**Technical Report**
**1256**

# Data Trust Methodology: a Blockchain-Based Approach to Instrumenting Complex Systems

G.C. Augeri
B.J. Courville
C. Flood
J. Hallapy
F.P. Hunsberger
J.T. Miller
A.A. Prasov
A.R. Wright

05 August 2020

**Lincoln Laboratory**
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
*LEXINGTON, MASSACHUSETTS*

This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001.

This report is the result of studies performed at Lincoln Laboratory, a federally funded research and development center operated by Massachusetts Institute of Technology. This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

Massachusetts Institute of Technology
Lincoln Laboratory

# Data Trust Methodology: a Blockchain-Based Approach to Instrumenting Complex Systems

G.C. Augeri
F.P. Hunsberger
*Division 9*

B.J. Courville
J.T. Miller
A.A. Prasov
*Group 93*

A.R. Wright
*Group 99*

C. Flood
J. Hallapy
Formerly MIT Lincoln Laboratory

Technical Report 1256

05 August 2020



Lexington Massachusetts

This page intentionally left blank.

# TABLE OF CONTENTS

This page intentionally left blank.

# LIST OF ILLUSTRATIONS

This page intentionally left blank.

# 1. PROBLEM STATEMENT

Increased data sharing and interoperability has created challenges in maintaining a level of trust and confidence in Department of Defense (DoD) systems. As tightly-coupled, unique, static, and rigorously validated mission processing solutions have been supplemented with newer, more dynamic, and complex counterparts, mission effectiveness has been impacted. On the one hand, newer deeper processing with more diverse data inputs can offer resilience against overconfident decisions under rapidly changing conditions. On the other hand, the multitude of diverse methods for reaching a decision may be in apparent conflict and decrease decision confidence. This has sometimes manifested itself in the presentation of simultaneous, divergent information to high-level decision makers. In some important specific instances, this has caused the operators to be less efficient in determining the best course of action.

In this paper, we will describe an approach to more efficiently and effectively leverage new data sources and processing solutions, without requiring redesign of each algorithm or the system itself. We achieve this by instrumenting the processing chains with an enterprise blockchain framework. Once instrumented, we can collect, verify, and validate data processing chains by tracking data *provenance* using smart contracts to add dynamically calculated metadata to an immutable and distributed ledger. This non-invasive approach to *verification* and *validation* in data sharing environments has the power to improve decision confidence at larger scale than manual approaches, such as consulting individual developer subject matter experts to understand system behavior.

In this paper, we will present our study of the following:

1. Types of information (i.e., knowledge) that are supplied and leveraged by decision makers and operational contextualized data processes (Figure 1)


*Figure 1: Levels of information processing.*

2. Benefits to verifying data provenance, integrity, and validity within an operational processing chain
3. Our blockchain technology framework coupled with analytical techniques which leverage a verification and validation capability that could be deployed into existing DoD data-processing systems with insignificant performance and operational interference

This page intentionally left blank.

## 2. FRAMEWORK OVERVIEW

We begin by describing the meaning of data, information, and knowledge and provide our definition of decision support. Data, information, and knowledge are often used interchangeably when describing the merits of a system, but each one represents a distinct level of contextualization and processing. Data is the lowest form of this, representing analytic elements with little or no filtering, such as signal detections processed from a sensor focal plane. Data is used to derive information, by the addition of context, such as the types of targets whose signals are important to detect. Knowledge is information that means something in broader situational awareness or a decision, such as the types of detected target behavior in a scenario that signifies danger. (Amidon, 1997)

As an example, there have been a number of efforts that have moved 'data integration' in the forward direction and have demonstrated operational capability for situational awareness. These efforts illustrated the conversion of data from various sources, to information via various software applications, to knowledge at presentation to the operator or user (Figure 2). By applying levels of processing to create an enhanced situational awareness picture, it allows senior leaders to act and make decisions more confidently. These levels of processing range from simple data ingestion for visualization purposes to classification services to provide context to mission events. All provide varying levels of information for decision support.

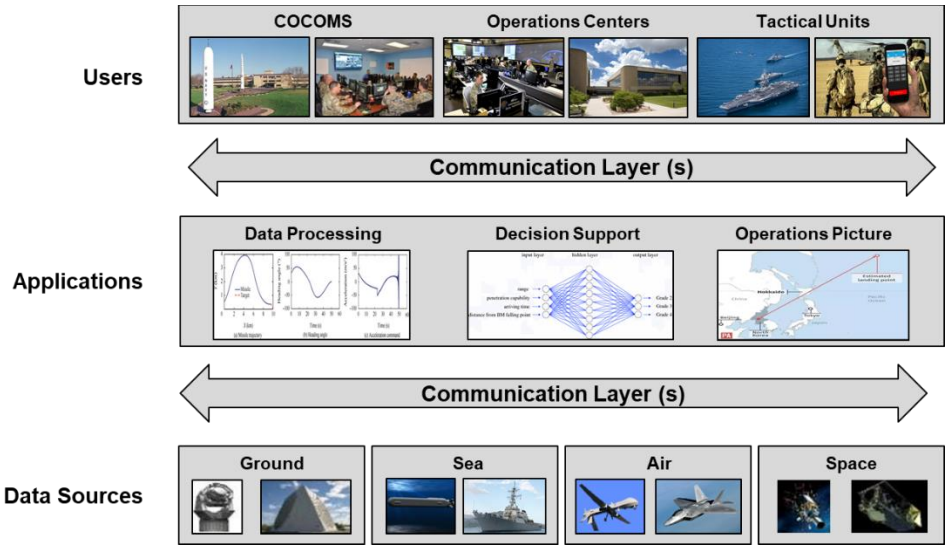


*Figure 2: An example of a data-driven decision support system.*

The ultimate goal of the systems we describe is to inform high-level decisions. This may require many steps of processing, each representing increased contextualization. This also means that there are many nuanced processing paths through which knowledge could be derived. When you consider the timeframe within which a response may be needed, understanding the subtleties of the processing becomes exponentially more difficult when more data, algorithms, and knowledge services are in the system. This is the risk we seek to mitigate.

