FintechExplainer

# “Smart Contracts”: A Smart Way to Automate Performance

Jenny Cieplak & Simon Leefatt

**April 2017**

**Cite as:** 1 GEO. L. TECH. REV. 414 (2017)

**Perma:** https://perma.cc/EUT6-RL6P

(Download PDF)

The freedom to contract is one of the oldest and most basic tenets of the American legal system. Subject to limited judicial and statutory exceptions, parties have been and are generally afforded carte blanche in determining the terms of a binding agreement and how those terms are memorialized. The recent emergence of “smart contracts,” that are stored and executed using distributed ledger technology, is another step forward in the process of computerized contracts, following electronic delivery of signatures through PDF and fax to today’s digital signature services. What makes smart contracts unique, however, is that they not only involve the automation of contract formation, but also the execution of the contract’s terms.

## What is a Smart Contract?

There exists no universally accepted definition of a smart contract. Generally, smart contracts are computer protocols that implement the terms of a negotiated contract in a self-executing manner. These contracts may either be written entirely in standalone code, coupled with traditional written agreements reflecting the same negotiated terms codified in the code, or partially governed by both code and a traditional written agreement that is incorporated by reference in the code itself. Smart contracts have broad applicability and, as a result, they may be used to govern or facilitate many types of financial transactions.

Nick Szabo, who is considered by many to have been the originator of the smart contracts concept, described the concept of incorporating contract terms into computer hardware and software by describing a car lien.[1] Without smart contracts, if the owner fails to make payments on the loan secured by the car, the lender must go through the process of repossessing the car. By using a self-executing smart contract to enable a hardware and software function in the car, a lender can make it impossible for the owner to start the car if the owner fails to make payments. Once the loan has been completely paid off, the smart contract can automatically add a new function that disables the previous function.[2]

Another example, which does not implicate problematic considerations of wealth inequities, is derivative contracts. Consider an interest rate swap, where Party A agrees to pay to Party B each month an amount equal to 5% of notional amount X, and party B agrees to pay to Party A each month an amount equal to some floating rate of interest of notional amount X. In real life, Party A and Party B determine whose payment is larger, and exchange a net amount. Basically, Party A is betting that the floating rate

**WHAT OTHERS ARE READING**

The Law and Legality of Smart Contracts
Smart After All: Blockchain, Smart Contracts, Parametric Insurance, and Smart Energy Grids
“Smart Contracts”: A Smart Way to Automate Performance
Blockchain Technology Impact on Energy Market Transformation: Secured Distributed Energy Transactions in the Cloud
Re-Identification of “Anonymized” Data
Law of the Foal: Careful Steps Towards Digital Competence in Proposed Rules 902(13) and 902(14)
Machine Learning & Fintech

**BROWSE TOPICS**

(Select Topic)

Twitter

of interest will, on average, be more than 5%, so that he always receives the monthly payment, and Party B is betting the opposite.

Interest rate swaps are currently documented through transaction confirmations, which incorporate by reference master agreements, schedules, and credit support annexes.[3] The master agreement, schedule and credit support annex, along with other optional documents, are general documents that govern the trading relationship of the parties, and apply to all swap transactions. These documents are typically executed manually, with signatures often delivered by fax or PDF, or by using DocuSign or another electronic signature service. A transaction confirmation is created for each swap which includes the terms of the particular swap, and may be executed “manually, electronically, or by some other legally equivalent means.”[4] The terms and conditions of a particular swap thus appear on multiple transaction documents that are separately viewable in static form, i.e. via local copies either in print form or saved on a hard drive document management system. The parties must then access rate providers to determine periodic payments and send these payments from their accounts.[5] Each of these processes may be automated to some degree,[6] but they are also open to error, both human and computer-based. Data may be entered mistakenly, and flaws in code can also cause errors in information to appear.

In addition to simply documenting the business terms of the swap, parties to swap transactions must undertake a large number of legal and compliance steps. These steps include checking counterparty eligibility, documenting the trade, determining whether the trade must be submitted to a clearinghouse, and regulatory reporting, as well as actually making payments. The process is extremely complex and typically involves multiple systems across multiple parties.[7] These systems may or may not be connected, and data may not properly transfer. Human errors such as typing mistakes (known as “fat finger” errors) are common as well. A misplaced decimal point in one party’s system could cause mistaken payments and serious disputes. Parties can even have disputes about whether a transaction exists or not.

Instead, a smart contract could be used to encode the terms of the swap, import information from a rates provider, and automate payments from the parties’ accounts. Because each of these processes is based on a smart contract in a shared ledger rather than on multiple systems that may or may not interact properly, there are fewer opportunities for the parties to have conflicting information. The smart contract on the ledger can incorporate the terms of the master agreement, schedule, credit support annex and other relevant documents just as swap transaction confirmations do today. Some solutions even offer the possibility of including an encoded copy of a pdf of a paper contract directly on the ledger.[8]

Of course, for either of the above use cases to function properly, there needs to be a system wherein the parties to the contract are connected. In the car lien example, the computerized contract that is stored in the car’s onboard computer needs to have a way of confirming that payments on the loan have been properly paid. In the interest rate swap example, the computerized contract which is stored on a party’s recordkeeping system needs to have several different types of connectivity – it must communicate with each party’s bank account to enable payments from one party to another, and it must receive information from an interest rate provider to determine the amount of the required payment. Many industries are looking to distributed ledger technology (DLT) to make this communication possible using only one system, rather than multiple different connection systems.

