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.

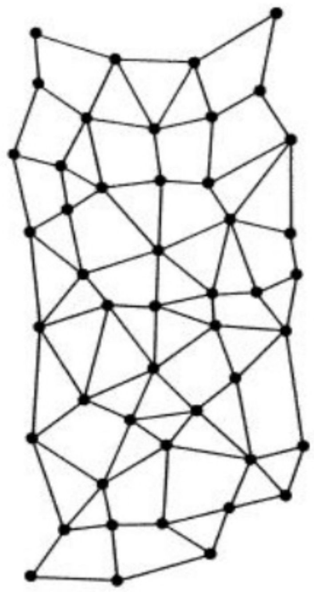
Other activities:

- Manager, HIPERFIT Research Center, DIKU (2012-2018)
- CTO and partner in iAlpha - a London-, DK-, and Swiss-based startup specialising in financial analytics.

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).



DISTRIBUTED
(C)

Smarter Blockchain Systems

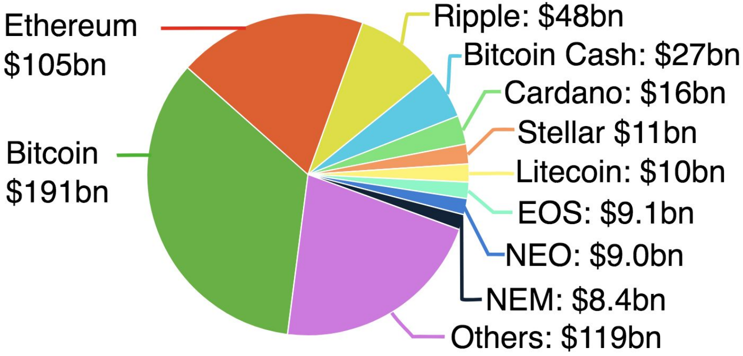
Modern blockchain systems, such as Ethereum, allows for so-called ***smart contracts*** to be executed on the blockchain system.

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).

Some smart blockchain systems:

- Bitcoin contracts
- Ethereum
- NEO (Chinese-Ethereum)
- EOS
- Ripple, ...



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.

ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER
EIP-150 REVISION

DR. GAVIN WOOD
FOUNDER, ETHEREUM & ETHCORE
GAVIN@ETHCORE.IO

Abstract. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not least Bitcoin. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

1. Introduction

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated, through the power of the default, consensus mechanisms and voluntary respect of the social contract that it is possible to use the internet to make a decentralised value-transfer system, shared across the world and virtually free to use. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adapted this original “currency application” of the technology into other applications albeit rather simplistic ones.

Ethereum is a project which attempts to build the generalised technology; technology on 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 hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework.

1.1. **Driving Factors.** There are many goals of this project; one key goal is to facilitate transactions between consenting individuals who would otherwise have no means to trust one another. This may be due to geographical separation, interfacing difficulty, or perhaps the incompatibility, incompetence, unwillingness, expense, uncertainty, inconvenience or corruption of existing legal systems. By specifying a state-change system through a rich and unambiguous language, and furthermore architecting a system such that we can reasonably expect that an agreement will be thus enforced autonomously, we can provide a means to this end.

Dealings in this proposed system would have several attributes not often found in the real world. The incorruptibility of judgement, often difficult to find, comes naturally from a disinterested algorithmic interpreter. Transparency, or being able to see exactly how a state or judgement came about through the transaction log and rules or instructional codes, never happens perfectly in human-based systems since natural language is necessarily vague, information is often lacking, and plain old prejudices are difficult to shake.

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

1.2. **Previous Work.** Buterin [2013a] first proposed the kernel of this work in late November, 2013. Though now evolved in many ways, the key functionality of a block-chain with a Turing-complete language and an effectively unlimited inter-transaction storage capability remains unchanged.

Dwork and Naor [1992] provided the first work into the usage of a cryptographic proof of computational expenditure (“proof-of-work”) as a means of transmitting a value signal over the Internet. The value-signal was utilised here as a spam deterrence mechanism rather than any kind of currency, but critically demonstrated the potential for a basic data channel to carry a *strong economic signal*, allowing a receiver to make a physical assertion without having to rely upon *trust*. Back [2002] later produced a system in a similar vein.

