An Introduction to Smart Contracts and Their Potential and Inherent Limitations

Skadden

Skadden, Arps, Slate, Meagher & Flom LLP and Affiliates

The Americas
Boston
Chicago
Houston
Los Angeles
New York
Palo Alto
São Paulo
Toronto
Washington, D.C.
Wilmington

Europe
Brussels
Frankfurt
London
Moscow
Munich
Paris

Asia Pacific
Beijing
Hong Kong
Seoul
Shanghai
Singapore
Tokyo

"Smart contracts" are a critical component of many platforms and applications being built using blockchain or distributed ledger technology. Below, we outline the background and functions of smart contracts, discuss whether they can be deemed enforceable legal agreements under contract law in the United States, and highlight certain legal and practical considerations that will need to be resolved before they can be broadly used in commercial contexts.

An Introduction to Smart Contracts

How Smart Contracts Function

"Smart contracts" is a term used to describe computer code that automatically executes all or parts of an agreement and is stored on a blockchain-based platform. As discussed further below, the code can either be the sole manifestation of the agreement between the parties or might complement a traditional text-based contract and execute certain provisions, such as transferring funds from Party A to Party B. The code itself is replicated across multiple nodes of a blockchain and, therefore, benefits from the security, permanence and immutability that a blockchain offers. That replication also means that as each new block is added to the blockchain, the code is, in effect, executed. If the parties have indicated, by initiating a transaction, that certain parameters have been met, the code will execute the step triggered by those parameters. If no such transaction has been initiated, the code will not take any steps. Most smart contracts are written in one of the programming languages directly suited for such computer programs, such as Solidity.

At present, the input parameters and the execution steps for a smart contract need to be specific and objective. In other words, if "x" occurs, then execute step "y." Therefore, the actual tasks that smart contracts are performing are fairly rudimentary, such as automatically moving an amount of cryptocurrency from one party's wallet to another when certain criteria are satisfied. As the adoption of blockchain spreads, and as more assets are tokenized or go "on chain," smart contracts will become increasingly complex and capable of handling sophisticated transactions. Indeed, developers already are stringing together multiple transaction steps to form more complex smart contracts. Nonetheless, we are, at the very least, many years away from code being able to determine more subjective legal criteria, such as whether a party satisfied a commercially reasonable efforts standard or whether an indemnifications clause should be triggered and the indemnity paid.

Before a compiled smart contract actually can be executed on certain blockchains, an additional step is required, namely, the payment of a transaction fee for the contract to be added to the chain and executed upon.

In the case of the Ethereum blockchain, smart contracts are executed on the Ethereum Virtual Machine (EVM), and this payment, made through the ether cryptocurrency, is known as "gas."[1] The more complex the smart contract (based on the transaction steps to be performed), the more gas that must be paid to execute the smart contract. Thus, gas currently acts as an important gate to prevent overly complex or numerous smart contracts from overwhelming the EVM.[2]

Developers already are stringing together multiple transaction steps to form more complex smart contracts.

Smart contracts are presently best suited to execute automatically two types of "transactions" found in many contracts: (1) ensuring the payment of funds upon certain triggering events and (2) imposing financial penalties if certain objective conditions are not satisfied. In each case, human intervention, including through a trusted escrow holder or even the judicial system, is not required once the smart contract has been deployed and is operational, thereby reducing the execution and enforcement costs of the contracting process.

As just one example, smart contracts could eliminate the so-called procure-to-pay gaps. When a product arrives and is scanned at a warehouse, a smart contract could immediately trigger requests for the required approvals and, once obtained, immediately transfer funds from the buyer to the seller. Sellers would get paid faster and no longer need to engage in dunning, and buyers would reduce their account payable costs. This could impact working capital requirements and simplify finance operations for both parties. On the enforcement side, a smart contract could be programmed to shut off access to an internet-connected asset if a payment is not received. For example, access to certain content might automatically be denied if payment was not received.

