SYM-AM-21-105

EXCERPT FROM THE PROCEEDINGS OF THE EIGHTEENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

Blockchain Data Management Benefits by Increasing Confidence in Datasets Supporting Artificial Intelligence (AI) and Analytical Tools using Supply Chain Examples

May 11–13, 2021

Published: May 10, 2021

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF DEFENSE MANAGEMENT
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Defense Management at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Acquisition Research Program
Graduate School of Defense Management
Naval Postgraduate School

Blockchain Data Management Benefits by Increasing Confidence in Datasets Supporting Artificial Intelligence (AI) and Analytical Tools using Supply Chain Examples

Anthony Kendall—is a lecturer and researcher at the Naval Postgraduate School where he is in the Information Sciences department researching and teaching in the field of data science including Investigating the emerging "Big Data" and machine learning concepts and technologies on how they enhance analytics for decision-making in DoD areas such as logistics and CTAP (Common Tactical Air Picture). He currently heads a project on using "money ball" and other methods to provide analytics for Navy aviation maintenance as well as a project on investigating how blockchain might assist in Navy logistics processes. Tony graduated from the University of Texas at Arlington. In his earlier career he was a Naval Flight Officer (lieutenant commander) He wrote an award-winning article (USNI) on creativity in the military. He received his Master of Science at the Naval Postgraduate School.

Arijit Das—is a Computer Science research faculty at the Naval Postgraduate school specializing in teaching CS courses and doing applied research trying to find industry solutions for DoD/DoN challenges. His educational background is BECS from MNREC (Allahabad University, India), followed by a MSCS from Oregon State University (Corvallis, Oregon). His other MS degree is an MSEE from UNLV (Las Vegas, Nevada). His work experience is in the Philadelphia tri-state area service industries as a developer using technologies running on Mainframe, Oracle, and UNIX as production servers. Arijit is a frequent speaker in Oracle Database user group events in the Silicon Valley. His current research focus is Big Data, AI/Analytics and Blockchain.

Bruce Nagy—is a Senior Scientist and Systems Engineer at the Naval Air Warfare Center, Weapons Division at China Lake. His research focuses on advanced game theory techniques, artificial intelligence and machine learning applications as applied to naval weapon systems and tactical decision aids. Nagy received an BS in Biology and BA in Mathematics from the Citadel. He went on to receive an BS and MS in Electrical Engineering from the Naval Postgraduate School. He continued his neural network interest with post graduate work in modeling brain stem activities to muscle fibers at UCLA, in cooperation with NIH. Nagy is a former Engineering Duty Office. bruce.nagy@navy.mil

Avantika Ghosh—is a sophomore at University of California, Berkeley studying Computer Science and Economics. She has been a research intern at NPS in 2018 and 2020. In 2018, she conducted a Data Analytics study on benchmarking various machine learning algorithms' to detect anomalous behavior in military aircraft data. Through her current research, she is exploring possible applications of blockchain technology in the Navy. Avantika has been awarded the Bay Area Affiliate Winner award by the National Center for Women in Technology.

Abstract

We describe how Hyperledger Fabric (HLF) blockchain (BC) technology that we previously applied to Navy logistics supply chains can be applied to data supporting artificial intelligence (AI) and software development in terms of system safety and the timely acquisition of data. Data-driven AI/machine learning (ML) requires trusted data for their use in AI functions and requires significant amounts of training data from diverse sources including Internet of Things (IoT) devices/sensors. Unauthorized alterations to data supporting AI/ML could go unnoticed within the AI function build process but surface during operation in hazards affecting unwanted human death or resource destruction. AI/ML controlling hardware usually falls into the two highest software control categories: Levels 1 and 2, risk of death, disability, or resource destroyed.

HLF BC is a tamper-resistant decentralized trusted ledger that provides proof of transaction where trust is implemented through distributed consensus to ensure that only authorized people can modify data and that the modification is traceable and transparent. Distributed ledgers provide system safety through BC provenance, immutability, and policy enforcement through smart contracts.

We show how BC can contribute to the safety of the data and transactions and provide data to the researchers in a timely manner through “smart repositories.”

Introduction

Artificial intelligence (AI) software exercises a high degree of control over a particular system function (e.g., movement/guidance of a missile). If the function creates a hazard, this can cause a mishap that has a consequence of a catastrophic and critical event, resulting in death or resources destroyed. There is no redo, reboot, or retraining of an AI function that fails in this scenario. Software safety engineering and test and evaluation efforts to ensure fidelity include data-related elements such as process flow, code level, and data structure analysis. These flows, for example, are similar to the use of blockchain (BC) in supply chains used in our previous research, the Navy supply chain process, which we believe can be adapted for system safety and software integrity.

AI/machine learning (ML), the training sets, algorithms, and associated software supporting weapons systems are targets for increasingly sophisticated adversarial machine learning attacks, which attempt to fool models through malicious input into the system such as AI poisoning and other attacks. Athalye et al. (2017) showed that it is even possible to fool an AI into having it identify a turtle wrongly as a rifle. BC could be used as a countermeasure to prevent such poisoning as well as safeguard system integrity. BC, specifically Hyperledger Fabric (HLF), is a tamper-resistant decentralized trusted ledger that provides proof of transaction where trust is implemented through distributed consensus to ensure that only authorized people can modify the code base, AI algorithm, or training set and that the modification is traceable and transparent. Distributed ledgers provide system safety through BC provenance and policy enforcement through a feature called smart contracts, which imbed logical code. Data in support of AI and software development can also suffer not only from deliberate sabotage or ruse but also from human error.

Machine learning increasingly requires complex data sources from repositories and sensors down to the edge for training sets supporting AI development. Getting the right and accurate data can be a complex process, and error or intentional manipulation is always a concern. The number of sensors and Internet of Things (IoT) devices, such as smart thermometers/oximeters to track COVID-19, has caused an explosion of data generation but not an increase in safeguards to ensure system safety if these edge devices are used to control machines or make life-critical decisions.

Centralized security and authentication controlling IoT devices could lead to a single point of failure, a new target for cyberattack, and cause a bottleneck and high latency (Jia et al., 2020). Typically, an ML project may require diverse data sources and modalities. One example may be drones flying over an urban area, which requires its ML training set data on the region, including crime rate, weather, and road conditions/constraints. For just this simple example, data needed include Naval War College (NWC) wargaming, Naval Postgraduate School (NPS) wargaming, Live Fire Event, Program of Record product performance specifications, contractor specifications, test evaluation results, a diverse set of sensors, IoT devices, and so on. Once an AI is trained, BC can be used to ensure the integrity of the data during operations. BC can be used to find the right data, what is in it, who owns it, and how to get it with quick authorization. Data scientists have long recognized that just getting the right data and permission to use it can be an arduous and long process.

The Problem of System Safety

AI has the potential of creating a technological leap (Eden et al., 2013). That potential leap, especially when dealing with weapon systems, needs scrutiny. This scrutiny focuses on the specificity of the composition and size of the training data algorithm. This research describes

how an HLF architecture can be used to increase safety and confidence in the deployment of AI functions. There must be confidence in the data and training sets and the algorithms, and there must be confidence that they are tamper-proof and free from anomalies, intentional or by accident. Acquisition communities cannot identify and certify operational constraints of an ML algorithm for deployment without having confidence in the training data quality, including any negative side effects (Everitt, 2018) that might result from the training process.

The *system safety* concept calls for a risk management strategy based on identification, analysis of hazards, and application of remedial controls using a systems-based approach. This is different from traditional safety strategies (Roland & Moriarty, 1990).

AI safety issues for naval weapon systems usually have not included consideration of adversarial attacks that might affect functional performance. AI adversarial network attacks using techniques like deepfakes, putting an image/video into another image/video for miscategorization (Chauhan, 2018), will be considered within our BC discussion.

When assessing safety, the goal is to identify anything that might be safety-critical. *Safety-critical* is "a term applied to a condition, event, operation, process or item whose mishap severity consequence is either catastrophic or critical (e.g., safety-critical function, safety-critical path, and safety-critical component)" (Defense Standardization Program Office, 2012). Specifically, the publication MILSTD-882E (Defense Standardization Program Office, 2012) helps software engineers determine the level of rigor (LOR), which specifies the depth and breadth of software analysis and verification activities necessary to provide a sufficient level of confidence that a safety-critical or safety-related software function will perform as required. ML/AI usually falls into the system safety two highest software control categories: Level 1 (autonomous) and Level 2 (semiautonomous). We contend that BC could contribute to the analysis and verification of software activities by ensuring data integrity and better accessibility to the data.

Applying Successful BC Techniques to Ensure System Safety of AI Deployed Weapon Systems:

Our previous research used the HLF BC to generate three general use cases for Navy logistics, including financial and inventory transaction audit trails, serial number tracking, and maintenance log integrity. We believe the BC network derived from these three use cases could be adapted for system safety purposes since all our previous demonstrations dealt with the integrity of the data supporting work processes and events. BC tracks food/parts items as assets recorded on ledgers, and training data are assets and also created with similar work processes and events. With HLF you can control who, what, and when and identify those who have access to the logistics data representing assets as well through an immutable ledger containing logistics data that cannot be tampered with. HLF is as transparent as needed but can hide data from those without a need to know.