Next, we will describe the data sharing and interoperability environment, which have been built to enable rapid decision support on a large-scale system. Such an environment has four functional requirements: (1) a low barrier to innovative idea implementation, (2) a large open development environment, (3) a messaging method that supports integration, and (4) an automated validation and verification of adopted components. Figure 3 shows the cycle of the four requirements, and highlights in blue the verification and validation requirement, which is currently under examined at the system scale and is the focus of this paper.

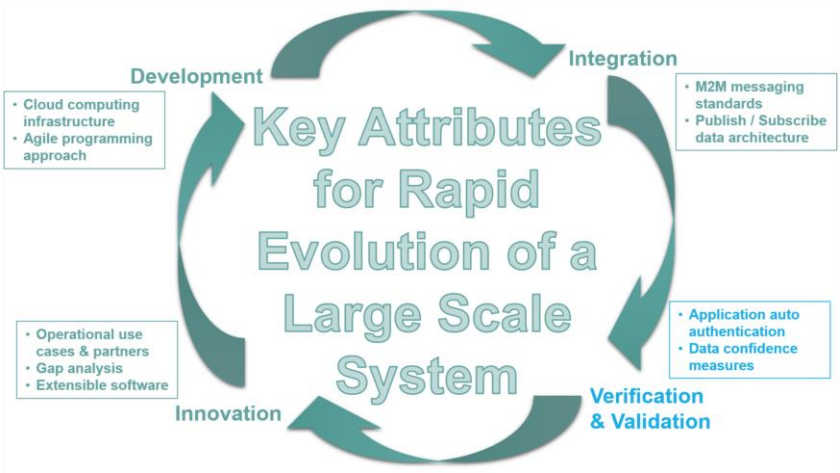


*Figure 3: Key attributes needed for system evolution.*

While individual developers may be required to meet basic expectations to act with integrity, such as compliance with their data and application contracts and the use of software security scans, there is no efficient manner to assess the decision support utility and veracity of the data and services as assembled in the overall system. Many large scale systems do not prioritize cybersecurity or data integrity early in development. End-to-end verification and validation testing is traditionally conducted, after the components are integrated at scale, prior to a large software release. Subsequently, smaller scale regression testing is performed for incremental software upgrades. After the testing is complete, the system is technically ready for operations. However, the performance risks in a loosely coupled, modern software processing system are increased due to the scale and dynamic nature of the applications and their data sources. In many cases, data in these systems are provided by remote sensing measurement sources. Under these circumstances, it is very important to perform *real-time* system-level verification of the data integrity and validation of the derived data products to detect potentially significant changes in performance and erroneous information presented to the operator. This real-time verification and validation approach we discuss requires the recording of the data origin, the historical data quality, and all paths traversed by the data through the system to assure detection of performance issues.

It is important to define the general meaning of verification and validation in this context:

Verification is the evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition (PMI, 2013). In other words, a verification method establishes authenticity.

Validation is the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders, to include suitability and confidence (PMI, 2013). In other words, a validation method establishes some circumstantial usability.

Verification and Validation, as defined above, provide the following:

Figure 4 provides a list of measurable subcomponents of verification and validation as defined above. We propose using *blockchain methods* to enable the real-time assessment of these metrics, which have been demonstrated on a real-time system (MIT Lincoln Laboratory, 2019).

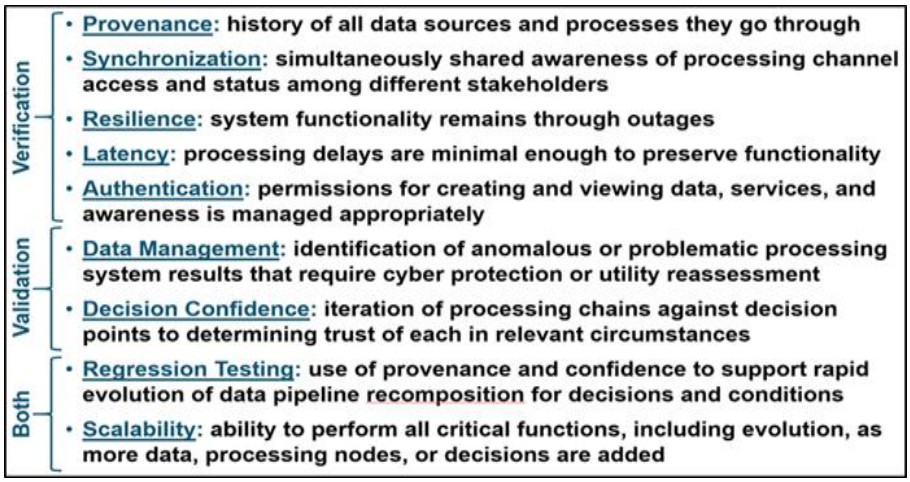


*Figure 4: Proposed metrics for system verification and validation.*

The blockchain approach addresses verification by providing an organic method for storing data, processing provenance and demonstrating synchronization all while offering system resilience and adding minimal latency. Figure 5 shows how we leverage the directed acyclic nature of a typical data processing chain to overlay crypto security and implement a blockchain. This data structure is a straightforward match to traditional blockchain applications. Instrumenting the source layers (S) and application (A) layers with lightweight crypto security at each step, users (U) can verify the data being presented. This architecture provides methods for: configuration management (each software update has a unique security label), error forensics (each path through the processing has a unique ID), and prevents against some types of malicious attacks (each unique ID is registered in a ledger which is available to the user).

