TECHNICAL REPORT

ISO/TR 23455

First edition
2019-09

Blockchain and distributed ledger technologies — Overview of and interactions between smart contracts in blockchain and distributed ledger technology systems

Reference number
ISO/TR 23455:2019(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword ..... **iv**
Introduction ..... **v**
1 Scope **1**
2 Normative references **1**
3 Terms and definitions **1**
4 Symbols and abbreviated terms **2**
5 Overview of smart contracts **2**
5.1 History of smart contracts 2
5.2 Different ways of understanding smart contracts 3
6 Operation of smart contracts **4**
6.1 The concept of a smart contract 4
6.2 Benefits and challenges of smart contracts 6
6.3 Difference between on-chain and off-chain smart contracts regarding deployment and execution 7
6.4 Access of real-world-information for smart contracts 8
6.4.1 General considerations about real-world-interaction 8
6.4.2 One-way event delivery from a smart contract to an event consumer 9
6.4.3 Transfer of control from a smart contract to an external process 11
6.5 Life cycle of smart contracts: creation, operation, termination 11
6.5.1 Overview 11
6.5.2 Modifying smart contracts in a public BC/DLT system 11
6.5.3 Update and roll-back mechanisms supported by the underlying ledger 12
6.5.4 Migration mechanisms defined by smart contracts 12
6.6 Security 12
7 Binding and enforceable smart contracts **14**
7.1 General 14
7.2 Legal enforceability of smart contracts 14
8 Smart contracts for information transfer between blockchains (cross-chain and sidechain transactions) **15**
8.1 Introduction 15
8.2 Implementations of cross-chain and sidechain transactions 16
8.3 Importance of semantics, syntax, inputs and languages for the interoperability of smart contracts 20
Annex A (informative) **Examples of smart contract implementations** **21**
Annex B (informative) **Role of domain specific languages and methods** **24**
Annex C (informative) **Applications and smart contract use cases** **26**
Bibliography **40**

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 307, *Blockchain and distributed ledger technologies*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Smart contracts, a synonym for automated applications on blockchain and distributed ledger technology-based (BC/DLT) systems, are an important development step from early stage, purely transaction oriented blockchains to more interactive technologies where the transactions on the blockchain or distributed ledger technology system are conditional on the terms of that application. According to the current working-definition of ISO/TC 307, WG1, Terminology, a smart contract is a

"computer program stored in a distributed ledger system wherein the outcome of any execution of the program is recorded on the distributed ledger".

In specific implementations of BC/DLT systems, such a program can vary from program code interpreted on single peers to (pre-)compiled programs recorded on the ledger to be executed on arbitrary virtual machines within the system (such as miners). It should be understood that the "effects" to be recorded on the distributed ledger will usually be the transaction that is the deterministic, predefined coded outcome from the smart contract code.

As the term smart contract in its original intention as created by Nick Szabo in 1994 had a different, mainly legally oriented (precise and legitimate) meaning, this has often caused confusion regarding "legally binding intentions": As this document discusses and describes smart contracts as a technology for BC/DLT automation in general, it is also important to understand that smart contracts may have a legal binding intention. Because of this, the legal binding application and structure of smart contracts also requires understanding of legal background, context and definitions.

This document mainly describes the aspects of automated software in a BC/DLT-system.

Blockchain and distributed ledger technologies — Overview of and interactions between smart contracts in blockchain and distributed ledger technology systems

1 Scope

This document provides an overview of smart contracts in BC/DLT systems, describing what smart contracts are and how they work. It also discusses methods of interaction between multiple smart contracts. This document focuses on technical aspects of smart contracts. Smart contracts for legally binding use and applications will only be briefly mentioned in this document.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
asset
anything that has value to a stakeholder

[SOURCE: ISO/TS 19299:2015, 3.3, modified — Note 1 to entry has been removed.]

3.2
ledger
information store that keeps records of *transactions* (3.10) that are intended to be final, definitive and immutable

3.3
miner
DLT node which engages in *mining* (3.4)

3.4
mining
block-building activity in some consensus mechanisms

Note 1 to entry: Participation in mining is often incentivized by block rewards and *transaction* (3.10) fees.

3.5
off-chain
related to a blockchain system, but located, performed or run outside a blockchain system

3.6
on-chain
located, performed or run inside a blockchain system

3.7
DLT oracle
oracle
distributed ledger technology oracle
service that updates a distributed *ledger* (3.2) using data from outside of a distributed ledger system

Note 1 to entry: *Smart contracts* (3.8) cannot access sources of data external to the distributed ledger system on their own; therefore, DLT oracles act as services designed to provide trustworthy data from external sources for use by a smart contract.

3.8
smart contract
computer program stored in a distributed *ledger* (3.2) system wherein the outcome of any execution of the program is recorded on the distributed ledger

Note 1 to entry: A smart contract might represent terms in a contract in law and create a legally enforceable obligation under the legislation of an applicable jurisdiction.

3.9
token
representation of a collection of data

Note 1 to entry: In this document, token is also used as synonym for a virtual asset (3.1).

[SOURCE: ISO/IEC 14776-323:2017, 3.1.85, modified — The original Note 1 to entry has been removed; a new Note 1 to entry has been added.]

3.10
transaction
smallest unit of a work process resulting in a state change

[SOURCE: ISO/TR 26122:2008, 3.5, modified — The words "consisting of an exchange between two or more participants or systems" have been replaced with "resulting in a state change".]

3.11
trust
relationship between two elements, a set of activities and a security policy in which element *x* trusts element *y* if and only if *x* has confidence that *y* will behave in a well defined way (with respect to the activities) that does not violate the given security policy

[SOURCE: ISO/IEC 13888-1:2009, 3.59, modified — Note 1 to entry has been removed.]

4 Symbols and abbreviated terms

BC/DLT: blockchain and distributed ledger technology

DSL: domain specific language

5 Overview of smart contracts

5.1 History of smart contracts

The term "smart contract" was first introduced by Nick Szabo in the early 1990s. But it was only with the advent of blockchain and distributed ledger technology (BC/DLT) that this concept gained widespread interest.

According to Szabo, a smart contract represents the idea of automatically fulfilling *contractual clauses* by embedding these clauses in a digital entity that has control over the property dealt with. This should be done “in such a way as to make breach of contract expensive [...] for the breacher” (Szabo, 1997). Szabo therefore proposed the use of secure, machine-executable transaction protocols that ensure

automatic performance of predefined, conditional actions in accordance with the contract clauses. He saw smart contracts as a promising opportunity to significantly reduce “mental and computational transaction costs”. In his seminal paper, Szabo also described various well-known cryptographic techniques suitable for ensuring the desired characteristics of a smart contract, such as security, confidentiality and unforgeability. At that time, however, technology and market demand were not yet ready to implement these requirements comprehensively.

BC/DLT now provides capabilities that not only satisfy Szabo's demands but also offer advanced possibilities that reach beyond Szabo’s ideas. Massively distributed by design and safeguarded by a variety of cryptographic instruments (for example, transactions with pseudonymized accounts, immutability, unknown numbers of parallel executed and evaluated transactions), this environment allows smart contracts to be written in full-fledged programming languages, to communicate and interact with each other as well as with external resources, and to transparently keep track of their current state of execution. Thus, smart contracts with any relation to a legal context and its automation are actually a subgroup of all smart contracts as used for process automation on blockchains (see Figure 1). Examples for smart contract use-cases are listed in Annex C, Table C.1.

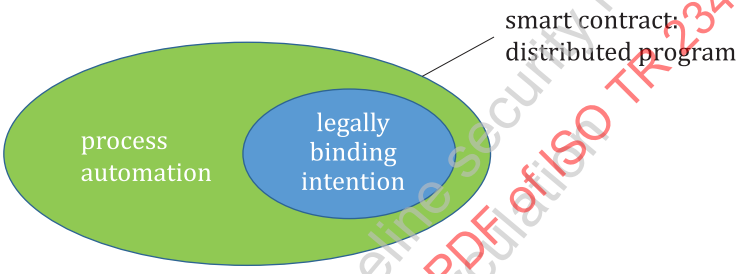


Figure 1 — Different understandings of the scope of “smart contract”.

Therefore, the smart contract concept is not limited to BC/DLT systems, smart contracts may also be used on traditional platforms (for example procurement portals); this document only considers smart contracts in the context of BC/DLT systems.

5.2 Different ways of understanding smart contracts

In the course of these technological developments, the understanding of the term “smart contract” has also evolved from its original meaning. Unfortunately, there is currently a lack of uniform understanding of the term "smart contract" in practice.

It is important to note that the term "smart contract" does not necessarily refer to a contract in the legal sense. Smart contracts can rather be taken to mean distributed applications that automate transactions by leveraging the security of DLT systems, and no implicit legal meaning should be inferred.

As already explained above, in 1996 Nick Szabo described smart contracts as being:

A set of promises, specified in digital form, including protocols within which the parties perform on the other promises[9].

In 2014, Vitalik Buterin[10] invented a new generation of smart contracts: decentralised and immutable once it exists in DLT systems. The notion of smart contracts within DLT systems is mainly developed on this new generation. For the purposes of this analysis, it is useful to consider three ways to understand smart contracts.

— **Piece of code:** According Szabo, he did not originally want to automate contracts, but instead to automate contractual clauses or exchanges:

  The basic idea behind smart contracts is that many kinds of contractual clauses (such as collateral, bonding, delineation of property rights, etc.) can be embedded in the hardware and software[11].

So it was intended to automate the evidence and consequence of contractual agreements and not the full act behind contracts and contracting; the binding and, in the event of dispute, enforcement.

— **Code as Law:** "Code is [shall be] law" has been widely used and specifically promoted by Lessig[41]. This terminology is not precise or reasonable as it blurs the distinction between "law" being "the system of rules which a particular country recognizes and enforces" and "a rule defining correct procedure (as in the laws of a game, the laws of physics, etc.)". A somewhat less contentious alternative is "the code is the (smart) contract", in which case the automation needs to include all consequences and their enforcement

— **As defined by legal professionals**: Common position of the legislators and legal professionals is the following:

The legal character of a smart contract is that which a judge or the law decides to be. So if the result of a smart contract should be enforceable, it is better to apply contractual standards to smart contracts.

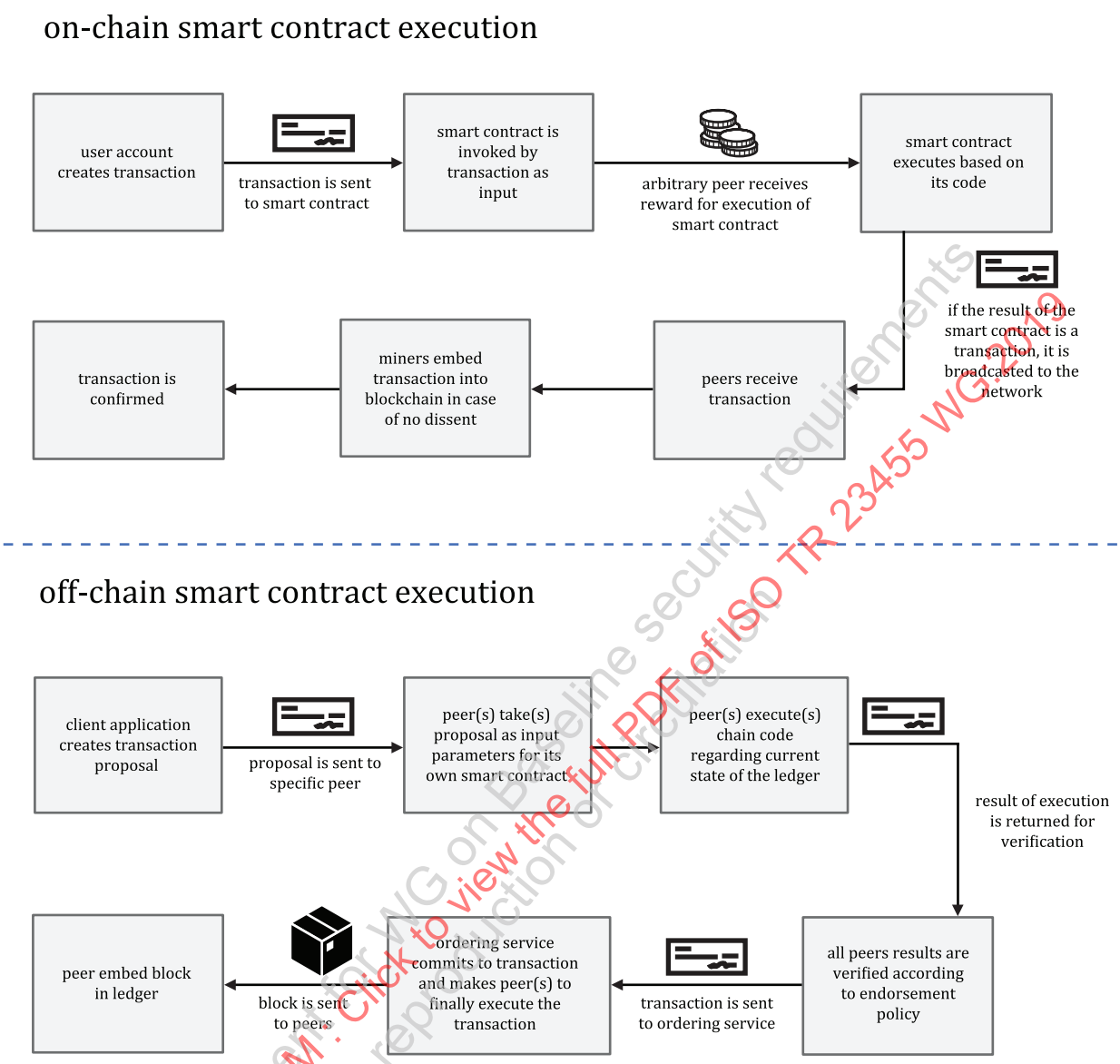
It is important to note that even a contract is not a statement of law; a contract is an agreement between parties that needs to comply with the requirements of applicable legislation including contract law.

