Federal Reserve
Bank of Boston®

February 2019

Beyond Theory: Getting Practical With Blockchain

Boston Fed Learns by Doing With Blockchain Technology

Contents

The views expressed in this report are those of the authors and do not necessarily represent positions of the Federal Reserve Bank of Boston or the Federal Reserve System.

Foreword

The Federal Reserve Bank of Boston is pleased to present "Beyond Theory: Getting Practical With Blockchain." This report is a resource for business professionals and technologists looking to experiment with blockchain, also known as distributed ledger technology.

Blockchain has the potential to impact many industries—including financial services—so the Boston Fed sought to understand its foundational technology with first-hand research. We wanted practical experience, the kind only trial and error can bring.

The report goes beyond the basics of distributed ledger technology and tells the story of the Boston Fed's journey to understand how a blockchain platform could help us perform specific functions within our operations. Our journey involved developing two use cases for learning purposes only—we don't intend to move them into production. This report describes the use cases, the technology we employed, and the insights we gained.

We would like to acknowledge the contributions of the Boston Fed's innovative and cross-functional blockchain team, which made this white paper possible. Its expertise in app development, information security, payments technology, and other underlying advanced infrastructures was invaluable. We would also like to offer a special thanks to our Federal Reserve colleagues—notably in Minneapolis—for their collaboration and insights.

We hope that readers will gain a better understanding of blockchain from this report and perhaps be inspired to develop their own proof of concept or use case. This technology holds tremendous promise for anyone willing to test its limits.

James S. Cunha
Senior Vice President
Federal Reserve Bank of Boston

Introduction

Soap factories in 1960s Liverpool, England, had a problem.[1] The process that turned liquid detergent into a powder required a nozzle that shot out boiling hot liquid at great pressure. Unfortunately, these nozzles frequently clogged, were inefficient, and produced granules of different sizes.

Companies hired mathematicians and physicists to try to design the perfect nozzle, but they relied too heavily on theoretical knowledge, and their efforts failed. Then, the companies turned to biologists committed to a different approach, one modeled after nature's process of trial and error. These experts designed a series of nozzles with slight variations. Next, they tested them, modifying them until, after 45 generations, they had an efficient nozzle. Problem solved.

These companies had to think differently, innovate, and try something new. They learned by doing, which is something the Liverpool soap factories and the Federal Reserve Bank of Boston have in common. Our team at the Boston Fed wanted to develop an understanding of blockchain and distributed ledger technology, but we knew theories weren't enough. We wanted practical experience—the kind only trial and error can bring. The following pages detail the strategies, insights, technical architecture, and ongoing efforts of the Boston Fed to build a proof of concept on blockchain platforms.

What Readers Can Expect From This Paper

This paper details why we executed our proofs of concepts, the successes and challenges we experienced, and what we learned along the way. We hope it is beneficial to those seeking to experiment on their own or simply understand the technology at a deeper level.

We assume the reader has a basic working knowledge of blockchain platforms and some key variations, such as public (permissionless) versus private (permissioned) models. However, we include an appendix to help build that foundational knowledge. Additionally, while some use the more general term "distributed ledger technology," we use the term "blockchain" throughout this paper for simplicity—even though not every distributed ledger technology platform utilizes chained blocks. For the purpose of this paper, "blockchain" refers to a distributed, decentralized database that employs cryptographic algorithms and a consensus mechanism as a platform to execute activities. All these activities are recorded in a single, shared ledger.

Additionally, we provide a timeline of key events, including those relevant to the use cases we describe. The capabilities of these platforms evolve rapidly, and our experiments and decision-making were based on information available at the time. Readers of this paper may find the technology has advanced past the functionality we describe. However, we expect the paper's insights will remain relevant beyond those changes.

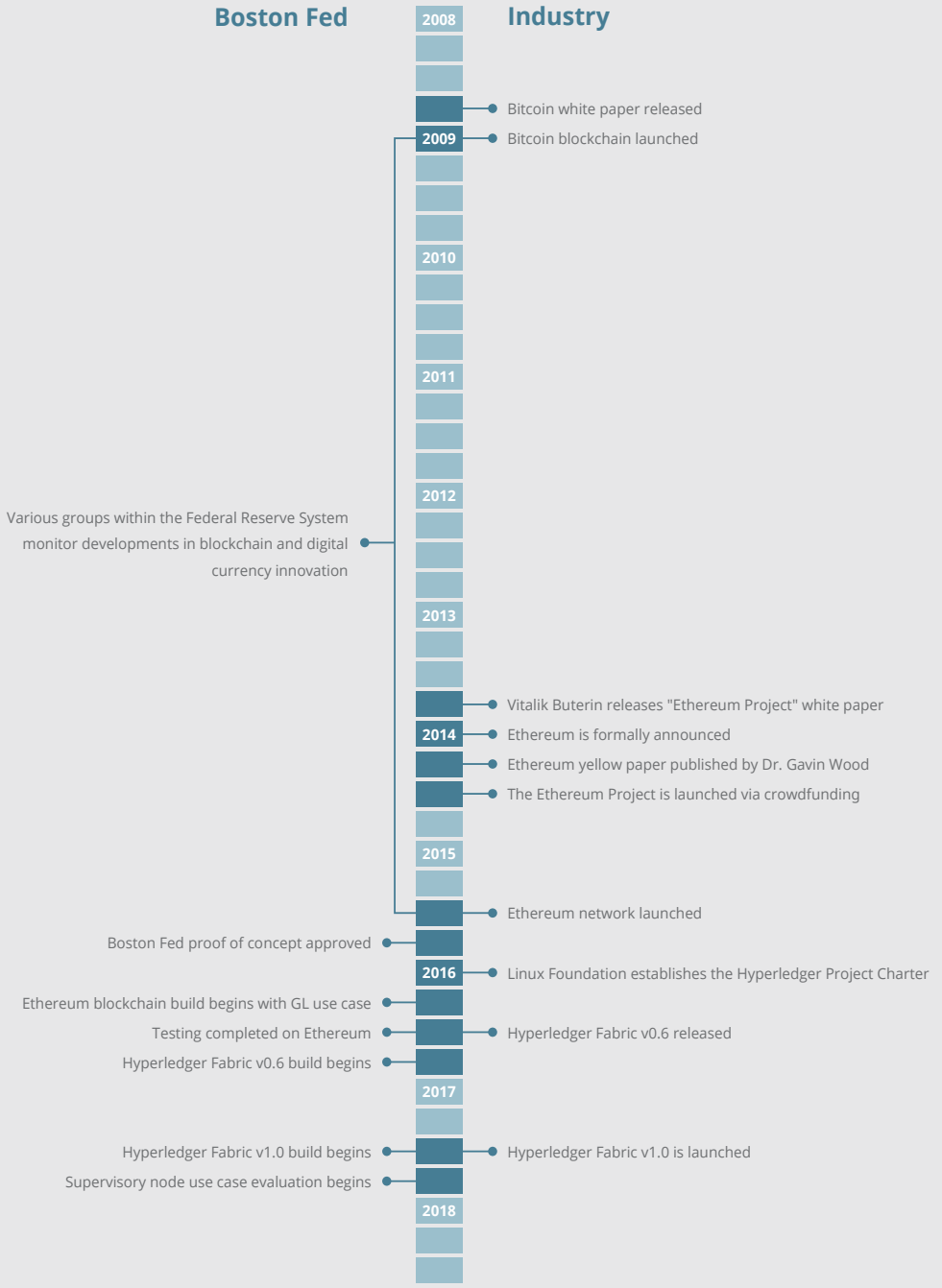
Following this introduction, we will detail two use cases: The first is a technological exploration of how a blockchain platform performs specific functions within a general accounting system. The second represents an evolution in our thinking and details the concept of a supervisory node within a distributed, decentralized network.

