Smarter Blockchains - from Transactions to Contracts

Blockchain Workshop, Dallund Slot, May 2018

Martin Elsman
Department of Computer Science
University of Copenhagen (DIKU)
The Speaker

Martin Elsman, Associate Professor at DIKU

Research activities:
- Certified management of financial contracts.
- Programming language design and implementation (functional languages)
- Parallel programming languages - getting programs, such as simulations, to run efficiently on GPUs.

Other activities:
- Manager, HIPERFIT Research Center, DIKU (2012-2018)
Plain Blockchain Recap

Features:

- A **distributed ledger** with **distributed authority**
- Useful to store transactions **securely** and **privately**
- Plain Bitcoin blockchain allows only for storing Bitcoin transactions
- Other blockchain systems allows for storing ad-hoc **user-defined transactions** involving other types of digital assets (anything with a hash-key).
Modern blockchain systems, such as Ethereum, allow for so-called **smart contracts** to be executed on the blockchain system.

A **smart contract** is a small program that runs on the blockchain system (i.e., in principle by every node).

The **smart contract** may hold assets (e.g., digital cash) and listen to events (i.e., react on transactions by issuing other transactions).
The Ethereum Blockchain System – I

Ethereum is specified openly (as the Bitcoin blockchain) in the “yellow paper”.

**Ether**: Ethereum’s own cryptocurrency.

**Ethereum Virtual Machine (EVM) code**: Bytecode instructions that execute on the system (i.e., by each node)

**GAS**: The cost associated with executing bytecode instructions (Turing-completeness).

**Data Feed**: Access to the external world from within Ethereum code.
Smart Contracts (i.e., programs) may not be written in low-level EVM code, but may be written languages that compile to EVM code:

- **Solidity**: A JavaScript-like object-oriented language
- **Vyper**: A simple Python-like language
- **LLL**: Low-level Lisp-like code

Smart contracts are neither smart nor non-contracts:

- **Not smart**: Not declarative: describes how not what
- **Not contracts**: They don’t describe agreed-upon obligations

---

The Ethereum Blockchain System – II

1. Introduction

With ubiquitous internet connections in most planes of the world, global (decentralized) transactions has become incredibly cheap. Technology-driven movements like Bit-coin have demonstrated, through the power of its decentralized, consensus mechanism and voluntary aspect of the social contract that it is possible to see the internet make a decentralized value-transfer system, shared across the world and virtually free to use. This system can be said to be a very special version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adopted this original "currency application" of the technology into other applications which built from the blockchain. Ethereum is a project which attempts to build the generalized technology-stack upon which all transaction-based state-machine concepts may be built. Moreover it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a kilobyte-size expanded computer-program in the mainstream: a truthful object managing computer framework.

1.1. Building Blocks. There are many goals of this project; one key goal is to facilitate transactions between consumers, individuals who would otherwise have no means to trust one another. This may be due to geographical separation, interacting difficulty, or perhaps the insufficiency of intermediaries, knowledge, experience, consciousness, incomprehension or continuous variations of the legal systems. By specifying a state-change system through a rich and inconspicuous language, and further articulating a system such that we can reason expectantly that an agreement will be thus enforced, it is possible to provide a means to such end.

Building on the proposed system would have several attributes not often found in the real world. The incorruptibility of judgement, often difficult to find, can be provided naturally from a shatterproof algorithmic interpreter. Transparency, or being able to see exactly how a state or judgement came about through the transaction log and/or instruction logs, never happens perfectly in human-based systems since natural language is necessarily vague. Information is often lacking, andplain facts are difficult to note.

Overall, I wish to provide a system such that users can be guaranteed that no matter with which other software, systems, or organizations they interact, they can do so with absolute confidence in the possible outcomes and how those outcomes might evolve.

1.2. Previous Work. Butonius [2016] first proposed the kernel of this work in late November, 2015. Though not evolved in many ways, the key functionality of a blockchain with a Turing-complete language and self-executing, unilaterally inter-transaction exchange capability remains the same.