The first example of utilising the proof-of-work as a strong economic signal to secure a currency was by Vishnumurthy et al. [2003]. In this instance, the token was used to keep peer-to-peer file trading in check, ensuring “consumers” be able to make micro-payments to “suppliers” for their services. The security model afforded by the proof-of-work was augmented with digital signatures and a ledger in order to ensure that the historical record couldn’t be corrupted and that malicious actors could not spoof payment or unjustly complain about service delivery. Five years later, Nakamoto [2008] introduced another such proof-of-work-secured value token, somewhat wider in scope. The fruits of this project, Bitcoin, became the first widely adopted global decentralised transaction ledger.

Other projects built on Bitcoin’s success; the alt-coins introduced numerous other currencies through alteration to the protocol. Some of the best known are Litecoin and Primecoin, discussed by Sprankel [2013]. Other projects sought to take the core value content mechanism of the

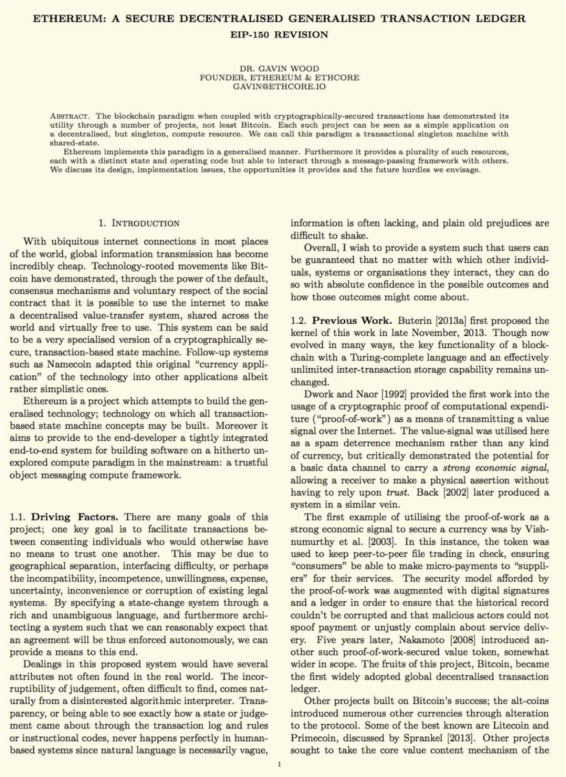
The Ethereum Blockchain System – II

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 contracts:

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



ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER
EIP-150 REVISION

DR. GAVIN WOOD
FOUNDER, ETHEREUM & ETHCORE
GAVIN@ETHCORE.IO

ABSTRACT. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not least Bitcoin. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

1. INTRODUCTION

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated, through the power of the default, consensus mechanisms and voluntary respect of the social contract that it is possible to use the internet to make a decentralised value-transfer system, shared across the world and virtually free to use. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adapted this original "currency application" of the technology into other applications albeit rather simplistic ones.

Ethereum is a project which attempts to build the generalised technology; technology on 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 hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework.

1.1. **Driving Factors.** There are many goals of this project; one key goal is to facilitate transactions between consenting individuals who would otherwise have no means to trust one another. This may be due to geographical separation, interfacing difficulty, or perhaps the incompatibility, incompetence, unwillingness, expense, uncertainty, inconvenience or corruption of existing legal systems. By specifying a state-change system through a rich and unambiguous language, and furthermore architecting a system such that we can reasonably expect that an agreement will be thus enforced autonomously, we can provide a means to this end.

Dealings in this proposed system would have several attributes not often found in the real world. The incorruptibility of judgement, often difficult to find, comes naturally from a disinterested algorithmic interpreter. Transparency, or being able to see exactly how a state or judgement came about through the transaction log and rules or instructional codes, never happens perfectly in human-based systems since natural language is necessarily vague, information is often lacking, and plain old prejudices are difficult to shake.

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

1.2. **Previous Work.** Buterin [2013a] first proposed the kernel of this work in late November, 2013. Though now evolved in many ways, the key functionality of a block-chain with a Turing-complete language and an effectively unlimited inter-transaction storage capability remains unchanged.

Dwork and Naor [1992] provided the first work into the usage of a cryptographic proof of computational expenditure ("proof-of-work") as a means of transmitting a value signal over the Internet. The value-signal was utilised here as a spam deterrence mechanism rather than any kind of currency, but critically demonstrated the potential for a basic data channel to carry a *strong economic signal*, allowing a receiver to make a physical assertion without having to rely upon *trust*. Back [2002] later produced a system in a similar vein.

