

Condensed Course: “Coding for Lawyers”

Session 2: Smart Contracts and Solidity Basics

Prof. Wulf A. Kaal

Tilburg University
Masters Program in Business Law

May 3, 2017

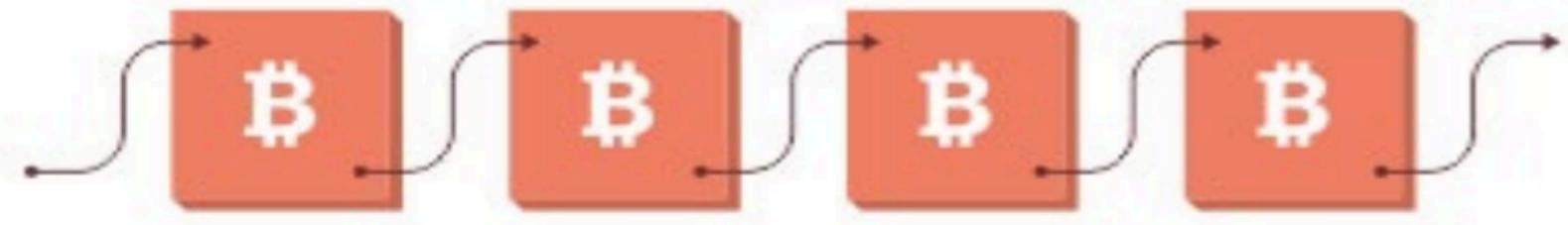
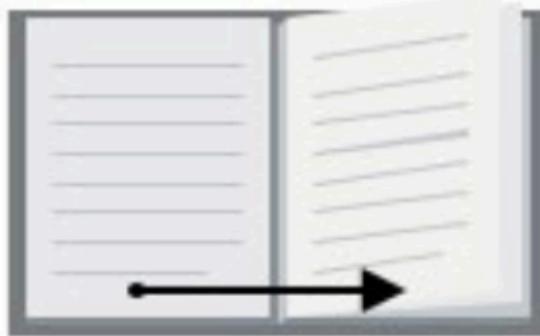
Outline

- A. BLOCKCHAIN
- B. ETHEREUM VIRTUAL MACHINE
- C. SMART CONTRACTS
- D. EXAMPLES OF SMART CONTRACTS IN LAW
 - I. DAO
 - II. HEALTH DATA
 - III. SMART PROPERTY
 - IV. BOND TRADING
- E. SOLIDITY
- F. SOLIDITY CODING EXERCISES:
 - I. BASIC CONTRACT IN SOLIDITY
 - II. STOCK ISSUANCE AND TRANSFER

Blockchain Technology for Smart Contracting

WHAT IS BLOCKCHAIN?

- A data structure
- Wait a minute, what is data structure?
- Arrangement of data in computer memory
- Any real life example?
 - PDF Docs
 - Excel sheets
 - MySQL Table



Blocks in a chain refer to previous blocks, like page numbers in a book.

- Imagine it as a book where each page reference to the previous page through a page number one less than a current page number.
- Easy to detect if a page has been removed
- Easy to arrange the pages & identify suspicious activity. That's why page numbers are important in agreements.
- Wait! What if the content of the page has been altered?
- Then in that case, lets produce the page number by crunching the content of the previous page.
- Ah! Now it makes sense. If anyone changes the content of the page then the next page number will not match if we try to generate it using altered page content.

BOOK ORDERING	BLOCK ORDERING
Page 1, 2, 3, 4, 5	Block n58uf0 built on 84n855, Block 90fk5n built on n58uf0, Block 8n6d7j built on 90fk5n.
Implicit that the page builds on the page whose number is one less. eg Page 5 builds on page 4 (5 minus 1).	84n855, n58uf0, 90fk5n, 8n6d7j represent fingerprints or hashes of the blocks.

- So in real Blockchain case, each block is built on top of the recent block and use its previous block's content as a signature.
- Building a block & adding it in the Blockchain is the task of the miner nodes.
- In public Blockchain it is made computationally difficult to add a block to prevent attacks.
- Miners try to guess a number in such a way that if it gets crunched with the most recent block's fingerprint than it will create a new fingerprint which will be less than the last/most recent block in the Blockchain.
- It takes time & computational power to add a Block in the Blockchain. Hence there is some reward (25 BTC in case of Bitcoin Blockchain)
- Private Blockchain can choose other methods to add a block as they can trust the miners using a contract etc.

PEEK INSIDE A SAMPLE BLOCK



Blockchain Inherently Solves:

- Paper waste. Payers alone \$375B in 2015
 - Information is stored in mixed formats, in different places
 - Data in transit and at rest is unencrypted; 35-40% of all healthcare data per HIMSS survey
 - Consumers do not own their information assets
-
- BMC report “Billing and insurance-related administrative costs in United States” goo.gl/sAVOco

“Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.”

- Kim Cameron, 7 Laws of Digital Identity

Overcoming Homeostasis

EMERGING CORE TECH		
TYPE	OLD	NEW
Data	Local Storage	Cloud/Blockchain
Assets	Physical	Dynamic Smart Contracts
Provenance	Physical Proof	Algorithmic Proof
Data Models	EDI, XML	JSON, Graph
Connectivity	Point to Point, VPN	API, SDK, Cloud
Processing	Mostly Manual	Async, real-time compute

Blockchain Main Themes

- Longitudinal immutability
- Reliable timestamps
- Cryptographic security
- Fully distributed replication
- Publicly verifiable code governing all transactions
- Value in Smart Contract Assets

SUMMARIZING BLOCKCHAIN

- It is a decentralized distributed ledger (data structure) where data is being stored inside blocks in form of transactions.
- Remove the dependency on the trusted third party for recording the data in Blocks.
- In public Blockchain More complex algorithm require to avoid the malicious activities.
- Since each block is built on top of previous Block immutability has been achieved.
- Very difficult to fake a block & very very easy to detect the fake Block.
- This all exist in memory of the computers.
- Every participants of the Blockchain contain the almost same copy of the Blockchain.

A large iceberg is shown floating in the ocean. Only a small portion of the iceberg is visible above the water's surface, while the majority of it remains submerged below. This visual metaphor represents how many blockchain applications, such as SmartContracts or IoT, are built on top of a foundation of existing technologies.

Bitcoin

Governance

SmartProperty

Voting

SmartContracts

Name Registration

Identity

Crowdfunding

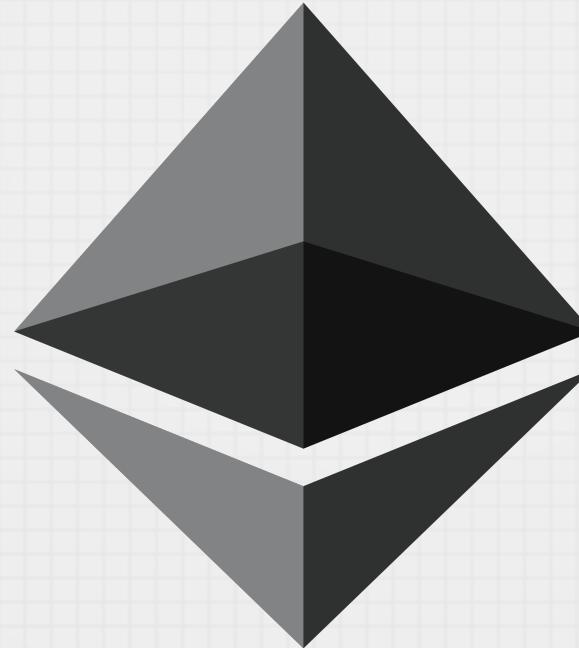
IoT

Climate

Ethereum

WHAT IS ETHEREUM

- An Open Source Blockchain-based distributed computing application platform
- Mainly used to building & implementing smart contracts functionality
- It offers Decentralized Virtual Machine aka Ethereum Virtual Machine
- Initiated by Vitalik Buterin in Late 2013
- Ethereum's live blockchain was launched on 30 July 2015



Ethereum

The vision of Ethereum is an unstoppable censorship-resistant self-sustaining decentralised world computer

ETH	Ethereum's inbuilt native cryptocurrency, used for paying for smart contracts to run
Ethereum Virtual Machine, Swarm, and Whisper	Decentralised computation, file storage, and communication protocols
Solidity, Serpent, and LLL	Smart contract programming languages
geth, eth, pyethapp	The main Ethereum software, written in different languages
Frontier, Homestead, Metropolis, Serenity	Friendly names for different software releases

WHAT IS ETHEREUM VIRTUAL MACHINE

- Ethereum is a programmable Blockchain
- It allows user to create their own operations
- Ethereum in the narrow sense refers to a suite of protocols that define a platform for decentralized applications.
- At the heart of it is the Ethereum Virtual Machine (“EVM”), which can execute code of arbitrary algorithmic complexity.
- In computer science terms, Ethereum is “Turing complete”.
- Developers can create applications that run on the EVM using friendly programming languages modelled on existing languages like JavaScript and Python.

What is Ethereum?

- ❖ “A planetary scale computer built on blockchain technology”
- ❖ Open for all to use; zero infrastructure
- ❖ Turing-Complete: Build arbitrarily complex applications
- ❖ “Ethereum is a decentralized platform that runs **smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference.”
- ❖ Ethereum Virtual Machine, runs on **Ether**, Pay per computational step

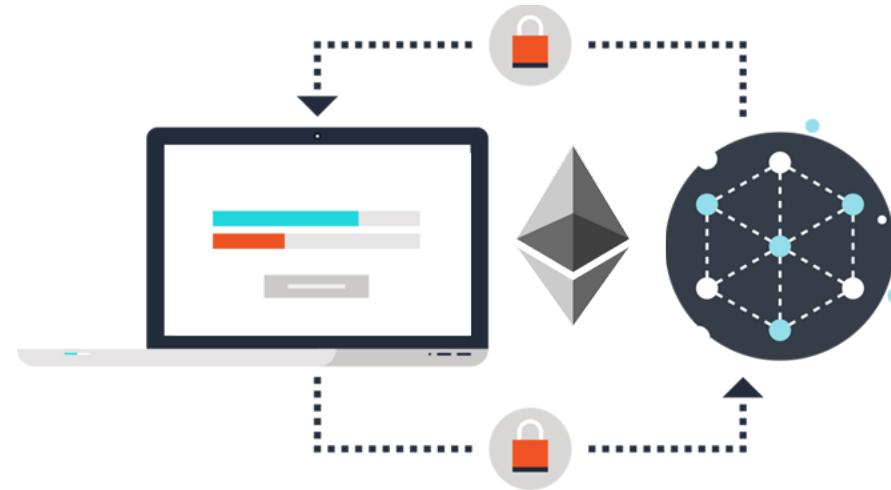


Ethereum Virtual Machine (EVM)

- ❖ A Turing Complete Virtual Machine accessible from anywhere in the world has a number of benefits for the automated-economy:
- ❖ The Ethereum Blockchain is:
 - ❖ A blockchain with a built-in programming language
 - ❖ A decentralized, massively replicated database where the current state of all accounts is stored
 - ❖ A consensus-based globally executed virtual machine

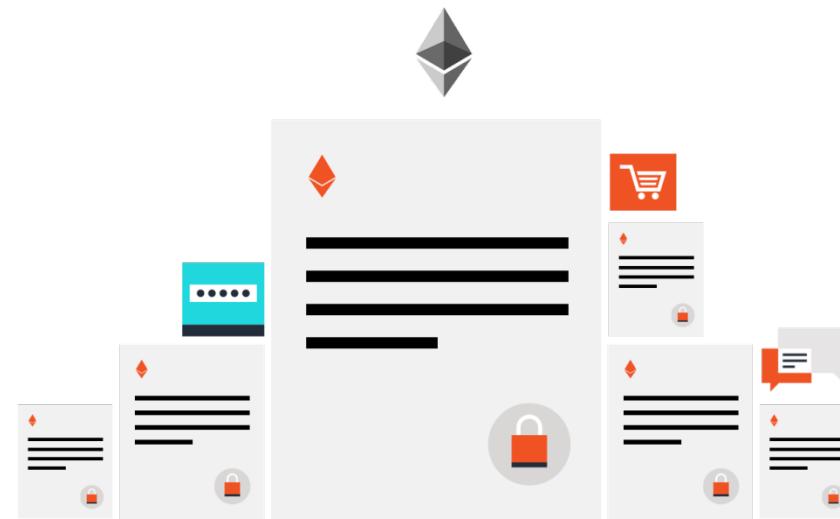
Source: Melanie Swan; <https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial>

WHAT IS ETHEREUM?



Ethereum is a platform that makes it possible for any developer to build and publish next-generation distributed applications.

CONTRACT-BASED



Code-based contracts are the main building blocks of Ethereum.

ETHEREUM IS VERSATILE



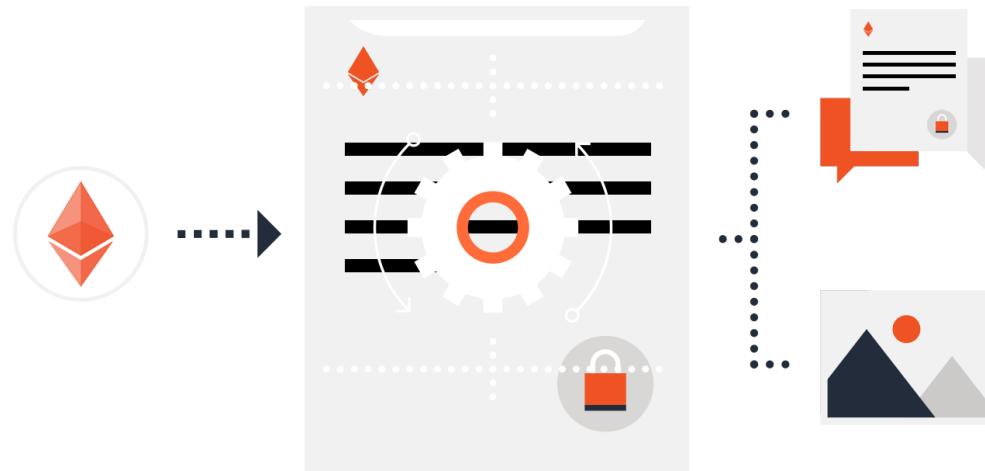
Contracts can be used to build currencies, financial derivatives, voting systems, decentralized organizations, data feeds, title registries and thousands of other applications.

ETHER IS THE FUEL



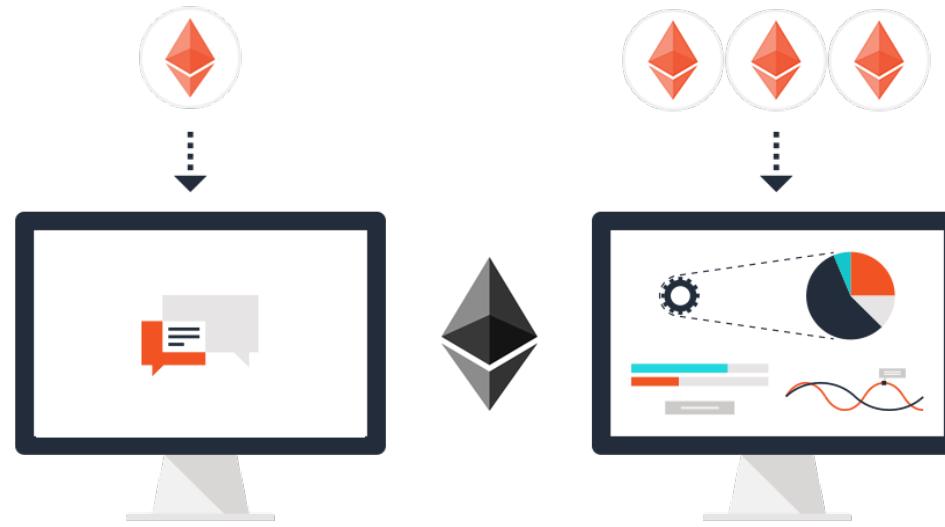
Ether is Ethereum's cryptofuel. It is a type of digital token that powers the applications on the decentralized network.

ETHER IS THE FUEL



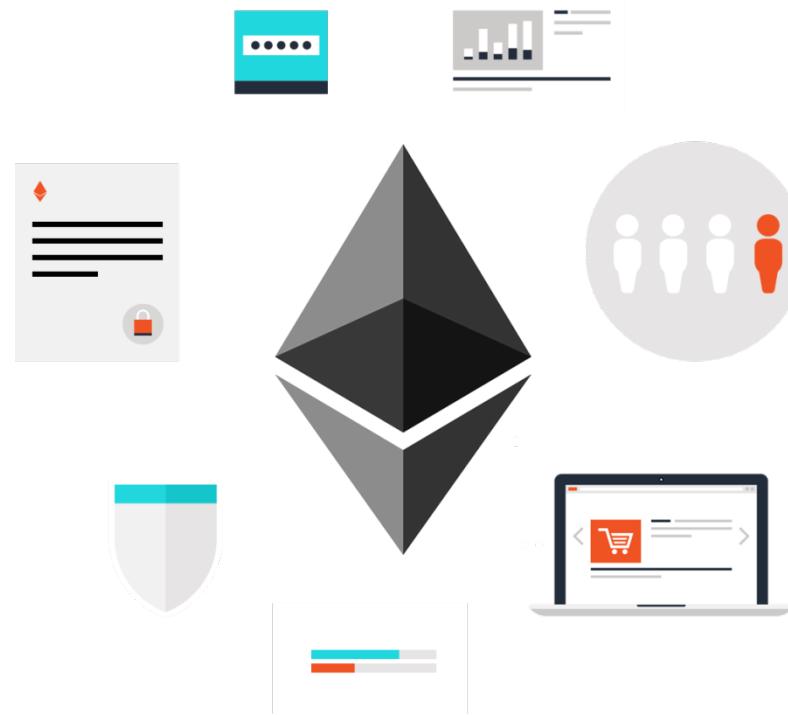
Sending a transaction to a contract causes its code to execute. Contracts can store data, send transactions and interact with other contracts.

