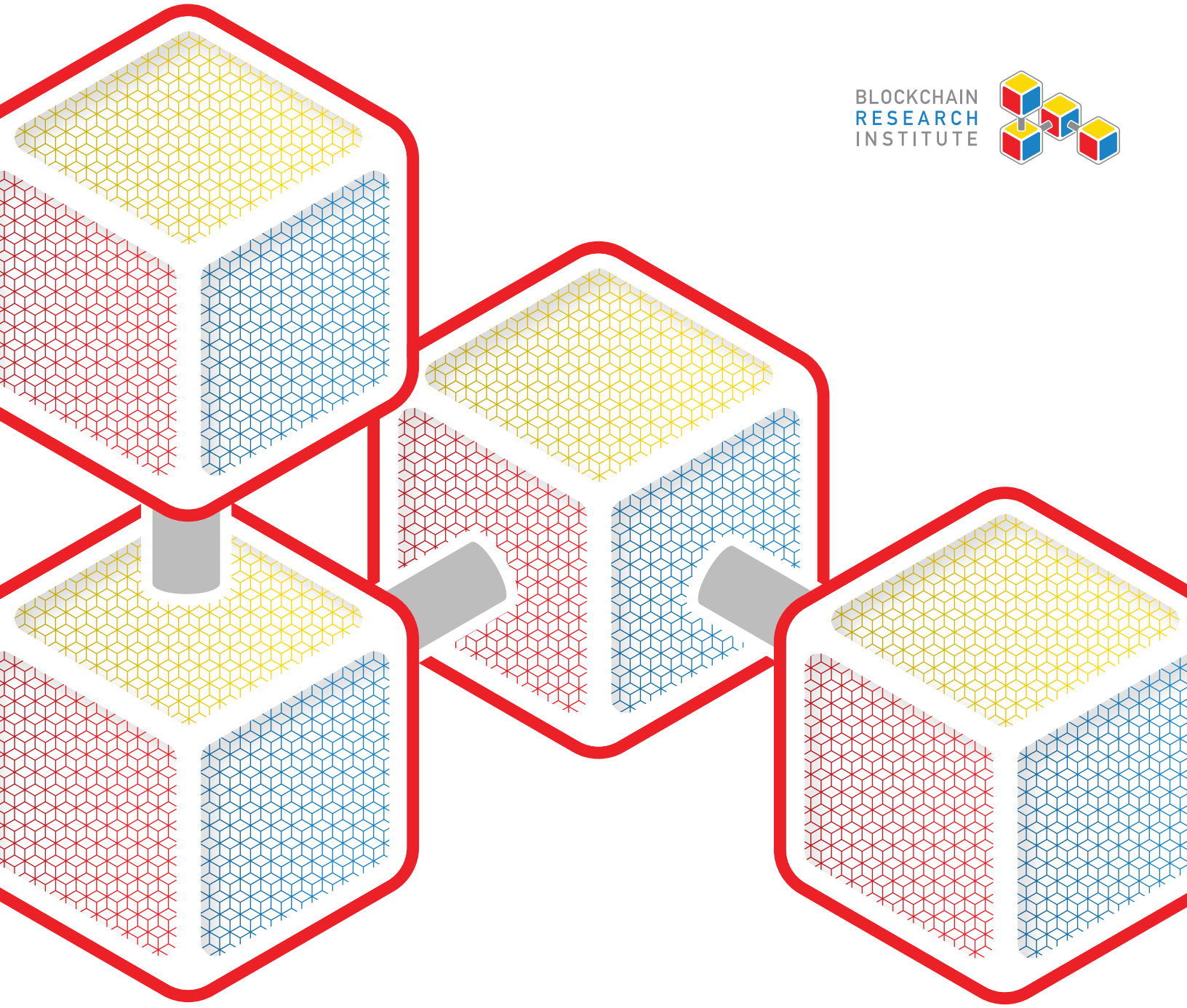
BLOCKCHAIN
RESEARCH
INSTITUTE

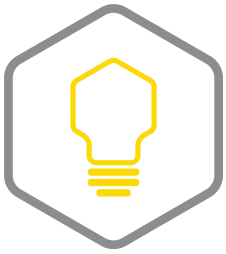


WINNING STRATEGIES FOR SMART CONTRACTS

Nick Szabo

December 2017

A BLOCKCHAIN RESEARCH INSTITUTE BIG IDEA WHITEPAPER



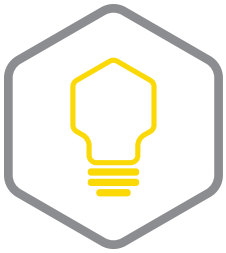
Realizing the new promise of the digital economy

In 1994, Don Tapscott coined the phrase, "the digital economy," with his book of that title. It discussed how the Web and the Internet of information would bring important changes in business and society. Today the Internet of value creates profound new possibilities.

In 2017, Don and Alex Tapscott launched the Blockchain Research Institute to help realize the new promise of the digital economy. We research the strategic implications of blockchain technology and produce practical insights to contribute global blockchain knowledge and help our members navigate this revolution.

Our findings, conclusions, and recommendations are initially proprietary to our members and ultimately released to the public in support of our mission. To find out more, please visit www.blockchainresearchinstitute.org.

Contents



Foreword

No one is better equipped to write about smart contracts than Nick Szabo, the brilliant inventor of the concept. Anyone who reads this seminal paper will agree with me that it is the single most important work on the topic since Nick's original publication over 20 years ago.

I'm a big believer in developing taxonomy to elucidate new concepts, and Nick does this masterfully. He looks at smart contracts across the deal cycle, distinguishes smart contracts from traditional ones, and teases out their relationships to traditional legal systems. He also explains why identity doesn't scale well and why conventional computer architecture is so vulnerable.

The paper is a practical one. Nick details the most important considerations for designing smart contracts. He gives us a tour of several areas of application—financial networks, commerce, application tokens, decentralized autonomous organizations (DAOs), logistics and supply chains, and algorithms to manage people and performance. Nick nicely summarizes best practices for their use.

Smart contracts will empower the multinational small business and enable larger multinationals to reach out to more culturally, linguistically, and jurisdictionally diverse populations. Overall, Nick argues convincingly that smart contracts can transform business relationships for the better.

DON TAPSCOTT
Co-Founder and Executive Chairman
Blockchain Research Institute

Idea in brief

This paper provides readers with clear definitions, a common language, and a framework for understanding and discussing smart contracts. It examines their role in the deal cycle and the social scalability of economic coordination and productivity, the differences between smart and traditional contracts, and their relationship to traditional legal systems. It explains why identity scales so poorly, why traditional computer architecture is so insecure, and how blockchains can solve security problems for smart contracts.

The paper details the most important smart contract design considerations, including the use of hardware wallets and oracles. It explores several important design patterns and areas of application—peripheral financial networks and financial plumbing, worry-minimized commerce, application tokens, decentralized autonomous organizations and multiple signature authority, logistics and supply chain innovation, and management by algorithm—and summarizes strategies and best practices for their use.

Smart contracts will empower the multinational small business and make larger multinationals able to reach out much further into more culturally, linguistically, and jurisdictionally diverse populations. This paper shows how smart contracts can expand reach and transform business relationships.

Introduction

In 1994, I coined the phrase *smart contract*[1] to describe the combination of secure protocol with user interface, inspired by contract law and practice, that would be able to formalize and secure commercial deals over computer networks.[2] Since then, I've explored and developed the concept of smart contracts and have tested many of these ideas. With the realization of blockchain technology in 2009, opportunities and implementations for smart contracts have proliferated. Definitions of smart contracts have also proliferated, some of them useful, some counterproductive, others just plain wrong.

What are smart contracts?

There are two useful definitions of a smart contract. The first is a machine with rules that we could have defined in a contract but instead wrote into a machine. For example, we could think of a vending machine as a contract. Our vending machine is old fashioned: a purely mechanical contraption that dispenses sodas for ten cents. Writing this machine logic as a contract might read as follows: "If the party of the first puts in two nickels, said party shall be dispensed a soda. If said party puts in a dime, said party gets a soda. If said party puts in a quarter, said party gets back a dime, a nickel, and a soda." Writing such objective and computable terms and conditions into a contract would be tedious, so we design it into the machine instead.

We could think of a vending machine as a contract.

With the Internet, smartphones, blockchains, sensors, cryptography, and other recent technological breakthroughs, we are in an era where we have a much wider variety of ways to negotiate terms and incentivize performance. We can now implement—in this vending-machine way—many terms and conditions traditionally handled by contracts. Using technology, we can augment many contracts to help us verify and incentivize performance and to protect the ability to perform. These capabilities make many new kinds of contractual relationships and deal cycles possible that were previously uneconomical or inaccessible.

Let's continue with the vending machine example and compare it to the general concept of smart contracts. One party chooses to perform by putting in coins. The machine then verifies this performance and responds with its contractual performance by dispensing the goods. More generally, parties to a smart contract negotiate their mutual promises (what future actions they expect to be done, by whom, and under what conditions). In a simplified situation (like the vending machine or most other retail commerce), one party offers a take-it-or-leave-it smart contract and the other party simply decides whether to accept it or not. Once the smart contract is formed, the machine that it runs on—whether a traditional computer or a blockchain virtual machine—determines when these conditions have been met and performs the promise or (more often for smart contracts) verifies and incentivizes performance by humans and/or machines outside the smart contract. The smart contract securely manages assets and changes control of them conditional on performance. In terms of computer technology, smart contracts are software modeled on contractual relationships that facilitate negotiations and incentivize performance via control of assets.

In terms of computer technology, smart contracts are software modeled on contractual relationships that facilitate negotiations and incentivize performance via control of assets.