As we describe the use cases, we provide:

- technical descriptions
- functional diagrams
- goals for each use case
- insights
- lessons learned

Finally, we offer key takeaways and conclusions that business and technology leaders can apply as they explore the options and potential opportunities of blockchain platforms.

Comparison Between Platform Emergence and Project Timeline



Sources:
https://blog.ethereum.org/2016/02/09/cut-and-try-building-a-dream/
https://www.hyperledger.org/announcements/2016/02/09/linux-foundations-hyperledger-project-announces-30-founding-members-and-code-proposals-to-advance-blockchain-technology
https://github.com/hyperledger/fabric#releases

Building an Ethereum Blockchain Proof of Concept

The Boston Fed has been monitoring the evolution of digital currencies and blockchain technologies for years, but it wasn't until late 2015 that we decided to deepen our working knowledge through experimentation. It was around this time that senior Federal Reserve officials were beginning to publicly speak about the need to study blockchain's potential applications. In June 2016, former Federal Reserve Board Chair Janet Yellen encouraged central banks from all over the world to learn about Bitcoin, blockchain, and distributed ledger technologies.[2] That October, Federal Reserve Governor Lael Brainard addressed the International Institute of Finance and affirmed the Fed's interest in understanding the developments, opportunities, and risks associated with innovations like blockchain.[3]

Also in 2016, the research firm Gartner placed blockchain near the peak of "inflated expectations" on its Hype Cycle graphic, which predicts when certain technologies will be adopted by the mainstream.[4] At that time, some analysts thought the technology was five to 10 years from widespread use. At the Boston Fed, we wanted to cut through the noise and share what we learned with others. From the start, we had this quote from the playwright George Bernard Shaw in mind: "The one lesson that comes out of all our theorizing and experimenting is that there is only one really scientific progressive method; and that is the method of trial and error."

Choosing Ethereum

From the early days of Bitcoin, our team had been watching developments in blockchain, but it wasn't until the Ethereum platform—which expanded on the concept of smart contracts—was introduced that we started devising a proof of concept. Ethereum co-founder Vitalik Buterin described the platform as "a blockchain with built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications."[5] Ethereum was managed through a community of open source developers, and it appeared to have good documentation and support from existing developers when we decided to use it for our early experiments. From the start, the team had some clear goals and milestones in mind, including:

- learn about the technology
- demonstrate viability
- test design features critical to central bank services
- validate a lean development effort with minimal investment

Priming the Pump

Before work could begin on a blockchain proof of concept, the team needed to prime the proverbial pump by procuring the basics. That included downloading and running Ethereum, as well as interacting with Ethereum via its software, Geth's Command Line Interface (CLI), and exposing Geth's JSON-RPC interface. These tools allowed a simple transaction to be executed.

While the setup—a relatively straightforward command-line console interface—wasn't difficult by technologist standards, our team soon recognized the platform was not simple enough for mainstream use. It would be one of many learnings to come.

The Use Case

Once the API was established and the team had a working knowledge of how inputs and outputs were created in the protocol, defining a specific use case became the next priority. An opportunity presented itself almost immediately.

The Boston Fed manages the development and operation of the Federal Reserve's accounting platform, which processes an average of 250,000 transactions per day. These transactions represent account-level settlement entries for items cleared by the Fed's retail payment applications (e.g., ACH, check, cash), and summary flows from the Fed's wholesale payment applications (e.g., Fedwire Funds and Securities), which are separately settled by a real-time account balance system. That gives us intimate knowledge of how the system performs key tasks such as balance sheet management, account reconciliation, and tracking high-volume, high-dollar financial transactions between banks.

We soon realized one relatively simple but important operation—reconciliation of adjustments between depository institutions and general ledger accounts—could be achieved, and possibly improved, using smart contract execution on a blockchain.

The process requires a ledger that records the value of the different transfers between parties that are enabled by the Fed's payment systems, reconciles external and internal applications, and has strong resiliency and security requirements.

Our proof of concept sought to test functionality, including the ability to:

- load the balances of depository institutions onto the Ethereum ledger (Mock data was used for depository institution identifiers and balances.)
- establish wallets to represent a small number of pseudo depository institutions
- mimic depository institutions transferring balances among themselves
- validate transactions, verify balances, and reach consensus among the nodes
- create corresponding general ledger entries via smart contracts to reflect those transfers

We gained two key advantages by choosing to constrain the use case to a subset of tasks within a larger, well-known system: 1) simplicity; 2) the ready availability of knowledgeable business, technical, and cybersecurity resources. It is important to note that the purpose of our experiments was to *learn* the technology. We had no explicit plans to implement the technology in our production systems.

Application Architectures

The existing general ledger application follows a traditional three-tier Java EE architecture that includes a clustered environment and single database cluster. The proof of concept has a web tier application, which is based on a Java EE architecture that uses updated frameworks. This application interacts with a Hyperledger Fabric membership service component and two Fabric peers per organization. Each Fabric peer is backed by its own couchdb, which is utilized to maintain the current state of values on the ledger.

The Development Process

Like many organizations beginning to experiment with blockchain, we had to acquire knowledge of the programming language and tools. Since the Boston Fed viewed the project as a learning opportunity for our internal technologists, we decided not to work with any external firms so we could maximize our hands-on learning.

