SANDIA REPORT
SAND2021-2208
Printed October 2019

Sandia National Laboratories

Prototype Distributed Ledger Technology of UF_6 Cylinder Tracking in Ethereum

Nicholas D. Pattengale
David R. Farley

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

We have created a demonstration permissioned Distributed Ledger Technology (DLT) datastore for the UF_6 cylinder tracking safeguards use-case utilizing the Ethereum DLT framework and using Solidity for smart contract code. Our demonstration creates a simulated dataset representing tracking of 75,000 UF_6 cylinders across 11 example nuclear facilities worldwide. Our DLT system allows for easy input and reading of shipping and receiving data, including a Graphical User Interface (GUI). Sandia's Emulytics capability was leveraged to help create the DLT node network and assess performance. We find that our DLT prototype can easily handle to ~150,000 UF_6 cylinder shipments per year worldwide, without any excessive computational or storage burden on the IAEA or Member States. Next steps could include a demonstration to the IAEA and potentially demonstrating integration with TradeLens, a DLT in use by a consortium of international shipping companies representing over half of world shipping trade.

ACKNOWLEDGEMENTS

We thank the NA-24 sponsor for their support of this distributed ledger technology demonstration project

CONTENTS

LIST OF FIGURES

This page left blank

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
CoK	Continuity of Knowledge
DLT	Distributed Ledger Technology
IAEA	International Atomic Energy Agency
GUI	Graphical User Interface
KMP	Key Measurement Point
MBA	Material Balance Area
NMA	Nuclear Material Accountancy
UID	Unique Identification
VM	Virtual Machine

1. INTRODUCTION

For the past several years there has been considerable enthusiasm regarding the potential of Distributed Ledger Technology (DLT), often simply referred to as Blockchain. As with many emerging, potentially disruptive technologies, there can be an associated "hype cycle" where initial enthusiasm fades after enhanced expectations, followed by the "trough of disillusionment". Gartner releases a yearly hype cycle graph illustrating the stage of industry acceptance of such technologies, as shown below in

. As observed, blockchain as of 2018 has entered the phase of disillusionment, but that does not mean it has failed as an emerging technology. Rather, reality is setting in, and industries now must grapple with actual implementations of blockchains and develop valid business cases. Note that blockchain is not listed on the Gartner 2019 hype cycle graph, but Gartner does provide an entire article in 2019 on the business merits of blockchain going forward.[1]