6 Operation of smart contracts

6.1 The concept of a smart contract

DLT systems are designed to record transactions immutably into a ledger (examples for implementations are provided in Annex A). Some implementations support the deployment of custom code to the ledger, and later the invocation of deployed code with some parameters. Such deployed code is called a smart contract on a public DLT, and can itself interact with the underlying ledger such as user accounts or other smart contracts. It has to be mentioned, that the term "smart contract" is most popularly used in the context of public BC/DLT systems, for example Ethereum, in which the smart contract code may be executed with a deterministic result in an arbitrary virtual machine of a mining peer. Other concepts, for example on private DLT-systems such as Hyperledger Fabric, originally named the distributed code differently, for example "chaincode", to emphasize differences in the execution concepts (see Figure 2). Meanwhile this term is also commonly referred to as smart contract. If there are any significant differences that require description, a precise distinction will be made between on-chain smart contracts and off-chain smart contracts.

Such code is installed and executed at selected peers. Consequently the deployment, execution and consensus process of such public and private systems differ.



NOTE See further explanation in 6.3.

Figure 2 — Difference between an on-chain and an off-chain smart contract execution

Thereby, smart contracts allow participants to implement and deploy custom functionality to augment the features of a running DLT system. Deployment and execution of smart contract code itself benefits from any transparency and immutability guarantees made by the principles and methods of the underlying DLT.

As well as generating changes to the underlying ledger state, some DLT implementations also allow smart contracts to control virtual assets (tokens). The clearest example of this is the case where smart contracts control an address, account or database item to which a cryptocurrency can be sent. The balance belonging to the address can only be sent by invoking the smart contract code itself, allowing a vast array of complex transactions to be performed under the security guarantees of the DLT system.

Similarly, the control of a smart contract, representing aspects of a legal contract between two parties can restrict handling and operation of the smart contract by checking the identity of potential users of a smart contract and the smart contract owner with the identity such as registered on the BC/DLT-system. This example also shows the benefit of using underlying security measures of the BC/DLT-system to increase the security of the control of a smart contract.

Once deployed, smart contracts are subject to the rules of the underlying BC/DLT platform. These implementation-specific rules do not necessarily allow smart contracts to be upgraded, modified or cancelled. Smart contract authors need to be aware of the implications of deploying to a live system, and carefully consider whether and how to build migration or termination features into their code. Here significant differences between private and public systems respectively off-chain and on-chain smart contracts may be observed: whereas on-chain smart contracts are usually operated and handled in a transaction-like manner, usually private off-chain smart contracts are part of a dedicated peer. Here off-chain smart contracts may be exchanged, upgraded or even manipulated without any significant notice to the remaining BC/DLT-system.

In order to control real-world assets via smart contracts, interactions with the outside need to be possible. For example, a smart contract's behaviour may depend on an external event or on information from external sources. In this case, trusted third party services or physical IoT-devices (called "DLT oracles") may be used to provide the data that triggers transactions in the DLT system (passing real-world information to smart contracts), or to monitor the ledger state and perform some action under specific conditions. Such DLT oracles may also be called by smart contract code.

6.2 Benefits and challenges of smart contracts

Even though they are generally considered a symbiotic alliance, smart contracts and distributed ledgers each are technologies on their own.

As an independent concept, smart contracts may reduce transaction cost in business relationships. As they formalize contractual terms or automation processes in clear computational logic, they significantly reduce room for misinterpretation and misunderstandings or accelerate processes. Furthermore, proven smart contracts for recurring tasks can be standardized and made available in libraries for re-use, which could be especially useful in government e-procurement from authorized suppliers. In this way, the contracting parties can quickly find consensus on implementations that best reflect their contractual intentions.

However, DLT systems provide an ideal environment to exploit the strengths of smart contracts. Instead of leaving it to the contracting parties to translate their individual interpretation of an agreement independently into proprietary business logic, the distributed ledger ensures that all parties involved see the same code at all times. Moreover, due to its consensus mechanisms, a distributed ledger provides each participant with unequivocal evidence of the occurrence of relevant events and the results obtained. Any replacement or update of a smart contract results in invalidating the old and replacing it with a new smart contract and needs to be validated by network participants under the terms of the consensus mechanism, so the network ensures full transparency and unforgeability of a smart contract and its current state of execution. As a result, distributed ledger-based smart contracts may reduce counterparty risk and transaction cost, and may make third party escrow services obsolete.

However, there are still some challenges with regard to distributed ledger-based smart contracts, with risks and benefits essentially stemming from the same characteristics. There is experience of malicious actors, also using smart contracts for various kind of profiling and discrimination or for involving the user into undesirable, unethical or illegal activities.

Of course, there are the well-known and widely discussed challenges regarding DLT systems in general, such as limited scalability, low performance, lack of privacy or danger of mining monopoly. Numerous projects are working on addressing these weaknesses.

However, there are also more specific risks and open questions that are closely related to the implementation of smart contracts on a distributed ledger. These will be elaborated in more detail in the following clauses. The following are some examples of typical problem areas.

From a conceptual perspective, most business use cases will require a smart contract to interact with the wider world, thus involving the support of trusted entities. This of course seriously questions the original idea of a fully decentralized system designed specifically to do away with third parties.

From a technical perspective, once switched on, a smart contract cannot be stopped from the outside as it is distributed to an unknown number of executing and evaluating parties. Unless the stop-mechanism

is actively programmed into the code, there is not a single “plug to pull” for stopping it. But there may be important reasons to halt the execution of a smart contract, for example in case of legal objections or a security breach. Moreover, as with all computer programs, a smart contract’s behaviour depends on the input data, which makes it impossible to predict its behaviour or whether it will even terminate at all in all cases (halting problem). So what happens if a smart contract gets trapped in an infinite loop? Exhaustive algorithms can be limited in their execution by the requirement of paying for execution time. If the “execution currency” is consumed, the program automatically stops. Also, coding errors are unavoidable and their fault handling have to be covered.

From a legal perspective, smart contracts also raise many questions. For example, the “Code is law” dogma, while gaining popularity among programmers and IT professionals, is in conflict with many national law systems. It is still unclear e.g. whether a legal contract can be written in a programming language at all. If smart contract is considered as a tool, there is the question whether and when its outcome becomes legally binding. In addition, in the event of a dispute, the circumstances surrounding the conclusion of the contract also need to be taken into account when interpreting the contract.

6.3 Difference between on-chain and off-chain smart contracts regarding deployment and execution

NOTE The following text is also illustrated in Figure 2.

On-chain smart contracts usually operate in public BC/DLT systems following a three-step-process: deploy-invoke-operate.

— **Deployment**: During this phase a smart contract is made available to the blockchain system with a transaction containing the raw or (pre-)compiled code. In the deployment process this code-transaction obtains a smart contract address for later invocation or operation. The code itself is not active yet. Once being added to the distributed ledger, it is practically immutable, however, later on it may be cancelled or replaced with upgraded version just like a business offer.

— **Invocation**: In the invocation step the code is activated by sending a transaction to the ledger, calling its primarily assigned address, and potentially also transferring invocation parameters. During this step the code is loaded from the blockchain into the executing instances of the blockchain-system, for example the miners.

— **Operation**: After invocation the code is operated during the mining period usually in multiple distributed instances such as miners, consuming mining resources for operation which again and usually has to be compensated either by the smart contract itself or by the invoking instance. The result of the operation may be diverse, starting from a simple value to a new transaction up to another code deployment or invocation. After the operation the result is added to the ledger again in the typical consensus-process and made publicly available.

Usually on-chain smart contracts can also be disabled for execution by two mechanisms:

— If required for operation, the currency for compensating the operation cost on a miner can be withdrawn from a smart contract. This prevents it from being executed by miners. Recharging the smart contract with new compensation units enables the operation again.

— Additional stopping or disabling mechanisms are also usually provided. These prevent a smart contract at the original assigned address from being executed in its environment. Nevertheless, as the smart contract is recorded on the ledger, the pure binary code can be simply read from this transaction and be redeployed with a new transaction at a new address. This reactivates at least the functionality of the old smart contract.

Executing a smart contract off-chain may provide significant benefits compared with executing code natively on-chain, including:

— **Scalability**: Public distributed ledger networks are not currently suited to managing large throughput. This is problematic when running commercial smart contracts (particularly in large numbers) and where smart contracts share a network with other applications. Running smart

contracts off-chain provides scalability benefits as many smart contracts can be processed in parallel.

— **Privacy**: Privacy of the smart contracts is protected since computation occurs off-chain and transactions and/or state may be instantiated on-chain when required.

— **Security**: Exposing commercial terms on-chain increases the vulnerability of smart contract-based transactions.

— **Cost**: The computation cost of running smart contracts on-chain can become too high, particularly when execution cost is compensated with currencies coupled with the cryptocurrency itself.

Off-chain smart contracts such as used in many private BC/DLT systems are operated in the three-step-process: install-instantiate-invoke.

— **Install**: During this step a smart contract is made available to a peer in the blockchain system (no transaction is being written into the blockchain yet). The code itself is not active yet.

— **Instantiation**: In the instantiation step the code is activated on each peer of a channel (a peer has to be selected through an address; other channel members must endorse according to an endorsement policy) and instantiation parameters are (optionally) transferred.

— **Invocation**: After the code has been instantiated, it can be invoked by sending a transaction proposal to one or more peers. Endorsing peers must simulate and sign the transaction and send results to the ordering service. Afterwards, the invoking party verifies whether the results are valid (checks whether endorsement policy has been met), batches the transactions into blocks, orders them, and sends the transactions to the committing peers. The committing peers write the transaction into their local copy of the blockchain (ledger).

6.4 Access of real-world-information for smart contracts

6.4.1 General considerations about real-world-interaction

As smart contracts do not only operate and control blockchain-internal or user-triggered activities, but may also interact with real-world-information, an access to cyberphysical system needs to be provided. However, the interaction with the real-world is problematic as it provides a single point of access, making it vulnerable to

— false data,

— inconsistencies,

— security attacks.

As BC/DLT is associated with increased trust real-world data may unconsciously be associated with the same level of trust by the user.

The sources of real-world-data may be manifold and depend on the BC/DLT being used. Private systems using off-chain smart contracts directly attached to dedicated peers may operate local software directly accessing sensor data. Public systems with various randomly changing miners cannot attach a sensor to a miner but get their data as a transaction from the blockchain. This data needs to be sent from a DLT oracle, an external data creating entity, which is able to connect to a blockchain and to include the data into a transaction stored in the ledger. Besides physical sensors, such data creating entities can also be a website providing information such as weather services, news, aircraft arrival times or similar.

Faulty data can be delivered by external entities such as sensors or complete cyberphysical systems. The provision of such data cannot be prevented by the BC/DLT system or a smart contract but may be detected, for example by applying redundancy to the sensor system.

Inconsistencies may be caused by using multiple sensors for the same measurement which are delivering different values exceeding an expected range of deviation. This exceeded deviation may be caused by

various reasons such as: physical measurement deviations or scattering, time dependent measures causing different values for unsynchronized DLT oracles or data sources with counters creating access-depending indices never delivering the same data set of values over time.

Additionally sensors or DLT oracles are usually singular entities and therefore provide security leaks as single-points-of-attack. Consequently sensors may be replaced by others and be subject to false-data-injection, destruction or similar.

An overview of potential design criteria of an optimized sensor/DLT oracle/smart contract architecture is shown in Figure 3:

— Sensors are redundant and control and secure their values mutually by plausibility checks and deliver a harmonized value to the DLT oracle.

— To prevent security leaks by single-points-of-attack, DLT oracles are redundant. Additionally, they could monitor a spatially more extended sensor system from different local positions to avoid or detect inconsistencies and contradictory measurement. The redundant DLT oracle network delivers the data to the BC/DLT-system or the processing application, respectively.

— The smart contract relies on authoritative real-world data recorded in BC/DLT-system or is able to discover and to harmonize contradictory values itself.