The first example of utilising the proof-of-work as a strong economic signal to secure a currency was by Vishnumurthy et al. [2003]. In this instance, the token was used to keep peer-to-peer file trading in check, ensuring "consumers" be able to make micro-payments to "suppliers" for their services. The security model afforded by the proof-of-work was augmented with digital signatures and a ledger in order to ensure that the historical record couldn't be corrupted and that malicious actors could not spoof payment or unjustly complain about service delivery. Five years later, Nakamoto [2008] introduced another such proof-of-work-secured value token, somewhat wider in scope. The fruits of this project, Bitcoin, became the first widely adopted global decentralised transaction ledger.

Other projects built on Bitcoin's success; the alt-coins introduced numerous other currencies through alteration to the protocol. Some of the best known are Litecoin and Primecoin, discussed by Sprankel [2013]. Other projects sought to take the core value content mechanism of the

Possible Uses of Ethereum

Financial contracts:

- Swaps, Options, OTC contracts, ...

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

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.

ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER
EIP-150 REVISION

DR. GAVIN WOOD
FOUNDER, ETHEREUM & ETHCORE
GAVIN@ETHCORE.IO

ABSTRACT. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not least Bitcoin. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

1. INTRODUCTION

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated, through the power of the default, consensus mechanisms and voluntary respect of the social contract that it is possible to use the internet to make a decentralised value-transfer system, shared across the world and virtually free to use. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adapted this original "currency application" of the technology into other applications albeit rather simplistic ones.

Ethereum is a project which attempts to build the generalised technology; technology on 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 hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework.

1.1. **Driving Factors.** There are many goals of this project; one key goal is to facilitate transactions between consenting individuals who would otherwise have no means to trust one another. This may be due to geographical separation, interfacing difficulty, or perhaps the incompatibility, incompetence, unwillingness, expense, uncertainty, inconvenience or corruption of existing legal systems. By specifying a state-change system through a rich and unambiguous language, and furthermore architecting a system such that we can reasonably expect that an agreement will be thus enforced autonomously, we can provide a means to this end.

Dealings in this proposed system would have several attributes not often found in the real world. The incorruptibility of judgement, often difficult to find, comes naturally from a disinterested algorithmic interpreter. Transparency, or being able to see exactly how a state or judgement came about through the transaction log and rules or instructional codes, never happens perfectly in human-based systems since natural language is necessarily vague, information is often lacking, and plain old prejudices are difficult to shake.

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

1.2. **Previous Work.** Buterin [2013a] first proposed the kernel of this work in late November, 2013. Though now evolved in many ways, the key functionality of a blockchain with a Turing-complete language and an effectively unlimited inter-transaction storage capability remains unchanged.

Dwork and Naor [1992] provided the first work into the usage of a cryptographic proof of computational expenditure ("proof-of-work") as a means of transmitting a value signal over the Internet. The value-signal was utilised here as a spam deterrence mechanism rather than any kind of currency, but critically demonstrated the potential for a basic data channel to carry a *strong economic signal*, allowing a receiver to make a physical assertion without having to rely upon *trust*. Back [2002] later produced a system in a similar vein.

The first example of utilising the proof-of-work as a strong economic signal to secure a currency was by Vishnumurthy et al. [2003]. In this instance, the token was used to keep peer-to-peer file trading in check, ensuring "consumers" be able to make micro-payments to "suppliers" for their services. The security model afforded by the proof-of-work was augmented with digital signatures and a ledger in order to ensure that the historical record couldn't be corrupted and that malicious actors could not spoof payment or unjustly complain about service delivery. Five years later, Nakamoto [2008] introduced another such proof-of-work-secured value token, somewhat wider in scope. The fruits of this project, Bitcoin, became the first widely adopted global decentralised transaction ledger.