This paper examines smart contracts in the full cycle of a contractual deal. It begins with search: matching buyer and seller. In the case of our immobile vending machine, this matching occurs by the machine owner's placing the vending machine in a high-traffic area and the manufacturer's advertising its products in various media. Negotiation is simplified into a standard take-it-or-leave-it deal. Formation of the particular smart contract with particular terms, as well as the initial performance, occurs when the customer inserts coins and selects a soda. The machine verifies performance (payment) and automates

the counter-performance (dispensing the correct soda) conditional on that first performance. The machine controls both the coins, after they have been inserted, and the sodas. For on-blockchain assets, we can achieve this control with much greater transparency and security. Think of a public blockchain as a vending machine that is fully transparent, yet bullet- and burglar-proof.

The second definition of a smart contract is an application that runs in a distributed and trust-minimized manner on a blockchain. We say that an application is a dec*entralized application* or "Dapp" if it runs on a secure consensus protocol across a network of computers rather than on an individual remote computer or centralized server. Dapps typically control *on-chain assets* such as cryptocurrencies or tokens. We say that a feature is *trust-minimized* if we can trust the feature while placing minimal trust in any particular person or organization to secure or not attack that feature. It is another way of saying that the feature is *secure* in the most general way possible. Dapps are trust-minimized because we can trust the code to run properly without having to trust the owners of any of the computers the code runs on. If we haven't read or understood the code ourselves, we are still trusting in the software development process, during which multiple eyeballs should review the code for security.

While smart contracts encompass all phases of the deal cycle, the core is the performance phase.

Since smart contracts have become closely associated with blockchains, and blockchains are indeed the most secure and reliable machine to run smart contracts, we often see the phrase, "smart contract," used to refer to any computer program that runs on a blockchain. That definition differs from the first definition ("a machine with rules embedded"), though they overlap. This paper uses "Dapp" to refer to smart contracts according to this second definition. Dapps ("smart contracts" by our second definition) are the most secure and reliable way to run the core fiduciary code of smart contracts (our first definition). As this paper describes below, twenty-first century technology, in particular public blockchain technology, allows more transparent and secure control over assets as well as more insight into counterparty behavior and collateral levels, than is possible by trusting individual remote computers.

In short, smart contracts should not only run on public blockchains (where they have the highest security); they should also facilitate composition and negotiation. Dapps often fail to do this; they typically come in take-it-or-leave it forms, and they require programming skills. The company Global Financial Access is developing technology to facilitate negotiation and composition of on-chain financial smart contracts by non-programmers.

While smart contracts encompass all phases of the deal cycle, the core is the performance phase. Smart contracts sometimes automate performance, as with the vending machine's dispensing a soda; but, more often, they verify and incentivize performance, as with verifying the user's inserting coins and the machine's rewarding the user with a soda conditional on that payment.

Why make smart contracts?

Why should we write and enter into smart contracts? Let us step back and ask: why do we make traditional contracts? Contracts have been a crucial underpinning of commerce for thousands of years. As William Markham observed, "Contract law lies at the heart of our system of laws and serves as the foundation of our entire society."[3]

Why should we write and enter into smart contracts? Let us step back and ask: why do we make traditional contracts?

Both parties to a contract, whether smart or traditional, want to enter a business relationship, however ephemeral or long-term, that they expect will make them both better off. Let's call our parties Alice and Bob. Alice and Bob are generally not wanting to play a mere zero-sum game of poker or craps, unless they happen to enjoy those games and are willing to lose money, in the net, for the joy of playing. Rather, in a typical contract, they expect a *positive-sum game*, a business deal that will make them both better off, even though both may be more strategic and combative with each other than if they were playing a game, and ignorant of each other's intentions.

Different people usually value the same things differently, and the same things have a different value in different combinations. If Alice can play a guitar and Bob cannot, that guitar is more valuable to Alice than to Bob. Factory machinery is more valuable when combined with skilled labor and raw materials than when it is sitting idle. Combining these diverse people and objects into these value-producing combinations requires doing deals, and doing deals requires laying ground rules of the relationship, rules that participants are motivated to follow by the threat of lawsuit. "Do the things you promised or else I will take you to court." These rules provide incentives to follow through with actions that were mutually desired when the contract was formed. We have traditionally used contracts to document these rules and incentives, and we have used contract law to frame and enforce them.[4]

Contracts are the foundation of our economy. *Freedom of contract* is an important idea both with traditional and smart contracts. The main inspiration and ideas of smart contracts come from traditional contracts, whence the phrase. As the law professors Rohwer and Skrocki observed,

> *Freedom of contract, the freedom of individuals and enterprises to make their own economic arrangements with each other, is a fundamental prerequisite of a free society. This notion of contract law being a fundamental freedom is easily overlooked in a community in which this right is taken for granted. However, adoption and enforcement of laws permitting freedom of contract have been among the major necessities in countries that have been evolving from a command economy to a market economy in recent decades.*[5]

Contracts are the foundation of our economy.

As attorney Sheila Baran has observed, contract law allows you to "write your own laws."[6] Two individuals or businesses can negotiate, agree on, and write up the rules of their commercial relationship.

Smart contracts, like traditional contracts, incentivize performance through rules that specify what happens to money and other assets under various conditions, including whether parties have performed their tasks. With smart contracts, these conditions must be verifiable by sensor and computer. In some cases (usually financial), smart contracts can also automate performance. Smart contracts are very much in the contract law spirit of writing our own rules. We can customize and agree on the ground rules of our business relationships. That's what a traditional contract does, that's what smart contracts do.

In many potential cross-border deals on the global Internet, traditional local legal systems cannot provide this assurance of a positive-sum relationship. The vast majority of individuals and small businesses cannot afford the cost of a cross-border lawsuit, nor can most afford the cost of a traditional intermediary that can bridge both systems. That is why the phrase, multinational business, is still almost synonymous with medium and large sized corporations. Only they can afford the legal expertise required to invoke the traditional law across more than one culture and across more than one language and jurisdiction. Smart contracts will empower the multinational small business and make larger multinationals able to reach out much further into more culturally, linguistically, and jurisdictionally diverse populations. This paper explores how smart contracts can expand reach.

Smart contracts will empower the multinational small business and make larger multinationals able to reach out much further into more culturally, linguistically, and jurisdictionally diverse populations.

Much of this paper compares and contrasts the strengths and weaknesses of traditional contracts and smart contracts. Let's preview a few of the advantages of the latter.

The first important benefit of smart contracts is reducing mental transaction costs, that is, doing what the human mind cannot do efficiently, accurately, or dispassionately on its own. For example, contracts for difference (CFDs) continually compute cash flows based on price differences. This is infeasible to do in the human mind; we want computers, not people, to do that. CFDs are a whole category of contract that couldn't exist without computers. Another example is Uber's price algorithm, which estimates the costs of driving a given route and computes a take-it-or-leave-it price, thereby saving the driver and rider from haggling. If Uber drivers or prospective passengers had to do the math behind every price, the service would never have taken off and we'd continue to rely on taxi rates and meters.

Another benefit of smart contracts is increased predictability. Smart contracts, drafted in a mathematical language executable by machine, enable more accurate loss expectations and risk management in areas such as cross-border finance where legal or political uncertainties are high.

The third main benefit of smart contracts is broad security. We need to think about the security of all aspects and functions of our business relationships—all the things important to us, not just of some particular kinds of things. For example, in the financial

community, the costs of trust and security often fall under the category of counterparty risks, but there are also intermediary costs and risks, including high costs from intermediaries collecting quasi-monopoly rents. Traditional contracts tend to leave holes and are disconnected from actual control over assets, whereas smart contracts are based on control over assets and can provide broad security in our business dealings.

Blockchain-based smart contracts typically focus on the performance stage, shaping incentives during or just after performance (e.g., by transferring or freeing hypothecated or escrowed assets).

The deal cycle: Innovations of the Internet era

Many of the most important economic revolutions in history have been facilitated by new ways for humans to interact with each other, reducing vulnerability (the need for mutual trust), or increasing the quality or quantity of information flow (whether in the form of prices or language), thereby improving one or more phases of the contracting process: search, negotiation, performance, and post-performance incentivization. Blockchain-based smart contracts typically focus on the performance stage, shaping incentives during or just after performance (e.g., by transferring or freeing hypothecated or escrowed assets).

Contract phases

Successful Internet companies can usually be characterized by the phases of a deal (economist's "contract") they improve the most (see Figure 1, "The deal cycle," next page). The phases of the deal are:

1. Search: During this phase, buyers and sellers find and assess each other. The most radical gains made possible by the pre-blockchain Internet came in this phase, deriving from the raw ability of the Internet to connect nearly everybody together and tools such as search engines for them to find each other. Search phase innovations that match buyers with sellers, vis-à-vis the pre-Internet era, have included very humble and boring but powerful capabilities that we now take for granted: text search to find products on sites such as Amazon and eBay, Google's ability to match purchased ad keywords with text searches, and Lyft's and Uber's use of smartphone global positioning system (GPS) and maps to match riders and drivers.

2. Negotiation: At the end of this phase, the terms of the contract have been agreed and committed to. Computer and the Internet have also made possible substantial innovations in the negotiation phase. These have included eBay's collectibles auctions, Google's keyword auctions, and various price-setting algorithms. The latter are especially valuable for situations that involve variable costs but require take-

it-or-leave-it prices (as is common in retail), such as Uber's price-setting algorithm, which factors in distance, driving conditions, and other transportation data it gathers into its computers.

3. Performance: At the end of this phase, the terms of the contract have been performed. This phase includes collateral management (the attachment, freeing, and/or seizing of collateral to incentivize performance). So far during the Internet and smartphone eras, comparatively few innovations have occurred in this phase. Regulation has stifled Innovation in payment systems. Automation of performance continues on its long road of progress since the early days of the industrial revolution, but the pace of this progress pales in comparison to the pace of innovation brought about by the Internet and smartphones in the search phase and, to some extent, in the negotiations and post-performance stages over the last few decades. Thanks to advancements in computers, sensors, and blockchain technology, verification of performance, especially in areas such as finance and logistics, has a far vaster potential now than it did a few decades ago, the potential of which we have so far barely scratched the surface.

Thanks to advancements in computers, sensors, and blockchain technology, verification of performance, especially in areas such as finance and logistics, has a far vaster potential now than it did a few decades ago.

