The Blockchain:

What It is & Why It Matters

Abhishek Dubey
Anastasia Mavridou
Douglas C. Schmidt

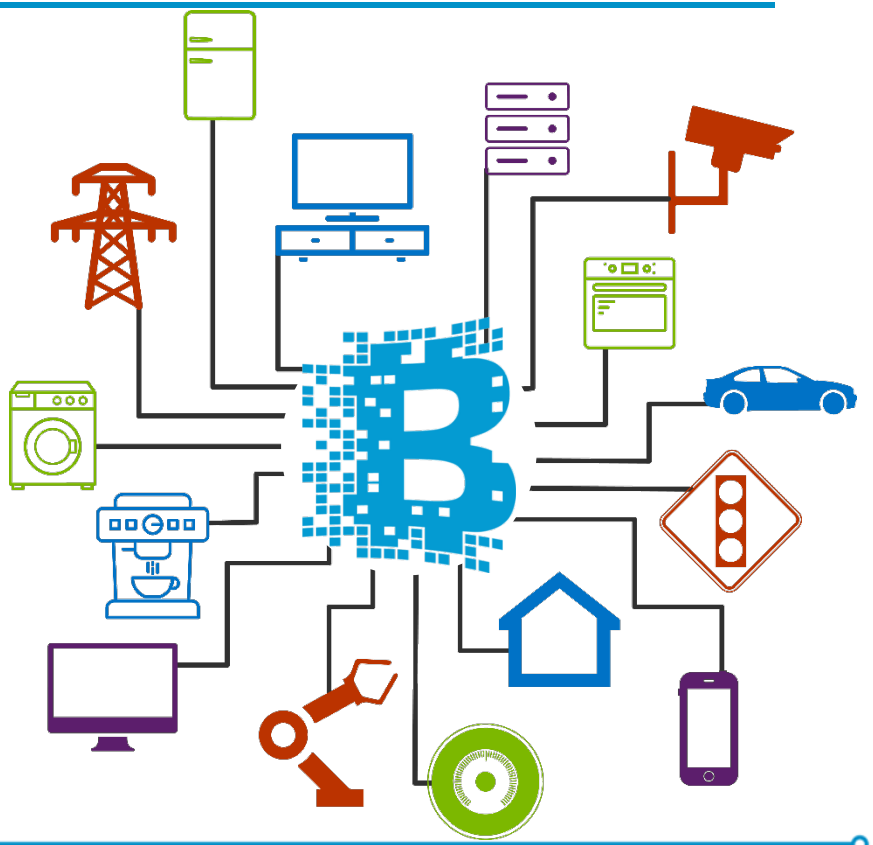
This work has been funded in part by Siemens, Varian, & Accenture

Institute for Software Integrated Systems
World-class, interdisciplinary research with global impact.

VANDERBILT UNIVERSITY

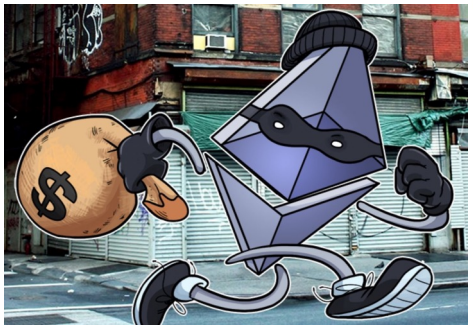
Overview of the Presentation

- Provide an introduction to the blockchain & why it matters to the community



Overview of the Presentation

- Provide an introduction to the blockchain & why it matters to the middleware IOT community
- Explore challenges to applying blockchain for various domains, including IOT & beyond



What is a Blockchain?

What is a Blockchain?

- A blockchain is a decentralized platform that supports “trustless” transactions

See en.wikipedia.org/wiki/Blockchain

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

Used as computational substrate for "crypto-currencies", plus **more**

What is a Blockchain?

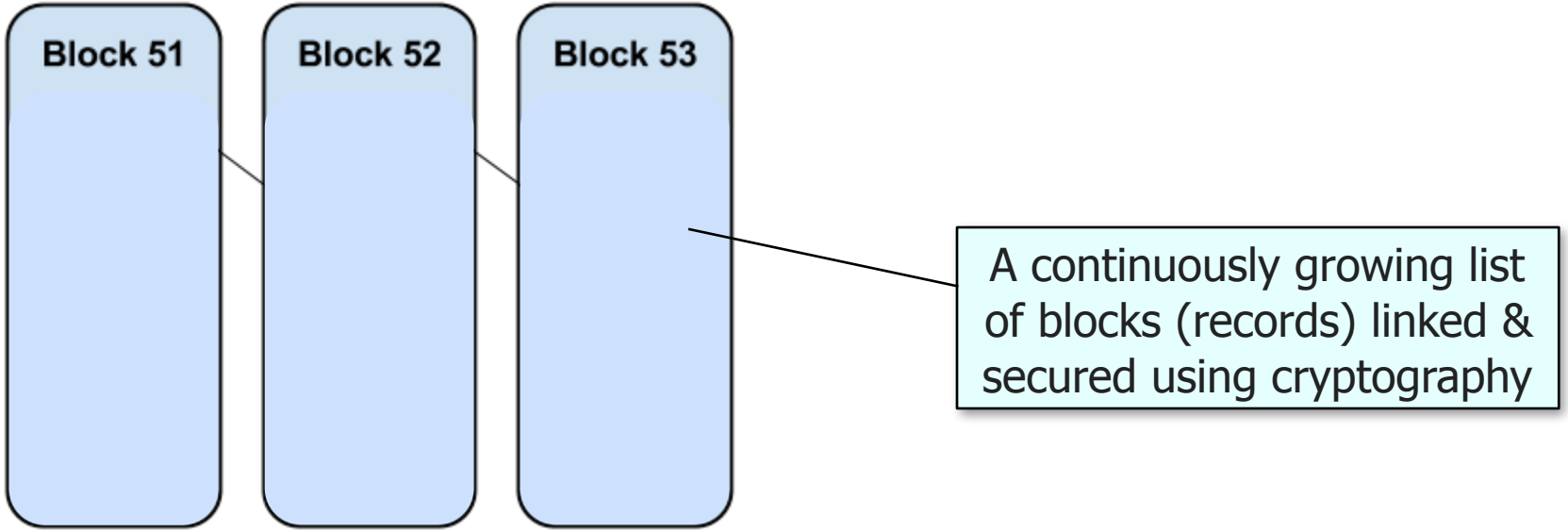
- A blockchain is a decentralized platform that supports "trustless" transactions

Used as computational substrate for "crypto-currencies", plus **more**

A cryptocurrency is a digital asset that uses cryptography to secure transactions, control the creation of additional units, & verify asset transfer

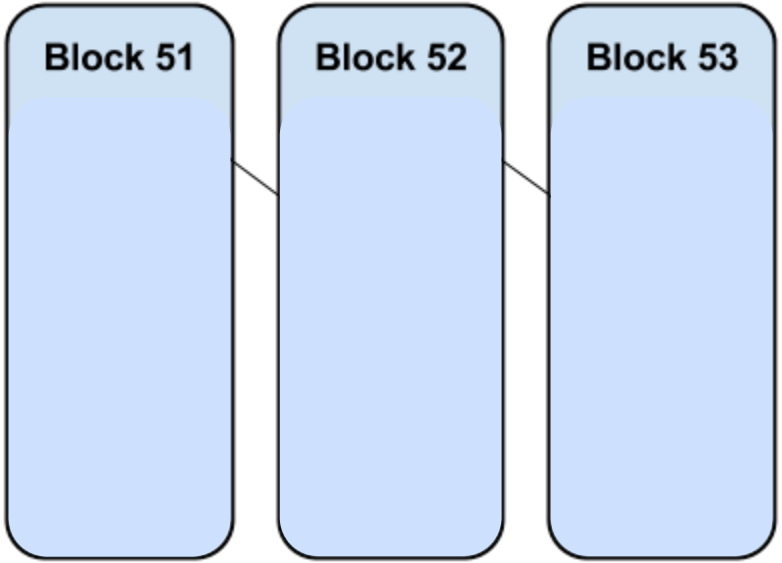
What is a Blockchain?

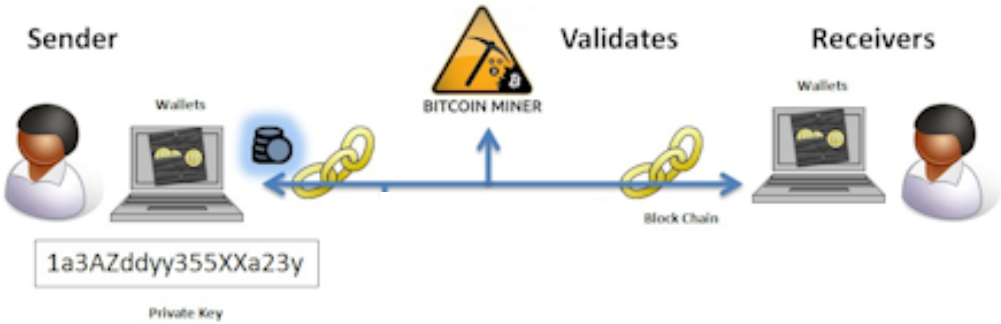
- A blockchain is a decentralized platform that supports "trustless" transactions



What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

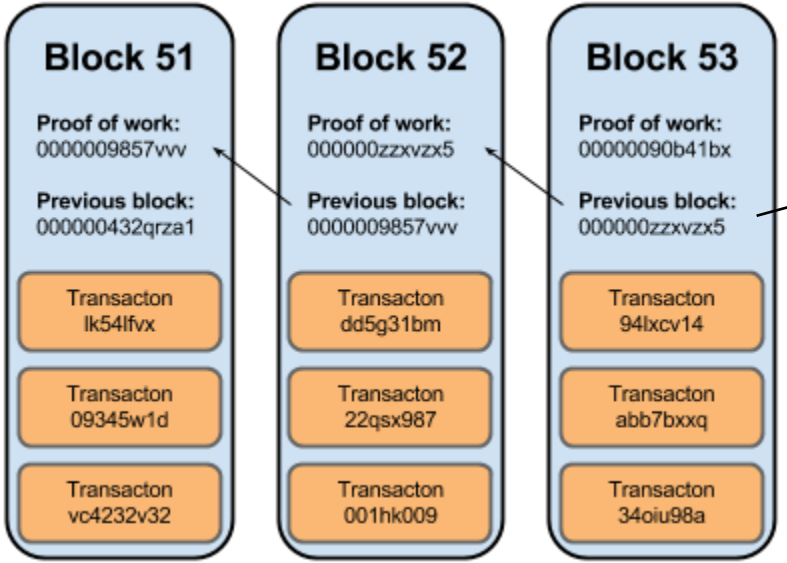




This chain of blocks provides an open, distributed ledger that immutably records transactions between two parties efficiently & verifiably

What is a Blockchain?

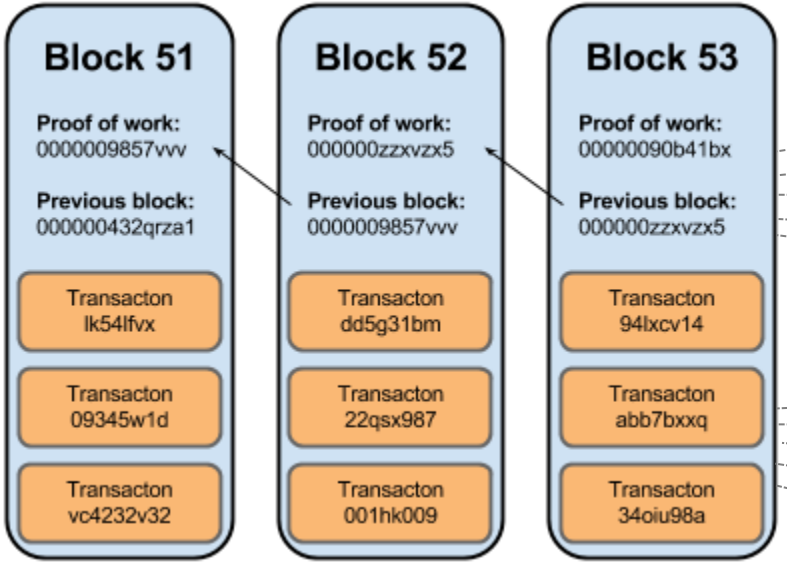
- A blockchain is a decentralized platform that supports "trustless" transactions

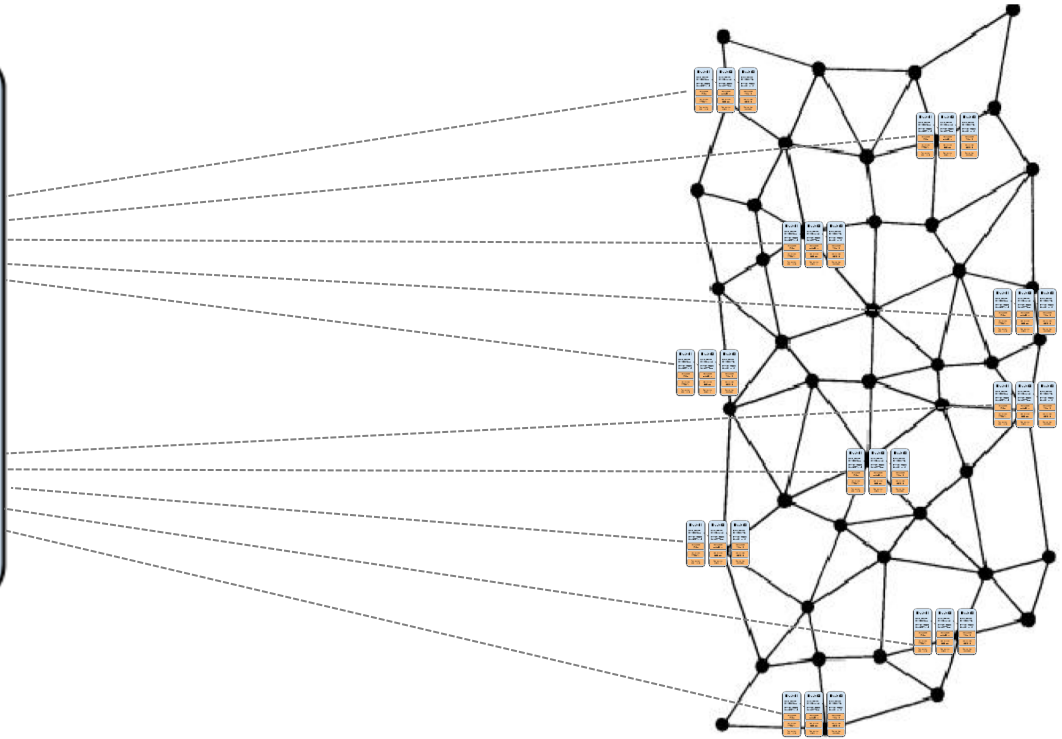


A block contains transaction data, a timestamp, & a hash pointer that links to the previous block (which forms the "chain" of blocks)

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions



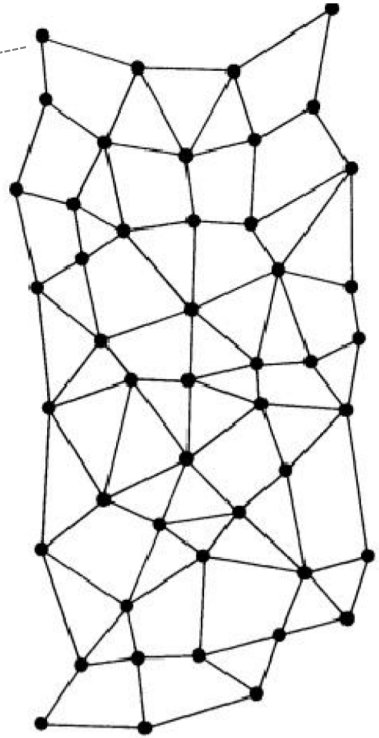


These blocks are replicated across many computers, rather than being stored on a central server

What is a Blockchain?