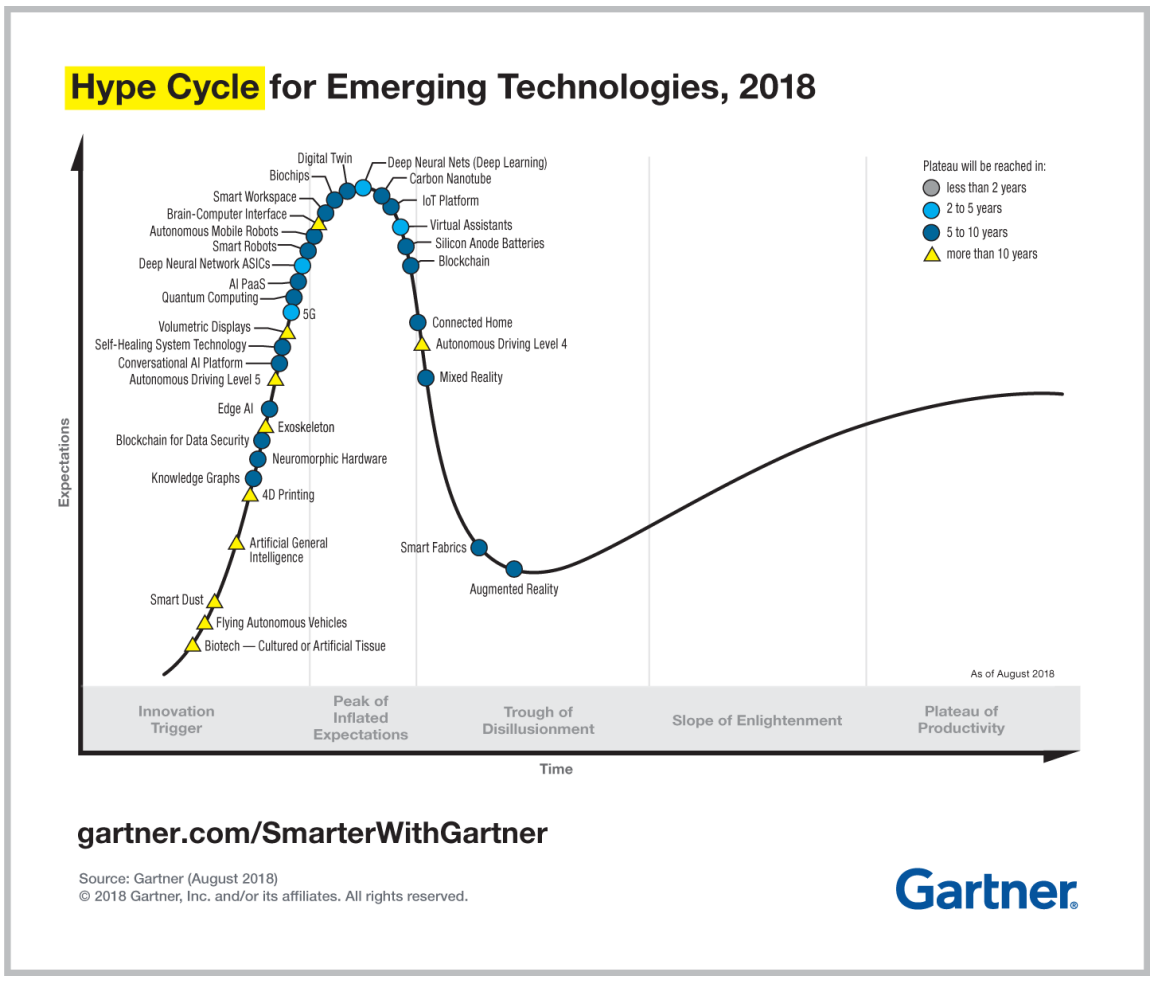

Figure 1. Gartner's 2018 "Hype Cycle" graph illustrating industry acceptance of potentially disruptive emerging technologies.

[1] Kasey Panetta, "The CIO's Guide to Blockchain", https://www.gartner.com/smarterwithgartner/the-cios-guide-to-blockchain/ (September 23, 2019).

The International Atomic Energy Agency (IAEA) has expressed interest in DLT. Applicability of DLT for IAEA purposes seems obvious since the IAEA relies heavily on Nuclear Material Accountancy (NMA), which in large part relies on counting of items and recording numbers on separate ledgers (i.e. the IAEA ledger and the Member State's ledger). Much time and expense are required to reconcile these ledgers if a defect is detected (a *gross defect* means a large, bulk amount of material is missing, such as a uranium hexafluoride (UF_6) cylinder; whereas a *partial defect* refers to a small fraction of material). Before the IAEA can implement DLT for its purposes, it would need to be certain of its efficacy at NMA for safeguards purposes, including practical issues such as code sustainability, authentication standards, cost, storage requirements, etc., but the IAEA itself is unprepared to undertake this testing and development effort on its own.

Pacific Northwest National Laboratory (PNNL) has been studying the applicability of DLT for IAEA safeguards purposes since 2016 [1], [2]. These studies provided a metric to judge the value of DLT in various IAEA safeguards scenarios. Some safeguards scenarios showed a strong applicability for DLT whereas other use-cases showed less applicability. One safeguards use-case that scored highly is the safeguards problem of UF_6 cylinder tracking. The nuclear industry commonly handles standardized steel cylinders containing UF_6 across many stages of the nuclear fuel cycle. Previous reports [3] have discussed aspects of the current tracking system that could be improved to reduce the risk of material diversion by increasing continuity of knowledge (CoK). As the former Director of Safeguards at the IAEA, Olli Heinonen, explains [4]

> "About 100,000 UF_6 cylinders are currently in worldwide use. Most of them are used to store depleted uranium, but there are annually about 15,000 movements of cylinders containing low- enriched or natural uranium. These cylinders move from country to country often overseas between uranium conversion, enrichment and fuel fabrication plants on journeys and voyages, which last several weeks. While industry is good at tracking valuable materials, it may take several weeks before missing cylinders are detected and the cylinders are located."

Continuing, Heinonen states,

> "Cylinders in transit or stocks can be diverted by a state or obtained by subnational groups or black market vendors. There are several diversion scenarios including diverting a known, declared cylinder for uranium enrichment in a clandestine facility, or misusing a declared cylinder without reporting to the regulatory body or the IAEA in a declared, safeguarded facility, or using an undeclared cylinder at a safeguarded facility."

To address the UF_6 cylinder tracking issue, various techniques are being explored to minimize the chance of accountancy defects, such as the Global ID concept of a applying a unique ID to cylinders to facilitate cylinder tracking [3]. But even assuming a unique and counterfeit-safe technology can be applied to UF_6 cylinders, the problem remains for accurately tracking ownership and physical location worldwide of ~100,000 items potentially holding accountable nuclear material. The number of digital transactions each year is estimated to total approximately 150,000 [2], and multiple ledgers are used by various parties (e.g. shipper, receiver, storage facility, IAEA, etc.) in an attempt to keep an accurate ongoing tally.