The data source flows of data and training sets supporting data scientists are similar to previous BC research on Navy supply chains to improve transparency and the safety of the related supply chain data and transactions, but there is a higher level of risk since they are often at Level 1 or Level 2 autonomous systems. In a sense, training sets and analytical data are like the tracking of parts and food since they point to resources represented by the information that needs to be protected and distributed in a friction-free manner. Control of these sources during the integration process to create training data and general analysis is vital to ensure the training sets and AI algorithms are transparent to those who need them, controlled, and their validity supported by an audit trail that BC provides. Training set alterations could go unnoticed within the AI function build process but revealed during operation in hazards affecting unwanted human death or resource destruction. Our previous research demonstrated how BC can provide

a needed data management technology through a tamper-resistant decentralized trusted ledger that provides proof of transaction where trust is implemented through distributed consensus. Only authorized people can modify the code base, AI algorithms, or training set modifications that are detectable, traceable, and transparent. Distributed ledgers provide system safety through BC provenance and policy enforcement through smart contracts.

HLF is a consensus-based network that the Department of Defense (DoD) can control and has no "Proof of Work" protocol, which is a wasteful use of computer resources. HLF uses channels to control who can see what data and through consensus; the DoD can control what is allowed to be put on the BC ledger. Such technologies can not only be used in Navy supply and logistics to streamline and improve effectiveness in terms of how workflow can be improved to provide more rapid and secure distribution of material and two-way financial transactions but can also be used on data transactions such as datasets requested by data scientists. Data scientists have long recognized that obtaining "clean data" and the permission to use it has been hampered by administrative friction, which can be caused by data owner's requirements, trust issues from generated data source transactions, and other administrative processes.

The benefits of BC technology described in this paper support system safety in terms of providing objective quality evidence about data integrity, as well as test and evaluation teams in terms of data management control. We believe elements of BC, such as smart contracts, could contribute to all acquisition groups involved. We will discuss our previous logistics use case as well as new use cases specifically for software safety.

The Hyperledger Fabric Blockchain Solution

HLF provides proof of transaction where trust is implemented through distributed consensus and not centralized policy enforcement. The specific version of BC we used is HLF, which is open-source from the Linux Foundation. HLF is a permissioned, distributed ledger that works on the consensus model that is an integral component of the "trust system" in the BC. Essentially, the Fabric environment provides the "common logging" and service management components on the platform, and the containerized infrastructure allows developers to build a BC network where data is recorded on distributed ledgers where the data written can be trusted, and transactions are immutable and tamper-proof. Smart contracts can embed legal knowledge, laws, and regulations, and enforce Navy data policy. BC/HLF can also provide "provenance" of an item, such as food or a part, and trace back to the source of that part or food item in case of contamination or counterfeit/defective parts as well as other times such as blocks of data in support of AI.

BC can be used for cyber currency such as bitcoin; cyber currency is not a part of this study, and a semiprivate BC in support of data integrity needs a specific set of BC features other than Everledger or Ethereum, which uses an inefficient way to verify blocks called Proof of Work (PoW) instead of the more efficient consensus algorithm such as Proof of Stake.

With our previous research questions—Could BC simplify and enable access and identity management for the Navy supply and logistics systems in a cost-effective manner to reduce this friction? How could BC improve Navy logistics to the last tactical mile?—we demonstrated the feasibility in previous research of using IBM and Oracle versions of HLF to track assets such as food items. Tracking and moving assets could be applied to data assets and adapted for software safety use because in both cases we care about the integrity of the data generated. There have been planned pilot projects in the DoD, usually supply chain scenarios (Simerly & Keenaghan, 2019).

Although HLF is a Linux open-source project, several software companies have adapted HLF as its core BC enterprise solutions and have added additional value through add-ons, cloud support, and company expertise that goes beyond the plain vanilla HLF. This is common with

open-source products as you pay for more capability and support. We compared to enterprise versions of HLF, the IBM, and the Oracle HLF BC platforms and evaluated their ability to maintain an efficient, streamlined, and accurate ledger of all shipment transactions during transportation. Additionally, the team developed a ledger serialization function in the smart contracts for synchronized connection on ships and bases to the HLF Framework. The characteristics of enterprise BCs include

- Permissioned architecture
- Highly modular
- Pluggable consensus
- Open smart contract model—flexibility to implement any desired solution model.
- Low latency of finality/confirmation
- Flexible approach to data privacy: data isolation using “channels,” or share private data on a need-to-know basis using private data collections
- Multilanguage smart contract support: Go, Java, JavaScript
- Designed for continuous operations, including rolling upgrades and asymmetric version support
- Governance and versioning of smart contracts
- Flexible endorsement model for achieving consensus across required organizations
- Queryable data (key-based queries and JSON queries)
- Uses X.509 public key infrastructure (PKI), which is quite familiar to the DoD for a signed data structure that binds a public key to a person, computer, or organization. Certificates are issued by certification authorities (CAs)
- Cloud support and SaaS (Software as a Service)

Figure 1 is an example of a very simple BC ordering network. A1, A2, and A3 are different “off-chain” applications that could be on IoT devices or web browsers on computers or smartphones. These applications connect the on-chain world with the BC network/database. These client applications represent the “last mile” and could include legacy programs pre-BC. The blue-shaded background represents the BC logical infrastructure layer—not whatever physical layer infrastructures might be used, such as satellite or fiberoptics. O4 is an ordering service. Network configuration (NC4) gives administrative rights to organizations R1 and R4. At the network level, Certificate Authority CA4 (DoD certs can be used) is used to dispense identities to the administrators and network nodes of the R1 and R4 organizations. Certification authorities CA1 and CA4 provide entity validation, as well as other CAs shown in the diagram. In this example, there are two consortiums (common interest parties), represented by R1 and R4 entities who set network configuration policies, seen CC1 and CC4 which set up channels. Channels are ways to decide who gets to see what ledgers. There are three peers: P1, P2, and P3. On the left, P1 has S5, which is a smart contract that provides the rules for the ledger L1. Only those who have access to Channel 1 (C1) have access to the ledger L1. You see that if you have access to A1 or A2 you have access to C1, but the A2 application has access to both C1 and C2 and, therefore, access to ledgers L1 and L2, which is set by configuration control (CCL).

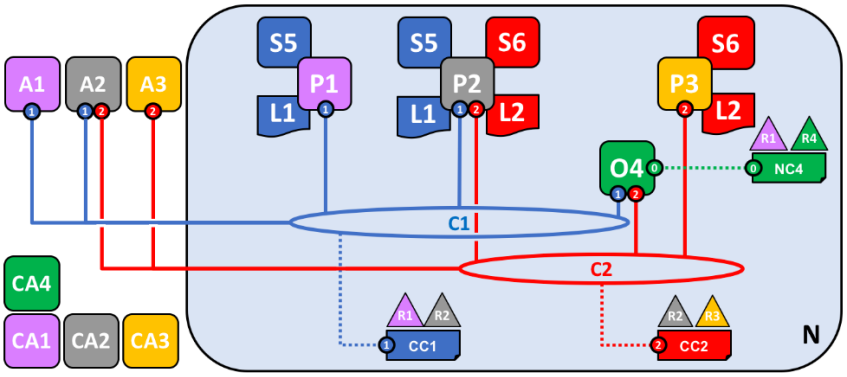


Figure 1. Generic HLF BC Network

Methodology

Our methodology involves two sets of use cases. The first set (original cases) were used in our previous research in Navy logistics, which we believe can also demonstrate BC use for system safety if modified, as both sets of use cases track assets—one tracks food items and the other tracks datasets as assets. The key for repurposing a supply chain for use in software safety support is through the addition of off-chain application programming interface (API), such as Representational State Transfer (REST, or many others), which provides an interface between the BC and the outside world and to what is called "the last mile," which in most of our use cases is a web client. In our first set of use cases (the original use cases) we built two demos (Oracle and IBM cloud versions) illustrating the Navy logistics/supply chain. We demonstrated how BC can document and authenticate transactions along the supply chain, which would be similar to a data supply chain used for data system safety. We worked with both Oracle and IBM enterprise BCs to demonstrate the first set of use cases. In a work in progress, we have an additional set of use cases (labeled new cases) specifically for use with system safety using the open-source version of HLF (https://www.hyperledger.org/).