NOT ALL CONTRACTS ARE ALIKE



The more computation a transaction requires the more ether it consumes.

BUILD (ALMOST) ANYTHING

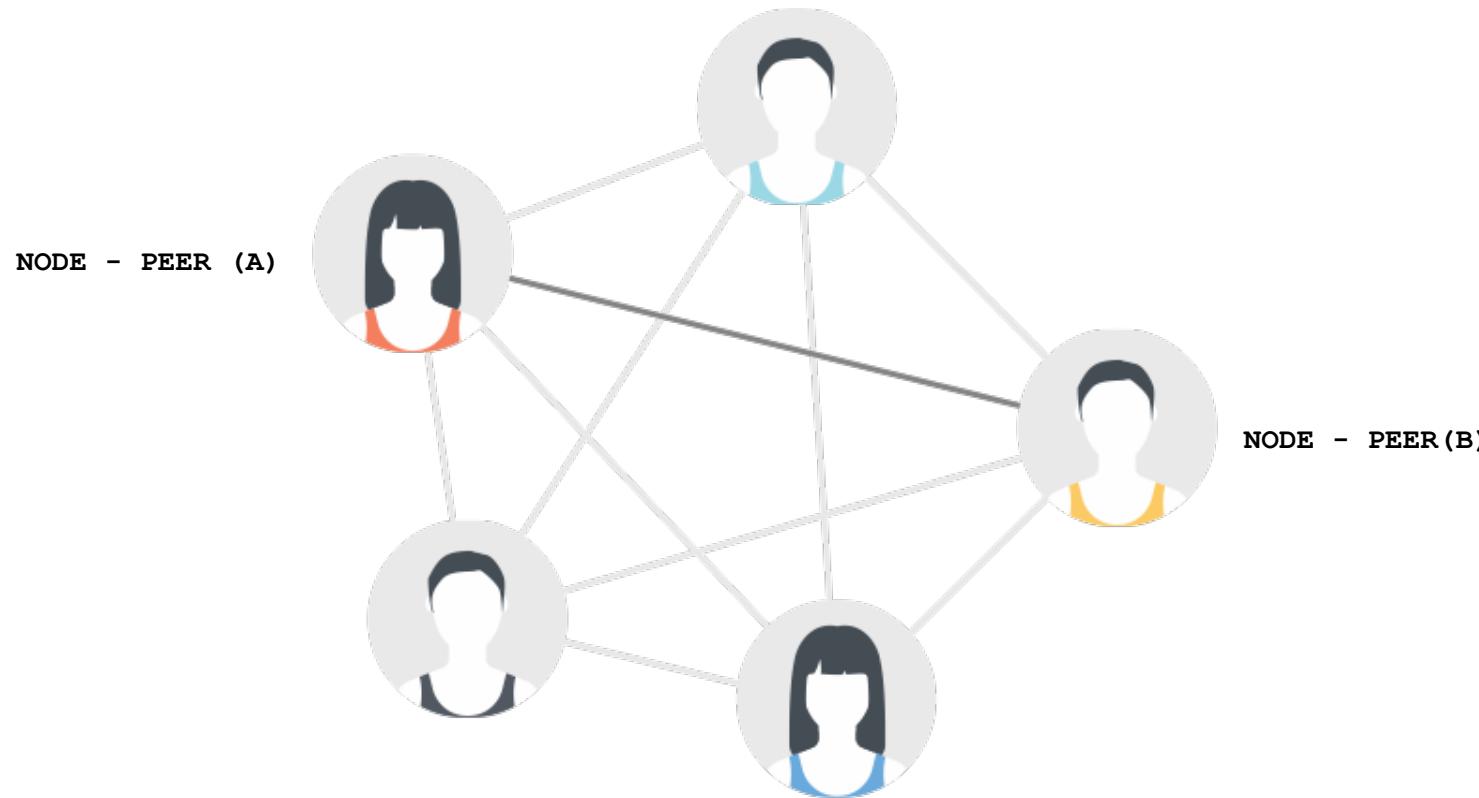


Ethereum can be used to codify, decentralize, secure and trade almost anything.

Ethereum Mission Statement

TO RESEARCH, DESIGN AND BUILD
SOFTWARE THAT, AS BEST AS
POSSIBLE, FACILITATES, IN A
SECURE, DECENTRALISED AND FAIR
MANNER, THE **COMMUNICATION** AND
AUTOMATICALLY-ENFORCED
AGREEMENT BETWEEN PARTIES.

DECENTRALIZED SYSTEMS



P2P file-sharing network

OPEN-SOURCE TRUST



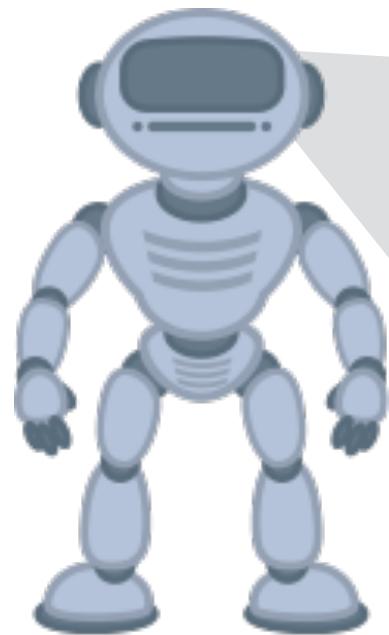
People can make agreements....

OPEN-SOURCE TRUST



...and not live up to them.

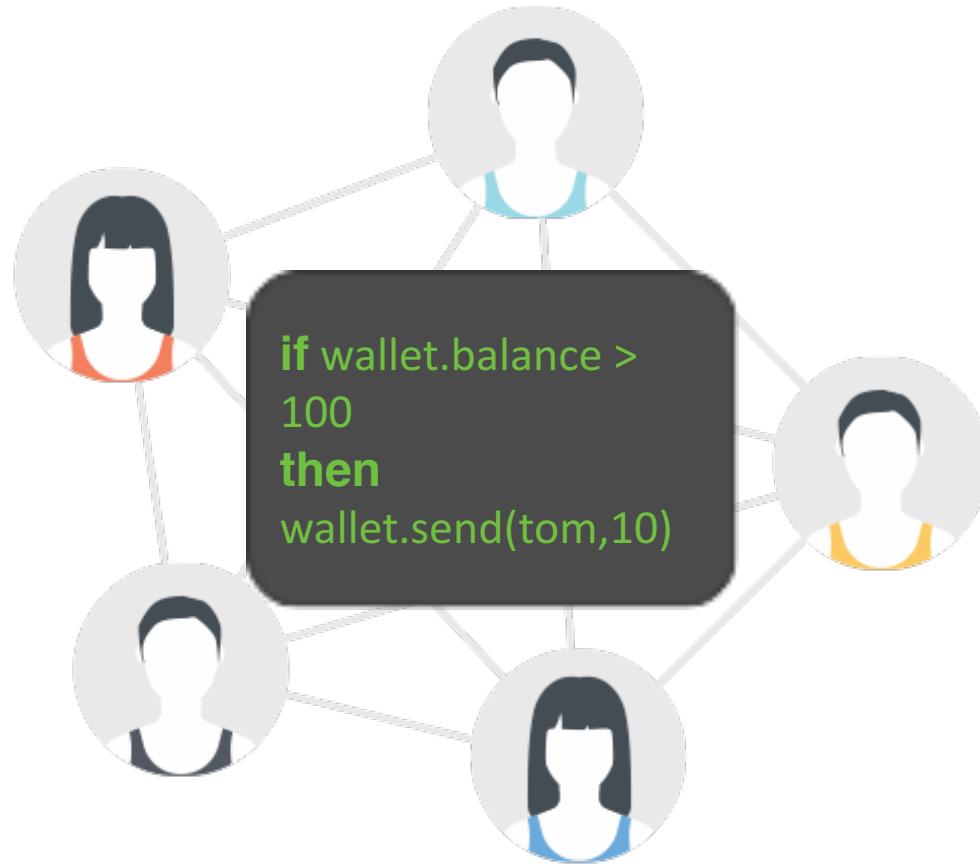
OPEN-SOURCE TRUST



```
if wallet.balance > 100  
then wallet.send(tom,10)
```

Software does what it is coded to do.

OPEN-SOURCE TRUST



Open Source software can be seen and audited by anyone.



1 Introduction

Over the last few years, developers have begun using Bitcoin's underlying technology - the Blockchain - for creative new applications. Ethereum is a next-generation platform that allows anyone - both developers and consumers - to easily take advantage of decentralized networks and realize the benefits of blockchain technology.

2 What are Decentralized Networks?

Decentralized networks redistribute functions and powers away from a central server, enabling peer-to-peer communication.



BitTorrent, used for file sharing, is an example of a decentralized network.

Advantages:

- ✓ No central point of failure
- ✓ Highly reliable
- ✓ Cost-effective

3 The Blockchain

Most networks function using a central authority to make final decisions. The blockchain, a type of decentralized network, is able to make agreements across the whole network, without any central authority.



Bitcoin uses Blockchain technology to record and verify transactions without the need for a central bank.

4 Mist

Mist will be Ethereum's end user interface to bring blockchain technologies to non-technical users.

It will include a catalog for decentralized applications and an assortment of other tools.



Mist will work similar to app stores and browsers that consumers are already familiar with.

ENTER ETHEREUM

Ethereum's vision is to decentralize the internet by creating a platform where applications can be built and run on a decentralized network. Ethereum is fast and flexible without the inherent limitations of the Bitcoin protocol.

“What Bitcoin does for payments, Ethereum does for anything that can be programmed”



5 Ether

Ether is the native token of Ethereum, and serves two key purposes. First, by requiring applications to pay ether for every operation they perform, broken or malicious programs are kept from running out of control. Second, ether is given as a reward to those who contribute their resources to the decentralized network.



Ether: The “fuel” that runs the Ethereum network

7 What will Ethereum be used for?

Decentralizing Existing Services



Services that are traditionally centralized can be decentralized using Ethereum. This will lead to reduced costs and fees by connecting individuals directly and removing 3rd parties.

Imagine a service like Uber or eBay without a company in the middle collecting fees!

Bringing Science Fiction to Life

Using Ethereum, IBM and Samsung worked on a proof of concept where a washing machine could:

- order its own detergent when it runs out
- call its own repairman when it breaks down
- do the laundry when electricity is cheapest!



Unimagined Possibilities



The creators of the internet didn't anticipate social media or cloud computing. We have no way of predicting which breakthrough technologies will be born on the Ethereum blockchain!

8 What is being built on Ethereum?



Decentralized crowdfunding platform.



Access protocol for smart property and the Internet of Things.



Project to increase the transparency and accountability of supply chains.



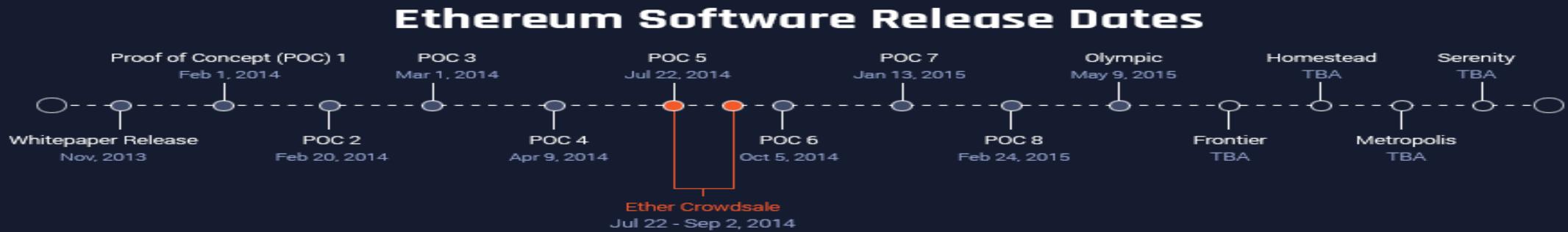
Decentralized prediction market platform.

9 Funding the Vision

On July 22, 2014, the non-profit Ethereum foundation launched a public crowdsale of Ether. The funds collected have helped carry out the development of the project. The sale lasted for 42 days and raised 31,591 BTC, or \$18,439,086, making it (at the time) the largest completed crowdfunded project of all time.

Crowdsale Numbers

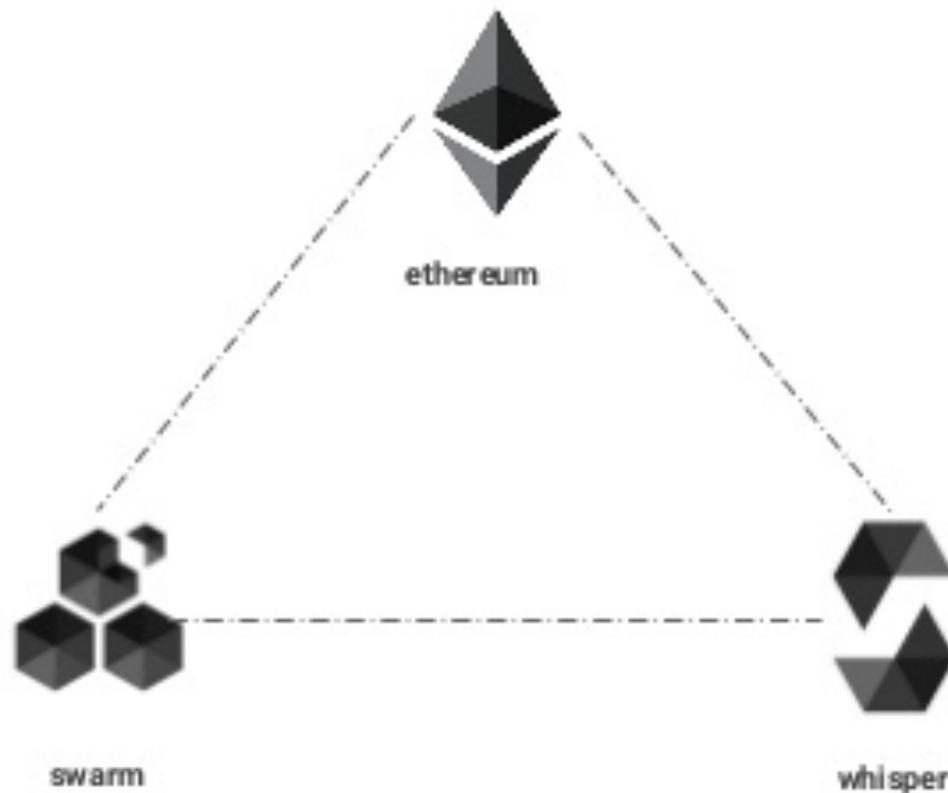
42 Days	31 Thousand BTC Collected	\$18 Million Equivalent
3 rd Largest Crowdfunded Project in History (current)	9 Thousand Participants	



	Hyperledger	Ethereum	Ripple	Bitcoin
Description	General purpose Blockchain	General purpose Blockchain	Payments Blockchain	Payments Blockchain
Governance	Linux Foundation	Ethereum Developers	Ripple Labs	Bitcoin Developers
Currency	None	Ether	XRP	BTC
Mining Reward	N/A	Yes	No	Yes
State	Key-value database	Account data	None	Transaction data
Consensus Network	Pluggable : PBFT	Mining	Ripple Protocol	Mining
Network	Private or Public	Public or Private	Public	Public
Privacy	Open to Private	Open	Open	Open
Smart Contracts	Multiple programming languages	'Solidity' programming language	None	Possible, but not obvious

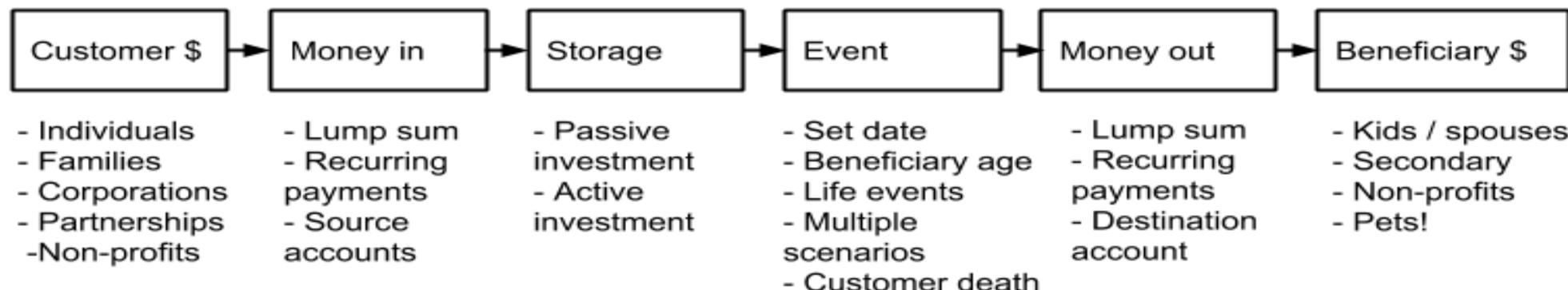
Messaging and File Sharing...

- In addition to the use of the ethereum virtual machine to execute contract logic. The ethereum project also introduced two additional protocols to provide peer to peer support for exchanging message as well as exchanging static files
- The peer to peer protocol used for exchanging message is named whisper and it provides a powerful distributed and private messaging capabilities with support for single cast, multicast and broadcast messages
- The peer to peer protocol used for exchanging static files is named swarm and it provides a new incentivized approach to distribute static content among peers and exchange them efficiently



*Simplifications are made to explain the concepts behind ethereum. For detailed technical specifications, please refer to the ethereum website.