4. Post-performance incentivization: This phase includes not only contract law itself, but what I call the "micro-contract laws" of ratings systems and credit card chargebacks. Post-performance shaping of incentives can include reputation (especially via social network), the ability to chargeback (e.g., as bundled into typical credit card payments) or recourse to local legal systems (which is expensive, often prohibitively so, and so a major goal of smart contract design is to minimize dependence on legal incentives). Some post-performance phase activities—such as consumer ratings of businesses and credit ratings of consumers—can feed back

Figure 1: The deal cycle

into subsequent search and negotiation phases by informing future parties about (summaries of) the previous activities of their prospective counterparties. While the chargeback system is still widespread, ratings systems take advantage of the Internet. On the one hand, they shape incentives of the earlier performance phase; and on the other hand, they inform future counterparties during the search phase of the next deal cycle, further increasing the quality of the search-phase matchmaking.

Performing all the phases of a deal cycle well are important to a company's success. Among Internet and smartphone era innovations in the deal cycle, a small handful—including the ones that seem most mundane now—are those that most radically improved the deal cycle. See Table 1, "Internet and smartphone era contract phase innovation," and box on how Internet companies improved the deal cycle.

How Internet companies improved the deal cycle

Google and Facebook serve ads that match buyers with sellers in a nearly unlimited variety of businesses. In their own deal cycle, they use text search, auctions, and price setting algorithms to create a deal flow for targeted advertisements with minimum mental transaction costs.

eBay introduced ratings and text search on its platform to match buyers with long-tail (uncommon or antique) item sellers. Its biggest innovation—the auction, which determines the seller's price—facilitated the negotiation phase. However, eBay fell down on the job on the next phase, performance, in particular the payment leg, since credit card companies would usually deny merchant accounts to the typical individual or very small business long-tail sellers. Into this gap came PayPal. eBay's algorithms facilitate the search phase of contracting (buyer and seller finding each other), negotiation (auction) and in part help incentivize and verify the performance (payment and shipping) of the contractual relationship between the buyer and seller.

Amazon used ratings and text search to match buyers with long-tail products (either its own or selected sellers). In the performance phase, it innovated in warehousing and related logistics oriented around existing home delivery shipping services.

Lyft and Uber pioneered on a large scale the use of GPS to match riders and drivers. Uber's price-matching algorithm allows retail-style take-it-or-leave-it pricing, which eliminates the stress of haggling between strangers.

In total, trillions of dollars of economic value have been created by using the Internet to improve the way people make and perform deals with each other. Now that so much innovation has occurred in most of the other deal phases, we now have far fewer problems

finding what we want, if it exists. But most of the problems of negotiation, performance, post-performance remain. In particular, the performance phase has become a major cost center and bottleneck in most lines of business, even for payments and credit that could be quite simple and straightforward. It is in this area that on-public-blockchain smart contracts technology, with its highly secure integrity and globally seamless reach, provides the potential for tremendous innovation and disruption.

Cryptocurrencies, tokens, and blockchain-based smart contracts create value by allowing seamless money storage and transfer within and between arbitrarily diverse parts of the Internet: between Albania and Zimbabwe, between India and Indiana.

The most general idea of smart contracts is to apply technology intentionally to all these phases to enable more kinds of deals between increasingly far-flung and differently lawed and cultured people. In the performance phase, cryptocurrencies, tokens, and blockchain-based smart contracts create value by allowing seamless money storage and transfer within and between arbitrarily diverse parts of the Internet: between Albania and Zimbabwe, between India and Indiana. Second-layer or peripheral networks such as Lightning will allow us to continue using these currency and settlement systems as payment systems at large scale.

Social scalability

Social scalability is the ability to coordinate more and more kinds of people with less cost in economically productive pursuits.[7] Many institutions and careers in our society such as accounting, law, contracts, auctions, markets, and now smart contracts, have expanded or are expanding the social scalability of interactions and as a result the divisions of labor and knowledge. For example,

Table 1: Internet and smartphone era contract phase innovation

	Search	Negotiation	Performance	Post-performance
eBay	Text search	Auction	PayPal payment services and home delivery shipping	Ratings, chargebacks
Amazon	Text search	Take-it-or-leave-it	Encrypted credit card, warehousing and logistics around shipping and home delivery	Ratings, chargebacks
Lyft, Uber	GPS matching of riders and drivers	Take-it-or-leave-it, price-matching algorithm	Encrypted credit card; GPS verifies end of trip	Ratings, chargebacks, algorithmic "firing"
Google	Matching of user text searches and purchased keywords	Keyword auctions plus prices set by algorithm	User tracking (AdSense)	Click-through analysis

eBay's relationship-facilitating algorithms running between buyer and seller over the Internet make eBay far more socially scalable than its local auction forebears. Online markets generally such as NASDAQ tend to be socially scalable (i.e., many market participants to few NASDAQ employees), but that's because marketplaces are already socially scalable and online markets more socially scalable still than the more primordial and ethically natural forms of human economic relationship. That's why going from high transaction cost relationships that were bilateral monopolies or nearly so was a historical revolution.

In modern economies, we see social scalability in network effects, markets, and the sizes of firms. A positive network effect occurs when the addition of a new person to a network increases its value for the people already on the network. The result is that, not only does the value of the network grow as people join, but it grows faster than its population. Social networks, telephone networks, roads, and many other kinds of networks often exhibit this effect. Negative social scalability is also possible: if new entrants add more noise or unintelligible talk about their different interests than they contribute in value according to the interests of the previous members, the value of the network can grow more slowly than its population growth, or in some cases, even shrink as the population grows.

A positive network effect occurs when the addition of a new person to a network increases its value for the people already on the network.

Division of labor (and thus productivity), as Adam Smith theorized, expands with the extent of (or the number of individuals and organizations that access) a market. Friedrich Hayek described how markets increase the social scalability of communicating and acting on knowledge that is widely dispersed throughout the population. Prices communicate succinct but accurate summaries of personal knowledge that is not otherwise available.[8]

A major historical revolution was going from a typical maximum firm size of under a hundred to firm sizes often in the thousands or tens of thousands, as happened in the industrial revolution. Uber substitutes employment with algorithmically negotiated and verified gig work in an even more socially scalable way. It may be much more scalable than the industrial era type firm that constituted the industrial revolution in social scalability. Since many more people have much more of their labor expended in employment relationships than in spot market relationships, Uber and its similar successors in other logistics industries may be an even bigger deal than eBay and Amazon—and those have been pretty big deals.

Smart versus traditional contracts

In smart contracts, rules and conditions are analyzed by software code, and performances verified by sensors and software code or executed by sensor-guided effectors and software code. We call this *dry code*. By contrast, traditional contracts take the form of human

language—even if a specialized legal language—that is interpreted primarily by lawyers (and, usually by a very summarized indirectness, by non-lawyers), whether it takes the form of speech, written text, or text in digital form. We call this traditional legal language *wet code*.

On-chain smart contract behavior is often driven by off-chain data, and if so, needs rewind ability where typically two parties to a smart contract can choose to undo a mutually disagreeable result. *Multiple signature authority* (multisig) rewind, involving more than two parties, is also possible, if the sophisticated rewind code is programmed into the smart contract.

Traditional law and smart contracts work best in synergy, when the lawyers and the software engineers act as a team to secure the terms and conditions of a deal, instead of operating in silos.

Interpretation and complexity

Dry code can give harsh results in presence of unanticipated exceptions, results that may be hard to undo if more traditional legal arrangements have not been made in parallel outside the smart contract. Traditional law and smart contracts work best in synergy, when the lawyers and the software engineers act as a team to secure the terms and conditions of a deal, instead of operating in silos.

When it comes to anticipating possible behavior of the parties and relevant events in the environment they will be operating in—the main task of software engineering and legal drafting alike—more anticipated cases give rise to more conditions. In dry code, this increases the complexity and *attack surface* of the code (i.e., the number of points at which a programmer error could allow an attacker unexpected control over assets) and often creates more reliance on fallible external data. On the other hand, when a machine handles fewer conditions, there are usually more conditions that a human mediator or the traditional legal system must handle, which can greatly add to the costs and risks in the deal.

Wet versus dry code

While smart contracts are inspired by and can replace some of the functions of traditional contracts, they are largely complementary (see Table 2, "Comparison of wet and dry code," next page). Traditional law is manual, local, and often uncertain. Public blockchains, and the trust-minimized smart contracts running on them, are automated, global, and fairly relentless and predictable in their operations. Wet code (legalese that "runs" on the brains of lawyers) and dry code (software) can set out to accomplish the same general goals such as providing a secure foundation for a business relationship, and they share much of the process and logic that we can fruitfully reverse-engineer from law to software, which is the topic of the next section. But wet and dry code differ in important ways and complement each other. Lawyers need not worry about losing their jobs to robots because their work complements the work of machines running smart contracts. Smart contracts make possible what we haven't been able to do before.

Law is very flexible, corruptible, and involves judgment, whereas software is rigid and predictable.

Logic in traditional law derives from subjective minds and analogy. Software grounds on bits or data and Boolean logic. Law is very flexible, corruptible, and involves judgment, whereas software is rigid and predictable. Traditional law is highly evolved: in old case

law, lawyers can find many important conditions that computer programmers haven't yet contemplated or that are not easy to discern on a computer (e.g., the death of a user). But that law lives in jurisdictional silos: it is very nationalistic and local. Blockchains apply same rules everywhere on the globe. Lawyers drafting contracts or initiating lawsuits are very expensive; software once written typically is very cheap to run.

In sharp contrast to regtech, smart contracts do not try to slavishly mimick or simulate particular contemporary laws.

