DOI:10.1145/3132259

Article development led by acmqueue
queue.acm.org

The concept of cryptocurrencies is built from forgotten ideas in research literature.

BY ARVIND NARAYANAN AND JEREMY CLARK

Bitcoin's Academic Pedigree

IF YOU HAVE read about bitcoin in the press and have some familiarity with academic research in the field of cryptography, you might reasonably come away with the following impression: Several decades' worth of research on digital cash, beginning with David Chaum,[10,12] did not lead to commercial success because it required a centralized, bank-like server controlling the system, and no banks wanted to sign on. Along came bitcoin, a radically different proposal for a decentralized cryptocurrency that did not need the banks, and digital cash finally succeeded. Its inventor, the mysterious Satoshi Nakamoto, was an academic outsider, and bitcoin bears no resemblance to earlier academic proposals.

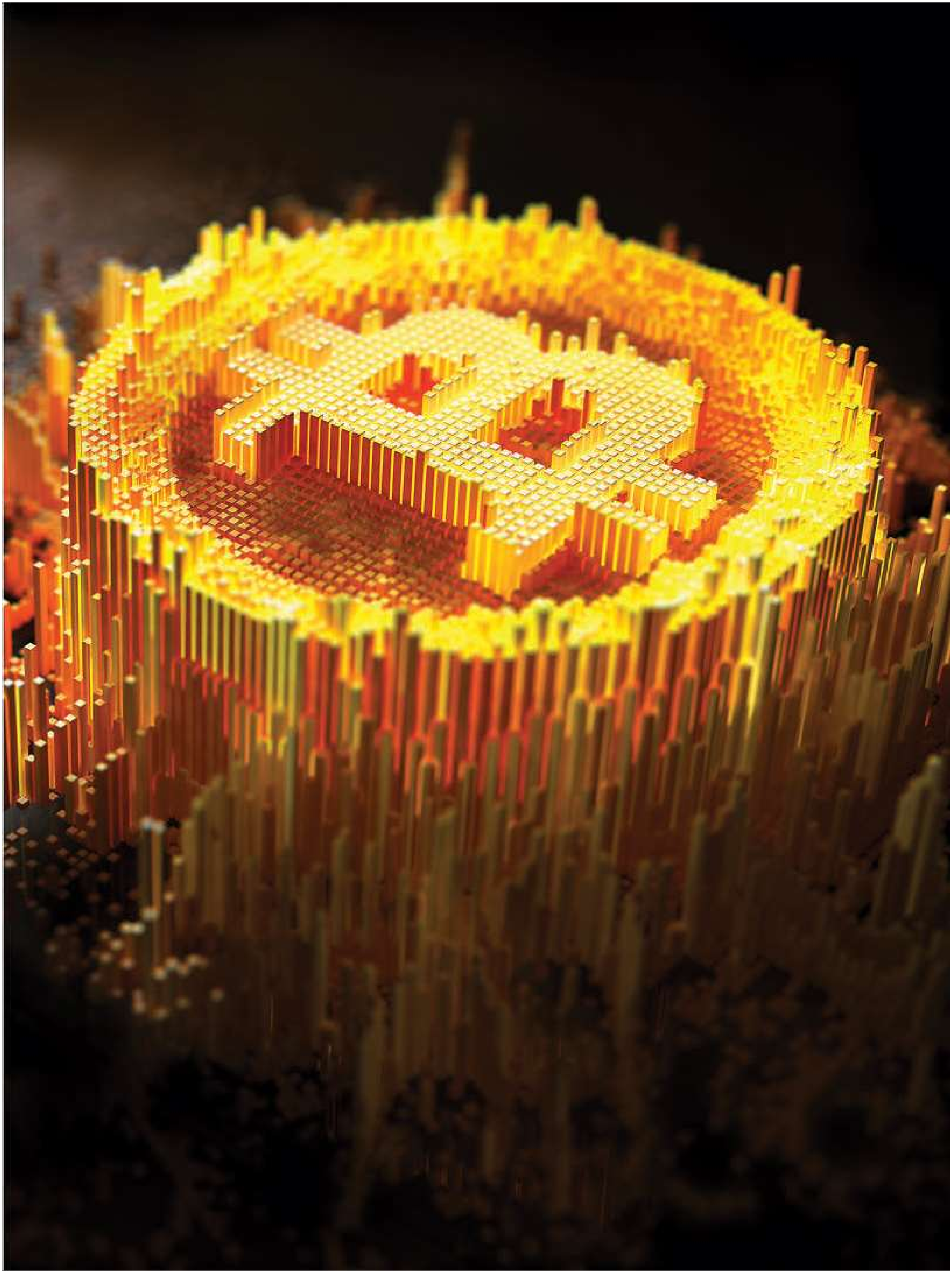
This article challenges that view by showing nearly all of the technical components of bitcoin originated in the academic literature of the 1980s and 1990s (see Figure 1). This is not to diminish Nakamoto's achievement but to point out he stood on the shoulders of giants. Indeed, by tracing the origins of the ideas in bitcoin, we can zero in on Nakamoto's true leap of insight—the specific, complex way in which the underlying components are put together. This helps explain why bitcoin took so long to be invented. Readers already familiar with how bitcoin works may gain a deeper understanding from this historical presentation. (For an introduction, see *Bitcoin and Cryptocurrency Technologies*.[36]) Bitcoin's intellectual history also serves as a case study demonstrating the relationships among academia, outside researchers, and practitioners, and offers lessons on how these groups can benefit from one another.

The Ledger

If you have a secure ledger, the process to leverage it into a digital payment system is straightforward. For example, if Alice sends Bob $100 by PayPal, then PayPal debits $100 from Alice's account and credits $100 to Bob's account. This is also roughly what happens in traditional banking, although the absence of a single ledger shared between banks complicates things.

This idea of a ledger is the starting point for understanding bitcoin. It is a place to record all transactions that happen in the system, and it is open to and trusted by all system participants. Bitcoin converts this system for recording payments into a currency. Whereas in banking, an account balance represents cash that can be demanded from the bank, what does a unit of bitcoin represent? For now, assume that what is being transacted holds value inherently.