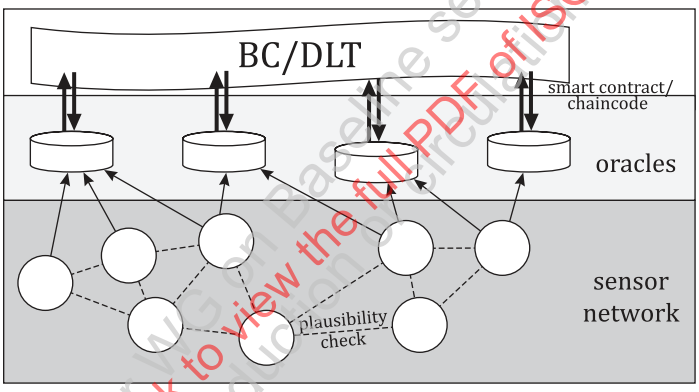


Figure 3 — Extended architecture for a more secure sensor connection to a BC/DLT-system via DLT oracles/interfaces

To increase trust in security and authenticity, DLTs generally require that all data is verified, in some sense. In the case of a transfer of value between two parties, the sender's address, the receiver's address and the amount to transfer are all signed by the sender before the transaction is transmitted to the network. Using only the data contained in the signed transaction it is straightforward for all nodes to verify that the controller of the sender's private key has authorized this operation on his or her data. All observers can then make this check and update the ledger in a deterministic way based on the data.

Digital signatures are an important tool in convincing validators of the veracity of real-world data. In particular multi-sig techniques can provide strong evidence, requiring that multiple parties sign (and thereby approve) any given value. Similarly, several DLT oracles could exist for the same data, allowing validators to consult multiple independent sources and ensure that they agree with each other. For the digital signatures to provide meaningful authenticity a secure binding of the real-world-identities to their signature keys is also needed, for example by means of a public key infrastructure (PKI).

6.4.2 One-way event delivery from a smart contract to an event consumer

In this scenario (see Figure 4), an event occurs within a smart contract, for example, a state change or the execution of a specific type of transaction. This event is broadcast to downstream consumers, and those consumers then take appropriate actions.

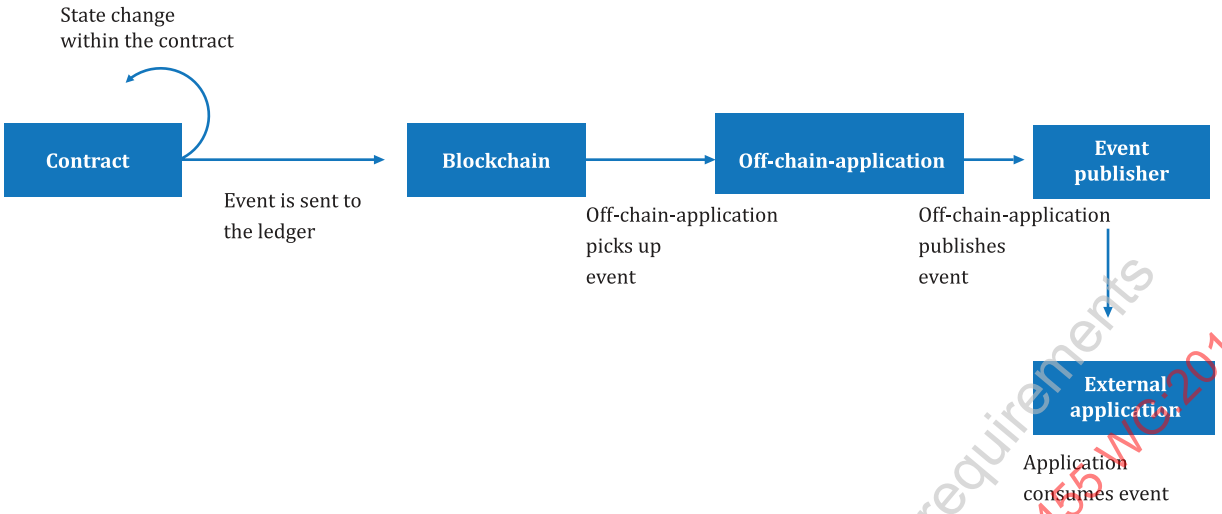


Figure 4 — Process of a one-way event delivery from a smart contract to an event consumer

A reciprocal scenario is also possible: in this case, an event is generated by a sensor or an external system and the data from that event is to a smart contract (see Figure 5). A common example is the delivery of data from financial markets, for example, prices of commodities, stock, or bonds, to a smart contract.

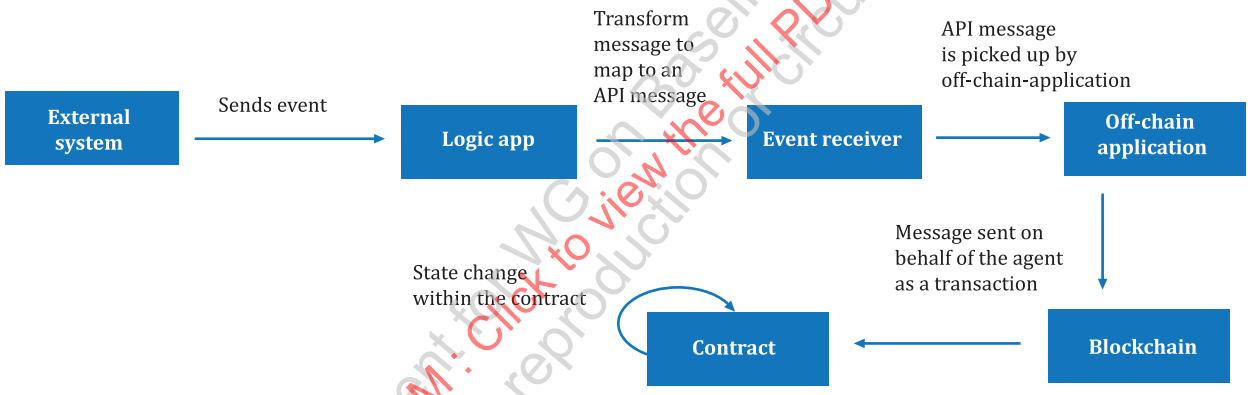


Figure 5 — Process of a one-way event delivery from an external event producer to a smart contract

In this model, the communication to the smart contract and subsequent state change occurs according to the following process:

— Upon reaching the completion or a specific milestone in the external code execution, an event is sent to an off-chain code.

— For systems that can't be directly adapted to send a message that conforms to the expectations of the API, it will be transformed.

— The content of the message is sent to a specific function in the smart contract.

— The function executes and typically modifies the state. The change of state moves forward the business workflow reflected in the smart contract, enabling other functions to now be executed as appropriate.

6.4.3 Transfer of control from a smart contract to an external process

This pattern is typically implemented using the following approach:

— The smart contract moves to a specific state. In this state, either no or a limited number of functions can be executed until an external system executes a desired action.

— The change of state is sent as an event to a downstream consumer.

— The downstream consumer receives the event and triggers external code execution.

— After execution the external code transfers the execution result to the ledger and returns control back to the smart contract.

6.5 Life cycle of smart contracts: creation, operation, termination

6.5.1 Overview

The details and rules depend heavily on the architecture of the underlying DLT platform. For a public BC/DLT-system in general smart contract code is initially deployed by means of a transaction which records it in the ledger. It can then be uniquely referenced and invoked by future transactions.

Changes to deployed smart contract code are generally only possible according to the rules of the underlying ledger. Either migration mechanisms are built into the smart contract code itself, or the underlying DLT platform may support update of code and data under certain circumstances.

Update mechanisms of the underlying DLT platform could include features to amend the results of transactions (discussed in later clauses), or special smart contract (de)activation transactions. In general terms, blockchain and DLT systems do not normally provide direct support for rolling-back transactions. A major objective of these systems is to prevent tampering or modification of the transaction history, as an audit record. Rolling back a transaction might "undo" parts of the transaction history, subverting that objective.

This immutability has obvious security benefits but may also present problems. Bugs may be discovered in smart contract code after it has been deployed, or legal concerns may warrant a change to the smart contract behaviour. Since there is no easy way to update the smart contract code, verification methods may provide assurance as to code behaviour. This is important when it comes to deploying smart contracts into immutable ledger spaces.

6.5.2 Modifying smart contracts in a public BC/DLT system

Direct changes to the ledger are usually not intended in BC/DLT-systems and therefore not directly applicable. Therefore the simplest form of a modification may be an inverse transaction to a previous transaction. This will not undo or delete the respective entry but will eliminate the cumulative outcome. This also implies leaving intact all previous versions the data. As a smart contract in a public BC/DLT-system is technically also recorded in a transaction, the same methods apply for smart contracts as for data.

Depending on the architecture, purpose and design of a BC/DLT-system, more sophisticated techniques may also exist to edit blockchain histories and even hide smart contracts or data that have been previously committed to the ledger.

Public blockchains are usually explicitly designed with the goal of having no central authority. In this case, it appears impossible to use the above-mentioned approach. Therefore especially public blockchains do not support methods to change smart contract behaviour or even to undo previous activity and remove its results from the ledger.

However, it is possible for blockchains to fork their transaction history, creating two blockchains sharing a common initial history before the forking point. In principle this is one mechanism by which

a BC/DLT-system could attempt to modify parts of a transaction history, giving the community the option to adopt any of those forks

6.5.3 Update and roll-back mechanisms supported by the underlying ledger

DLT implementations can include functionality whereby some entity or entities may be authorized to make changes to the ledger state. These special permissions represent a degree of centralization of the system (in the sense that users need to trust that the permission holders will not abuse their power) and may not be suitable for some DLT applications. However, such a mechanism may be desirable as a way to support smart contract bug fixes, resolve legal issues, and even allow previous transactions to be undone.

As a very simple example, a private blockchain used by a group of institutions could be configured to allow arbitrary ledger changes to be made only if such changes are signed by a pre-agreed subset of the stakeholders. The cost for stakeholders to inspect, agree to and digitally sign some change may be made quite high (intentionally or otherwise), and regular use of such privileges is likely to undermine some of the original security features of DLT. However, as a means of fixing security issues or rolling back the results of some illegal activity, such a mechanism may have a place in certain implementations. It is possible to imagine many other approaches.

A simple but effective safety mechanism supported by some DLT systems is deactivation transactions, which can prevent or later allow execution of specific smart contract code. These transactions mark the code in a DLT-specific way to be non-executable. Note that this in itself neither removes the code from previous blocks, or prevents it from being redeployed or reactivated. Another possibility is a "smart contract update-by-forwarding-mechanism which allows the smart contract creator to nullify an existing smart contract by pointing execution to an updated version of that smart contract located at a new smart contract address. Transactions could be forwarded to both new and old smart contract addresses; sending a transaction to the outdated smart contract address would only transfer control flow to the current version of the smart contract.

6.5.4 Migration mechanisms defined by smart contracts

In the case that no central authority exists and changes to smart contract code are not permitted by the underlying ledger, a deployed smart contract may not be overwritten, and the data it manages may not be changed, except by invoking the smart contract itself.

The smart contract may include migration mechanisms in its own code. That is, a smart contract may contain code to transfer all assets and data under its control to some other smart contract and deactivate itself. Naturally, care needs to be taken to ensure that this mechanism can only be triggered in response to some specific conditions (such as a transaction of a specific form, accompanied by some appropriate set of signatures).

Smart contracts can implement application-specific logic, creating a small closed-world on the blockchain or DLT. Within that controlled context, a smart contract might in principle implement any kind of logic, including support for rolling back the effects of previous transactions in that controlled context. The limitations of this are the complexity of implementing integrity-preserving roll-back operations, and the significant challenge of ensuring referential integrity with external systems or other smart contracts outside that closed world. For a single blockchain, there can only be one transaction history at any one time.

6.6 Security

Smart contract security cannot be considered in isolation, but instead needs to be considered in the context of the underlying capabilities of the blockchain or DLT platform, and broader system security requirements. “Security” is traditionally understood as a combination of confidentiality, integrity, and availability. Sometimes other properties are also considered, such as Privacy or Non-Repudiation. Security is applicable to whole systems. A blockchain or DLT never constitutes an entire system by itself, not least because user interfaces and facilities for cryptographic key management are inevitably separate. A system that uses blockchain or DLT as a component may have security requirements that

are supported by the blockchain or DLT. As part of a blockchain or DLT, smart contract infrastructure and specific smart contract instances can support (or put at risk) security requirements for the broader system. Each of the classic security properties are discussed below.

Smart contracts by themselves cannot normally impact confidentiality properties for information stored on a blockchain. In a blockchain, all nodes have access to the entire transaction history, which includes the contents and results of the execution of all smart contracts. A smart contract may have private local variables which are only able to be modified by that smart contract, and the values of which might not be displayed through the smart contract interface. Nonetheless, all nodes will have full access to the internals of the smart contract execution engine as it is recorded in the so-called storage of the smart contract in the ledger, and so will be able to see the values of all smart contract variables. Information stored in a smart contract variable or in a blockchain transaction may be encrypted. If so, this information will not be able to be decrypted by the smart contract whilst maintaining the desired confidentiality that led to the use of encryption as the decryption key would need to be accessed by the smart contract which will be publicly accessible. Consequently, provision of controlled access to that encrypted information will need to be provided, possibly outside the scope of the smart contract itself.