Blockchain Use Case Examples for the Navy Logistics/Supply Chain

Working with our Navy sponsor (Navy Logistics N4) we looked at three general use cases to apply BC technology using both cloud versions of IBM and Oracle BC platforms: (1) financial and inventory transaction audit trails; (2) serial number tracking, and (3) maintenance log integrity. Maintenance log integrity involves the same issues as AI dataset integrity. The three examples are

- **Original Case 1:** Financial and inventory transaction audit trials. An investigatory inventory and financial transactions via audit trails can be a costly and timely process, and the audit trails could encompass different systems throughout a vast network in such an organization as the Navy. The questions to be answered might include what, where, and who—where a distributed ledger would be able to track "what" through immutable data blocks that make up the ledger. One of the BC strengths is identity verification and management, which would be able to verify and track the "who" in any financial and inventory transactions on the BC.
- **Original Case 2:** Serial number tracking/BC tracking can also be applied to the tracking of specific items in the supply chain, such as serial numbers. Also, the tracking could include a visual identification of the item by an individual, which would automatically be identified as a trusted agent to make that verification along with the where and the when.
- **Original Case 3:** Maintenance log integrity/maintenance repairs—such as on naval aircraft, ground, or ship systems—typically generate data on various transactional databases, which in turn may be sourced to other databases or repositories such as data warehouses'

Enterprise Resource Planning (ERP) systems. Our past research on aviation and ground maintenance systems databases shows that there are errors in the databases, and often information is not updated. At the tactical and operational levels, this could have an impact on the effective efforts to ensure maximum mission readiness. Smart contracts, which are integral to HLF, are code that can check, enforce, or flag bad data. Certainly, relational databases can have triggers to check for illogical data entries, but it isn't always being done, and typically several databases and sources may be involved in a maintenance information system to make such error checking costly or not practical. While some minor errors may be acceptable in transactional databases, these errors could have an impact on data analysis and ML/AI if the data in these systems are used as training datasets. BC could use smart contracts to flag errors over a diverse set of data sources and provide basic provenance.

Blockchain Use Case Examples for System Safety

In our second set of use cases, we specifically address three software system safety use cases applied to the open-source HLF:

- **New Case 1:** A researcher/data scientist needs to manage data or training sets for research or ML to process text or binaries (images, RFI signals), structured and unstructured.
- **New Case 2:** A data scientist needs to derive metrics on a dataset but is not allowed to see raw data.
- **New Case 3:** BC is used as a database for relatively small source code.

Figure 2 is a simplified HLF BC network that could support our three scenarios for software safety in the blue background square on the right (see https://www.hyperledger.org/). This is the BC. This BC is supported by a physical network that could be cloud-based and supported by the internet. The "off-chain" applications, IoT, and storage are shown outside of the square. These are applications developed in a normal way and not a new technology. The applications use standard APIs such as REST to interface between the user, databases, and the outside world to connect to the BC. They are called off-chain because while they interface with the BC, they are not part of the BC. From left to right are the identify certificates--CAs such as CA1, CA2, CA3 in our example to identify those who have access. BC is good at leveraging existing technologies, and CA is old technology using X.509 Public Key Infrastructure (PKI), which used to encrypt and sign email. A1, A2, A3, and so on are off-chain client applications that have access to various ledgers (our database) which are controlled through CC1 and CC2 (CCL), which sets up channels and their access. P1 and P2 are peer nodes that in the example host ledgers L1 and L2 for P1, and L3 for P2. Each ledger is supported by smart contracts (S5, S6, S7) that determine the business rules and logic of how the ledger is to be written and who can write on it. C1 and C2 are channels to determine what applications or entities are allowed to see what ledger, which makes Hyperledger very powerful as you can control who sees and changes what—such as Navy personnel and contractors having access to different data.

Off-chain A1 is an application that administers access to the repository and writes to the ledger, which records the metadata in each dataset and provides a digital signature/hash value. CCL provides access to Channels 1 and 2 and, as shown, access to all ledgers. For structured data in the repository (maybe more than the one shown in the diagram), AI would post/write the metadata of a dataset of interest including, if practical, all of the data fields, DTG, and record a hash value or signature. This would be entered either in L1, L2, L3, or other ledgers created. It is not practical to record/post large datasets on a BC ledger, but metadata and pointer/anchors to the data could be provided through URLs. It is possible that through the administrator interfacing with a peer node, the BC could store some small datasets through CouchDB, which would provide the current information/state of an asset such as a dataset.

New Case 1: Figure 2 shows application A2, which could be a customer/client such as a data scientist that is interested in datasets or training sets for an AI project. This customer per the diagram (set up by CC) has access to Channels 1 and 2, which means he can view Ledgers 1 through 3, which would be information about various datasets that can be accessed. In one scenario, the person using A2, the web application, for example, could search for a specific dataset or topic and then request that dataset through the application, which would check the smart contract—let's say for L2—to see if the system allows read access to the repository. Existing off-chain software would complete the task and send an anchor or link (URL) to retrieve that dataset. The customer could later check back and see if the data have changed/been tampered with, or if the data were given to another user. Also, the client would be provided the provenance and metadata and even points of contact, including subject matter experts and the owner of the data. The client can check to see if the dataset has changed and who changed it, since any changes to the repository would be recorded in the appropriate ledger as to who, when, and what. Smart contracts could also provide some prefiltering through smart contracts to reduce unintentional errors. In the past, this has been done pre-analysis but by using smart contracts this would only need to be done once and not by each researcher or customer. This AI system safety idea is similar to the IBM concept (Sarpatwar et al., 2019), where the authors sought a trusted AI environment through provenance with a BC library exposed by REST or Python APIs that provided support for "immutable recording of the AI process, querying for traceability and audit, fair value attribution, etc." We take it a further step to suggest that BC can be part of a smart repository solution that allows clients to search and find trusted datasets and safeguard them. A variation of this use case is a federated learning (FL) scenario that uses a collaborative ML technique whereby the devices collectively train and update a shared ML model while preserving their datasets. Even in a trusted military network using a private BC, some devices on the edge may prove untrustworthy, and ur Rehman et al. (2020) propose a reputation-aware FL that enables trust through BC consensus and trust algorithms through BC smart contracts.

New Case 2: A user wants to compile metrics but is not allowed access to the raw data because of security or cross-domain restrictions. Lampropoulos et al. (2019) proposed a similar scenario, where one Telco A holds private datasets and internally processes a data request by another Telco B, and Telco A only returns the results to Telco B and not the raw sensitive data. The whole process is performed with transparency, ensuring the quality of the results and the privacy of the processed data. A3 in Figure 2 is an application that only has access to Channel 2. The user then picks the dataset to use and looks at the metadata and fields; then the smart contract (S7) executes the query through A1 and post the results in the ledger L3. This use case could also be used for a cross-domain solution setting up rules when a user could have access to a different domain, the raw data, or just the results.

New Case 3: Our last scenario is the data are not stored off-chain but in the BC itself. HLF has the option of using CouchDB that can use standard JSON queries to get the "World" or current state of an asset (like a dataset). Perhaps this use case would apply to IoT devices where you want real-time data from sensors but still want to ensure software safety. The data would be immutable but replicated throughout the network.

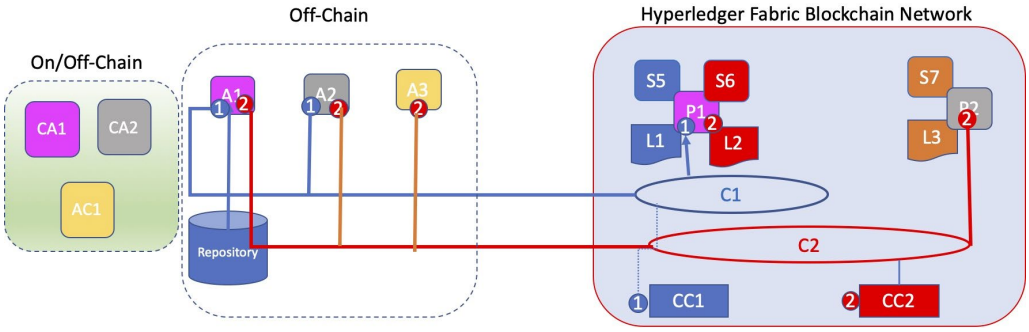


Figure 2. HLF Scenarios

Figure 3 summarizes the flow in our simple scenarios. First, the "customer"—a data scientist or developer—wants to access data such as for training sets in ML, or a developer wants access to code. The customer wants to find the right data quickly, know who owns it, and know that it can be reasonably trusted. In our example, this data resides in a repository that may include both structured data (relational databases) and semistructured and unstructured data such as in the form of .JSON files, text, or graphics. The customer starts a request for the data, and an answer comes back with the metadata, data fields, a date–time group, and a hash value of the set. This information is in a ledger in addition to an encrypted link to access the dataset. The customer can also see the complete history of changes to the data and can verify that the training set, data, or code has not been tampered with through the hash code both in the metadata and the ledger on the BC. Only those authorized can add to the chain, and it is immutable.