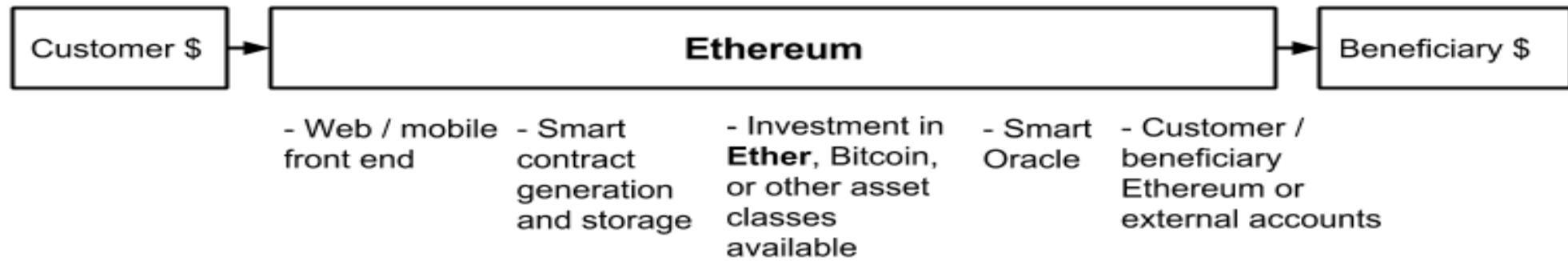
Generic Investment Plan Structure



Traditional Providers



Etherplan Model



Element 5: Virtual Machine & Prog Lang



The Bitcoin virtual machine enables narrowly programmable money.

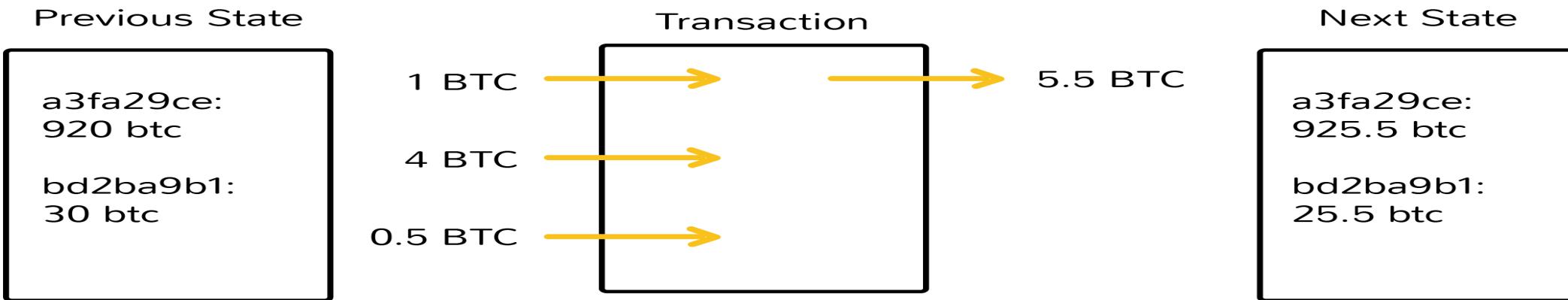
- It is like a pocket calculator at each node of the network.
- Data is decentralized; program operating on that data are not.

The Ethereum virtual machine and powerful high-level programming language enables fully decentralized applications.

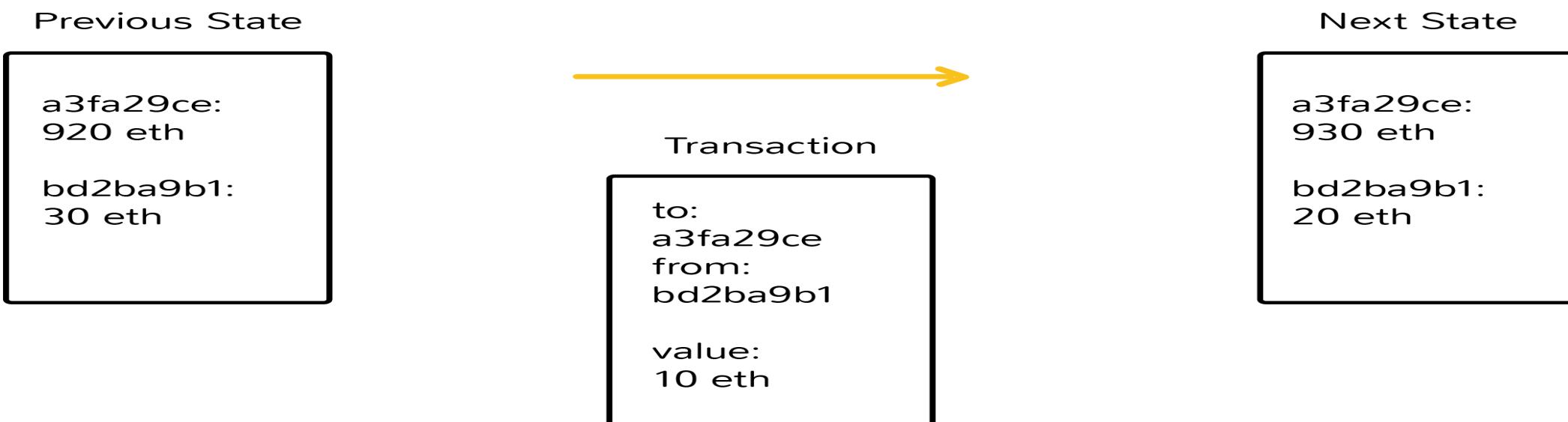
- It is like a general purpose computer at each node of the network.
- Data and their programs are decentralized.



Bitcoin



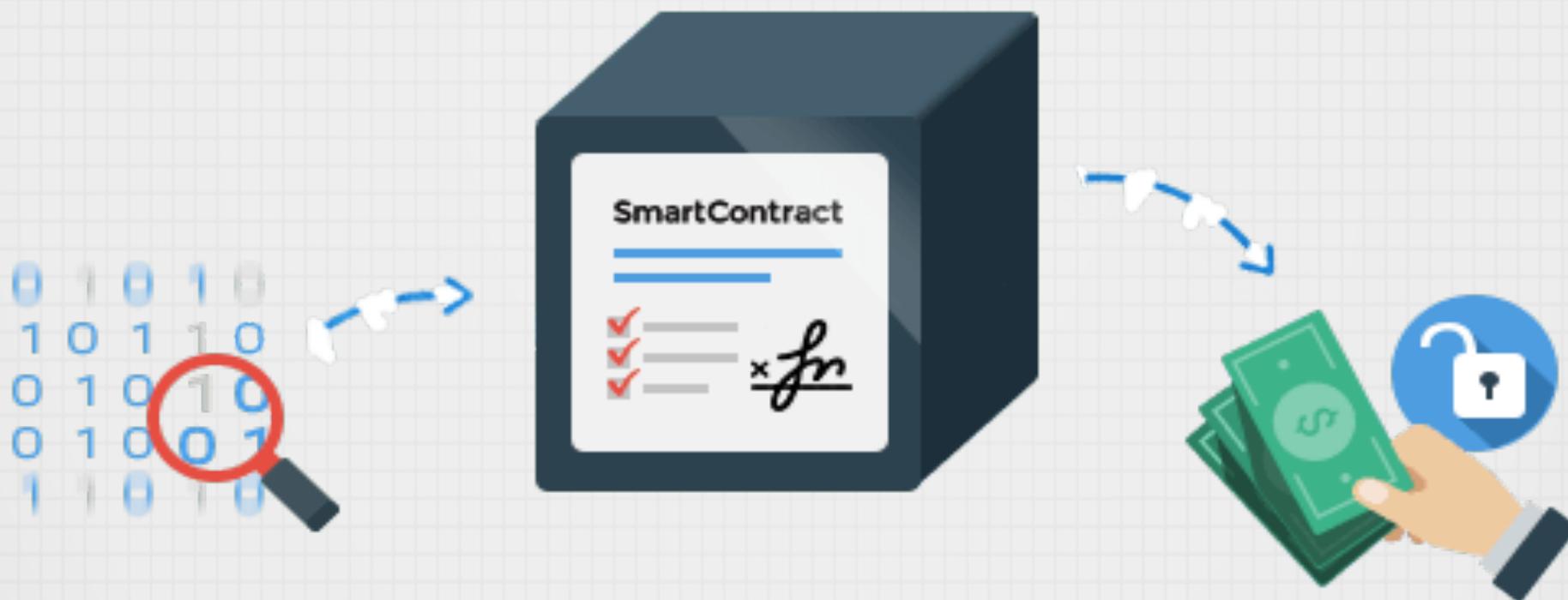
Ethereum



Smart Contracts

What is a Smart Contract

- Smart contract is a term used to describe computer program code that is capable of facilitating, executing, and enforcing the negotiation or performance of an agreement (i.e. contract) using blockchain technology.
- The entire process is automated can act as a complement, or substitute, for legal contracts, where the terms of the smart contract are recorded in a computer language as a set of instructions.



“Controls allow a quarrelsome species ill-suited to organizations larger than tribes to work together on vast projects like manufacturing jumbo jets and running hospitals.”

- *Nick Szabo on Smart Contracts, 1997*

A Short History

- Nick Szabo formalizes concept for Smart Contracts in 1994
- Szabo releases formal paper on Smart Contracts in 1997
- “*Bitcoin: A Peer-to-Peer Electronic Cash System*,” a paper written by Satoshi Nakamoto released in 2008
- Source code released in 2009
- Bitcoin network began in 2009 when Satoshi Nakamoto “mined” the first Bitcoins
- Satoshi Nakamoto disappeared from the public – that is, from Bitcoin forums, papers, and code contributions – in April 2011.
- NASDAQ announces private security exchange on blockchain (LINQ) Dec. 2015
- Pokitdok demos production ready ASC 5010 X12 at Health 2.0 Sept 2016

Smart Contracts

A digitally signed computable agreement between two or more parties that relies on a **consensus system**. A virtual third party – a **software agent** – can execute and enforce at least some of the terms of such agreements.

Smart Contracts

- A snippet of code that is **programmed** to execute whenever certain parties enter their **keys**, thereby agreeing to a **contract**.
- The same **code** reads anything that can be **parsed** by a computer
- To **dynamically** create contracts that are **automatically** filed when certain conditions are met.

Smart Contract Considerations

- Expressing business logic as computer programs that self amend
- **Representing the events which trigger that logic as messages to the respective programs**
- Using digital signatures to prove who sent/received the messages
- Connecting and placing the programs, messages and signatures on a blockchain

Smart Contract Considerations

- Permission system can be used to restrict the right to create assets
- Users must consider per asset permissions in which each type of asset has its own set of administrator agents.
- Allows interoperability by design
- Users must separate *policy* from mechanism design

Smart Contract Considerations

- The asset(s) function via rulesets based on context
- Rule Sets can be greatly simplified if based on behaviors and expected outcomes
- Rulesets are executed via higher order *agents*
- Both the rulesets and the asset(s) are the Asset Under Consideration (AUC)

Smart Contracts

- ***Algorithm 1.0:*** Parties C and D create a 2-of-2 multi-sig output based on asset:
 - 1: D sends a list I_C of inputs with Assets_(C) to C
 - 2: C selects its own inputs with I_C with Assets_(C)
 - 3: C creates transaction sets $t(s)$ $\{I_C, I_D\}$
 - 4: Create time locked transaction sets T_r
 - 5: Sign said transactions $T_r \rightarrow \{C, D\}$
 - 6: Updates are then sent to Network

Traditional contracts



1-3 Days



Manual remittance



**Escrow
necessary**



Expensive



**Physical presence
(wet signature)**



**Lawyers
necessary**

Smart contracts



Minutes



**Automatic
remittance**



**Escrow may not be
necessary**



**Fraction of
the cost**

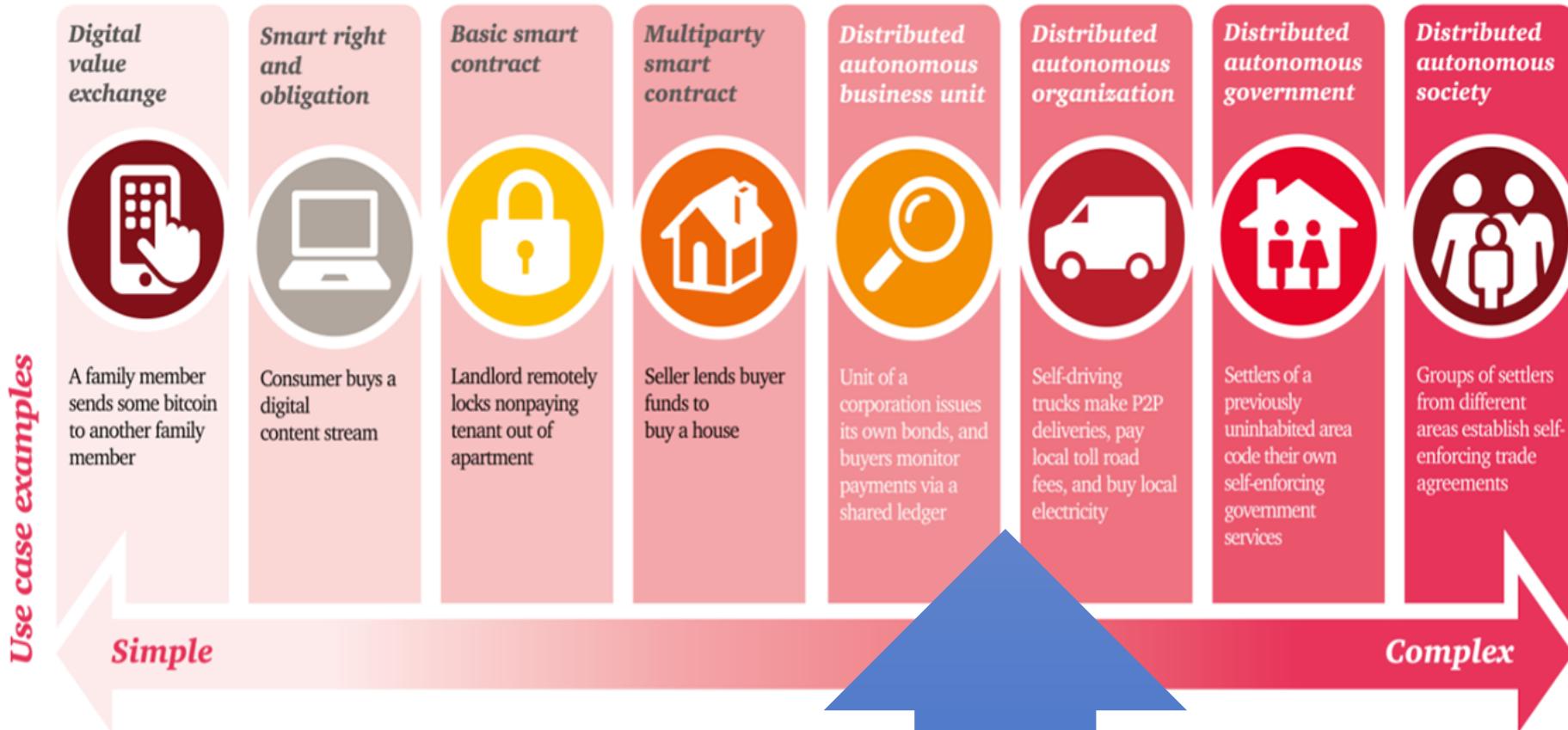


**Virtual presence
(digital signature)**



**Lawyers may not be
necessary**

Smart contracts – simple to complex



1



Pre-defined Contract

2



Trigger Events

3



Execution & Value Trans.

4



Transaction Settlement

**On-chain
(digital)**

**Off-chain
(physical)**

Terms est. by
counterparties:

- Variables interest rate (e.g., LIBOR)
- Settlement Currency
- Exchange Rate

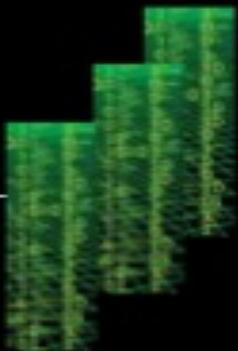
Event triggers
contract
execution:

- Transaction initiation
- Information retrieval

Terms of
contract
specify
movement of
value based
on conditions
met.

For on-chain assets (e.g.,
digital currency), accounts
are settled atomically

For off-chain assets (e.g.
stocks), ledger changes
will be 'matched'



Smart Contract-DOA/DAC Progression

- Smart Contract
 - Transaction protocol that executes the terms of a contract
 - Smart property: property whose ownership is controlled via the blockchain using contracts (examples: cars, phones, houses)
- Dapp (Decentralized Application)
 - Contract plus graphical interface for contract execution
 - JavaScript API 'eth object' interacts with Ethereum blockchain
- DAO (Decentralized Autonomous Organization)
 - Self-enforcing smart contract (group of contracts) on a cryptographic blockchain, multiparty complexity
 - (Like remittances) avoid local business jurisdictional costs
 - Own Ethereum address (key) and balance, send and receive transactions, EtherScript scripts can modify their own code

Smart Contracts Use Cases

Blockchain is a cryptographic or encoded ledger (database) of transactions in the form of blocks arranged in a chain

Smart contract, a complex set of software codes with components designed to automate execution and settlement, is the application layer necessary to make blockchain technology a reality. Although smart contracts technology is in its infancy, use cases are evolving quickly.



Trade finance: Smart contracts can be set up as escrow accounts that monitor an exchange between two parties. It can track the location of the goods and when ownership has been transferred it can trigger payments.



P2P insurance: Insurance firms can automate the insurance policy by writing it into a smart contract. This technology can enable P2P insurance business models through templated smart contracts.



Loyalty and rewards: E-commerce, retail, and travel & tourism are some of the industries that can create a smart contract-driven loyalty and rewards system stored on a distributed ledger allowing interoperability.