Other projects built on Bitcoin's success; the alt-coins introduced numerous other currencies through alteration to the protocol. Some of the best known are Litecoin and Primecoin, discussed by Sprankel [2013]. Other projects sought to take the core value content mechanism of the

1

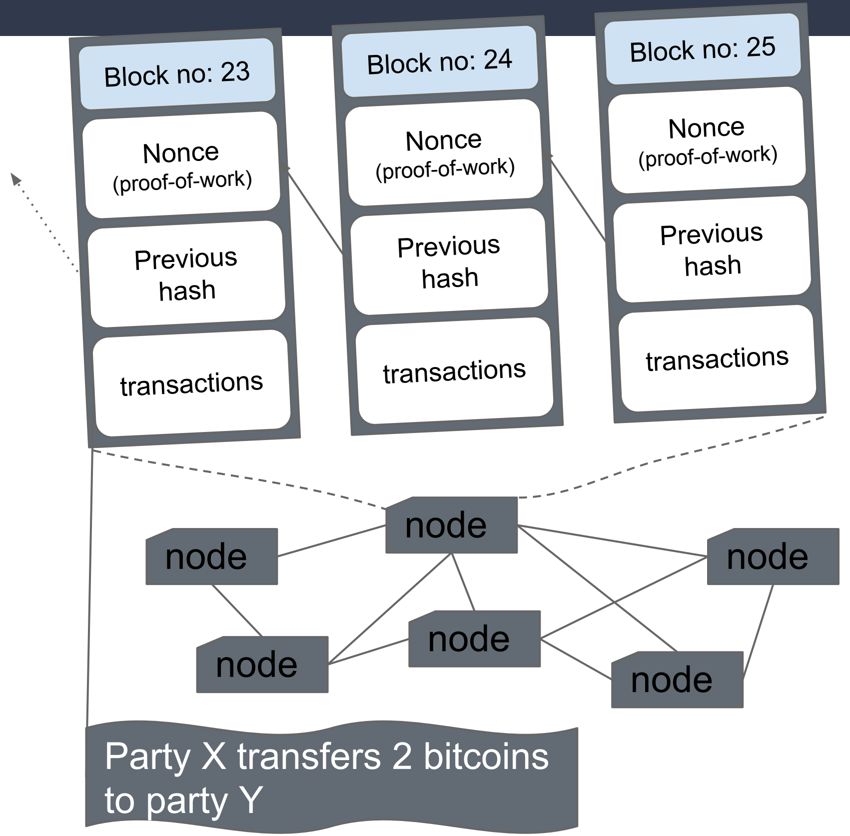
Existing Blockchain Implementations

Constructed to record mutual agreed-upon transactions…

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

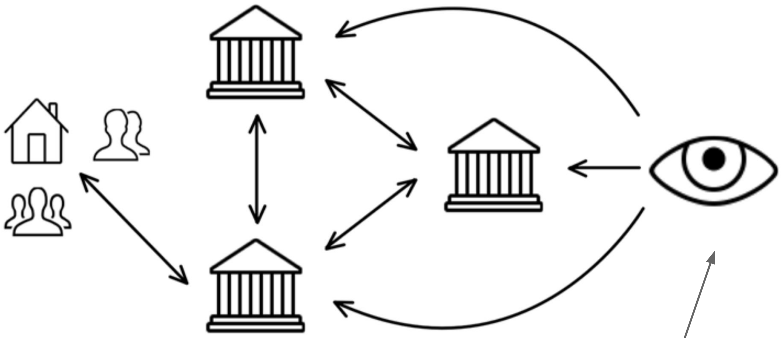
A *mutual consensus mechanism* (proof-of-work) ensures that nodes agree on transactions.

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



Today’s Financial System

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

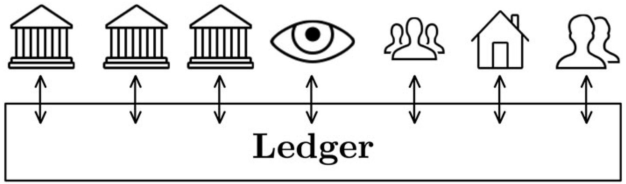


A small group of **large institutions** communicate bilaterally.