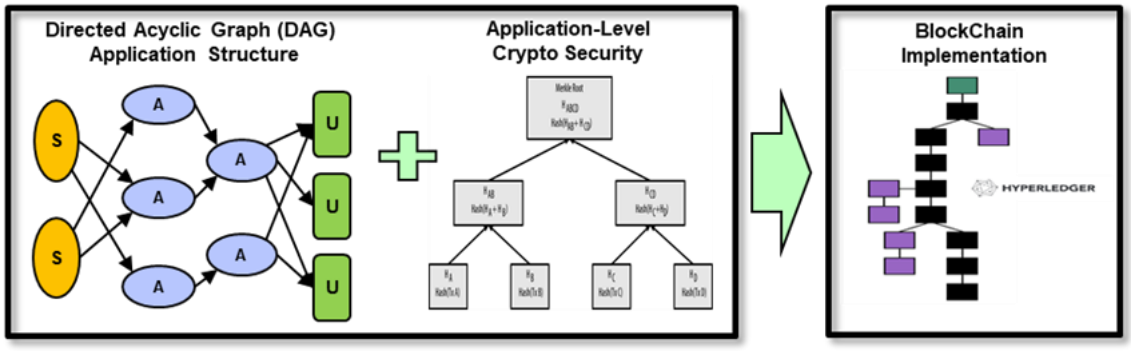


*Figure 5: System verification using blockchain overlay.*

Our method for enabling data validation involves adding data feature analytics to the blockchain infrastructure that are coupled to the processing chain. Pre-compiled data feature statistics and artificial intelligence techniques can be incorporated to provide validation metrics (for example, calibrated performance distributions). These metrics would be calculated in real-time and compared to the pre-compiled features for validation. Figure 6 illustrates this concept.

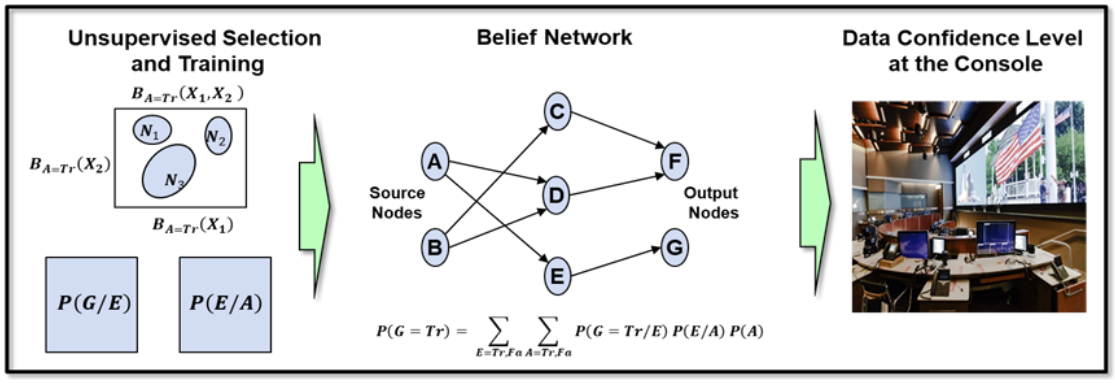


*Figure 6: System validation using machine learning approach.*

# 3. **ENABLING BLOCKCHAIN TECHNOLOGY**

In this section, we describe the basics of blockchain technology, which aspects of it that are applicable to our problem, and a rationalization of our implementation framework.

## 3.1 BLOCKCHAIN BACKGROUND

Blockchain is best known for its use as the cryptographic engine behind decentralized cryptocurrency. Bitcoin, the widely-recognized pioneer in this field, uses a secure and immutable means of consensus on a public ledger amongst users on an open, global, peer-to-peer network. Through a sequence of cryptographically-secure keys, creation of a blockchain affords a secure transfer of information with the key properties of immutability and data integrity. Initially, the underlying consensus mechanism was designed for permissionless blockchain on trustless networks, thus leveraging proof-of-work (i.e., a mathematical challenge which requires high computational power) as a means of creating currency valuation and establishing legitimacy of participants, but this is merely a corollary method, not a requirement, of a blockchain.

## 3.2 WHAT IS A BLOCKCHAIN?

Blockchains are blocks of recorded transactions chained together in a ledger, which refers to a record of transactions. Transactions are transfers of data, for example, bitcoins from one person's account to another person's account. Blocks are records of transactions permitted on the larger chain, which consists of all the blocks in its history.

Traditionally, such data transfer operations are carried out using databases, where data, along with actions to read or modify it, are stored and managed. Blockchains allow for similar operations but allow the system to have multiple writers to the same database without a preconceived notion of trust on the participants. Blockchains also provide an unchangeable past record of transactions that have already happened, along with cryptographic guarantees and transparency of the record to all users. The block contents that record and guarantee the transactions are depicted in Figure 7.

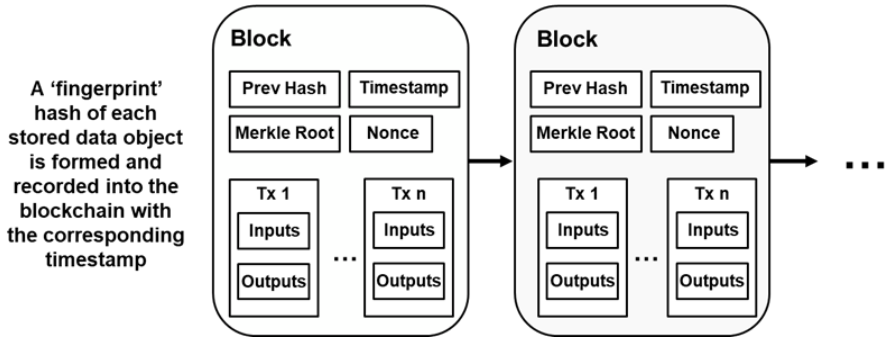


*Figure 7: Description of a blockchain block.*

A final critical feature of a blockchain system is the fact that it is distributed and decentralized. This is achieved by maintaining it amongst a group of peers via consensus protocols. This means that blockchains have increased resilience and availability with more participants, whereas databases are typically subject to single points of failure.

### 3.3 TYPES OF BLOCKCHAINS

Figure 8 depicts two broad types of blockchains that can be used: 1) public networks where any user can validate, write, and reach transaction; 2) private networks where only invited members can validate, write, and reach transactions.

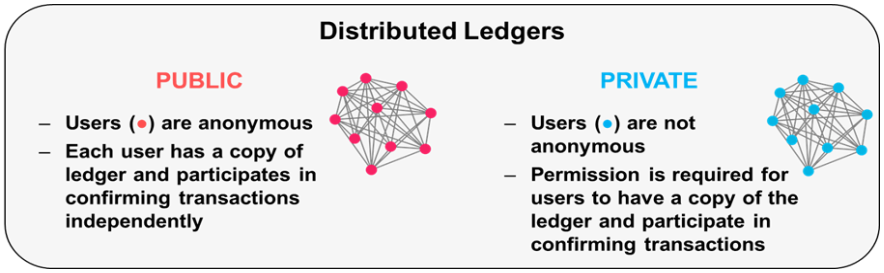


*Figure 8: Types of distributed ledgers.*

Public and permissionless blockchain protocols are designed on the assumption that any given user or contributor is potentially compromised. Tokenized incentives, such as proof of work, can be used to make the untrusted networks safe, but they also make them slow. Typically, public and permissionless networks can only handle a few transactions per second; Bitcoin and Ethereum, for example, process fewer than a dozen transactions per second. This makes public, permissionless blockchains infeasible for large-scale applications with high transaction volumes.