Dwork and Naor [1992] provided the first work into the range of a cryptographically proof of computational operations ("proof-of-work") as a means of transmitting a value signal over the Internet. The technology was utilized here as a space determinate mechanism rather than any kind of currency, but essentially demonstrated the potential for a basic data structure to carry a strong economic signal, allowing a monitor to make a physical assertion without having to rely on smart. Butonius [2016], later, provided a system in a similar vein with the idea of utilizing the proof-of-work as a strong economic signal to receive a money was by Viskovtsov et al. [2015]. In this instance, the notion was used to keep peer-to-peer the firing line is direct, ensuring "payment" is able to take extra payments to "negotiate" for their services. The security model allowed by the design was augmented with digital signatures and a ledger is in order to ensure a historical record wouldn’t be corrupted and that malicious actions could not spoof payments or unfairly complaints about service delivery. Five years later, Nakamoto [2009] (2008) developed a entirely different proof-of-work-based value token, sometimes refer to as the Bitcoin project. The result was not just a blockchain, because the first widely adopted global decentralized transaction ledger.

Other projects built on Ethereum’s success, the air-born introduced numerous other innovations through alterations to the protocol. Some of the best known are Lisk’s and Personon, discussed by Sprakel [2016]. Other projects sought to take the core value system mechanism of the
Possible Uses of Ethereum

Financial contracts:
- Swaps, Options, OTC contracts, ...

Digital rentals:
- Car/hotel rental: A personal digital code for hotel room or car is swapped with Ether (rental & deposit). Deposit is returned when car/hotel room has been inspected.

Contracts on goods:
- Ether is transferred when merchandise is delivered.

Let’s look into this possibility... We can even manage margin accounts...

Etherscan is a project which attempts to build the gen- eralized technology architecture on which all transaction-based state machine concepts may be built. Moreover it aims to provide the end developer a tightly integrated tool-to-tool system for building software on a kilobit- sized computer program in the non-trivial: a treatable object managing compute framework.

Let’s look into this possibility... We can even manage margin accounts...

1. Introduction
With ubiquitous internet connections in most planes of the world, global (decentralized) transmissions has become incredibly cheap. Technology-rooted movements like Bit- coin have demonstrated, through the power of the market, consumer enthusiasm and voluntary respect of the social contract that it is possible to see the internet to make a decentralized value-transfer system, shared across the world and virtually free to use. This system can be said to be a very specialized version of a cryptography-based au- thorized, transaction-based state machine. Follow-up systems such as Namecoin adopted this original "currency appli- cation" of the technology into other applications which had yet to be invented. Ethereum is a project which attempts to build the gen- eralized technology architecture on which all transaction-based state machine concepts may be built. Moreover it aims to provide the end developer a tightly integrated tool-to-tool system for building software on a kilobit- sized computer program in the non-trivial: a treatable object managing compute framework.

1.1. Defining Parameters. There are many goals of this project, one key goal is to facilitate transactions between consumers that would otherwise have no means to transact with one another. This may be due to geographical separation, financial difficulty, or perhaps the immorality, transgressiveness, knowliness, expensive, unaccountability, inaccuracies or corruption of existing legal systems. By specifying a state-change through a risk and consequence language, and furthermore redefining a system such that its data is reasonably expect that an agreement will be trust enforced automatically, we can provide a means to this end.

Business is this proposed system would have several attributes not often found in the real world. The incor- ruptibility of judgment, often difficult to find, means na- turally from a disembarrassed algorithmic interpret- er, trans- parent, or being able to see exactly how a state or judg- ment came about through the transaction log and rules or instructional rules, never happens perfectly in human- based systems since natural language is inherently vague.
Existing Blockchain Implementations

**Existing Blockchain Implementations**

constructed to record mutual agreed-upon transactions...

Each node has a copy of the blockchain. New nodes get the chain from their peers.

A *mutual consensus mechanism* (proof-of-work) ensures that nodes agree on transactions.

Classical ledgers, records only transactions that has happened (facts), not transactions that are meant to occur in the future!

Party X transfers 2 bitcoins to party Y
Today’s Financial System

Individual, companies, and smaller service providers access the system by partnering with a large institution.

A small group of large institutions communicate bilaterally.

Regulatory authorities ensure consistency through audits of institutions.
Tomorrow’s Financial System

Based on blockchain technology!

The overhead of bilateral communication is eliminated.

All parties **enjoy direct access** to the financial system.

The ledger manages contracts and **automatically settles them** in accordance with participants’ strategies for doing so.