Digital rights management and micropayments: Smart contracts that allow access to digital content such as music, images, and videos with access keys stored on distributed ledgers and also automates micropayments.



Land registry: Store the ownership of land/property on a distributed database and create safeguards for secure updates to this registry when transferring ownership through involvement of government/central body.



Securities issuance: Securities based on payments and rights to be executed according to predefined rules can be written as smart contracts. Current live examples being issuance of smart bonds and private stock markets.



Syndicated loans: Smart contracts can help reduce the settlement time for syndicated loans and help reduce loan issuance time and operational risks.



Event-driven insurance: Connected devices that store data on a distributed ledger help underwrite insurance and automate claims servicing.



Post-trade services: Smart contracts are triggered to ensure regulatory compliance of trades, ensure trade is executed as per the requirements, and take corrective steps as needed.



Distributed smart power grid: Ability for users to generate power and sell it over the grid that enables P2P payments and micro-transactions using smart contracts and distributed ledger.



Examples of Legal Smart Contract Applications

Decentralized Autonomous Organizations (DAO)

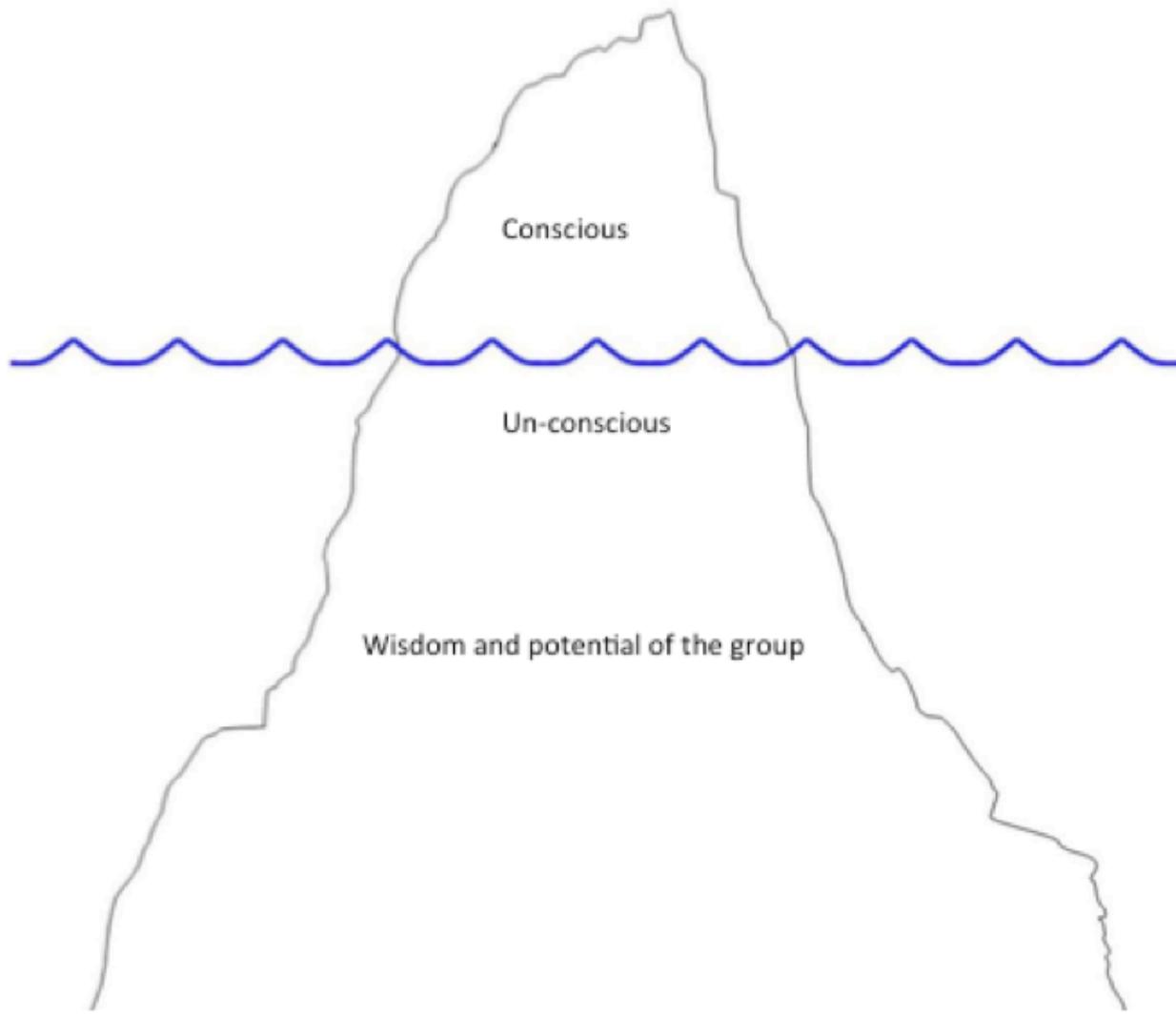


DAOs (Decentralized autonomous organizations)

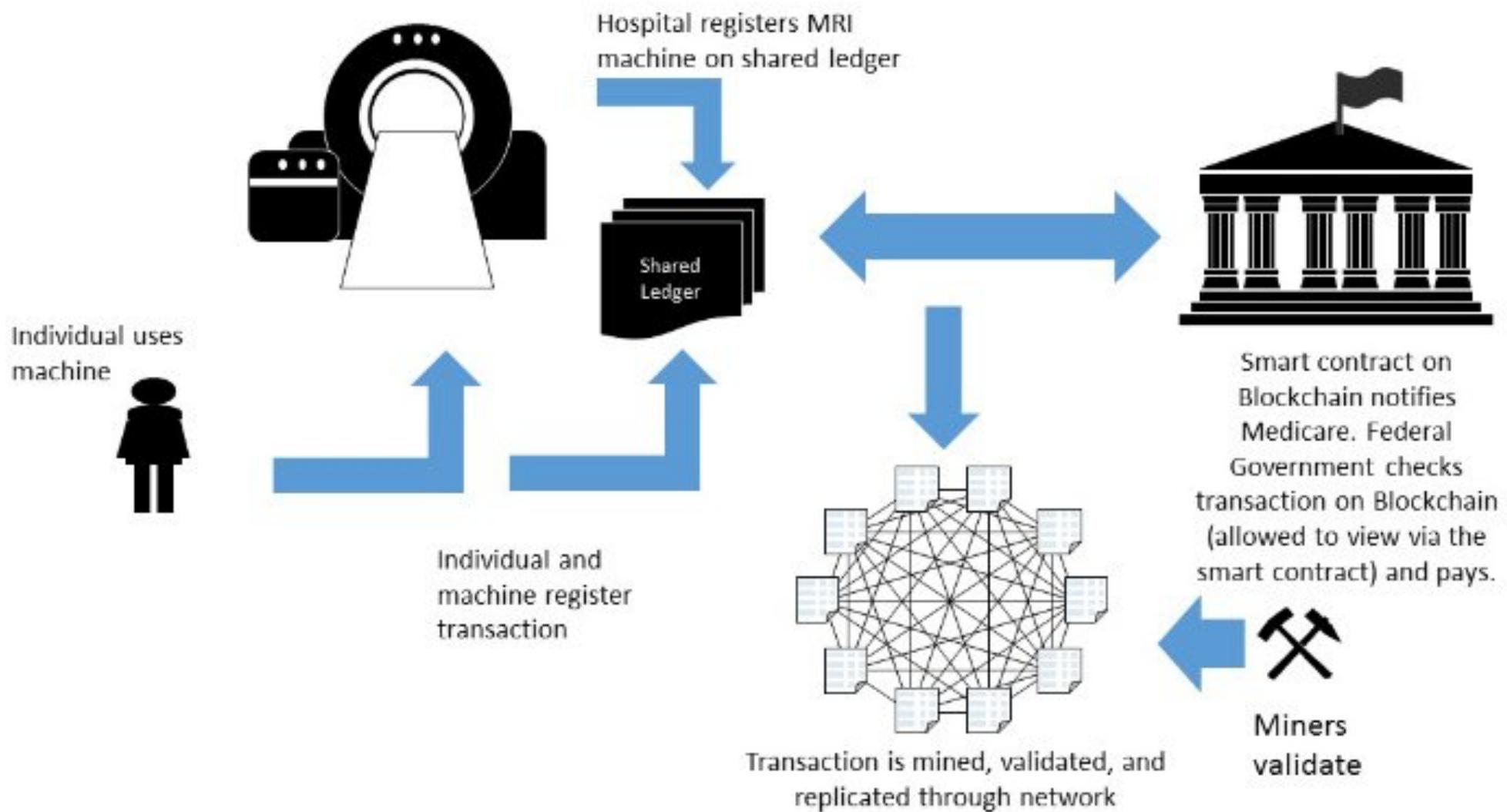
- It can be thought of as a corporation run without any human involvement under the control of an incorruptible set of business rules. These rules are typically implemented as publicly auditable open-source software distributed across the computers of their stakeholders.
- Autonomous entity
- No central control
- No dependency on legal contracts
- All resources and funds autonomously managed
- Self-enforcing smart contract on a cryptographic blockchain

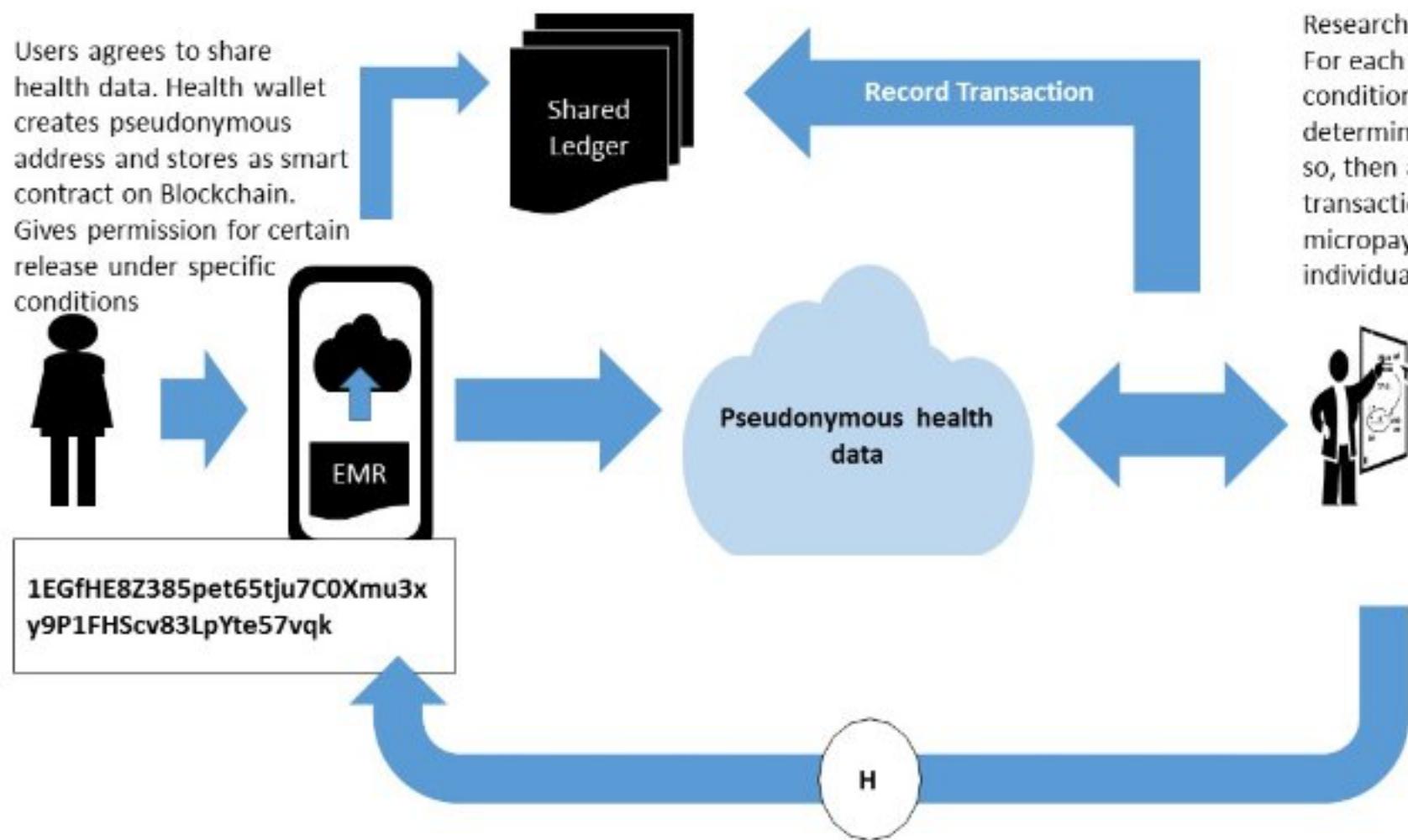
DAO Basics

- Creator: Authors of the open software through which platform arose
- Investors: DAO token holders who obtained voting rights in exchange for Ether (ETH)
- Contractors: Presents the ventures which the DAO could invest in
- Curators: Collects and verifies proposals and puts them up to a vote



Example: Smart Contracting of Health Data





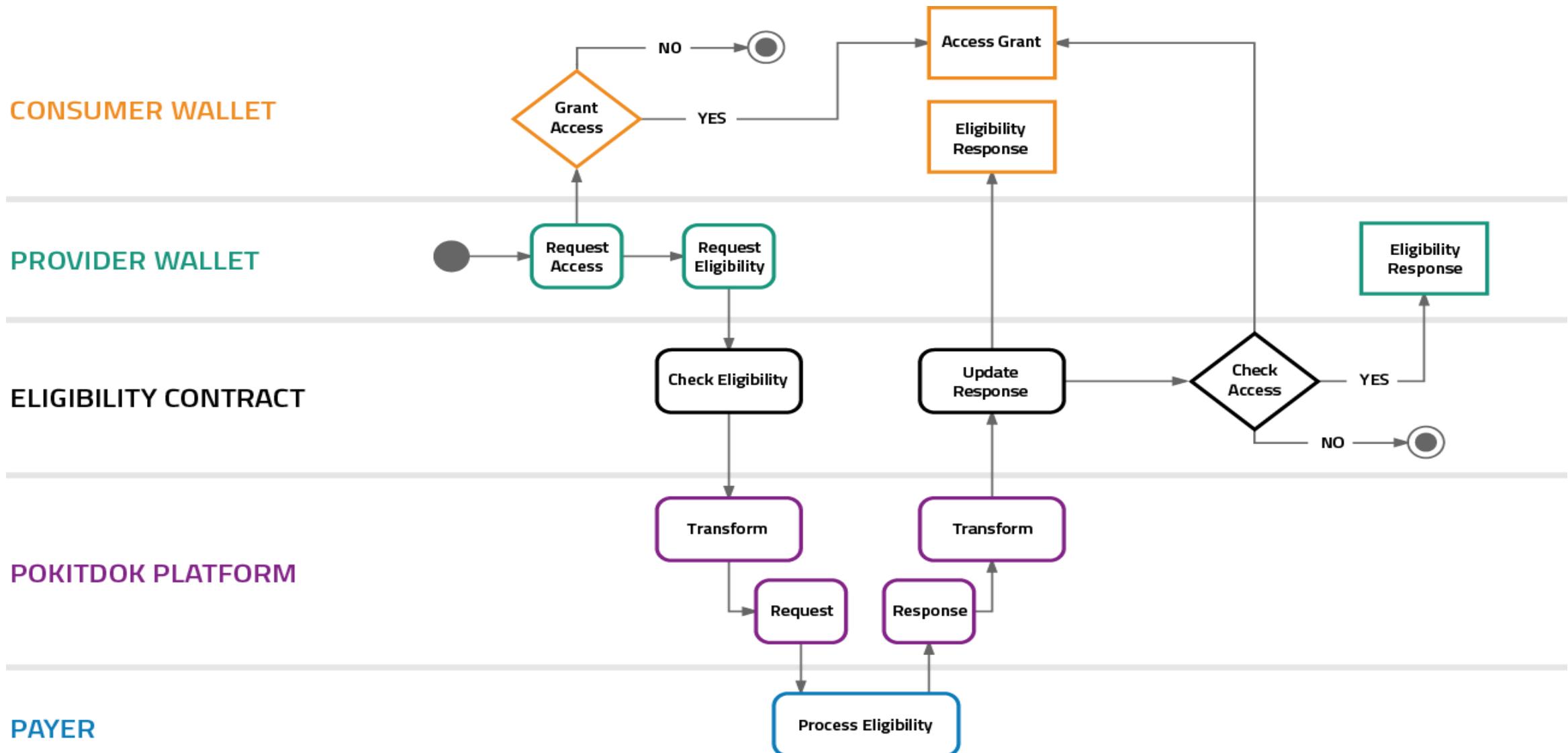
Users agrees to share health data. Health wallet creates pseudonymous address and stores as smart contract on Blockchain. Gives permission for certain release under specific conditions

Researchers wish to access data. For each record, they check conditions of smart contract to determine if the use is allowed. If so, then access the data, record transaction on Blockchain, make micropayment (in this case) to individuals health wallet

Health coin payment is made – can be converted to money, used in HSA, or to purchase medical services. Could be Bitcoin