Reverse engineering highly evolved processes

The inspiration of smart contracts comes from the highly evolved nature of contract law and the mature nature of contracts that have in many industries evolved over dozens to hundreds of years. Smart contract practitioners can also learn lessons from property, secured transactions, bankruptcy, traditional governance in corporations and governments (especially in the area of *decentralized autonomous organizations* or DAOs), as well as from business processes such as accounting and internal controls. We can model on a blockchain rules and processes (such as those of negotiating and determining when and what kind of breach of contract has occurred and when collateral should be attached, freed, or seized) from these legal and business process areas. In addition to drawing knowledge and inspiration from these studies of law, smart contract practitioners can discover code and process by looking at traditional or historical contracts in their own fields of application. In sharp contrast to regtech (see below), smart contracts do not try to slavishly mimick or simulate particular contemporary laws. Instead, smart contract practitioners discover and take advantage of general and highly evolved patterns from either historical or contemporary legal or business processes. Smart contract drafters also reason in particular cases about incentives and program correctness.

Table 2: Comparison of wet and dry code

	Law	Software
Reasoning method	Subjective minds, analogy	Boolean logic, bits
Security	Contempt/imprisonment	Replication plus cryptography
Predictability	Flexible	Rigid
Maturity	Highly evolved/many cases	Larval/few experiences
Area	Jurisdictional silos	(On blockchain) independence from financial and political institutions and seamless operation across borders
Costs	Lawsuits: expensive	Extremely low

Relationship of smart contracts and traditional legal systems

From the point of view of traditional law, the most accurate way to view smart contracts is to treat them as security protocols that control the burden of lawsuit. For example, when repo man takes our car, the burden of lawsuit shifts from creditor to debtor. Asking a court to enforce a contract (or un-enforce security protocol) is expensive, and someone has to bear the cost of taking the case to court. A repo man places the burden on the person with the largest obligation, and thus most likely to breach rather than be victim of a breach of contract. With car starter interrupt devices or a full-fledged, trust-minimized smart contract, the auto-repo auto can perform a similar service.[9] By removing the burden of lawsuit from the creditor, lenders will have a lower cost of lending and be able to offer lower rates and provide more loans.

We can think of all sorts of security mechanisms like this. What if our security protocol and actual control corresponded to the rights and obligations people expect out of a relationship, so that our recourse to traditional law could be minimal? Ideal smart contracts make that correspondence happen. Trust-minimized smart contracts control assets jointly with authorized addresses (public key pairs controlled by users). As such, a smart contract, like a repo man, is a security protocol that controls the burden of lawsuit. If "possession is nine tenths of the law," then a cross-border, blockchain-based financial smart contract may be 99 percent of the law.

If "possession is nine tenths of the law," then a cross-border, blockchain-based financial smart contract may be 99 percent of the law.

Smart contracts in court?

A smart contract generally makes no attempt to be a legally binding contract. It is called a smart contract because it mimics or improves upon the effects of a traditional legal contract. It incentivizes performance using software control over money or other assets rather than the threat of litigation.

Smart contracts control assets, and, thus, the burden of lawsuit. In their best use cases, this control and the handling of common conditions also reduce the risks of lawsuit. If the smart contract does not handle sufficiently probable and important events, and as a result the risk of lawsuit remains unacceptably high—or if our lawyer convinces us that this is the case—then we will still need to think about what traditional contract has been formed and how courts might interpret it. In such cases, it is risky to expect either end users, lawyers, judges, or jurors to interpret dry code. Instead they will very likely look to the user interface and traditional legal text (wet code). In such cases, we may need to supplement our dry code with wet code designed for lawyers of the appropriate jurisdiction(s). This traditional step will usually cost us far more than using a smart contract, but our lawyer will nevertheless try to convince us that it is necessary.

Asking whether smart contracts need to be legally enforceable is akin to asking whether courts need to examine the guts of a vending machine to figure out what the parties intended.

Asking whether smart contracts need to be legally enforceable is akin to asking whether courts need to examine the guts of a vending machine to figure out what the parties intended. Such a question should be translated to: how much traditional contract do we still need, and what are the costs of enforcing the traditional contract? In some situations, such as many potential cross-border relationships between small businesses and individuals, the business relationship would have been impossible without a smart contract's greatly reducing the need for reliance on a traditional contract. On the flip side, if an event occurs that makes the smart contract harshly penalize one of the parties, using the traditional system to fix the problem will likely cost too much and the negatively effected party against whom the smart contract unexpectedly operated will be out of luck. But if the parties use the smart contract as a supplement to a traditional contract to form a business relationship where using the traditional legal system to enforce a traditional contract is viable—so that the "dry" smart contract and "wet" traditional contract are complementary, and each used to its best advantage—then the parties can likely use the traditional legal system to roll back an unexpectedly harsh result of the smart contract.

A number of smart contract programming options are available to deal with situations where a contract is breached, that is, a failure to perform is detected by the performance verification code. A common pattern will be the seizure of the breachor's on-chain collateral to pay liquidated damages.

If unexpected conditions occur, parties have a number of smart contract programming options available. Some kinds of transactions can be rewound by unanimous agreement of the two (or more) contracting parties. By original agreement of the contracting parties, rewinding could also occur with authorization of one party and some specific arbiters or trustees through the aforementioned multisig, a programming pattern requiring certain combinations of digital signatures to proceed.[10] Parties can also agree to undo a payment or pay compensation with an additional separate money transfer.

Smart contracts generally cannot take into account the legality of their subject matter in the various jurisdictions around the globe.

Smart contracts generally cannot take into account the legality of their subject matter in the various jurisdictions around the globe. Smart contracts are not so called *regulatory technology* ("regtech"), much less magical regtech imbued with knowledge of all the world's legal systems. Compliance of the subject matter of a smart contract with the wide variety of laws around the world is a matter of, on the one hand, ethical restraint on the part of the original writers of the smart contract and, on the other hand, off-chain processes and actions of various jurisdictions impinged by a smart contract after the smart contract has performed its task at assigning control over assets and thus managing the burden of lawsuit.

Regtech versus smart contracts

Those working in legal or accounting related areas in the United States have probably encountered compliance checkers such as those in TurboTax and for recent financial regulation schemes such as Dodd-Frank. What if our computers could check our contracts to see whether they were compliant? That's essentially what regtech

attempts to do: it checks the work of human beings. Some people have called these compliance checkers "smart contracts." That is not accurate; regtech is peripheral to the idea of smart contracts. Smart contracts are about creating and encoding rules into a security protocol (dry code) that incentivizes, via control of assets, performance of obligations that people have agreed to in a business relationship. Regtech is about checking whether human beings are complying with traditional laws and regulations (wet code), which is local and can only be approximated by simulations of wet code in certain areas (mostly financial laws). Smart contracts are global and persist on a blockchain. Regtech is only a simulation of how human language may be enforced by a court; smart contracts control assets. They are very different beasts.

Designing for security, integrity, and social scalability

Starting with the foundation: Security

Trusted third parties are security holes.

The main function of traditional contracts, incentivizing performance, is based on the threat of lawsuit; and the threat of lawsuit, and of traditional law generally, is ultimately based on a court's ability to securely seize and control people and assets. Business deals and markets, in turn, have traditionally been based on contracts and property enforced via this traditional legal means. Smart contracts, which involve a more directly rule-based control of assets, must start with securing themselves and the assets they control. (See Figure 2, "Society's protocol stack," next page.)

In its most general and important sense, security means the minimization of vulnerability—minimization of the need to trust in the behavior of people, whether insiders or outsiders, employees, or counterparties or third parties, rather than in mathematically demonstrated strong technological security.

A basic principle of blockchain and smart contract design is that trusted third parties are security holes. We might be tempted to assume a trusted third party in security protocol design, but we'd be cheating, like the "and here a miracle occurs" step of a mathematical "proof" in the classic Gary Larson cartoon. This trusted third party is a security hole that must be plugged by highly evolved institution or blockchain.

A feature—any desired feature of any kind—is more secure if it is less vulnerable to counterparties, third parties, or insiders. In the blockchain space, this property often goes under the somewhat exaggerated shorthand of "trustlessness." There is no such thing as a perfectly trustless system; computer science has demonstrated

the limits. No system is perfectly free of vulnerability to any and all combinations of attackers, whether outsiders or insiders. But the integrity properties of public blockchains come much closer to the ideal of trustlessness than any previous technology.

We can apply the principle of trust minimization to any desired feature of any product by asking ourselves, "Is that feature vulnerable to any arbitrarily malicious behavior of any insider or outsider?" Public blockchain is a technology that allows us to minimize such broadly defined vulnerability of the integrity of data or computations to a much greater degree than what we previously thought feasible.

The integrity properties of public blockchains come much closer to the ideal of trustlessness than any previous technology.

Because of its trust-minimized design, which takes great advantage of cryptography, decentralization, and replication, the core of Bitcoin[11] is probably the most secure and reliable financial system ever deployed. As of this writing, Bitcoin has a market capitalization of over $70 billion and has been running flawlessly for over eight years. But if we look at the periphery of the Bitcoin ecosystem, we still have centralized exchanges running under the control of individual computers. This makes them very insecure places to handle money, especially irreversible cash-like money like bitcoin.

Public blockchains enable trust-minimized smart contracts. We can think of blockchains as an army of robots with green eyeshades checking up on each others' work. We each have the option of running our own personal robot to check the work. Networked computers distribute copies to each other—computers are very good at making copies. Added to this liberally copied transaction history are cryptographic mechanisms for integrity. Cryptographically protected copies distributed across the globe create a very strong

Figure 2: Society's protocol stack

The "protocol layers" of society: Each layer requires and builds on the lower layers, and all require a secure foundation. As the scale of society grows larger—from village to city to province to nation to globe—each step requires the greater securing of all features desired to work at that larger social scale.

"append-only" transaction record. Using this transaction record and the globally decentralized computer network, we can run smart contracts in replicate with high security, integrity, and reliability.

Social scalability via trust-minimized computing

Social scalability is often confused with *computational scalability*, used for statistical modeling and predicting the performance of a computer system.[12] In many cases, they are parallel. For example, when we improve the computational scalability of the algorithm for parsing a social graph (who "follows" or "is friends with" whom), we can improve (and not reduce) the number of people that the social network (which has adopted that algorithm) can handle.