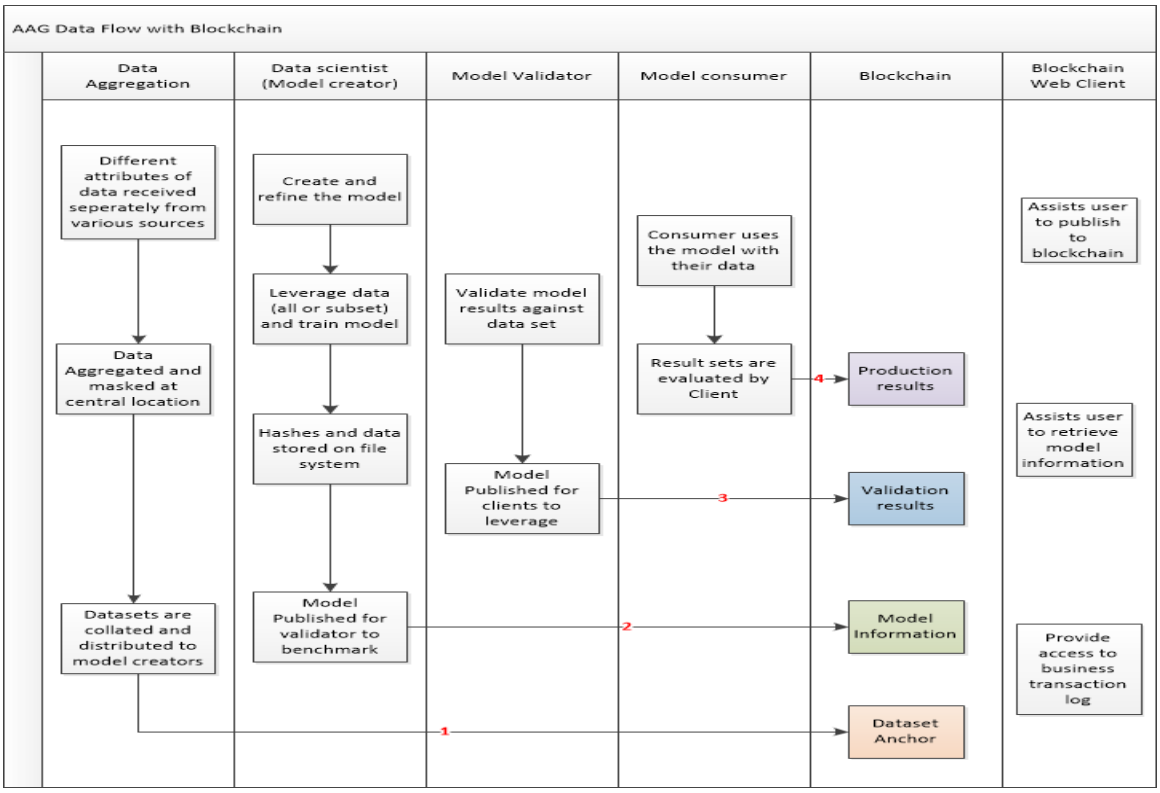


Figure 3. Summary of New Use Cases (adapted from Oracle diagram)

Use Cases Using Three Hyperledger Fabric Versions

We discuss our results using the IBM, Oracle, and Linux Foundation versions of HLF and their application to system safety scenarios. Figure 4 provides a simplistic view of the system safety scenario where the data scientist is looking for training sets or related data.

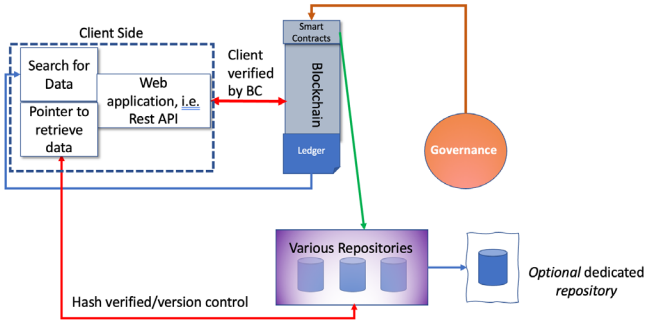
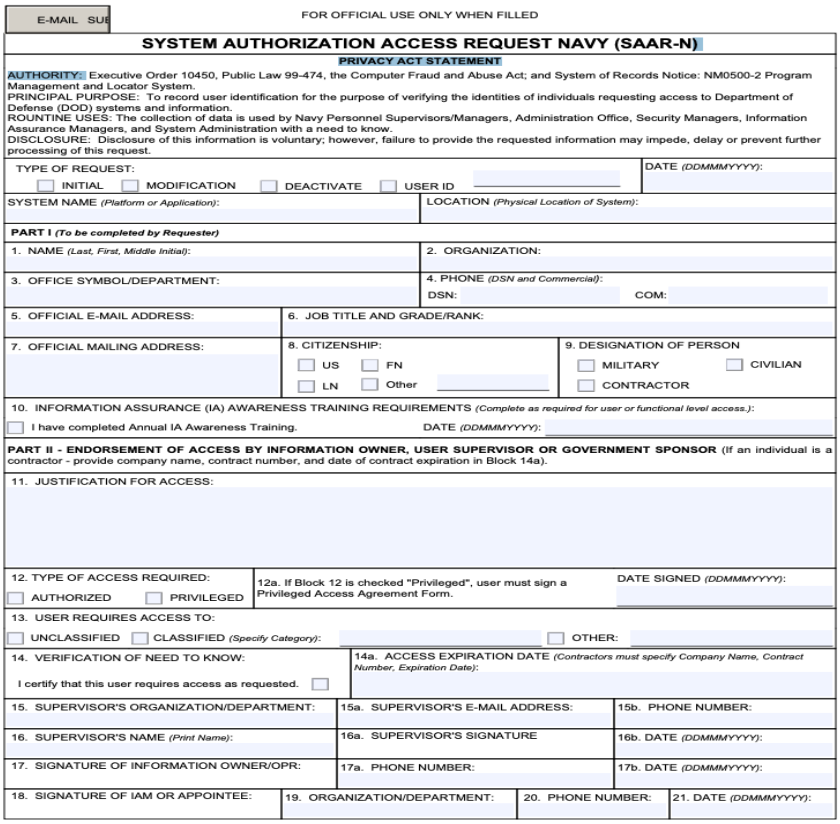


Figure 1. Data Scientist Use Case Example and Smart Repositories

The data scientist (the client) uses a web browser, enabled by Rest API or other development interfaces, and searches for a dataset or training set through the BC which, through certificates (x.509) and smart contracts, knows who the client is. Based on governance, the BC and smart contract will decide if that data scientist has the authority to retrieve the data. If so, the client will be sent a link to access the repository or even an IoT device or a BC repository with frequently used datasets. Through x.509 certificates, which will have the identity verified by the BC, as well as the sending to the client hash value making sure the dataset hasn't been tampered with. The smart contract may do some initial cleaning up and filtering of the data. What normally takes months to get the data may only take a day and comes with assurance that the data had not been tampered with through an immutable BC. Such forms as in Figure 5 or other members of understanding could be eliminated.



E-MAIL SUB

FOR OFFICIAL USE ONLY WHEN FILLED

SYSTEM AUTHORIZATION ACCESS REQUEST NAVY (SAAR-N)

PRIVACY ACT STATEMENT

AUTHORITY: Executive Order 10450, Public Law 99-474, the Computer Fraud and Abuse Act; and System of Records Notice: NM0500-2 Program Management and Locator System.
PRINCIPAL PURPOSE: To record user identification for the purpose of verifying the identities of individuals requesting access to Department of Defense (DOD) systems and information.
ROUNTINE USES: The collection of data is used by Navy Personnel Supervisors/Managers, Administration Office, Security Managers, Information Assurance Managers, and System Administration with a need to know.
DISCLOSURE: Disclosure of this information is voluntary; however, failure to provide the requested information may impede, delay or prevent further processing of this request.

TYPE OF REQUEST: ☐ INITIAL ☐ MODIFICATION ☐ DEACTIVATE ☐ USER ID _____ | DATE (DDMMMYYYY):

SYSTEM NAME (Platform or Application): | LOCATION (Physical Location of System):

PART I *(To be completed by Requester)*

1. NAME (Last, First, Middle Initial): | 2. ORGANIZATION:

3. OFFICE SYMBOL/DEPARTMENT: | 4. PHONE (DSN and Commercial): DSN: COM:

5. OFFICIAL E-MAIL ADDRESS: | 6. JOB TITLE AND GRADE/RANK:

7. OFFICIAL MAILING ADDRESS: | 8. CITIZENSHIP: ☐ US ☐ FN ☐ LN ☐ Other _____ | 9. DESIGNATION OF PERSON ☐ MILITARY ☐ CIVILIAN ☐ CONTRACTOR

10. INFORMATION ASSURANCE (IA) AWARENESS TRAINING REQUIREMENTS (Complete as required for user or functional level access.):
☐ I have completed Annual IA Awareness Training. DATE (DDMMMYYYY):

PART II - ENDORSEMENT OF ACCESS BY INFORMATION OWNER, USER SUPERVISOR OR GOVERNMENT SPONSOR (If an individual is a contractor - provide company name, contract number, and date of contract expiration in Block 14a).