More sophisticated mechanisms have been proposed for private smart contract execution using homomorphic encryption or zero-knowledge proofs. These are not yet widely understood and implemented, but may enable smart contracts to directly support Confidentiality of information stored on blockchains.

However, smart contracts might be used to implement access control logic supporting system-level Confidentiality requirements, either for other components in the broader system, or for the control of distribution of information within a DLT which limits visibility of transactions to specific parties of interest. In either case, the smart contracts support Confidentiality by correctly implementing access control logic, which for the smart contract itself can be considered to fall under Integrity issues discussed below.

Availability properties for a system can be impacted by smart contract execution, if a smart contract unexpectedly failed to terminate within a specific timeframe. The “halting problem” is famously undecidable, so it is not possible to automatically determine for any smart contract written in a Turing-complete language whether that smart contract will terminate. For public blockchain or DLT systems, termination of smart contracts is dealt with by either limiting the time available to execute a smart contract (such as with the “gas” concept in Ethereum[8]), or by using languages which are not Turing-complete where all smart contracts must terminate. Private blockchain or DLT systems may use similar protections, or may leave the issue of termination to be ensured by programmers of each specific smart contract.

Of the classic security properties, Integrity can be most directly supported by the proper use of smart contracts. The seminal security policy model for integrity in commercial information systems is the Clark-Wilson model[15]. In this model the integrity of a system is maintained if constrained data items are ensured to always (initially, and after all transformation procedure transactions) be valid (that is, to ensure some defined integrity verification procedure will be satisfied for the system), if all transactions are only performed by authorised parties, and if there is an appropriate separation of duties maintained between users. In a blockchain or DLT, a constrained data item will either be the protected part of a normal blockchain transaction (such as records of cryptocurrency holdings) or the internal state of a smart contract. A transformation procedure will be either the protected part of a normal blockchain transaction (such as transfer of cryptocurrency) or a smart contract invocation executed within a transaction. A good blockchain or DLT platform will ensure that the internal state of a smart contract cannot be modified except through execution of that smart contract. There is no inherent integrity verification procedure for a smart contract, but one could be manually implemented as part of a smart contract function. Similarly, there is often no inherent support for authorised use of smart contracts, and so authorised access control and separation of duty policies would have to be manually implemented as part of smart contracts. For example in Ethereum, every smart contract must explicitly implement an initial check for authorised invocation, otherwise the smart contract would be able to be invoked by anyone.

7 Binding and enforceable smart contracts

7.1 General

In the context of smart contracts, standardization efforts address, how smart contracts are written, how they are enforced, and how to ensure that the automated performance of a smart contract is faithful to the meaning of any relevant contractual documentation[16].

This clause explores binding and enforceable features of programmable smart contracts enabled by DLT systems. Smart contracts have existed for more than two decades, but the type that are described in the earlier clauses require particular explanation, in order to make sense of their effect, consequences, and enforceability.

This clause is not intended to be an exhaustive statement of either the law or the attributes of contracts or smart contracts. It is important to note that some smart contracts are not intended to result in binding contracts in the legal sense; but, such a smart contract may still give rise to enforceable obligations. These issues may be treated differently from country to country.

Meanwhile, at the time of writing, different jurisdictions are grappling with, enacting, or repealing different legislative provisions to regulate the use of DLT systems in different contexts. For example, smart contracts that underpin transactions in initial coin offerings (ICOs) may be completely unacceptable in some jurisdictions, while a smart contract that handles intra-institutional banking and other financial transactions may be quite acceptable, in the same jurisdiction or elsewhere. This is important in light of the decentralized, global nature of the internet and public blockchain networks.

Already some first attempts to regulate smart contracts are appearing. Just like with other innovative technologies, lawmakers' approach towards smart contracts is based on the universal principle that legal validity of a smart contract cannot be denied solely because of its smart contract form, unless applicable jurisdiction explicitly requires another form of performing certain transactions.

7.2 Legal enforceability of smart contracts

Can a smart contract be regarded as a contract or a part of a contract that is binding and enforceable in law?

It is important to remember that for civil law countries the law is something created and, of importance for consideration in the context of blockchain and DLT, changed by legislators (in the jurisdiction for which they are responsible) and subsequently interpreted by courts and other formally recognized dispute resolution bodies in the legal system. For common law countries in addition to legislators creating the law, often the courts create law. It has to be mentioned that ISO, or any other standards body, is neither a legislator responsible for setting law nor a recognized and approved interpreter of that law. As such, it does not make statements concerning the law or its interpretation.

A contract is a legally binding agreement containing mutuality of obligation (offer and acceptance rather than simply an offer without acceptance), definitive terms and a consideration. The term "legally binding" means that the contract is enforceable under law through the legal system (which may involve alternate dispute resolution rather than a court of law). Therefore, when a contract, smart or otherwise, is "legally binding" it does not mean the contract is law or can be interpreted as law; it is the enforceability that is the key consideration.

A component of smart contracts which is often contended is its legal enforceability. More broadly speaking, the question is whether smart contracts are legally enforceable, and if so, which legal jurisdiction applies. In order to understand this, it is necessary to contrast traditional contracts with smart contracts.

With respect to traditional contracts, there are three distinct, universally accepted characteristics which need to be satisfied for a contract to be legally enforceable. In short:

— the parties need to engage in an offer that is then accepted in exchange for valid consideration;

— consideration is the exchange of a benefit between the parties;

— the terms and conditions may be implicit in the offer and acceptance.

Indeed, the three aforementioned characteristics are implicit in smart contract models. Take for example, the following transaction involving an ERC-721 token in Ethereum:

— an ERC-721 token is offered to the public;

— consideration exists where an individual or organisation willingly exchanges a cryptocurrency for the token on offer; and

— acceptance occurs where the network confirms the transaction and it is propagated across all ledgers.

Provided both parties have entered into the transaction willingly, that is, without undue pressure or duress, the smart contact is enforceable. The smart contract will be executed in accordance with its agreed characteristics. In addition to templates such as ERC-20 for Ethereum, further methods to support software developers in programming smart contracts are domain specific languages (DSLs). Examples for DSLs are listed in Annex B.

Accordingly, the question of which jurisdiction applies to the smart contract will normally arise where there is a dispute as to its execution. At the time of writing, there is limited legal practice around specific jurisdictional issues of smart contracts in case of a dispute between the parties. Corresponding judicial and legislative guidance could be based upon the existing practice for traditional contracts for example by including in the terms of smart contracts dispute resolution clauses.

A valid concern may arise where developers operate out of several countries, and accordingly, several jurisdictions. In the absence of judicial guidance, little certainty can be had to the prevailing jurisdiction. In this instance, it may also be necessary to consider international statutes, which address inconsistent laws. This will ultimately help identify the prevailing jurisdiction.

However, the absence of explicit state or international laws does not provide parties with an excuse to act nefariously. The common position of most jurisdictions is that parties cannot contract out of the jurisdiction of their State to the inequitable detriment of other parties.

As a result, it is of paramount importance that all parties to a smart contract, especially developers, recognize the operation of the law.

For further guidance on this topic, it is recommended to peruse the Technical Specifications from ISO/TC 307 for legally binding smart contracts.

8 Smart contracts for information transfer between blockchains (cross-chain and sidechain transactions)

8.1 Introduction

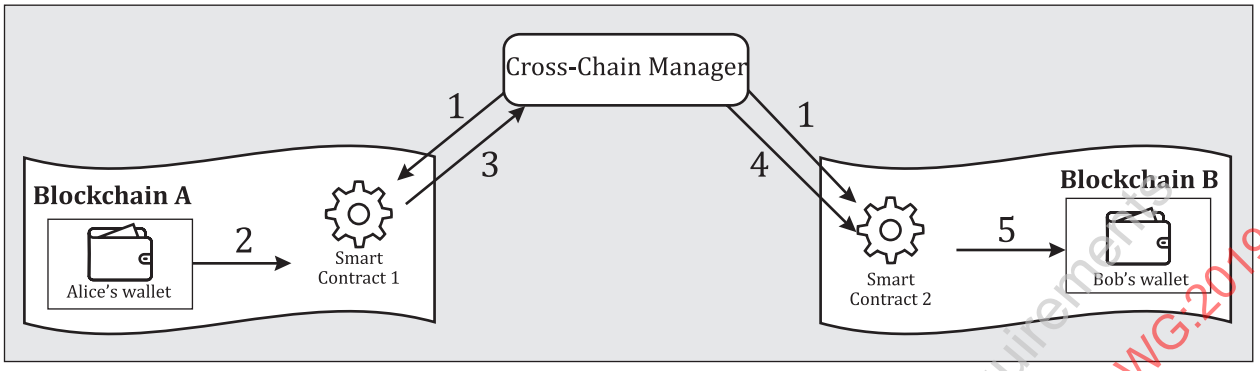
Hileman and Rauch[17] identified that the current lack of interoperability between different blockchain frameworks is a major business concern. The authors proceed to state that interoperability generally falls into one of two major categories:

— cross-chain interoperability; and

— enterprise system integration/interoperability.

This clause will only deal with the first category as the second concern is outside of scope though it does need some addressing from a standards perspective. It involves the technical connection between separate blockchains that facilitate cross-chain communication, interaction and value transfer (Figure 6). The authors continue by stating that there are several ways being explored to implement interoperability between separate blockchains: common interchain messaging protocols. This

approach involves the development of a common inter-chain messaging protocol, based either on a (to be) standardized protocol or an emerging industry framework or application.



Key

1 deploy trusted smart contracts
2 send transaction for locking to smart contract 1
3 notifying Cross-Chain Manager about successful lock
4 send transaction to smart contract 2
5 create transaction on blockchain B

Figure 6 — General principle of cross-chain management using interchange instances

Major challenges for cross-chain-transactions are

— the avoidance of double-spending, and

— the general trust into the link between transactions on two different chains.

By double-spending cross-chain-transactions persist on both chains without being locked and can be used for further transactions. Smart contracts or equivalent software may require a trusted entity, for example a cross-chain manager, managing the locking and unlocking mechanisms. In such a scenario, this instance could deploy a smart contract or equivalent to each system, assuming that the smart contracts are now considered as trusted. For cross-chain-transactions, the smart contract in blockchain A locks a transaction and notifies the cross-chain-manager about the successful lock. The cross-chain-manager again sends the transaction to blockchain B and creates an equivalent transaction there.

Such a procedure raises different challenges: for example the locked transaction on blockchain A may be permanently lost if the transfer process between A and B fails. Additionally, the problem of exchange rates needs to be resolved between A and B. Furthermore, although the cross-chain manager must not fail, a clearing system may be useful for the cross-chain manager to work effectively in the event of failures.

The second problem as indicated above is the trust: as there is no semantic connection between the data of blockchain A and B no trust in terms of a consecutive chain of transactions to a common (Genesis) block can be provided. The trust of the correctness of the transactions between A and B is imposed to the cross-chain manager.

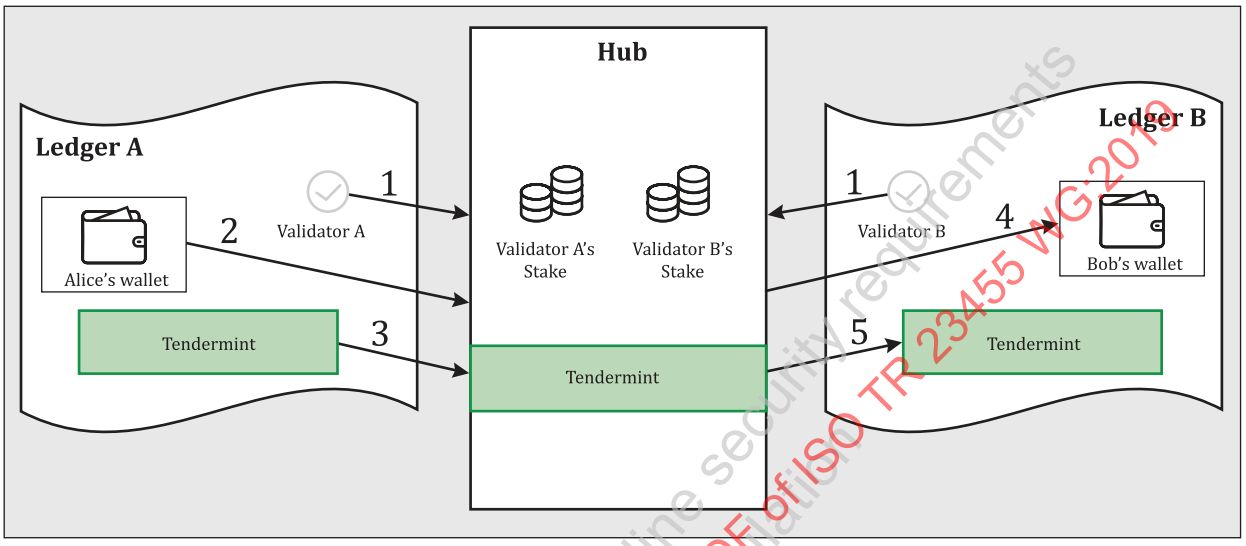
8.2 Implementations of cross-chain and sidechain transactions

Examples of current implementations for cross-chain and sidechain-transactions could be realized by using approaches like

— Ripple's "Interledger Protocol"[18].