Historical Background

The term "smart contract" was first introduced by computer scientist and cryptographer Nick Szabo some 20 years ago as a graduate student at University of Washington. According to Szabo:

> New institutions, and new ways to formalize the relationships that make up these institutions, are now made possible by the digital revolution. I call these new contracts "smart," because they are far more functional than their inanimate paper-based ancestors. No use of artificial intelligence is implied. A smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises.[3]

Szabo's use of quotes around the word "smart" when comparing smart contracts to paper-based contracts, and his eschewing of artificial intelligence are important. Smart contracts may be "smarter" than paper contracts because they automatically can execute certain pre-programmed steps, but they should not be seen as intelligent tools that can parse a contract's more subjective requirements. Indeed, the classic example of a smart contract offered by Szabo is a vending machine. Once a purchaser has satisfied the conditions of the "contract" (*i.e.*, inserting money into the machine) the machine automatically honors the terms of the unwritten agreement and delivers the snack.

Smart contracts today also find their origin in Ricardian Contracts, a concept published in 1996 by Ian Grigg and Gary Howland as part of their work on the Ricardo payment system to transfer assets. Grigg saw Ricardian Contracts as a bridge between text contracts and code that had the following parameters: a single document that "is a) a contract offered by an issuer to holders, b) for a valuable right held by holders, and managed by the issuer, c) easily readable by people (like a contract on paper), d) readable by programs (parsable like a database), e) digitally signed, f) carries the keys and server information, and g) allied with a unique and secure identifier."[4]

[1] *See* "What is the 'Gas' in Ethereum?" *Cryptocompare*, November 18, 2016, available here.

[2] *Id.*

[3] Nick Szabo, "Smart Contracts: Building Blocks for Digital Market," 1996, available here.

[4] Ian Grigg, "The Ricardian Contract," available here.

The Interplay With Traditional Text Agreements

One of the difficulties with discussing smart contracts is that the term is used to capture two very different paradigms. The first involves smart contracts that are created and deployed without any enforceable text-based contract behind them. For example, two parties reach an oral understanding as to the business relationship they want to capture and then directly reduce that understanding into executable code. We refer to these below as "code-only smart contracts." The second paradigm involves the use of smart contracts as vehicles to effectuate certain provisions of a traditional text-based contract, in which the text itself references the use of the smart contract to effectuate certain provisions. We refer to these as "ancillary smart contracts."

Are Smart Contracts Enforceable?

There is no federal contract law in the United States; rather, the enforceability and interpretation of contracts is determined at the state level. Thus, while certain core principles apply consistently across state lines, and there has been a drive to harmonize state laws by the National Conference of Commissioners on Uniform State Laws, any conclusions regarding smart contracts must be tempered by the reality that states may adopt different views.

A discussion regarding the enforceability of smart contracts must start with the fundamental distinction between an agreement and a "contract." States generally recognize that although two parties can enter into a variety of "agreements," a contract means that the agreement is legally binding and enforceable in a court of law.[5] In order to determine enforceability, state courts traditionally look to whether the common law requirements of offer, acceptance and consideration are satisfied. These basic requirements surely can be achieved through ancillary smart contracts. For example, an insurer might develop a flight insurance product that automatically provides the insured with a payout if a flight is delayed by more than two hours.[6] The key terms, such as delineating how the delay is calculated, can be set forth in a text-based contract, with the actual formation of the contract (payment of the premium) and the execution (automatic payout upon a verifiable delay) handled through an ancillary smart contract. Here, the insurer has made a definite offer for a flight insurance product that is accepted by the insured upon payment of the premium as consideration.

Although, today, certain contracts must be in writing, and additional formalities may be required such as those under the Uniform Commercial Code (UCC) and state statutes of frauds,[7] agreements do not always need to be in writing to be held enforceable.[8] Thus, many code-only smart contracts also will be enforceable under state laws governing contracts. Szabo's example of a vending machine is instructive in this regard. There, while the buyer has many implied rights, a contract was formed without any meaningful written terms other than a price display for each item. Thus, the fact that an agreement is rendered only in code, such as the case with code-only smart contracts, presents no particular barrier to contract formation outside the barriers imposed by the UCC and statutes of frauds. Indeed, a variety of laws and legal constructs have long considered the role of information technology in contract formation.

For example, the Uniform Electronic Transactions Act (UETA) which dates back to 1999 and forms the basis for state law in 47 states, provides that, with limited exceptions, electronic records, which include records created by computer programs, and electronic signatures (*i.e.*, digital signature using public key encryption technology) be given the same legal effect as their written

[5] *See, e.g.*, "Restatement (Second) of Contracts," Section 1, American Law Institute, 1981. In the U.S., contract law is ordinarily a function of state law. Although this article outlines general contract law principles that are common across states, we note that state law differences may impact the enforceability of smart contracts in certain states.