Therefore, it is suggested that DLT may be an effective solution to the UF_6 cylinder tracking problem for nuclear safeguards. To take a step towards testing this hypothesis, we have implemented a prototype UF_6 cylinder tracker in a private Ethereum ledger. Executing this work is consistent with the opinion [5] that now is an appropriate time for prototyping potential DLT applications to better understand how purported potential reconciles with implementation realities.

Our prototype demonstrates that Ethereum is more than sufficient to support the data requirements of such a system, achieving 150,000 events per year with no perceived difficulty.

2. BACKGROUND

There are multiple safeguards use-cases where a DLT solution may improve NMA for the IAEA, such as UF_6 cylinder tracking, nuclear material transit matching (matching reports of domestic and international shipments and receipts), or as a resource for the noncompliance process when there is a disputed irregularity [2]. This list is not comprehensive, and has analogies with other industries and concerns, such as improving international shipping manifest procedures, combating counterfeit parts, or healthcare data. Thus, we can learn from developments in other arenas beyond IAEA needs, and there is a vibrant open-source community of developers that can also be leveraged.

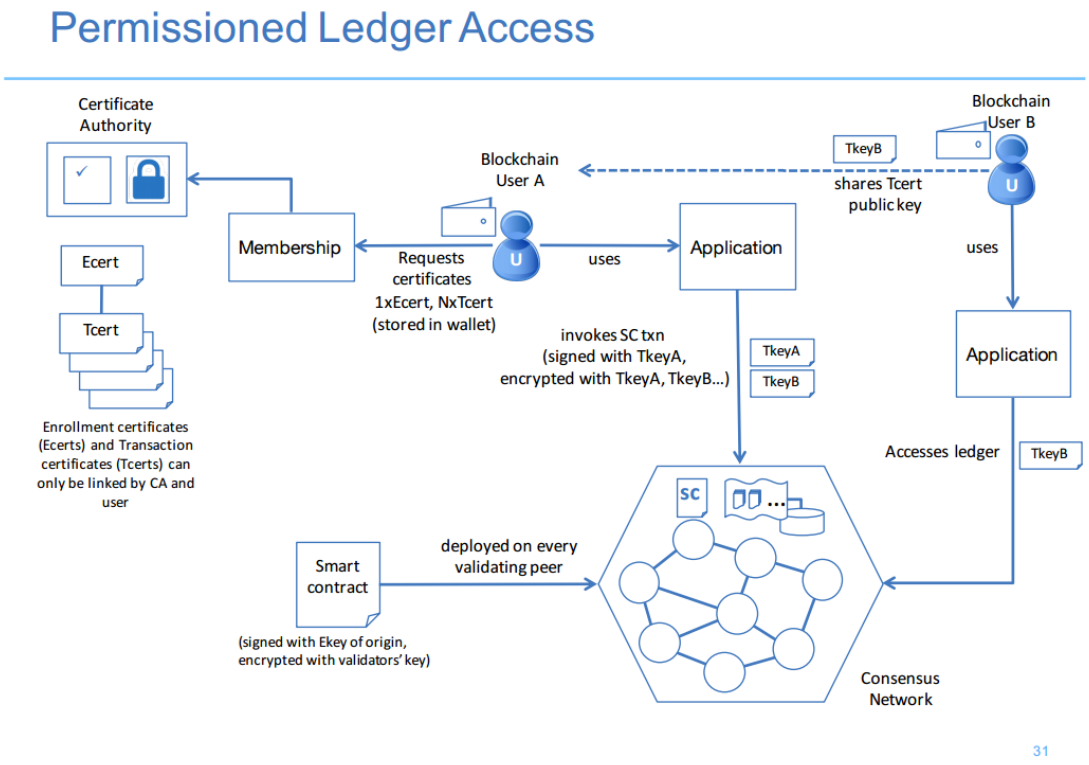

Figure 2. Illustration of a permissioned DLT, where a Central Authority (Certificate Authority) allows access to the DLT, but has the consensus network of a decentralized DLT.