How can you build a ledger for use in an environment like the Internet where participants may not trust each other? Let's start with the easy part: the choice of data structure. There are a



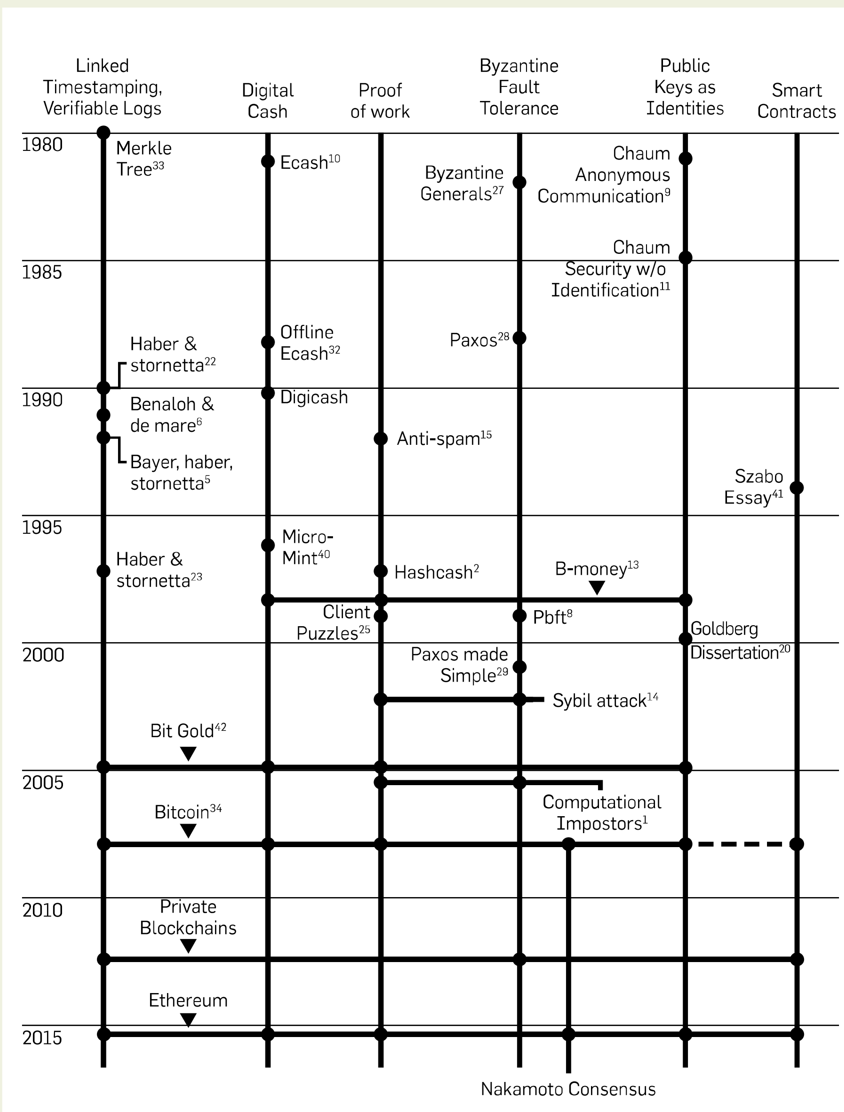
few desirable properties. The ledger should be immutable or, more precisely, *append only*: you should be able to add new transactions but not remove, modify, or reorder existing ones. There should also be a way to obtain a succinct *cryptographic digest* of the state of the ledger at any time. A digest is a short string that makes it possible to avoid storing the entire ledger, knowing that if the ledger were tampered with in any way, the resulting digest would change, and thus the tampering would be detected. The reason for these properties is that unlike a regular data structure that is stored on a single machine, the ledger is a *global* data structure collectively maintained by a mutually untrusting set of participants. This contrasts with another approach to decentralizing digital ledgers,[7,13,21] in which many participants maintain local ledgers and it is up to the user querying this set of ledgers to resolve any conflicts.

Linked timestamping. Bitcoin's ledger data structure is borrowed, with minimal modifications, from a series of papers by Stuart Haber and Scott Stornetta written between 1990 and 1997 (their 1991 paper had another co-author, Dave Bayer).[5,22,23] We know this because Nakamoto says so in his bitcoin white paper.[34] Haber and Stornetta's work addressed the problem of document timestamping—they aimed to build a "digital notary" service. For patents, business contracts, and other documents, one may want to establish that the document was created at a certain point in time, and no later. Their notion of document is quite general and could be any type of data. They do mention, in passing, financial transactions as a potential application, but it was not their focus.

In a simplified version of Haber and Stornetta's proposal, documents are constantly being created and broadcast. The creator of each document asserts a time of creation and signs the document, its timestamp, and the previously broadcast document. This previous document has signed its own predecessor, so the documents form a long chain with pointers backwards in time. An outside user cannot alter a timestamped message since it is signed by the creator, and the creator cannot alter the message without also altering the entire chain of messages that follows. Thus, if you are given a single item in the chain by a trusted source (for example, another user or a specialized timestamping service), the entire chain up to that point is locked in, immutable, and temporally ordered. Further, if you assume the system rejects documents with incorrect creation times, you can be reasonably assured that documents are at least as old as they claim to be. At any rate, bitcoin borrows only the data structure from Haber and Stornetta's work and reengineers its security properties with the addition of the proof-of-work scheme described later in this article.

In their follow-up papers, Haber and Stornetta introduced other ideas that make this data structure more effective and efficient (some of which were hinted at in their first paper). First, links between documents can be created using hashes rather than signatures; hashes are simpler and faster to compute. Such links are called hash pointers. Second, instead of threading documents individually—which might be inefficient if many documents are created at approximately the same time—they can be grouped into batches or blocks, with documents in each block having essentially the same timestamp. Third, within each block, documents can be linked together with a binary tree of hash pointers, called a Merkle tree, rather than a linear chain. Incidentally, Josh Benaloh and Michael

Figure 1. Chronology of key ideas found in bitcoin.



de Mare independently introduced all three of these ideas in 1991,[6] soon after Haber and Stornetta's first paper.