[6] At least one company, AXA, currently offers such a product. *See* here.

[7] *See, e.g.*, UCC § 2-201.

[8] *See, e.g.*, *Lumhoo v. Home Depot USA, Inc.*, 229 F. Supp. 2d 121, 160 (E.D.N.Y. 2002) (holding that the plaintiffs adduced sufficient evidence to support an inference that the parties formed an oral contract for payment by their employer at an overtime rate for any hours worked in excess of eight hours per day).

counterparts.[9] UETA even goes so far as recognizing the validity of "electronic agents," which it defines as "a computer program or an electronic or other automated means used independently to initiate an action or respond to electronic records or performances in whole or in part, without review or action by an individual."[10] Under UETA, an electronic agent is "capable within the parameters of its programming, of initiating, responding or interacting with other parties or their electronic agents once it has been activated by a party, without further attention of that party,"[11] arguably a prescient acknowledgment of smart contracts.

Similarly, the federal Electronic Signatures Recording Act (E-Sign Act) not only recognizes the validity of electronic signatures and electronic records in interstate commerce, but also provides that a contract or other record relating to a transaction "may not be denied legal effect, validity, or enforceability solely because its formation, creation, or delivery involved the action of one or more electronic agents so long as the action of any such electronic agent is legally attributable to the person to be bound."[12] The term "electronic agent" means a computer program or an electronic or other automated means used independently to initiate an action or respond to electronic records or performances in whole or in part without review or action by an individual at the time of the action or response."[13]

Though an understanding of the current legal framework is important to evaluating the enforceability of smart contracts today, those using smart contracts in the future may not need to rely on laws that pre-date the development of blockchain technology. Arizona and Nevada already have amended their respective state versions of UETA to explicitly incorporate blockchains and smart contracts.[14] The fact that these states have adopted decidedly different definitions of those critical terms suggests that as more states follow their lead, there may be increasing pressure to adopt unified definitions to reflect blockchain and smart contract developments.

Challenges With the Widespread Adoption of Smart Contracts

Given the existing legal frameworks for recognizing electronic contracts, it is quite likely that a court today would recognize the validity of code that executes provisions of a smart contract — what we have classified as ancillary smart contracts. There is also precedent to suggest that a code-only smart contract might enjoy similar legal protection. The challenge to widespread smart contract adoption may therefore have less to do with the limits of the law than with potential clashes between how smart contract code operates and how parties transact business. We set forth below certain of these challenges.

A variety of laws and legal constructs have long considered the role of information technology in contract formation.

How Can Non-Technical Parties Negotiate, Draft and Adjudicate Smart Contracts?

A key challenge in the widespread adoption of smart contracts is that parties will need to rely on a trusted, technical expert to either capture the parties' agreement in code or confirm that code written by a third party is accurate. While some analogize this to hiring a lawyer to explain "the legalese" of a traditional text-based contract, the analogy is misplaced. Non-lawyers typically can understand simple short-form agreements as well as many provisions of longer agreements, especially those setting forth business terms. But a non-programmer would be at a total loss to understand even the most basic smart contract and is therefore significantly more beholden to an expert to explain what the contract "says."

To some extent, the inability of contracting parties to understand the smart contract code will not be a hindrance to entering into ancillary code agreements.

[9] Uniform Electronic Transactions Act (Unif. Law Comm'n 1999). New York, Illinois and Washington have state-specific laws relating to the validity of electronic transactions.

[10] *Id.* § 2(6).

[11] *Id.* § 2 cmt. 5.

[12] 15 U.S.C. § 7001(h).

[13] 15 U.S.C. § 7006(3).

[14] *See* 2017 Ariz. HB 2417 44-7061 and Nev. Rev. Stat. Ann. § 719.090.

This is because for many basic functions, text templates can be created and used to indicate what parameters need to be entered and how those parameters will be executed. For example, assume a simple smart contract function that extracts a late fee from a counterparty's wallet if a defined payment is not received by a specified date. The text template could prompt the parties to enter the amount of the expected payment, the due date and the amount of the late fee. However, a party may want to confirm that the underlying code actually will perform the functions specified in the text, and that there are no additional conditions or parameters — especially where the template disclaims any liability arising from the accuracy of the underlying code. This review will require a trusted third party with programming expertise.