Our DoD data processing problem is focused on real-time transactions at a large scale, but we do not have a requirement to operate in a completely trustless environment. While blockchains are best known for the trustless cryptocurrency application, they can also be employed amongst parties with some trust to mitigate incentive overhead. Other innovations and modifications to the methods and circumstance of the original employment have been developed by commercial companies with the motivation of improved scalability, speed, and versatility.

## 3.4 BLOCKCHAIN PLATFORMS

Initial successes in the creation and use of cryptocurrency have generated significant growth and interest in blockchains. This has led several companies and public foundations to develop blockchain platforms that are widely open sourced and available for use. These platforms allow for rapid prototyping, development, and deployment of new blockchain applications. Each blockchain platform has been designed with specific goals, which dictate its features. In assessing existing technology for suitability to our application, we considered a number of factors, which are depicted in Figure 9.

| | Centralized Ledger | Shared Ledger | Permissioned Private Blockchain | Permissioned Public Blockchain | Permissionless Public Blockchain |
|---|---|---|---|---|---|
| Data Storage | Database | Database replicated across data centers | Blockchain | Blockchain | Blockchain |
| Data Protection | No restrictions | Data replication/ synchronization (Append only) | Data replication/ synchronization (Finality) | Data replication/ synchronization (Finality) | Data replication/ synchronization (No finality) |
| Participants | Predetermined (Identified, authenticated, trusted) | Anonymous (No trust, no authentication) | Predetermined (Identified, authenticated, trusted) | Predetermined (Identified) | Anonymous (No trust, no authentication) |
| Access | Permissioned read/write | Open read/write | Fully private or permissioned read/write | Open read, permissioned write | Open read/write |
| Scalability | Low | High | High | High | High |
| Latency | Low | High | Low | Low | High (~10 min) |
| Security | Single point of failure, data manipulation | Lack of transparency | Strong consistency, weak availability | Strong consistency, weak availability | Lack of privacy & security controls, 51% attack |

*Figure 9: Comparison of blockchain implementation performance factors.*

## 3.5 BLOCKCHAIN IMPLEMENTATION

The system we built employs Hyperledger Fabric, a distributed *permissioned private* blockchain network (visit https://www.hyperledger.org/use/fabric). We selected Hyperledger because it is designed to meet Enterprise requirements for providing robust security and authentication features, it allows for an authenticated set of permissioned participants, specific management of consensus methods between them, and a modular architecture that allows for component implementation, such as high-speed channels to meet system transport-delay constraints and custom code to manage transaction rules. Furthermore, Hyperledger, used by the Linux Foundation in association with a number of corporations (e.g., IBM, Oracle), has a robust user base and demonstrated usability in a variety of applications.

Specific features of a permissioned platform are designed to meet enterprise use-case requirements for real-time transactions between securely identified participants. These features include: 1) participants must be identified/identifiable; 2) networks are required to be permissioned; 3) high transaction throughput performance; 4) low latency of transaction confirmation; 5) authentication, privacy, and confidentiality of communication between clients and providers (e.g., operations and data centers) is consistently supported.

All members of a Hyperledger Fabric network are required to enroll through a trusted Membership Service Provider (MSP). The enrollment protocols provide capabilities that enable flexibility in consensus management. Whereas most coin-oriented ledger platforms employ Byzantine Fault Tolerant (BFT) methods to determine consensus, another distinct design feature of Fabric that differentiates it from other permissioned platforms is its support for alternative consensus protocols, such as Crash Fault Tolerance (CFT), that are both more computationally efficient and practical for many transactional use-cases.

Hyperledger Fabric's consensus protocol flexibility allows for parallel code execution, effectively increasing system-wide performance and eliminating vulnerabilities caused by non-determinism. Fabric does not require specialized domain-specific languages (DSL) for smart contract coding to preserve the reliability of the network; therefore, Fabric differs from DSL-constrained platforms by supporting smart contracts (chaincode) implemented in general-purpose programming languages such as Node.js, Python, Java, and Go. Furthermore, every execution of a smart contract in most DSL-based systems is public since the transaction and often the source code itself are usually visible by other participants. This public feature of DSL implementations often complicates or impedes exchange of private data and private (algorithmic) agreements between participants. Many use-cases require subgroups of participants to share information that is kept private from other participants. Fabric achieves such confidentiality by using its channel architecture to restrict the distribution of confidential information exclusively to authorized nodes. Figure 10 highlights the Hyperledger components used in our framework.