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)…

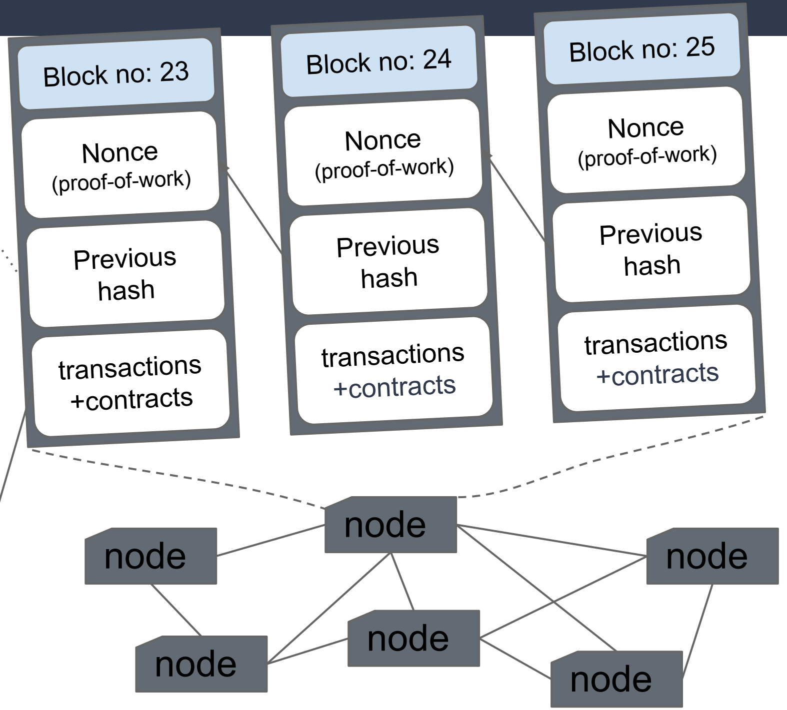
The blockchain makes evident that all involved parties have signed the contract.

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..

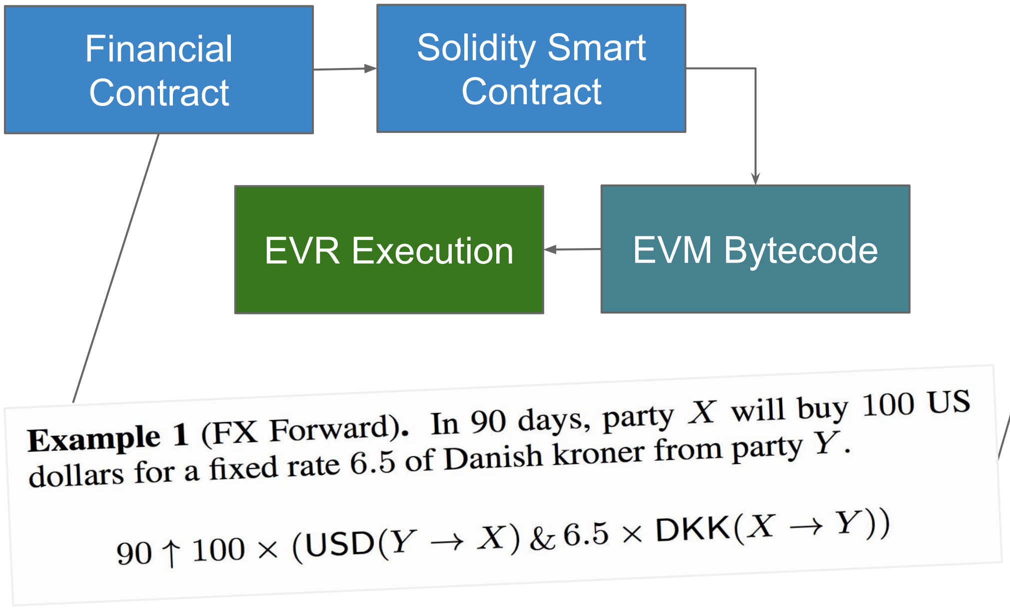
Example 1 (FX Forward). In 90 days, party X will buy 100 US dollars for a fixed rate 6.5 of Danish kroner from party Y.