The challenge to widespread smart contract adoption may have less to do with the limits of the law than with potential clashes between how smart contract code operates and how parties transact business.

In cases where such templates do not exist, and new code must be developed, the parties will need to communicate the intent of their agreement to a programmer. Simply handing that programmer a copy of the legal agreement would be inefficient since it would require the programmer to try and decipher a legal document. Parties relying on ancillary smart contracts therefore may need to draft a separate "term sheet" of functionality that the smart contract should perform and that can be provided to the programmer.

The parties also may want written representations from the programmer that the code performs as contemplated. The net result is that for customized arrangements that do not rely on an existing template, the parties may need to enter into a written agreement with the smart contract programmer, not unlike the contract that parties may enter into with a provider of services for Electronic Data Interchange (EDI) transactions today.

When parties fail to review the code, they run the risk of agreeing to terms that they did not know formed part of the smart contract. For example, in a recent transaction, a Singaporean company (Soar Labs) purchased a stake in an Australian crypto-exchange company (Byte Power Party) using Soar Lab's Ethereum-based tokens as consideration. Soon after Soar Labs accused Byte Power Party of acting in violation of its agreement, Soar Labs took back its tokens. Soar Labs did not appear to violate the terms of the smart contract. In fact, the code permitted Soar Labs to take back its tokens, something Byte Power Party was not aware of. When asked why Soar Labs did not inform Byte Power Party of this term in the smart contract, Soar Labs' CEO said that Byte Power Party should have looked at the code. The transaction serves as a cautionary tale for parties eager to enter into smart contracts without confirming that the code reflects the intent of the parties.

Insurance companies could also create policies to protect contracting parties from the risk that smart contract code does not perform the functions specified in the text of an agreement. Although the parties would also want to review (or have third parties review) the code, insurance can provide additional protection given that the parties might miss errors when reviewing the code. The parties would also take some additional comfort from the fact that the insurance company likely conducted its own code audit before agreeing to insure the code.

Code-only smart contracts used for business-to-consumer transactions could pose an additional set of issues that will need to be addressed. Courts are wary of enforcing agreements where the consumer did not receive adequate notice of the terms of the agreement,[15] and may be hesitant to enforce a smart contract where the consumer was not also provided with an underlying text agreement that included the complete terms.

[15] *See, e.g.*, *Nicosia v. Amazon.com, Inc.*, 834 F.3d 220 (2d Cir. 2016) (reversing the district court's dismissal for failure to state a claim and holding that reasonable minds could disagree as to whether Amazon provided the consumer with reasonable notice of the mandatory arbitration provision at issue).

Finally, as the validity or performance of smart contracts increasingly become adjudicated, courts may need a system of court-appointed experts to help them decipher the meaning and intent of the code. Today, parties routinely use their own experts when technical issues are at the center of a dispute. While both federal courts and many state courts have the authority to appoint their own experts, they rarely exercise that authority.[16] That approach may need to change if the number of standard contract disputes that center on interpreting smart contract code increases.

Smart Contracts and the Reliance on "Off-Chain" Resources

Many smart contract proposed use-cases assume that the smart contract will receive information or parameters from resources that are not on the blockchain itself — so-called off-chain resources. For example, assume a crop insurance smart contract is programmed to transfer value to an insured party if the temperature falls below 32 degrees at any point. The smart contract will need to receive that temperature data from an agreed source. This presents two issues. First, smart contracts do not have the ability to pull data from off-chain resources; rather, that information needs to be "pushed" to the smart contract. Second, if the data at issue is in constant flux, and since the code is replicated across multiple nodes across the network, different nodes may be receiving different information, even just a few seconds apart. In our example, Node-1 may receive information that the temperature is 31.9 degrees, while Node-2 may receive information that the temperature is actually 32 degrees. Given that consensus is required across the nodes for a transaction to be validated, such fluctuations can cause the condition to be deemed "not satisfied."

Contracting parties will be able to solve this conundrum by using a so-called "oracle." Oracles are trusted third parties that retrieve off-chain information and then push that information to the blockchain at predetermined times. In the foregoing example, the oracle would monitor the daily temperature, determine that the freezing event has occurred and then push that information to the smart contract.