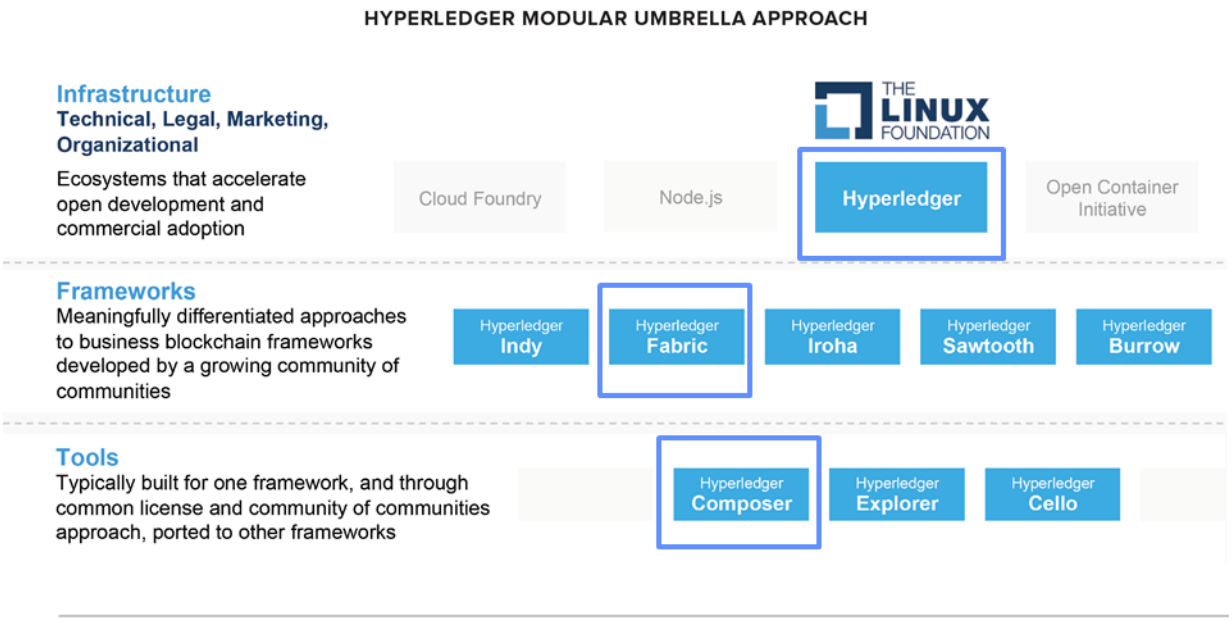


*Figure 10: Hyperledger architecture.*

## 3.6 DESCRIPTION OF BLOCKCHAIN PROTOTYPE

We leveraged Hyperledger Fabric Certificate Authority (CA) to provide features for registration of digital identities, and the issuance, renewal, and revocation of Enrollment Certificates (ECerts) to users, which, in our use case, would be operations and data centers. These are standard cryptographically validated

digital certificates, where all communication to the CA server is accomplished via secure Representational State Transfer (REST) application program interfaces (API) as specified in a JavaScript Object Notation (JSON) configuration file. The Fabric CA is a private Root CA capable of providing and managing its own certificates as well as using a third-party commercial CA (e.g., Symantec) to provide identification. The Fabric Membership Provider uses the enrollment CA ECerts to define members of functional organizations, roles, and access privileges. The connections and various means and methods that peer nodes may use to accept input from other users are heterogeneous and may vary greatly in local implementations. Once a participant is Enrolled into a defined Membership, the modes of communication are well-defined by Fabric standards.

All blockchain network peer nodes hold copies of ledgers and chaincode. Fabric's privacy-oriented design differs in that it allows private chaincode relationships between any two (or more) members and multiple ledgers into private data collections. Fabric Membership conveys permissions to transact with other members through their private data collections. A single enrolled Identity may be a member of several different Fabric organizations, and many different relationships and transactions may be defined in chaincode.

To test this, we simulated real-world operation and data center identities (created by a Fabric CA for test purposes) which were enrolled by default as Members of an assigned organization. All simulated data from each of the ops floor Members were sent (by default) with no editing or redaction. Each operator creates its own ledger, viewable only by itself and possibly a central entity, which in turn, creates a ledger of ALL ops floor transmission from all operators, and it is the only Member with permission to access. Figure 11 illustrates the mapping from the data processing system components into blockchain components.

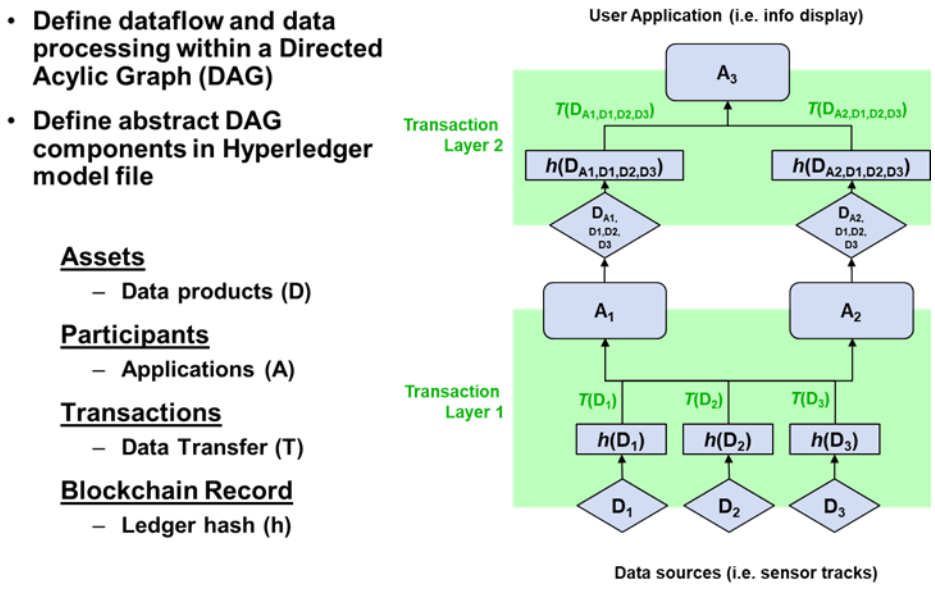


*Figure 11: Mapping a processing system to a directed-acyclic-graph.*

This page intentionally left blank.

# 4. SYSTEM VERIFICATION AND SYSTEM VALIDATION DEMONSTRATION

We now describe an example of an operationally-representative software system in order to demonstrate our objectives of system verification and on system validation using the blockchain approach. Figure 12 depicts the three configurations of a simple missile trajectory assessment system. The first configuration, shown in the top panel, is the initial system. Missile position and velocity track data are ingested from several sources. This data is filtered for anomalies using fixed, developer-defined thresholds. Then an application is used to propagate the track data to determine the time and location of ground impact with the earth. The resulting impact is then presented to the users (MIT Lincoln Laboratory, 2019).