Likewise, there are a variety of DLT development platforms, including Ethereum, Hyperledger, Bitcoin, and many for fintech (e.g. Ripple). We divide DLT approaches into two general categories: Decentralized (also called Public) and Permissioned (also called Private). Perhaps the most widely known decentralized DLT is the Bitcoin blockchain, where anyone can join and all parties have full access to all the transaction history (not necessarily the underlying data). A permissioned (private) DLT is distinguished from a decentralized DLT with the additional concept of a Central Authority. A Central Authority provides permissions through "memberships" for access to the network. The permissioned DLT concept is illustrated in Figure 2. The permissioned DLT still maintains the consensus node network of a decentralized DLT, but the Central Authority (Certificate Authority in Figure 2, which is nomenclature used for internet authentication purposes) has overarching authority over the memberships. We believe the permissioned DLT is the best approach since the IAEA

would be the Central Authority, yet is also viewed internationally as a trusted 3rd party (indeed, the IAEA is part of the United Nations). While having some central control is somewhat contrary to the purist view that a blockchain should be completely decentralized (like Bitcoin), this can lead to questions of who is in charge of fixes, etc., and for our purposes we believe the IAEA should be such a Central Authority.

A key aspect of DLT is the ability of participants to achieve consensus on transactions, as well as to have access to the DLT transaction history. The consensus process for a simple majority is illustrated in Figure 3. There are a variety of consensus mechanisms available, and one needs to determine which version is most appropriate for a given DLT solution.

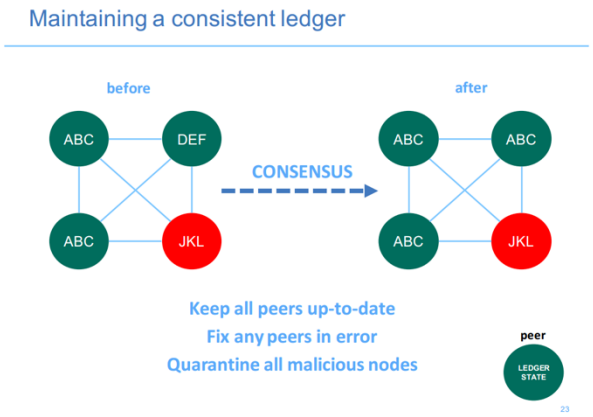

Figure 3. Illustration of a DLT consensus process. Here, the majority decides on the correctness of the data, whereupon a consensus achieves a permanent entry into the DLT.

For this project, we will utilize the Ethereum framework in a Permissioned Ledger implementation. We will use the UF_6 cylinder tracking use-case as a relevant problem to the IAEA that could be improved with such a Permissioned DLT, as described below.

2.1. UF_6 Cylinder Tracking State of Practice

The IAEA utilizes the concept of Material Balance Areas (MBAs) to designate a physical space where certain operations take place, and where accountancy measurements should occur upon crossing a boundary between MBAs, known as a Key Measurement Point (KMP). Figure 4 illustrates a likely scenario for shipment of UF_6 cylinders from the fuel enricher to the fuel fabricator. Cylinders leave the enrichment MBA and measurements to verify contents are done at the KMP and verification of the cylinder ID are entered into the ledger of the enrichment facility as having left the MBA. The shipper likewise should enter the ID of the cylinders into its ledger, which may actually be the shipping manifest. The cylinders are then conveyed towards the fuel fabrication destination. Although a ship is illustrated in Figure 4 as the mode of transportation, for the case of out-of-country enrichment services, the transportation mode could also be a truck, train, or combination of all of the above. Not shown in Figure 4 are the possible ports of call of the ship, which depending on the country may involve an inspection of contents and thus another ledger. Eventually the cylinder shipment arrives at its final destination and is offloaded to the receiver entity. This receiver may be a port authority, or could be the entrance to the fuel fabrication facility. In either case, the UF_6 cylinders would now undergo counting and at least ID verification and entered into a ledger.

Prior to entering the fuel fabrication MBA, measurements can be made to verify contents and once again the cylinder IDs noted in a ledger. There may be further accounting done within the fuel fabrication facility, but the above details illustrated by Figure 4 provides a sense for the need of a better ledger system. It is therefore not surprising that the IAEA, as echoed by Olli Heinonen above, has serious concerns regarding continuity of knowledge for the vast numbers of UF_6 cylinders in transit all over the world at any given time.

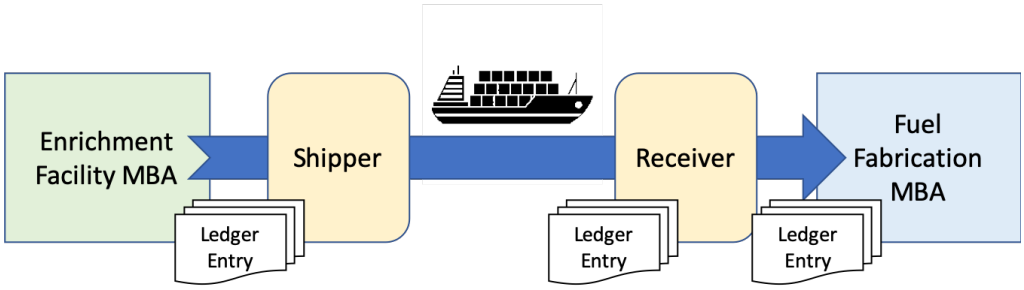

Figure 4. Illustration of transshipment of UF_6 cylinders from an enrichment facility to a fuel fabrication facility.