> Oracles are trusted third parties that retrieve off-chain information and then push that information to the blockchain at predetermined times.

Although oracles present an elegant solution to accessing off-chain resources, this process adds another party with whom the parties would need to contract to effectuate a smart contract, thus somewhat diluting the decentralized benefits of smart contracts. It also introduces a potential "point of failure." For example, an oracle might experience a system flaw and be unable to push out the necessary information, provide erroneous data or simply go out of business. Smart contracts will need to account for these eventualities before their adoption can become more widespread.

What Is the "Final" Agreement Between the Parties?

When analyzing traditional text-based contracts, courts will examine the final, written document to which the parties have agreed in order to determine whether the parties are in compliance or breach. Courts have long emphasized that it is this final agreement that represents the mutual intent of the parties — the "meeting of the minds."

In the case of code-only smart contracts, the code that is executed — and the outcome it produces — represents the only objective evidence of the terms agreed to by the parties. In these cases, email exchanges between the parties as to what functions the smart contract "should" execute, or oral discussions to that effect, likely would yield to the definitive code lines as the determinative manifestation of the parties' intent.

[16] *See* Charles Alan Wright & Arthur R. Miller, *Federal Practice and Procedure,* Section 6304 (3d ed. supp. 2011) ("In fact, the exercise of Rule 706 powers is rare under virtually any circumstances. This is, at least in part, owing to the fact that appointing an expert witness increases the burdens of the judge, increases the costs to the parties, and interferes with the adversarial control over the presentation of evidence."), and Stephanie Domitrovich, Mara L. Merino & James T. Richardson, *State Trial Judge Use of Court Appointed Experts: Survey Results and Comparisons*, 50 Jurimetrics J. 371, 373–74 (2010).

With respect to ancillary smart contracts, a court likely would look at the text and code as a unified single agreement. The issue becomes complicated when the traditional text agreement and the code do not align. In the crop insurance example described above, assume the text of an agreement specifies that an insurance payout will be made if the temperature falls below 32 degrees, while the smart contract code triggers the payment if the temperature is equal to or below 32 degrees. Assuming that the text agreement does not state whether the text or code controls in the event of an inconsistency, courts will need to determine — perhaps on a case-by-case basis — whether the code should be treated as a mutually agreed amendment to the written agreement or whether the text of the agreement should prevail. In some respects, the analysis should be no different than a case where the provisions of a main agreement differ from what is reflected in an attached schedule or exhibit. The fact that here the conflict would be between text and computer code and not two text documents should not be determinative, but courts may take a different view.

One solution will be for parties to use a text based contract where the parameters that trigger the smart contract execution are not only visible in the text but actually populate the smart contract. In our example, "less than 32 degrees" would not only be seen in the text, but also would create the parameter in the smart contract itself, thereby minimizing the chances of any inconsistency.

One of the key attributes of smart contracts is their ability to automatically and relentlessly execute transactions without the need for human intervention.

The Automated Nature of Smart Contracts

One of the key attributes of smart contracts is their ability to automatically and relentlessly execute transactions without the need for human intervention. However, this automation, and the fact that smart contracts cannot easily be amended or terminated unless the parties incorporate such capabilities during the creation of the smart contract, present some of the greatest challenges facing widespread adoption of smart contracts. For example, with traditional text contracts, a party can easily excuse a breach simply by not enforcing the available penalties. If a valued customer is late with its payment one month, the vendor can make a real-time decision that preserving the long-term commercial relationship is more important than any available termination right or late fee. However, if this relationship had been reduced to a smart contract, the option not to enforce the agreement on an *ad hoc* basis likely would not exist. A late payment will result in the automatic extraction of a late fee from the customer's account or the suspension of a customer's access to a software program or an internet-connected device if that is what the smart contract was programmed to do. The automated execution provided by smart contracts might therefore not align with the manner in which many businesses operate in the real world.

Similarly, in a text-based contractual relationship, a party may be willing to accept, on an *ad hoc* basis, partial performance to be deemed full performance. This might be because of an interest in preserving a long-term relationship or because a party determines that partial performance is preferable to no performance at all. Here, again, the objectivity required for smart contract code might not reflect the realities of how contracting parties interact.