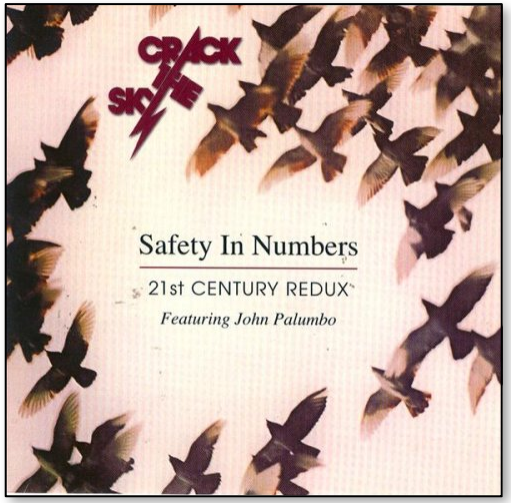
- A blockchain is a decentralized platform that supports "trustless" transactions

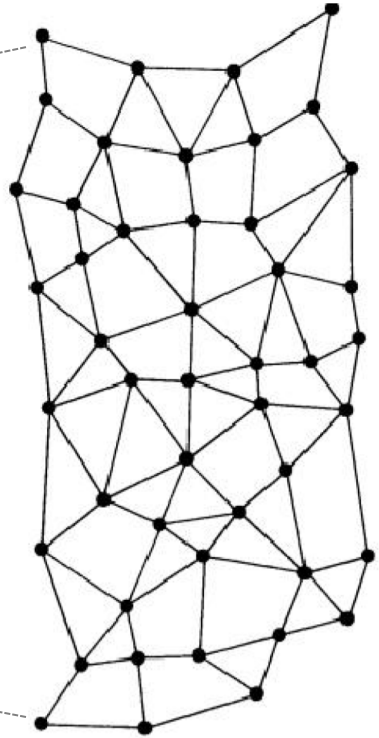
This platform may be distributed globally (public blockchain)



What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions



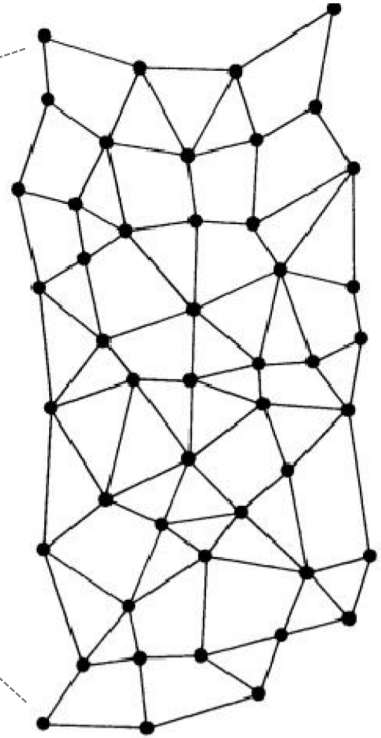


Public blockchains are less efficient, but most useful when not all participants can be trusted to behave

What is a Blockchain?

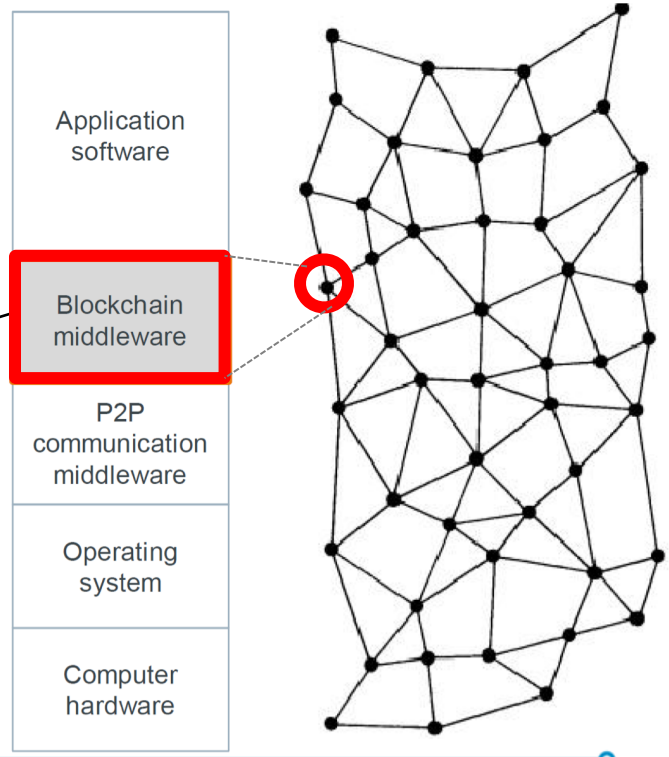
- A blockchain is a decentralized platform that supports "trustless" transactions

It could be localized to a limited group (private blockchain)



What is a Blockchain?

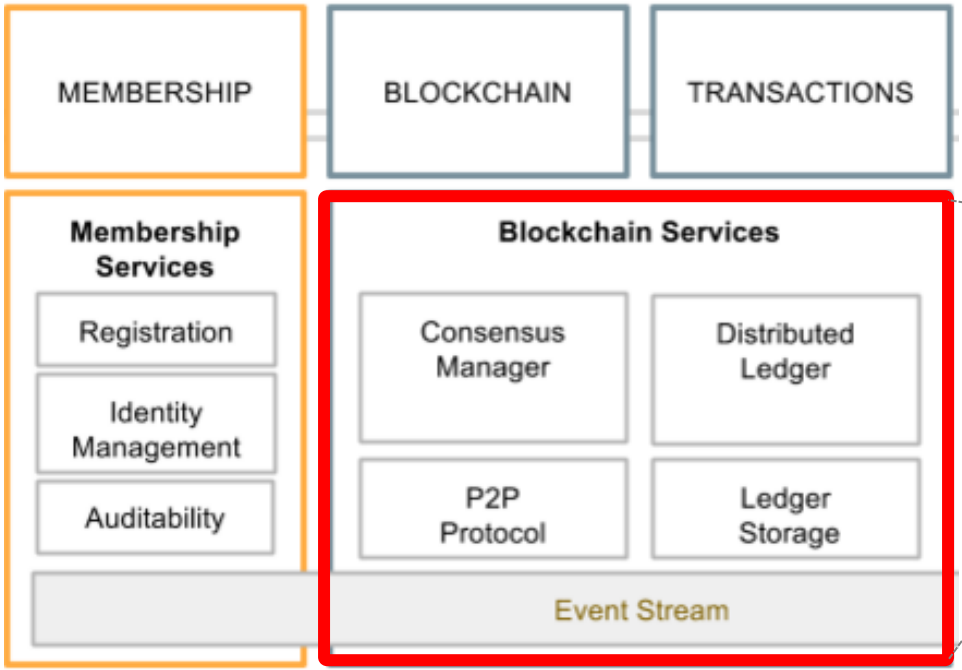
- A blockchain is a decentralized platform that supports "trustless" transactions

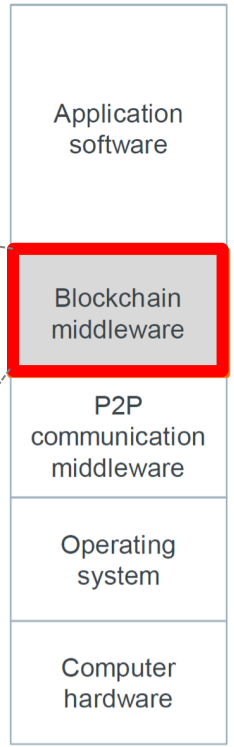


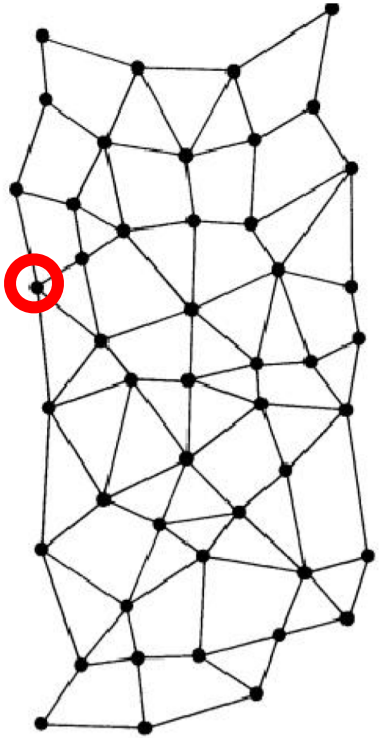
Each node in a blockchain network runs common middleware & all nodes have equal level of privilege & access

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions





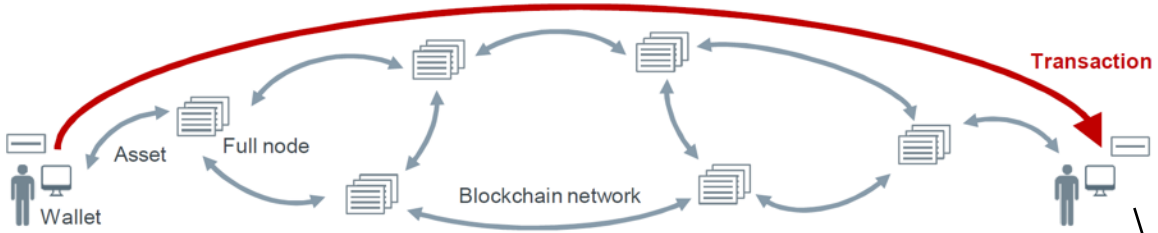


Blockchain middleware provides services to applications beyond what's provided by the OS & communication protocols

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

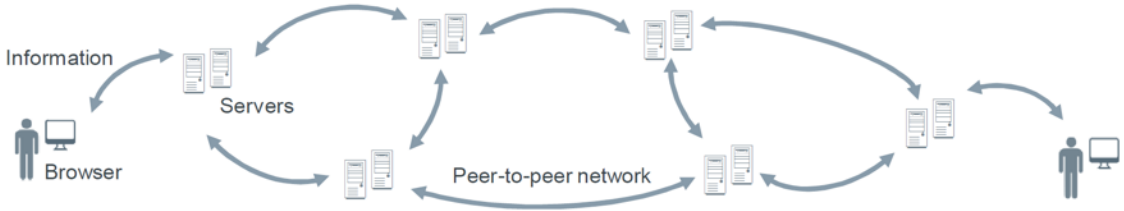
DIGITAL ASSET TRANSACTION
Record & transfer



- **Blockchain** protocol
- Immutable replicated database
- Identity, reputation

ON TOP OF

TRADITIONAL INTERNET
Store & copy



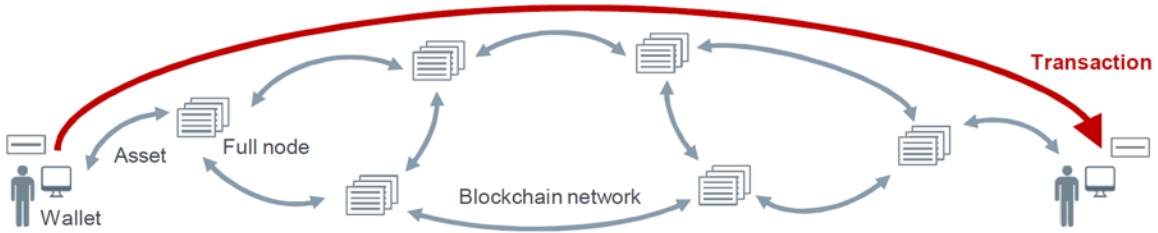
- Communication
- Storage
- Computation

Blockchain middleware enables anonymous exchange of digital assets without the need for a central authority to verify trust & transfer of value

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

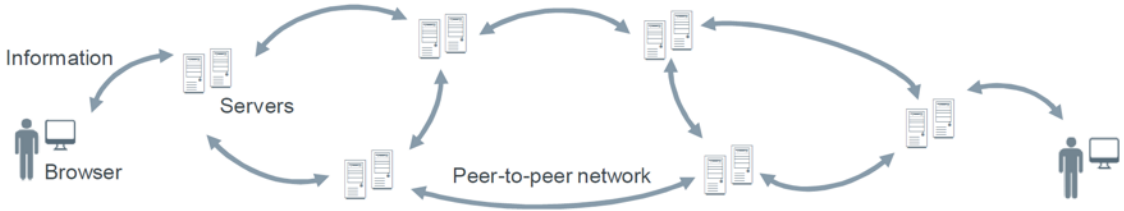
DIGITAL ASSET TRANSACTION
Record & transfer



- **Blockchain** protocol
- Immutable replicated database
- Identity, reputation

ON TOP OF

TRADITIONAL INTERNET
Store & copy

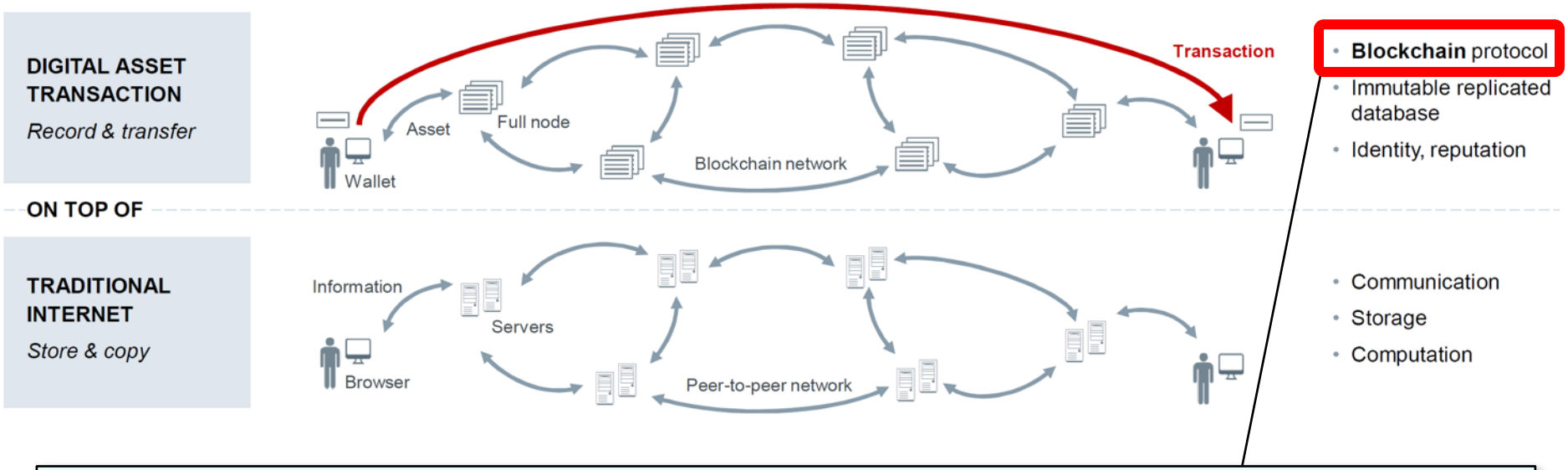


- Communication
- Storage
- Computation

Key characteristics of a blockchain

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

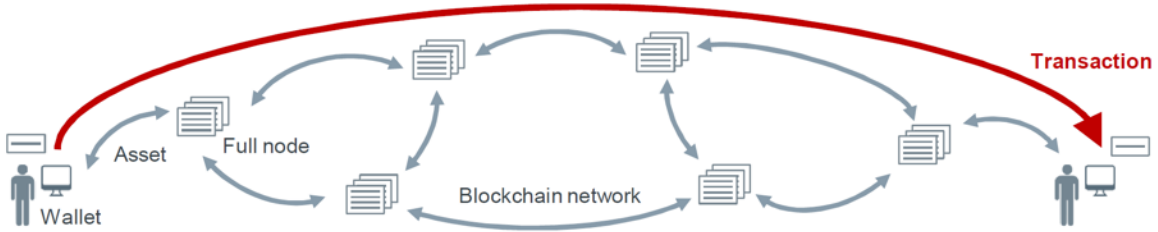


Ensures a common, unambiguous ordering of blocks & guarantees the (eventual) integrity & consistency of the blockchain across (geographically) distributed nodes