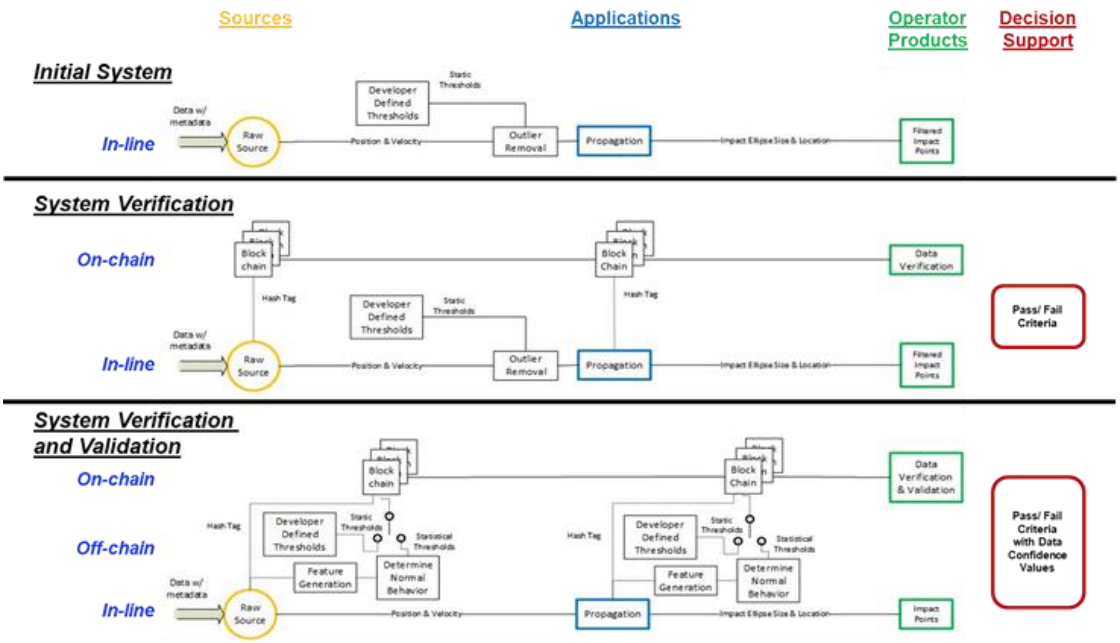


*Figure 12: Blockchain verification and validation demonstration architectures.*

This initial system has several shortfalls:

1. The user does not have insight into the pedigree of the data and its processing.
2. The user does not have access to the data that was filtered out
3. The data filtering rules are fixed, based on previous validation testing and developer insight.

The other two systems shown in Figure 12 attempt to address these shortfalls and are discussed next.

## 4.1 VERIFICATION DEMONSTRATION

The overall goal of our verification system demonstration was to address shortfall #1 using open-source blockchain code-based components on a non-interference basis. The main goal of this implementation was to establish provenance on a multifaceted chain. In addition we were able to show resilience in the presence of connectivity outages from either a data source or user, and verify data authenticity in the presence of unregistered data products. The middle panel of Figure 12 details the implementation of our verification system.

The first aspect of verification we demonstrated was data provenance, which describes the chronology of the ownership, custody, or location of information throughout its history. This was accomplished by including a cryptographic tag in each message from each data source or application. This tag was recorded on the blockchain each time a new message was received. This blockchain ledger of transactions is available to each user.

The second aspect of verification we demonstrated was resilience, which means the system can continue functioning with peer outages. Since each peer has access to the blockchain information, data verification can be accomplished through the consensus of the collective users. As long as the majority of users are available, the data can be verified.

The final aspect of verification we demonstrated was authentication, which we accomplished by introducing an unregistered data stream into the processing pipeline and showing it failed the verification protocol since its crypto tag was not in the blockchain ledger. Non-valid codes would automatically trigger a warning. Since the crypto tags of the processing system are mapped to a directed-acyclic-graph (discussed in previous section), the users could then determine the point in the processing from which the warning originated.

This verification system demonstration successfully integrated a blockchain with a realistic data processing system. The blockchain component was very lightweight with minimal additional overhead on the message content in the system. We focused on mapping the blockchain to this use case and evaluating the system with provenance, resilience, and authentication metrics (as defined previously in Figure 4).

## 4.2 VALIDATION DEMONSTRATION

We now address shortfalls #2 and #3 through the additional step of data validation. This system is shown in the bottom panel of Figure 12. Two different approaches were introduced – static thresholding and statistical thresholding. Both approaches employ the concept of data flagging rather than data filtering, as was done in the initial system. The initial system used developer-specified thresholds to filter out data that exceeded the thresholds. In our system, we developed a flagging mechanism, albeit on static or dynamic thresholds, where the flags are stored, along with the verification data, on the blockchain. To consider the merits of static configuration file filters vs. blockchain flagging, we introduce the following definitions:

*Filtering: suppression or removal of data from the processing pipeline*

*Flagging: automatically adding metadata that is carried through the processing pipeline*

The initial flagging thresholds can be statically set by the developer or statistically determined by machine learning techniques run on historical data. The users are given access to these thresholds, on the

blockchain, and can adjust in real-time if needed. In addition, machine learning can take place during real-time operations and the statistical thresholds can be continuously updated. Figure 13 compares the benefits and regrets between the input filtering and blockchain flagging methods.

| | Static configuration file filter | Chaincode data flagging |
|---|---|---|
| Multiple features filtered | | |
| Filters at multiple processing steps | | Chaincode can flag data products at any and many transactions |
| Filter history over time | | Ledger preserves copies of all flag changes |
| Filter commonality / transparency at all sites | | All participants can see data flags and history |
| Distributed filtering | | Flagging is done everywhere and all sites can see what data was flagged |
| Optional, user-specified interpretation of filters | | All data still transmitted, but with flags attached for user interpretation |
| Simple to implement | | Chaincode is slightly more complex than file |
| Extensible to more data types and steps | | Subject to real-time computation requirement |

*Figure 13: Comparison between filtering and flagging data.*

We implemented the flags in our demonstrated blockchain based on static flag settings and statistical methods. The advantages and disadvantages of each method are listed in Figure 14.

| Thresholding Method | Advantages | Disadvantages |
|---|---|---|
| Static | • Developer or user defined<br>• Simple to implement<br>• Minimal latency<br>• User adjustable | • Subject matter or user expertise needed to adjust |
| Statistical | • Data defined<br>• Little expertise needed<br>• Dynamically updated | • Significant amount of training data needed<br>• Computationally expensive |

*Figure 14: Types of thresholding used for data flagging.*

The thresholds generated for each method are stored on the blockchain and used for flagging data. However since these thresholds have to be determined a priori, the work done to create them is done *off-chain* and can be computationally expensive, particularly in the case of dynamically updating the statistical thresholds.