Access scales to an arbitrary number of participants as consensus protocols keep the ledger consistent.
Financial Contracts on the Ethereum Blockchain System

Constructed to record mutual agreed-upon future transactions (e.g., financial contracts)...

The blockchain makes evident that all involved parties have signed the contract.

When times passes, transfers and decisions (events) occur and are recorded in the blockchain.

An Ethereum smart-contract can arrange for the transfer to occur...

Example 1 (FX Forward). In 90 days, party X will buy 100 US dollars for a fixed rate 6.5 of Danish kroner from party Y.

\[ 90 \uparrow 100 \times (\text{USD}(Y \rightarrow X) \& 6.5 \times \text{DKK}(X \rightarrow Y)) \]
How do we Know that the Smart Contract is Implemented Correctly?

Lots of trusted components, incl:

- Financial Contract
- Solidity Smart Contract
- EVR Execution
- EVM Bytecode

**Example 1** (F-X Forward). In 90 days, party $X$ will buy 100 US dollars for a fixed rate 6.5 of Danish kroner from party $Y$.

$$90 \uparrow 100 \times (\text{USD}(Y \rightarrow X) \& 6.5 \times \text{DKK}(X \rightarrow Y))$$
A Certified Contract Management Engine

Contract combiners for specifying financial derivatives [2].

Contract kernel written in Coq, a functional language and proof assistant for establishing program correctness (wrt a cash-flow semantics).

Certified management code extracted from the Coq implementation (fixings, decisions).

American Option contract in natural language:
At any time within the next 90 days, party X may decide to buy USD 100 from party Y, for a fixed rate 6.65 of Danish Kroner.

Specified in the contract language:

\[
\text{if } \text{obs}(X \text{ exercises option}) \text{ within } 90 \text{ then }
100 \times (\text{USD}(\text{Y} \rightarrow \text{X}) \& 6.65 \times \text{DKK}(\text{X} \rightarrow \text{Y}))
\text{ else } \emptyset
\]

A Financial Contract Language

**Features:**

**Compositionality**
Contracts are time-relative ⇒ compositionality

**Multi-party**
Possibility for specifying portfolios

**Contract management**
Contracts can be managed (fixings, splits, ...)
Contracts gradually reduce to the empty contract

**Contract utilities (symbolic)**
Contracts can be analysed in a variety of ways (find horizon, potential cash-flows, ...)

**Assumptions**

- \(d\) integer (specifies a number of days)
- \(p\) ranges over parties (e.g., YOU, ME, X, Y)
- \(a\) assets (e.g., USD, DKK)

**Expressions (extended expressions for reals and booleans)**

- \(\text{obs}(l,d)\) observe the value of \(l\) (a label) at time \(d\)
- \(\text{acc}(f,d,e)\) accumulate function \(f\) over the previous \(d\) days

**Contracts (c)**

- \(\emptyset\) empty contract with no obligations
- \(a(p_1 \rightarrow p_2)\) \(p_1\) has to transfer one unit of \(a\) to \(p_2\)
- \(c_1 \& c_2\) both \(c_1\) and \(c_2\)
- \(e \times c\) multiply all obligations in \(c\) by \(e\)
- \(d \uparrow c\) shift \(c\) into the future by \(d\) days
- \(\text{let } x = e \text{ in } c\) bind today’s value of \(e\) to \(x\) in \(c\)

**if e within d then c1 else c2**

behave as \(c_1\) when \(e\) becomes true if \(e\) does not become true within \(d\) days, behave as \(c_2\)
Asian Option

90 \( \uparrow \) if obs(X exercises option) \textbf{within} 0 \textbf{then} \\
\hspace{1cm} 100 \times (\text{USD}(Y \rightarrow X) \& (rate \times \text{DKK}(X \rightarrow Y))) \\
\hspace{1cm} \textbf{else} \emptyset

\textbf{where}

rate = \frac{1}{30} \cdot \text{acc}(\lambda r. r + \text{obs}(\text{FX USD/DKK}), 30, 0)

\textbf{Notice:} the special acc-construct is used to compute an average rate.