— Hyperledger Quilt project which is derived from Ripple's "Interledger Protocol".

— The Cosmos "Network Zones Project"[19] which involves a network of independent blockchains called zones which are powered by the Tendermint Core, that is according to its authors is a high-performance, consistent, secure PBFT-like consensus engine, where strict fork-accountability guarantees a hold over the behaviour of malicious actors. The genesis zone is identified as the Cosmos Hub (Figure 7), which is a multi-asset proof-of-stake cryptocurrency with a simple control mechanism that enables the network to adapt and can be extended by connecting other zones to the network.



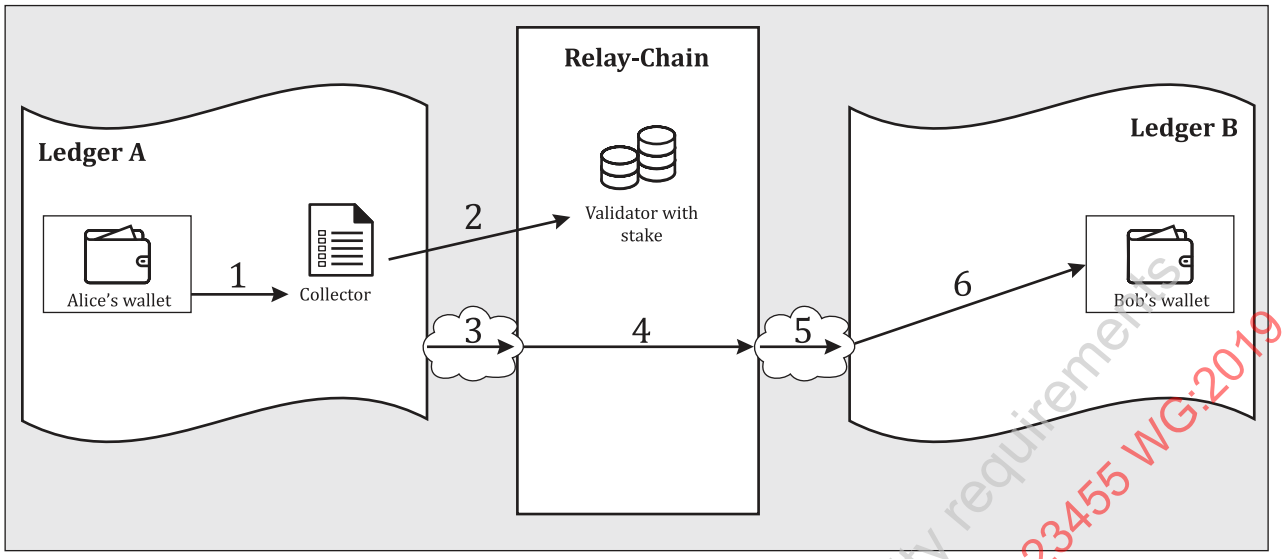
Key

1 validators put stakes on hub
2 send transaction from Alice to Hub
3 send recent block-hash of ledger A and Merkle-proof of transaction to Hub
4 send transaction from Hub to Bob
5 send recent block-hash of Hub and Merkle-proof of transaction to ledger B

Figure 7 — Example of blockchain interconnectivity using central hubs between two chains such as Cosmos.

— The Polkadot project which according to its author[20] is a complimentary protocol that will allow different blockchains to leave their silos and interact seamlessly (Figure 8). The Polkadot white paper is not according to its author a specification but is a starting point description of the core protocol. To quote the author:

"Polkadot is a scalable heterogeneous multi-chain. This means that unlike previous blockchain implementations which have focused on providing a single chain of varying degrees of generality over potential applications, Polkadot itself is designed to provide inherent applications functionality to all. Rather, Polkadot provides the bedrock 'relay-chain' upon which a large number of validatable, globally-coherent dynamic data-structures may be hosted."



Key

1 collect transaction from ledger A to ledger B
2 collector from ledger A proposes block to validator from Relay-Chain
3 queue transactions from ledger A designated to ledger B in output-queue of ledger A
4 forward transactions designated to ledger B to input-queue of ledger B
5 queue transactions from Relay-Chain in input-queue of ledger B
6 send transactions to Bob

Figure 8 — Blockchain interconnectivity using relay chains such as Polkadot

— API Calls and Middleware Layers: This methodology involves the use of API calls and middleware layers which constitutes a common way to connect distinct platforms and networks. Small services ferry messages between distinct and separate systems. However, it involves the translation between different data structures and cryptographic validation techniques, which requires the presence of trusted third parties as validators and gatekeepers. This approach does not render networks truly interoperable in a trust-minimised fashion. Despite this failure it could be an approach to assist in interoperable communications between blockchains.

— Sidechains[21][22] according to Dilley et al: Sidechains are blockchains that allow users to transfer assets to and from other blockchains. At a high level, these transfers work by locking the assets in a transaction on one chain, making them unusable there, and then creating a transaction on the sidechain that describes the locked asset. Effectively, this moves assets from a parent chain to a sidechain.

The authors explain that an effective way to implement this as follows:

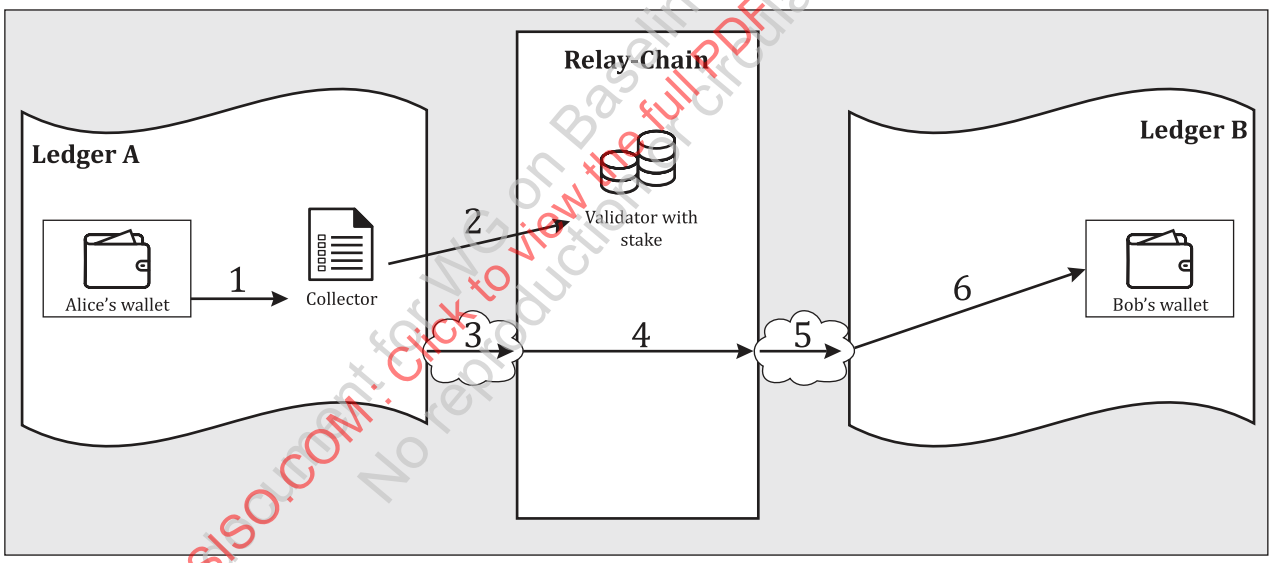
1. The user sends their asset to a special address that is designed to freeze the asset until the sidechain signals that asset is returned.
2. Using the "in" channel of a federated peg, the user embeds information on the sidechain stating that the asset was frozen on the main chain and requests to use it on the sidechain.
3. Equivalent assets are unlocked or created on the sidechain, so that the user can participate in an alternative exchange under the sidechain rules, which can differ from the parent chain.
4. When the user wishes to move her asset, or a portion thereof, back via the "out channel", she embeds information in the sidechain describing an output on the main blockchain.
5. The strong federation reaches consensus that the transaction occurred.

6. After consensus is reached, the federated peg creates such an output, unfreezing the asset on the main blockchain and assigning it as indicated on the sidechain.

— The most generalization of inter-ledger or cross-chain interoperability are overlaying operating systems (Figure 9) such as Quant Network's Overledger[23]. Such an OS operates as a vendor-independent wire-level protocol for message-oriented middleware. By decoupling the transaction layer with a shared messaging layer.

Hereby, general-purpose applications can sit on different ledgers at the same time, with the ability to communicate with each other. In fact, such a system is an "over layer" on top of existing blockchains which applications can run on. Thus, the applications can communicate, migrate and exchange information and value regardless of the ledgers on which they have been deployed. This allows users to run applications, smart contracts, treaties or move data across different blockchain technologies. This original approach will empower the adoption of blockchain technologies across various sectors and use cases enabling the large-scale adoption of the technology without tying it to a particular vendor or chain.

By taking an "overlay" operating system approach, the transferability of smart contracts across multiple blockchains is possible without forcing or limiting the execution methods. Blockchain operating systems provide the underlying framework to run and execute multi-chain smart contracts. Overledger provides the foundation to build smart contracts in an agnostic modular manner. Ensuring an open, transferable and interoperable approach to smart contracts across different technologies and blockchains.



Key

1 collect transaction from ledger A to ledger B
2 collector from ledger A proposes block to validator from Relay-Chain
3 queue transactions from ledger A designated to ledger B in output-queue of ledger A
4 forward transactions designated to ledger B to input-queue of ledger B
5 queue transactions from Relay-Chain in input-queue of ledger B
6 send transactions to Bob

Figure 9 — Blockchain interconnectivity using connector software between two chains such as Interledger

8.3 Importance of semantics, syntax, inputs and languages for the interoperability of smart contracts

A key factor to interoperability and smart contracts is leveraging standards around the development, review and implementation of those smart contracts. Smart contracts are written using programming languages, where the permissible languages are limited only by the distributed ledger framework(s) chosen; the code to work with the inputs, outputs and logic, when not leveraging standards, can be written in many different ways, with accompany challenges to initial quality, review, and ongoing maintenance.

The option to select the programming language of choice for the developers, when not limited to one or two by the chosen framework, leaves the developers free to design the smart contracts and call the input and output variables whatever they want within the constraints of the language. While this makes it relatively simple to develop the initial code, it makes it more difficult for others to understand the code and compare it with the business requirements.

This is not a problem new to blockchains and distributed ledgers, but the developer community is actively developing new chains with little reference to decades of experience from standards organizations to deal with this issue, from traditional EDI (electronic data interchange) efforts to more recent groups collaborating on data exchange standards. Leveraging existing data standards in referencing input and output data can provide guidance to the developer and simplify review by business owners and others. In addition, these data standards will simplify any ongoing maintenance of the code, especially in response to data information needs changing, driven by those same standards.

ISO is responsible for a wide variety of semantic standards, useful in the exchange of information to and from a smart contract. At a foundational level, standards such as (spoken)_language code (ISO 639-1), region codes (the ISO 3166 series), currencies (the ISO 4127 series), and date formats (the ISO 8601 series) are not a requirement in every programming language, but disambiguate key pieces of information to simplify understanding and feeding the system or consuming the output. Knowing that if 31-December 2022 is written 2022-12-31 rather than 31-12-2022 or 12-31-2022 or 31-Dec-22 can simplify the exchange of information.

Other standards-development bodies have been working for decades to facilitate the exchange of information in areas such as trade facilitation (UBL, UN/CEFACT), accounting and business reporting (XBRL International), and enterprise data modelling (OAGIS). Along with semantic agreement, these specification bodies also provide guidance on the exchange syntax, such as XML Schema, XBRL (as an extension of XML Schema) or JSON.

This abstraction of data requirements from the blockchain and proprietary smart contracts can be of even more importance in situations where and when the impact of smart contracts extends beyond a single smart contract. In some blockchains (for example Hyperledger Fabric), the user is really invoking smart contract code on every normal request to the blockchain; they are not communicating directly to some native blockchain functions. Thus, the format of the data they get back (or the format understood on input) is not related to the native blockchain itself at all; it is based on the smart contract code that the blockchain owner wrote and installed.

In addition to simplifying the references to input to and output from a smart contract, abstracting the logic itself from the programming language can have significant benefit. Leveraging the standardized syntax and semantics, calculations, validations, verifications, rules, assertions and formulae can be designed in a standardized fashion independent of the programming language, facilitating review and reuse, amongst other benefits.

The market is looking to smart contracts to function on behalf of the market in a new and exciting fashion. To efficiently create, review, approve, modify and update smart contract programming, it is important to simplify how business owners and developers collaborate, and facilitate review for those with governance oversight. The market is wondering how to ensure that developers share the fiduciary responsibilities of smart contract development with other appropriate stakeholders[24]. As smart contracts proliferate, mature and standardized semantics could simplify the creation, review and agreement and use of smart contracts.