Amending and Terminating Smart Contracts

At present, there is no simple path to amend a smart contract, creating certain challenges for contracting parties. For example, in a traditional text-based contract, if the parties have mutually agreed to change the parameters of their business deal, or if there is a change in law, the parties quickly can draft an amendment to address that change, or simply alter their course of conduct. Smart contracts currently do not offer such flexibility. Indeed, given that blockchains are immutable, modifying a smart contract is far more complicated than modifying standard software code that does not

reside on a blockchain. The result is that amending a smart contract may yield higher transaction costs than amending a text-based contract, and increases the margin of error that the parties will not accurately reflect the modifications they want to make.

Similar challenges exist with respect to terminating a smart contract. Assume a party discovers an error in an agreement that gives the counterparty more rights than intended, or concludes that fulfilling its stated obligations will be far more costly than it had expected. In a text-based contract, a party can engage in, or threaten, so-called "efficient breach," *i.e.*, knowingly breaching a contract and paying the resulting damages if it determines that the cost to perform is greater than the damages it would owe. Moreover, by ceasing performance, or threatening to take that step, a party may bring the counterparty back to the table to negotiate an amicable resolution. Smart contracts do not yet offer analogous self-help remedies.

Projects are currently underway to create smart contracts that are terminable at any time and more easily amended. While in some ways this is antithetical to the immutable and automated nature of smart contracts, it reflects the fact that smart contracts only will gain commercial acceptance if they reflect the business reality of how contracting parties act.

Objectivity and the Limits of Incorporating Desired Ambiguity Into Smart Contracts

The objectivity and automation required of smart contracts can run contrary to how business parties actually negotiate agreements. During the course of negotiations, parties implicitly engage in a cost-benefit analysis, knowing that at some point there are diminishing returns in trying to think of, and address, every conceivable eventuality. These parties no longer may want to expend management time or legal fees on the negotiations, or may conclude that commencing revenue generating activity under an executed contract outweighs addressing unresolved issues. Instead, they may determine that if an unanticipated event actually occurs, they will figure out a resolution at that time. Similarly, parties may purposefully opt to leave a provision somewhat ambiguous in an agreement in order to give themselves the flexibility to argue that the provision should be interpreted in their favor. This approach to contracting is rendered more difficult with smart contracts where computer code demands an exactitude not found in the negotiation of text-based contracts. A smart contract cannot include ambiguous terms nor can certain potential scenarios be left unaddressed. As a result, parties to smart contracts may find that the transaction costs of negotiating complex smart contracts exceed that of a traditional text-based contracts.

The objectivity and automation required of smart contracts can run contrary to how business parties actually negotiate agreements.

It will take some time for those adopting smart contracts in a particular industry to determine which provisions are sufficiently objective to lend themselves to smart contract execution. As noted, to date, most smart contracts perform relatively simple tasks where the parameters of the "if/then" statements are clear. As smart contracts increase in complexity, parties may disagree on whether a particular contractual provision can be captured through the objectivity that a smart contract demands.

Do Smart Contracts Really Guarantee Payment?

One benefit often touted of smart contracts is that they can automate payment without the need for dunning notices or other collection expenses and without the need to go to court to obtain a judgment mandating payment. While this is indeed true for simpler use cases, it may be less accurate in complex commercial relationships. The reality is that parties are constantly moving funds throughout their organization and do not "park" total amounts that are due on a long-term contract in anticipation of future payment requirements. Similarly, a person obtaining a loan is unlikely to keep the full loan amount in a specified wallet linked to the smart contract. Rather, the borrower will put those funds to use, funding the necessary repayments on an *ad hoc* basis.

If the party owing amounts under the smart contract fails to fund the wallet on a timely basis, a smart contract looking to transfer money from that wallet upon a trigger event may find that the requisite funds are not available. Implementing another layer into the process, such as having the smart contract seek to pull funds from other wallets or having that wallet "fund itself" from other sources, would not solve the problem if those wallets or sources of funds also lack the requisite payment amounts. The parties might seek to address this issue through a text-based requirement that a wallet linked to the smart contract always have a minimum amount, but that solution simply would give the party a stronger legal argument if the dispute was adjudicated. It would not render the payment operation of the smart contract wholly automatic. Thus, although smart contracts will render payments far more efficient, they may not eliminate the need to adjudicate payment disputes.

One of the key promises of blockchain technology, and by extension smart contracts, is the development of robust, decentralized and global platforms.

