable, duplicated, and cryptographically secure
  - Records can't be altered by corrupt intermediaries
  - Open and transparent; well-suited for auditing and analysis
- At final destination, check that physical diamond matches its blockchain token

# SUPPLY CHAIN AND PROVENANCE

## BLOCKCHAIN FUNDAMENTALS

Everledger is a decent use case of blockchain for supply chain.

- Not true for supply chain ideas in general.

Main Drawbacks to Supply Chain applications:

- How to prove that the token on the blockchain corresponds with the real, physical good?
  - Counterfeit goods might be swapped in place of the real good
  - Possible solutions:
    - Tamper-evident and hard to forge identifiers traveling along with product
      - Graft a dollar bill onto your product?
      - Everledger records the optical fingerprint of each diamond. E.g. GemPrint
    - Cross-validation of information.
      - i.e. IoT enabled sensors verifying info from nearby sensors
- **Caveat:** Forgery is possible if the data entered into the blockchain is not secure
  - Everledger still needs to trust the entities authenticating diamonds onto the chain

# 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

**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

# 5 GENERALIZATIONS

## DOES IT MAKE SENSE TO USE A BLOCKCHAIN?



# WHEN SHOULD I USE A BLOCKCHAIN?

## 1. SHARED DATABASE

Do you want to use a database in the first place?

## 2. MULTIPLE WRITERS

Are multiple entities generating and modifying data?

## 3. ABSENCE OF TRUST

Is there some degree of mistrust between entities?

## 4. DISINTERMEDIATION

Is having a trusted central party risky or wrong?

## 5. TRANSACTION INTERACTION

Are the transactions created by different writers dependent on each other?

# WHEN SHOULD I USE A BLOCKCHAIN?

~~1. SHARED DATABASE~~

REGULAR FILE STORAGE

~~2. MULTIPLE WRITERS~~

CENTRALIZED DATABASE

~~3. ABSENCE OF TRUST~~

MULTIPLE COPIES OF A CENTRAL DATABASE

~~4. DISINTERMEDIATION~~

TRUSTED THIRD PARTY

~~5. TRANSACTION INTERACTION~~

MASTER-SLAVE DATABASE REPLICA

# LIMITATIONS OF BLOCKCHAIN TECH

## BLOCKCHAIN FUNDAMENTALS

**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
- Potential Solution: **TEEs like SGX, ARM**
  - Drawback: Trust Intel

# LIMITATIONS OF BLOCKCHAIN TECH

## BLOCKCHAIN FUNDAMENTALS

**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
- Compromise: Implement a decentralized reputation system

**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**

# ESSENTIAL PROPERTIES OF BLOCKCHAIN KILLER APPS

**Trustless environments that need consensus or coordination** (ex. Land Titles)

**Disintermediation, censorship-resistance**

**Data is "correct by design"** (ex. Everledger)
- **Traceability and Authenticity**
- But the data entering blockchain must also be correct

**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. Lightning)

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

**New ways of creating incentives** (ex. Augur)

**Robust management and security**
- But often breaks down at the interface with the real world

**New governance models** (ex. DAOs, futarchy)

Contrast with centralization:
**Deep integration**, **cohesive user experience**
- **Efficiency** - blockchains are slow in general
- **Centralized control** over data and read/write permissions
- Manages **complexity** well

AUTHOR: MAX FANG

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

# CONCLUSION

## BLOCKCHAIN FUNDAMENTALS

Blockchain technology (and distributed tech) is going to lead to a new world of decentralization

Remember: Why is blockchain better than using a centralized database?





AUTHOR: MAX FANG

4.3

# BAD USE CASES

AUTHOR: MAX FANG

# AIRLINE INDUSTRY

## BLOCKCHAIN FUNDAMENTALS

Problem:

- Need quicker way to get around flight cancellations
  - Can’t get on a flight by another airline quickly
  - Huge lines, phone calls to customer support inefficient
- Each airline only has access to data of its own flights
  - Example: space on a flight, flight status, take off gate etc.

Possible Solution 1: Common Central Database for all the airline operators in a country

- Malicious attacker can stall all air travel in a country
  - Building a system that is secure against this potential attack is complex
    - Needs to be more secure than any existing database

# AIRLINE INDUSTRY

## BLOCKCHAIN FUNDAMENTALS

- The Blockchain solution:
  - A private blockchain shared with the different airline operators
    - Share information only with other airlines
  - Transparency of the blockchain allows for access of data
    - Can get booked on a flight by a different airline operator in time
  - Efficiency gains compared to today's system
    - A few seconds confirmation time vs an hour on the phone/line
    - Allows customer to reach final destination quicker
  - Malicious attacker needs to take down multiple servers to stall air travel