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

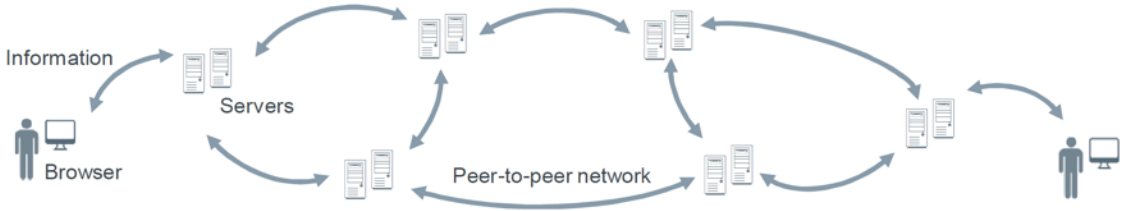
DIGITAL ASSET TRANSACTION
Record & transfer



- **Blockchain** protocol
- Immutable replicated database
- Identity, reputation

ON TOP OF

TRADITIONAL INTERNET
Store & copy



- Communication
- Storage
- Computation

A database containing immutable time-stamped information for every transaction that's replicated on servers (may be around the world)

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

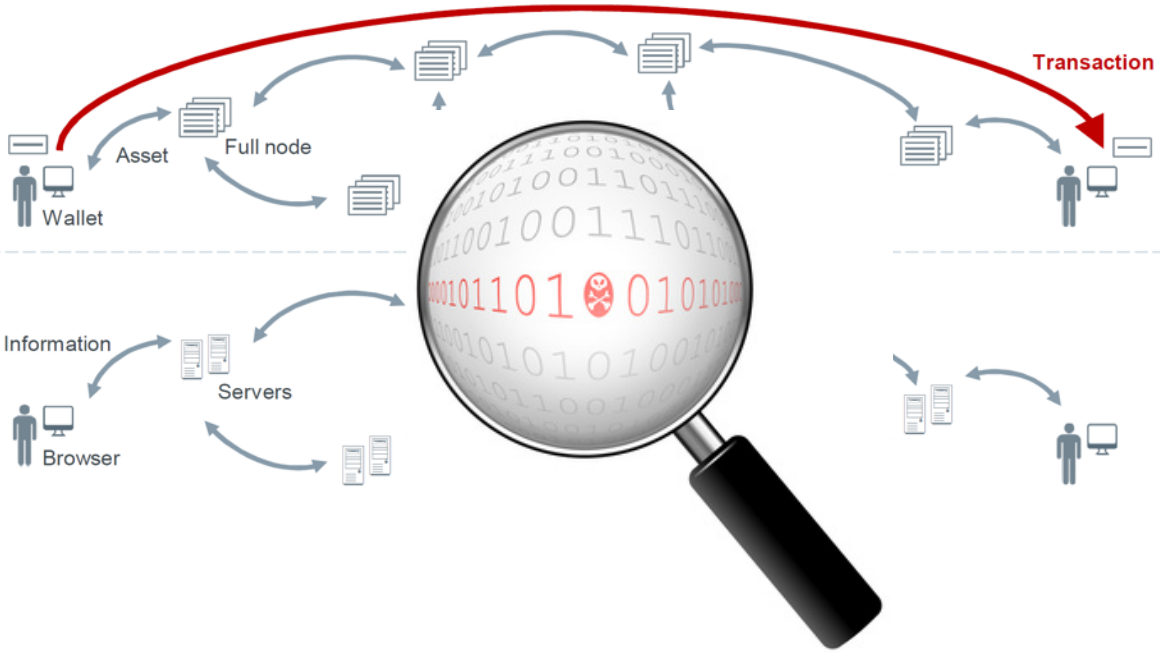
DIGITAL ASSET TRANSACTION

Record & transfer

ON TOP OF

TRADITIONAL INTERNET

Store & copy



- **Blockchain** protocol
- Immutable replicated database
- Identity, reputation

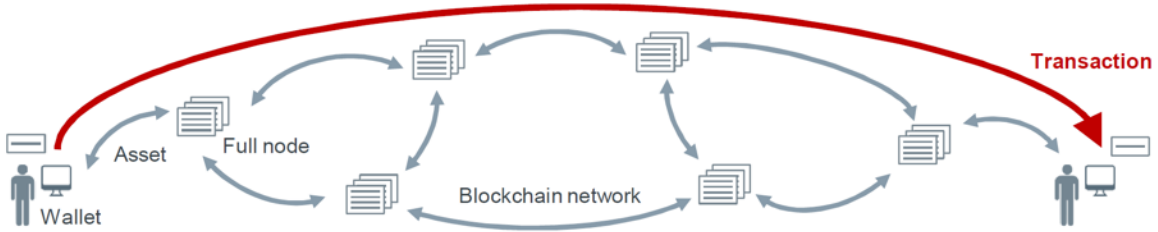
- Communication
- Storage
- Computation

It's extremely hard to change a blockchain without collusion
& it's extremely easy to detect the attempt if anyone tries

What is a Blockchain?

- A blockchain is a decentralized platform that supports "trustless" transactions

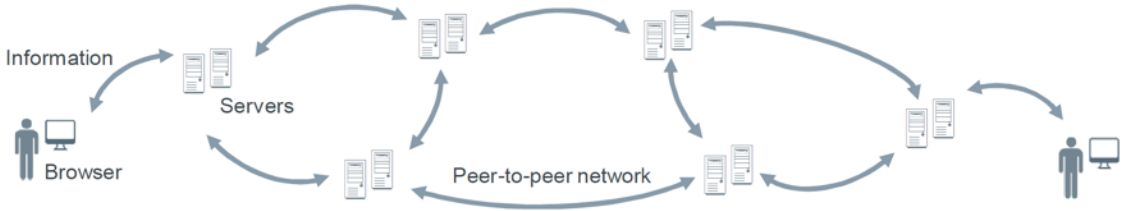
DIGITAL ASSET TRANSACTION
Record & transfer



- **Blockchain** protocol
- Immutable replicated database
- Identity, reputation

ON TOP OF

TRADITIONAL INTERNET
Store & copy



- Communication
- Storage
- Computation

Build a sense of trust among anonymous users of online communities

Why Blockchain Matters

Why Blockchain Matters

- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service"

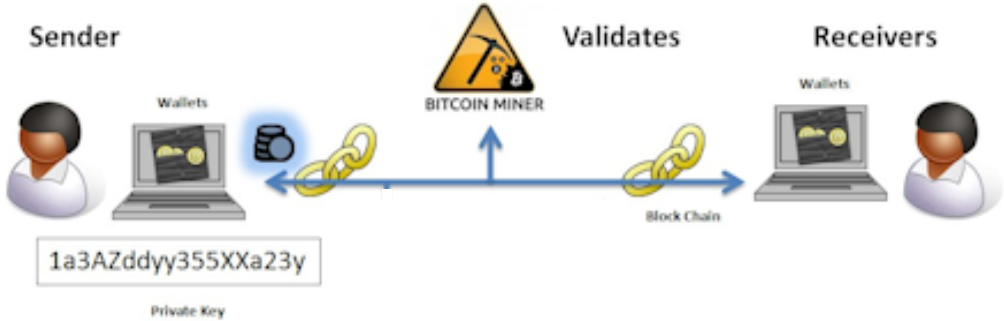
Why Blockchain Matters

- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service"

A key goal is to "disintermediate" centralized brokers, yet still allow multiple parties (who don't trust each other) to share a single database

Why Blockchain Matters

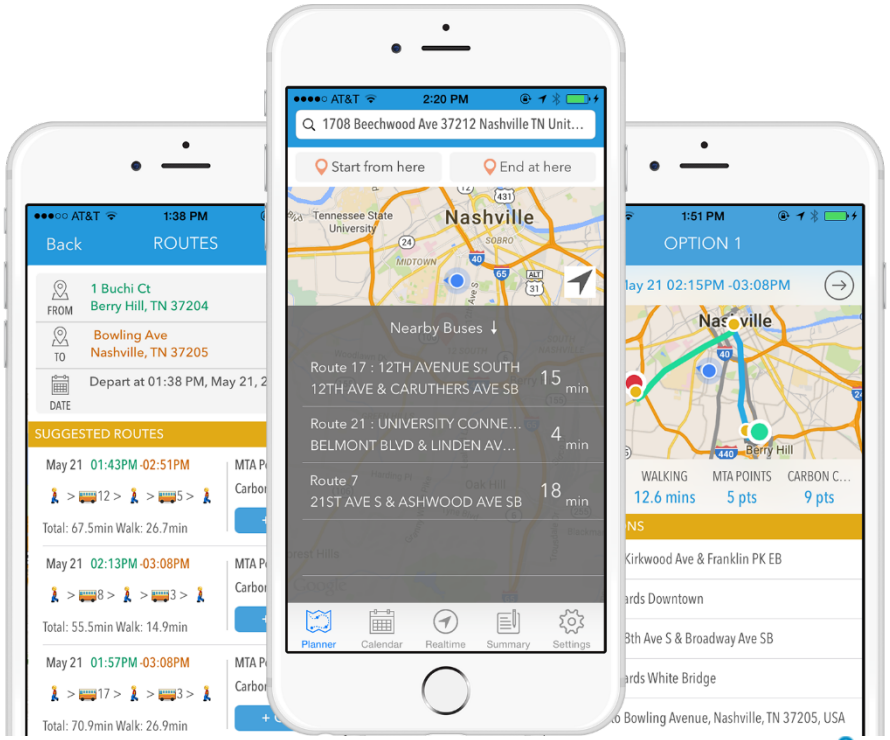
- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service", e.g.
  - Payments in the financial sector
    - e.g., use on-chain tokens to represent cash, stocks, bonds, etc



Turns out to be problematic in practice due to lack of confidentiality..

Why Blockchain Matters

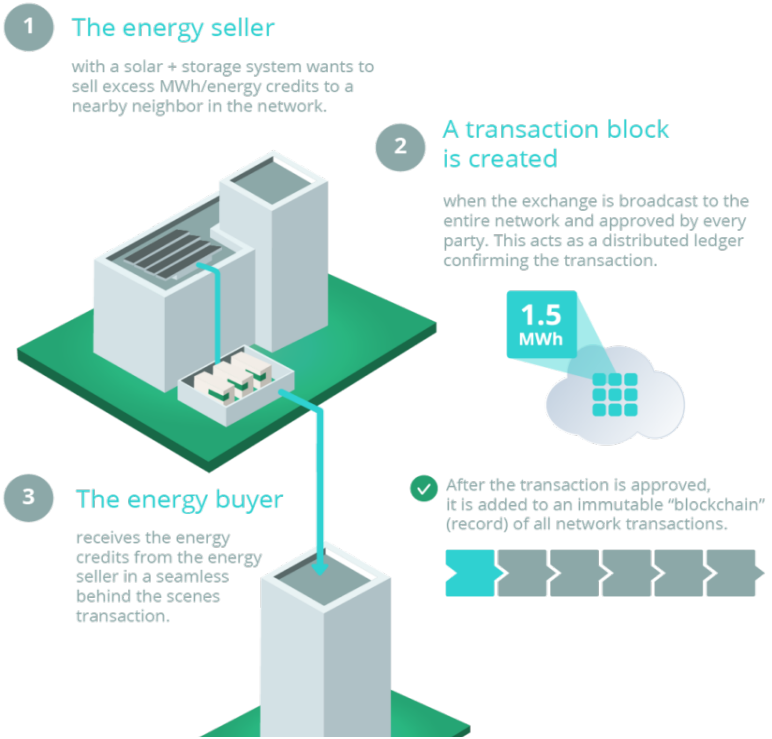
- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service", e.g.
  - Payments in the financial sector
  - Lightweight financial systems, e.g.
    - Streamline interactions between commuters & multi-modal transit



See www.vanderbilt.edu/strategicplan/undergraduate-residential-education/universitycourses-2017/smart-city-applications.php

Why Blockchain Matters

- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service", e.g.
  - Payments in the financial sector
  - Lightweight financial systems, e.g.
    - Streamline interactions between commuters & multi-modal transit
    - Improve efficiency of energy transactions in "smart grids"



See www.dre.vanderbilt.edu/~schmidt/PDF/IOT-2017.pdf

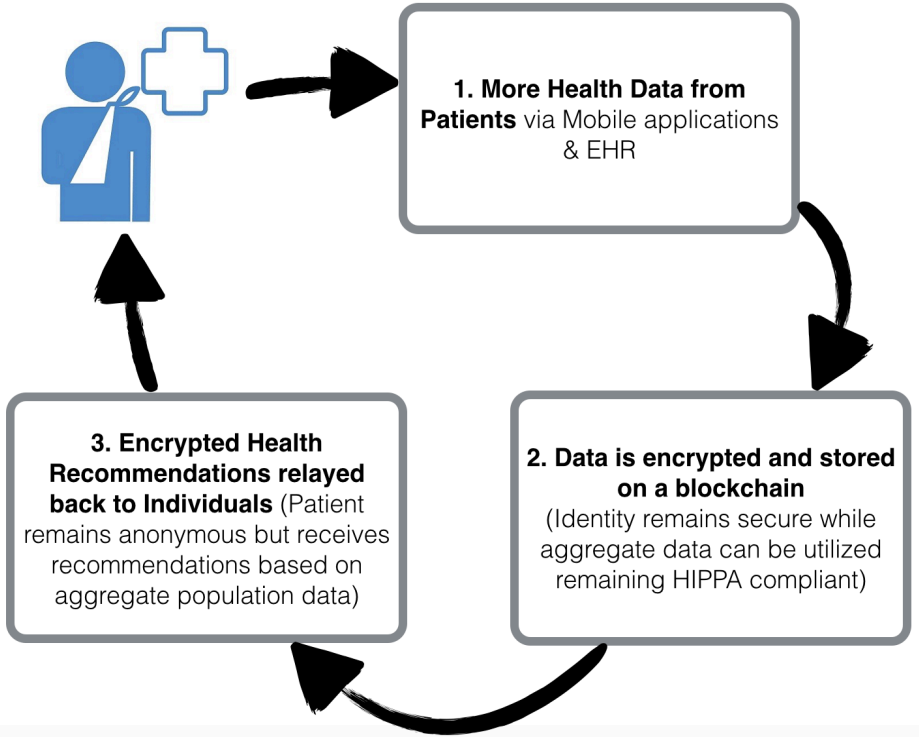
Why Blockchain Matters

- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service", e.g.
  - Payments in the financial sector
  - Lightweight financial systems, e.g.
    - Streamline interactions between commuters & multi-modal transit
    - Improve efficiency of energy transactions in "smart grids"
    - UN provides thousands of Syrian refugees in Jordan with food, clothing, & other aid in a cost effective manner