> *"In addition to making sure that the development tools were safe to use, it was important for us to document the scope and assumptions surrounding the work so that everyone remained on the same page.*
>
> *Naturally, the Boston Fed takes the security of its data and system seriously. To avoid confusion, we were very clear to say that our proof of concept did not interact with any of the Fed's payment applications or use any real-world data. This changed the risk posture dramatically and gave the development teams more freedom to experiment.*
>
> *As the team iterated over new tools, frameworks, hosting environments, etc., we would regularly document our common understanding, including testing goals, products we were using, known risks, mitigating factors, assumptions, and terms of use.*
>
> *This had an added benefit of making sure that our testing was focused and aligned with management expectations."*
>
> *—Matthew McHugh, Director of Information Security Risk & Compliance, Federal Reserve Bank of Boston*

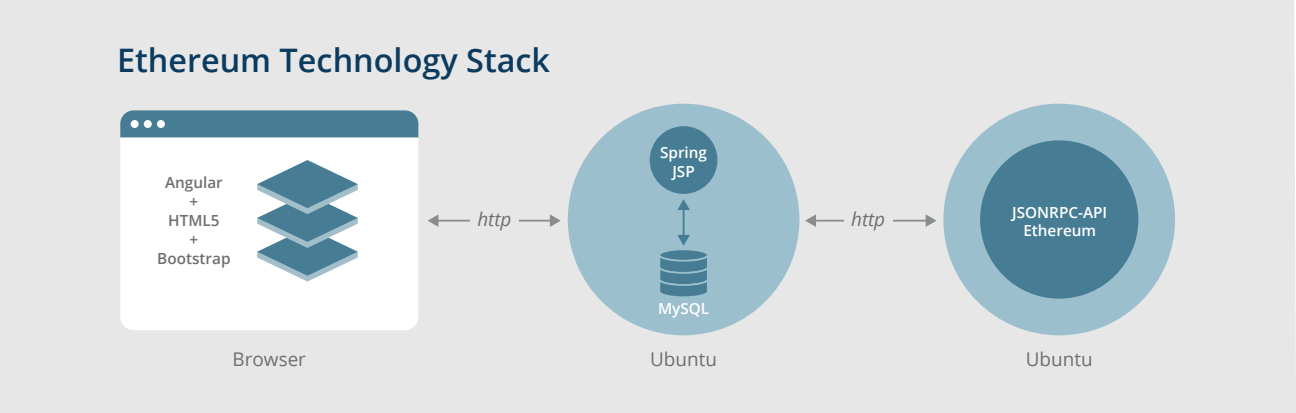
Additionally, we worked in close partnership with our cybersecurity team, because many of the tools needed to develop the application were new and/or open source. The team worked with us to detail designs, assess risks, evaluate tools, and verify processes. With every iteration, we were guided first by a commitment to security, while remaining open to new ways the technology could help us meet our

objectives. The Bank also had to make sure our internal laboratory environment applied the same control objectives required by existing systems. If, for example, we demonstrated viability, but without the expected controls in a production environment, the value of the project could be negatively impacted.

The Tech Stack: Ethereum

With everyone from leadership to IT on board, the tech stack took shape. In addition to the Ethereum platform described above, the stack running on Ubuntu in Amazon Web Services (AWS) included:

- Ethereum, Geth, JSON-RPC
- Tomcat, Angular 1, Java, Spring MVC
- supplemental database (for reporting and journaling purposes)



Ethereum Results

The success of any experiment depends not only on how well the solution works, but also what's learned along the way. This first trial provided many learning opportunities.

The team used the existing framework and methods provided through Ethereum's Geth JSON-RPC API, and a series of JavaServer Pages (JSP) to interact with the Ethereum back end. These pages enabled the submission of a transaction and pages for searching transactions on different criteria. Transactions and queries could all be performed as expected from a user-interface perspective. The user interface, the front end, would remain the same through the next phase of the project. We were interested in the portability of the code. Our implementation of the back-end Ethereum platform, however, offered many insights, though it ultimately would not prove viable for our use case for several reasons.

First, at the time Ethereum offered limited ability to query by a range. For example, if we sought results from transactions between $200 and $400, a query would return all transactions. Picking a subset of those transactions required a supplemental database external to the Ethereum back end.

Second, though the appeal of Ethereum as the platform of choice was based on many factors, one of them—the availability of support and documentation—proved more challenging than expected. Though it has a large developer community, there weren't many members equipped to answer questions about functionality beyond the basics.

Third, ensuring consensus among participating nodes still required a proof-of-work protocol. Even within a confined environment, the time necessary to create a block was slower than could be tolerated in a production environment. Viability in this case could not be demonstrated.

Another core element of the Ethereum protocol involved a concept known as "gas," which, as the name implies, is the fuel—in the form of cryptocurrency—of the platform. Gas is required as smart contracts execute transactions and the "miners" in the network create the blocks enshrining the changed state. Gas is used to pay the miners for the work they do, but it's also a security feature. If gas is required for each executed transaction, then an attacker seeking to cause disruption to the network by spamming it with endless transactions would also need to pay for each fraudulent transaction. This feature means a supply of "ether"—the account-based cryptocurrency native to Ethereum that represents the "gas"—needs to be maintained to facilitate transaction execution. Our use case focused on accurate transfers to perform account reconciliations—adding gas to each transaction was impractical and unnecessary, given that the "miners" were not decentralized, untrusted participants, but rather nodes within a closed network.

Finally, when functionality is not native to the platform, challenges abound. A core value proposition of Ethereum was the ability to build smart contracts and applications beyond digital currency. However, it was built as a permissionless and public network. That meant running a cloned instance of the platform within an internal, and therefore "private," laboratory environment—and native Ethereum at that time did not provide membership services. So, onboarding users and restricting access to the platform and read/write functions proved incompatible with the design of the Ethereum platform of 2016. But since a membership service is a critical feature of our accounting system, the inability to enable these controls ultimately made Ethereum ill-suited as a platform for this use case.

Though Ethereum couldn't support the project goals, the first trial successfully demonstrated the strengths and weaknesses of the platform at that time. The team learned much about basic performance, stability, encryption, reporting, diagnostic tools, and compliance options for meeting security requirements. These lessons would serve us well in the next trial.

Building a Hyperledger Fabric Blockchain Proof of Concept

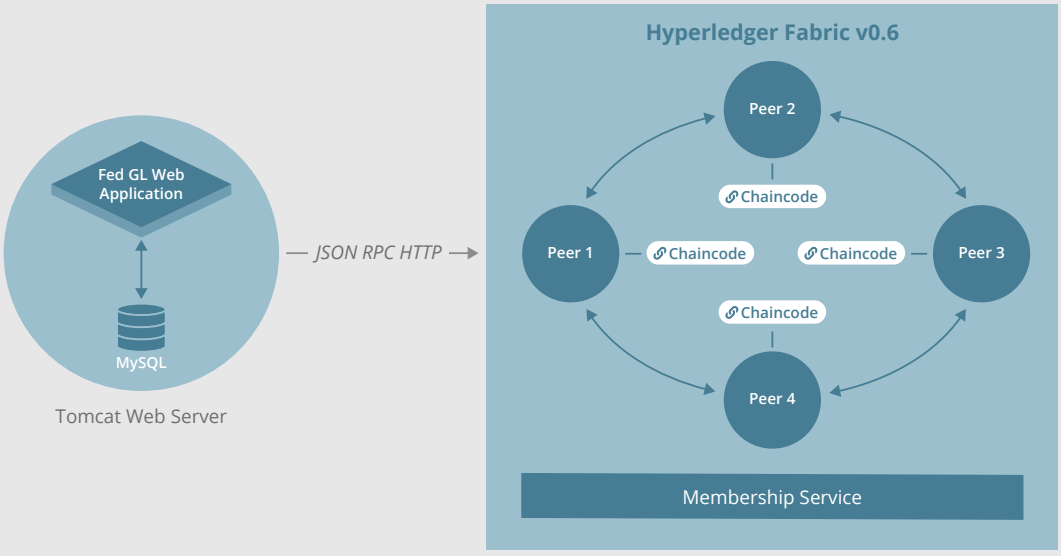
"Don't pretend you know the answers. Experiment; get feedback."
—Stephen J. Dubner, co-author of Think Like a Freak