Merkle trees. Bitcoin uses essentially the data structure in Haber and Stornetta's 1991 and 1997 papers, shown in simplified form in Figure 2 (Nakamoto was presumably unaware of Benaloh and de Mare's work). Of course, in bitcoin, transactions take the place of documents. In each block's Merkle tree, the leaf nodes are transactions, and each internal node essentially consists of two pointers. This data structure has two important properties. First, the hash of the latest block acts as a digest. A change to any of the transactions (leaf nodes) will necessitate changes propagating all the way to the root of the block, and the roots of all following blocks. Thus, if you know the latest hash, you can download the rest of the ledger from an untrusted source and verify that it has not changed. A similar argument establishes another important property of the data structure—that is, someone can efficiently prove to you that a particular transaction is included in the ledger. This user would have to send you only a small number of nodes in that transaction's block (this is the point of the Merkle tree), as well as a small amount of information for every following block. The ability to efficiently prove inclusion of transactions is highly desirable for performance and scalability.

Merkle trees, by the way, are named for Ralph Merkle, a pioneer of asymmetric cryptography who proposed the idea in his 1980 paper.[33] His intended application was to produce a digest for a public directory of digital certificates. When a website, for example, presents you with a certificate, it could also present a short proof that the certificate appears in the global directory. You could efficiently verify the proof as long as you know the root hash of the Merkle tree of the certificates in the directory. This idea is ancient by cryptographic standards, but its power has been appreciated only of late. It is at the core of the recently implemented Certificate Transparency system.[30] A 2015 paper proposes CONIKS, which applies the idea to directories of public keys for end-to-end encrypted emails.[32] Efficient verification of parts of the global state is one of the key functionalities provided by the ledger in Ethereum, a new cryptocurrency.

Bitcoin may be the most well-known real-world instantiation of Haber and Stornetta's data structures, but it is not the first. At least two companies—Surety starting in the mid-1990s and Guardtime starting in 2007—offer document timestamping services. An interesting twist present in both of these services is an idea mentioned by Bayer, Haber, and Stornetta,[5] which is to publish Merkle roots periodically in a newspaper by taking out an ad. Figure 3 shows a Merkle root published by Guardtime.

Byzantine fault tolerance. Of course, the requirements for an Internet currency without a central authority are more stringent. A distributed ledger will inevitably have forks, which means that some nodes will think block A is the latest block, while other nodes will think it is block B. This could be because of an adversary trying to disrupt the ledger's operation or simply because of network latency, resulting in blocks occasionally being generated near-simultaneously by different nodes unaware of each other's blocks. Linked timestamping alone is not enough to resolve forks, as was shown by Mike Just in 1998.[26]

A different research field, fault-tolerant distributed computing, has studied this problem, where it goes by different names, including *state replication*. A solution to this problem is one that enables a set of nodes to apply the same state transitions in the same order—typically, the precise order does not matter, only that all nodes are consistent. For a digital currency, the state to be replicated is the set of balances, and transactions are state transitions. Early solutions, including Paxos, proposed by Turing Award winner Leslie Lamport in 1989,[28,29] consider state replication

Figure 2. The ledger data structure in linked timestamping.

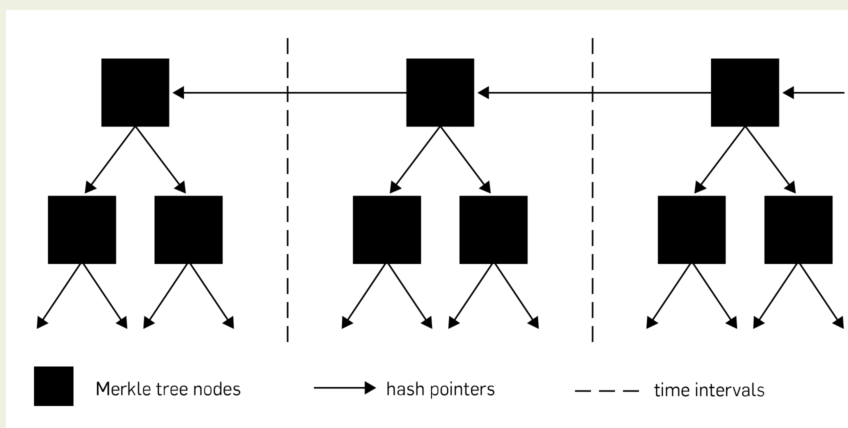


Figure 3. Guardtime Merkle root published in newspaper.

when communication channels are unreliable and when a minority of nodes may exhibit certain "realistic" faults, such as going offline forever or rebooting and sending outdated messages from when it first went offline. A prolific literature followed with more adverse settings and efficiency trade-offs.

A related line of work studied the situation where the network is mostly reliable (messages are delivered with bounded delay), but where the definition of "fault" was expanded to handle *any* deviation from the protocol. Such Byzantine faults include both naturally occurring faults as well as maliciously crafted behaviors. They were first studied in a paper also by Lamport, cowritten with Robert Shostak and Marshall Pease, as early as 1982.[27] Much later, in 1999, a landmark paper by Miguel Castro and Barbara Liskov introduced practical Byzantine fault tolerance (PBFT), which accommodated both Byzantine faults and an unreliable network.[8] Compared with linked timestamping, the fault-tolerance literature is enormous and includes hundreds of variants and optimizations of Paxos, PBFT, and other seminal protocols.

In his original white paper, Nakamoto does not cite this literature or use its language. He uses some concepts, referring to his protocol as a consensus mechanism and considering faults both in the form of attackers, as well as nodes joining and leaving the network. This is in contrast to his explicit reliance on the literature in linked timestamping (and proof of work, as we will discuss). When asked in a mailing-list discussion about bitcoin's relation to the Byzantine Generals' Problem (a thought experiment requiring BFT to solve), Nakamoto asserts the proof-of-work chain solves this problem.[35]

In the following years, other academics have studied Nakamoto consensus from the perspective of distributed systems. This is still a work in progress. Some show that bitcoin's properties are quite weak,[45] while others argue that the BFT perspective does not do justice to bitcoin's consistency properties.[41] Another approach is to define variants of well-studied properties and prove that bitcoin satisfies them.[19] Recently these definitions were substantially sharpened to provide a more standard consistency definition that holds under more realistic assumptions about message delivery.[37] All of this work, however, makes assumptions about "honest," that is, procotol-compliant, behavior among a subset of participants, whereas Nakamoto suggests that honest behavior need not be blindly assumed, because it is incentivized. A richer analysis of Nakamoto consensus accounting for the role of incentives does not fit cleanly into past models of fault-tolerant systems.