While there are standards [6] and working groups [7] for UF_6 transportation, including applying unique identification (UID), there is still much effort underway at creating UID technologies, such as the Global ID project [3] and consistent tracking of cylinders [8]. For this project, however, we will assume a UID exists for each cylinder and that it can be read and entered into our DLT system. That said, if other potentially identifying information is also entered into the DLT, such as tare weight, origin and destination, for example, then one may be able to utilize the DLT data to infer the identity of a missing cylinder or one that arrives with no readable ID.

2.2. Ethereum Concepts

There are many resources to understand Ethereum at any desired level of detail [9]. For the purposes of this effort, Ethereum can be thought of as an immutable and audit-friendly decentralized datastore, where not only the data itself can be audited, but so can the application of business logic applied to the data in order to calculate the global state (e.g. the location/status of all UF_6 cylinders worldwide).

The decentralization of data arises from not relying on any particular organization to host a datastore, but rather by having a community of organizations each host copies of the datastore. The organizations are connected to each other via a peer-to-peer network, and the network protocol by which organizations communicate ensure that data consensus is achieved across the redundant datastores.

Figure 5 shows a notional permissioned (private) Ethereum peer-to-peer network whereby shipping/receiving facilities for UF_6 cylinders each run an Ethereum node. Each nuclear facility interacts with its local node to attest to cylinders being shipped and received from their facility, and their Ethereum node gossips with all other Ethereum nodes to reconcile a consistent global state of the shipping network. Consensus in this Ethereum DLT is achieved through the proof-of-work algorithm.

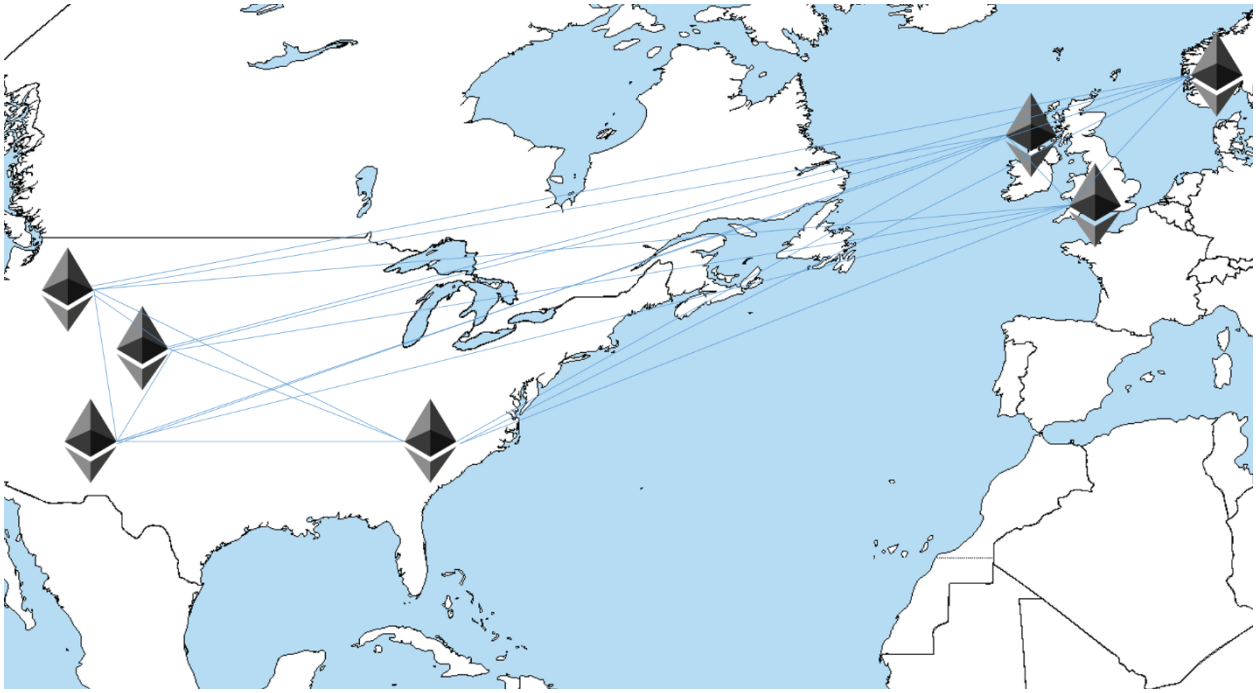

Figure 5. A notional Ethereum permissioned overlay topology, consisting of 7 interconnected nodes.