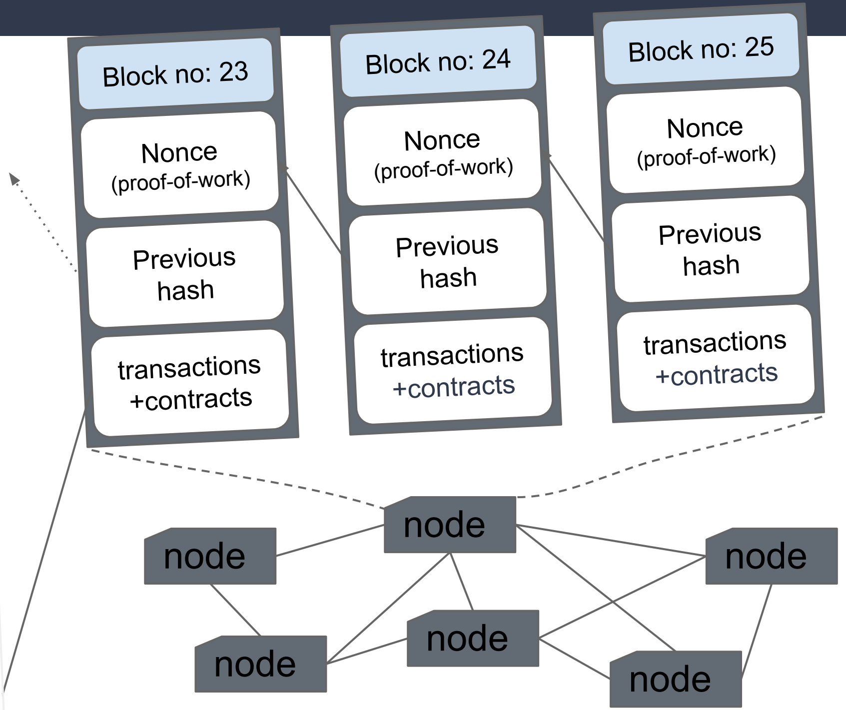
$$90 \uparrow 100 \times (\mathrm{USD}(Y \to X) \,\&\, 6.5 \times \mathrm{DKK}(X \to Y))$$



How do we Know that the Smart Contract is Implemented Correctly?

Lots of trusted components, incl:





A Certified Contract Management Engine

Contract combinators 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:

if obs(X exercises option) **within** 90 **then**

100 × (USD(Y→X) & 6.65 × DKK(X→Y))

else ∅

[2] Patrick Bahr, Jost Berthold, and Martin Elsman. **Certified Symbolic Management of Financial Multi-Party Contracts**. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP'15)*. September, 2015.

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)

obs(l,d)	observe the value of l (a label) at time d
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
let x = e **in** c	bind today's value of e to x in c
if e **within** d **then** c_1 **else** c_2	behave as c_1 when e becomes true if e does not become true within d days, behave as c_2

Expressibility: More Contract Examples

Asian Option

90 ↑ **if** obs(X exercises option) **within** 0 **then**
100 × (USD(Y→X) & (*rate* × DKK(X→Y)))
else ∅

where

rate = 1/30 · acc(λr.r + obs(FX USD/DKK), 30, 0)

Notice: the special acc-construct is used to compute an average rate.

Simple Credit Default Swap (CDS)

The bond:

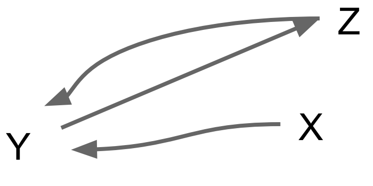
Cbond = **if** obs(X defaults, 0) **within** 30 **then** ∅
else 1000 × DKK(X→Y)

Insurance:

Ccds = (10 × DKK(Y→Z)) &
if obs(X defaults, 0) **within** 30 **then**
900×DKK(Z→Y)
else ∅

Entire Contract:

C = Cbond & Ccds



Benefits of the Formal Framework

Some contract equivalences (algebra)

$$e1 \times (e2 \times c) \simeq (e1 \cdot e2) \times c$$
$$d1 \uparrow (d2 \uparrow c) \simeq (d1 + d2) \uparrow c$$
$$d \uparrow (c1 \,\&\, c2) \simeq (d \uparrow c1) \,\&\, (d \uparrow c2)$$
$$e \times (c1 \,\&\, c2) \simeq (e \times c1) \,\&\, (e \times c2)$$

$$d \uparrow \varnothing \simeq \varnothing$$
$$r \times \varnothing \simeq \varnothing$$
$$0 \times c \simeq \varnothing$$
$$c \,\&\, \varnothing \simeq c$$
$$c1 \,\&\, c2 \simeq c2 \,\&\, c1$$

With a netting semantics:

$$(e1 \times a(p1 \rightarrow p2)) \,\&\, (e2 \times a(p1 \rightarrow p2)) \simeq (e1 + e2) \times a(p1 \rightarrow p2)$$