11. JUSTIFICATION FOR ACCESS:

12. TYPE OF ACCESS REQUIRED: ☐ AUTHORIZED ☐ PRIVILEGED | 12a. If Block 12 is checked "Privileged", user must sign a Privileged Access Agreement Form. | DATE SIGNED (DDMMMYYYY):

13. USER REQUIRES ACCESS TO: ☐ UNCLASSIFIED ☐ CLASSIFIED (Specify Category): _____ ☐ OTHER:

14. VERIFICATION OF NEED TO KNOW: I certify that this user requires access as requested. ☐ | 14a. ACCESS EXPIRATION DATE (Contractors must specify Company Name, Contract Number, Expiration Date):

15. SUPERVISOR'S ORGANIZATION/DEPARTMENT: | 15a. SUPERVISOR'S E-MAIL ADDRESS: | 15b. PHONE NUMBER:

16. SUPERVISOR'S NAME (Print Name): | 16a. SUPERVISOR'S SIGNATURE | 16b. DATE (DDMMMYYYY):

17. SIGNATURE OF INFORMATION OWNER/OPR: | 17a. PHONE NUMBER: | 17b. DATE (DDMMMYYYY):

18. SIGNATURE OF IAM OR APPOINTEE: | 19. ORGANIZATION/DEPARTMENT: | 20. PHONE NUMBER: | 21. DATE (DDMMMYYYY):

Figure 2. SAAR-N Form

Successful Applications of Blockchain for Naval Supply Chain Tracking

As discussed, our previous research investigated how BC could simplify and enable access and identity management for the Navy supply and logistics systems in a cost-effective manner to reduce administrative friction and how BC could improve Navy logistics to the last tactical mile. In our scenario, the first destination transportation (FDT) refers to the movement and cost of moving shipments from free on board (FOB) points of origin to the location at which the shipment is first received for use or storage. As naval regulations apply, the first checkpoint of where a shipment is received, whether within the United States (CONUS) or outside (OCONUS), begins with a supplier outside of the DoD supply system or industrial activity that creates the shipment. The labor and transportation charges, including freight drayage, cartage, port handling, and other in-transit costs, are processed at the FDT. Freight cartage refers to any inland transit of cargo between locations, which serve as the “checkpoints” in the BC network. When a location is assigned responsibility for “cartage of consignments” to land-based activities, ships, or other transport units, the charges of transportation are given to the location of assigned responsibility, which acts as a peer node checkpoint in the network. At this point, the initial entry in the ledger may be created and committed by the peer node belonging to the FDT and the orderers. It is important to note that FDT does not only include shipments of equipment but also the initial transportation of Navy-owned materials that are provided to a contractor for research. This indicates that the charges of a shipment from a contractor’s facility to its final destination point are paid by the government. However, to maintain the legitimacy of a decentralized ledger in this research study, the network for which the ledger is maintained consists of only contractors, supply facilities, and the final base destinations. Essentially, tracking responsibility is passed down from supplier to checkpoint. The checkpoint managers responsible for the charges in a shipment delivery may create and commit the transaction over the BC network,

and the next checkpoint manager may agree or disagree about the condition and extraneous details of the shipment that the previous manager signed. Currently, the DON uses service-wide transport (SWT) as a clearinghouse, which is a centralized operations and maintenance manager created to provide transportation funds for naval shipments and mail. Since naval cargo and the movement of mail to bases is not a responsibility of a destination location, the SWT was created to pay for the movement of material, such as aircraft engines, mission module packages, catapult and arresting gear, propellers, shafts, civil engineering support equipment, safety equipment, drones, overseas mail, and Navy Exchange Service Command (NEXCOM) merchandise shipped from within the United States to international locations.

For disconnected operations, to maintain an accurate ledger with the consensus algorithm, the peer nodes must be connected to the Fabric environment unless the peer node decides to save the ledger as a .JSON file and re-upload the ledger as a .CSV file once back online. The ledger is automatically updated after the node reconnects following disconnections due to shipboard communications. The Fabric environment will make BC technology a more viable option for all naval transportation activities.

The Navy requires a multifunctional and secure platform that enables personnel to track multiple shipments from production facilities to bases and a secure ledger of inventory that can only be modified with either an undisputed consensus or access to the smart contract. Once a peer node administrator or user in the network has access to their smart contract, they can modify the transaction protocol that occurs on transactions in the network. However, the network will not instantiate a new version until there is an agreement with the channel creator or the majority of the channel members.

In this simplified logistics BC network, the smart contract contains six methods that carry out the protocol for each transaction on the ledger: foodAssetExists, createFoodAsset, readFoodAsset, updateFoodAsset, trackFoodAsset, and deleteFoodAsset. The method of using names indicates that each shipment is checked to verify if it already exists at a location denoted by a string. After checking for duplication, the asset is created in the ledger using a key-value pair, such as “001: a shipment of supplies.” Once the asset is created, it is always a good practice to read the asset’s details into the ledger so that users further down the network have a detailed understanding of what a package is supposed to contain. Also, if a shipment is changed—say, a package is redirected to a base that requires supplies urgently—the shipment’s location is updated within the ledger and deleted once the shipment arrives.

A multifunctional and secure platform that enables personnel to track multiple shipments from production facilities to bases or ships in transactions involving money, items, material, and history should be trusted, transparent, and traceable back to the origin of the item. These transactions involving information, money, or physical items such as food or parts usually involve the enforcement of policy, technical, or legal requirements that require the enforcement of business rules. BC can maintain a secure ledger of inventory (or transactions involving data or information) that can only be modified with either an undisputed consensus or access to the smart contract, which can enforce business rules and flag “violations.” Once a peer node administrator or user in the network has access to their smart contract, they can modify the transaction protocol that occurs on transactions in the network. However, the network will not instantiate a new version until there’s an agreement with the channel creator or the majority of the channel members.

Based on the above process, we showed how a food or item tracking scenario would work using both IBM and Oracle cloud versions of HLF (see Figure 6). In these BC networks we set up, the smart contracts contain six methods that carry out the protocol for each transaction on the ledger: foodAssetExists, createFoodAsset, readFoodAsset, updateFoodAsset,

trackFoodAsset, and deleteFoodAsset. In our food/item tracking scenario, the method of using names indicates that each shipment is checked to verify if it already exists at a location denoted by a string. After checking for duplication, the asset is created in the ledger using a key-value pair, such as “001: a shipment of supplies.” Once the asset is created, it is always a good practice to read the asset’s details into the ledger, so that users further down the network have a detailed understanding of what a package is supposed to contain. Also, if a shipment is changed, say, a package is redirected to a base that requires supplies urgently, the shipment’s location is updated within the ledger and deleted once the shipment arrives.

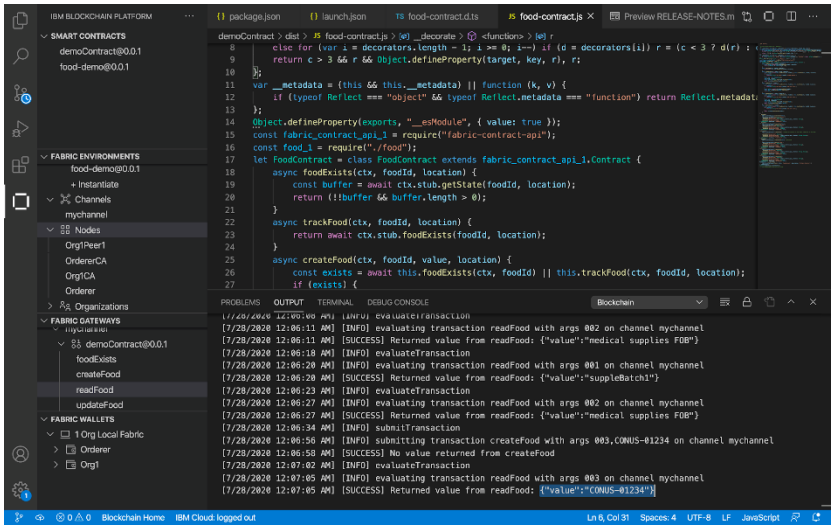


Figure 3. Sample Ledger of Shipments That Are Added and Updated (Contents/Location)

Blockchain Use Case Examples for the Navy Logistics/Supply Chain

Using IBM BC Platform™: To use the IBM BC Platform, users are required to install four vital components: (1) the Virtual Studio Code environment, (2) Node.js, (3) Docker, and (4) Kubernetes. The Virtual Studio Code environment is the offline integrated development environment (IDE), where developers create smart contracts using the open-source programming language Typescript, which was developed by Microsoft.

Smart contracts serve as the fundamental basis of all enterprise BCs because they give certified users the ability to create new transactions and assets, as well as other functions specific to a project. In this project, the team’s main goal was to create a consensus network that has the power to create food shipment assets, update or delete them from the ledger when required, and track their location using the “foodId” string, which may be replaced by radio-frequency identification (RFID).