### 4.3 DATA ANALYTICS: OFF-CHAIN ANALYSIS FOR ON-CHAIN IMPLEMENTATION

In this section, we will describe the various methods of off-chain outlier detection that we implemented on the blockchain for our data products. We chose outlier detection because it can be performed using basic mathematic statistical techniques and it inherently provides some organic insight into system performance. In addition, almost no subject matter expertise is needed to develop the outlier thresholds; the thresholds are set based on the historical data. The probabilistic results also lend themselves to dynamic, tunable, and combinable flags that can be standardized across the chain.

To understand the utility trades of different off-chain processing methods, we chose three independent statistical techniques to exercise on the same training and test data.

- Method A: Single variable distribution estimation
- Method B: Multi-variable joint distribution estimation
- Method C: Clustering using non-parametric estimators (Xiaowei, 2017)

The first two methods involve selecting and conditioning feature data from compiled historic data, building distributions for each transaction, and storing the resulting distributin parameters on the blockchain. The third method employs Empirical Data Analytics (EDA), and provides a more optimal anomaly detection solution. However, this method in its current form does not readily map to a blockchain implementation.

The purpose of these techniques is to determine whether a data point is in-family or an outlier. This information is depicted using features (via distributions) as a representation of all possible transactions across the system, each described by a particular input and output source pair (i.e., a node on the directed-acyclic-graph). Since we are focusing on the feasibility of translating off-chain analytic distributions to on-chain processing, we chose basic features such as missile altitude, impact time, position and velocity rates as well as other features relevant to the simple trajectory assessment system we used for demonstration. After the baseline ‘nominal’ statistical distributions were calculated off-chain, the distributions were implemented on the blockchain. Figure 15 shows how the parameters of a multi-variate Gaussian distribution were stored on the blockchain and Figure 16 shows how the test for an outlier was performed using a standard mahalanobis distance.

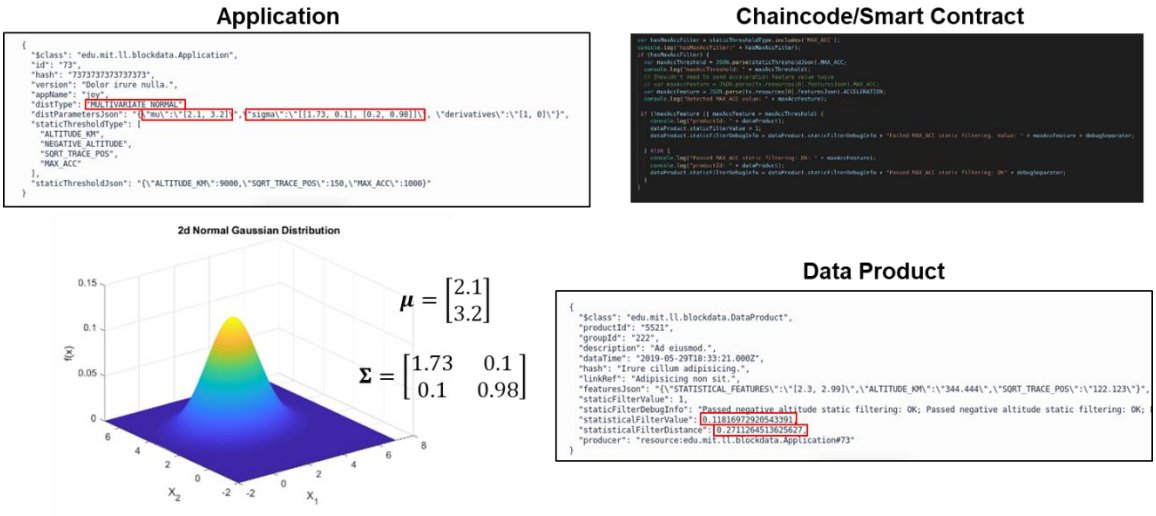


*Figure 15: Embedding statistical distribution parameters on the blockchain.*

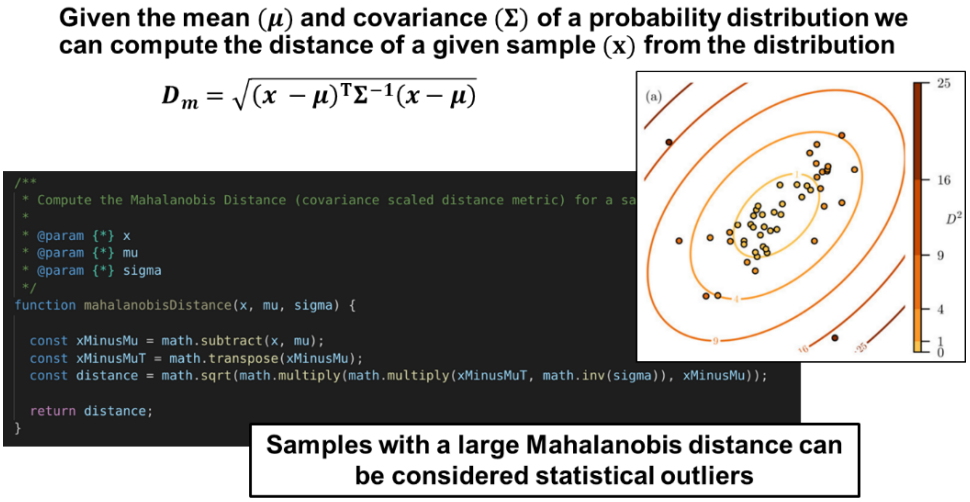


*Figure 16: Outlier detection on the blockchain.*

After testing the three outlier detection techniques on the historical data, we identified 10 factors to consider in evaluating the implementation of each. We summarized the benefits and regrets of the techniques in Figure 17.