Virtually all fault-tolerant systems assume that a majority or supermajority of nodes in the system are both honest and reliable.

Proof Of Work

Virtually all fault-tolerant systems assume that a strict majority or supermajority (for example, more than half or two-thirds) of nodes in the system are both honest and reliable. In an open peer-to-peer network, there is no registration of nodes, and they freely join and leave. Thus an adversary can create enough *Sybils*, or sockpuppet nodes, to overcome the consensus guarantees of the system. The Sybil attack was formalized in 2002 by John Douceur,[14] who turned to a cryptographic construction called *proof of work* to mitigate it.

The origins. To understand proof of work, let's turn to its origins. The first proposal that would be called proof of work today was created in 1992 by Cynthia Dwork and Moni Naor.[15] Their goal was to deter spam. Note that spam, Sybil attacks, and denial of service are all roughly similar problems in which the adversary amplifies its influence in the network compared to regular users; proof of work is applicable as a defense against all three. In Dwork and Naor's design, email recipients would process only those email messages that were accompanied by proof that the sender had performed a moderate amount of computational work—hence, "proof of work." Computing the proof would take perhaps a few seconds on a regular computer. Thus, it would pose no difficulty for regular users, but a spammer wishing to send a million email messages would require several weeks, using equivalent hardware.

Note that the proof-of-work *instance* (also called a puzzle) must be specific to the email, as well as to the recipient. Otherwise, a spammer would be able to send multiple messages to the same recipient (or the same message to multiple recipients) for the cost of one message to one recipient. The second crucial property is that it should pose minimal computational burden on the recipient; puzzle solutions should be trivial to verify, regardless of how difficult they are to compute. Additionally, Dwork and Naor considered functions with a *trapdoor*, a secret known to a central authority that would allow the authority to solve the puzzles without doing the work. One possible application of a trapdoor would be for the authority to approve posting to mailing lists without incurring a cost. Dwork and Naor's proposal consisted of three candidate puzzles meeting their properties, and it kicked off a whole research field, to which we will return.

Hashcash. A very similar idea called hashcash was independently invented in 1997 by Adam Back, a postdoctoral researcher at the time who was part of the cypherpunk community. Cypherpunks were activists who opposed the power of governments and centralized institutions, and sought to create social and political change through cryptography. Back was practically oriented: he released hashcash first as software,[2] and five years later in 2002 released an Internet draft (a standardization document) and a paper.[4]

Hashcash is much simpler than Dwork and Naor's idea: it has no trapdoor and no central authority, and it uses only hash functions instead of digital signatures. It is based on a simple principle: a hash function behaves as a random function for some practical purposes, which means the only way to find an input that hashes to a particular output is to try various inputs until one produces the desired output. Further, the only way to find an input that hashes into an arbitrary *set* of outputs is again to try hashing different inputs one by one. So, if I challenged you to find an input whose (binary) hash value begins with 10 zeros, you would have to try numerous inputs, and you would find that each output had a $1/2^{10}$ chance of beginning with 10 zeros, which means that you would have to try on the order of 2^{10} inputs, or approximately 1,000 hash computations.

Sybil-Resistant Networks

In his paper on Sybil attacks, John Douceur proposed that all nodes participating in a BFT protocol be required to solve hashcash puzzles. If a node were masquerading as *N* nodes, it would be unable to solve *N* puzzles in time, and the fake identities would be purged. Karma, an early peer-to-peer digital cash system, uses a hashcash-like puzzle to rate-limit nodes joining the Karma network and receiving credits for file sharing.[44] A malicious node, however, could still obtain a moderate advantage over an honest node that claimed only a single identity. A follow-up paper in 2005[1] suggested honest nodes should instead mimic the behavior of malicious nodes and claim as many virtual identities as they computationally can afford to claim. With these virtual identities executing a BFT protocol, the assumption, "At most a fraction *f* of nodes are faulty" can be replaced with the assumption "The fraction of total computational power controlled by faulty nodes is at most *f*." Thus, it is no longer necessary to validate identities, and open peer-to-peer networks can run a BFT protocol. Bitcoin uses exactly this idea. But Nakamoto asks: What motivates nodes to perform computationally expensive proof of work? The answer requires a further leap: digital currency.

Smart Contracts

A smart contract takes the idea of putting *data* in a secure ledger and extends it to *computation*. In other words, it is a consensus protocol for the correct execution of a publicly specified program. Users can invoke functions in these smart-contract programs, subject to any restrictions specified by the program, and the function code is executed in tandem by the miners. Users can trust the output without having to redo the computation and can write their own programs to act on the output of other programs. Smart contracts are especially powerful when combined with a cryptocurrency platform, because the programs in question can handle money—own it, transfer it, destroy it, and, in some cases, even print it.

Bitcoin implements a restrictive programming language for smart contracts. A "standard" transaction (that is, one that moves currency from one address to another) is specified as a short script in this language. Ethereum offers a more permissive and powerful language.

The idea of smart contracts was proposed by Nick Szabo in 1994[42] and so named because he saw them as analogs of legal contracts, but with automated enforcement. (This view has been critiqued by Levy[31] and Felten.[16]) Presciently, Szabo presented smart contracts as extensions of digital-cash protocols and recognized that Byzantine agreement and digital signatures (among others) could be used as building blocks. The success of cryptocurrencies has made smart contracts practical, and research on the topic has bloomed as well. For example, programming languages researchers have adapted their methods and tools to automatically discover bugs in smart contracts and to write verifiably correct ones.