**What is a Distributed Ledger Technology?**

A distributed ledger is essentially a database for tracking assets and information that can be shared among multiple participants. For example, imagine a ledger with a record of all the transactions in shares of a company’s stock, beginning with the initial issuance of the stock to the initial purchasers, and including all subsequent transfers.[9]

The interesting thing about distributed ledger technology is that the ledger is replicated across multiple participants in a network.[10] The ledger can be replicated in its entirety among all network participants, so that each participant can see all changes to the ledger, or segments can be replicated so that participants only see portions of the ledger that are relevant to them. [11]

In each case, the ledger is not just copied from one network participant to another – each copy is considered the "original" copy. [12] Network rules provide that when an asset changes hands or a transaction is created or modified, that resulting change in the ledger is broadcast to all copies of the ledger or, in a ledger system where not all participants have access to the full ledger, the transaction is broadcast only to the relevant parties. [13]

Network participants access their assets on the ledger through cryptographic keys. [14] Only the party or parties with the correct key or combination of keys can transfer or otherwise modify an asset or transaction. [15]

**Smart Contracts on a Distributed Ledger – Automating Performance**

Distributed ledgers can be used to record information such as the interest rate swap contract described above. The portion of the contract that automates performance should be deterministic (i.e., it should provide for all possible outcomes based on relevant facts). However, to automate performance of the contract, the distributed ledger must also have access to the means of performance and any metric by which performance must be measured. [16] In the interest rate swap example, the distributed ledger must have access to some asset of the parties' in order to fulfill the parties' payment obligations, and it must have access to a provider of interest rate information. [17]

Some distributed ledgers, such as the blockchain for the cryptocurrency Ether, provide for the automated performance of smart contracts by utilizing a token that is native to the distributed ledger itself. [18] Users create smart contracts by uploading them to the blockchain and the contract is then propagated through the system as described above. On the Ethereum blockchain, a smart contract consists of program code, a storage file, and an account balance. The smart contract can receive money into its account balance and send money from its account balance. In order to invoke the smart contract process, the parties to the contract "contribute" a certain amount of Ether to the contract. This contributed Ether becomes subject to the smart contract and is used to fulfill the parties' payment obligations. The program code runs automatically once the parties contribute their Ether, and pays Ether to the party that is supposed to receive it in accordance with the terms of the contract.

However, contributing all of the currency necessary to make all payments under a smart contract is likely impracticable in many situations. Banks that are party to interest rate swaps do not want currency representing the entire amount potentially payable over the course of the swap to be locked in an account. Solutions such as R3's Corda solve for this issue by creating "state objects," and in particular "cash states." A cash state represents an amount of currency that one ledger participant, typically a bank, owes to another ledger participant. A cash state is like a bank account maintained outside the distributed ledger context, in that it does not represent physical fiat currency held by the bank but instead represents an amount owed by the bank to the account holder. The smart contract can access this "cash state" as if it were a bank account, and require the bank to transfer a portion of the "cash state" to the payee. [19]

In addition to having access to the means of performance, on occasion smart contracts may need access to outside information to determine what is required to perform the contract. If smart contracts, like other computer code, can be described as a series of "if-then" statements, to activate the process, one must know whether the condition has occurred. [20] For example, an interest rate swap transaction would consist of the following "if-then" statements:

• If fixed rate exceeds floating rate on first day of any month N, fixed rate payor pays to floating rate payor an amount equal to [fixed rate – floating rate] * notional amount on date that is 15 days after the end of month N
• If floating rate exceeds fixed rate on first day of any month N, floating rate payor pays to fixed rate payor an amount equal to [floating rate – fixed rate] * notional amount on date that is 15 days after the end of month N

Here, you would need someone to determine what the floating rate of interest is on the first day of each month. The smart contract can then calculate whether the floating rate is higher or lower than the fixed rate, which will be encoded in the smart contract. The concept of "oracles" is useful here. An oracle is a third-party information services provider that will digitally "sign" a transaction, attesting to the occurrence of specific conditions. [21]

Turning again to the interest rate swap example, an oracle could be used to provide interest rate information on a payment calculation date. The oracle's digital signature would be retained on the distributed ledger so that parties could review the payment process and confirm that payments were made correctly.

Note that parties to a smart contract will need the oracle to be a trusted party so that there are no insinuations that the oracle has colluded with one of the contract parties, or has reported incorrectly. In the interest rate swap example, neither Party A nor Party B can rely on the other to report interest rate information correctly, because both parties have an economic incentive to make their payment smaller than the other party's payment. Party A, the payer of a fixed rate of interest, has an incentive to make the floating right higher so that Party B has to pay more than Party A. Party B has an incentive to make the floating rate lower. While there would be a penalty if either party lied, as performance is automated under the smart contract, the lie would cause a payment to be made in error, and the parties would need to correct the mistake. A more efficient solution would be a trusted data provider to serve that function, which will be neutral to both parties.