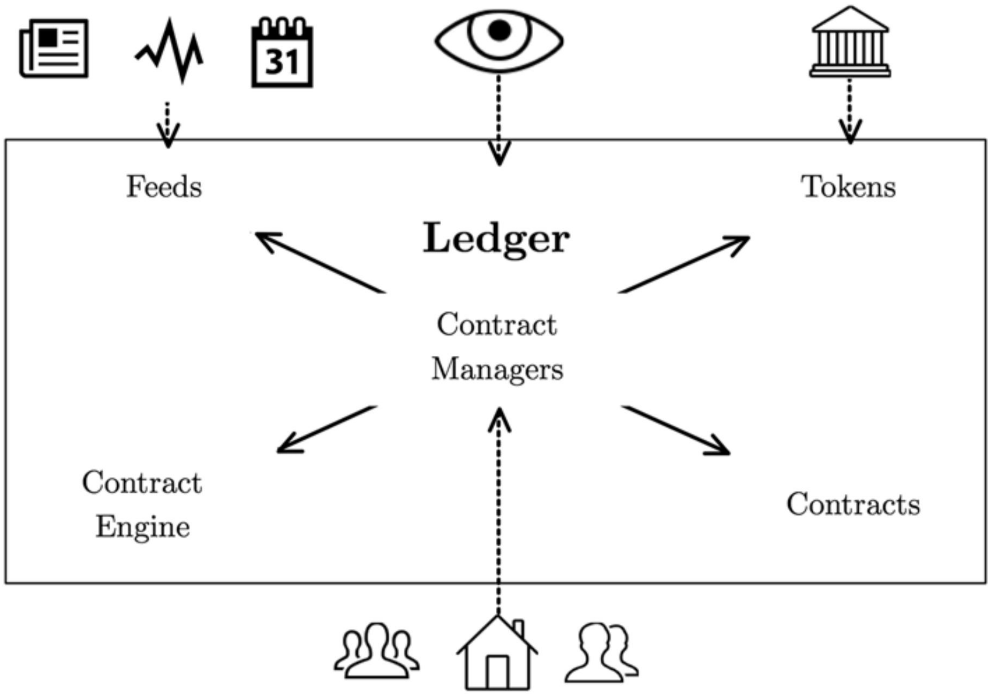
Other benefits:

- 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



Bus Inf Syst Eng
https://doi.org/10.1007/s12599-017-0507-z

CrossMark

RESEARCH PAPER

Automated Execution of Financial Contracts on Blockchains

Benjamin Egelund-Müller · Martin Elsman ·
Fritz Henglein · Omri Ross

Received: 10 January 2017 / Accepted: 16 August 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 expressing complex multi-party derivatives and is conducive to automated execution. The authors propose an architecture for separating contractual terms from contract execution: a contract evaluator encapsulates the syntax and semantics of financial contracts without actively performing contractual actions; such actions are handled by user-definable contract managers that administer strategies for the execution of contracts. Hosting contracts and contract managers on a distributed ledger, side-by-side with digital assets, facilitates automated settlement of commitments without the 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

1 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. Schneider et al. (2016) estimate savings from blockchain-based technologies to be in the region of tens of billion of US dollars annually across the financial sector with \$11–12 billion in annual savings on 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* or *blockchain*[1] 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 logbook of events shared amongst a set of non-trusting participants, 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 to a log of events. In Bitcoin (Nakamoto 2009) the built-in protocol ensures that a Bitcoin transfer can only occur from an authenticated owner, whose transaction history sums to a positive balance, where the amount transferred is at most that balance and has not already been spent. Bitcoin thus enforces a specific *contract* amongst an open-ended number of

Accepted after two revisions by the editors of the special issue.

B. Egelund-Müller · M. Elsman · F. Henglein · O. Ross (✉)
Department of Computer Science, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen, Denmark
e-mail: omriross@gmail.com

[1] The term *blockchain* arises from the technique of sequencing blocks of atomic payments between pseudonymous participants into tamper-resistant verified (implicit) asset balances, which underlies Bitcoin, an unstructured peer-to-peer system with its own virtual currency. We use the term more generally here for peer-to-peer systems without central control, but varying performance, privacy and authentication mechanisms, and for the applications conceived for and made possible by such technology.

Published online: 27 November 2017

Consequences – II

Bye-bye Banks

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!