See www.coindesk.com/united-nations-sends-aid-to-10000-syrian-refugees-using-ethereum-blockchain

Why Blockchain Matters

- Blockchain helps address needs in various domains by providing decentralized "transactions-as-a-service", e.g.
  - Payments in the financial sector
  - Lightweight financial systems
  - Interorganizational record keeping
    - e.g., provide providers, patients, (& surrogates) better access to —& control over—health info

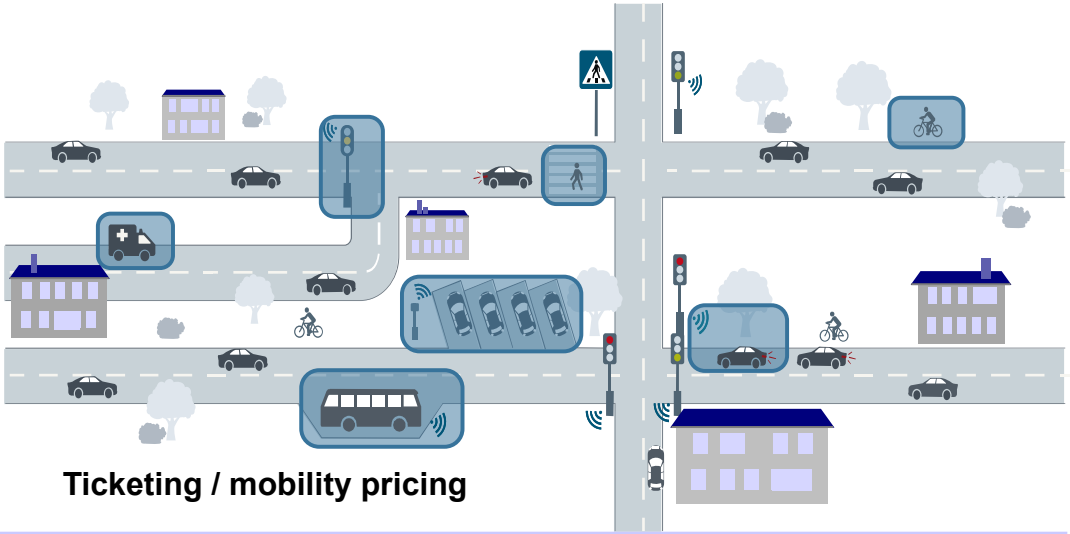


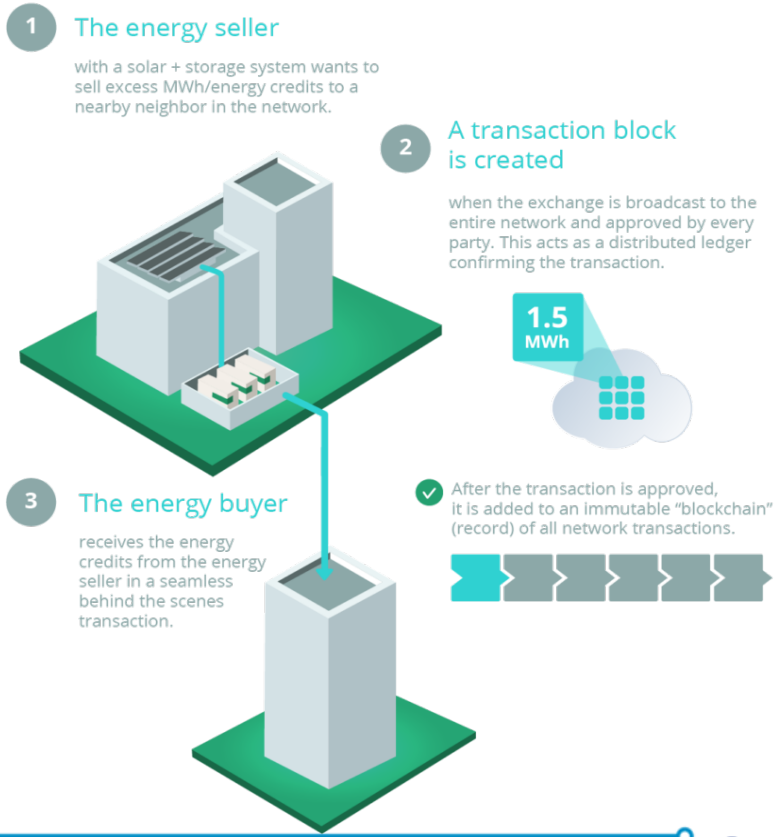
See www.dre.vanderbilt.edu/~schmidt/PDF/PLoP-2017-blockchain.pdf

Blockchains and Cyber-Physical Systems (CPS)

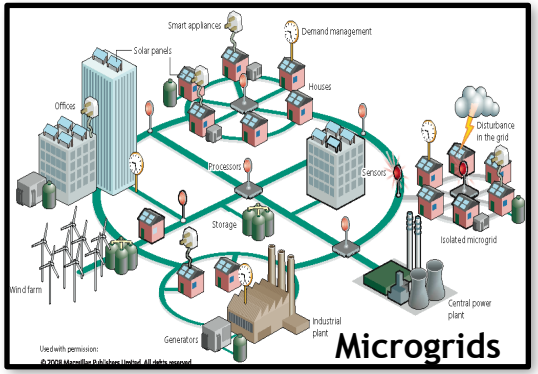
Blockchains Increasingly are Being Used in CPS

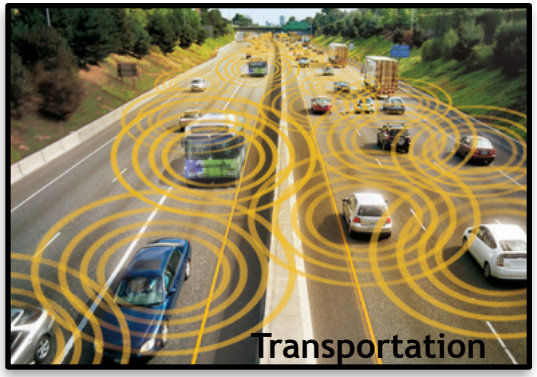
- The notion of decentralized computation & trustless computing also provides opportunities/challenges in the IoT

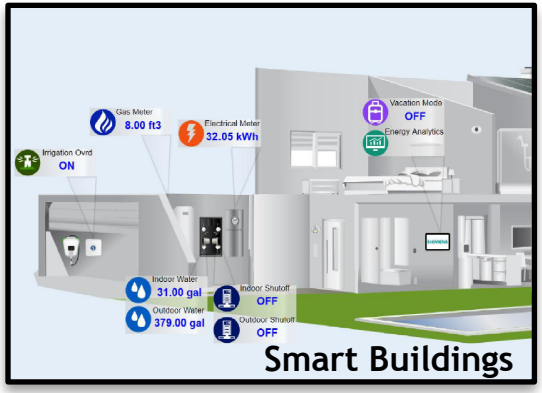




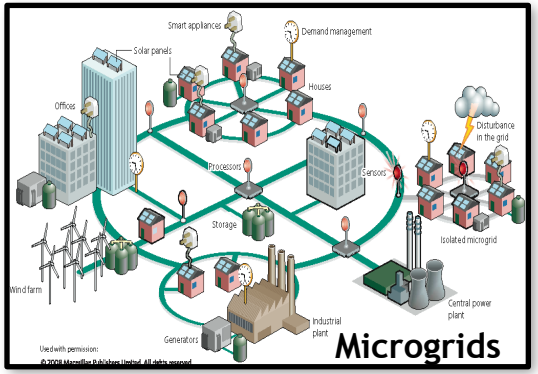
The Reason is the Focus on Decentralization

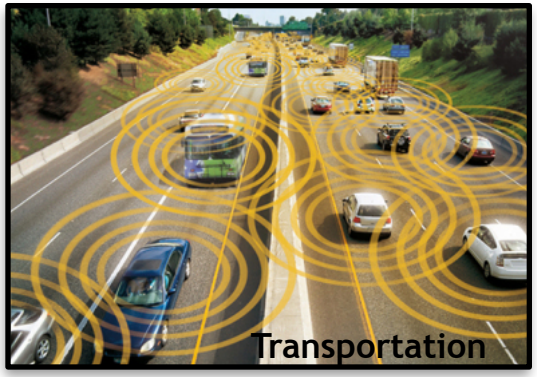


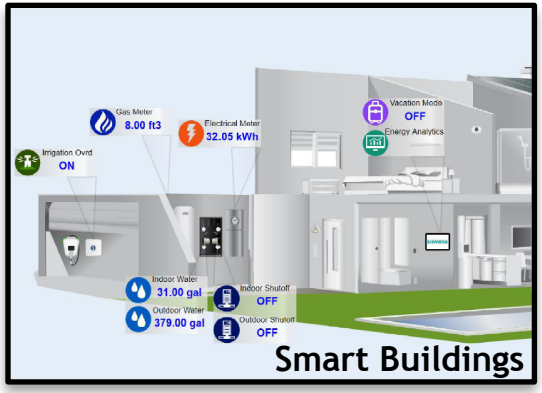




Key Requirements For Decentralized CPS



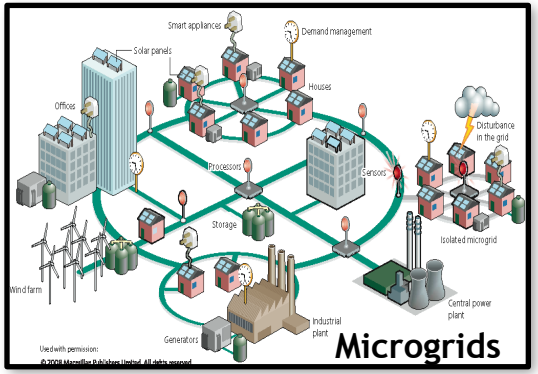


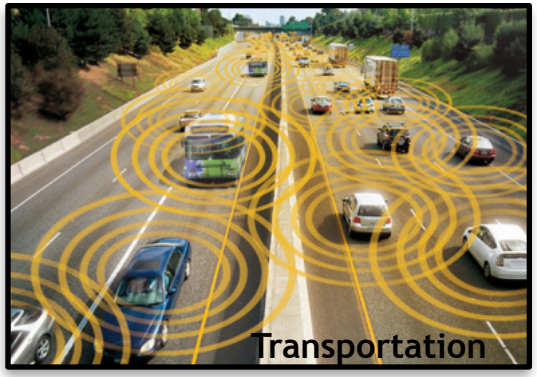


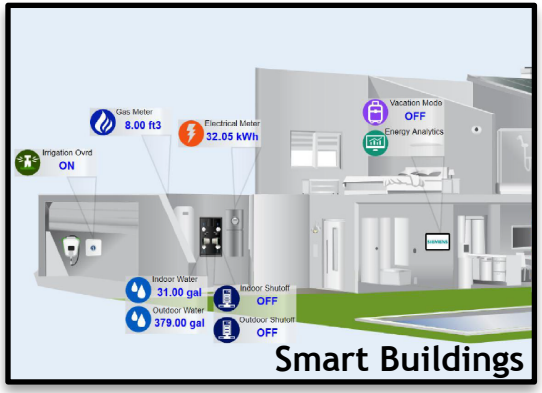
Decentralized Control

- Compute control actions using distributed averaging consensus within a specific time limit
- Requires real-time information dissemination and time synchronized task execution

Key Requirements For Decentralized CPS





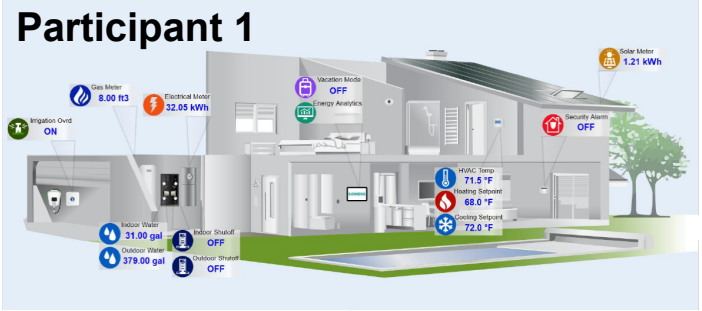


Decentralized Information

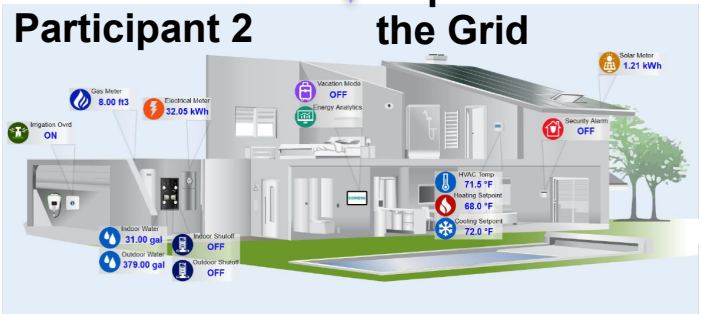
- Preserve integrity of Information across all actors in the system
- Support for information aggregation and transactions
- Requires consensus, and distributed ledger