In late 2016, our team began following an open source blockchain development project known as the "Hyperledger Project," which was started by the Linux Foundation earlier that year. We learned Hyperledger encourages cross-industry collaboration on blockchain and distributed ledger platforms.[6] The project, according to its website, "welcomed two business blockchain framework codebases into incubation." One of these, named Hyperledger Fabric, would become the platform for the next phases of our project.

The Tech Stack: Hyperledger Fabric v0.6

Hyperledger Fabric, the open source project spearheaded by IBM along with 27 other organizations, sought to create a modular, enterprise-level architecture, where components can be easily swapped out without affecting other parts of the system.[7] Importantly, Fabric also includes a membership service that requires prospective participants to prove their identity and be granted access before inclusion into the network. These and other features offered an improved value proposition over the available Ethereum options based on the Boston Fed use case.

Hyperledger Fabric v0.6 Architecture



However, at this point in Hyperledger's development cycle, the Fabric project was still considered in the "incubation" phase. A project under the Hyperledger umbrella does not graduate to "active" until certain criteria, including documentation

requirements, defect tracking, and infrastructure support, are in place.[8] We would be among the first to use code intended as a "preview" for developers. Nevertheless, we were eager to begin.

In the winter of 2016 – 2017, the team installed the necessary components on AWS to build a Hyperledger Fabric blockchain.

These components included:

- Docker and Docker Compose (a computer program that enables "containerization")[9]
- Chaincode/ smart contract (a small program that runs on the blockchain platform)
- Golang (programming language used for smart contract/chaincode)
- Hyperledger Fabric REST API

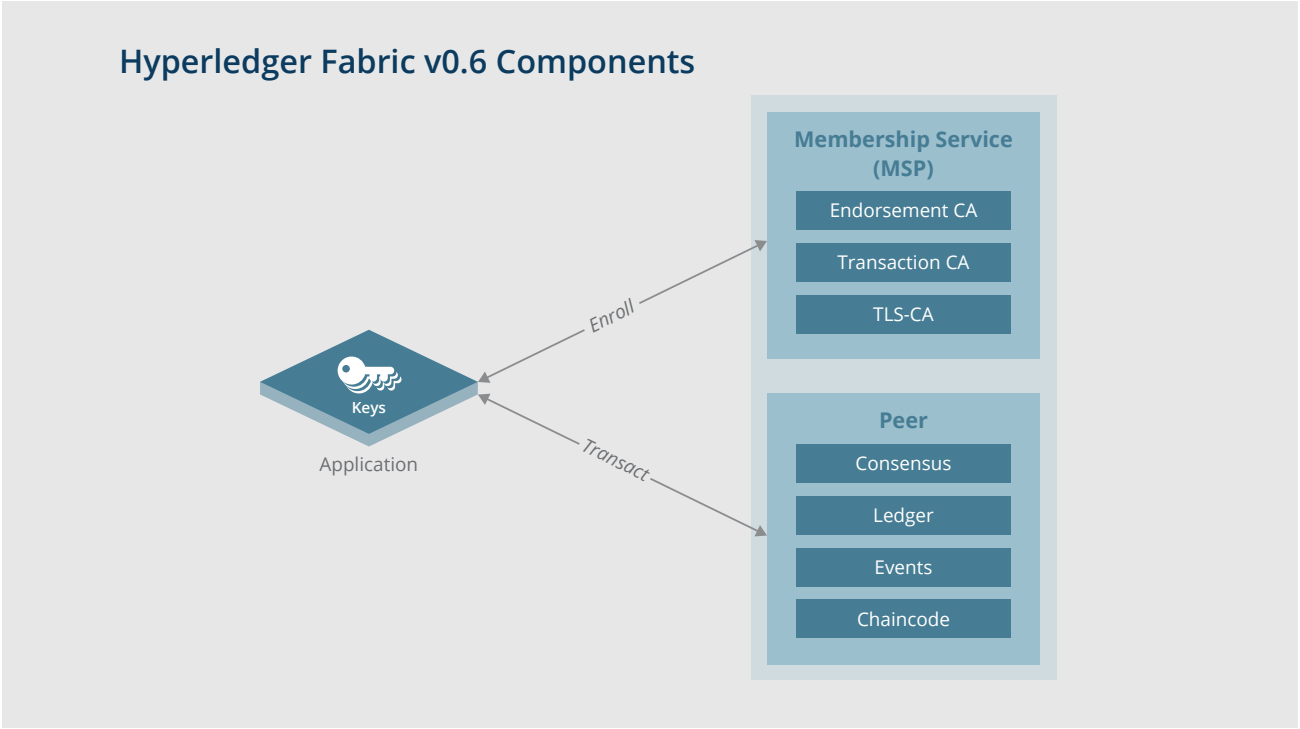
The project goals, use case and requirements, user interface built using Spring MVC and Angular 1, and Ubuntu remained the same during this trial.

Containers

A container is a self-contained, lightweight, executable package of software that includes all necessary dependencies needed to run an application. Containers run on top of the host computer on a single operating system. Multiple containers running on the same host share the operating system kernel, making them much more lightweight than a traditional virtual machine. Because an application's dependencies are all included in a container image, the image can be instantiated much more quickly than in a traditional environment, where all dependencies need to be met before running the application.

Hyperledger Fabric v0.6 Results

Through native membership services, Hyperledger Fabric version 0.6 (v0.6) solved one of the core challenges the team encountered during the Ethereum trial, but it was more difficult to deploy. Specifically, though documentation was available, there simply wasn't any support beyond basic implementation—in other words, when any customization was needed. (This wasn't wholly unexpected, given the code was still in the preview stage.) But with hours of watching instructional YouTube videos and some false starts, our team was able to build the necessary knowledge and create a simple, functioning architecture. The difficulty with setup, however, was worth the effort because the more robust permissioned capabilities showed promise.



Interestingly, before we had done much testing on v0.6, Fabric version 1.0 (v1.0) was launched in spring 2017. This version implemented many solutions to the challenges the team had encountered with v0.6, such as adding the ability to create privacy among participants.