However, with Bitcoin, something like the opposite is true. Bitcoin achieves social scalability by its secure permissionlessness and integrity (censorship resistance) through such techniques as proof-of-work mining and broadcast replication that greatly increase its use of computational resources and reduce its computational scalability.[13] This also gives Bitcoin a very low transactions-per-second throughput compared to retail systems such as PayPal or Square. Thus Bitcoin alone is not, at scale, suitable as a direct retail payment network; it requires a second or peripheral layer. Such a layer requires use of smart contract techniques (described further below).

Whether we are in Albania or Zimbabwe, Africa or North America, the rules of the blockchain are the same everywhere.

Bitcoin's trade-off of computational scalability for security allows somebody in Zimbabwe to send money to somebody in Albania without vulnerability or need to pay quasi-monopoly rents to an intermediary. It thus greatly increases social scalability far beyond that of, for example, Fedwire and SWIFT, which are permissioned and accessible to only certain politically connected banks.

The Bitcoin blockchain is the most successful of the public and global blockchains. Whether we are in Albania or Zimbabwe, Africa or North America, the rules of the blockchain are the same everywhere.

Public blockchains provide platforms for *geosecure financial systems*, financial networks whose security, whether in parts or overall, does not depend on politics, jurisdiction, or law. Much as a gold atom is the same in Albania and Zimbabwe and has about the same exchange-weighted price, *geosecure financial rules*—smart contracts as vulnerability-minimized Dapps running on premium public blockchains such as Bitcoin—implement and execute the same rules in Guatemala as in Canada, in Estonia as in Venezuela, in China as in the United States—everywhere on the globe the Internet can reach. Bitcoin is therefore especially useful for cross-border financial applications such as remittances and capital preservation via global mobility.

By market capitalization, the second largest public blockchain is Ethereum, designed to be a geopolitically secure yet general-purpose computer.[14] As such it is attempting and will potentially succeed in implementing secure and reliable tokens and financial instruments and forming the core fiduciary code for smart contracts that seamlessly span the globe. This potential, and the huge global demand for tokens in initial coin offerings (ICOs) have given

Ethereum and the tokens it hosts collectively over $35 billion of market capitalization as of this writing.

However, Ethereum is very immature. Not only is it much younger than Bitcoin—at the same age, Bitcoin had less than one percent of Ethereum's current market capitalization—but plans are afoot to radically re-architect it, which would set the clock on its maturity back to the beginning. Bitcoin was already a year old when the first purchase was made—a pizza for 10,000 bitcoin—now that bitcoin is worth over $30 million, as of this writing. On top of its youth, Ethereum has a much larger attack surface than Bitcoin because of its Turing-complete smart contracts language and the relative abundance of applications enabled by high-level languages. More applications make Ethereum more useful, but also more vulnerable than the simpler, less abundant, and less functional smart contracts running on Bitcoin in the more limited Bitcoin Script language. For example, the $150 million attack against the DAO took advantage of flaws in the DAO code's use of recursion, a Turing-complete feature not found in Bitcoin Script. As public blockchains mature, Dapp writers will learn to program more carefully and to reuse code that has withstood the test of time, much as lawyers reuse mature templates in their contracts.

As public blockchains mature, Dapp writers will learn to program more carefully and to reuse code that has withstood the test of time, much as lawyers reuse mature templates in their contracts.

Software upgrades to blockchain pose security risks and involve labor-intensive politics. Software updates to Dapps themselves range from difficult (where we would have to unwind or otherwise handle a pre-existing state) to impossible (since most Dapps today are designed not to be upgradeable at all). Both traditional and smart contracts suffer from incompleteness: they cannot guarantee expected or mutually desirable operation in all circumstances, especially for more complicated or nondeterministic Dapps and DAOs.

Why identity scales so poorly

Identity is not very socially scalable; it is local, insecure, and labor-intensive. The very basis of identification in developed countries is the birth certificate, but more than one-third of the babies born in the world do not receive this foundation of identity. Without a birth certificate as a basis, their ability to prove their legal or other unique identity is very insecure.

Even in developed countries where people start out with birth certificates, convincing copies of these certificates, other identity documents, and other identifying information (such as mother's maiden name) are readily available to identity thieves, thanks to the insecurity of computers on the Internet and the ease of copying by computer. In contests between two or more entities, each of which is trying to convince a remote third party that each is "the real Alice," the actual Alice often cannot easily prove her case, since the remote party typically does not have access to more information about the actual Alice than the identity thief has. In situations where control over an important account or other resources is based on identity, Alice will spend extensive time trying to convince a stranger that she is Alice, with no guarantee of accurate results. Even then, the third party will often not be aware of the importance of, or have a sufficient incentive to protect, Alice's account or resource.

What should we look for in an "identity service"? (Hint: these services don't actually solve these problems, because they are too hard.) They should address the sock puppet issue (i.e., a Sybil attack) issue, that on the Internet "no one knows you are a dog," and it is easy (and not at all uncommon) for many different-looking accounts or computers to in fact be controlled by a single user, like a single hand controlling multiple finger-puppets. No one can securely know whether two distinct-looking names are actually controlled by two distinct people. One person can control dozens or (with bots) many thousands of sock puppets. In the world of face-to-face relationships, where our social instincts evolved, we use common sense and take for granted that one name and one face belong to one person. But this property does not apply on the Internet. If identity services assume it, claim that it is not a big problem, or assert that they have solved it in one fell swoop, they are leading themselves and their stakeholders toward failure at best and quite possibly disaster at worst.

There have been many attempts to establish global identity for more than the small "passport elite," for example, by using credit cards, addresses, phone numbers, and social network accounts for identity that is supposed to work the same and be suitable for almost any fiduciary purpose globally. These efforts have inevitably failed, especially in terms of identity theft and social engineering, since they are fundamentally based on information that identity thieves as well as identity providers can obtain. That's why an identity provider—be it a bank or a government agency—cannot easily, accurately, or cheaply distinguish a remote identity thief from the remote original identity.

In the world of face-to-face relationships, we use common sense and take for granted that one name and one face belong to one person. But this property does not apply on the Internet.

Identity theft attacks are already common with government documents. They are even easier when identity is based on social network accounts. None of the multitude of identity providers actually know any exclusive information about any of their users—they don't know anything that an identity thief could not also learn. That's why a remote third party has such difficulty distinguishing between the parties in an identity theft dispute. Meanwhile, the identity thief has plenty of time to execute the control that the stolen identities provide him, such as thefts of the valuable resources controlled by those identities, especially money. Identity theft thus poses a huge risk for any entity that controls valuable resources based on remote identities that were not thoroughly vetted and investigated—a labor-intensive process that cannot scale beyond a few business partners.

The insecurity of operating systems and network access control

Computer security techniques that do not socially scale are built into the very heart of today's computer operating systems such as MacOS, Windows, iOS, and Android. These security mechanisms, which trust a centralized "root" system administrator with all-powerful computer access, are based on an early Internet vision of an office "intranet" with "interpersonal computing." Back in the days when Bell Labs developed Unix, security was all about a bunch of

Blacklists yield many false positives and false negatives, since there is no highly reliable way to distinguish a "black hat" from a "white hat."

chummy friends in a company sharing files with each other, whether one at a time via email attachments or whole folders at a time via shared file systems. These computers were (and usually still are) programmed using languages that often give rise to security problems such as, for example C with its buffer overflow security holes. Billion-dollar security problems have resulted from trying to scale this security model, with its implicit trust assumptions, beyond the small office. In attempts to solve these problems, institutions have created crude (but vast in scale) blacklists, which exclude a wide variety of software and persons from participation in these networks (or associated institutions). Blacklists yield many false positives (people or software wrongly excluded) and false negatives (criminals and malicious software wrongly included), since there is no highly reliable way to distinguish a "black hat" from a "white hat."

Client/server architectures such as the Web are based on these single computers and computer clusters that also put full trust in a root administrator, who can control everything that happens on the server. This administrator, unknown to the end user, can read, alter, delete, or block any data on that computer at will. If somebody on the other end wants to ignore or falsify what the user has instructed the web server to do, no strong security is stopping the change, only fallible and expensive human institutions which often stop at national borders.

Paper-based controls, which had evolved for hundreds of years and had achieved a high degree of integrity, were naively implemented on computers and networks not designed with such controls in mind and without due care for security consequences. Actual security usually falls far short of the perceived controls.

To illustrate this mismatch between perceived controls and actual insecurity, let's look at the central bank of Bangladesh. To authorize large transactions, the bank used a computer authorization system that required six bank officials to place their hands on a touch screen in order. After going to all this trouble to try to distribute trust, the bank connected this authorization screen to a single computer—a single point of trust and failure. To compromise the security, all hackers needed was a single system administrator or malware with root access. In 2016, hackers successfully infected the trusted computer with malware and managed to steal $81 million via SWIFT messages. SWIFT approved these messages because they appeared to have been authorized by the central bank of Bangladesh.[15]

This illustrates a mismatch between the perceived security and actual security, due to the inherent insecurity of bottlenecking important transactions or input authorizing them through a single computer. The bank thought that requiring six different people to authorize a transaction on a single computer secured its system. The problem is that any system administrator or malware with root access on that computer could take any action at all, including any authorization controlled by that computer, regardless of however many users the centrally bottlenecked software required to authorize the transaction.

In physical security, the cost of attack is usually and on average less, and often much less, than the cost of defense. Combined with economies of scale in security, this cost difference has given rise to the nation-state with its general monopoly on the most effective means of force (i.e., the biggest and baddest weapons). Traditionally, based on the greater ease and effectiveness of offensive than defensive security, governments have enforced the rules that make commerce specifically, and civilization generally, possible.

In physical security, the cost of attack is less than the cost of defense. With the best digital security technologies such as cryptography, this equation is reversed: the cost of attack is generally far greater than the cost of defense.

However, this equation is reversed with the best digital security technologies, such as the use of cryptography to protect privacy and integrity: the cost of attack is generally far greater than the cost of defense. An encryption operation that takes a tenth of a second for a computer to perform can take billions of years (i.e., is effectively impossible) for a non-quantum computer to crack, or alternatively requires an expensive investigation and violent physical raid to obtain the private key—a process dubbed in the cryptography literature as a *rubber hose attack*. For this reason, governments have fallen behind in maintaining cybersecurity.