Permissioned Blockchains

While this article has emphasized that private or permissioned blockchains omit most of bitcoin's innovations, this is not meant to diminish the interesting work happening in this space. A permissioned blockchain places restrictions on who can join the network, write transactions, or mine blocks. In particular, if miners are restricted to a list of trustworthy participants, the proof of work can be dropped in favor of a more traditional BFT approach. Thus, much of the research is a rebirth of BFT that asks questions such as: Can we use hash trees to simplify consensus algorithms? What if the network can fail only in certain ways?

Further, there are important considerations around identity and public-key infrastructure, access control, and confidentiality of the data stored on the blockchain. These issues largely do not arise in public blockchain settings, nor are they studied in the traditional BFT literature.

Finally, there is also the engineering work of scaling blockchains for high throughput and adapting them to various applications such as supply-chain management and financial technology.

As the name suggests, in hashcash Back viewed proof of work as a form of cash. On his webpage he positioned it as an alternative to David Chaum's DigiCash, which was a system that issued untraceable digital cash from a bank to a user.[3] He even made compromises to the technical design to make it appear more cashlike. Later, Back made comments suggesting that bitcoin was a straightforward extension of hashcash. Hashcash is simply not cash, however, because it has no protection against double spending. Hashcash tokens cannot be exchanged among peers.

Meanwhile, in the academic scene, researchers found many applications for proof of work besides spam, such as preventing denial-of-service attacks,[25] ensuring the integrity of Web analytics,[17] and rate-limiting password guessing online.[38] Incidentally, the term *proof of work* was coined only in 1999 in a paper by Markus Jakobsson and Ari Juels, which also includes a nice survey of the work up until that point.[24] It is worth noting that these researchers seem to have been unaware of hashcash but independently started to converge on hash-based proof of work, which was introduced in papers by Eran Gabber et al.[18] and by Juels and Brainard.[25] (Many of the terms used throughout this paragraph did not become standard terminology until long after the papers in question were published.)

Proof of work and digital cash: A catch–22. You may know that proof of work did not succeed in its original application as an anti-spam measure. One possible reason is the dramatic difference in the puzzle-solving speed of different devices. That means spammers will be able to make a small investment in custom hardware to increase their spam rate by orders of magnitude. In economics, the natural response to an asymmetry in the cost of production is trade—that is, a market for proof-of-work solutions. But this presents a catch-22, because that would require a working digital currency. Indeed, the lack of such a currency is a major part of the motivation for proof of work in the first place. One crude solution to this problem is to declare puzzle solutions to *be* cash, as hashcash tries to do.

More coherent approaches to treating puzzle solutions as cash are found in two essays that preceded bitcoin, describing ideas called b-money[13] and bit gold[43] respectively. These proposals offer timestamping services that sign off on the creation (through proof of work) of money, and once money is created, they sign off on transfers. If disagreement about the ledger occurs among the servers or nodes, however, there isn't a clear way to resolve it. Letting the majority decide seems to be implicit in both authors' writings, but because of the Sybil problem, these mechanisms are not very secure, unless there is a gatekeeper who controls entry into the network or Sybil resistance is itself achieved with proof of work.

Putting It All Together

Understanding all these predecessors that contain pieces of bitcoin's design leads to an appreciation of the true genius of Nakamoto's innovation. In bitcoin, for the first time, puzzle solutions don't constitute cash by themselves. Instead, they are merely used to secure the ledger. Solving proof of work is performed by specialized entities called *miners* (although Nakamoto underestimated just how specialized mining would become).

Miners are constantly in a race with each other to find the next puzzle solution; each miner solves a slightly different variant of the puzzle so that the chance of success is proportional to the fraction of global mining power that the miner controls. A miner who solves a puzzle gets to contribute the next batch, or block, of transactions to the ledger, which is based on linked timestamping. In exchange for the service of maintaining the ledger, a miner who contributes a block is rewarded with newly minted units of the currency. With high likelihood, if a miner contributes an invalid transaction or block, it will be rejected by the majority of other miners who contribute the following blocks, and this will also invalidate the block reward for the bad block. In this way, because of the monetary incentives, miners ensure each other's compliance with the protocol.

Bitcoin neatly avoids the double-spending problem plaguing proof-of-work-as-cash schemes because it eschews puzzle solutions themselves having value. In fact, puzzle solutions are *twice* decoupled from economic value: the amount of work required to produce a block is a floating parameter (proportional to the global mining power), and further, the number of bitcoins issued per block is not fixed either. The block reward (which is how new bitcoins are minted) is set to halve every four years (in 2017, the reward is 12.5 bitcoins/block, down from 50 bitcoins/block). Bitcoin incorporates an additional reward scheme—namely, senders of transactions paying miners for the service of including the transaction in their blocks. It is expected the market will determine transaction fees and miners' rewards.

Nakamoto's genius, then, was not any of the individual components of bitcoin, but rather the intricate way in which they fit together to breathe life into the system. The timestamping and Byzantine agreement researchers didn't hit upon the idea of incentivizing nodes to be honest, nor, until 2005, of using proof of work to do away with identities. Conversely, the authors of hashcash, b-money, and bit gold did not incorporate the idea of a consensus algorithm to prevent double spending. In bitcoin, a secure ledger is necessary to prevent double spending and thus ensure that the currency has value. A valuable currency is necessary to reward miners. In turn, strength of mining power is necessary to secure the ledger. Without it, an adversary could amass more than 50% of the global mining power and thereby be able to generate blocks faster than the rest of the network, double-spend transactions, and effectively rewrite history, overrunning the system. Thus, bitcoin is bootstrapped, with a circular dependence among these three components. Nakamoto's challenge was not just the design, but also convincing the initial community of users and miners to take a leap together into the unknown—back when a pizza cost 10,000 bitcoins and the network's mining power was less than a trillionth of what it is today.

Public keys as identities. This article began with the understanding that a secure ledger makes creating digital currency straightforward. Let's revisit this claim. When Alice wishes to pay Bob, she broadcasts the transaction to all bitcoin nodes. A transaction is simply a string: a statement encoding Alice's wish to pay Bob some value, signed by her. The eventual inclusion of this signed statement into the ledger by miners is what makes the transaction real. Note that this doesn't require Bob's participation in any way. But let's focus on what's *not* in the transaction: conspicuously absent are Alice and Bob's identities; instead, the transaction contains only their respective public keys. This is an important concept in bitcoin: public keys are the only kinds of identities in the system. Transactions transfer value from and to public keys, which are called *addresses*.