The team migrated from v0.6 to v1.0 in the second quarter of 2017.

The Tech Stack: Hyperledger Fabric v1.0

Though it retained some of the basic building blocks of v0.6, such as Docker, Golang, and chaincode, our team needed to make several updates. First was the removal of the REST API in v1.0, which required the team to pick a Software Development Kit (SDK) to utilize. The Java SDK was not yet as mature as the Node SDK, so we installed the Node Server and Node SDK and also modified the web applications service layer to communicate with the Node SDK, rather than directly with the Fabric network.

In addition, the Hyperledger Fabric development team re-architected the consensus model in v1.0. The Bitcoin blockchain uses a proof of work consensus method to determine which node in the network "wins" the right to add a valid block to a chain. Because the system is intended to enable nodes that do not know or trust each other to participate in the network together, a mechanism to reach agreement and affirm the validity of the chain as the canonical record is essential. The protocol for consensus must provide participants with reasonable assurance that the block is valid.

In a private, permissioned blockchain environment, access to the network is controlled through the membership service and established through some kind of authority, agreement, or business rule. Thus, though consensus is necessary to ensure all of the nodes within the network agree on the state of the canonical ledger, the method to arrive at agreement does not need:

- an incentive structure to reward "miners" for validation and the addition of a block to the chain
- multiple nodes in competition to "win" the right to add a block

This means different options are available to keep the network synced and compliant with specified protocols.

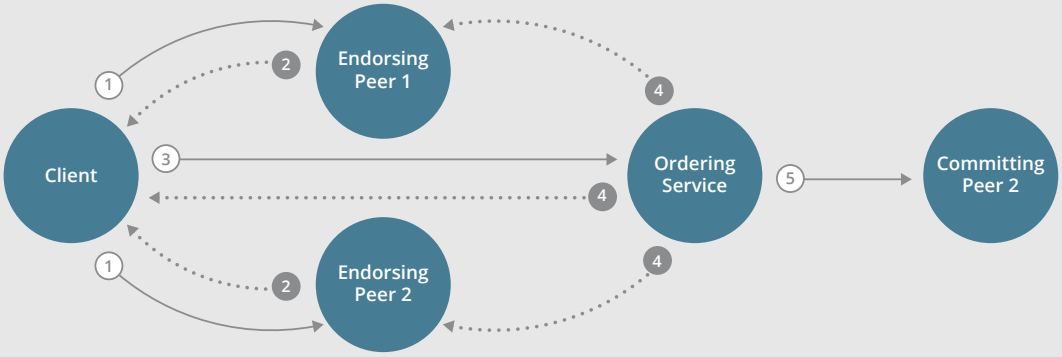
Hyperledger Fabric v1.0 defines consensus as "a broader term overarching the entire transactional flow, which serves to generate an agreement on the order and confirm the correctness of the set of transactions constituting a block."[10] Because neither incentive nor competition is needed to achieve consensus, the design of Fabric allows users to define the conditions under which a transaction can occur. These conditions are created as part of the smart contract using "chaincode." The chaincode is the program establishing the basic environment and business rules transactions must follow. It is the smart contract. These business rules are then used to "confirm correctness."

Several components in Fabric v1.0 work together.[11] A "peer" is a node in the network which can have various roles. Peers can also be removed, or their roles can be changed according to need. A peer node houses both a ledger and the chaincode logic. It also commits a block to the chain. A peer may also endorse a transaction based on the business rules or "policy" established as part of the smart contract.

In addition, the ordering service—or "orderer" in Fabric—batches and broadcasts transactions to peers and contains the core "system chain" with information about who belongs to the network and what rules they will all follow. There can be a number of orderers, but in our case, we needed only a single orderer ("Solo") responsible for receiving transactions and broadcasting them to the appropriate peer in the network to perform the commitment functions, which then adds a block to the chain.

Further, the endorsing peer receives the transaction as a proposal and affirms whether a particular transaction is valid according to the required logic. Once endorsed, the result is sent back for validation to the client application—the originator of the transaction. The client application then sends the endorsed transaction to the ordering service, which batches it and sends it to the committing peer. The committing peer appends the endorsed transaction to the appropriate ledger, which—as we will discuss shortly—may not be viewable for all participants in the network.

Hyperledger Fabric v1.0 Transaction Flow

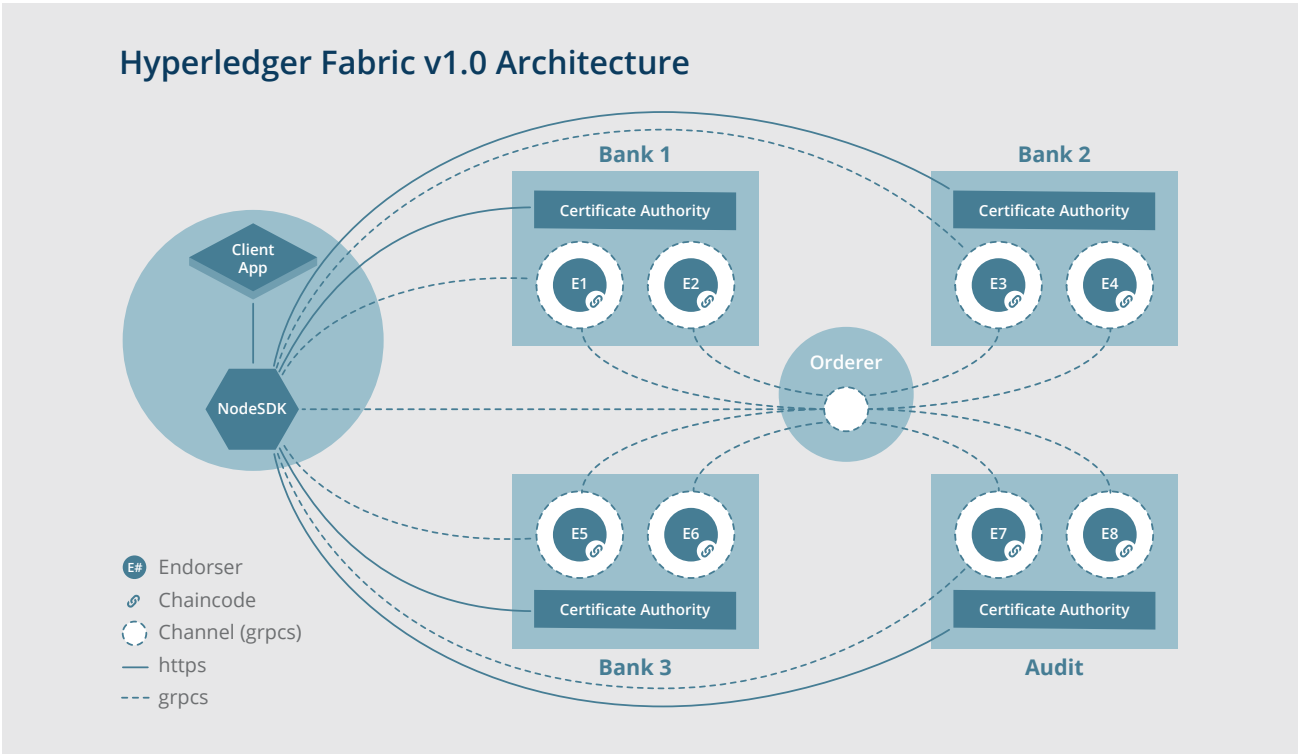


1. Simulate proposed transaction (execute chaincode), then endorse transaction.
2. Collect transaction-endorsed message into a valid endorsement that satisfies endorsement policy.
3. Submit endorsed transaction to orderer.
4. Verify endorsement, readset.
5. Apply writeset to state.