Parties will also need to ensure that an oracle does not "go dark" and stop providing information, either due to technical errors or because the oracle simply decides to stop providing services. The oracle should agree to minimum standards of availability and a minimum subscription period. Alternatively, multiple oracles can be used for the same smart contract, using a "majority rules" method to determine when a condition has occurred.

**Concerns with the Smart Contract Model**

One notable recent example of smart contracts is the Decentralized Autonomous Organization ("DAO"), a pseudonymous, crowd-sourced investment vehicle using the digital currency Ether. To participate in the DAO smart contract, investors transferred their Ether to a common pool, similar to paying cash to invest in a mutual fund. The DAO smart contract code was designed to enable these investors to vote on how the Ether pool would be invested. The smart contract also contained a function that an investor could invoke to enable him or her to exit from the DAO. This function, when executed, told the DAO where to distribute their Ether. [22]

However, a flaw in the DAO smart contract code enabled a user to continually exercise the removal request – even though he had already taken out more Ether than he had put in. The flaw existed because the removal function could be exercised recursively – that is, the recall function could be exercised continually without checking whether the user had already withdrawn the total amount he contributed to the DAO. [23] Because the Ethereum blockchain [24] is designed to prevent rollback of transactions, and because there is no central authority to force the user to undo the transaction, there was no mechanism in the code to put the stolen Ether back into the right hands. [25] Further, remedies outside the Ethereum blockchain, such as litigation, were not viable because due to the pseudonymous nature of the Ethereum blockchain¬, which made it impossible to determine the identity of the malfeasant user. [26]

Users of smart contracts should be aware of the risks of using untested code in a pseudonymous or anonymous context without remedies for hacking or flaws in code. On networks such as the Ethereum network, anyone can become a network participant simply by downloading and running the code, which is open source and available to everyone. No identification or authorization is necessary. In contrast, many of the distributed ledger platforms being built now are meant for use on a permissioned-only basis. In a permissioned-only ledger, one or more network operators act as gatekeepers on the network and only allow participants to access the network once they have been identified and met any applicable access criteria.[27] If the identity of all participants is known, a malfeasant participant can be subject to legal remedies.

## Enforcement outside the Distributed Ledger Context

Provisions such as payment requirements can easily be automated, and with oracles automatic termination can be instituted upon the occurrence of specified events. However, even for relatively standardized contracts such as interest rate swaps, enforcement of provisions such as confidentiality requirements is likely to require court intervention. And in cases such as the DAO where flaws in code allow a participant to take actions that are not permitted by the terms of the agreement among the parties, court invention may also be needed.

In such a situation, courts should be able to look to enforcement of digitally-signed contracts as a roadmap. For example, the Uniform Electronic Transactions Act provides for a broad variety of electronic methods of assenting to a contract, including “an electronic sound, symbol, or process attached to or logically associated with a record and executed or adopted by a person with the intent to sign the record.”[28] Digital signatures using public/private key cryptography should fall comfortably into this definition.

Of course, to take advantage of remedies only available in court, the counterparty to the contract must be identifiable. The recent DAO hack illustrates this point best. Unknown hackers exploited a weakness in the code of the DAO contract and withdrew Ether from investors who were parties to the DAO contract. The reason why other participants in the DAO had no recourse against the hackers was not due to some perceived difference between smart contracts and traditional contracts, but because the parties against whom the contract would be enforced were unknown.[29] This is a key argument in favor of permissioned ledgers, where parties’ identities are known and validated.

## Conclusion

Smart contracts can be viewed as merely another means to evidence legally binding relationships – however, their emergence has and will continue to change the way parties transact. As the use of smart contracts becomes more widespread, the efficiency gains they promise will become reality. However, market participants will need to be aware of potential security flaws and ensure that they can trust not only the counterparty to the contract, but also the code itself.

Jenny Cieplak & Simon Leefatt

* Jenny Cieplak is a counsel in the Corporate Group at Crowell & Moring LLP and is the head of the firm’s blockchain and distributed ledger technologies initiative. © 2017, Jenny Cieplak and Simon Leefatt. * Simon Leefatt is an associate in the Corporate Group at Crowell & Moring LLP and is a member of the firm’s blockchain and distributed ledger technologies initiative. © 2017, Jenny Cieplak and Simon Leefatt.

Share this:

Related

The Law and Legality of Smart Contracts
April 2017
In "Articles"

Insurtech: An Industry Ripe for Disruption
April 2017
In "Legal News"

A New Journal at the Intersection of Law and Technology
November 2016
Similar post

FintechExplainer
Technology Explainer
Vol. I, Issue II (2017)
ZZZ--Technology Explainer Featured
Tags: Bitcoin, Blockchain, Fintech, Smart Contracts