Annex A
(informative)

Examples of smart contract implementations

A.1 Bitcoin script

Bitcoin[25] uses a scripting system for transactions. The programming language "Script" is a Forth-like simple, stack-based, and processed from left to right. It is purposefully not Turing-complete, allowing for no loops. Officially Bitcoin script is defined by its reference implementation[26].

A.2 BigChainDB crypto-conditions (DLT)

The BigchainDB[27] design starts with a distributed database (DB), and through a set of innovations adds blockchain characteristics: decentralized control, immutability, and creation & movement of digital assets.

One can store the source code of any smart contract in BigchainDB, but BigchainDB won't run arbitrary smart contracts. BigchainDB will run the subset of smart contracts expressible using crypto-conditions. Crypto-conditions are part of the Interledger Protocol[28].

The crypto-conditions specification defines a set of encoding formats and data structures for conditions and fulfilments. A condition uniquely identifies a logical "boolean circuit" constructed from one or more logic gates, evaluated by either validating a cryptographic signature or verifying the preimage of a hash digest.

A fulfilment is a data structure encoding one or more cryptographic signatures and hash digest preimages that define the structure of the circuit and provide inputs to the logic gates allowing for the result of the circuit to be evaluated. It is validated by evaluating that the circuit output is TRUE but also that the provided fulfilment matches the circuit fingerprint, the condition.

Since evaluation of some of the logic gates in the circuit (those that are signatures) also takes a message as input, the evaluation of the entire fulfilment takes an optional input message which is passed to each logic gate as required. As such, the algorithm to validate a fulfilment against a condition and a message matches that of other signature schemes and a crypto-condition can serve as a sophisticated and flexible replacement for a simple signature where the condition is used as the public key and the fulfilment as the signature.

A.3 Corda smart contract primitives/interaction patterns (DLT)

Corda is a distributed ledger platform designed to record, manage and automate legal agreements between business partners. It is designed by (and for) the world's largest financial institutions, yet has applications in multiple industries. It offers an approach to meet the privacy and scalability challenges facing decentralised applications[29].

Corda smart contract summary[30]:

— a valid transaction needs to be accepted by the smart contract of each of its input and output states;

— the smart contract is written in a JVM programming language (for example Java or Kotlin);

— smart contract execution is deterministic and its acceptance of a transaction is based on the transaction's contents alone.

Smart contract limitations: Since a smart contract has no access to information from the outside world, it can only check the transaction for internal validity. It cannot check, for example, that the transaction is in accordance with what was originally agreed with the counterparties. Sometimes, transaction validity will depend on some external piece of information, such as an exchange rate. In these cases, DLT oracles as the interface to real-world sensors or third-party services are required.

Legal prose: Each smart contract also refers to a legal prose document that states the rules governing the evolution of the state over time in a way that is compatible with traditional legal systems. This document can be relied upon in the case of legal disputes.

A.4 Ethereum smart contract primitives/interaction patterns

Ethereum[31] is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference. Therefore, they are principally designed to be unstoppable once they have started.

These apps run on a custom-built blockchain, with shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middle man or counterparty risk.

Ethereum uses the programming language "Solidity" running applications on Ethereum Virtual Machines (EVMs). Rather than to give users a set of pre-defined operations (for example bitcoin transactions), Ethereum allows users to create their own operations of any complexity they wish. In this way, it serves as a platform for many different types of decentralized blockchain applications, including but not limited to cryptocurrencies.

Ethereum in the narrow sense refers to a suite of protocols that define a platform for decentralised applications. At the heart of it is the EVM, which can execute code of arbitrary algorithmic complexity. In computer science terms, Ethereum is "Turing complete". Developers can create applications that run on the EVM using friendly programming languages modelled on existing languages like JavaScript and Python.

Like any blockchain, Ethereum also includes a peer-to-peer network protocol. The Ethereum blockchain database is maintained and updated by many nodes connected to the network. Each and every node of the network runs the EVM and executes the same instructions. For this reason, Ethereum is sometimes described evocatively as a "world computer"[32].

A.5 Hyperledger Fabric

In Hyperledger Fabric[33], an open source DLT, smart contracts are written in chaincode and are invoked by an application external to the blockchain when that application needs to interact with the ledger. In most cases chaincode only interacts with the database component of the ledger, the world state (querying it, for example), and not the transaction log.

Chaincode can be implemented in several programming languages. The currently supported chaincode language is Go with support for Java and other languages coming in future releases.

A.6 Codius (Blockchain)

A.6.1 General

Originally developed by Ripple, Codius[34] is now an open source hosting protocol. It also has built-in billing. That means once a program is uploaded, anyone can pay to keep it running — the author, the users and even the program itself.

Codius objectives include making it easy to develop applications that are

— secure and distributed,

— modular, forkable, and constantly improving,

— integrated with web technologies,

— independent of their developers.

A.6.2 Smart contracts platform[35]

Smart contracts are programs that encode the logic, conditions, and outcomes of agreements and automatically carry out the specified terms. Translating agreements to code is relatively straightforward, but the code then needs to be executed in a way that all involved parties can trust.

They can

— hold assets in one or multiple math-based distributed ledgers, such as Bitcoin and Ripple,

— collect information from any source connected to the internet,

— be written in standard programming languages.

Depending on the security model required for a particular smart contract, the participating parties can select the host or group of hosts they trust. A larger number of hosts can be used to add more redundancy, and thus security, to the smart contract's execution and to ensure that the terms are carried out exactly as specified.

Annex B
(informative)

Role of domain specific languages and methods

B.1 Accord Project

The Accord Project[36] develops an open source technology implementation for smart legal contracts. The Project currently consists of 40+ of the world's leading law firms, along with industry bodies, and corporations. The Project open sourced a software stack built by Clausel[37] that has been developed specifically for legal application and smart legal contracts as opposed to "blockchain smart contracts".

Contracts may run off-chain in a deterministic and verifiable manner, but interact with a blockchain or distributed ledger network when required. This enables a smart legal contract to have a variety of attributes that are not present in some "blockchain-based smart contract" systems but are of interest for commercial and legal usage, such as scalability, privacy, security, etc.

The software stack may be embedded in a node of a distributed ledger network, be run as a centralized service, or in any other suitable form factor. This enables significant flexibility in the manner in which smart contracts are executed. The software stack currently consists of:

1. a templating system — 'Cicero';
2. a domain specific programming language for smart legal contracts — 'Ergo'; and
3. a runtime environment for executing smart contracts.

Details of the interaction between these components may be found here: https://ergo.readthedocs.io/en/latest/Overview.html.

The templating system enables a user to bind natural language prose to executable logic that represents the terms and conditions of a given clause. Clauses may interact with one another and are reusable across smart contracts.

Smart legal contracts may be executed off-chain and interact with a distributed ledger to perform a variety of operations, such as sharing state, performing on-chain transactions, using data instantiated on a distributed ledger as an input to a smart contract, and many others. The Accord Project technology may be used with any distributed ledger system, including Ethereum, Corda, and Hyperledger Fabric. Compiler backends enable Ergo code to be compiled to a variety of execution targets including (but not limited to): JavaScript, Solidity, Kotlin, and a range of other languages used to express blockchain-based smart contracts. By doing so, users are able to build smart legal contracts at a higher level of abstraction and through a common means of creation and execution rather than users working with each underlying distributed ledger individually.

B.2 CommonAccord

CommonAccord[38] is an open source approach to legal documents and legal codification, specifically designed to manage the prose and key parameters within a smart contract. Model legal documents are published on GitHub like software, in "Prose Object"[39] format. That allows legal codification to be handled with the methods and tools used for software codification, using modularity, versioning, pulls, forks and merges. The critical innovation is a recursive model of document construction from component sections, clauses, phrases and the like. This permits any set of documents, and all the steps in transactions to be expressed as connected objects in a "graph". That systematizes the handling of transactions and relationships, making them both computable and human-readable.

Prose objects are a complement to smart contracts, blockchains and other algorithmic approaches to transacting[40]. They provide a systematic way to handle legal text, and therefore to interface to legal systems.

For reference CommonAccord is mentioned in an article from Ian Grigg (author of the original paper on Ricardian Contracts[41]) as the "the most advanced thinking on a clause editor I have seen"; see the section "Declare your variables, for the contract win!" about contract templating and End notes in https://steemit.com/eos/@iang/towards-a-ricardian-constitution.

Annex C
(informative)

Applications and smart contract use cases

Table C.1 — List of use-cases for smart contracts with a focus on legal applications in blockchains

Number	Short name	Description/Example	Optional: special legal topics	comments
1	Supply chain	A good is transported from a to b. Sales contract is individually agreed. A smart contract can monitor the conditions of good. After arrival in time and with well treatment, payment will automatically be initiated.		
2	License agreement	Music is bought (?) via internet (Spotify). The association of owners, licenses, ... is managed on the blockchain.	What was really bought? Music, the license to play, rented?	
3	machine-machine-interaction	An empty fridge orders new food online.	Who really buys? The fridge or the owner?	
4	machine-machine-interaction	There is only 1 license for a robot to produce goods. But there is an additional spare robot for replacement. The first robot "simulates" a defect and the license for production is automatically transferred via the chain to the 2nd robot. Now the 1st robot was quickly disconnected from the network; and it is not really defect. Therefore the transferred license was deactivated for the 1st robot on the chain, but never physically deactivated. Now 2 robots are producing with 1 license.		
5	Monthly payment	Somebody has agreed on monthly payments to his children for financing their studies. It is transferred on Ethereum by a smart contract. Now this person dies and his login data are lost.	How can the payment be stopped?	
6	Energy Trade	The grid-energy balance could be automatically regulated by smart contracts by buying missing energy automatically from power plants, or a surplus can be stored. This can directly be combined with trading and bidding for getting most energy for the lowest price.	Legal? Highly regulated area!	
7	Grid stability	Energy can automatically be routed via the optimal way through the grid for the most efficiency grid use and least energy loss. This data should be public as every grid node could have an own (financial) interest in getting most of the energy through its own part of the grid.		UK startup Electron wants to use smart contracts on the Ethereum blockchain to develop a smart grid that will always deliver energy. Currently, they are in the testing phase and they use 53 million metering points and data of 60 energy suppliers to run experiments.
8	Debt	A smart contract coordinates and records some debt ("A" lends money to "B", including amount, interest rate, repayment terms, etc). The debt may be represented and traded as a token. As repayments are made, the smart contract distributes them to token owners.	If B does not repay, what mechanisms can be put in place to recover the funds?	
9	Sharing Economy	A smart contract can be used to govern sharing economy transactions, for instance by unlocking the door to an autonomous car once a payment has been made.		
10	Property sale and purchase/ Real estate management	A smart contract can be used for signing an agreement digitally between a buyer and seller with conditions of the purchase as part of contract. It could be broadcast between peer nodes through a blockchain		

Table C.1 *(continued)*

Num-ber	Short name	Description/Example	Optional: special legal topics	comments
11	DAO	A decentralized autonomous organisation makes investments in projects selected based on votes by token holders. Token holders also receive some share of the return on these investments.	The DAO (and token holders) could end up funding and profiting from illegal activity. (May overlap with 5?)	
12	**Supply chain transaction tracking**	A smart contract can make more transparent the activities within a supply chain by tracking each step and each transaction between multiple contracted and sub-contracted parties via IoT and the submission of electronic information by parties to the smart contract and/or agreed upon "DLT oracles". This can ensure inventory tracking and timely delivery as well as enhanced tracing and verification in order to reduce the risk of fraud and theft and allow for timely insurance payments when these do occur.	Who would have liability in the case of malfunctioning of the GPS or IoT links?	http://www-03.ibm.com/press/us/en/pressrelease/51712.wss https://blogs.iadb.org/integration-trade/en/how-blockchain-will-track-your-sneakers-around-the-world-and-get-them-to-you-faster/ https://www.provenance.org/news/technology/blockchain-enabled-supply-chain-transparency-can-help-retailers-suppliers-reduce-risk/
13	**Assuring food quality and safety throughout the supply chain** **Also could be used for other products where shipping conditions (temperature, humidity, vibration, etc.) affect quality**	Manage commercial relationships among different actors in the food supply chain by using tokens to represent "food assets" and give dedicated ids to each product/package shipped. The different players in the supply chain will use the tokens to trace possession (and, separately, ownership) of the goods as well as services rendered. This will create a record of stewardship and shipping conditions in order to ensure compliance to quality and safety standards. It will also allow for payments based on services rendered and will identify (and make accountable) those providing services across the supply chain. Companies will be able to create ID tags for food batches or individual food units that will allow access to product information through the blockchain. Retailers and consumers could use these to review the history of a product and check what farms the ingredients came from. The information can also indicate whether the farm follows sustainable agricultural practices and, eventually, its impact on the environment. The token/id for products could also allow consumers to reward growers for the quality of their products and for following sustainable agricultural practices, either through ratings or increased customer loyalty.		The smart contract will receive information at each point in the supply chain starting with the product's origin and any related certificates and then will record whenever ownership changes, to have a complete record of stewardship and, depending upon the food article, may also use information from IoT devices to monitor the temperature, humidity and other key environmental factors that could impact product quality. The smart contract will check IF recorded token contains data that match with established food quality and safety criteria THEN it will allow the product to be sold. Also IF product comes from a certified farm the smart contract can acknowledge that sustainable practices and/or strong ethics were followed. IBM along with Multinational Suppliers and Wal-Mart have developed a Blockchain to track all participants – growers, suppliers, processors, distributors, retailers, regulators and consumers who can get permissioned access to known and trusted information regarding the origin and state of food. https://www.ethnews.com/ibm-to-develop-blockchain-based-supply-chain-solutions-with-partners-including-dole-nestl-and-kroger **Other examples:** http://www.agriledger.com/ https://www.blockchaintechnews.com/articles/from-farm-to-table-how-blockchain-benefits-food-supply-chains/?utm_source=Email_marketing&utm_campaign=reviewBTN05132017141005&cmp=1&utm_medium=HTMLEmail https://www.coindesk.com/walmart-blockchain-product-supply-chains/ http://www.the-blockchain.com/2017/03/30/alibaba-tackle-counterfeit-food-china-blockchain/