While major changes were made from v0.6 to v1.0, the two of greatest interest for our use case were the improvement in the methodology around correctness and agreement and the creation of "channels" for transaction privacy. With the ordering service and separation of responsibilities among peers, greater efficiency, flexibility, and resilience are possible. We thought the channel architecture would address the challenge of protecting confidential data, which was impossible in v0.6 and the native Ethereum environment.

Effectively, Hyperledger Fabric v1.0 enables each participant in the network to transact with any other participant in the network without exposing confidential information to others who are not parties to that particular transaction. So, when a peer node commits a transaction, it is only committing that transaction to the ledgers of the participants privy to that particular event.

Therefore, we set up channels for representative "banks" simulating (on a much reduced scale) client accounts our general ledger reconciliation case would support.



Hyperledger Fabric v1.0 Results

In our implementation, we elected to keep the structure simple. The project goals did not include specifically testing the throughput of the network, latency, or resilience. Because we were seeking to understand the technology, we hoped to better define additional testing or proof of concept efforts while we monitored the technology as it matured. To that end, this trial delivered exceptional results.

Based on our experiments overall, setting up a basic environment was relatively straightforward. We performed each of the functions we outlined in our proof of concept goals, so Hyperledger Fabric clearly shows promise as one of multiple platforms for further experimentation.[12]

The results also prompted us to initiate a relationship with the Linux Foundation and join Hyperledger working groups. Not only did our project help us ask better questions about how this technology can be used, it fostered a commitment to extend our learning into the community.

Along with the benefits, we found some clear challenges. For instance, building a proof of concept architecture in a rapidly evolving open source development environment carries inherent complications. When one must install new or updated versions of tools repeatedly, the risk of having to repeat the learning process or materially revise code is high. Additionally, the volunteer-based, open source

community was not always available and the advice was not always high quality. We experienced this with Ethereum, and the same was true for Hyperledger Fabric, both before and after the launch of v1.0.

Another important insight about the channel architecture introduced by Hyperledger Fabric v1.0 was that while the channels do appear to enable privacy among participants, the channel setup was complicated and they did not scale well.

Channel Setup

To enable privacy among participants, a channel is established, and network participants subscribe to the channel. In our model, if only "bank 1" and "bank 2" subscribe to a channel, only "bank 1" and "bank 2" have access to the transactions they are party to. However, even in our simple model, we observed that construction with three "banks" and a "supervisory node" required at least three channels for bilateral transactions, where the "supervisory node" is subscribed to each channel to "listen" to all transactions. In fact, any number of nodes can be subscribed to a channel with the understanding that all channel participants will receive the transaction details related to that channel. But this creates a problem.

In channel architecture, if three parties, let's call them "Alice," "Bob," and "Claire," are involved in a one-time transaction, a unique channel would need to be established. But if Alice and Bob continue to transact, and Claire is not involved any longer, Claire continues to have access to the transaction records of Alice and Bob, whether they pertain to her or not. To prevent this, another channel would need to be created between Alice and Bob only. Meanwhile, the original three-party channel would exist as a separate ledger until removed from the network or deleted. When there are hundreds of entities on a given network, the number of potential channels multiplies exponentially, as does the effort to manage them. The considerable trade-offs that come with either including all transactions in one channel or creating separate channels for each group of transacting entities means that neither option may be desirable.

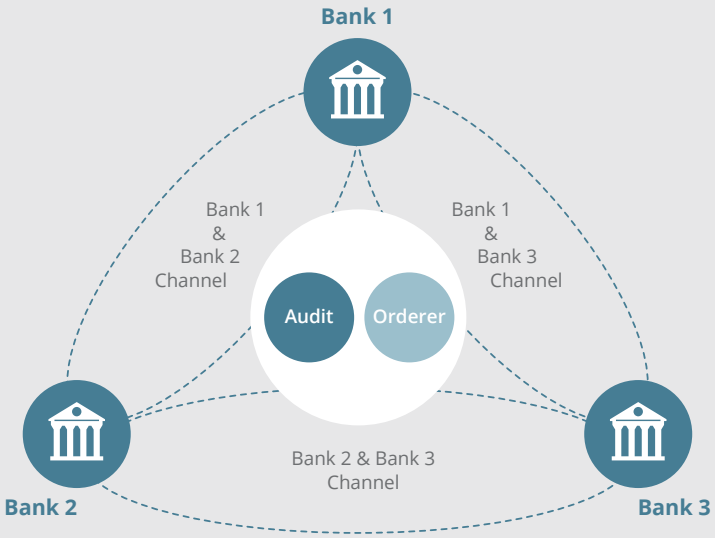
In sum, though Hyperledger Fabric is the most mature of the Hyperledger projects, it may not be ready for primetime when mission-critical systems are involved.

A New Use Case: A Supervisory Node

"Intelligence is the ability to adapt to change." —Stephen Hawking, Theoretical Physicist

The Federal Reserve "is responsible for supervising—monitoring, inspecting, and examining—certain financial institutions to ensure that they comply with rules and regulations, and that they operate in a safe and sound manner."[13] We decided to investigate what we could learn from developing a rich set of capabilities for the supervisor peer subscribed to each of the channels in our simulated network. As with our previous experiments, our goal was simply to learn about the technology and what business questions are generated by its use. It was not our intention to imply that we would actually implement this technology as part of our supervisory function.

Fed GL Hyperledger Fabric 1.0 Architecture



It is worth noting that "supervisory node" is a generic term and can include many roles beyond the Fed's regulatory supervisor role, such as an auditor, payments network rule-enforcer, or data reporting entity. As such, some of our questions and potential experiments are more generic and not necessarily tied to any of our specific roles as a regulatory supervisor. The following are some questions we have about what blockchain technology and smart contract logic, possibly coupled with artificial intelligence and machine learning, could accomplish:

- What business functions (audit, regulatory supervisor, payment network rule-enforcer) could supervisory nodes perform?
- What architectural questions do supervisory nodes create?