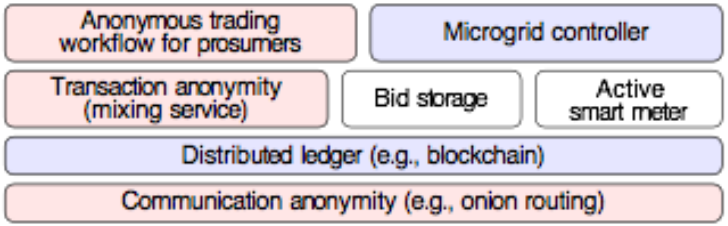
Transactive Energy Systems: An Example

Example Application: Private And Decentralized Energy Transactions

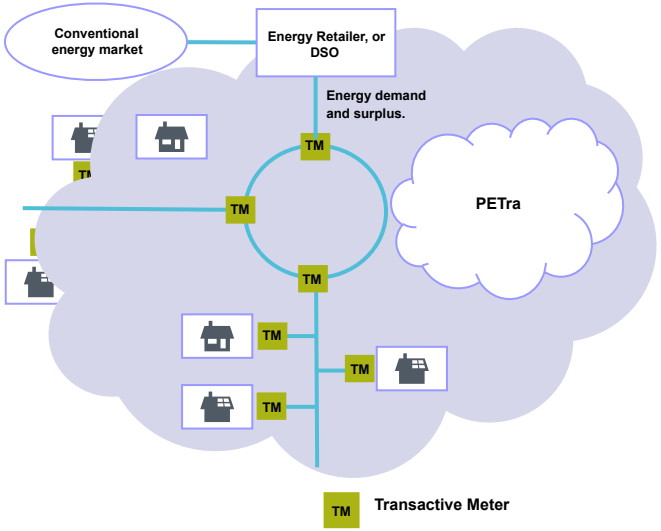


Energy Exchange Reduces Dependence on the Grid





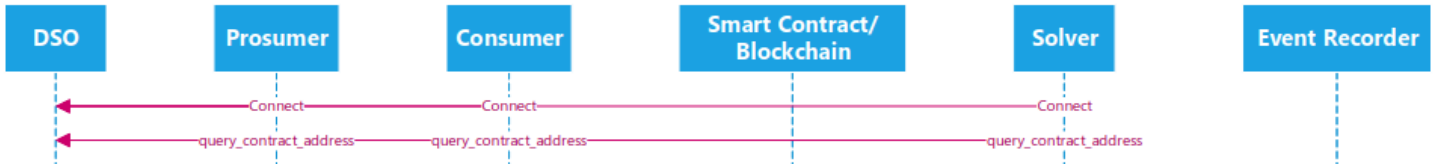
PETra Framework



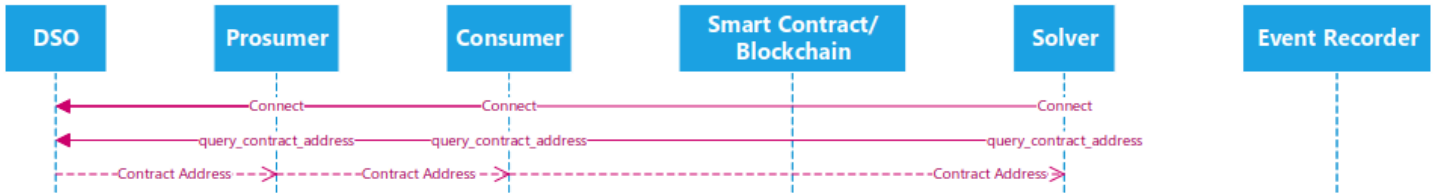
Application use case for integration of RIAPS and Blockchains