Example: Patient-Centered Transactions

Smart contracts are a centerpiece



Blockchain Health Applications

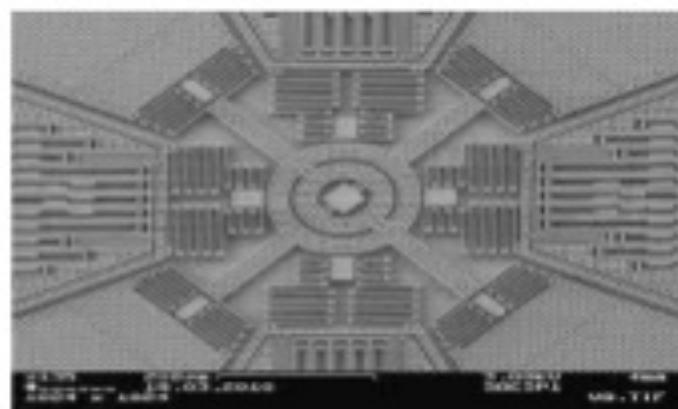
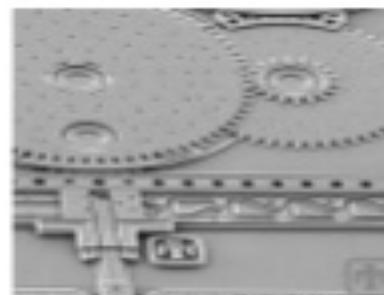


Immediate	Near-Future	Longer-term Future
Universal EMR	Smart Contracts and Health Insurance	Blockchain Happiness: Personal Development Chains
Health Databanks	Smart Property: Drug and Inventory Management	Virtual Patient Modeling
Quantified Self Data Commons	Health Token: HSN, Research	<i>Blockchain AI Applications:</i>
Big Data File Storage, Access, and Analysis	Smarthome, Personal Robotics, QS IOT	Friendly AI
Health Document Notarization and Tracking	Environmental Monitoring	Blockchain Deep-Learners
Identity Verification	Demurrage Redistributions	Blockchain Health Advocates
Health Vendor RFPs	Health Policy Voting	Digital Mindfile Services

Smart Property

Smart property

- ▶ Property whose ownership is controlled via a smart contract, which may or may not reside on a cryptoledger
- ▶ ‘Proplet’
- ▶ MEMS
- ▶ Internet of Things



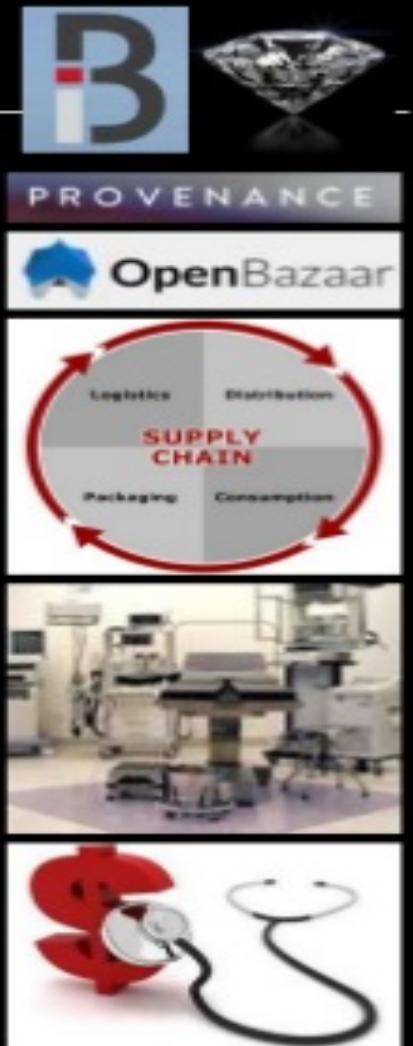
Smart Property and Smart Contracts

- Implications: a future of cryptographically-activated assets and actions...
- ...physical and intellectual property might be registered and transacted via blockchains as *smart property*, and
- ...agreements, contractual relationships, societal record-keeping, and governance might be enacted through code-based *smart contracts*



What is Smart Property?

- Register assets to blockchain via unique key
 - Real-time GPS ‘LoJack’ tracking for any asset
- Blockchain becomes an inventory, tracking, and exchange mechanism for hard assets
- Smart Property example projects
 - *Blocktrace* ledger tracks diamonds
 - *Provenance.org* tracks supply chain authenticity
 - *OpenBazaar* decentralized Craigslist exchange
 - *Factom-HealthNautica* medical billing and claims
 - Drug and equipment inventory, including origin and servicing records





PROBLEM

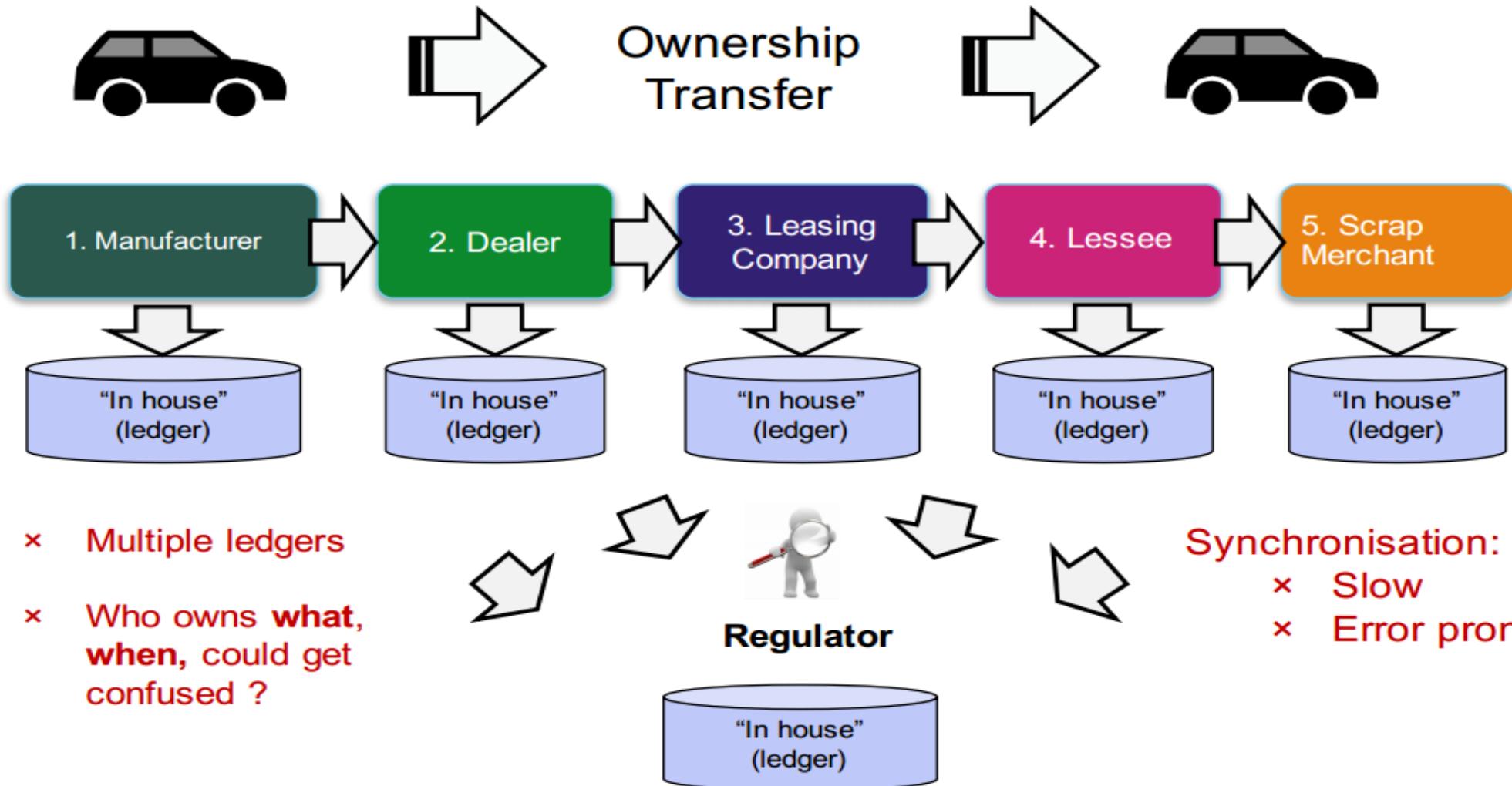
your property



How to rent/sell/share it?



Car Leasing Business Network



[HOME](#)[TRUSTATOM ID](#)[JOBS](#)[LEGACY APP LOGIN](#)

TRUST: the glue of life

Privacy-respecting identity & smart contract solutions for enabling
credibility, trust and safety

[Learn More](#)

your property

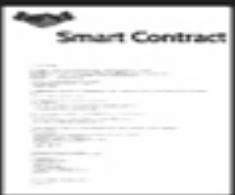


used only -5% of the time

How to rent/sell/share it?



DAO



tasks & orders
≡ →



Tasks:

- Fund the development
- Vote on major decisions
- Control the funds (!)
- Profitable

Service provider



Reward

- ←
- % fee of every transaction
- one time deployment fee



Tasks:

- Produce Slocks
- Marketing
- Partnerships

Slock Home Server



supports:

- Z-Wave
- Zigbee
- Bluetooth LE

Slock Power Switch

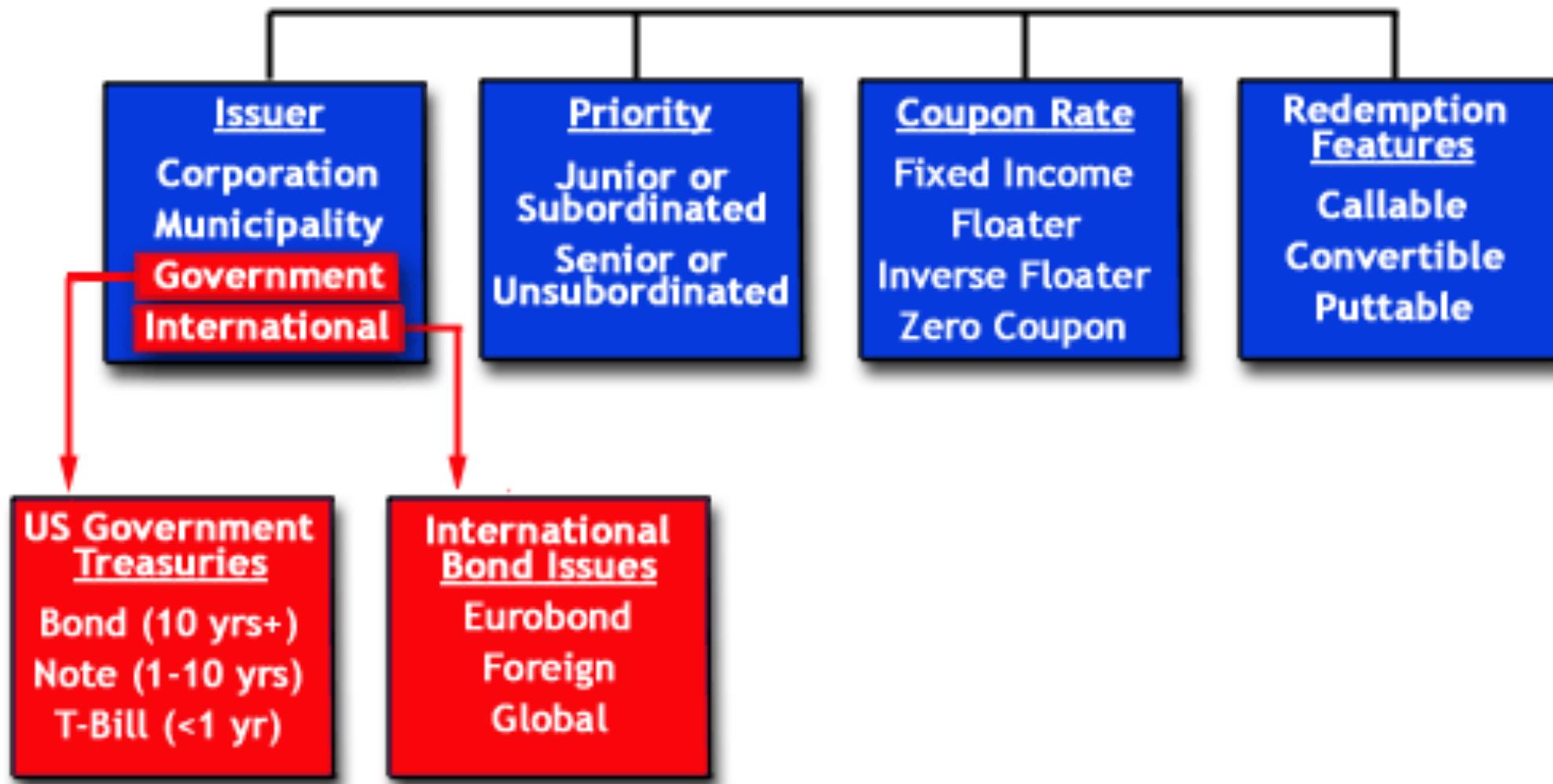


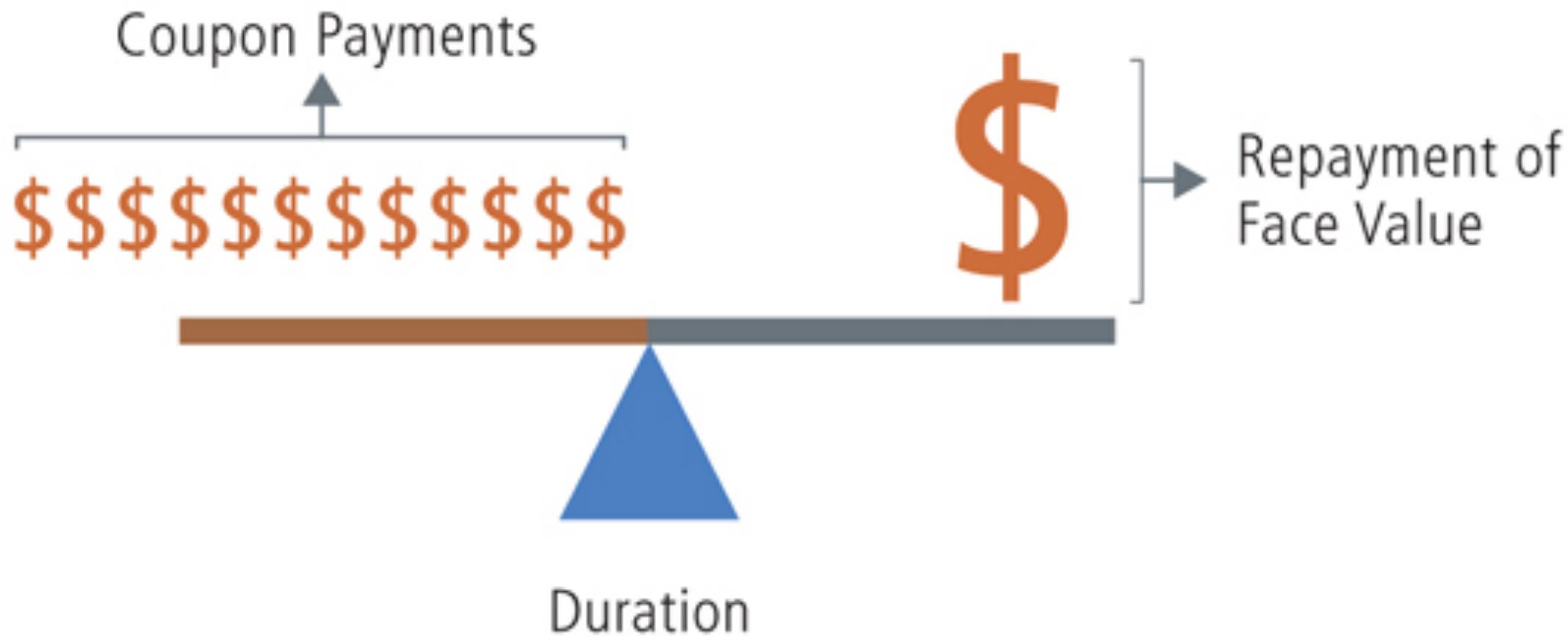
In Progress (with partners)

- Slock Door Lock
- Slock Bike Lock
- Slock Pad Lock
- Slock Car Lock

Example: Smart Contracting of Bond Trading

Bond Characteristics





	Coupon	Mat. date	Bid \$	Yld%
Corporate				
AGT Lt	8.800	Sep 22/25	100.46	8.75
Air Ca	6.750	Feb 02/04	94.00	9.09
AssCap	5.400	Sep 04/01	100.01	5.38
Avco	5.750	Jun 02/03	100.25	5.63
Bell	6.250	Dec 01/03	101.59	5.63
Bell	6.500	May 09/05	102.01	5.95
BMO	7.000	Jan 28/10	106.55	6.04
BNS	5.400	Apr 01/03	100.31	5.24
BNS	6.250	Jul 16/07	101.56	5.95
CardTr	5.510	Jun 21/03	100.52	5.27
Cdn Pa	5.850	Mar 30/09	93.93	6.83
Clearn	0.000	May 15/08	88.50	8.61
CnCrTr	5.625	Mar 24/05	99.78	5.68
Coke	5.650	Mar 17/04	99.59	5.80

Column 1

Column 2

Column 3

Column 4

Column 5

Bond Basics

Bond Trading

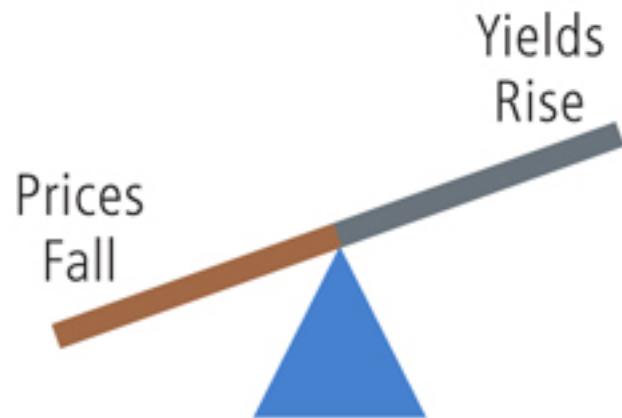
- Bonds traded on national securities exchanges.
- Newspapers and the financial press publish bond prices and trading activity daily.