Risk Allocation for Attacks and Failures

Smart contracts introduce an additional risk that does not exist in most text-based contractual relationships — the possibility that the contract will be hacked or that the code or protocol simply contains an unintended programming error. Given the relative security of blockchains, these concepts are closely aligned; namely, most "hacks" associated with blockchain technology are really exploitations of an unintended coding error. As with many bugs in computer code, these errors are not glaring, but rather become obvious only once they have been exploited. For example, in 2017 an attacker was able to drain several multi-signature wallets offered by Parity of $31 million in ether.[17] Multi-signature wallets add a layer of security because they require more than one private key to access the wallet. However, in the Parity attack, the attacker was able to exploit a flaw in the Parity code by reinitializing the smart contract and making himself or herself the sole owner of the multi-signature wallets. Parties to a smart contract will need to consider how risk and liability for unintended coding errors and resulting exploitations are allocated between the parties, and possibly with any third-party developers or insurers of the smart contract.

Governing Law and Venue

One of the key promises of blockchain technology, and by extension smart contracts, is the development of robust, decentralized and global platforms. However, global adoption means that parties may be using a smart contract across far more jurisdictions than might exist in the case of text-based contracts. The party offering terms under a smart contract would therefore be best-served by specifying the governing law and venue for that smart contract. A governing law provision specifies what substantive law will apply to the interpretation of the smart contract, whereas a venue clause specifies which jurisdiction's courts will adjudicate the dispute. In cases where governing law or venue is not specified, a plaintiff may be relatively unconstrained in choosing where to file a claim or in arguing which substantive law should apply given the wide range of jurisdictions in which a smart contract might be used. Given that many early disputes concerning smart contracts will be ones of first-impression, contracting parties will want some certainty surrounding where such disputes will be adjudicated.

Best Practices

Given that we are at the nascent stages of smart contract adoption, best practices for implementing such code is still evolving. However, the checklist below should help developers design effective smart contracts and guide companies who plan to use them.

[17] *See* Haseeb Qureshi, "A Hacker Stole $31M of Ether—How it Happened, and What it Means for Ethereum," *FreeCodeCamp*, (July 20, 2017), available here.

- For now, parties entering into any type of contractual arrangement would be best served using a hybrid approach that combines text and code. As noted, there are strong arguments that code-only smart contracts should be enforceable, at least under state contract law in the U.S. However, until there is greater clarity on their validity and enforceability, code-only smart contracts should be used only for simpler transactions. Parties will continue to want text versions of agreements so they can read the agreed-upon terms, memorialize terms that smart contracts are not equipped to address and have a document they know a court will enforce.
- In a hybrid contract using text and code, the text should clearly specify the smart contract code with which it is associated, and the parties should have full visibility into the variables that are being passed to the smart contract, how they are defined and the transaction events that will trigger execution of the code.
- When relying on oracles for off-chain data, the parties should address what would happen if the oracle is unable to push out the necessary data, provides erroneous data or simply goes out of business.
- The parties should consider risk allocation in the event of a coding error.
- The text agreement accompanying the code should specify the governing law and venue, as well as the order of precedence between text and code in the event of a conflict.
- The text agreement should include a representation by each party that they have reviewed the smart contract code, and that it reflects the terms found in the text agreement. Although such a representation cannot force a party to examine the code, it will help the counterparty defend against a claim that the code was never reviewed. Parties may also choose to insure against the risk that the code contains errors. As noted, parties may need to involve third-party experts to review the code.

Future of Smart Contracts

Today, smart contracts are a prototypical example of "Amara's Law," the concept articulated by Stanford University computer scientist Roy Amara that we tend to overestimate new technology in the short run and underestimate it in the long run. Although smart contracts will need to evolve before they are widely adopted for production use in complex commercial relationships, they have the impact to revolutionize the reward and incentive structure that shapes how parties contract in the future. To that end, and when thinking about smart contracts, it is important not to simply think how existing concepts and structures can be ported over to this new technology. Rather, the true revolution of smart contracts will come from entirely new paradigms that we have not yet envisioned.

Contacts

Stuart D. Levi
Partner / New York
212.735.2750
stuart.levi@skadden.com

Alex B. Lipton
Associate / New York
212.735.3006
alex.lipton@skadden.com

Four Times Square
New York, NY 10036
212.735.3000