- How can data access be limited only to whatever is needed to perform the stated function?
- Is there any way the supervisory node could be compromised or create operation risk for the network?
- When multiple blockchain platforms are utilized for a given business process (e.g., delivery versus payment, or DvP), what impact does this have on the supervisory node(s) architecture and performance?
- If sister supervisory agencies that share oversight responsibilities were able to develop a single supervisory node structure, what new architectural/technical questions does it raise?
- What can be done to detect and control malicious actors in a private payment network? (This may extend beyond the Federal Reserve's role, and applies more appropriately to any parties charged with enforcing rules, such as a private payment network operator.)
- How can fraud be detected? If it can be detected in a shared network by a node supported by AI logic, what possible actions are appropriate?

While we have no results on this use case yet, we hope sharing these ideas will inspire others.

Until Next Time

As the soap factories of the 1960s taught us, the complexity of seemingly simple problems sometimes defies theoretical models. In our view, a combination of reading, listening, and thoughtful repeated experimentation delivers the best results. This paper wasn't written to be an instruction manual. By detailing how we planned for and charted our own journey, the challenges we encountered, the lessons we learned, and the areas we'd like to continue to explore, we did not endeavor to give definitive answers or recommendations. Rather, we want others to benefit from our experiences and draw their own conclusions.

Key Takeaways

- Small-scale experimentation offers cost-effective opportunities to learn about blockchain technology with reduced risk.
- Engaging with security partners early and often ultimately supports an assessment of viability. Even in an internal laboratory environment where some compromises may be possible, a full vetting of security is needed, and much can be learned from it.
- Open source platforms offer many advantages. However, comprehensive documentation and support may be lacking for earlier stage platforms, given the volunteer nature of the development process. Additionally, significant customization

may be required depending on the use case, which means organizations should expect to invest in knowledge development for technologists.

- The lack of a stable platform means risk is significant for major changes in the architecture or tools used to impact in-progress experimentation.
- The technology is complicated. Blockchain is the platform underlying the application, so business users will not necessarily see the difference between the legacy process and one that includes a blockchain application. Therefore, much of the benefit may be in gains in efficiency and resilience, or automation of previously manual processes, rather than new or improved functionality for the end user.
- New partnerships and consortium efforts continue to form around this technology, offering additional insights and new opportunities for learning.

Conclusion

Development takes time, and it's tough to predict how a technology evolves and changes based on multiple variables, including the availability of resources in funding, time, and expertise. For our purposes, blockchain technology is not yet mature. The ability to scale, manage privacy, and support interoperability with both new and legacy systems will be critical to its adoption for financial services. While we believe blockchain technology is not currently viable for significant production use at the Federal Reserve, we believe continued experimentation is essential and can contribute to the continued maturing of the platforms.

In addition, we are interested in exploring the capabilities of a supervisory node. The Boston Fed team is also active in the Hyperledger Fabric community and working with other consortium groups like R3 Corda on projects to expand Know Your Customer capabilities. There are currently many other potentially promising efforts and platforms in the market. We aren't endorsing those we chose to work with. We simply selected those that made the most sense to us at the time.

We encourage others to share what they've learned from their experiments. Some hype invariably exists, and perhaps blockchain technology will not on its own bring about the kind of massive improvements in transparency, efficiency, and other areas some have predicted. But we believe it shows promise.

Acknowledgments

White paper sponsors: James S. Cunha (Federal Reserve Bank of Boston), Paul W. Brassil (Federal Reserve Bank of Boston), Guy Berg (Federal Reserve Bank of Minneapolis). White paper author: Angela Lawson (Federal Reserve Bank of Minneapolis). Members of the white paper working team: Rachel Bissett, Nicolas Brancaleone, Charles Buhecker, Peter Davis, Heidi Furse, Jay Lindsay, Matthew McHugh, Aniruddha Sinha, Lucy Warsh, Ken White (Federal Reserve Bank of Boston).

Appendix

What is Blockchain?

The word "blockchain" never appears in the paper that popularized the concept. Satoshi Nakamoto, the pseudonymous publisher of the 2008 white paper "Bitcoin: A Peer-to-Peer Electronic Cash System," sought a way to overcome an "inherent weakness of the trust based model" in today's payment systems.[14] Essentially, he, she or they seemed to think the current financial system worked pretty well, but could benefit from a means to make payments peer-to-peer without needing a trusted intermediary like a financial institution. This new system, Nakamoto thought, could reduce fraud and transaction costs.

What later became known as blockchain was the answer to the fundamental problem the author(s) needed to solve in order to achieve their stated goals. To change the trust model from one reliant on a central authority, the system had to prevent double-spending of electronic "cash." To do that, Nakamoto suggested a combination of known elements, including:

- accounting ledgers
- distributed computing
- a decentralized consensus protocol
- cryptography (digital signatures, asymmetric key pairs, hash algorithms)
- an incentive or reward structure

Bitcoin was the first application to run on a public, decentralized platform with all of these features. Though none of the elements were new in and of themselves, the combination represented an innovation potentially applicable in a number of business use cases.

Some have suggested blockchain will enable the "internet of value," where transfers of assets, resources, intellectual property, and data can be exchanged between parties without needing an existing relationship or the establishment of trust either directly or through a third party. Supporters claim the math supplies valuable features like non-repudiation to prevent fraud, resilience through decentralized systems, and efficiency through the elimination of intermediaries between transacting parties.

The Sum of Its Parts

Nodes

The public nature of the Bitcoin blockchain makes it attractive for its users in many ways. Because it is public, everyone who participates in the network has a copy

of the ledger. To participate, a user typically establishes one or more "nodes" on the network. A user may decide to become part of the verification network or may decide to be a more passive participant. In a public blockchain, everyone has the ability to view all of the transactions committed to the ledger.

Block

A block is a data structure containing information on a group of occurred transactions. Blocks are added to the end of the chain and, once added, the chain cannot be altered. Every node has a copy of the blockchain.

Digital Wallet

In addition to establishing a node, a participant needs a means to transact on the network. They do this through establishing a "digital wallet," a piece of software capable of managing the inputs and outputs of transactions and generating asymmetric key pairs, one private and one public.

Public and Private Keys

The public key is seen by all and is used by participants to direct and receive funds, similar to your home address or email address. The private key is used to "sign" or authenticate transactions. Think of a private key as something like a personal identification number. It is supposed to be a secret that only you know, and it is used to validate that the owner of the secret is also the owner of the value being exchanged across the blockchain network. The public and private keys are computationally related to each other, but can't be derived from each other, so knowing one does not mean you can figure out the other. When a transaction is initiated, a participant enters the public address of the wallet he or she is sending funds to and signs the transaction with their private key. But before this transaction can be considered complete, it must be checked for correctness in formatting, bundled into a block, and added to the chain. Now, every node in the network can check that the block is valid. As mentioned, some participants just transact. Others earn incentives by doing the work of bundling the transactions into blocks.

Miners

Participants who wish to earn the incentives built into the platform "compete" for the right to add the next block in the chain. These participants are known as "miners."