In order to "speak for" an identity, you must know the corresponding secret key. You can create a new identity at any time by generating a new key pair, with no central authority or registry. You do not need to obtain a user name or inform others that you have picked a particular name. This is the notion of decentralized identity management. Bitcoin does not specify how Alice tells Bob what her pseudonym is—that is external to the system.

Although radically different from most other payment systems today, these ideas are quite old, dating back to David Chaum, the father of digital cash. In fact, Chaum also made seminal contributions to anonymity networks, and it is in this context that he invented this idea. In his 1981 paper, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,"[9] he states: "A digital 'pseudonym' is a public key used to verify signatures made by the anonymous holder of the corresponding private key."

Now, having message recipients be known only by a public key presents an obvious problem: there is no way to route the message to the right computer. This leads to a massive inefficiency in Chaum's proposal, which can be traded off against the level of anonymity but not eliminated. Bitcoin is similarly exceedingly inefficient compared with centralized payment systems: the ledger containing every transaction is maintained by every node in the system. Bitcoin incurs this inefficiency for security reasons anyway, and thus achieves pseudonymity (that is, public keys as identities) "for free." Chaum took these ideas much further in a 1985 paper,[11] where he presents a vision of privacy-preserving e-commerce based on pervasive pseudonyms, as well as "blind signatures," the key technical idea behind his digital cash.

The term *blockchain* has no standard technical definition but is a loose umbrella term used by various parties to refer to systems that bear varying levels of resemblance to bitcoin and its ledger.

The public-keys-as-identities idea is also seen in b-money and bit gold, the two precursor essays to bitcoin discussed earlier. However, much of the work that built on Chaum's foundation, as well as Chaum's own later work on ecash, moved away from this idea. The cypherpunks were keenly interested in privacy-preserving communication and commerce, and they embraced pseudonyms, which they called *nyms*. But to them, nyms were not mere cryptographic identities (that is, public keys), but rather, usually email addresses that were linked to public keys. Similarly, Ian Goldberg's dissertation, which became the basis of much future work on anonymous communication, recognizes Chaum's idea but suggests that nyms should be human-memorable nicknames with certificates to bind them.[20] Thus Bitcoin proved to be the most successful instantiation of Chaum's idea.

The Blockchain

So far, this article has not addressed the blockchain, which, if you believe the hype, is bitcoin's main invention. It might come as a surprise to you that Nakamoto doesn't mention that term at all. In fact, the term *blockchain* has no standard technical definition but is a loose umbrella term used by various parties to refer to systems that bear varying levels of resemblance to bitcoin and its ledger.

Discussing example applications that benefit from a blockchain will help clarify the different uses of the term. First, consider a database backend for transactions among a consortium of banks, where transactions are netted at the end of each day and accounts are settled by the central bank. Such a system has a small number of well-identified parties, so Nakamoto consensus would be overkill. An on-blockchain currency is not needed either, as the accounts are denominated in traditional currency. Linked timestamping, on the other hand, would clearly be useful, at least to ensure a consistent global ordering of transactions in the face of network latency. State replication would also be useful: a bank would know that its local copy of the data is identical to what the central bank will use to settle its account. This frees banks from the expensive reconciliation process they must currently perform.

Second, consider an asset-management application such as a registry of documents that tracks ownership of financial securities, or real estate, or any other asset. Using a blockchain would increase interoperability and decrease barriers to entry. We want a secure, global registry of documents, and ideally one that allows public participation. This is essentially what the timestamping services of the 1990s and 2000s sought to provide. Public blockchains offer a particularly effective way to achieve this today (the data itself may be stored off-chain, with only the metadata stored on-chain). Other applications also benefit from a timestamping or "public bulletin board" abstraction, most notably electronic voting.

Let's build on the asset-management example. Suppose you want to execute trades of assets via the blockchain, and not merely record them there. This is possible if the asset is issued digitally on the blockchain itself, and if the blockchain supports smart contracts. In this instance, smart contracts solve the "fair exchange" problem of ensuring that payment is made if and only if the asset is transferred. More generally, smart contracts can encode complex business logic, provided that all necessary input data (assets, their prices, and so on) are represented on the blockchain.

This mapping of blockchain properties to applications allows us not only to appreciate their potential, but also to inject a much-needed dose of skepticism. First, many proposed applications of blockchains, especially in banking, don't use Nakamoto consensus. Rather, they use the ledger data structure and Byzantine agreement, which, as shown, date to the 1990s. This belies the claim that blockchains are a new and revolutionary technology. Instead, the buzz around blockchains has helped banks initiate collective action to deploy shared-ledger technology, like the parable of "stone soup." Bitcoin has also served as a highly visible proof of concept that the decentralized ledger works, and the Bitcoin Core project has provided a convenient code base that can be adapted as necessary.

Blockchains are frequently presented as more secure than traditional registries—a misleading claim.

Second, blockchains are frequently presented as more secure than traditional registries—a misleading claim. To see why, the overall stability of the system or platform must be separated from endpoint security—that is, the security of users and devices. True, the systemic risk of blockchains may be lower than that of many centralized institutions, but the endpoint-security risk of blockchains is far worse than the corresponding risk of traditional institutions. Blockchain transactions are near-instant, irreversible, and, in public blockchains, anonymous by design. With a blockchain-based stock registry, if a user (or broker or agent) loses control of his or her private keys—which takes nothing more than losing a phone or getting malware on a computer—the user loses his or her assets. The extraordinary history of bitcoin hacks, thefts, and scams does not inspire much confidence—according to one estimate, at least 6% of bitcoins in circulation have been stolen at least once.[39]

Concluding Lessons

The history described here offers rich (and complementary) lessons for practitioners and academics. Practitioners should be skeptical of claims of revolutionary technology. As shown here, most of the ideas in bitcoin that have generated excitement in the enterprise, such as distributed ledgers and Byzantine agreement, actually date back 20 years or more. Recognize that your problem may not require any breakthroughs—there may be long-forgotten solutions in research papers.