Sequence Diagram

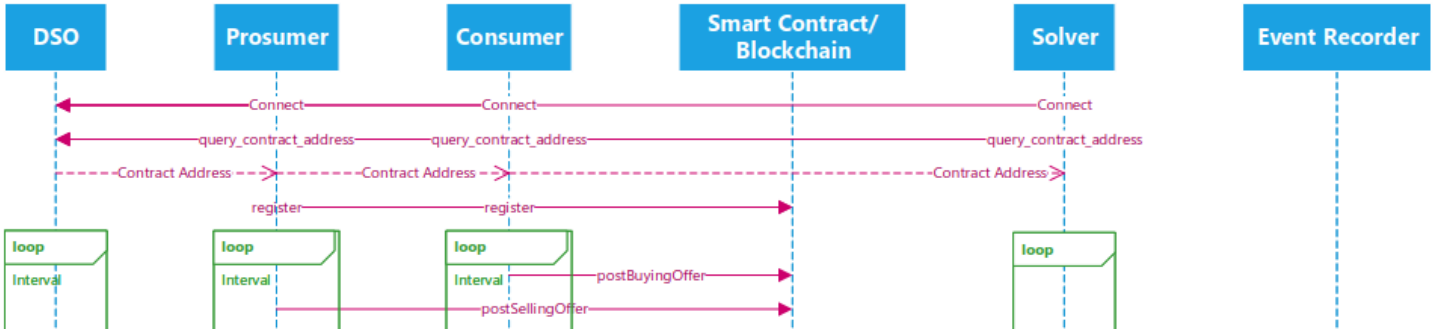
Sequence Diagram



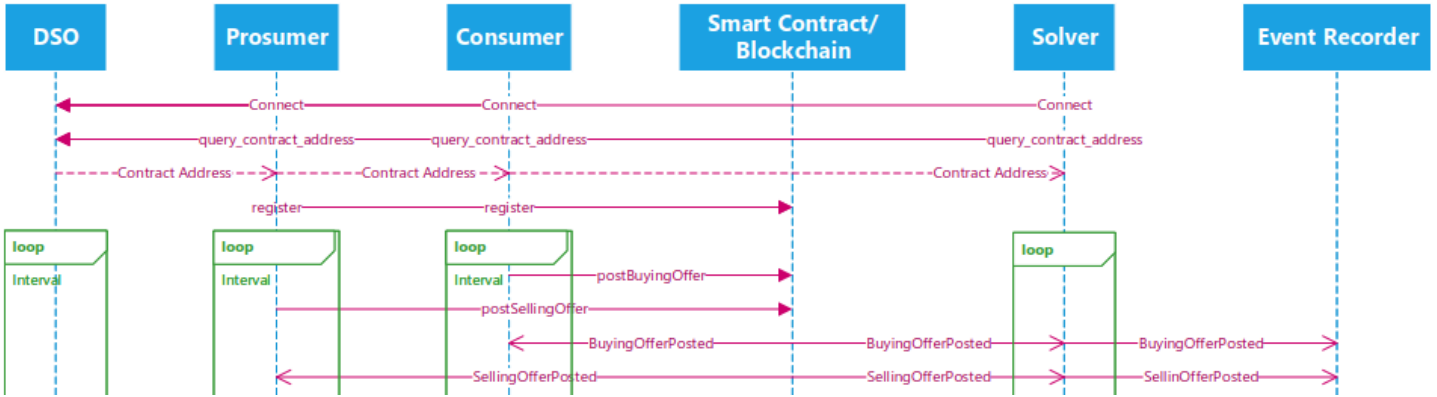
Sequence Diagram



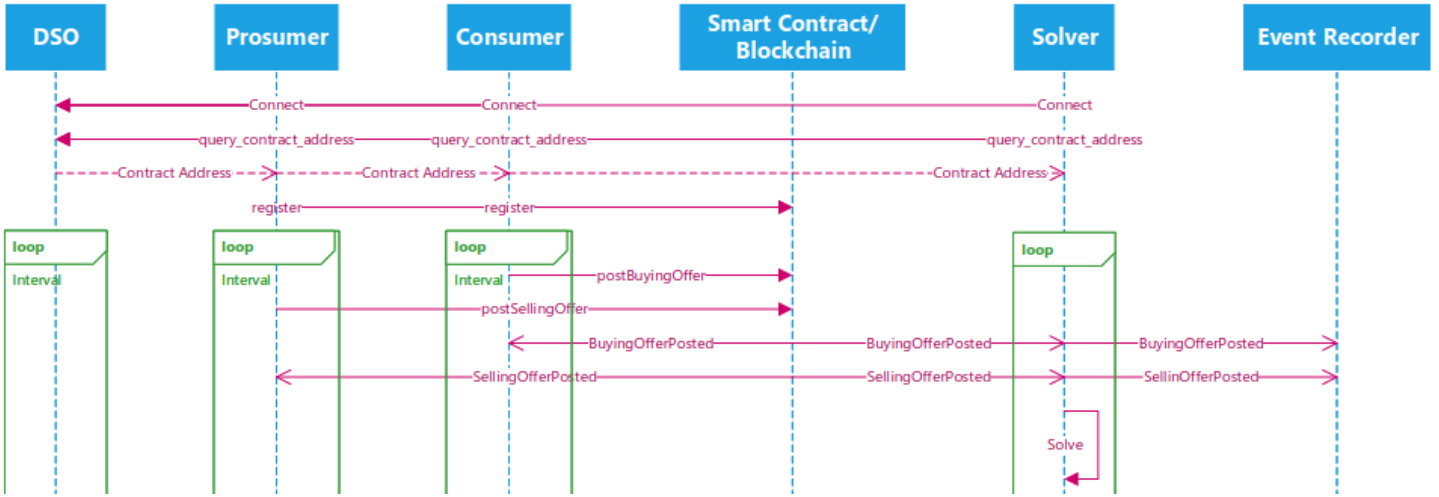
Sequence Diagram



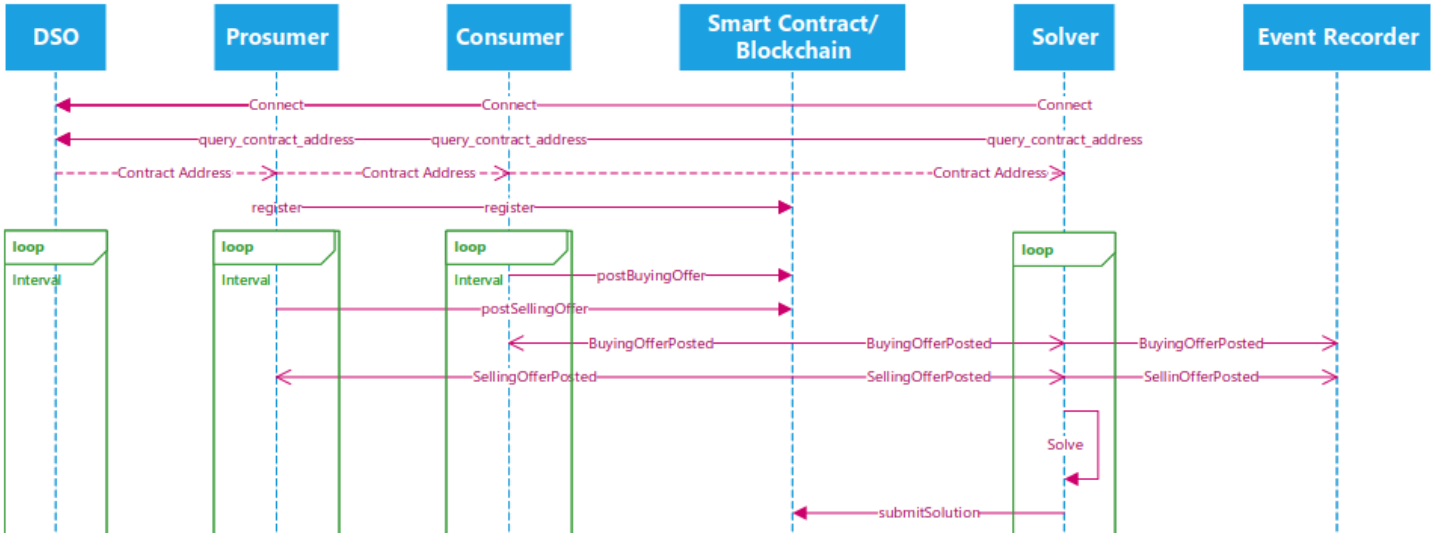
Sequence Diagram



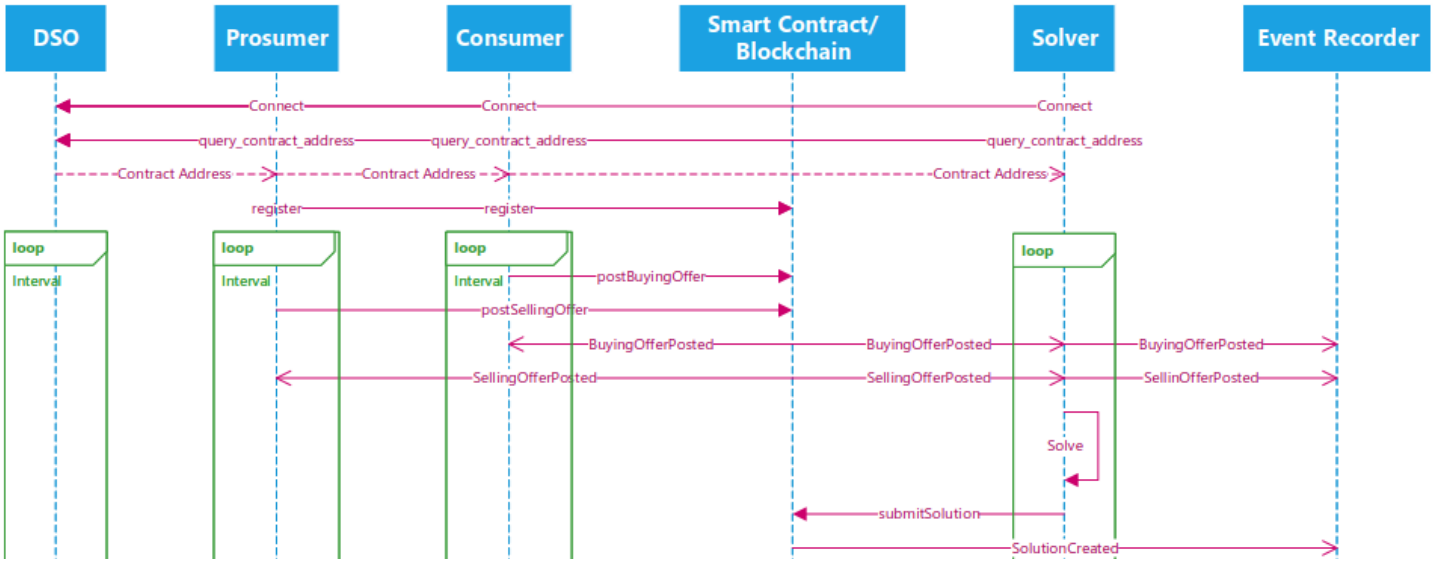
Sequence Diagram



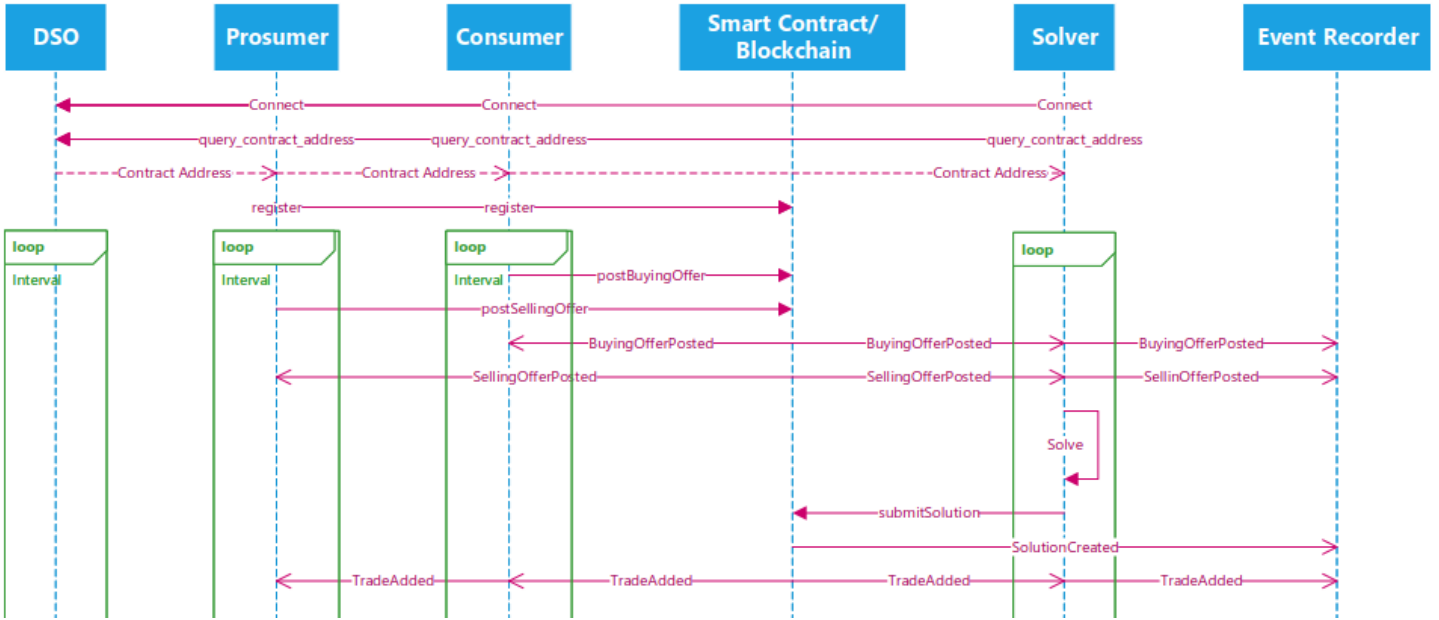
Sequence Diagram



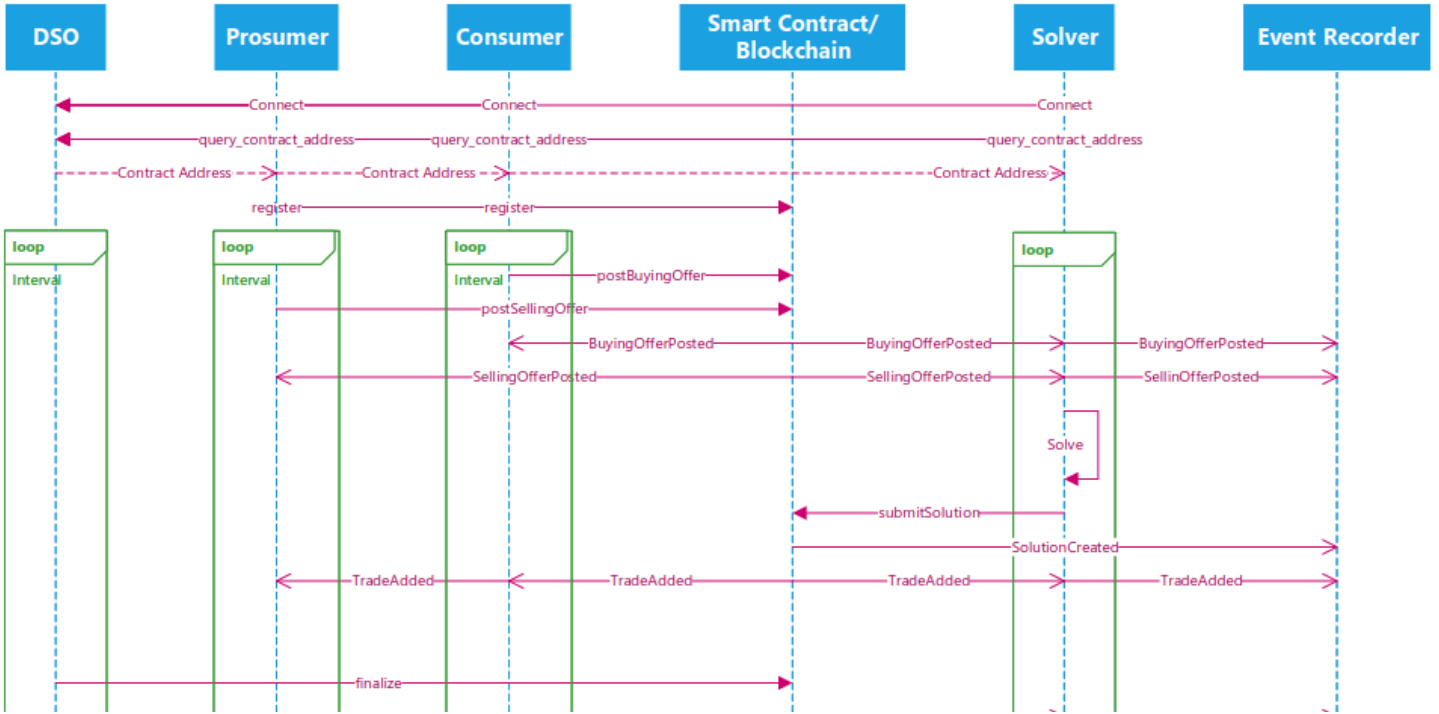
Sequence Diagram



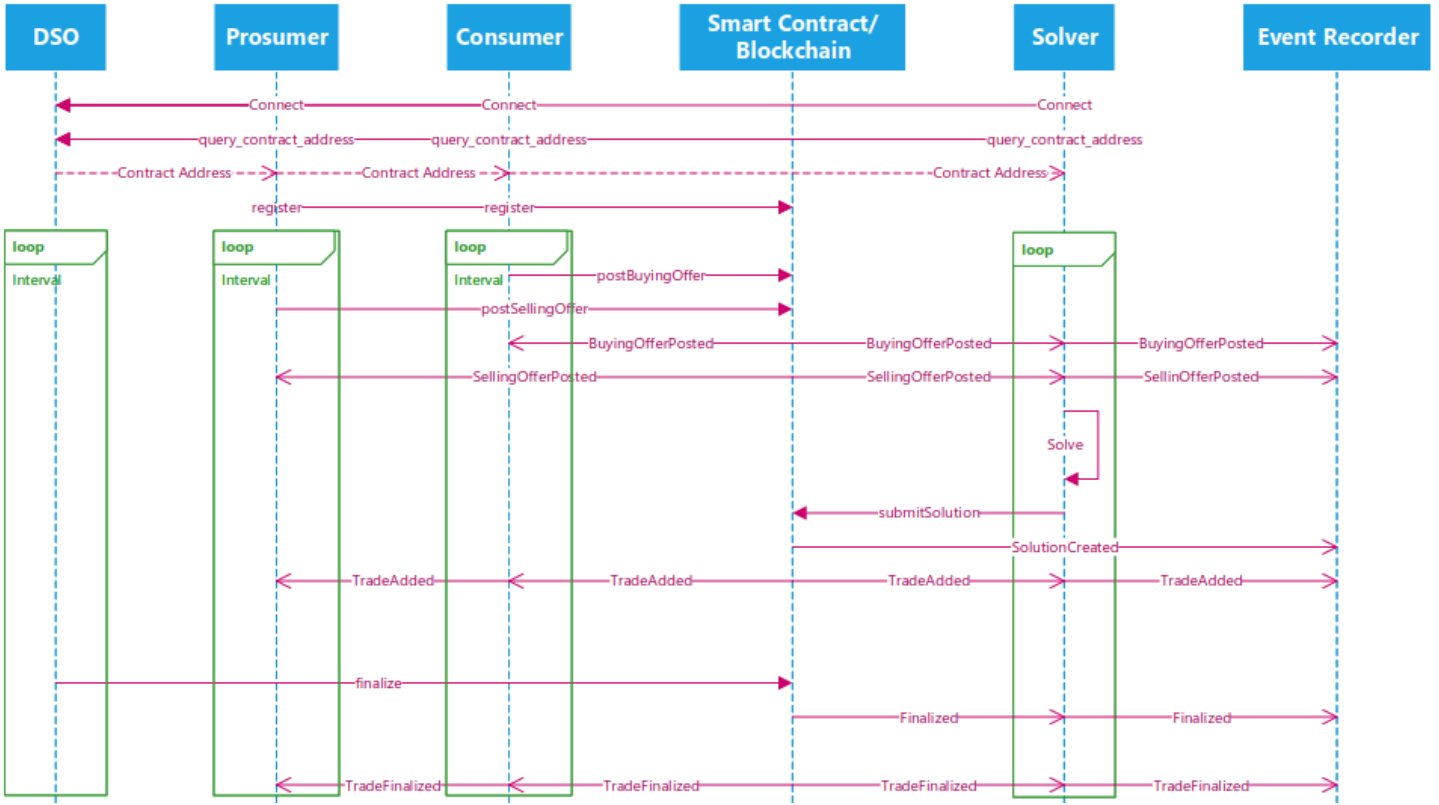
Sequence Diagram



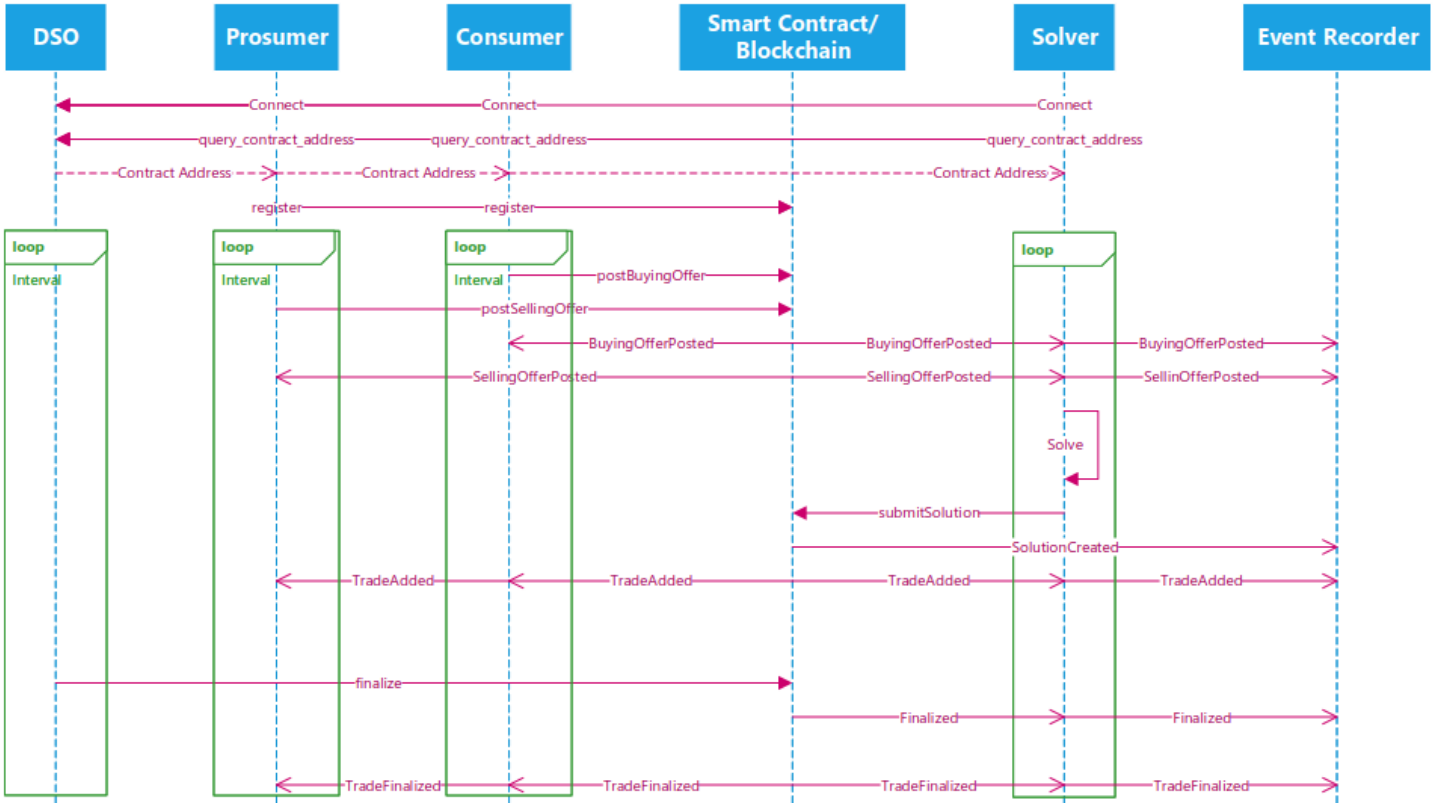
Sequence Diagram



Sequence Diagram

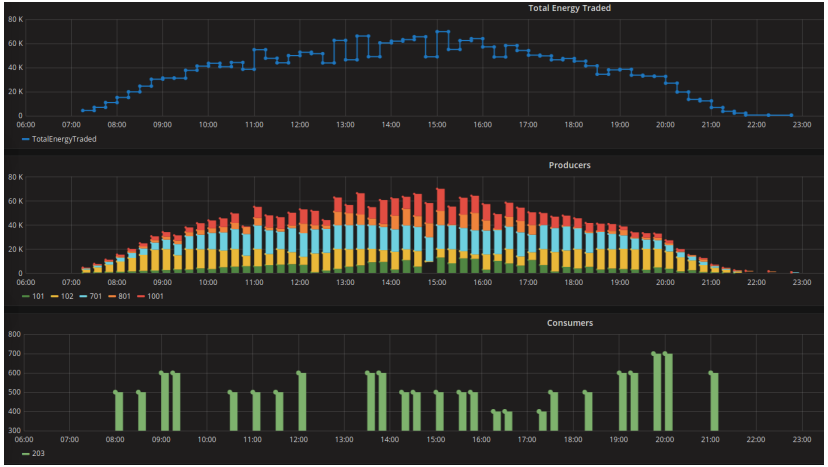


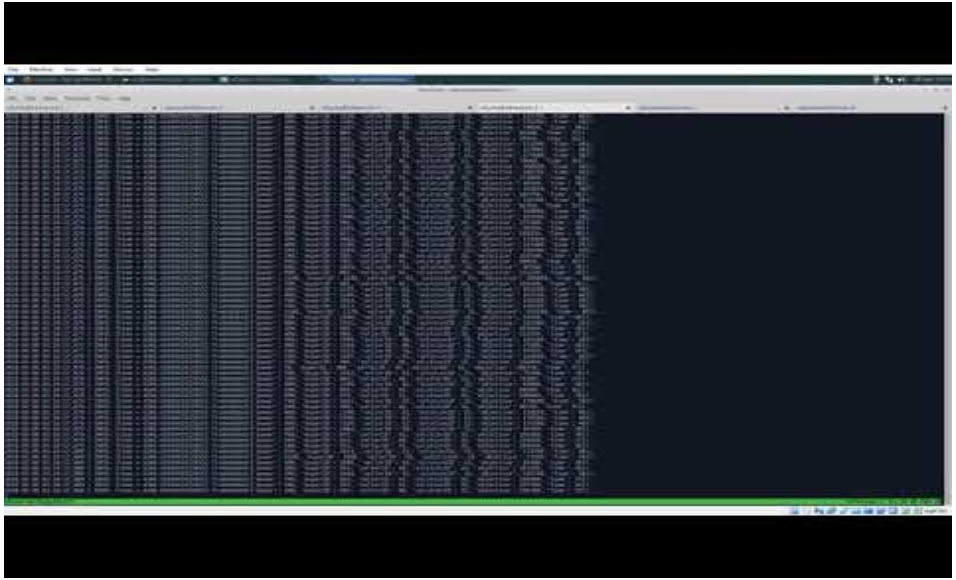
Sequence Diagram



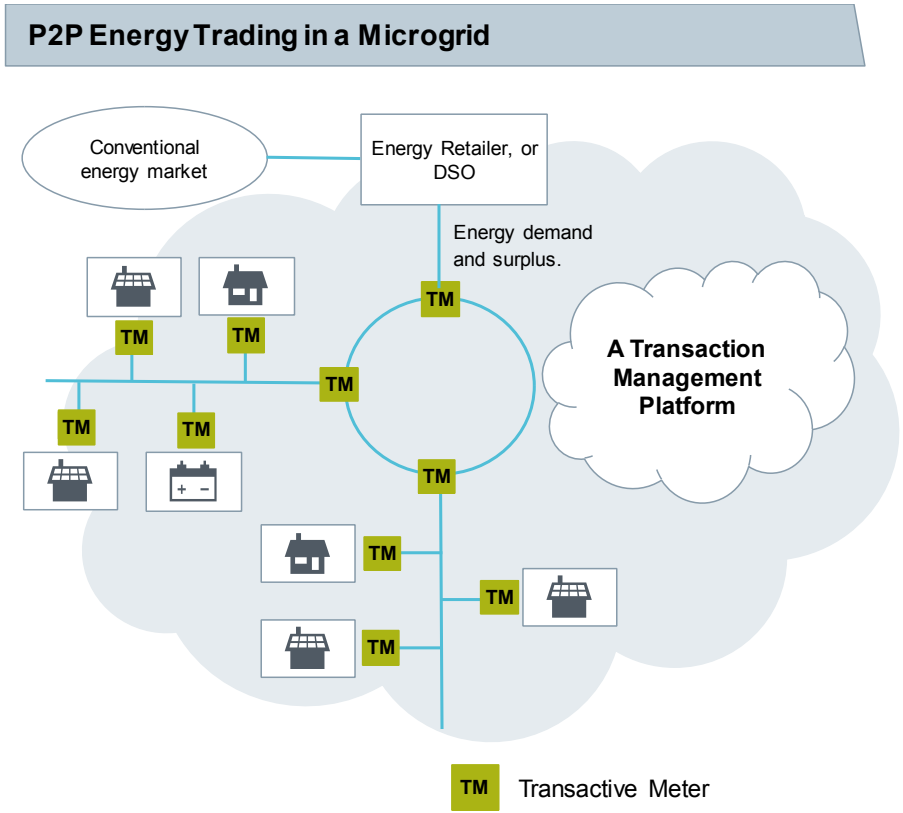
Evaluation

- We used real-world energy production / consumption data from a German microgrid provided by Siemens, CT
- We deployed our system on a **private** Ethereum network
- 5 producers
- ~97 consumers





Transactive Energy: Smart Homes → Smart Prosumers



Privacy-Preserving Platform for Transactive Energy Systems

Karla Kvaternik
Siemens Corporate Technology
karla.kvaternik@siemens.com

Aron Laszka
Vanderbilt University
aron.laszka@vanderbilt.edu

Michael Walker
Vanderbilt University
michael.a.walker.1@vanderbilt.edu

Douglas Schmidt
Vanderbilt University
d.schmidt@vanderbilt.edu

Monika Sturm
Siemens Corporate Technology
monika.sturm@siemens.com

Martin Lehofer
Siemens Corporate Technology
martin.lehofer@siemens.com

Abhishek Dubey
Vanderbilt University
abhishek.dubey@vanderbilt.edu

Abstract

Transactive energy systems (TES) are emerging as a transformative solution for the problems faced by distribution system operators due to an increase in the use of distributed energy resources and a rapid acceleration in renewable energy generation. These, on one hand, pose a decentralized power system controls problem, requiring strategic microgrid control to maintain stability for the community and for the utility. On the other hand, they require robust financial markets operating on distributed software platforms that preserve privacy. In this paper, we describe the implementation of a novel, blockchain-based transactive energy system. We outline the key requirements and motivation of this platform, describe the lessons learned, and provide a description of key architectural components of this system.

Keywords Transactive energy platforms, blockchain, privacy, security, safety, smart contracts

ACM Reference format:
Karla Kvaternik, Aron Laszka, Michael Walker, Douglas Schmidt, Monika Sturm, Martin Lehofer, and Abhishek Dubey. 2017. Privacy-Preserving Platform for Transactive Energy Systems. In *Proceedings of ACM/IFIP/USENIX Middleware conference, Las Vegas, Nevada USA, December 2017 (Middleware'17)*, 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 Introduction

Emerging Trends: Transactive energy systems (TES) have emerged as an anticipated outcome of the shift in electricity industry, away from centralized, monolithic business models characterized by bulk generation and one-way delivery, toward a decentralized model in which end users play a more active role in both production and consumption [10] [24]. In this paper, we consider a class of TES that operates in grid-connected mode. The main actors are the consumers, connection of the network. Such installations are equipped with an advanced metering infrastructure consisting of TE-enabled smart meters. In addition to the standard functionalities of smart meters: i.e. the ability to measure line voltages, power consumption and production, and communicate these to the distribution system operator (DSO); TE-enabled smart meters are capable of communicating with other smart meters, have substantial on-board computational resources, and are capable of accessing the Internet and cloud computing services as needed. Examples of such installations include the well-known Brooklyn Microgrid Project, [3] and the Sterling Ranch learning community (currently under development) [12]. A key component of TES is a transaction management platform (TMP), which handles all market clearing functions in a way that balances supply and demand in the local market.