Governments necessarily have a far less useful role to play in cybersecurity than they do in physical security. Cybersecurity has thus become chiefly the responsibility of disparate organizations to ensure that the privacy and integrity of their own and their stakeholders' data remains intact. Encryption has become a powerful tool for individuals and organizations to preserve the on-wire privacy of communications (even as the ease of copying data has tended to erode privacy). For example, cryptographic hash functions, as embodied as Merkle trees in public blockchains, provide a powerful tool to securely preserve the integrity of our data.

Solution: Blockchain computers

Even though today's computers are not very trustworthy, they are so astronomically faster than humans at so many important tasks that we use them heavily anyway. We reap the tremendous benefits of computers and public networks at large costs of identity fraud and other increasingly disastrous attacks.

Cryptographic hash functions provide a powerful tool to preserve the integrity of our data securely.

Into this gap have stepped public blockchains.[16] Public blockchains cut through this Gordian knot of computer insecurity, providing a general solution to the many integrity problems that our unscalable computer security architectures have created. Cryptocurrencies like bitcoin and globe-spanning smart contract platforms like Ethereum would not be possible without the important computer science breakthrough, made by a person or group working under the pseudonym of Satoshi Nakamoto. This breakthrough allowed a widespread network of many computers to engage in secure and reliable consensus transactions and computations while putting minimal trust in the integrity of any one computer. As a result, for the first time in the history of computer code—albeit very slow and expensive computer code—the network does not have to rely on human institutions to work with secure integrity. This ability makes it censorship resistant and securely permissionless, allowing it to be used by anybody anywhere

on the Internet who has the computer and network resources to run a node and pay the transaction fees. No legal identity or other pre-existing relationship is required to participate. Nodes, users, and miners alike are free to come or go at will.

These blockchain computers run Dapps. A public blockchain securely settles transfers in its native cryptocurrency. It is also a virtual computer, a computer in a secure cloud, shared across many traditional computers and protected by cryptography and consensus technology. Each time the blockchain network processes a block, the cryptographic proof of work encases transactions in another layer of amber. Each layer is one block cycle and is also called a *confirmation*. On Bitcoin, during typical operations, six layers or confirmations render it astronomically improbable that a transaction can be reversed (other than by an extremely difficult long-chain attack).

On the Bitcoin blockchain, during typical operations, six confirmations render it astronomically improbable that a transaction can be reversed.

> Trust-minimized code means we can trust the code without trusting the owners of any particular remote computer. A smartphone user in India can use the blockchain to interact with a computer controlled by somebody in Indiana. They don't have to know or trust each other in any way, nor do they need to depend on the institutions of either's countries, for the underlying block chain computer to run its code securely and reliably. Regardless of where any of the computers or their owners are, the blockchain computer they share will execute as reliably and securely as consensus technology allows, up to the aforementioned limits.

The very high level of security of a good public blockchain comes at a great cost in performance, low response times, and substantial transaction fees. For settling the trade of high-value cryptocurrencies or tokens, and for some kinds of smart contracts, these performance losses are more than made up for in value by the far higher security and reliability a public blockchain provides. This allows blockchains to handle far larger and more sensitive kinds of money, act as a currency issuing and settlement layer.

Cryptocurrencies and blockchain-based tokens (the latter in terms of their titling and settlement only) are the safest kinds of assets in terms of security and reliability (but not volatility) that smart contracts can operate on.

We have looked at smart contracts or pieces of smart contracts that don't run on blockchains (vending machine, eBay, Uber), but, where computationally feasible, public blockchains are the premier way of running them—the most secure, reliable, and seamlessly global. Cryptocurrencies and blockchain-based tokens (the latter in terms of their titling and settlement only) are similarly the safest kinds of assets in terms of security and reliability (but not volatility) that smart contracts can operate on. Other assets require still largely ill-defined connections to traditional finance.

Hardware wallets

Hardware wallets are the most secure and reliable way for an individual or organization to control blockchain-based smart contracts. As discussed, normal computers are insecure and cannot properly handle substantial amounts of cryptocurrency, tokens,

or other private-key-based control over blockchain-based value. Two important features to look for in hardware wallets are (1) encrypted key backup, and (2) multiple signature authority (multisig) capabilities, which are ways for multiple people to share management of private-key-controlled assets such as cryptocurrencies and blockchain-based tokens.

Multiple signature authority capabilities are ways for multiple people to share management of private-key-controlled assets such as cryptocurrencies.

Dapp patterns

A full compendium of Dapp patterns is not possible, because the field is very immature and we are still learning about what patterns are useful (and what anti-patterns to avoid). There are more patterns emerging in Dapp programming than we can list here; this section is intended only to provide a brief taste. Among the simplest and most common kind of Dapp code is the kind used to track the ownership of, and settle the transfer of, on-chain tokens that can represent a variety of off-chain expectations. ERC20 is emerging as a standard in this area.[17]

In the area of performance verification and authorization, patterns include multiple signature authority (multisig), automated escrow (delivery-versus-payment or delivery-versus-delivery), manual performance verification escrow, and time-locked funds or tokens. Most performance verification is done off-chain. Useful for this and other purposes is the commitment, or posting, to the blockchain of secure codes summarizing off-chain data based on cryptographic hashes. We can think this as secure timestamping or as wrapping that data in secure amber, mathematically proving that the data existed at the time its code was posted. Another useful pattern is end-of-life cleanup, which can include final disbursement of funds and the "suicide" or deletion of the Dapp itself.

Among the simplest and most common kind of Dapp code is the kind used to track the ownership of, and settle the transfer of, on-chain tokens that can represent a variety of off-chain expectations.

In blockchain parlance, an *oracle* is commonly used but vague term that tends to refer to data inputs from off-chain, or the entity or entities responsible for or trusted with that data input. Unlike the code of the trust-minimized Dapp itself, the integrity of data input relies on the integrity of off-chain entities. Data input oracles are often used in on-chain smart and may input single-datum events (such as price quotes), data streams (such as a price ticker). Oracles can be bots that scan social networking services and trigger smart contracts conditional on social network user events or statistics (such as likes, follows, and retweets on Twitter). Oracles can rely on input from contractual parties or other users (such as third parties tasked to verify performance or the occurrence of conditions).

Other examples include measured values for parametric contracts conditioned on them (such as the books of a company that report its revenue for a contract that insures against losses in revenue), and reports of off-chain performances of obligations defined, and their performance verified and incentivized, in smart contracts. Data quality can be monitored and motivated by a variety of controls, traditional or novel. In some cases, rewind/undo code can be programmed into smart contracts so that they can recover from data errors based on later errata.

Since financial contracts tend to be very objectively defined, a wide variety of financial smart contracts is possible: bonds, forwards or futures, options (American, European, etc.), binary options, and contracts for difference (total swaps). I have been working for many years on ways to manage collateral in Dapps for trust-minimizing such contracts, and these form a major part of the products we are developing at my company, Global Financial Access.

A wide variety of useful patterns involves pooled money and often go under the somewhat exaggerated label of decentralized autonomous organization or DAO. DAOs are typically pools of money controlled by a group of people according to Dapp-defined rules. A simple version, one of the few smart contracts that doesn't require an input oracle, is a *threshold assurance contract*. This implements simple crowdfunding without an intermediary: people (possibly an unlimited number of them) put money into the pot until it reaches a threshold amount, at which point the money is distributed to a beneficiary address. If the threshold is not reached by a certain time, the money is refunded to the funders.

Patterns are also starting to emerge for careful programming of Dapps. Programming Dapps, and smart contracts generally, is more akin to the careful programming needed for medical equipment or an airplane than the looser programming typical for a web site or a game. This topic is however very technical, evolving rapidly, and beyond the scope of this paper.

Since financial contracts tend to be very objectively defined, a wide variety of financial smart contracts is possible: bonds, forwards or futures, options (American, European, etc.), binary options, and contracts for difference (total swaps).

Application areas

Peripheral (retail) financial network

Computer science dictates that public blockchains cannot scale to very large volumes without consuming prohibitively large amounts of network bandwidth that would cause full nodes to fail in their security function, compromising the security of the network. For this reason, a variety of second-layer or peripheral (i.e., off-blockchain) systems are under development, such as Lightning[18] for Bitcoin and Sprite[19] and Raiden[20] for Ethereum. These operate in smart contract fashion by posting collateral on the public blockchain, to enable a long series of off-blockchain transactions between settlement events, when the transactions are settled against the collateral. Similar ideas like Plasma[21] may speed up the execution of trust-minimized smart contracts generally.

Financial plumbing

Global Financial Access is developing collateralized versions of loans, bonds, futures, and options as composable financial objects that can automatically translate into smart contracts on a blockchain.

A similar concept has been the area of straight-through processing to automate back office, collateral management, and clearing and settlement operations on Wall Street.

Much of the geosecure financial plumbing of the coming decades will likely be based on cryptocurrencies, the core financial elements of public blockchains. Neither the issuance nor the control of cryptocurrency depends on legal or any other unique identity (i.e., central bank); instead, identity is cryptographic and the mechanisms are robust against Sybil (sock puppet) attacks from diverse-looking identities controlled by the same individual or organization. Cryptocurrencies are an important prerequisite to trust-minimized and worry-minimized smart contracts.

Cryptocurrencies are an important prerequisite to trust-minimized and worry-minimized smart contracts.

Worry-minimized commerce

Analysis of electronic commerce starts with the deal funnel. Our prospective customers enter the funnel when they find us, and come out the other end when each side has performed their part of the deal. In between, we lose progressively more and more customers as they move down the funnel: the funnel narrows. At each step, one can measure the drop-off or, conversely, the conversion rate.

Simplifying forms often greatly increases conversion rates; in one study, cutting the number of lines on a form in half increase conversions by a third. Drop-off or lack of conversion is caused by customer worries. The step causing our prospects the greatest worry is usually the step we most want them to complete: paying us.