Illustration 15-4

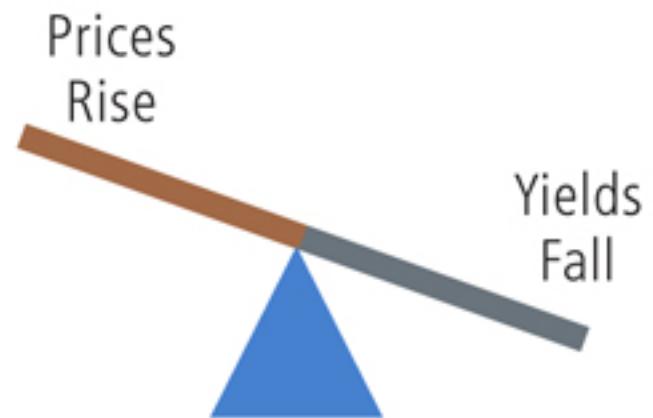
Bonds	Maturity	Close	Yield	Est. Volume (000)
Boeing Co. 5.125	Feb. 15, 2011	96.595	5.747	33,965

Read as: Outstanding 5.125%, \$1,000 bonds that mature in 2011. Currently yield a 5.747% return. On this day, \$33,965,000 of these bonds were traded. Closing price was 96.595% of face value, or \$965.95.

If interest rates rise:



If interest rates fall:

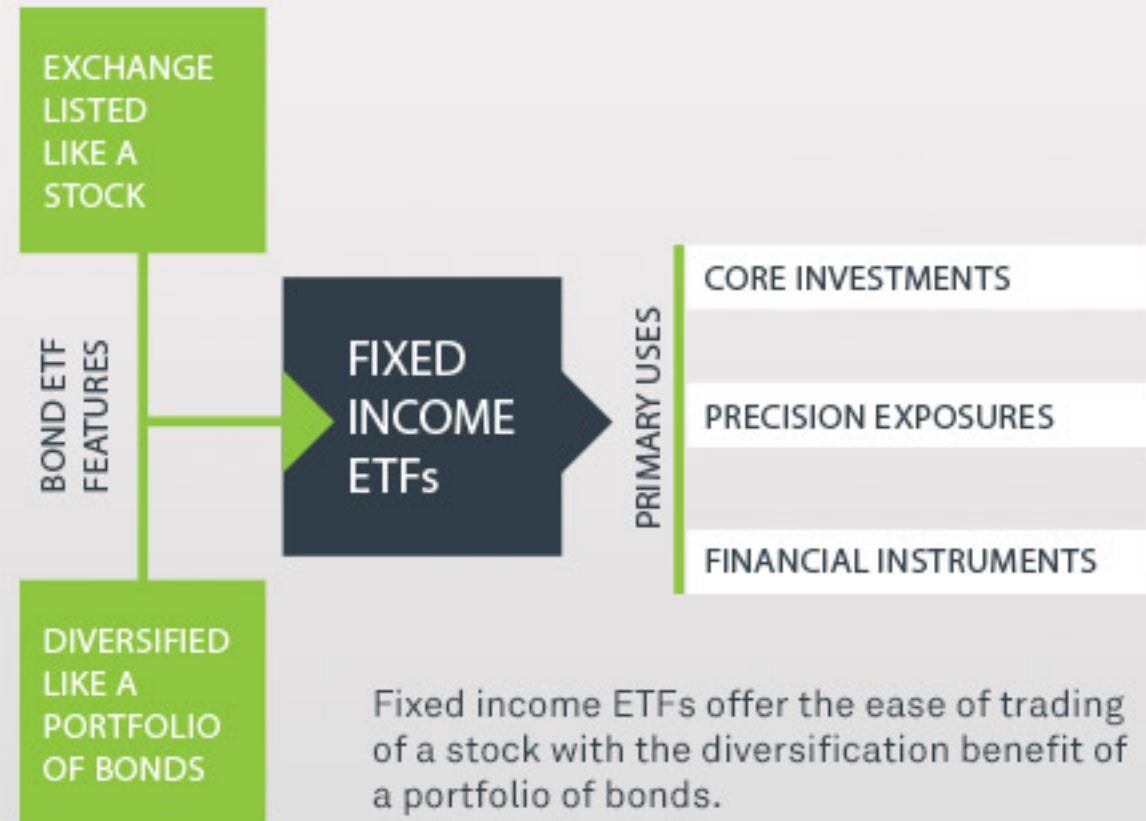


Risk Associated with Bonds

1. Interest-rate risk
2. Reinvestment risk
3. Call risk
4. Credit risk
5. Inflation risk
6. Exchange-rate risk
7. Liquidity risk – institutional investor must trade frequently in some extent
8. Risk risk

THE BASICS OF FIXED INCOME ETFs

Fixed income ETFs consist of a portfolio of bonds and are traded on an exchange like an equity security.



Fixed income ETFs offer the ease of trading of a stock with the diversification benefit of a portfolio of bonds.

A simplified version of today's market structure coded as smart contracts

- In order to trade bonds, people need to find bonds for sale.
- Developer Bob spots a business opportunity and develops an on-line bonds-for-sale list website.
 - Bond sellers can add a new for-sale bond to the list, along with their contact details.
 - Bond buyers can contact the seller by phone or email and negotiate the sale.
 - Once sold, the seller removes the bond from the list.
- **All of the business logic for the bonds-for-sale list application is coded into an Ethereum smart contract, that stores every bond as value data under a unique key within contract's permanent store.**
 - To earn money from the application, Bob's smart contract charges a small Ether fee paid for new listing transactions.

A simplified version of today's market structure coded as smart contracts

- Bob further enhances the bond online application by:
 - Expanding from store for-sale bonds to all bonds owned by customers – benefit: no need to pay a securities depository agent, to manage bond record keeping for customers.
 - Bob's application manages the bond ownership transfer process – benefit: Bob's application adds additional value data in his smart contract to indicate if a bond is for sale or not, which can only be changed by a transaction created by the current owner.
 - Adding transfer function to his code that again only the owner can call, once the owner receives money in their bank account.

A simplified version of today's market structure coded as smart contracts

- Problem:

- bond owners call Bob and ask him when their coupon payments will be paid to them.
- Bob never considered this use case, and as he is in a rush, he decides to find a bond servicing business to partner with.
- He finds Sally, who runs such a business, and Sally is able to quickly take a data feed from the smart contract and process coupon payments off-chain, for which she of course charges a fee.
- Alice, another developer, spots an opportunity to create an alternative to Bob's application, which automates both coupon payments in Ether and bond versus Ether atomic transfer within the smart contract.
- She is not incentivized to collaborate with Bob, as he collects all of the fees, see [DApp Centralisation](#)

A simplified version of today's market structure coded as smart contracts

- Some people are willing to accept Ether as money and wish to transfer bond listings to Alice's application, whilst others choose to stick with Bob's application as they are only willing to accept fiat money.
- Everyone agrees that double-listing is a bad idea, so Terence proposes that he becomes a listing transfer agent.
 - He writes a smart contract that allows an owner to transfer their bond records from Bob's application to Alice's in a way that guarantees removal of bond listing in Bob's application, but only if and once it is listed in Alice's application.
- **So whilst Bob removed the need for Derek, a securities depository agent, and Alice removed the need for Sally, a servicing agent, she introduced the need for Terence, a transfer agent. She also split the for-sale list, making it harder for buyers and sellers to find each other**

A simplified version of today's market structure coded as smart contracts

- Eric decides to create an exchange smart contract, that will include bonds from both Bob's and Alice's applications.
- Because some buyers will want to buy bonds from Mary's application, but are only willing to use fiat money, Eric creates an e-money token smart contract and acts as a market maker, working together with Terence, to allow purchase of bonds in Mary's application using fiat e-money instead of Ether and transfer to Bob's application once completed.
- Since many of the described transactions are still manual in the contemporary lifecycle flow, the process takes a long time as humans operate at a slower pace than machines, and typically only nine till five.
- Processes that are automated, are still often batch based, so dependent transactions that miss a batch window push out total time to complete the front-to-back flow.
- Even if all processes are based on automated and real time web services provided by different organizations, each web service represents a single point of failure, and no one participant has an overview of the front-to-back state i.e. it's hard for an end user to figure out where something is stuck

Benefits of Coding Current Market Structure

- Smart contracts offer an alternative to web services as a means of functional decomposition i.e. it is clear which organization owns which part of the process.
- Mitigates nicely blockchain's regulatory risk i.e. only regulator licensed transfer agent's smart contract would be accepted for use. After all, regulators regulate behavior of licensed agents not technology.
- Since all smart contracts run in full on all consensus forming nodes, you eliminate today's single point of failure risks.
- As all inter-contract depended transactions that are generated in response to a user's transaction occur within the same block, even highly complex inter-organizational front-to-back flows occur in near real-time, **reducing settlement risk and freeing up liquidity and capital reserves.**
- Since the entire flow is stored on chain and available to all nodes, users are never left wondering as to when and what went wrong where.
- **Regulators can dispense with running own monitoring solutions and instead verify that the smart contracts enforce their controls and rely on the self-auditing nature of blockchain networks to guarantee enforcement of those controls, plus they get near real time data that can be used to fight systemic risk build up.**

Benefits of Coding Current Market Structure

- Further improvement:
 - current market model introduces a lot of friction due to data silos e.g. need to transfer listings, but data silos occur naturally due to competition.
 - Data silos are much easier to reconcile on-chain than in today's world because essentially all data is in the same database, so reconciliation is between contract stores, as such data availability is not an issue, but still inter-contract data transfer is required, and that costs Ether.
- Cutting out lawyers:
 - Lawyer uses user answers fill in a template, the product of which is the sales contract, that he hands over to you for a fee, and which is yours to keep and use how you see fit.
 - This is in stark contrast to most of today's on-line services that bundle business logic and data services together, allowing them to monetize user data and making it hard for you to switch service providers.
 - But on-chain, the operators of the platform are different to application developers, so users can de-couple the two and give back data ownership.

Bond Contracts as Smart Contracts

- Just like with a paper contract, users can write their own smart contract, assuming it is an unregulated activity, but may not have the expertise or time to do so.
- So instead, users find a distributed application, most likely accessed as a website, that collects example bond details, and pass them to a factory smart contract, that charges a fee and spits out a new smart contract with users' bond details baked in.
- The created smart contract and its data now belongs to the bond owner, and the creation fee includes code usage license.
 - Since many bonds are regulated, users would have to use a distributed application written by a regulated entity
 - Regulators could even create a registry smart contract to track approved by them factory smart contracts, so you are sure that you are using a regulated one.
- Different organizations can compete to write the best possible bond smart contract factory. They can also differentiate themselves on the quality of guarantees, warranties, liability, support, jurisdiction coverage etc.

Regulatory Approval of Smart Contract Bond

- Query the regulator's registry contract to find all approved factory contract addresses, and then find all bond contracts that have been created by those addresses.
- As a convenience, each factory provider could also create a private registry for quick look up. However, just because anyone with a full node can access the data, it does not mean they have the right to use it, as under my proposal the bond owners, own the data and the code license to operate that specific data set.
- This opens up huge data monetization opportunities. Let's say someone wanted to advertise something to bond owners.
 - Like Lunyr, someone could create a new service, BondAds, and a new coin that would have to be purchased by advertisers in order to advertise to bond owners.
 - Whilst BondAds could just spin up a node and scrape data to populate their dataset, they would be in violation of data owner's rights and litigation would follow.
 - On the other hand, BondAds could create a registry smart contract, and invite bond owners to register their bond smart contracts in return for coins.
 - By registering the bond contract, user in effect assign rights to monetize the data in return for share of generated advertising revenues.

Regulatory Approval of Smart Contract Bond

- Because BondAds would only register the smart contract address, not hold it's own data on it, users eliminate data reconciliation needs.
 - Same bond smart contract could be registered not just with the BondAds registry, but also with Bob's, Alice's and Eric's applications, which are in effect all types of a registry contract that holds contract links, thereby eliminating reconciliation.
 - Of course each registry may require a different set of methods and privileges from a bond contract, but that's trivial to achieve by simply adding new delegate contracts to the bond smart contract, all operating on the same underlying data, but in a controlled manner.
- Registry smart contracts on blockchain in effect become the securities depositories, each deposit being a bond smart contract.
- Regulation is not violated, as it's easy to prove that registry, bond and delegate smart contracts, which together manage bond's lifecycle were created by regulated entities.
- Some lifecycle events end up migrating from current model but are performed by smart contracts instead of by humans and web services, whilst others, like transfer events, are entirely deprecated.
- In this model, it's possible to achieve competition, yet eliminate data silos and associated friction, whilst keeping mentioned benefits of migrating to blockchain.

WHAT IS SOLIDITY & ETHEREUM VIRTUAL MACHINE?

What is Solidity?

- High-level object-oriented language for smart contracts
- Solidity was initially proposed in August 2014 by Gavin Wood
- Solidity lets you program on [Ethereum](#), a blockchain-based virtual machine
- Solidity is a statically typed programming language
- A Contract programming language that has similarities to Javascript and C
- Contract-specific features include modifier (guard) clauses, event notifiers for listeners, and custom global variables.
- Solidity is compiled to [bytecode](#) that is executable on the EVM

SOLIDITY

- **Solidity** is an object oriented domain-specific language. Popular language to write Ethereum's smart contract.
- Ethereum VM and solidity are Turing Complete



Everything can be implemented in a Turing complete environment



EXTENSIONS



...



Extension: solidity

...

Search Extensions in Marketplace

Material Icon Theme

Material Icon Theme

Philipp Kief

**solidity** 0.0.18

Ethereum Solidity Lang...

Juan Blanco

**solidity**

JuanBlanco.solidity

Juan Blanco

2913



Ethereum Solidity Language for Visual Studio Code

Disable ▾

Uninstall

DetailsContributionsChangelogDependencies

Solidity support for Visual Studio code

Solidity is the language used in Ethereum to create smart contracts, this extension provides:

- Syntax highlighting
- Snippets
- Compilation of the current contract (Press F1 Compile Current Solidity Contract).
- Compilation of all the contracts (Press F1 Compile all Solidity Contracts)
- Compilation supporting EIP82 (dappfile and dependency packages)

Solidity

compiles to

Ethereum Bytecodes

executed by

Ethereum VM

A Solidity Contract

```
contract Coin {  
    address minter;  
    mapping (address => uint) balances;  
    function Coin() {  
        minter = msg.sender;  
    }  
    function mint(address owner, uint amount) {  
        if (msg.sender != minter) return;  
        balances[owner] += amount;  
    }  
    function send(address receiver, uint amount) {  
        if (balances[msg.sender] < amount) return;  
        balances[msg.sender] -= amount;  
        balances[receiver] += amount;  
    }  
}
```

Global Variables

`block.coinbase (address)`: current block miner's address
`block.difficulty (uint)`: current block difficulty
`block.gaslimit (uint)`: current block gaslimit
`block.number (uint)`: current block number
`block.timestamp (uint)`: current block timestamp
`msg.data (bytes)`: complete calldata
`msg.gas (uint)`: remaining gas
`msg.sender (address)`: sender of the message (current call)
`msg.value (uint)`: number of wei sent with the message
`now (uint)`: current block timestamp (alias for `block.timestamp`)
`tx.gasprice (uint)`: gas price of the transaction
`tx.origin (address)`: sender of the transaction (full call chain)

Global Functions

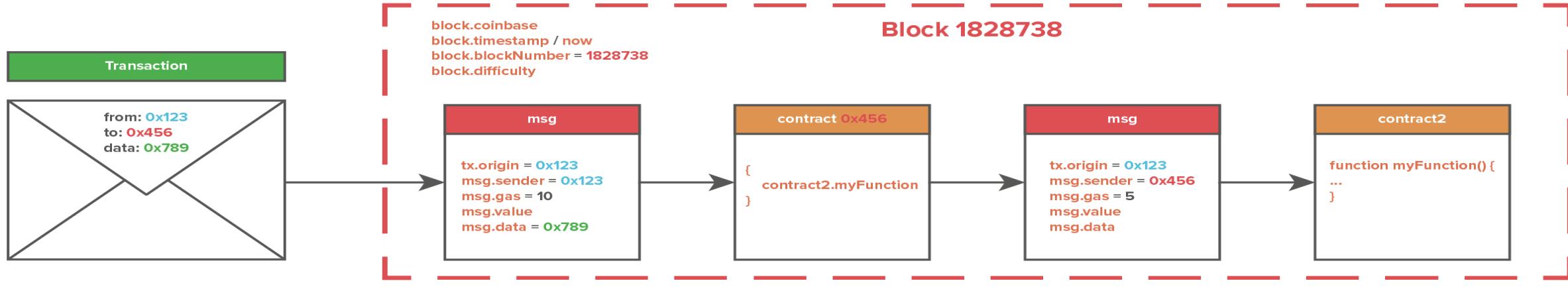
`block.blockhash(uint blockNumber) returns (bytes32)`: hash of the given block - only works for 256 most recent blocks
`sha3(...)` returns (bytes32): compute the Ethereum-SHA-3 (KEC-CAK-256) hash of the (tightly packed) arguments
`sha256(...)` returns (bytes32): compute the SHA-256 hash of the (tightly packed) arguments
`ripemd160(...)` returns (bytes20): compute the RIPEMD-160 hash of the (tightly packed) arguments
`ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address)`: recover address associated with the public key from elliptic curve signature
`addmod(uint x, uint y, uint k) returns (uint)`: compute $(x + y) \% k$ where the addition is performed with arbitrary precision and does not wrap around at 2^{256}
`mulmod(uint x, uint y, uint k) returns (uint)`: compute $(x * y) \% k$ where the multiplication is performed with arbitrary precision and does not wrap around at 2^{256}
`selfdestruct(address recipient)`: destroy the current contract, sending its funds to the given address