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
14	**Provenance (traceability)**	Chain of custody process: A smart contract could indicate that IF product certification is provided by a 3rd-party authority and stored in the blockchain, THEN the purchaser will accept and pay for the goods. One application being explored by PEFC (Programme for the Endorsement of Forest Certification) is the use of blockchain to ensure that wood comes from forests that are certified as being sustainably managed. Tracking provenance of wood from the forest to the wood in your pencil, paper or desk is extremely complex due to the large number of parties and material transformations involved.		3rd party authorities ("DLT oracles") need to be identified and their agreement to participate obtained. In addition, all participants in the chain of custody have to participate. Standardized identification processes, data, and agreements would support wider implementation because of the involved many-to-many relationships. To illustrate: the appropriate 3rd party authority may change by country and product so supply-chain participants need to deal with multiple authorities; at the same time, each of these authorities will need to interact with multiple supply-chain participants, blockchains and smart contracts. https://ensia.com/features/blockchain-environment-sustainability/ https://www.nature.com/news/the-environment-needs-cryptogovernance-1.22023
15	**Increased security in the sale or trade of controlled substances**	Smart contracts could check those wishing to purchase or sell controlled substances and only allow completion of the transaction when requirements are met (such as having an export or import license for environmentally dangerous products or having a prescription from a licensed medical professional for drugs).		https://oncprojecttracking.healthit.gov/wiki/download/attachments/14582699/43-Blockchain%20based%20drug%20monitoring%20and%20dispensation.pdf?version=1&modificationDate=1474478476000&api=v2
16	**Anti-counterfeit**	Each item is given a unique ID, which is paired with either an NFC (Near-Field-Communication) chip or a QR (Quick Response) code. To verify an item's authenticity, users can use a dedicated app, which shows the product's profile. The platform will be responsible for managing the blockchain, which is built off Ethereum, as well as a library of smart contracts, like registering a product's ID or changing its ownership. Each object's ID comes with a public and private key pair.		IF the key paired to the QR code that is attached to an object matches the public key in the server, THEN item is original and attributed to owner of private key. http://bitcoinagile.com/A43299/blockchain-meets-fashion-in-bid-to-fight-the-fakes-nikhilam_stream http://copyrightalliance.org/ca_post/bitcoin-next-big-thing-fight-piracy/
17	**Crowdsourced "Ticketmaster" – fraud prevention**	Is that Justin Bieber, Taylor Swift or Super Bowl ticket authentic, or is someone trying to cheat me and selling me a fake? Smart contracts can be used to oversee ticket exchanges and usage by only transferring proven tickets between parties.		
18	**Trade in Music and Film**	1. Smart contract that continually directs the flow of funding to, and revenues from, projects per the terms of agreement, including royalty payments. 2. Smart contracts to divide profits fairly and without delays according to each person's contribution to the creative process. In addition to actors, screenwriters, and directors, this also benefits other artists and engineers.		https://hbr.org/2017/03/blockchain-could-help-artists-profit-more-from-their-creative-works
19	**Transport contract management**	A smart contract can coordinate the implementation of transport contracts between 2 or more parties (Shipper A and Carrier B). This can include initiating the payment of the transport invoice to the carrier if there are no specific events declared. For example, if the carrier (B) delivers goods to the consignee (C), payment of the transport by the consignor (A) is initiated.	For payment, blockchain provides the opportunity to make invoices functionally unnecessary. At the same time, many legal (and fiscal) systems require invoices, so what needs to be done to allow blockchain information to be used to meet the legal requirements currently met by invoices?	The smart contract contains the conditions under which B (the carrier) can generate an electronic consignment note on behalf of the shipper A (for example up to a quantity X between dates Y and Z). Then, if A needs a transport to be handled by B, B can generate the electronic consignment note. Particularly useful in just-in-time manufacturing scenarios. http://cwt.top/news/378/maersk-tests-blockchain-while-registration-of-consignment-notes

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
				The following is a link to a blockchain in the trucking industry association which is looking for standards to support blockchain use in the sector where goods often pass from one transport service provider to another. https://bitcoinmagazine.com/articles/blockchain-trucking-alliance-seeks-revolutionize-transport-industry/ http://www.ey.com/gl/en/newsroom/news-releases/news-ey-advancing-future-of-transportation-with-launch-of-blockchain-based-integrated-mobility-platform
20	**Vehicle Ownership Sharing**	EY has a project called Tesseract that will digitally log vehicles and trips onto a blockchain to facilitate shared vehicle ownership. Currently for passenger vehicles at the same time it is also highly relevant for lorries/trucks.		This may seem like Uber the ride sharing system, however this project is for vehicle ownership sharing. http://www.ey.com/gl/en/newsroom/news-releases/news-ey-advancing-future-of-transportation-with-launch-of-blockchain-based-integrated-mobility-platform
21	**Transport notifications**	A smart contract can generate an automatic message to the consignee and the consignor when the carrier is entering a predefined zone (geofencing) before delivery.	Who would have liability in the case of malfunctioning of the GPS or IoT links?	If a contract between a consignor (A), a Carrier (B) and a consignee (C) stipulates that information must be sent to C prior to delivery and as soon as the truck enters a predefined zone (in Km, Miles, or timeframe) then, as soon as the truck enters that zone as recognized by its GPS, a verifiable message is automatically sent to the consignee and the consignor. In the light of the havoc wreaked by recent "hacks" on transport networks, the privacy and "unchangeableness" of blockchain data may have a value in and of itself.
22	**Customs clearance and transit**	Customs may use blockchain smart contracts to register documentary requirements and then control, automatically clear, or order further action on goods passing under its control. For example, In the case of a transport under a Transit (T) procedure, Customs may wish to check the consignment note to verify compliance with the information declared to Customs. A consignor (A) ships some goods under a transit document to a consignee (C) using a carrier (B). To do this, the carrier (B) or the consignor (A) requests and is issued a Customs transit document using the dematerialized custom transit system. Customs (Cu) may then wish to check the consignment note and validate that it matches the information declared for the transit operation/document.		If B issues a Transit document and an electronic consignment note for the handling of a transport between A and C, then Cu is notified that a Transit document has been issued and, if needed, can have direct access to the electronic consignment note. https://www.ibm.com/blogs/research/2016/07/bringing-blockchain-innovation-singapore-asia/ http://brexitcentral.com/blockchain-innovative-solution-brexit-customs/ https://www.facebook.com/SingaporeCustoms/photos/a.301028300650.323372.297996035650/10159018167585651/?type=3&theater
23	**Track origin and status of goods moving through a Port**	Melbourne Port's new owners are considering using blockchain to allow parties in the logistics chain to keep track of the origin and status of goods that move through the port. The system will use a private ledger.		They also intend to use blockchain to enhance maintenance, improve security and generally enhance trade at the facility. http://www.theaustralian.com.au/business/companies/port-of-melbourne-ponders-driverless-trucks-blockchain/news-story/587df9776ebb4119348c379a1775cf7f
24	**Authorization to release goods**	Tokens, representing rights to goods, keys to locations/lockers, or 3rd party digital assets, are transferred between parties. For example, an organization that outsources its warehousing can transfer a token for possession of goods from the warehouse to the buyer, thus providing clearance to the third-party warehouse to release the goods to the buyer.	What recourse will the buyer have if the third party only partially fulfils the transfer of goods?	Smart contracts need to be written to allow changes to quantities and timings. For example, perhaps 100 items are sent by A to a warehouse under a contract with B. Then B discovers it only needs 50 and A discovers a buyer C who will pay more than B – so both parties want to change the quantities and instead of one token for 100 items, 2 tokens for 50 items each are needed. In another scenario, perhaps 20 items are damaged in shipment and A and B agree that only 80 will be delivered, requiring, again, a change. These scenarios are not unusual in the "real" world.

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
25	**Trade Finance – Letters of Credit**	A smart contract can validate the receipt of the information required for an international transaction to be paid by a letter of credit (LC) and ensure that payment is made by the bank at destination according to the terms of the letter of credit and without any delay The consignor (A) sends a shipment overseas to (C) under a letter of credit. A smart contract could also transfer payments upon multi-signature approval for LCs and issue payments to a port for warehousing/storage of goods based upon a received bill of lading.	If A or an authority "delivers" documents required by an LC to the blockchain, how can they be validated? If the documents are delivered to the blockchain by "DLT oracles" then who determines the legal status and the trustworthiness of the DLT oracle? The legal implications in case of an execution failure or problem have to be determined, particularly in the case of disputes or defaults. For example, it is not uncommon to find that a bill of lading does not 100% match the actual goods delivered and then adjustments are needed.	If A delivers the documents needed to comply with the terms of the letter of credit to the bank at destination, then the payment of the amount stipulated on the letter of credit is liberated to the account of A. It is also possible to have many or all of the documents required by the letter of credit submitted directly to the blockchain by the parties generating the document (customs, inspection agencies, warehouses, shipping companies, etc.). In other words, to have them submitted by "DLT oracles". In this last scenario, how can the identification of DLT oracles and the submissions by DLT oracles to the blockchain be standardized so that these do not have to be customized for each individual company to company "pair" within the many to many networks used by supply chains (many shippers using many transporters and vice versa). https://www.gtreview.com/news/asia/banks-blockchain-innovation-letters-of-credit/ https://coinidol.com/sberbank-conducts-letter-of-credit-on-the-blockchain/ https://www.ethnews.com/chinese-bank-uses-blockchain-to-deploy-letters-of-credit
26	**Establishment, sharing and management of KYC (Know your Customer) information)**	Smart contracts could be used to establish Know Your Customer (KYC) information, according to agreed rules and legislative requirements. This KYC information could then be shared across banks and financial institutions (so that each one would not need to establish a separate KYC) and used by other smart contracts. Know Your Customer is an expensive element of on-boarding a new client and each bank must create their own KYC. Smart contracts can make provision for a situation where the customer changes their KYC information, such as address The coding in the smart contract would require the customer to be notified automatically that they need to resubmit their proof of residence for it to be accepted by the participating bank without having to require the bank to do this manually.	Such use would need to conform with privacy legislation across a wide number of jurisdictions A smart contract would need to include provisions allowing it to be amended, renewed or terminated before the contractual goal is complete.	For KYC information to be shared across banks, standards will be needed, including for data. https://bitcoinmagazine.com/articles/deloittes-regtech-offering-blockchain-powered-kyc-service-solution/ https://www.finextra.com/blogposting/13903/kyc-and-blockchain
27	**Anti-Money Laundering (AML) often closely linked to KYC**	Using Artificial Intelligence (AI) to detect suspicious transactions and recording them on a Blockchain network. Smart Contracts can then be used to stop money transfers based on data available.		This can significantly cut the time needed for sharing information between regulators, banks. http://www.antimoneylaundering.lawyer/blockchain-anti-money-laundering/
	Identity control for verification and upload of notarized Documents	Ministry of Planning in Brazil used uPort, an Ethereum-based identity platform that allows people to own their identity and control their personal information, "as an identity application for its proof of concept (PoC)" for the purpose of employing blockchain technology for the verification of important notarized documents		"This enables the Ministry of Planning to provide citizens greater access to its software and services in a more secure, private, and efficient manner, and puts the user in control of their own identity." uPort Project Lead, Rouven Heck https://www.coindesk.com/brazils-ministry-planning-testing-blockchain-identity-tech/