Why Blockchains?: The capabilities of TE-enabled meters allow them to form a blockchain (BC) based TMP executing a market mechanism, using smart contracts [29]. Examples of BC systems capable of executing smart contracts include Ethereum [7] and Hyperledger Fabric [9]. There are a number of appealing properties of BC systems that motivate their use in a TMP. Firstly, BC technology enables the digital representation of energy and financial assets, and their secure transfer from one set of parties to another. By design, the security of this value transfer is guaranteed by the interaction protocol itself and obviates the need for trusted transaction intermediaries. Secondly, the execution of smart contracts (i.e. code that captures the market logic and participant roles) is automated and guaranteed. Thirdly, the blockchain constitutes an immutable, complete, and fully auditable record of all transactions that have ever occurred in the BC system. These properties ensure market transparency, as well as the availability of a detailed market load profile, and grid utilization data. Thus, [1, 4, 27] have already considered such implementations.

Open challenges: Existing initiatives such as [1, 4, 27] do not

Enabling safe & private interactions with blockchains in smart grid
(see arxiv.org/abs/1709.09597?context=cs.DC)

There are still Challenges

Ethereum Smart Contracts

Ethereum Smart Contracts

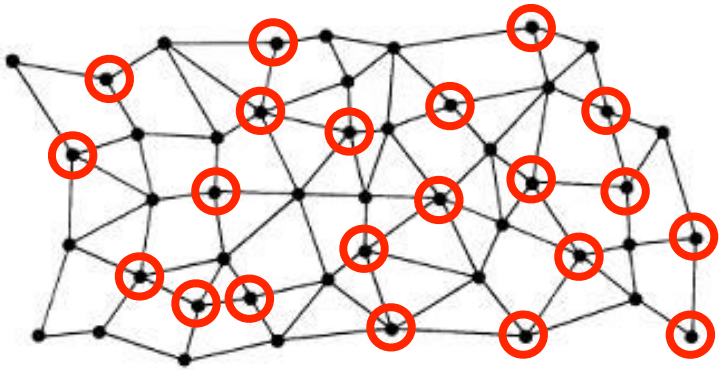
- Smart contracts are programs that run on the blockchain

```
contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;
    /* Initializes contract with initial supply tokens to the creator of the contract */
    function MyToken(uint256 initialSupply) {
        balanceOf[msg.sender] = initialSupply;              // Give the creator all initial tokens
    }
    /* 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
    }
}
```

Security Vulnerabilities

- Contracts are riddled with bugs and security vulnerabilities

- A recent automated analysis of **19,336** contracts Ethereum contracts
  - **8,333** contracts suffer from at least one security issue



Luu, Loi, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. "Making smart contracts smarter." In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 254-269. ACM, 2016.

Why is that Important?

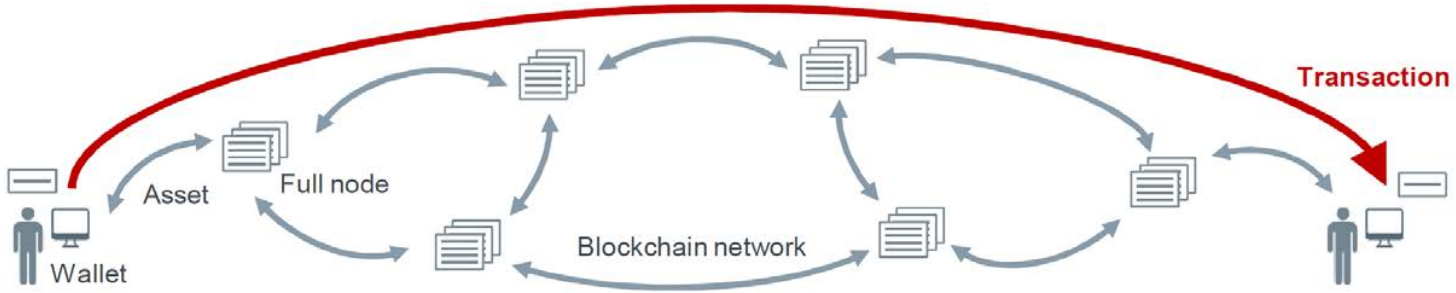
- Smart Contracts handle financial assets of significant value!

- The value held by Ethereum contracts is: **12,205,760 Ethers**
  - This is around **$11 Billion**

Why is that Important?

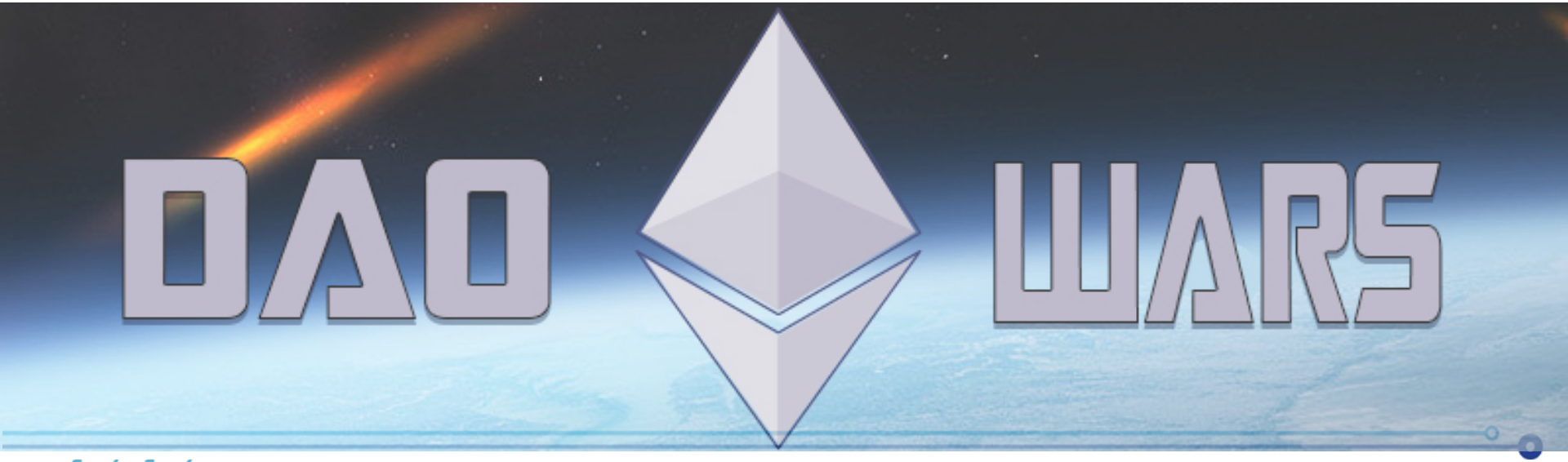
- Smart contract bugs cannot be patched
  - Once a contract is deployed, its functionality cannot be altered

- Blockchain transactions cannot be rolled back
  - Once a malicious transaction is recorded it cannot be removed
  - "Code is law" principle



A Transaction can be Rolled Back..

- .. with a hard fork of the blockchain
  - Requires consensus among all stakeholders
  - Undermines the trustworthiness of the platform
- Ethereum forked the blockchain to undo the Dao attack



The Infamous DAO Attack

- The DAO was a contract with ~$150M built by Ethereum creators
  - A combination of vulnerabilities was exploited
  - Attackers stole 3.6M Ethers, worth ~$60M at the time of the attack
  - Re-entrancy vulnerability

Re-entrancy Vulnerability

- In Ethereum, when there is a function call
  - The caller has to wait for the call to finish - synchronous calls
  - A malicious callee might take advantage of this


```
function withdraw(uint amount) {
  if (credit[msg.sender]>= amount) {
    msg.sender.call.value(amount)();
    credit[msg.sender]-=amount;
}
```

Re-entrancy Vulnerability

- In Ethereum, when there is a function call
  - The caller has to wait for the call to finish - synchronous calls
  - A malicious callee might take advantage of this


```
function withdraw(uint amount) {
  if (credit[msg.sender]>= amount) {
    msg.sender.call.value(amount)();
    credit[msg.sender]-=amount;
}
```

```
function() {
  dao.withdraw(dao.queryCredit(this));
}
```

Re-entrancy Vulnerability

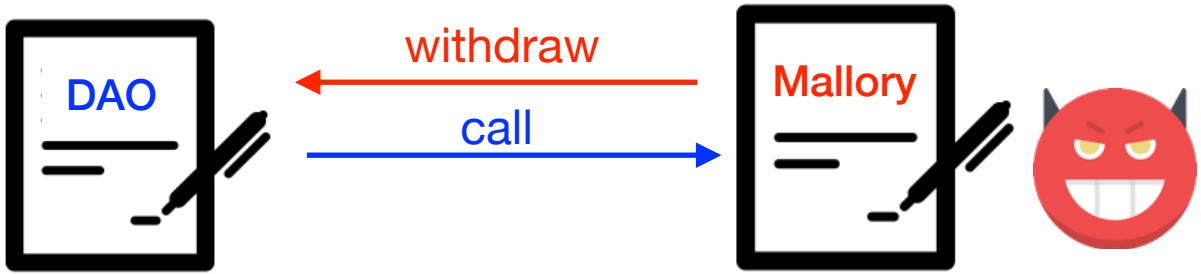
- In Ethereum, when there is a function call
  - The caller has to wait for the call to finish - synchronous calls
  - A malicious callee might take advantage of this


```
function withdraw(uint amount) {
  if (credit[msg.sender]>= amount) {
    msg.sender.call.value(amount)();
    credit[msg.sender]-=amount;
}
```

```
function() {
  dao.withdraw(dao.queryCredit(this));
}
```