Scope Variables

`this` (current contract's type): the current contract, explicitly convertible to `address`
`super`: the contract one level higher in the inheritance hierarchy

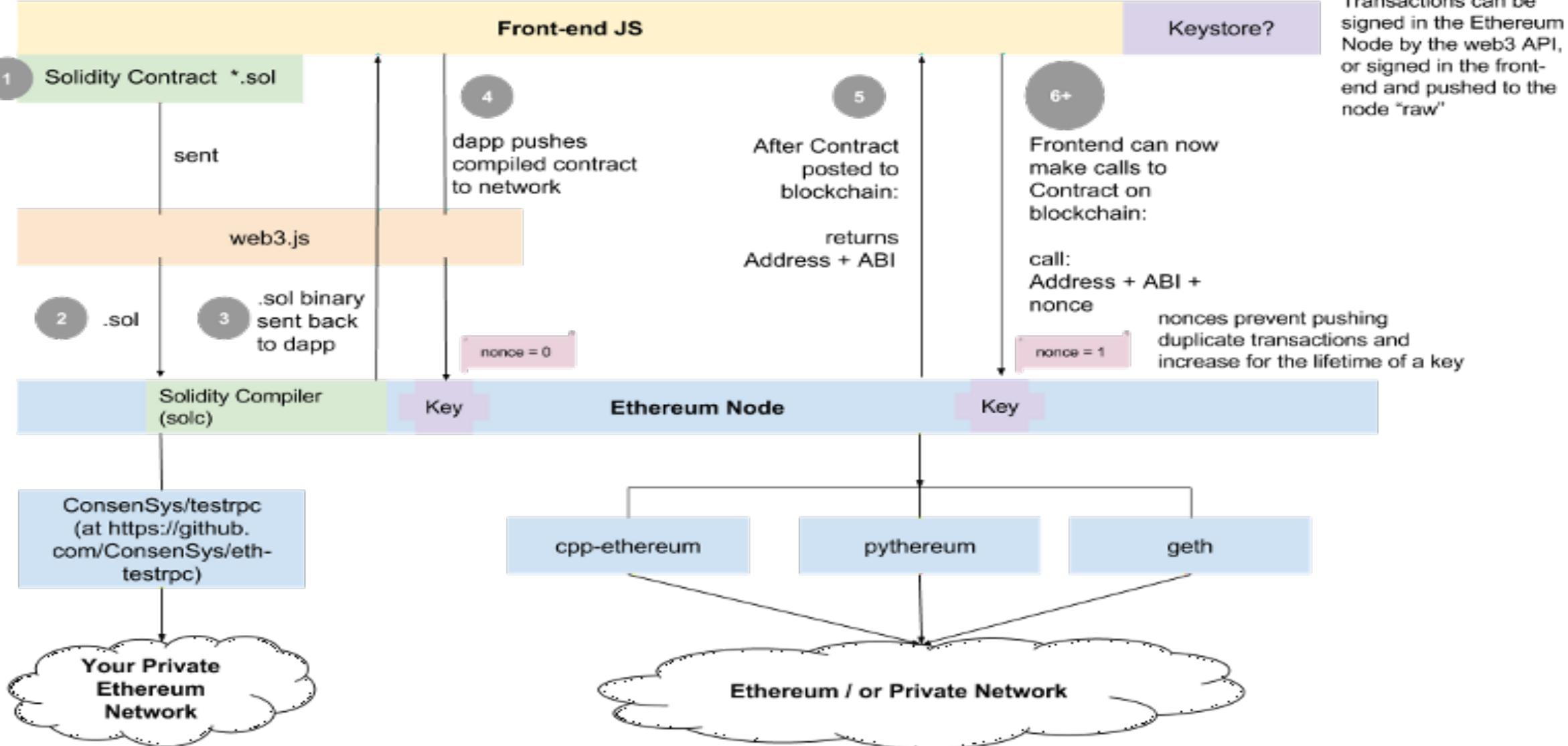
Function Visibility Specifiers

`public`: visible externally and internally (creates accessor function for storage/state variables)
`private`: only visible in the current contract
`external`: only visible externally (only for functions) - i.e. can only be message-called (via `this.fun`)
`internal`: only visible internally



dApp Front-end Steps

?



A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer or Function Call Transaction** is assumed in step 6.

Shortcomings and Possible Pitfalls of Code as Law

Prof. Wulf A. Kaal

WARNING: Code is NOT Law

- The blockchain industry needs pragmatism – not idealism.
- While blockchains [can be immutable](#), at some point in time, all cryptocurrency development teams have faced situations where they had to go back and rewrite the past.
- Code is not law.
- The code has bugs:
 - Even bitcoin has seen its fair share of snafus
 - 2010, when a [bug in bitcoin's code](#) led to the creation of 92m bitcoins (thereby breaking the hard-coded rule that only 21m bitcoin will ever exist)
 - 2013, when the bitcoin network split after a bug in the software created two divergent chains
 - Infamous hack on [The DAO](#) that ultimately led the [ethereum smart contract platform](#) to hard fork

Problems with Blockchains

- Exchanges where cryptocurrencies are traded (along with our mobile phones and computers that interface with them) are simply not built for handling high value digital assets.
- Private blockchains that use byzantine fault tolerant protocols are "doing it wrong":
 - *All of your nodes must fail independently, and yet you are deploying the same code on every machine.*
 - This could create situations where issues with smart contract code held on private blockchain networks could cause all the computers in the network to be compromised.
- Another problem is that smart contracts are being coded in languages too similar to Javascript, making it difficult for coders to spot mistakes or to predict whether a smart contact will work the way it is intended.
- Blockchain is an exciting field, it needs to be approached in a rational, scientific manner that takes failure into account.
- *There is a great promise at the end, but there will be many failures.*

Smart Contracts as a Legal Challenge

- *Smart contracts present a problem for the forensic exercise of contractual interpretation.*
 - *Currently, when the meaning of a contract is disputed by the parties to it, a court will consider what that agreement would mean to a reasonable human observer.*
 - *Where that agreement is written in computer code, however, and intended for communication to an artificial intelligence, the significance of a reasonable human observer's interpretation is a matter of contention.*
 - *There is no such thing as a reasonable computer.*
 - *Whilst the court can of course enlist the services of a professional coder to translate the code into human language, the process of interpreting that language remains the responsibility of the judge.*
 - *Given that the logical architecture of computer code is different to that employed by human language, the task of interpreting that translation is not the same as the task of interpreting language intended for human understanding*

Smart Contracts as a Legal Challenge

- *In more general terms, smart contracts are also “trustless” in the sense that they can be relied upon even in the absence of trust between the parties concerned. This is because they facilitate a complete symmetry of information, and a completely transparent process. In doing so, they may well augur a new era of contracting practices; a “post co-operative” commercial environment.*
- *These technological developments are likely to have, therefore, a disruptive effect on private law, both conceptually and practically. If the law is to remain relevant to commercial dispute resolution on a global scale, it needs to upgrade its doctrinal tools and concepts. The challenge for lawyers is to determine how best to do that.*

Condensed Course: Coding Exercises

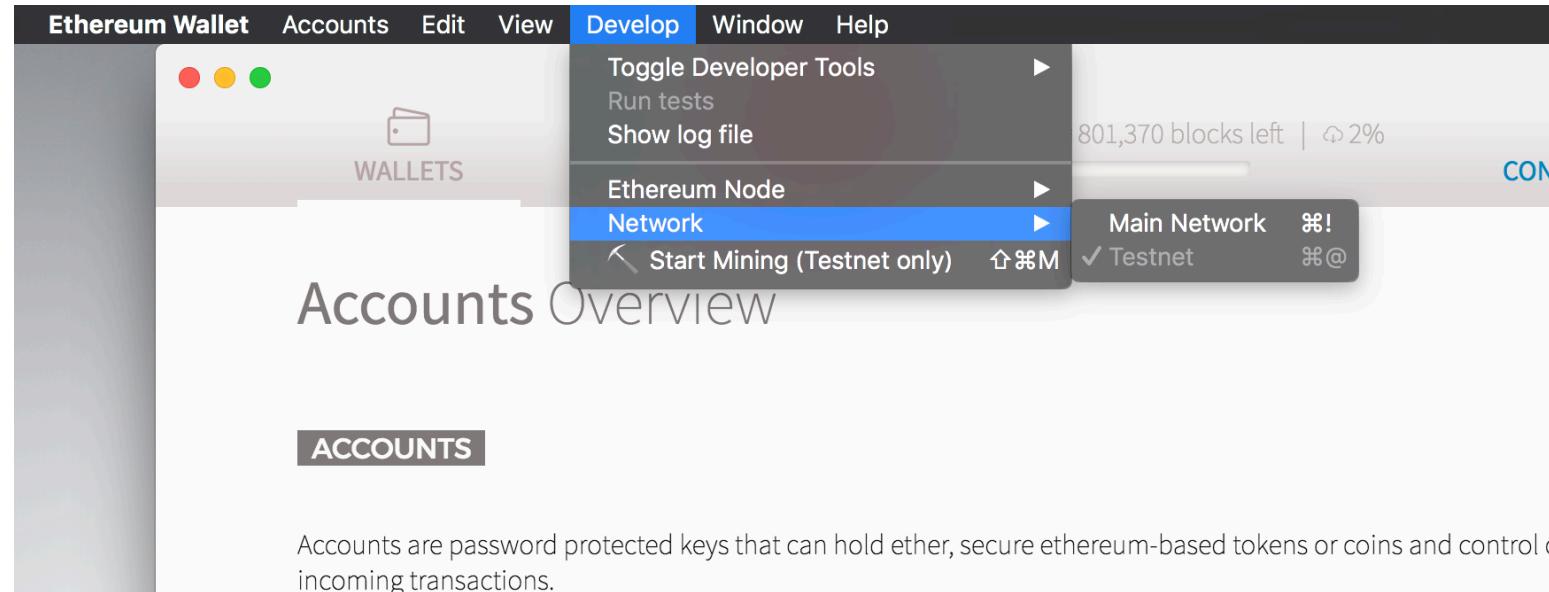
1. Basic Contract in Solidity
2. Blockchain Issuance of Stock and Transfer of Stock

Condensed Course: Coding Exercise - Basic Contract in Solidity

Before you begin: installations

- Ethereum Wallet
 - Available at: ethereum.org
- Sublime Text (coding text editor)
 - Available at: sublimetext.com

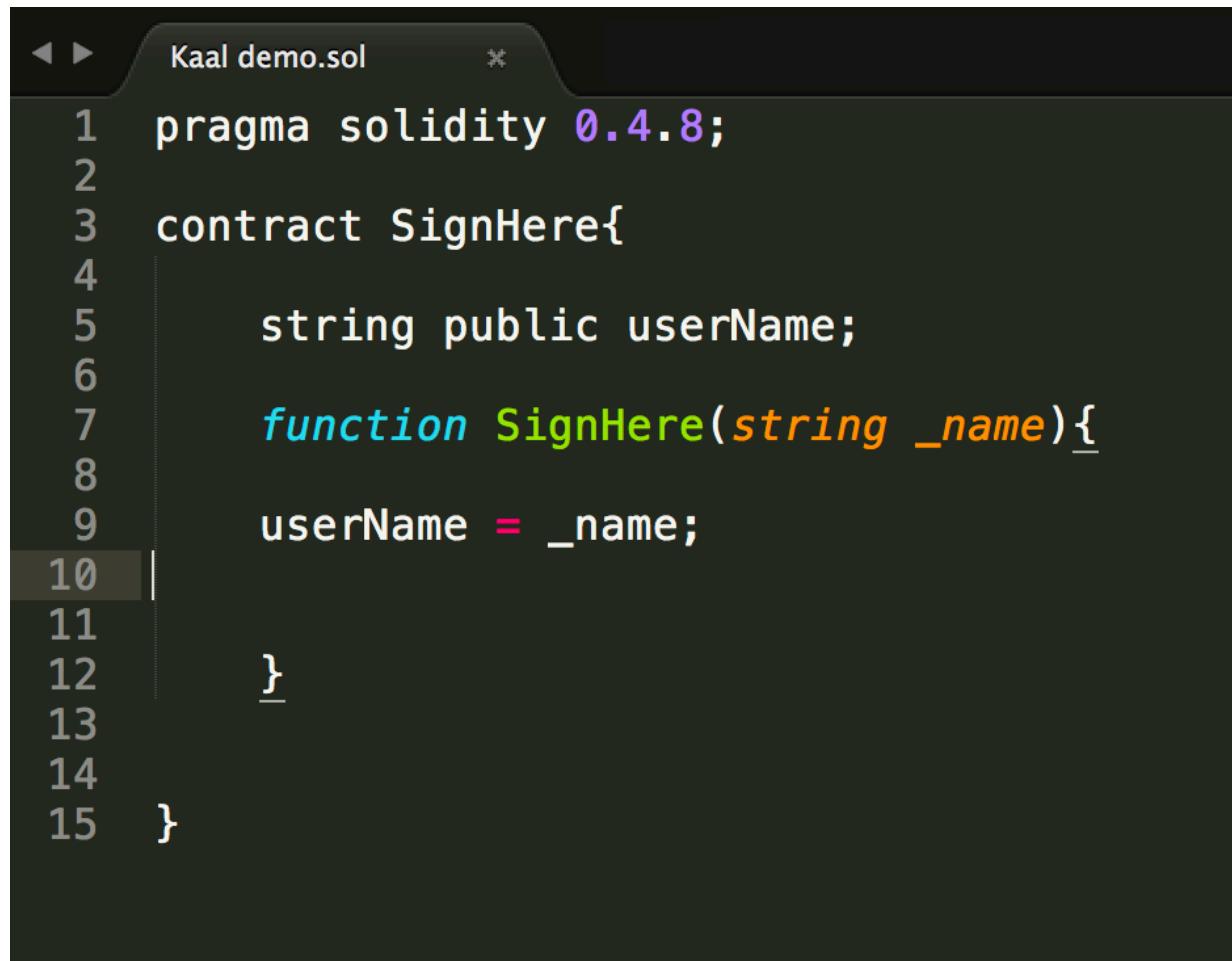
Configuring Ethereum Wallet



- VERY IMPORTANT - ensure that you are working on Ethereum's testnet blockchain
- After switching to the testnet, select “Start Mining (Testnet only)”
- After a few minutes, your main account will have “Ether” the digital currency used with Ethereum blockchain. Ether allows us to deploy test contracts. Select “Stop Mining” after you have a few Ether.

Basic contract in Solidity

- Using a coding text editor, you can write a contract that is executable on Ethereum’s Testnet blockchain.
- Our example: a single variable smart contract called “Sign Here”. This is a contract that deploys on the blockchain when a user signs his or her name.
- “Sign Here” code could be a part of any contract on the blockchain, such as a sales contract.



A screenshot of a Sublime Text editor window titled "Kaal demo.sol". The code is written in Solidity and defines a contract named "SignHere". The code includes a public variable "user_name" and a function "SignHere" that sets the value of "user_name" to the input parameter "_name".

```
1 pragma solidity 0.4.8;
2
3 contract SignHere{
4
5     string public user_name;
6
7     function SignHere(string _name){
8
9         user_name = _name;
10
11
12     }
13
14
15 }
```

**code drafting in Sublime Text

Code Breakdown

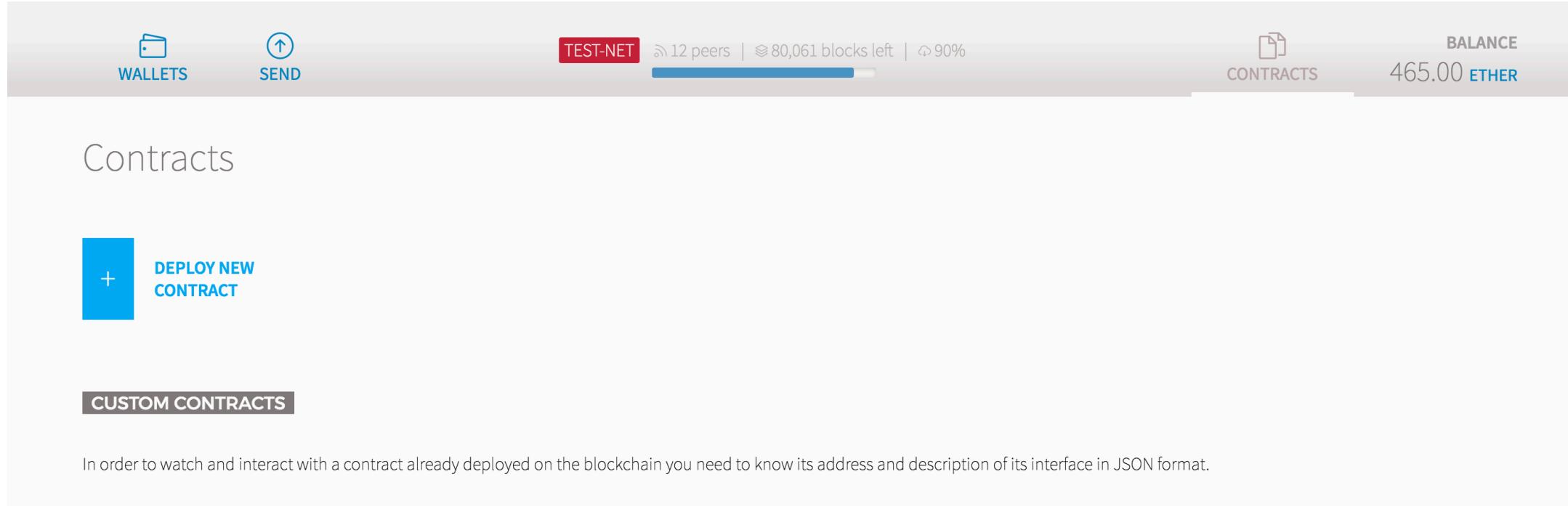
`pragma solidity 0.4.8;` = the syntax the code was developed with. Allows code to properly compile. It says “Hey Ethereum, I am from Solidity, compile what follows please.”