The credit card form, which demands personal information from our prospect that can compromise both their security and ours (through identity theft), has the highest abandonment rate of any form in the funnel. As consumers, we know how tedious filling out forms is, on top of costing us time and our ability to enjoy that time. This tediousness is a decent proxy measure of the odds of our identity being stolen and of our privacy being violated. The fewer forms we fill out when interacting with the world's institutions, the fewer time wasters, worries, and risks in our lives.

The step causing our prospects the greatest worry is usually the step we most want them to complete: paying us.

Replacing advertisements and payments such as credit cards and credit card processors (PayPal, Square, Spotify) based on spoofable identities with payments that don't require identity, such as a second-layer retail bitcoin solution, can greatly reduce these worries, lowering the barriers and hesitations that currently prevent consumers from paying for our services.

> *Index of worry = number of lines on forms + number of repeated charges for content or services of variable value*

The ideal worry-free commerce is to "stick the coin into the machine" once, and then never have to pay again for an entire year. Reduce your customers' worries across the board: eliminate forms and eliminate recurring charges.[22]

App tokens

App tokens, as they are intended to be used, are digital tickets that provide access to apps. They can be looked at as "in-app currencies," but really they are more akin to coupons for free or deeply discounted goods or services, or tickets to a concert or ballgame. By advanced sale, the token seller can fund development of the service to be sold. The tokens are titled and their transfer settled by Dapps running on a blockchain; the most popular are the ERC20 standard Dapps running on Ethereum.[23]

App tokens, as they are intended to be used, are digital tickets that provide access to apps.

The number of tokens and services they buy is exploding; we will need tools to manage this complexity, especially to minimize the resulting mental transaction costs. Otherwise it will make far more sense for customers to use a few or one popular cryptocurrency for all their purchases rather than a plethora of tokens whose mutual value will be infeasible to mentally compare or to input numbers into a calculator or spreadsheet every time such a comparison is needed.[24]

Pools and DAOs

The canonical smart contract, such as the financial contracts discussed above, is an agreement negotiated and performed between two parties. However, there are many financial interactions where multiple parties put pools of money on blockchain or withdraw money from them. A completely automated and simple example is the assurance contracts. Everybody puts some money in a pool. If the pool reaches a threshold, money gets paid to beneficiary; if it fails to reach threshold by a certain time, then it gets paid back to donors. Either outcome is enforced by a smart contract on the blockchain. More broadly, pools can hold funds for groups (for example, acting as an organization's treasury). Pools that implement procedures for multiple parties to collectively decide what happens to funds, or the resulting groups of people using these tools, are often called *decentralized autonomous organization*s (DAOs).

Pools that implement procedures for multiple parties to collectively decide what happens to funds, or the resulting groups of people using these tools, are often called decentralized autonomous organizations (DAOs).

Insurance and parametric contracts

A parametric contract pays based on measurable data (e.g., actual loss in revenue from all causes following an insured event) rather than subjective and debatable criteria (e.g., estimates of lost revenue, particularly caused by that event). Given a quality data input oracle (e.g., to monitor the books for the insured for revenue losses), parametric contracts are far more amenable to smart contract automation (i.e., they need fewer or no manual steps) than traditional insurance, which depends on claims adjustment that is often subjective and based on qualitative data.

Logistics

Smart contracts can improve interoperability across supply chain, especially across borders and other trust boundaries. They can optimize supply chain logistics. Participants can track packages

via bar codes at physical portals such as ports and warehouses, with their entries and exits recorded and securely time stamped. Performances in time and space can be verified using computer networks and the GPS built in to smartphones and, increasingly, many other devices.

An entire supply chain, starting with the machines that make the parts to be tracked on the factory floor and ending with their final usage, may soon be able to connect through machine-to-machine interfaces. Performances in space, time, and conserving on the same may be measured and verified. Inventory management, transportation, distribution, accounting, payment processes, and more may be improved across entire and even multiple deal cycles.

With blockchain tech and smart contracts, drivers may be able to form their own decentralized organizations running open source software (on and off blockchains) and not have to go through Uber to reach riders.

Uber achieves social scalability via algorithmizing the contracting process: search (finding counterparties), negotiation, and verifying performance. As a result, Uber has grown in a few short years into a vast network of drivers and riders, where "[m]ore people earn a paycheck . . . from Uber than from any other private employer except for Walmart and McDonald's."[25]

The Uber ride-matching app subjects millions of riders and drivers to algorithmic control. Drivers choose when to work but, once they log on to the app, they only have ten to twenty seconds to respond to trip request. They do not learn the customer's destination until pick up. If drivers miss three trip requests in a row, they are logged out automatically for two minutes. Uber sends drivers a weekly report that includes confirmation rates and average customer rating. If our rating gets too low, Uber can essentially fire us as an independent contractor.

Now Uber is a trusted third party turning into a major profit center, because everybody has to go through Uber (or Lyft) to share rides. With blockchain tech and smart contracts, drivers may be able to form their own decentralized organizations running open source software (on and off blockchains) and not have to go through Uber to reach riders.

Algorithmic management

Algorithmic management involves verifying the performances of various objectively measurable subsets of jobs in the context of employment or similar relationships, and/or using computers to generate instructions to employees.[26]

Anybody who has obeyed a stoplight has been algorithmically managed.

Anybody who has obeyed a stoplight has been algorithmically managed. Computers decide the schedules based on sensors embedded in the pavement and watch the traffic flow. Algorithms decide when we can stop and when we can go. Most of us dutifully follow the computer. We are like little computer instructions.

In employment situations, where performances can be measured, computer bosses can keep track of far more performances and far more people in far more detail than human bosses and, as with stoplights, can communicate some simple rights and obligations to employees.

Consider the task of timely pickup and delivery of goods and passengers. For example, Uber measures timeliness getting to passengers, delivering them, and so forth. Such verification of performance is also common with the delivery of goods, offloading or onloading in ports and warehouses, and similar logistical tasks. In retail, measures of customer flow, cash and goods flow, and turnover give us metrics such as "shopper yield" for rewarding franchises and employees.

Algorithmic management is a close cousin or perhaps even a subset of smart contracts and can lead to redefining the contractual relationship itself, in some cases transforming employees into outsourced contractors. The financial fund management firm Bridgewater gathers and uses data on its employees to produce individualized "baseball cards." These cards contain various statistics, scores, and ratings. Many employees and contractors in many companies, doing logistical jobs in particular (such as Uber drivers and Amazon warehouse personnel), where performance can be measured in time and space, now are said to work for "algorithmic bosses."

We think of algorithmic management as involving employees, but these smart contract abilities can and often should change the contractual relationship. Sooner or later, the ability to more completely specify a task and verify its performance generally will turn an employment relationship into outsourcing to an independent contractor or small business.[27] So the ability to verify an employee's performance, whether we call it algorithmic management or a smart contract, enables a new relationship. It may no longer necessarily involve an employment contract, though some people may argue for an employment contract and the perquisites of employment.

Smart contract abilities can and often should change the contractual relationship.

Strategies and best practices for the organization

If your organization is contemplating the use of blockchains and smart contracts, then I recommend the following:

Examine your organization's business processes. Look for performances and conditions that are or might potentially be verified by sensors and computers such as payments and more sophisticated financial arrangements, logistical events, and tasks involving space and time, public arenas such as social networks with observable events, measures, statuses, and status changes. Look also for other potentially measurable conditions or tasks particular to your business.

- **Leverage your current systems for identifying your employees, users, customers, and other stakeholders.** Identity is local, insecure, and labor intensive. Blockchain and smart contracts technologies have not solved any of the most important weaknesses of identity systems. You and those you have reliably outsourced identity tasks to in the past are and will continue to be the experts on how to securely identify these people and the proper representatives of these organizations.

- **Make your business processes more globally scalable by reducing their dependence on legal identities.** Use fewer forms via more measurement. Depend less on local law enforcement and more on customized and global security agreed by you and your counterparties, as embodied as much as possible in trust-minimized, on-public-blockchain smart contracts.

- **Use hardware wallets to integrate smart contracts and on-chain assets into your internal controls.** You or your identity providers should develop business processes to map identities to the hardware wallets (private keys and corresponding blockchain addresses) they control. This function combines IT and identity information providers. Integrate hardware multisig controlling on-chain resources into your broader signature authority policies (finance/accounting departments).

- **Prefer public blockchains to private blockchains for running your smart contracts across borders.** Public blockchains depend on computer science rather than identity and local law enforcement for security and reliability properties. Mature public blockchains are securely permissionless and seamlessly global, which makes them more secure and reliable over a wide variety of local, regional, and global conditions.

- **Increase your organization's global reach by using smart contracts to lessen the dependence of your business processes on intermediaries, identity, and local laws.** Make your business processes, both internal and outward facing, less like a bureaucracy and more like a vending machine. Shrink forms and improve metrics and levers.

- **Design friendly smart contract user interfaces to minimize the worries of your customers, users, employees, and other stakeholders when they deal with your organization.** Reduced dependency on stakeholder-input information will reduce the tedium and the vulnerability to identity theft that organizations often inadvertently impose on their stakeholders. It will generally reduce your company's legal risks and will increase your organization's reputation as a friendlier, less worrisome entity to do business with.

- **Hire lawyers who know computer science and software engineers who know law, and ask them to collaborate on creating the secure underpinnings of your business relationships.** Your lawyers and software engineers should come to know the strengths and weaknesses of each other's approach, and your software engineers should obtain knowledge and inspiration from the long, highly evolved history of contract and secure transactions law and practice.

- **Outsource to small and remote entities with whom you couldn't do business before.** Once you can measure the most important of certain employee tasks, the next step—the most valuable step, but the one traditional scientific management has failed to recognize and take—is to restructure your contractual relationships. Employment relationships occur because contracts are too incomplete; they involve too many tasks that cannot be specified ahead of time, and so a relationship where a boss can determine on the fly what needs to be done and can instruct his employee to do so substitutes for a contract where the task is specifiable and its performance verifiable.[28]

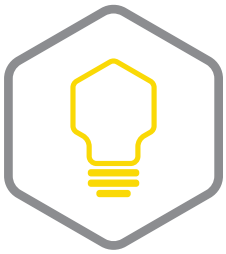
- **Convert employment contracts to independent contractor or other outsourcing contracts.** Once you can verify the performance of certain employee tasks, mostly by measurement than by subjective human judgment, you can make sufficiently complete contracts regarding those tasks so that you can outsource them.