# REFERENCES (INCOMPLETE)

## BLOCKCHAIN FUNDAMENTALS


- 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

# DECENTRALIZED SHARING ECONOMY

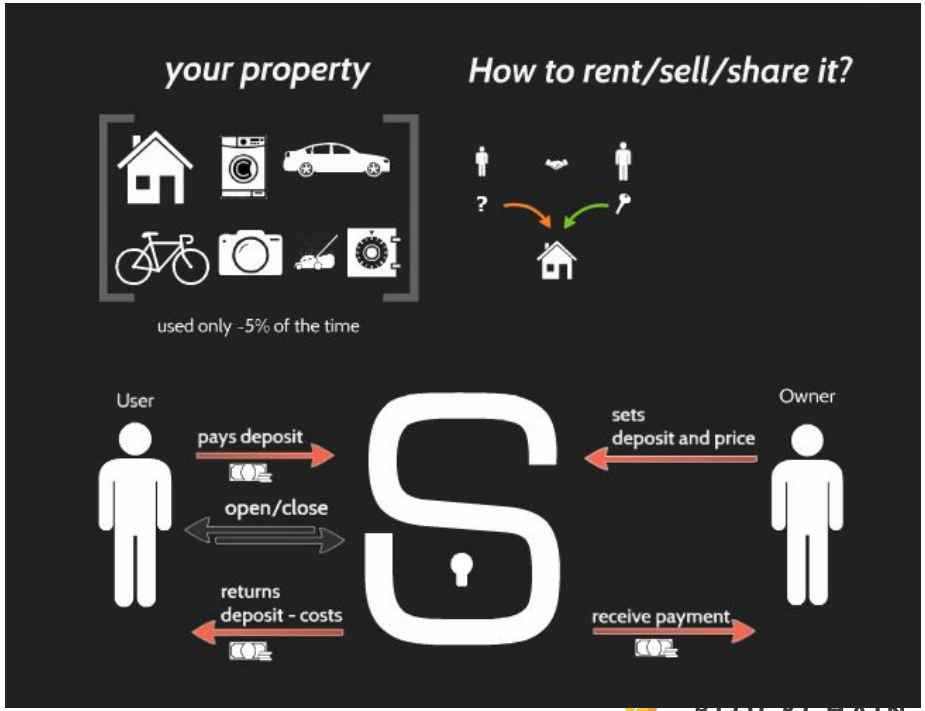
## BLOCKCHAIN FUNDAMENTALS

**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




AUTHOR: MAX FANG

# DECENTRALIZED SHARING ECONOMY

## BLOCKCHAIN FUNDAMENTALS

Benefits of being decentralized (Slock.it):

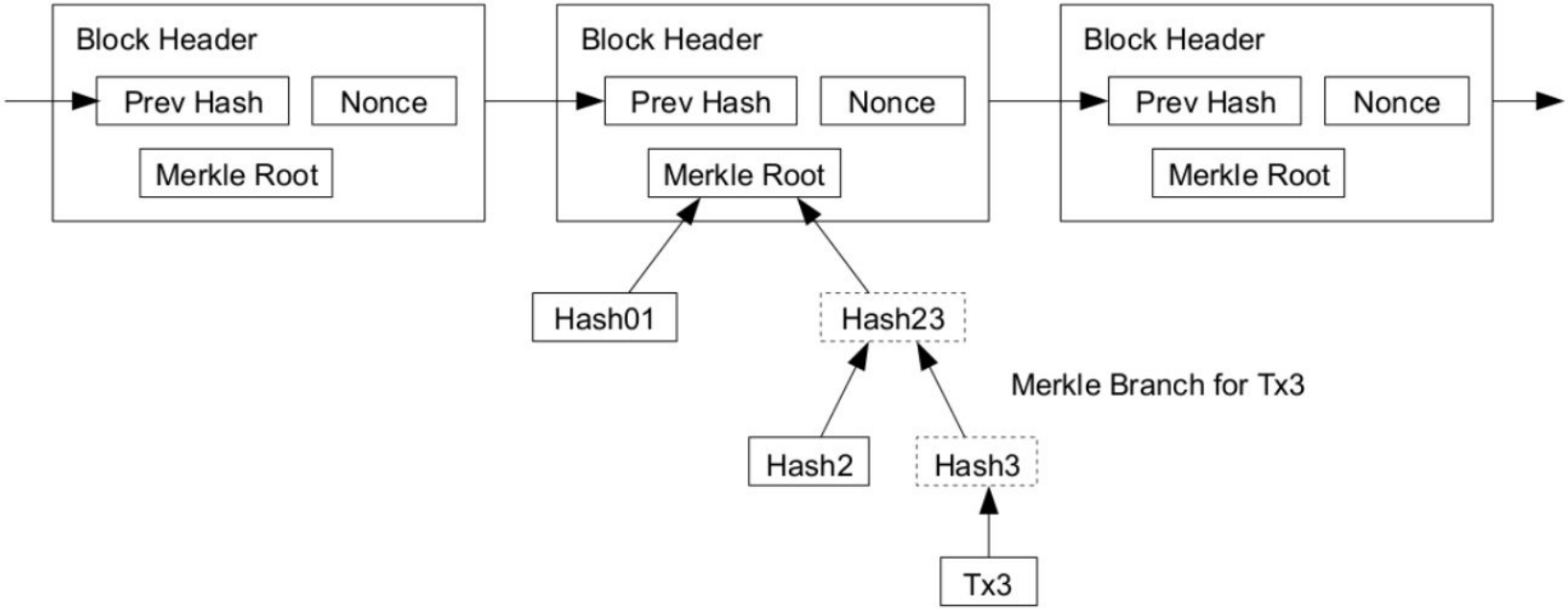
- ...Trustless?
  - Decentralized reputation is very hard to implement
  - Centralized solutions probably better
- Programmable money
  - Centralized solutions still programmable
  - Public blockchains (Bitcoin/Ethereum etc) have easier technological integration
  - No personal information needed to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

5

# EXTRA SLIDES

# BLOCKCHAIN STRUCTURE

## BITCOIN



# BLOCKCHAIN STRUCTURE

## ETHEREUM