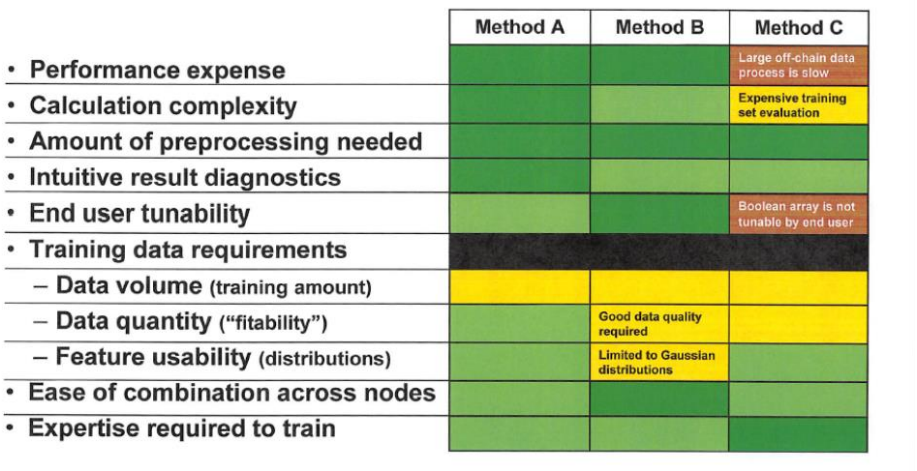


*Figure 17: Comparison of methods for statistical parameter estimation.*

In summary, methods A and B, single versus multi-variate parameter estimation, are both feasible approaches that lend themselves to a blockchain implementation. Method C performed well, but does not match up well with this type of tunable implementation.

# 5. POTENTIAL FUTURE WORK

Given the results of our work to date, our next objective will be to prove the ease of implementing free, open-source blockchain codebase components on a non-interference basis to an existing real-time data processing pipeline. The amount of modifications needed to the processing and messaging will be assessed as well as the potential benefits to the users. It is envisioned that this proof-of-concept will be in concert with the exploration, and potential development, of applying 'Verification and Validation' to existing operational systems.

As a part of this activity, we see follow-on demonstrations that would continue on the path of less-to-more autonomy, from fusion applications,to gain confidence in the verification and validation process, to artificial intelligence and machine learning applications. In all cases, the goals of these implementations would be to show establishment of provenance on a multifaceted chain, resilience in the instance of participant connectivity outages, and data verification in the presence of unregistered data products.

This page intentionally left blank.

# 6. **SUMMARY**

In this paper, we have discussed the challenges that arise with increased data sharing and interoperability in maintaining a level of trust in DOD systems. But have also discussed an approach to more efficiently and effectively leverage an open-source enterprise blockchain framework. We have also demonstrated that we can collect, verify, and validate data processing chains by tracking provenance and using smart contracts to add dynamically calculated metadata to an immutable and distributed ledger.

We also claim that this non-invasive approach to verification and validation in data sharing environments has the power to improve decision contextualization at larger scale than manual approaches, such as consulting individual developers or mandating large amounts of additional metadata in schema. But to achieve full functionality, this will require us to define an operational decision support structure that can handle the informational content and complexity of this infrastructure on behalf of the user.

In the introduction of this paper, we described levels of processing (see Figure 1) to refer to the types of information that are supplied and leveraged by decision makers and operational contextualized data processes. Operational decision support structures can be thought of as a hierarchy — a series of building blocks where higher-level compute processing increases with converting data to features and with converting features to decisions. However important decision support services that address cross-mission domains are still lacking.

Figure 18 builds upon the levels of processing previously discussed and makes use of our hypothesized operational decision support hierarchical structure to provide a mission-agnostic support to achieving the ultimate goal of informing high-level decisions.

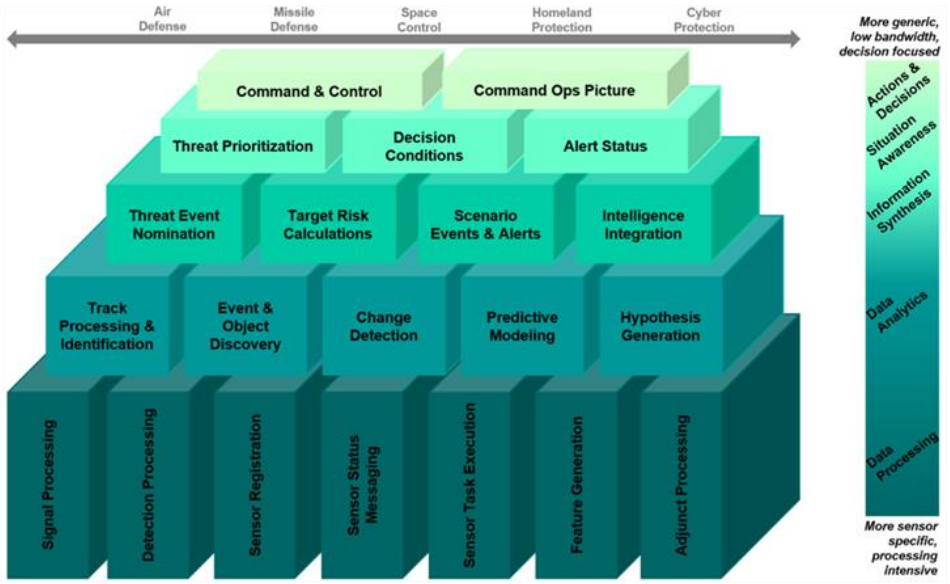


*Figure 18: Structure of a DoD multi-mission decision support system.*

In summary, one could think of Blockchain-based Data Analytics in a broader context. Applying it to the basics of data provenance and validation, to multiple independent sources corroborating data-based inferences, to anomaly detection with confidence-based metrics alerting users of differences in the expected behavior, to machine learning identifying change detection and pattern of life. In all these cases, the key is enabling the assignment of significance to deviations in expected behavior, thus providing a level of confidence to the data user.

# REFERENCES


Amidon (1997). *Innovation Strategy for the Knowledge Economy.* Elsevier INC.

MIT Lincoln Laboratory (2019). *Demonstration of Blockchain Approaches for Data Analytics.* (Classified briefing).

PMI (2013). *Program Management Body of Knowledge* (5 ed.). Newtown Square, PA: Project Management Institute.

Xiaowei, G. (2017). Autonomous Anomally Detection. *Evolving and Adaptive Intelligent Systems (EAIS)*.

This page intentionally left blank.