Simple Credit Default Swap (CDS)

\textbf{The bond:}
\[ C_{\text{bond}} = \text{if} \ \text{obs}(X \text{ defaults}, 0) \text{ within} 30 \text{ then} \emptyset \]
\[ \hspace{1cm} \textbf{else} \ 1000 \times \text{DKK}(X \rightarrow Y) \]

\textbf{Insurance:}
\[ C_{\text{cds}} = (10 \times \text{DKK}(Y \rightarrow Z)) \& \]
\[ \hspace{1cm} \text{if} \ \text{obs}(X \text{ defaults}, 0) \text{ within} 30 \text{ then} \\
\hspace{2cm} 900 \times \text{DKK}(Z \rightarrow Y) \]
\[ \hspace{1cm} \textbf{else} \emptyset \]

\textbf{Entire Contract:}
\[ C = C_{\text{bond}} \& C_{\text{cds}} \]
Benefits of the Formal Framework

Some contract equivalences (algebra)

\[
\begin{align*}
e_1 \times (e_2 \times c) & \equiv (e_1 \cdot e_2) \times c \\
d_1 \uparrow (d_2 \uparrow c) & \equiv (d_1 + d_2) \uparrow c \\
d \uparrow (c_1 \& c_2) & \equiv (d \uparrow c_1) \& (d \uparrow c_2) \\
e \times (c_1 \& c_2) & \equiv (e \times c_1) \& (e \times c_2)
\end{align*}
\]

With a netting semantics:

\[
(e_1 \times a(p1\rightarrow p2)) \& (e_2 \times a(p1\rightarrow p2)) = (e_1 + e_2) \times a(p1\rightarrow p2)
\]

Other benefits:
- Type system for causality
- Correctness of contract evolution

One cannot pay today an amount that depends on a value tomorrow.
Consequences – I

Bye-bye Banks

Automated Execution of Financial Contracts on Blockchains

Benjamin Eckerle-Müller · Martin Elmanz · Fette Hogrange · Uwe Ross

Received: 16 January 2017 Accepted: 16 August 2017
© Springer Fachmedien Wiesbaden 2017

Abstract The paper investigates financial contract management on distributed ledgers and provides a working solution implemented on the Ethereum Blockchain. The system is based on a domain-specific language for financial contracts that is capable of representing complex multi-party derivations and is conducive to automated execution. The authors propose an architecture for spanning contractual terms from contract execution, a contract evaluation mechanism for the validation of execution of financial contracts without actively performing contractual actions; such actions are handled by non-fungible contract managers that administer strategies for the execution of contracts. Housing contracts and contract managers on a distributed ledger, side-by-side with digital assets, facilitates automated settlement of commitments without need for an intermediary. The paper discusses how the proposed technology may change the way financial institutions, regulators, and individuals interact in a financial system based on distributed ledgers.

Keywords Blockchain · Domain-specific language · Financial services · Distributed ledger

I Introduction

The pillars on which the financial industry has been based for the last century are being challenged. The disruptive nature of new technologies such as modern machine learning and blockchain technology are changing the rules that form the financial sector and the financial system as a whole. Silbak et al. (2014) estimate savings from blockchain-based technologies to be in the region of tens of billions of US dollars annually across the financial sector with 0.1–0.3 billion in annual savings for the settlement of cash securities alone. In this paper, we demonstrate how a financial contract management system built upon a generalized distributed ledger can automate the execution of contracts, including clearing and settlement, thus potentially inducing drastic changes in the financial industry. Essentially, a distributed ledger on Blockchain offers participants the opportunity to establish distributed consensus on a set of shared facts without assuming mutual trust. It does so by implementing a single coherent high-level of events shared amongst a set of participating parties, which acts as a single point of truth. Critically, no privileged parties are required to maintain the ledger.

In its basic form, a distributed ledger provides a fixed protocol for adding new events in a log of events. In Bitcoin (Nakamoto 2008) the basic protocol ensures that a Bitcoin miner can only occur as an authenticated owner, whose transaction history must be in a positive balance, where the amount transferred is at most that balance and has not already been spent. Blockchain thus ensures a specific, constant amongst open-ended number of
Consequences – II

No need for classic banks to interpret paper contracts.

No need for central players, such as clearing houses.

Even margin accounts can be implemented using smart-contracts that themselves can hold digital assets.

Needs and opportunities:

- Secure (i.e., certified) and transparent blockchain implementations.
- Solutions to orchestrate new blockchain variations.
- Possibility for linking with real world assets (e.g., mortgages, car loans).

Other applications:

- Software contracts...
- Other contracts...
Thanks!