The HLF (from the Linux Foundation) is the basis of both IBM and Oracle platforms. Its components are created in a Kubernetes cluster usually within the IBM Cloud. A Kubernetes cluster contains a set of working machines (nodes) that run containerized applications. The nodes within the cluster host the components of the application workload. Within the cluster, the control plane manages the nodes and workloads that run across multiple machines, as shown in Figure 7:

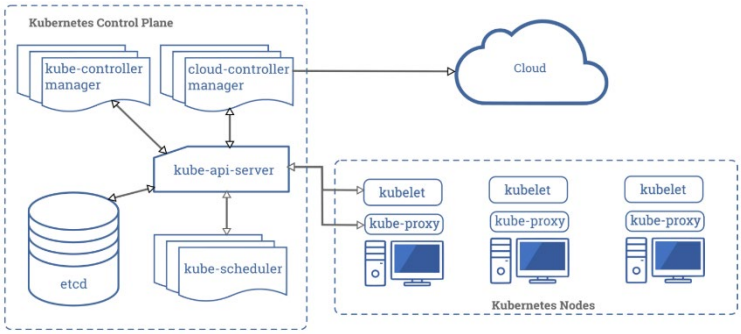


Figure 4. Visual Representation of the Interaction Between Kubernetes and Cloud

Figure 8 illustrates the ordering service. When the Fabric environment is running, you can create the ordering service. The ordering service is a group of orderers that accepts approved transactions endorsed by the peer nodes based on the smart contracts and organizes the transactions in the appropriate order in the ledger blocks based on the consensus algorithm.

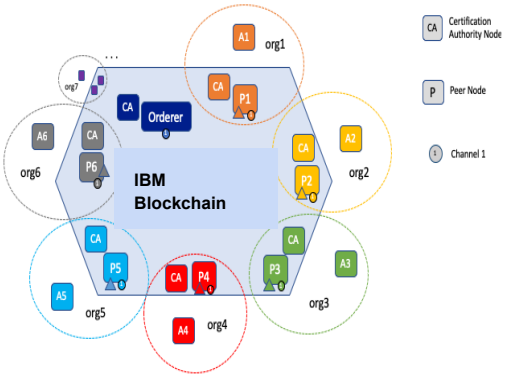


Figure 5. Visual Representation of the Integration of Security and Nodes in a BC Channel

The peer nodes host ledgers and smart contracts—the backbone of the BC network. The smart contract—the transaction protocol—automatically executes, controls, and documents transactions or events occurring on the network.

Like all BC frameworks, the network's integrity is upheld by the consensus algorithm. Each node in the network reviews the entire BC and checks that all previous blocks are valid so that a new transaction may be initiated into the network. However, alternatively, in a permissionless public BC, the consensus algorithm is replaced by the PoW, which creates a hash system of all of the transactions.

In a PoW system, miners constantly attempt to solve the algorithm so that they may mine new blocks and be the first to extend their BC. HLF doesn't use the wasteful PoW but uses a system closer to the "Proof of Stake" as a consensus mechanism. Essentially, decisions are authorized by users who are permitted to join the system and specific channel, as not everyone can join the network. Unlike PoW, computational power is not required, since there are no puzzles needed to obtain "currency." In a "Proof of Stake" system, "validators" are discouraged from creating faulty empty blocks because they have the motivation to incorporate a maximum number of transactions for gains.

To ensure security, the hash must be solved by all the peer nodes in the network so that the new transactions may be approved for the network. While this alternate approach is viable, it

is also time-consuming because ensuring that the ledger is tamper-free requires each ledger copy in the nodes to be changed and hashes to be solved.

Developers should install Node.js and Docker unless the developer exports both items into a .JSON file and re-uploads both the files onto the peer nodes as a .CSV file. Docker serves as an OS-level platform to package containers and bundled software, libraries, and configuration files.

Figure 9 shows that using well-defined channels within the software, these containers communicate with each other to allow the user to connect to the Fabric environment and add to or change the ledger. Finally, the Kubernetes system, which was designed by Google and maintained by the Cloud Native Computing Foundation, is the main system that allows the IBM BC Platform to package, install, deploy, and manage the multiple peer nodes in the platform.

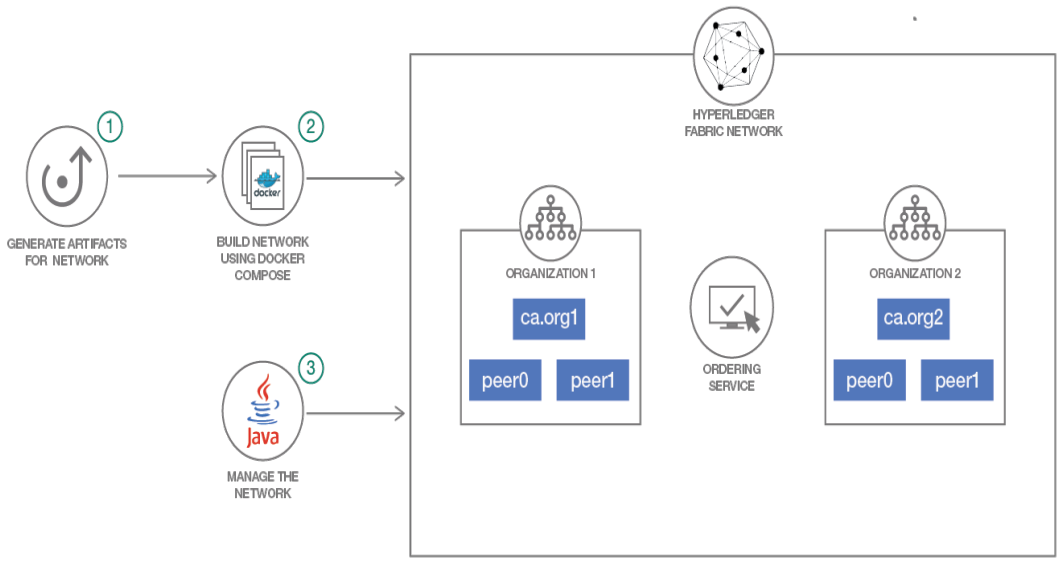


Figure 6. Visual Representation of the Interaction Between the External Software and the HLF Environment

Figure 10 provides an overview of how you would manage the offchain (UI[2]) and the actual BC network consisting of three Fabric components: CA(4), the peer nodes, and the ordering service. Compare Figure 8 to Figures 1 and 2.

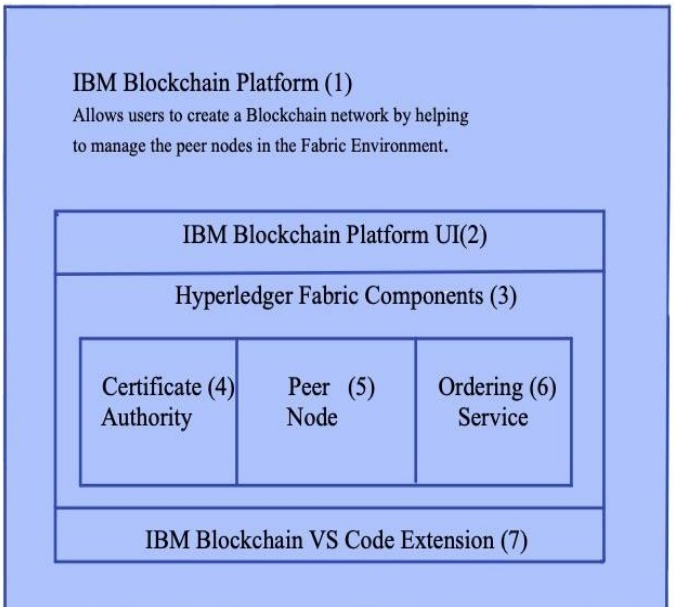


Figure 10: High-Level Representation of IBM Blockchain Platform Architecture

Using the Oracle Blockchain Platform: On both Oracle and IBM platforms, we were able to set up a BC network with peer nodes (stakeholders) with smart contracts that set up the rules for transferring and tracking assets such as food items discussed previously. More work needs to be done on enhancing the network to accurately represent this aspect of the supply chain.

The team set up the network using an Oracle cloud with four peer nodes set up over a single channel and used Oracle Identify Management for role-based access. Separate roles are required for adding users to a role with BC provisioning entitlement, which requires tenancy admin. Additionally, the cloud platform was used instead of the software package due to the amount of storage memory required to host the software appliance VM packages on a local computer. However, the fundamental concepts of using an HLF environment and consensus algorithm remained the same for both platforms to build a BC network.

Oracle Blockchain Platform also provides wizards to simplify joining multiple instances to the network, creating new channels, and deploying chaincodes. Implementation of smart contracts is through Typescript (see Figure 11). These and other DevOps functions are also available via extensive REST APIs for off-chain applications to interface the BC network.