3. RESULTS

To evaluate and demonstrate the concepts as described thus far, we have built a rudimentary shipping/receiving information system, with data backed by an Ethereum private ledger. This primarily consists of:

- An Ethereum smart contract (see Appendix A) which accepts shipping event data, and reflects the up-to-date state and location of all known cylinders
- A web application which displays the data that has been written to the contract thus far in a user-friendly fashion (see Figs. 7-10).

Additional minimal effort was spent on building an automated pipeline for producing simulated cylinder data and coordinating the submission of that data to the Ethereum contract from a population of agents (computer programs representing nodes where shipping data is produced). This can readily be packaged up in experiment containers compatible with Sandia's FIREWHEEL network analysis capability, to enable automated deployment for demonstration and test.

As such, we have produced a repeatable and automated environment for deploying a fresh private Ethereum network, publishing the UF_6 inventory repository smart contract mechanism to the private Ethereum network, simulating the movement of cylinders (which gives rise to events that are submitted to the contract), and visualizing the real-time state of the system at any point in time. The configurable parameters to this environment are:

- the number of Ethereum nodes/agents
- the number of distinct UF_6 cylinders in play
- the number of UF_6 cylinder events that will be submitted to the contract

3.1. The DLT Contract

The Ethereum Solidity contract, called `UF6InventoryRepo.sol` (and listed in full form in Appendix A), is a notional inventory tracking contract containing minimal, but sufficient, functionality for ledgering shipping events. The workhorse function is `insertEvent`, as shown in Figure 6, which is called to indicate that a cylinder has been shipped or received by/at a facility.

The remainder of the contract provides convenience methods for efficiently examining the body of submitted data. The contract as written efficiently supports

- Retrieving a count of all known cylinders (constant time lookup)
- Iterating over all known cylinders (linear time lookup in number of cylinders)
- Iterating over all events concerning a specific cylinder (linear time lookup in number of events)
- Retrieving a count of all known facilities (constant time lookup)
- Iterating over all known facilities (linear time lookup in number of facilities)
- Retrieving a count of cylinders AT REST at a specific facility (constant time lookup)
- Retrieving a count of cylinders IN TRANSIT (constant time lookup)

The contract could easily be extended to capture different attributes of shipping events or efficiently provide different views of the data (e.g. return all cylinders in a specific geographic region). The implemented functionality is sufficient for assessing high level properties of systems like this.

```
contract UF6InventoryRepo {

function insertEvent (
           string memory action, //e.g. SHIPPED or RECEIVED
           string memory reporter, //the organization reporting the
event (the shipper/receiver)
           string memory cylinderID, //the UID for the cylinder
           string memory lat, //geographical latitude
           string memory long, //geographical longitude
           string memory flag, //state with accountability for
cylinder
           string memory timestamp //unix timestamp for event
    )
```

Figure 6. Code snippet of Ethereum Solidity contract `UF6InventoryRepo.sol` which captures cylinder data upon entry by a user.

3.2. The DLT User Interface

Using command line tools to interact with smart contracts and inventory data can be cumbersome (akin to a database administrator interacting with data via the console). Any realistic enterprise deployment of a smart contract will likely contain a user interface layer, such as a graphically oriented user interface (GUI), via which typical users will interact with the system, which we assume will be the desired interface for the IAEA.

We have built a simplistic but functional user interface layer based on Tabulator (https://tabulator.info) to explore data that has been submitted to an `UF6InventoryRepo` contract. This interface is implemented as a web application using modern javascript components for data reactivity and design cleanliness. But whereas typical web applications query data from a database or other web pages, our web application relies on a plugin called metamask (https://metamask.io/, which "brings Ethereum to your browser" that enables web pages to call smart contracts, and use the returned data to populate web page data elements.

For example, Figure 7 shows a web page displaying high level summary information about a deployed `UF6InventoryRepo` contract, as retrieved by metamask. Figures 8-10 show varying views of the contract-managed data including a list (with current state) of all known cylinders, a cylinder-specific listing of all historical events involving that cylinder, and a facility-centric view showing counts of cylinders at various facilities, respectively. With this prototype, we have demonstrated the essence of a full shipment tracking system. It would be a matter of implementation effort to mature this minimal web application into a fully functional operational solution.