Academia seems to have the opposite problem, at least in this instance: a resistance to radical, extrinsic ideas. The bitcoin white paper, despite the pedigree of many of its ideas, was more novel than most academic research. Moreover, Nakamoto did not care for academic peer review and did not fully connect it to its history. As a result, academics essentially ignored bitcoin for several years. Many academic communities informally argued that Bitcoin could not work, based on theoretical models or experiences with past systems, despite the fact it *was* working in practice.

We have seen repeatedly that ideas in the research literature can be gradually forgotten or lie unappreciated, especially if they are ahead of their time, even in popular areas of research. Both practitioners and academics would do well to revisit old ideas to glean insights for present systems. Bitcoin was unusual and successful not because it was on the cutting edge of research on any of its components, but because it combined old ideas from many previously unrelated fields. This is not easy to do, as it requires bridging disparate terminology, assumptions, and so on, but it is a valuable blueprint for innovation.

Practitioners would benefit from being able to identify overhyped technology. Some indicators of hype: difficulty identifying the technical innovation; difficulty pinning down the meaning of supposedly technical terms, because of companies eager to attach their own products to the bandwagon; difficulty identifying the problem that is being solved; and finally, claims of technology solving social problems or creating economic/political upheaval.

In contrast, academia has difficulty selling its inventions. For example, it's unfortunate that the original proof-of-work researchers get no credit for bitcoin, possibly because the work was not well known outside academic circles. Activities such as releasing code and working with practitioners are not adequately rewarded in academia. In fact, the original branch of the academic proof-of-work literature continues today without acknowledging the existence of bitcoin! Engaging with the real world not only helps get credit, but will also reduce reinvention and is a source of fresh ideas.

Acknowledgments. Thanks to Adam Back, Andrew Miller, Edward Felten, Harry Kalodner, Ian Goldberg, Ian Grigg, Joseph Bonneau, Malte Möser, Mike Just, Neha Narula, Steven Goldfeder, and Stuart Haber for their valuable feedback.

References

1. Aspnes, J., et al. Exposing computationally challenged Byzantine imposters. Yale University Department of Computer Science, 2005; http://cs.yale.edu/publications/techreports/tr1332.pdf.
2. Back, A. A partial hash collision based postage scheme, 1997; http://www.hashcash.org/papers/announce.txt.
3. Back, A. Hash cash, 2001; https://web.archive.org/web/20010614013848/http://cypherspace.org/hashcash/.
4. Back, A. Hashcash—a denial of service counter measure, 2002; http://www.hashcash.org/papers/hashcash.pdf.
5. Bayer, D., Haber, S. and Stornetta, W.S. Improving the efficiency and reliability of digital time-stamping. In *Proceedings of Sequences* (1991); https://link.springer.com/chapter/10.1007/978-1-4613-9323-8_24.
6. Benaloh, J., de Mare, M. Efficient broadcast timestamping, 1991; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.9199.
7. Boyle, T.F. GLT and GLR: Component architecture for general ledgers, 1997;https://linas.org/mirrors/www.gldialtone.com/2001.07.14/GLT-GLR.htm.
8. Castro, M. and Liskov, B. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (1999); http://pmg.csail.mit.edu/papers/osdi99.pdf.
9. Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24*, 2 (Feb. 1981), 84–90; https://dl.acm.org/citation.cfm?id=358563.
10. Chaum, D. Blind signatures for untraceable payments. *Advances in Cryptology*, 1983, 199-203.
11. Chaum, D. Security without identification: transaction systems to make Big Brother obsolete. *Commun ACM 28*, 10 (Oct. 1985), 1030–1044; https://dl.acm.org/citation.cfm?id=4373.
12. Chaum, D., et al. Untraceable electronic cash. *Advances in Cryptology*, 1988, 319–327; https://dl.acm.org/citation.cfm?id=88969.
13. Dai, W. 1998; http://www.weidai.com/bmoney.txt.
14. Douceur, J.R. The Sybil attack, 2002; https://dl.acm.org/citation.cfm?id=687813.
15. Dwork, C. and Naor, M. Pricing via processing or combatting junk mail, 1992; https://dl.acm.org/citation.cfm?id=705669.
16. Felten, E. Smart contracts: neither smart nor contracts? Freedom to tinker, 2017; https://freedom-to-tinker.com/2017/02/20/smart-contracts-neither-smart-not-contracts/.
17. Franklin, M.K. and Malkhi, D. Auditable metering and lightweight security, 1997; http://www.hashcash.org/papers/auditable-metering.pdf.
18. Gabber, E., et al. Curbing junk e-mail via secure classiffication, 1998; http://www.hashcash.org/papers/secure-classification.pdf.
19. Garay, J A., et al. The bitcoin backbone protocol: analysis and applications. *Advances in Cryptology*, 2015, 281–310; https://eprint.iacr.org/2014/765.pdf.
20. Goldberg, I. A pseudonymous communications infrastructure for the Internet. Ph.D. dissertation. University of California Berkeley, 2000; http://moria.freehaven.net/anonbib/cache/ian-thesis.pdf.
21. Grigg, I. Triple entry accounting, 2005; http://iang.org/papers/triple_entry.html.
22. Haber, S. and Stornetta, W.S. How to timestamp a digital document. *J. Cryptology 3*, 2 (1991), 99–111; https://link.springer.com/chapter/10.1007/3-540-38424-3_32.
23. Haber, S. and Stornetta, W.S. Secure names for bit-strings. In *Proceedings of the 4th ACM Conference on Computer and Communications Security,* 1997, 28–35; http://dl.acm.org/citation.cfm?id=266430.
24. Jakobsson, M. and Juels, A. Proofs of work and bread pudding protocols, 1999; http://www.hashcash.org/papers/bread-pudding.pdf.
25. Juels, A. and Brainard, J. Client puzzles: a cryptographic countermeasure against connection completion attacks. In *Proceedings of Networks and Distributed Security Systems*, 1999, 151–165; https://www.isoc.org/isoc/conferences/ndss/99/proceedings/papers/juels.pdf.
26. Just, M. Some timestamping protocol failures, 1998; http://www.isoc.org/isoc/conferences/ndss/98/just.pdf.
27. Lamport, L., et al. The Byzantine Generals Problem. *ACM Trans. Programming Languages and Systems 4*, 3 (1982), 382–401; https://dl.acm.org/citation.cfm?id=357176 .
28. Lamport, L. The part-time parliament. Digital Equipment Corp., 1989; https://computerarchive.org/files/mirror/www.bitsavers.org/pdf/dec/tech_reports/SRC-RR-49.pdf.
29. Lamport, L. Paxos made simple, 2001; http://lamport.azurewebsites.net/pubs/paxos-simple.pdf.
30. Laurie, B. Certificate transparency. *acmqueue 12*, 1 (2014); https://queue.acm.org/detail.cfm?id=2668154.
31. Levy, K.E.C. Book-smart, not street-smart: blockchain-based smart contracts and the social workings of law. *Engaging Science, Technology, and Society 3* (2017), 1–15; http://estsjournal.org/article/view/107.
32. Melara, M., et al. CONIKS: Bringing key transparency to end users. In *Proceedings of the 24th Usenix Security Symposium*, 2015; https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-melara.pdf.
33. Merkle, R.C. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy,* 1980; http://www.merkle.com/papers/Protocols.pdf.
34. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system, 2008; https://bitcoin.org/bitcoin.pdf.
35. Nakamoto, S. Re: Bitcoin P2P e-cash paper, 2008; http://satoshi.nakamotoinstitute.org/emails/cryptography/11/.
36. Narayanan, A., et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, 2016; http://bitcoinbook.cs.princeton.edu/.
37. Pass, R., et al. Analysis of the blockchain protocol in asynchronous networks. In *Proceedings of the 2017 International Conference on the Theory and Applications of Cryptographic Techniques*; https://link.springer.com/chapter/10.1007/978-3-319-56614-6_22.
38. Pinkas, B. and Sander, T. Securing passwords against dictionary attacks. In *Proceedings of the Ninth ACM Conference on Computer and Communications Security*, 2002, 161–170; https://dl.acm.org/citation.cfm?id=586133.
39. Reuters. Mind your wallet: Why the underworld loves bitcoin, 2014; http://www.cnbc.com/2014/03/14/mind-your-wallet-why-the-underworld-loves-bitcoin.html.
40. Rivest, R.L. and Shamir, A. PayWord and MicroMint: Two simple micropayment schemes. In *Proceedings of the 1996 International Workshop on Security Protocols.*
41. Sirer, E.G. Bitcoin guarantees strong, not eventual, consistency. *Hacking, Distributed*, 2016; http://hackingdistributed.com/2016/03/01/bitcoin-guarantees-strong-not-eventual-consistency/.
42. Szabo, N. Smart contracts, 1994; http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.
43. Szabo, N. Bit gold. Unenumerated, 2008; https://unenumerated.blogspot.com/2005/12/bit-gold.html.
44. Vishnumurthy, S., Chandrakumar, S. and Sirer, E.G. Karma: A secure economuc framework for P2P resource sharing. In *Proceedings of the Workshop on the Economics of Peer-to-Peer Systems* (Berkeley, CA, June 2003).
45. Wattenhofer, R. *The Science of the Blockchain*. Inverted Forest Publishing, 2016.