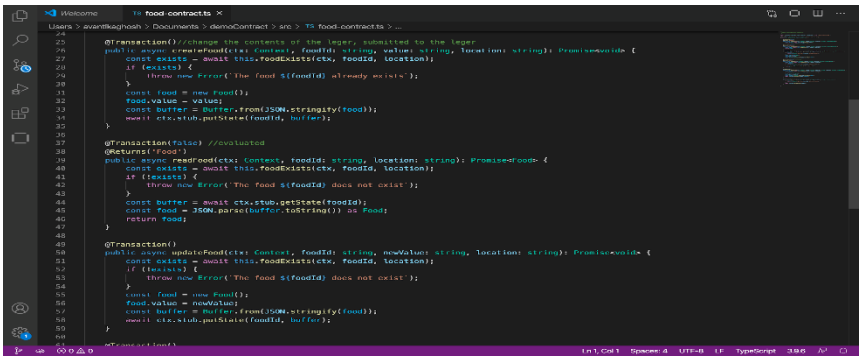


Figure 11. Sample Smart Contract for Tracking Food Shipments (Language: Typescript).

Oracle offers both a managed Cloud version (Oracle Blockchain Platform) of OBP (Blockchain-as-a-Service) and a customer-managed OBP Enterprise Edition for on-premise (or 3rd party cloud) deployment, and nodes can be deployed using both for a hybrid network deployment (see Figure 12). The Cloud SaaS version was used for this project. To access this platform, users must log in with authenticated credentials in Oracle Cloud Infrastructure. Once logged in, users can provision an instance, which comes with a default channel and participant nodes, along with "orderers" that are responsible for maintaining the order of the ledger. An operations Console is provided, and users are not required to download any external software to work with the platform, other than an Integrated Development Environment (i.e., Visual Studio Code) to develop the chaincode and the REST API Testing tool, such as Postman and/or HLF Software Development Kit, which is downloadable from the OBP Console under the Developer Tools tab.

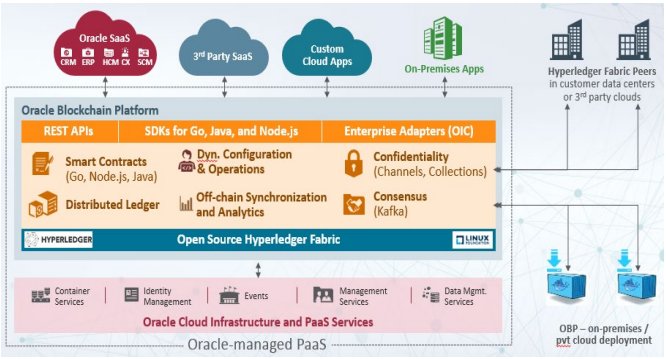


Figure 7. Oracle Blockchain Platform Cloud Service Architecture

The Oracle Blockchain Platform (see Figure 10) comes with an API Gateway that supports REST API so that developers can invoke a transaction, invoke a query, subscribe events with a registered callback, and view the status of a transaction within the ledger as well as a set of DevOps REST APIs for administration, configuration, and monitoring tasks.[1]

Current Use Cases Using Linux version of HLF for System Safety

While both IBM and Oracle are HLF-based, their complete solutions use their respective cloud services are enhanced by their specific products. For our new set of system safety use cases (a work in progress), we installed HLF on a Naval Postgraduate School virtual Red Hat server and installed HLF from the Linux open-source foundation, which provides all the needed images and tools to set up a BC. Unix tools include the Git client, CURL, and Docker with Docker-Compose without Kubernetes, which are key components to build the BC network in a rapid manner. This model suits the researcher who wants to study and test out the concepts before moving to production, at which point a vendor-supported option can better address the challenges. Typical enterprise BC platforms provide dashboards for BC management such as the status and health of the HLF network. In the case of the open-source version, no such tools are provided; instead, everything is done via command; thus one has to have a good idea of Unix command line tools and scripting languages like BASH. Both IBM and Oracle allow you to use an IDE to build the applications. All three platforms offer interfaces via APIs to programming languages like JavaScript, Java, Microsoft Visual Studio, and others. For most production instances, we think a cloud-based BC is usually the right way to go for maintainability, support, ease of use, and security.

[1] The team was given access to the Oracle Cloud Platform thanks to the NPS liaison relationship with the Oracle Blockchain team.

For the Linux Foundation version of HLF, the complete install includes commands to set up an HLF network, issue certificates, set up the ledger, create channels, install chaincode, and more. A sample BASH script is provided that goes over all these steps and can be customized for new projects such as for our three system safety use cases. The Docker container-based platform allows one to have several HLF projects to coexist. The test network is shown in Figure 13 with two organizations, R1 and R2. Organization R0 owns the ordering service (O) of channel C1. A copy of the ledger L1 is on all nodes. The root CA issues the certificates CA0, CA1, and CA2 for the three organizations.

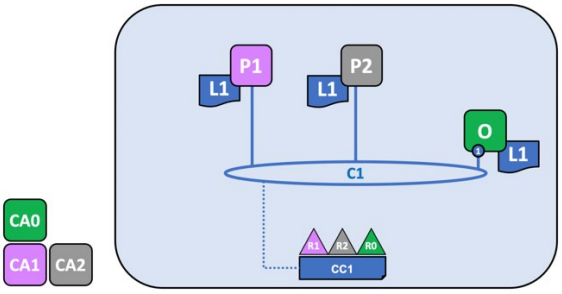


Figure 13. Initial Blockchain Network on Open-Source HLF

Use Case 1: First a channel is created, and member organizations are added into the channel. The ledger contains the needed URLs to access the different binaries. Using APIs, depending on the requester, the chaincode will craft a unique response to be sent back. When the response is received, the URL and text are preprocessed; this happens in a middleware server outside the HLF, thus a custom webpage is created and served to the end user. This webpage has links to authenticated repositories (database back end). Once authenticated access is granted to the data scientist (data and training sets used in research and stored in a database), an encrypted anchor or URL may be sent to the data scientist to download the dataset.

Use Case 2: When a member organization needs to see a part of the ledger, then channels have to be created. Membership to a channel is restricted to a subgroup of the organizations. Using chaincode, the metrics part of the ledger is provided to members of the metrics channel. There exists another channel where members can see the raw data using queries—again dictated by the chaincode. Membership to the two channels is a different set of organizations. Similar development methods used in Use Case 1 apply to Use Case 2.

Use Case 3: In this scenario again, a channel is created for a certain member organization (not all). These members will be able to access the data in CouchDB via API queries. Chaincode will decide which source code (stored in the CouchDB) is provided as a returned result of the query. The database is replicated on every node, which might be an advantage on the edge and an I/O to IoT devices on the edge. Specific use cases for this capability haven't been developed.

Summary and Conclusions

From our work, the following lessons learned can be applied to protecting datasets such as training sets for AI:

1. Various versions of HLF will work adequately, but due to complexity, we recommend not using open-source but software vendors such as Oracle, IBM, Microsoft Azure BC, and others. BC is not a DoD core competency; therefore, contractor support is needed.

2. HLF or other BCs alone are not the entire solution, since BC is an enabling or general purpose technology (GPT)—so in itself, it is not a solution. You must use a BC protocol within an integrated network infrastructure that also provides for the last mile to bring the data to the user, and this is through APIs. We recommend, ceteris paribus, that you consider using the same company that runs your relational databases or ERP, as your team will be familiar with that architecture.

We used a qualitative methodology that included three general logistic use cases: (1) financial and inventory transaction audit trails, (2) serial number tracking, and (3) maintenance log integrity. These were used in consultation with the topic sponsor. We created simple scenarios where items were tracked through a BC network, and smart contracts would check for certain conditions that would simulate quality control and tracking. We selected two enterprise HLF platforms, Oracle and IBM, and evaluated them in terms of functionality, development ease, and security.

We found that both the IBM and Oracle BC platforms may be used to create a secure network of peer nodes and a consensus for the legitimacy of the shipment ledger, which can only be modified using smart contracts. A special concern with Navy logistics is the possibility of unreliable networks, especially from shore to ship. The BC protocol creates a multitude of copies of the blocks (the public ledger) and if connectivity is lost, the blocks will be updated once the network node communications are reestablished. Both IBM and Oracle BC platforms were accessed through the cloud, but the option is for the Navy to put either platform on its implementation of the cloud or servers.

There were differences between IBM and Oracle implementation of HLF, such as how the whole network infrastructure was implemented, user interfaces, the developer tools and application programming interfaces provided, and how the implementation would connect to the Navy's legacy systems to reach the last mile—such as on the ship. These were real value-added capabilities since HLF alone cannot make an enterprise BC system that supports the existing logistics information system.

BC technologies offer the potential to reduce costs and logistical friction by providing a trusted ledger in support of logistic transactions and processes. Errors can be reduced through smart contracts, as demonstrated in both IBM and Oracle BC platforms. BC tracks assets, and therefore, BC can track data assets just as well as a partial solution to software safety.