`contract SignHere{` = “contract” is the highest object that can exist on the Ethereum blockchain. “SignHere” is the name of the contract.

`string public userName;` = the only variable in this contract, called “userName”. It is a *public* string, which means that this contract could interact with other contracts deployed on the Ethereum blockchain.

`function SignHere(string _name){` `userName = _name;` = This function is a constructor, which means that this contract will not be deployed on the blockchain until the function completes/has a value. The constructor function is asking for a name. Once the person’s name is entered, the function is complete and the contract can be deployed.

Test it



- On testnet, select the “Contracts” tab, then “Deploy New Contract”.

- Copy the code from the coding text editor and paste into the “Solidity Contract Source Code” box.
- If done properly the code will compile. Note that if you remove the first line of code, it will not compile.
- If compiled, select the “Pick a Contract” drop down and select “Sign Here”. Sign your name, and deploy the contract on the Testnet.



WALLETS

SEND

TEST-NET

3 peers | 820,841 5s since last block

CONTRACTS

0.00 ETHER

SOLIDITY CONTRACT SOURCE CODE

```
1 pragma solidity 0.4.8;
2
3 contract SignHere{
4
5     string public userName;
6
7     function SignHere(string _name){
8
9         userName = _name;
10
11    }
12
13
14
15 }
```

CONTRACT BYTE CODE

SELECT CONTRACT TO DEPLOY

Pick a contract

SELECT FEE

0.006 ETHER

CHEAPER

FASTER

This is the most amount of money that might be used to process this transaction. Your transaction will be mined **usually within a minute.**

TOTAL

0.006 ETHER



WALLETS

SEND

TEST-NET

4 peers | 820,871 22s since last block

CONTRACTS

0.00 ETHER

You want to send **0 ETHER**.

SOLIDITY CONTRACT SOURCE CODE

```
1 pragma solidity 0.4.8;
2
3 contract SignHere{
4
5     string public userName;
6
7     function SignHere(string _name){
8
9         userName = _name;
10
11    }
12
13
14
15 }
```

CONTRACT BYTE CODE

SELECT CONTRACT TO DEPLOY

Sign Here

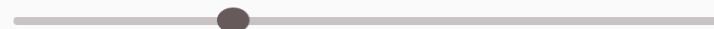
CONSTRUCTOR PARAMETERS

`_name - string`

Wulf

SELECT FEE

0.006 ETHER



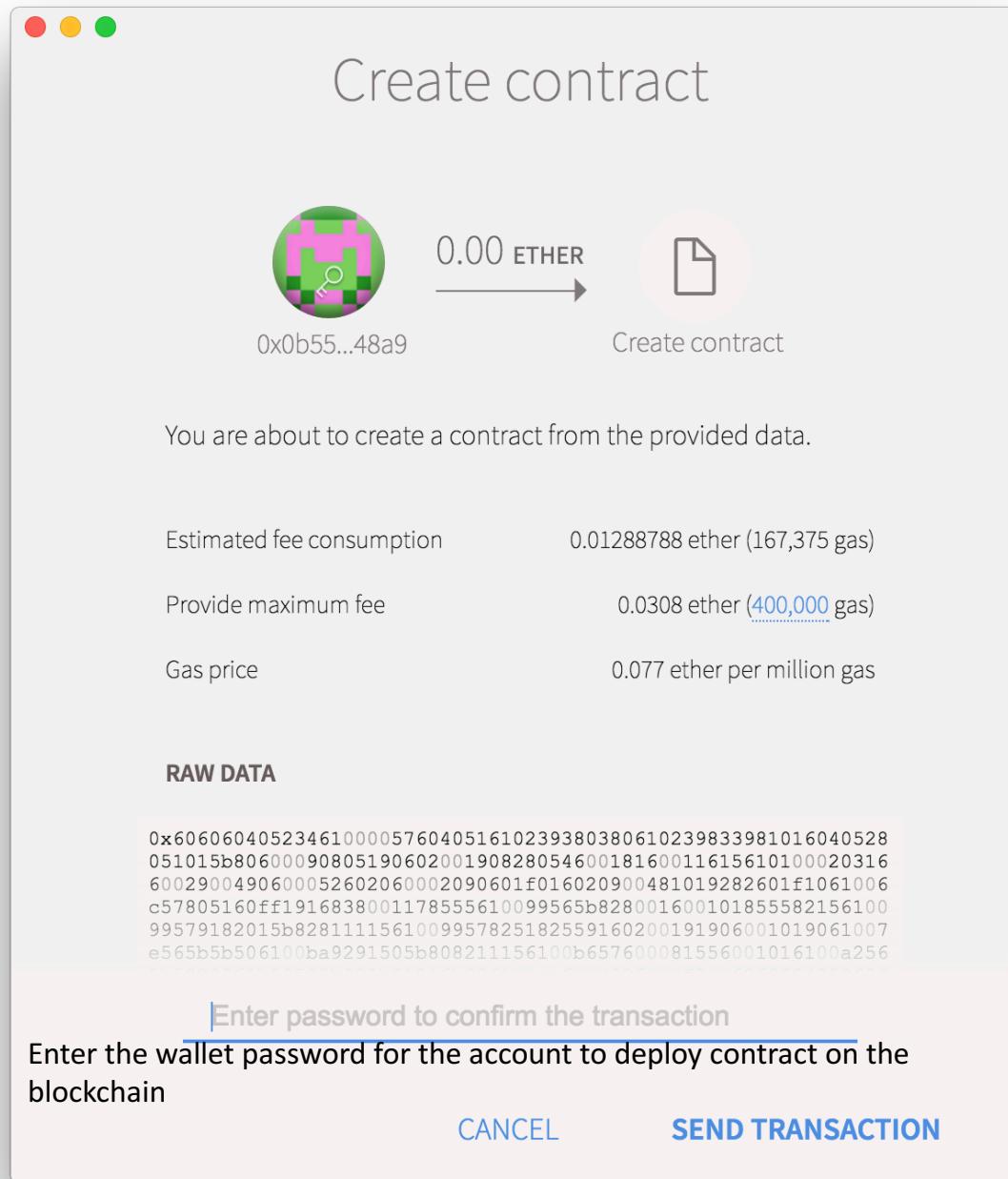
CHEAPER

FASTER

This is the most amount of money that might be used to process this transaction. Your transaction will be mined **usually within a minute**.

TOTAL

0.006 ETHER



Welcome to the future!

The “Sign Here” contract is now an immutable part of the blockchain

*Because this is the Testnet, the contract will be destroyed automatically

Condensed Course: Coding Exercise Blockchain Issuance of Stock and Transfer of Stock

- XYZ Corp. wants to use blockchain to issue and transfer its stock.
- This a verified, reliable and transparent method for tracking share ownership.

Applicable code:

```
pragma solidity ^0.4.8;
contract XYZCorpStock {
    string public standard = 'Token 0.1';
    string public name;
    string public symbol;
    uint8 public decimals;
    uint256 public totalSupply;
    mapping (address => uint256) public balanceOf;

    event Transfer(address indexed from, address indexed to, uint256 value);
    function XYZCorpStock(
        uint256 initialSupply,
        string tokenName,
        uint8 decimalUnits,
        string tokenSymbol
    ) {
        balanceOf[msg.sender] = initialSupply;
        totalSupply = initialSupply;
        name = tokenName;
        symbol = tokenSymbol;
        decimals = decimalUnits;
    }
    function transfer(address _to, uint256 _value) {
        if (balanceOf[msg.sender] < _value) throw;
        if (balanceOf[_to] + _value < balanceOf[_to]) throw;
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;
        Transfer(msg.sender, _to, _value);
    }
}
```

Code with Annotation

- pragma solidity ^0.4.8;
- contract XYZCorpStock {
 - /* Public variables of the token */
 - string public standard = 'Token 0.1';
 - string public name;
 - string public symbol;
 - uint8 public decimals;
 - uint256 public totalSupply;
 - - /* This creates an array with all balances */
 - mapping (address => uint256) public balanceOf;
 - - /* This generates a public event on the blockchain that will notify clients */
 - event Transfer(address indexed from, address indexed to, uint256 value);

Code with Annotation

```
• /* Initializes contract with initial supply tokens to the creator of the contract */
• function XYZCorpStock(
•     uint256 initialSupply,
•     string tokenName,
•     uint8 decimalUnits,
•     string tokenSymbol
• ) {
•     balanceOf[msg.sender] = initialSupply;          // Give the creator all initial tokens
•     totalSupply = initialSupply;                    // Update total supply
•     name = tokenName;                            // Set the name for display purposes
•     symbol = tokenSymbol;                        // Set the symbol for display purposes
•     decimals = decimalUnits;                     // Amount of decimals for display purposes
• }
• /* Send coins */
• 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
•     Transfer(msg.sender, _to, _value);             // Notify anyone listening that this transfer took place
• }
```

The screenshot shows the MetaMask interface on a Mac OS X desktop. The top navigation bar includes 'WALLETS' (with three colored dots), 'SEND', 'TEST-NET' (highlighted in red), 'CONTRACTS' (with 0.00 ETHER), and a 'Send everything' button.

You want to send **0 ETHER**.

SOLIDITY CONTRACT SOURCE CODE

```
5  string public symbol;
6  uint8 public decimals;
7  uint256 public totalSupply;
8
9  mapping (address => uint256) public balanceOf;
10
11 event Transfer(address indexed from, address indexed to, uint256 value);
12
13 function XYZCorpStock(
14     uint256 initialSupply,
15     string tokenName,
16     uint8 decimalUnits,
17     string tokenSymbol
18 ) {
19     balanceOf[msg.sender] = initialSupply;
20     totalSupply = initialSupply;
21     name = tokenName;
22     symbol = tokenSymbol;
23     decimals = decimalUnits;
24 }
25
26 function transfer(address _to, uint256 _value) {
27     if (balanceOf[msg.sender] < _value) throw;
28     if (balanceOf[_to] + _value < balanceOf[_to]) throw;
29     balanceOf[msg.sender] -= _value;
30     balanceOf[_to] += _value;
31     Transfer(msg.sender, _to, _value);
32 }
33
34 }
```

CONTRACT BYTE CODE

SELECT CONTRACT TO DEPLOY

XYZCorp Stock

CONSTRUCTOR PARAMETERS

Initial supply - 256 bits unsigned integer
10000

Token name - string
XYZPreferred

Decimal units - 8 bits unsigned integer
2

Token symbol - string
XYZ

Defining Variables for the Issuance of Stock

CONSTRUCTOR PARAMETERS

Initial supply - 256 bits unsigned integer

10000

= # of shares

Token name - string

XYZPreferred

= name/category of shares

Decimal units - 8 bits unsigned integer

2

= size of fractional shares

Token symbol - string

XYZ

= symbol/short name for the shares

```
51     transfer(msg.sender, _to, _value);  
52 }  
53  
54 }
```

XYZ

SELECT FEE

0.006 ETHER



This is the most amount of money that might be used to process this transaction. Your transaction will be mined **usually within a minute**.

CHEAPER

FASTER

TOTAL

0.006 ETHER

DEPLOY

Select “Deploy”, and the stock is “issued”.

Let's transfer the stock to John Shareholder

The screenshot shows the MetaMask wallet interface with the "CONTRACTS" tab selected. At the top, there are three colored dots (red, yellow, green), followed by "WALLETS", "SEND", and "TEST-NET". Below the network status, it shows "25 peers | 852,810 20s since last block". The "CONTRACTS" tab is active, and the balance is "4.74 ETHER".

Contracts

DEPLOY NEW CONTRACT

CUSTOM CONTRACTS

In order to watch and interact with a contract already deployed on the blockchain you need to know its address and description of its interface in JSON format.

 XYZPREFERRED (ADMIN PAGE)
0.00 ether
0x08713428b5A0cC00941Af5F133c9123882F63D66

WATCH CONTRACT

CUSTOM TOKENS

Go back to the “Contracts” tab to interact with the stock issuance contract we just created.

WALLETS SEND TEST-NET 25 peers | 852,811 19s since last block CONTRACTS 4.74 ETHER

XYZPREFERRED (ADMIN PAGE)

Name: XYZPreferred

Total supply: 10000

Decimals: 2

Standard: XYZCorpStock

Balance of Address: 0x123456...

Select function: Pick A Function

WALLETS SEND TEST-NET 25 peers | 852,814 21s since last block CONTRACTS 4.74 ETHER

XYZPREFERRED (ADMIN PAGE)

READ FROM CONTRACT

Name: XYZPreferred

Total supply: 10000

Decimals: 2

Standard: XYZCorpStock

Balance of Address:

WRITE TO CONTRACT

Select function: Transfer

to - address: 0xA36aeBF10fBD25311F39E8Ac7e

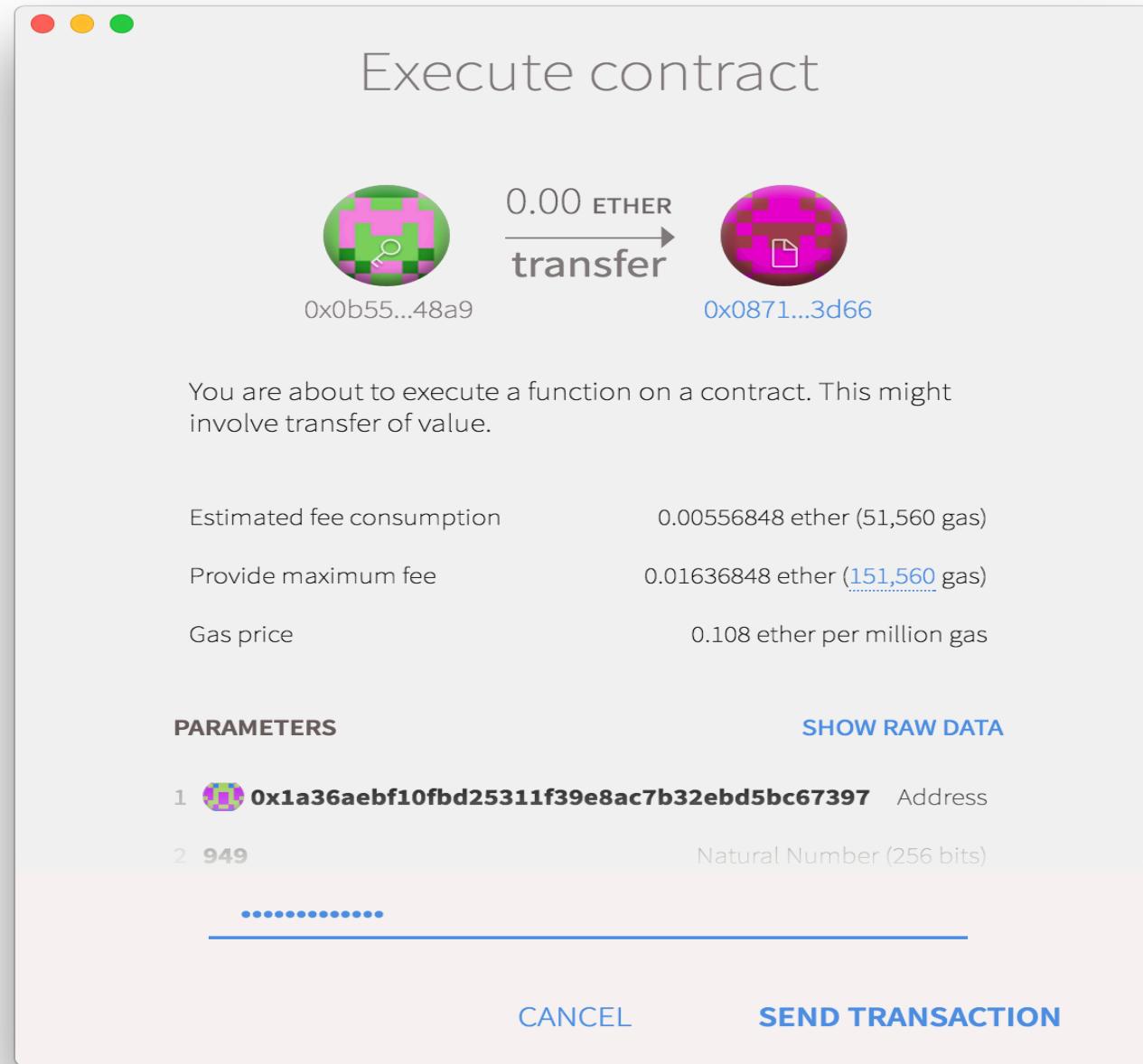
value - 256 bits unsigned integer: 949

Execute from: XYZ Corp. - 1.74 ETHER

Send ETHER: 0

EXECUTE

The stock issue contract only has one function: “Transfer”.
Let’s transfer the stock to John Shareholder.
In the “to-address field, type JohnShareholder’s wallet address. Click Execute.



Contract executes with XYZ Corp's authorization (password).

XYZPreferred - Token transfer



XYZ Corp.



→ Johnny Shareholder

2 of 12 Confirmations

9.49 XYZ



John Shareholder now has 10 shares of XYZ Corp.
Recall that the decimal was set to the hundredth place.
The ~.5 shares missing was paid to miners for verifying the
contract/transfer.

And that's how to issue and transfer stock/money/anything imaginable.

Thank You

Join the Conversation!



wulfkaal@me.com



Wulf Kaal



Wulf Kaal



@wulfkaal



medium.com/@wulfkaal



Facebook.com/wulf.kaal



wulfkaal.com



wulfkaal