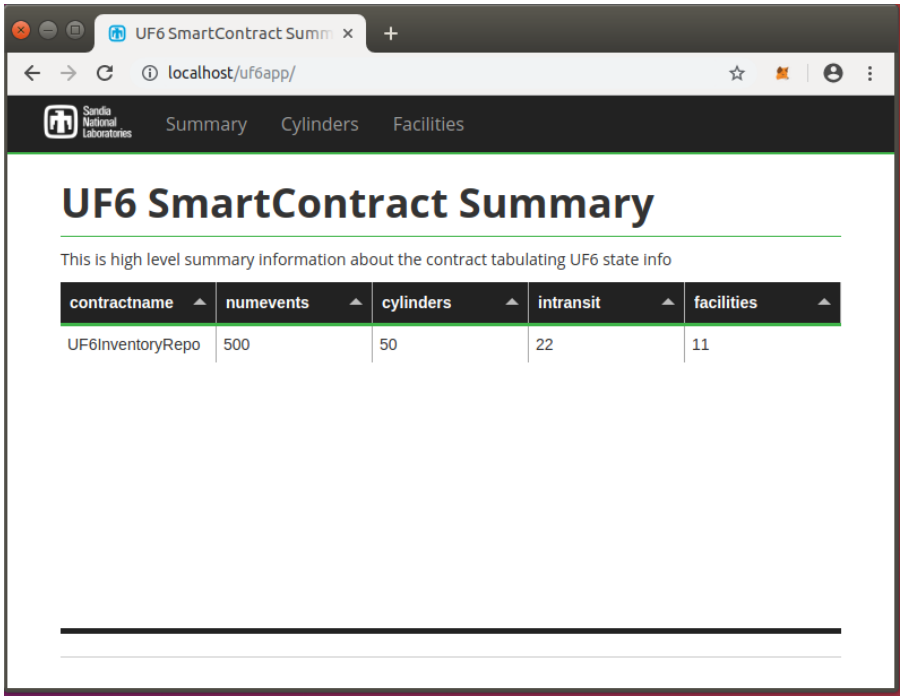

Figure 7. A webpage that leverages the metamask plugin to call a deployed `UF6InventoryRepo` contract and disply high level information about the data state of the contract.

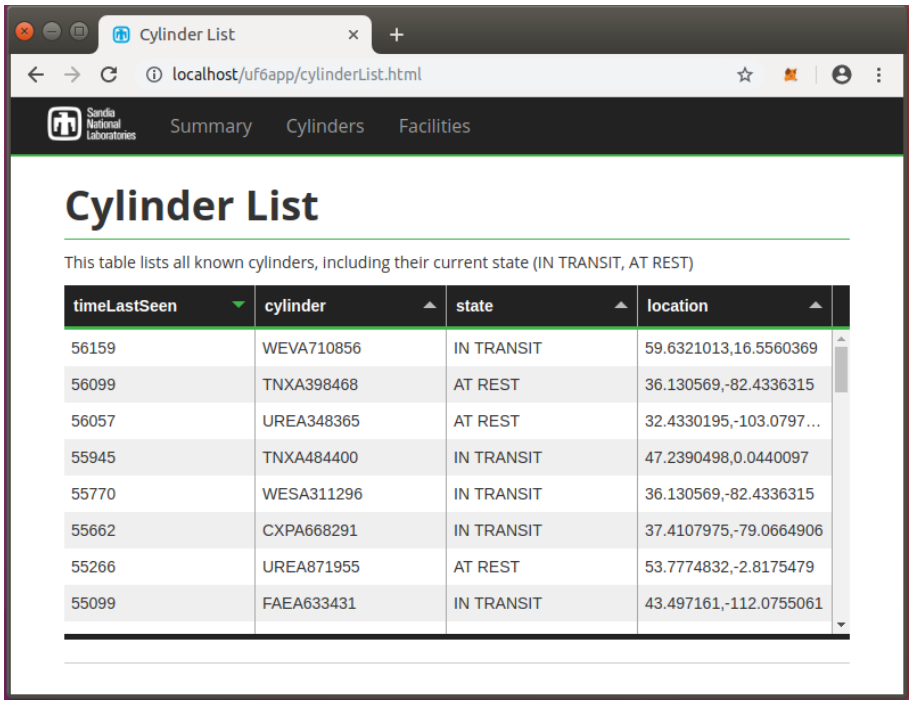

Figure 8. A view of `UF6InventoryRepo` data where all cylinders are listed in a table. The table uses progressive loading such that data is only retrieved in batches as a user scrolls the table.