Intermittent Communications

The Navy primarily operates at sea, which means the communications infrastructure supporting the BC network may not always be available or reliable, or provide bandwidth. A significant concern when implementing BC technology in cargo shipments is its dependence on a continuous connection to the Fabric environment. However, HLF is a robust distributed database (ledger) that has many copies of itself.

The BC platform does require you to be connected to the Fabric environment at all times or to consistently re-upload the ledger to the peer nodes to have a constant accurate ledger. BC provides an update method that if a node is offline, it will have an update of its BC once reliable network is reestablished.

The Issue of Governance

Figure 4 showed a simple notational circle labeled "Governance," but this issue is far from simple and is the key to any implementation of BC in support of data. While a detailed discussion of governance is beyond the scope of this paper, Gaur and Gaur (2018) presented a variety of frameworks, some of which would apply to permissioned BC networks. Previous discussions of BC governance tended to be about public BCs supporting cyber currencies. They

noted that while BC is about decentralization, there will have to be some aspects of centralized governance—especially ones involving policy and legal aspects in the storage and use of data. For example, governance could include safeguards through smart contracts that could flag possible AI bias, especially ones used for human resources. Governance can consist of different layers, and one classification recognizes the different levels the data serves and classified as strategic, operational, and tactical governance. Since BC is decentralized by nature, the governance should be at the lowest level if diversity and flexibility are important. Ziolkowski et al. (2020) looked at governance that includes demand and data management, system architecture design and development, membership, and data ownership. Each one represents a possible off- or on-chain solution that involves technical and policy considerations—both of which may include smart contracts as solutions and resources (Feagan, 2020). System architecture design and development are not trivial tasks and are based to a great extent on governance and policies. To resolve this, IT network engineers must work as part of a consortium to determine the appropriate way to expose their peers to other organizations to receive transaction endorsement proposal/simulation requests while minimizing an attacker's ability to gain access to sensitive information stored in the simulating peer's database (Feagan, 2020). The level of rigor is ultimately determined by the policies derived by governance. Data accessibility is also a key, so governance should have policies that allow scientists working for the DoD to find data not through randomness but structure, without undue delay, and data that complies with software safety. BC supporting "smart repositories" may facilitate this goal. The default should be to allow our data scientists and analysts timely access to data unless there is a good reason not to. Our adversaries work for AI superiority, and withholding data from their researchers is something they avoid. We refer to unclassified and non-PII/medical data.

Findings

We demonstrated through IBM and Oracle examples that HLF could meet logistics/audit and security requirements through smart contracts and the inherent trust systems with embedded certificates. Data entry errors could be reduced through smart contracts, which is an inherent feature of HLF. We believe a consortium BC through HLF would be a way to go to be able to share information (through the ledger) with suppliers and other third parties but also have the capability not to share when appropriate. BC could add the capability for secure transactions through certificates and the immutability of the transactions on the BC. The additional capability of BC on Navy logistics and supply would be able to catch some data entry errors, to trace back to the source, and basically to better know the what, the who (verified), and the where of various transactions generated by the supply chain.

We found that both the IBM and Oracle BC platforms may be used to create a secure network of peer nodes or naval hotspots that can generate a consensus for the legitimacy of the shipment ledger, which can only be modified using smart contracts. Since a key component of both platforms is maintaining accuracy and security of the ledger, all users must consistently export and import the smart contracts and ledgers onto their respective peer nodes every time an update is made on the ledger or if the transaction protocol on the smart contract is changed. A special concern with Navy logistics is the possibility of unreliable networks, especially from shore to ship. The BC protocol creates a multitude of copies of the blocks (the public ledger), and if connectivity is lost, the blocks will be updated once the network node communications are reestablished. Both IBM and Oracle BC platforms were accessed through the cloud, but the option is for the Navy to put either platform on its implementation of the cloud or on servers.

There were differences between IBM and Oracle implementation of HLF—such as how the whole network infrastructure was implemented, user interfaces, the developer tools and application programming interfaces provided, and how the implementation would connect to the

Navy's legacy systems to reach the last mile, such as on the ship. These were real value-added capabilities, since HLF alone cannot make an enterprise BC system that supports the existing logistics information system.

We found a "consortium BC" with a BC consensus network to be the best fit for the use cases. A consortium allows both private and public users to use the BC while control is maintained by the private users (the Navy) through a consensus network, which means by the consensus of trusted Navy entities. This is contrasted by PoW BC networks used in cyber currency, which are inefficient and not appropriate for a government entity. BC technology has the potential for revolutionizing the logistics process by ensuring the quality and trustworthiness of logistical generated data as well as providing provenance of parts and food, but it is new and risky.

The team also compared the IBM and Oracle BC platforms on efficiency and maintainability of a ledger of shipments and discovered that it was easier to use the IBM platform to create and export smart contracts and ledger; however, in September 2021, Oracle will provide similar capabilities for developing and deploying smart contracts. The IBM platform required users to develop their smart contract on the Visual Studio Code environment, export the contract as a .JSON file, log in to the online BC network, and import the contract and ledger as a .CSV file using a converter.

The Oracle Blockchain Platform, on the other hand, allowed users greater flexibility to join ledgers more cohesively. The Oracle platform allowed users to log in to the Oracle cloud after they were approved by an administrator and used simple software like IDE and the Software Development Kit. Furthermore, the Oracle Blockchain Platform employed chaincode as a smart contract for transactional protocols in the network. A chaincode is written in either Java, Node.js, or Go and packaged into a ZIP file, which can be installed on the network. This is similar to how smart contracts are exported as .JSON files and uploaded on the IBM network as .CSV files. More specifically, chaincodes outline the structure of the ledger, initialize it, create updates (such as reading or updating entries), and respond to queries.

Should HLF be used for software safety for ML and AI development? BC is general purpose technology (GPT) like the Internet, so BC isn't a solution in and of itself, but it acts as an enabler that provides a trusted, distributed ledger that could be used for smart repositories and software safety. If other technologies are better, then why aren't they commonplace? BC isn't *the* solution but, along with off-chain technology, may be a technology that enhances existing business processes.

References

Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2017). *Synthesizing robust adversarial examples*.

Chauhan, G. (2018, September 14). *AI safety*. Towards Data Science. https://towardsdatascience.com/ai-safety-9aeb9ca42907#:~:text=AI%20Safety%20is%20collective%20termed,of%20real%2Dworld%20AI%20systems

Defense Standardization Program Office. (2012). *System safety* (MIL-STD 882E). Pentagon.

Eden, A. H., Moor, J. H., Soraker, J. H., & Steinhart, E. (2013). *Singularity hypotheses: A scientific and philosophical assessment*. Springer.

Everitt, T. (2018). *Towards safe artificial general intelligence* [Doctoral thesis, Australian National University]. https://www.tomeveritt.se/papers/2018-thesis.pdf

Feagan, L. (2020, April). *Hyperledger fabric myths and reality*. Object Computing. https://objectcomputing.com/resources/publications/sett/april-2020-hyperledger-fabric-myths-and-reality

Gaur, N., & Gaur, N. (2018). *Hands-on blockchain with Hyperledger: Building decentralized applications with Hyperledger fabric and composer* (1st ed.). Packt Publishing.

Jia, X., Hu, N., Yin, S., Zhao, Y., Zhang, C., & Cheng, X. (2020). A2 chain: A blockchain-based decentralized authentication scheme for 5G-enabled IoT. *Mobile Information Systems*, *2020*. https://doi.org/10.1155/2020/8889192

Lampropoulos, K., Georgakakos, G., & Ioannidis, S. (2019). Using blockchains to enable big data analysis of private information. *IEEE 24th International Workshop on Computer-Aided Modeling and Design of Communication Links and Networks*, *2019*, 1–6, https://doi.org/10.1109/CAMAD.2019.8858468

Roland, H., & Moriarty, B. (1990). *System safety engineering and management* (2nd ed.). Wiley.

Sarpatwar, K., Vaculin, R., Min, H., Su, G., Heath, T., Ganapavarapu, G., & Dillenberger, D. (2019). Towards enabling trusted artificial intelligence via blockchain. In S. Calo, B. Seraphin, & D. Verma (Eds.), *Policy-based autonomic data governance* (1st ed., pp. 137–153). Springer International Publishing. https://doi.org/10.1007/978-3-030-17277-0_8

Simerly, M. T., & Keenaghan, D. J. (2019). Blockchain for military logistics. *Army Sustainment*, *51*(4), 48–49.

ur Rehman, M., Salah, K., Damiani, E., & Svetinovic, D. (2020). Towards blockchain-based reputation-aware federated learning. *IEEE Conference on Computer Communications Workshops*, 183–188. https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9163027

Ziolkowski, R., Miscione, G., & Schwabe, G. (2020). Decision problems in blockchain governance: Old wine in new bottles or walking in someone else's shoes? *Journal of Management Information Systems*, *37*(2), 316–348. https://doi.org/10.1080/07421222.2020.1759974

Acquisition Research Program
Graduate School of Defense Management
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net