Re-entrancy Vulnerability

- In Ethereum, when there is a function call
  - The caller has to wait for the call to finish - synchronous calls
  - A malicious callee might take advantage of this

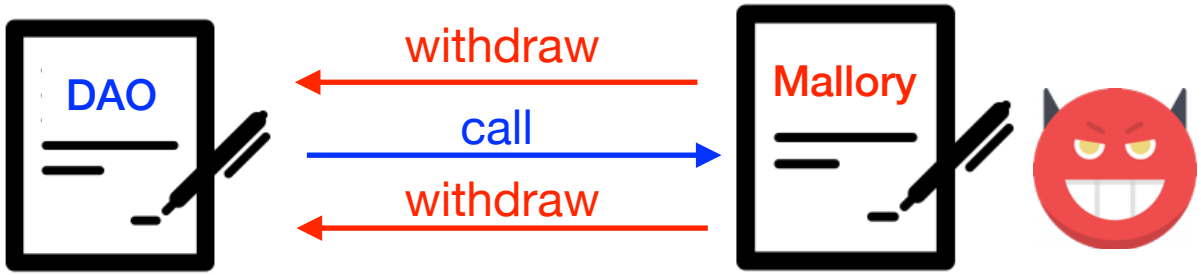



```
function withdraw(uint amount) {
  if (credit[msg.sender]>= amount) {
    msg.sender.call.value(amount)();
    credit[msg.sender]-=amount;
}
```

```
function() {
  dao.withdraw(dao.queryCredit(this));
}
```

Re-entrancy Vulnerability

- In Ethereum, when there is a function call
  - The caller has to wait for the call to finish - synchronous calls
  - A malicious callee might take advantage of this




```
function withdraw(uint amount) {
  if (credit[msg.sender]>= amount) {
    msg.sender.call.value(amount)();
    credit[msg.sender]-=amount;
}
```

```
function() {
  dao.withdraw(dao.queryCredit(this));
}
```

Unpredictable State Vulnerability

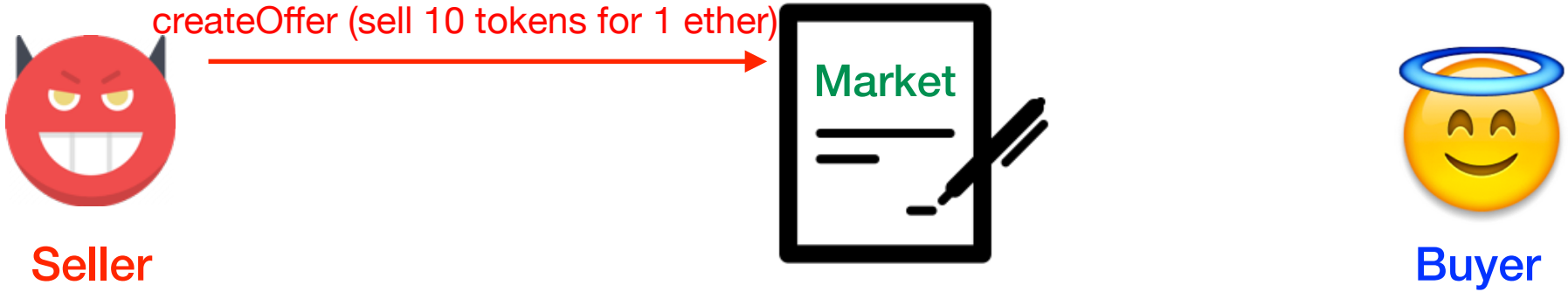
- The order of execution of function calls cannot be predicted
- No prior knowledge of a contract's state during call execution

Seller

Buyer

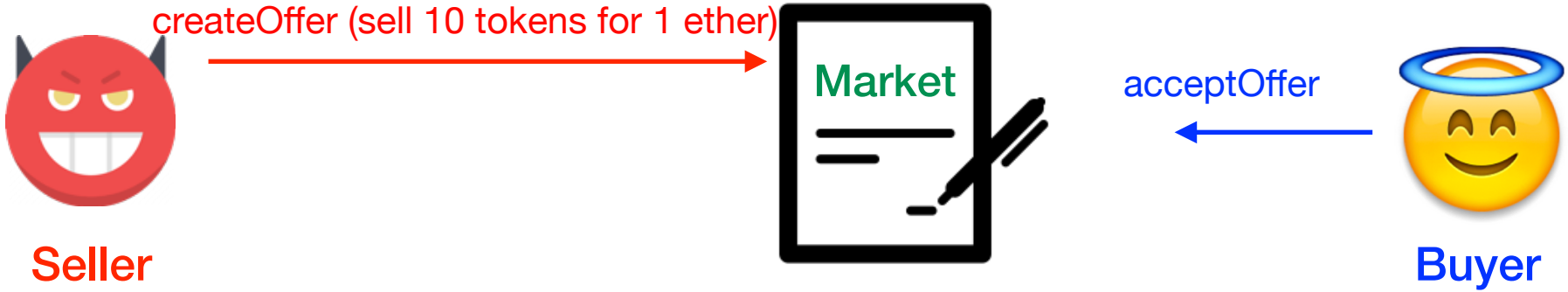
Unpredictable State Vulnerability

- The order of execution of function calls cannot be predicted
- No prior knowledge of a contract's state during call execution



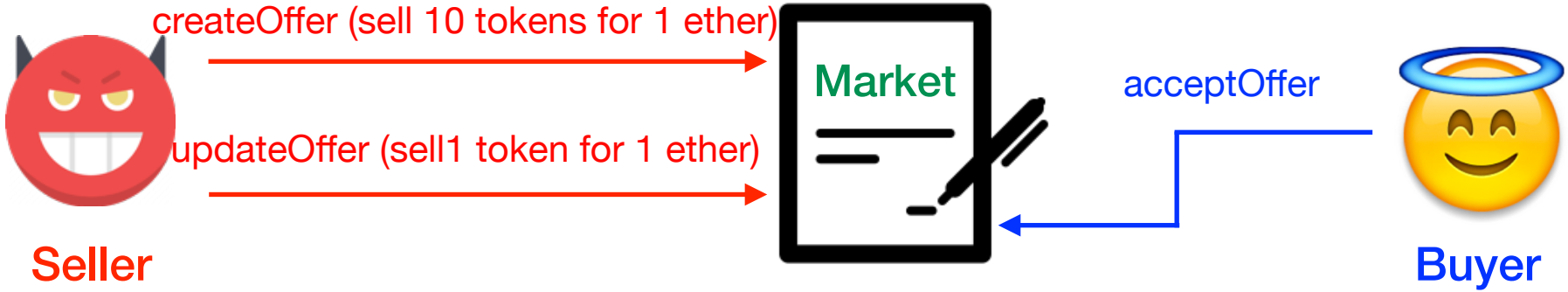
Unpredictable State Vulnerability

- The order of execution of function calls cannot be predicted
- No prior knowledge of a contract's state during call execution



Unpredictable State Vulnerability

- The order of execution of function calls cannot be predicted
- No prior knowledge of a contract's state during call execution

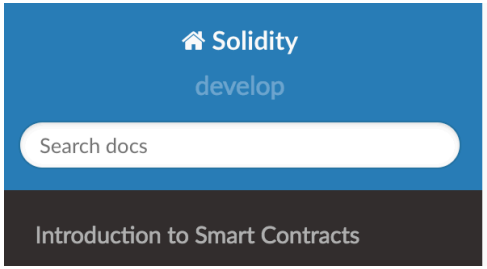


The Reason behind many Vulnerabilities

- Vulnerabilities often arise due to the semantic gap between
  - The underlying execution semantics
  - The actual semantics of smart contracts
- There exist tools for identifying common vulnerabilities in existing contracts
- We explore a different avenue
  - We want to help developers to create secure and correct smart contracts

Our Approach

- Relies on the following observations. Smart contracts:
  - Have states
  - Provide functions that can be invoked and change the contract state
- Smart contracts can be naturally represented by state machines
- Adequate level of abstraction for reasoning about their behavior



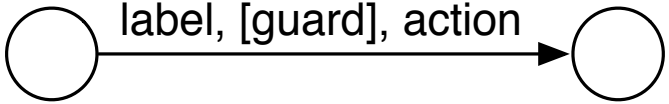
State Machine

Contracts often act as a state machine, which means that they have certain **stages** in which they behave differently or in which different functions can be called. A function call often ends a stage and transitions the contract into the next stage (especially if the contract models **interaction**). It is also common that some stages are automatically reached at a certain point in **time**.

http://solidity.readthedocs.io/en/develop/common-patterns.html#state-machine

State Machines

- Taking a transition
  - Is allowed if the guard evaluates to true
  - Executes the action
  - Updates the contract's current state



Definition 1. *A Smart Contract is a tuple* $(S, s_0, C, I, O, \rightarrow)$*, where:*

- S *is a finite set of states;*
- $s_0 \in S$ *is the initial state;*
- C*,* I*, and* O *are disjoint finite sets of, respectively, contract, input, and output variables;*
- $\rightarrow \subseteq S \times \mathcal{G} \times \mathcal{F} \times S$ *is a transition relation, where:*
  - $\mathcal{G} = \mathbb{B}[C, I]$ *is a set of guards;*
  - $\mathcal{F}$ *is a set of action sets, i.e., a set of all ordered powersets of* $\mathbb{E}[C, I, O]$

FSolidM Demo

Examples of FSolidM Plugins

- Locking plugin

```
bool private locked = false;
modifier locking {
  require(!locked);
  locked = true;
  _;
  locked = false;
}
```

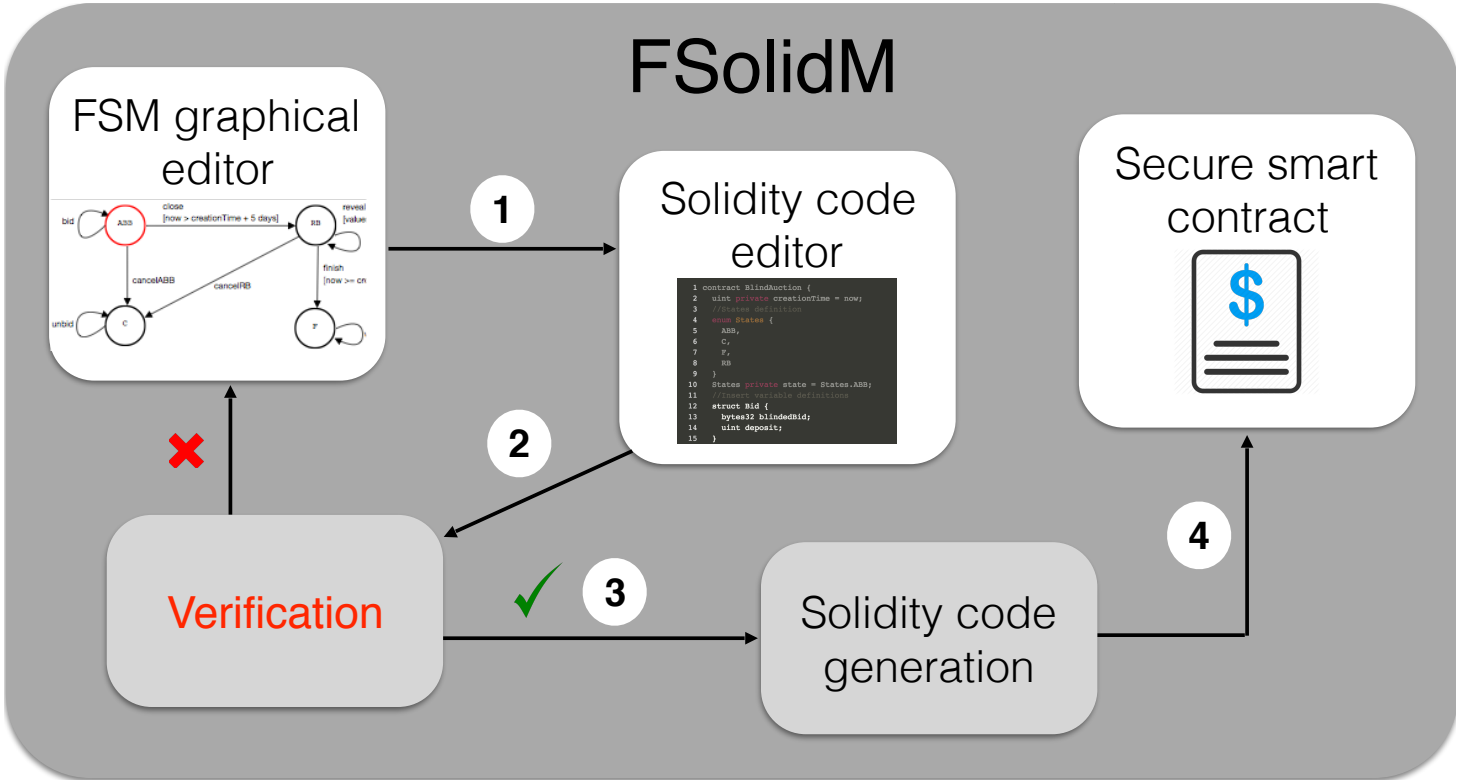
- Reentrancy vulnerability

- Transition Counter

```
uint private transitionCounter = 0;
modifier transitionCounting(uint nextTransitionNumber) {
  require(nextTransitionNumber == transitionCounter);
  transitionCounter += 1;
  _;
}
```

- Unpredictable state vulnerability

Current Work on FSolidM



Verification of Smart Contracts

- Deadlock-freedom analysis
  - Parity wallet vulnerability was based on a deadlocked contract
- Functional property analysis

$\langle$***Transitions*** $\cup$ ***Statements***$\rangle$ cannot happen after $\langle$***Transitions*** $\cup$ ***Statements***$\rangle$.	If $\langle$***Transitions*** $\cup$ ***Statements***$\rangle$ happens, $\langle$***Transitions*** $\cup$ ***Statements***$\rangle$ can happen only after $\langle$***Transitions*** $\cup$ ***Statements***$\rangle$ happens.

DAO attack	if `call` happens, `call` can only happen after `subtract`	$\mathtt{AG}\big(call \rightarrow \mathtt{AX}\ \mathtt{A}\big\lfloor \neg call\ \mathtt{W}\ subtract \big\rfloor\big)$

The FSolidM Framework

- Formal model, clear semantics easy-to-use graphical editor
  - Decreasing the semantic gap
- Rigorous semantics
  - Amenable to analysis and verification
- Code generation + functionality and security plugins
  - Minimal amount of error-prone manual coding
- Tool and publications: http://cps-vo.org/group/SmartContracts
- Source code: http://github.com/anmavrid/smart-contracts

Anastasia Mavridou, Aron Laszka."Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach." Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC). 2018.

The FSolidM Framework

- Formal model, clear semantics easy-to-use graphical editor
  - Decreasing the semantic gap
- Rigorous semantics
  - Amenable to analysis and verification
- Code generation + functionality and security plugins
  - Minimal amount of error-prone manual coding
- Tool and publications: http://cps-vo.org/group/SmartContracts
- Source code: http://github.com/anmavrid/smart-contracts

Anastasia Mavridou, Aron Laszka."Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach." Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC). 2018.

Correct-by-design smart contracts