In other words, once tasks become sufficiently measurable, this degree of ad-hoc management is no longer necessary, and organizations can outsource jobs among competing bidders. Smart contracts taking advantage of these measurements to incentivize performance will become an increasingly important aspect of this outsourcing, often enabling it in the first place, especially across national borders and other trust boundaries.

Conclusion

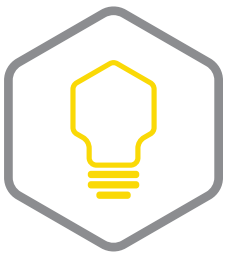
Smart contracts are new ways to formalize and secure our commercial relationships, which can be far more functional than their inanimate paper-based ancestors. Computers, networks, sensors, and blockchains in combination give us tremendous capabilities we've never had. Wet (traditional) legal code and smart contracts are very complementary. Going forward, we will have both wet and dry contracts for a very long time. This paper has outlined this revolution and described ways organizations can take advantage of it.

It is exciting for me to see that the basic vision of long ago of virtual vending machines, composable financial contracts, and the auto-repo auto has blossomed into a wide variety of areas based on blockchain technology and cryptocurrencies. Rapidly growing market capitalizations of public blockchains and the tokens they host have created a great deal of risk. Nevertheless, a number of short-term applications are worth the risk, and the long-term potential far outweighs these risks. We can look forward to more advances on these fronts and a growing industry of applications in upcoming years and decades.



About the author

Nick Szabo is a computer scientist known for his research in smart contracts (he coined the phrase), tokens, and cryptocurrency. He graduated from University of Washington with a degree in computer science and from George Washington University with a *juris doctor.* For his work in cryptocurrency and smart contracts he also holds an honorary professorship at the Universidad Francisco Marroquin. His company, Global Financial Access, is developing technology to facilitate the negotiation and composition of on-chain financial smart contracts by non-programmers.



About the Blockchain Research Institute

Co-founded in 2017 by Don and Alex Tapscott, the Blockchain Research Institute is a knowledge network organized to help realize the new promise of the digital economy. It builds on their yearlong investigation of distributed ledger technology, which culminated in the publication of their critically acclaimed book, *Blockchain Revolution* (Portfolio|Penguin).

Our syndicated research program, which is funded by major corporations and government agencies, aims to fill a large gap in the global understanding of blockchain technology and its strategic implications for business, government, and society.

Our global team of blockchain experts is dedicated to exploring, understanding, documenting, and informing leaders of the market opportunities and implementation challenges of this nascent technology.

Research areas include financial services, manufacturing, retail, energy and resources, technology, media, telecommunications, healthcare, and government as well as the management of organizations, the transformation of the corporation, and the regulation of innovation. We also explore blockchain's potential role in the Internet of Things, robotics and autonomous machines, artificial intelligence, and other emerging technologies.

Our findings are initially proprietary to our members and are ultimately released under a Creative Commons license to help achieve our mission. To find out more, please visit www.blockchainresearchinstitute.org.

Leadership team

Don Tapscott – Co-Founder and Executive Chairman
Alex Tapscott – Co-Founder
Joan Bigham – Managing Director
Kirsten Sandberg – Editor-in-Chief
Jane Ricciardelli – Director of Marketing
Hilary Carter – Director of Research
Jenna Pilgrim – Director of Business Development
Maryantonett Flumian – Director of Client Experience
Luke Bradley – Director of Communications

Notes

1. Nick Szabo, "Smart Contracts," *Hacker News,* 1994. www.tuicool.com/articles/U7veauY, accessed 27 Sept. 2017.
2. Nick Szabo, "Formalizing and Securing Relationships on Public Networks," *First Monday 2, no. 9*, 1 Sept. 1997. firstmonday.org/ojs/index.php/fm/article/view/548/469, accessed 27 Sept. 2017.
3. William Markham, "An Overview of Contract Law," Markham Law Firm, *Markham Law,* 2002. www.markhamlawfirm.com/law-articles/contract-lawyer-san-diego/, accessed 27 Sept. 2017.
4. Charles L. Knapp, Nathan M. Crystal, and Harry G. Prince, *Problems in Contract Law: Cases and Materials,* 5th ed. (New York: Aspen Publishers, 2003): 3.
5. Claude D. Rohwer**,** Gordon D. Schaber, and Anthony M. Skrocki**,** *Contracts in a Nutshell*, 5th ed. (St. Paul: West Publishing, 2000).
6. Sheila Baran, "Don't underestimate the boilerplate," *Smart Business Magazine,* 2 Feb. 2004. www.sbnonline.com/article/don-t-underestimate-the-boilerplate-legal-terms-you-need-to-know/, accessed 27 Sept. 2017.
7. Nick Szabo, "Money, blockchains, and social scalability," *Unenumerated,* www.blogger.com*,* 9 Feb. 2017. unenumerated.blogspot.com/2017/02/money-blockchains-and-social-scalability.html, accessed 27 Sept. 2017.
8. Friedrich A. Hayek, "The Use of Knowledge in Society," *American Economic Review* 35, no. 4 (Sept. 1945): 519-30. doi:10.1016/b978-0-7506-9749-1.50005-3.
9. Nick Szabo, "Smart Contracts," *Hacker News,* 1994. www.tuicool.com/articles/U7veauY; see also Nick Szabo, "A Formal Language for Analyzing Contracts (Preliminary Draft)," Satoshi Nakamoto Institute, 2002. nakamotoinstitute.org/contract-language, both accessed 27 Sept. 2017.
10. Morgan, Pamela. "Multi-Signature Accounts for Corporate Governance," *Empowered Law,* 25 May 2014. empoweredlaw.wordpress.com/2014/05/25/multi-signature-accounts-for-corporate-governance/, accessed 27 Sept. 2017.
11. Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Bitcoin Project, 31 Oct. 2008. bitcoin.org/bitcoin.pdf, accessed 31 Aug. 2017. Given the continued confusion between blockchains and cryptocurrencies, we've adopted uppercase B to denote the Bitcoin blockchain, its community of developers, and its features (proof of work, miners) and lowercase b to denote the cryptocurrency bitcoin associated with it. Likewise, Ethereum is the blockchain, ether is the token.
12. Neil J. Gunther, "A General Theory of Computational Scalability Based on Rational Functions," arXiv, 25 Aug. 2008. arxiv.org/abs/0808.1431, accessed 31 Aug. 2017.
13. For a good formal analysis of Bitcoin security, see Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," 23 June 2017. Cryptology ePrint Archive, eprint.iacr.org/2014/765.pdf, accessed 3 Oct. 2017.
14. Vitalik Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," *GitHub*, 10 Dec. 2014. github.com/ethereum/wiki/wiki/White-Paper, accessed 27 Sept. 2017.
15. Nick Szabo, "The Dawn of Trustworthy Computing," *Unenumerated,* www.blogger.com*,* 11 Dec. 2014. unenumerated.blogspot.com/2014/12/the-dawn-of-trustworthy-computing.html, accessed 27 Sept. 2017. SWIFT stands for the Society for Worldwide Interbank Financial Telecommunication (swift.com).

16. Elaine Ou, "The Rubber Hose Factor," *Elaine's Idle Mind, 29* July 2017. elaineou.com/2017/07/28/the-rubber-hose-factor/, accessed 27 Sept. 2017.
17. "ERC20 Token Standard," The Ethereum Wiki, 19 Sept. 2017. theethereum.wiki/w/index.php/ERC20_Token_Standard, accessed 27 Sept. 2017.
18. Joseph Poon and Thaddeus Dryja, "The Bitcoin Lightning Network," ver. 5, n.d. lightning.network/lightning-network-paper-DRAFT-0.5.pdf, accessed 3 Oct. 2017.
19. Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry, "Sprites: Payment Channels that Go Faster than Lightning," arXiv, 19 Feb. 2017. arxiv.org/pdf/1702.05812.pdf, accessed 3 Oct. 2017.
20. Heiko Hees et al, "Raiden Network: Raiden," *GitHub,* 19 Sept. 2015. github.com/raiden-network/raiden, accessed 3 Oct. 2017.
21. Joseph Poon and Vitalik Buterin, "Plasma: Scalable Autonomous Smart Contracts," www.plasma.io, 11 Aug. 2017. www.plasma.io/plasma.pdf, accessed 3 Oct. 2017.
22. Nick Szabo, "Estimating and minimizing consumer worry," *Unenumerated,* www.blogger.com, 8 Oct. 2014. unenumerated.blogspot.co.uk/2015/10/minimizing-consumer-worry.html, accessed 27 Sept. 2017.
23. "ERC20 Token Standard," The Ethereum Wiki, 19 Sept. 2017. theethereum.wiki/w/index.php/ERC20_Token_Standard, accessed 27 Sept. 2017.
24. Nick Szabo, "Scarce Objects," *Nakamoto Institute*, 2004. nakamotoinstitute.org/scarce-objects/, accessed 27 Sept. 2017.
25. Miguel Helft, "How Travis Kalanick is Builing the Ultimate Transportation Machine," *Forbes,* 14 Dec. 2016. www.forbes.com/sites/miguelhelft/2016/12/14/how-travis-kalanick-is-building-the-ultimate-transportation-machine/, accessed 27 Sept. 2017.
26. Elaine Ou, "Working for an Algorithm Might Be An Improvement," *Bloomberg,* 13 Jan. 2017. www.bloomberg.com/view/articles/2017-01-13/working-for-an-algorithm-might-be-an-improvement, accessed 27 Sept. 2017.
27. On incomplete contracts and employment versus outsourcing, see generally Oliver E. Williamson, *The Economic Institutions of Capitalism* (New York: The Free Press, 1985).
28. Williamson, *The Economic Institutions of Capitalism*.

blockchainresearchinstitute.org