Miners, like most participants in the network, are known only by the public key address used by all participants to send and receive transactions. In addition to transaction fees, miners earn rewards in the form of Bitcoins when they solve a computationally-intensive puzzle and win the right to add the next block in the chain.

Proof of Work

The puzzle requires a miner's computer to crunch numbers until it finds a combination that produces a number that meets the criteria designated by the protocol. This math problem is difficult to solve, but a correct solution is easy for others in the network to verify quickly. Known as "proof of work," this protocol produces a consensus among participants that the blocks in the chain are valid, showing that the "state of truth" for the blockchain is accepted by all participants.

The process of investing computing power to earn rewards maintains the network because it does the work of adding transactions to the chain, but it also incentivizes the continued participation, maintenance and existence of the network over time.

When a Strength Becomes a Weakness

Bitcoin's blockchain and its governing protocol deliver much of what Nakamoto and others envisioned: a secure way to create a peer-to-peer value transfer system without a central authority. Central banks are among the many that have recognized the potential of the intersection of distributed database technology, cryptography, incentive structures, and ledger-based accounting—the blockchain.

But the same qualities that make the Bitcoin blockchain useful for its participants (it's public, maintained through a decentralized participant-driven network, and pseudonymous) make it an unacceptable tool for most financial services business applications. There are no controls or barriers to directly joining a network beyond a computer and technical know-how. This is a problem when a business such as a financial institution is required to comply with Know Your Customer or anti-money laundering rules. If anyone can join the network without any processes to confirm identity, a business can't know who's on the other end of a transaction without an intermediary of some kind.

Blockchain Permission Matrix

	Who Can Make Changes to the Blockchain?	Who Can Read Information from the Blockchain?
Permissioned Public	Only authorized parties	Anyone
Permissioned Private	Only authorized parties	Only authorized parties
Permissionless Public	Anyone	Anyone
Permissionless Private	Anyone	Only authorized parties

Both Bitcoin and Ethereum are meant to support public participation. The barriers to entry are low. You need only a computer capable of downloading the software and the skills to utilize it.[15]

A Focus on Smart Contracts

Innovators, inspired by the idea of blockchain, envisioned a more generalized platform capable of supporting numerous different applications. Among the first to consider how businesses might effectively utilize a blockchain platform was a 19-year-old Russian-Canadian. In 2013, Vitalik Buterin wrote a paper titled, "A Next Generation Smart Contract and Decentralized Application Platform."[16] The idea behind the Ethereum platform was born.

The ability of the Ethereum platform to support "smart contracts," what Buterin describes as, "systems which automatically move digital assets according to arbitrary pre-specified rules," offers a particularly compelling opportunity. With a smart contract, actions can be pre-programmed to execute based on different events. While "if/then" logic isn't new, there's potential in the ability to use sophisticated, self-executing if/then logic and complete transfers of value secured with cryptography in such a way that authenticity of the data, non-repudiation of the transaction, and a single shared data source among transacting parties can be achieved without necessarily trusting each (or any) party.

Blockchain Evolution Moving Forward

Just as blockchain platforms evolved to provide more robust (i.e., Turing complete) programming languages, more limited access to data on the blockchain and new consensus methods (e.g., proof-of-stake) will continue to evolve to meet the growing needs of the business and government entities looking to exploit them. Keeping up with the evolution will be worth the challenge.

Endnotes

[1] "Evolution Special: Origin of a Theory, the End of Evolution." (2009, Dec.-Jan.). Cosmos Magazine. https://cosmosmagazine.com/issues/evolution-special-origin-of-a-theory-the-end-of-evolution

[2] Rapier, G. (2016, June 6). "Yellen Reportedly Urges Central Banks to Study Blockchain, Bitcoin." *American Banker*. https://www.americanbanker.com/news/yellen-reportedly-urges-central-banks-to-study-blockchain-bitcoin

[3] Brainard, L. "The Use of Distributed Ledger Technologies in Payment, Clearing, and Settlement." Speech presented at Institute of International Finance Blockchain Roundtable, Washington, D.C. https://www.federalreserve.gov/newsevents/speech/brainard20160414a.htm

[4] Gartner, Inc. (2016, August 16). "Gartner's 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage" [Press release]. https://www.gartner.com/newsroom/id/3412017

[5] "iEthereum: A Global Decentralized Cryptocurrency." http://iethereum.trade/whitepaper/

[6] The Linux Foundation, Hyperledger. (2016, February 9). "Linux Foundation's Hyperledger Project Announces 30 Founding Members and Code Proposals to Advance Blockchain Technology" [Press release]. https://www.hyperledger.org/announcements/2016/02/09/linux-foundations-hyperledger-project-announces-30-founding-members-and-code-proposals-to-advance-blockchain-technology

[7] "Hyperledger Fabric." https://www.hyperledger.org/projects/fabric

[8] Behlendorf, B. (2017, March 3). "Our Incubator's First Graduate: Hyperledger Fabric" [Web blog post]. https://www.hyperledger.org/blog/2017/03/03/our-incubators-first-graduate-hyperledger-fabric The decision to move to an "Active" status is made by the Hyperledger Technical Steering Committee (TCS). In Fabric's case, the TCS voted to "graduate" this project—the first under the Hyperledger umbrella—in March of 2017. "The exit criteria by which projects are evaluated in order to graduate from Incubation include legal compliance, community support, test coverage and continuous integration support, documentation, architectural alignment, published releases, and infrastructure support for such things as requirements and defect tracking, code reviews, continuous integration testing and more."

[9] "What Is a Container: A Standardized Unit of Software." https://www.docker.com/resources/what-container.

[10] "Glossary Release-1.1." Hyperledger Fabric: Read the Docs. https://hyperledger-fabric.readthedocs.io/en/release-1.1/glossary.html.

[11] "Welcome to Hyperledger Fabric." Hyperledger Fabric: Read the Docs. https://hyperledger-fabric.readthedocs.io/en/release-1.0/.

[12] Other platforms have evolved considerably since our initial experiments and could also be appropriate for our business case. For a side-by-side comparison of three platforms based on a financial services business case, see Project Ubin: http://www.mas.gov.sg/~/media/ProjectUbin/Project%20Ubin%20Phase%202%20Reimagining%20RTGS.pdf.

[13] "Board of Governors of the Federal Reserve System." Supervision & Regulation. https://www.federalreserve.gov/supervisionreg.htm.

[14] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008. https://bitcoin.org/en/bitcoin-paper.

[15] Wüst, Karl, and Arthur Gervais. "Do You Need a Blockchain?" 2017. https://eprint.iacr.org/2017/375.pdf. Description of public, private and permissioned, permissionless.

[16] Buterin, Vitalik. "A Next Generation Smart Contract & Decentralized Application Platform." 2013. https://github.com/ethereum/wiki/wiki/White-Paper.