Arvind Narayanan is an assistant professor of computer science at Princeton University. He leads the Princeton Web Transparency and Accountability Project to uncover how companies collect and use our personal information. Twitter @random_walker.

Jeremy Clark is an assistant professor at the Concordia Institute for Information Systems Engineering. He has also worked with several municipalities on voting technology and testified to the Canadian Senate on bitcoin. Twitter @PulpSpy.

DOI:10.1145/3132269

Article development led by acmqueue
queue.acm.org

Cardboard surrounds and protects stuff as it crosses boundaries.

BY PAT HELLAND

XML and JSON Are Like Cardboard

IN TODAY'S WORLD, cardboard is an ever-important part of life. Given the major investment of resources and money, you might question whether it's worth it. It turns out the efficiencies and savings from cardboard outstrip the costs to manufacture and later recycle it.

Semi-structured representations of data are not the cheapest format. There is typically a lot of extra stuff like angle brackets contained in it. JSON, XML, and other semi-structured representations allow for wonderful flexibility and dynamic interpretation. The efficiencies and savings gained from flexibility more than make up for the overhead.

Cardboard surrounds and protects stuff as it moves across boundaries. No one uses cardboard to move parts around within a factory. Instead, they use custom-designed containers that are specially purposed for the parts being produced. Cardboard is used to protect the stuff as it leaves the factory.

JSON and XML are used to protect data when it moves across trust boundaries. Semi-structured data wraps a single message or a single item in a key-value store in a way that allows for flexibility and extensibility. Inside an application, relational data is more tightly controlled and well formed. Evolving your relational data inside the trust and management boundary of an app is tractable.

SQL and relational data are easier and better for processing data *within a trust boundary*. XML and JSON are more flexible and dynamic as they capture the information and its metadata. This makes it easy and flexible to squirt data across trust boundaries.

Self-defining and self-identifying. Cardboard is usually self-describing. Your new TV has printing on the outside of the box telling you what's inside the box. As you move your old TV to your new home, you write "TV" on the outside of the moving box.

JSON and XML are usually self-describing. This can be done by referencing a schema or by examining the attributes expressed within the document/file itself.

Generic vs. custom. The last time we moved to a new home, I bought a bunch of boxes of varying sizes, tape, wrapping paper, and padding, and a bunch of marking pens. Like most other folks in the throes of packing and moving, we worked hard to describe the contents of every box we filled, but we occasionally messed up and omitted some items from the list as everything went into a box. Most things fit well into one of the standard boxes, although some of our household items involved really creative uses of cardboard, tape, and padding as we worked to protect our stuff.

Manufactured items frequently have custom-made cardboard protection. My wife loves the vacuum cleaners from one particular manufacturer. Indeed, the shape, form, and workings of the vacuums can be fun and surprising. To me, half the fun is disassembling the cardboard protection used inside the cardboard box. There are dozens of spe-