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
28	**Identity based eSignatures for Cross Border Trade smart contracts**	In India, we are evaluating the use of qualified eSignatures based on an existing Digital Id (AADHAAR) and supplied by a Trusted Service Provider to sign transactions on Blockchain. This will allow transactions to be both secure and traceable.	These Qualified signatures (AADHAARs) are legally non-repudiable under India's Information Technology Act and presents a use case for Blockchain and qualified signatures to coexist.	Combining smart contracts with Verified Identity makes for better enforcement as neither transaction nor identity can be denied. Over 1.1 billion Indians now have AADHAAR digital IDs. https://medium.facilelogin.com/making-aadhaar-better-with-blockchain-ec3aef9852b0 This article is about how blockchain could be used to improve AADHAAR for its holders, so it is relevant, although not focused on their use for transaction signatures.
29	**Real Time Gross Settlement**	The Central Bank of Brazil is involved within several Blockchain projects. One project (SALT) has developed a prototype for Transaction Settlement on the Blockchain. The difference with this system and other similar projects, is it is designed to be monitored by the Central Bank.		They are also considering other use cases such as the Agreement on Agreement on Reciprocal Payments and Credits (CCR), Local Currency Payment System (SML), and identity management system. http://www.bcb.gov.br/htms/public/microcredito/Distributed_ledger_technical_research_in_Central_Bank_of_Brazil.pdf
30	**Bank settlement using digital cash equivalents to major fiat currencies**	UBS joined by six other banks to launch a blockchain settlement system. The system uses a "utility settlement coin" (USC) which is a digital cash equivalent of each of the major currencies backed by central banks. The advantage of using the blockchain for settlement would be increased speed and lower administrative costs. The utility settlement coin would be convertible at parity with a bank deposit in the corresponding currency, making it fully backed by cash assets at a central bank. Spending a coin would be the same as spending the real currency it is paired with according to UBS.		"Digital cash is a core component of a future financial market fabric based on blockchain technologies," UBS Investment Bank's head of fintech innovation Hyder Jaffrey said. http://www.irishtimes.com/business/financial-services/ubs-leads-team-of-banks-on-blockchain-settlement-system-1.2766742
31	**Detecting Fraud in financial transactions**	The Bank of England is testing a system using Blockchain and Artificial Intelligence to detect fraud in Financial Transactions. The results showed that the blockchain takes fewer steps for settlement and allowed AI teams to detect fraud at the same time.		The bank set up an entirely experimental simulation using a Blockchain company called Ripple and an AI company called MindBridge to test settlements across two different currencies. The bank is looking to do a longer experiment with MindBridge, to test its effectiveness with data as it comes in over time. http://observer.com/2017/07/bank-of-england-mindbridge-ripple/
32	**Microloans**	Tracking and releasing microloans based on conformance with certain requirements registered on the Blockchain.	What is the recourse of the borrower if they believe they have met the requirements but the "DLT oracle" submitting this information to the blockchain does not agree or fails to communicate this information?	One of the largest obstacles to micro financing is administrative costs and many of these can be dramatically reduced through use of the blockchain. The security "weak point" in blockchain applications is very seldom the blockchain itself, rather it is the systems that provide information to the blockchain (and sometimes poorly designed smart contracts). Therefore, especially in financial applications such as this, security standards for the "DLT oracles" providing information to the blockchain could provide important implementation support. https://www.bitcoinnews.ch/4237/microfinance/ https://www.finextra.com/pressarticle/69432/everex-announces-mobile-blockchain-powered-microfinance-and-fiat-transfer-platform https://blog.everex.io/problems-with-microlending-and-how-blockchain-solves-them-1582f98e2a7c

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
33	**Stock Exchange – setting trading parameters and voting**	Use smart contracts to set client trading margins based on records and verified collateral. Blockchain can also be used as a secure environment for shareholder voting and recording the results (via smart contracts).		Securities exchanges have very high overhead costs and mistakes can create significant financial risks. Blockchain smart contracts can help address some of these issues. http://www.nasdaq.com/article/how-stock-exchanges-are-experimenting-with-blockchain-technology-cm801802 https://www.coindesk.com/india-stock-exchange-blockchain-kyc/ Voting examples http://business.nasdaq.com/marketinsite/2017/Is-Blockchain-the-Answer-to-E-voting-Nasdaq-Believes-So.html https://www.economist.com/sites/default/files/plymouth.pdf
34	**Land title recording/ land registries**	Smart contracts could record and make publicly available information related to property transfers and liens, thus deterring fraud, improving transaction transparency and efficiency, and strengthening confidence in the identity of owners and property.	Related legislation may need to be revised.	Common protocols and standards need to be developed for electronic record filing. https://www.forbes.com/sites/laurashin/2017/02/07/the-first-government-to-secure-land-titles-on-the-bitcoin-blockchain-expands-project/#6f1e29b84dcd https://qz.com/947064/sweden-is-turning-a-blockchain-powered-land-registry-into-a-reality/ https://www.cryptocoinsnews.com/u-k-land-registry-looks-register-property-blockchain/
35	**Release of mortgage/lien**	A smart contract could track debt payments where there is a mortgage or lien and automatically end/release the mortgage or lien when the debt has been paid off in its entirety.		http://www.pwc.com/us/en/financial-services/publications/assets/pwc-financial-services-qa-blockchain-in-mortgage.pdf https://www.bsa.org.uk/BSA/files/72/725f716b-a436-4f74-bcc6-6101fad63c20.pdf https://www.housingwire.com/articles/40274-from-bitcoin-to-blockchain-how-new-ledger-tech-can-morph-the-mortgage-industry?page=2
36	**Escrow / earnest money, deposits (for example, security deposit, Kickstarter)**	A smart contract can ensure that amounts placed in escrow (for potential home sales) or deposits (for rental agreements) are either transferred to the selling/landlord party, are returned to the buying/tenant party in whole or in part as agreed in advance based on conditions that could range from a simple date to a good inspection report to the finalization of a home purchase.		This would provide home purchasers and renters guarantees that their escrow money and rental deposits will be refunded in a timely manner after conditions are met and will not be retained for extended periods or "stolen". http://blockchained.blogspot.fr/2015/05/escrow-services-with-blockchain.html
37	**Transport Insurance**	A smart contract can generate the automatic opening of an insurance file in case of damages to the goods transported between the consignee (C) and the carrier (B) and also between the consignor (A) and the carrier (B).		If B delivers a consignment to C and exceptions to the electronic consignment note are signalled by either B or C, then a file is opened at the insurance company in charge or designated by B or C (same applies for A). http://www.ey.com/Publication/vwLUAssets/EY-blockhain-in-insurance/$FILE/EY-blockhain-in-insurance.pdf
38	**Truck/Car Insurance**	A smart contract can record insurance policies, driving records and driver reports, allowing Internet-of-Things-equipped vehicles to submit expedited claims shortly after an accident.		Cross-industry collaboration is needed to develop standards and address, technological, regulatory and financial challenges. http://www.ey.com/Publication/vwLUAssets/EY-blockhain-in-insurance/$FILE/EY-blockhain-in-insurance.pdf Page 3, column 2 https://github.com/Capgemini-AIE/blockchain-insurance (for programmers; very technical)
39	**Life and Health Insurance**	A partnership of Berkshire Hathaway and iXledger provides a re-insurance system that could provide more transparency and trust.		http://www.lifeinsuranceinternational.com/news/company-news/ixledger-partners-gen-re-develop-insurance-solutions-using-blockchain/

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
40	**Flight Insurance and Transport (Airline) Claims Processing**	Automate the process of insurance claims processing for cancelled or delayed flights using a smart contract based on flight data information that is fed to a blockchain smart contract which automatically compensates passengers according to legal and contractual requirements.		A company in the UK, InsurETH is working on a system to allow passengers to take out flight insurance directly through an Ethereum smart contract. http://insureth.mkvd.net/ This could eventually also be used by airlines, thus eliminating administrative costs and improving customer relations while meeting legal obligations. In addition, the experience would be replicable for cargo and, eventually, railways where there are also records available showing actual departure and arrival times.
41	**Crop Insurance**	Automatically process crop insurance claims using a smart contract based on set of parameters where weather data can be used to determine payments.		Smart contracts for crop insurance that integrate digital payment platforms can bring the cost of insurance down and make it available to smaller and poorer farmers — those who need it most. https://dailyfintech.com/tag/crop-insurance/ https://www.ibm.com/blogs/insights-on-business/insurance/insurance-crystal-ball/
42	**Access to rentals including hotel/home entry (AirBnB)**	Using smart contracts and IoT, locks could be installed that only allow access to rented facilities, vehicles or equipment to those who have the correct numeric keys (distributed by the blockchain) or whose identity matches that registered on the blockchain. These could be subject to both payments and other conditions and time-limited. The smart contract could even require users to match registered biometric data (eye scans, fingerprints, etc) before issuing the instruction for an IoT device to open a lock.		For example, such blockchain-based systems could simplify hotel room and AirBnB rentals, eliminating the need for checkin and the collection of keys for facilities. It could also allow car and truck rentals to be automated and increase security (for example for the rental of trucks that could be used for terrorist attacks) by requiring biometric validation of the renter's identity. https://skift.com/2017/07/11/blockchain-will-disrupt-expedia-and-airbnb-tui-ceo-says/
43	**Building and facilities security and access**	**Security management:** Security agencies commonly use a daybook for the chronological recording of all events notified to the agency, that have implications for the security of a certain environment. Smart contracts could record security information, with time stamps, from IoT devices, from geographically distributed security agents and other information such as the presence of visitors and staff in restricted areas, and then alert security staff, police or fire services when circumstances call for active intervention. This could increase the granularity of what is reported as well as the responsiveness to emergency situations. **Access:** Using smart contracts and IoT, locks could be installed on buildings or facilities that smart contracts will only open for individuals meeting conditions that could include: an employment contract, a technical certificate, a work assignment, security clearance, matching registered biometric data (for example eye scans, fingerprints), etc.		Blockchain could completely replace the current need for security agencies to maintain a document-based daybook. https://www.chainofthings.com/news/2016/8/8/blockchains-ideal-for-iot-in-facilities-management For example, blockchain-access systems could allow repair or cleaning people to enter facilities, workers to enter high-security buildings or restricted areas, etc., all without human intervention.
44	**Zipcar/Bike rental/boat rental/WIFI/phone**	Physical items as well as services can be used, tracked and billed by smart contacts based on transactions and IoT information entered into the Blockchain.		For example, for a car-rental smart contract, the dates or times of the rental could be entered and, perhaps, biometric data for the renter. The car could then be picked up without human intervention, based on biometrics. IoT devices could then communicate to the blockchain the miles travelled, gasoline used, location of the car, etc. and then the car could automatically lock itself when the rental contract ends. A technician could then send a "state of the car" report to the blockchain and based on all this information the renter would be automatically charged.

Table C.1 *(continued)*

Number	Short name	Description/Example	Optional: special legal topics	comments
				http://www.telegraph.co.uk/technology/news/11961296/Visa-uses-bitcoins-blockchain-technology-to-cut-paperwork-out-of-car-rental.html https://www.rtinsights.com/blockchain-pilot-smart-contracts-docusign-visa/ http://www.trustnodes.com/2017/05/28/toyota-prototypes-ethereum-blockchain-based-car-sharing-uber-alternative
45	**Community management of local energy micro grids and tracking solar/clean power contributions to a shared energy network**	A smart contract can allocate power use on a local/community shared energy grid according to pre-determined rules and track power generated, power used and sold, and the cost of any power purchased from an external grid in order to determine balances and make charges or payments to participants as appropriate. A smart contract can also automatically reimburse those who contribute clean energy to an energy network based on IoT data sent to the blockchain from meters.		Because using smart contracts for these purposes is automatic, reliable and inexpensive, it should make local grids more cost effective and also make it more cost effective for large energy networks and distributors to manage the purchase of excess clean energy from hundreds of thousands of small energy generators (such as home owners with solar roof installations or small wind mills). Once the self-consumption energy community has been created, the community manager is in charge of the electricity billing of each community participant. The administrative costs can affect the profitability of this venture, therefore smart contracts can be used to automate the administrative procedures in a transparent manner. https://www.technologyreview.com/s/604227/blockchain-is-helping-to-build-a-new-kind-of-energy-grid/ http://energy.sia-partners.com/20170220/microgrids-can-be-optimised-blockchain https://www.siemens.com/innovation/en/home/pictures-of-the-future/energy-and-efficiency/smart-grids-and-energy-storage-microgrid-in-brooklyn.html
46	**Smart contracts to enable machine to machine (m2m) energy economy**	This is a similar case to peer-to-peer but instead of performing a transaction among households, this would allow a single smart device to purchase its own electricity using a smart contract and unique identifier from another smart device (smart battery). This transaction may occur without the intervention of any human, however the history of such transaction can be registered in the blockchain and later accessed by human auditors.		http://www.3e.eu/blockchain-technology-energy-sector/ https://como.cheng.cam.ac.uk/preprints/c4e-Preprint-178.pdf
47	**Use blockchain smart tokens management of "carbon credits" on energy trading carbon-credit markets**	Enforcement can be written into smart contracts to follow the suggestions from the Paris Climate Agreement. Additionally the use of smart tokens (as a digital asset) can provide attestation of the guarantee of origin for carbon credits on the trading market, as well as track their ownership and avoid the double spending of carbon credits.		https://btcmanager.com/ibm-launches-carbon-credit-management-using-hyperledger-fabric-blockchain/ http://www.prnewswire.com/news-releases/energy-blockchain-labs-and-ibm-create-carbon-credit-management-platform-using-hyperledger-fabric-on-the-ibm-cloud-300425910.html http://www.sustaineurope.com/ipci-carbon-compliance-units-to-be-carved-in-blockchain-tablets.html