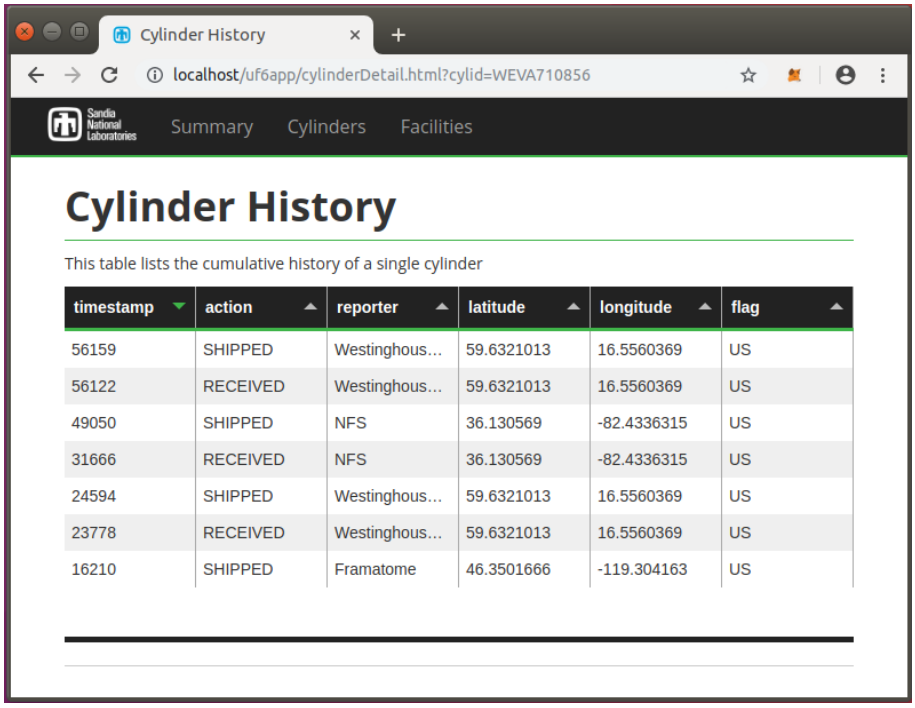

Figure 9. Clicking on a specific cylinder in the previous view yields a cylinder centric view whereby all shipping events (in history) for that particular cylinder are shown.

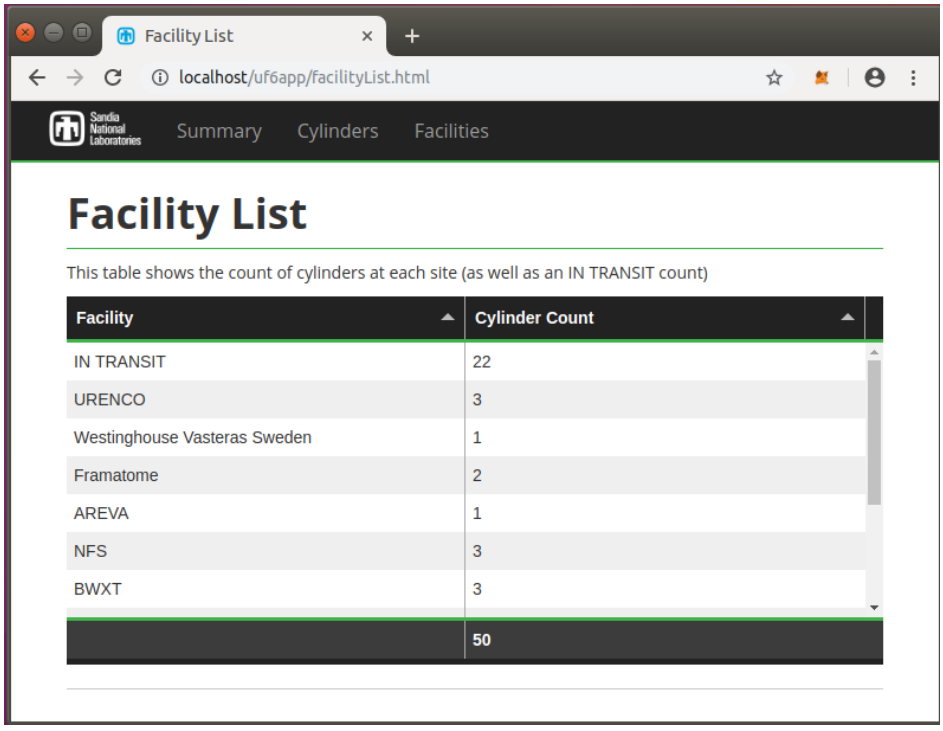

Figure 10. `UF6InventoryRepo` affords efficient querying of a data in a facility-centric fashion, exhibited here. The contract could be easily extended to efficiently provide other views of the underlying data (e.g. retrieve all cylinders currently residing in a particular region).

3.3. The UF_6 Transit Data Simulator

To test and demonstrate the capacity of our prototype we wrote a simplistic UF_6 cylinder shipment simulator relevant to safeguards. The simulator accepts two parameters: number of cylinders C, and number of events to produce E. With these two parameters, our simulator generates a randomized stream of events as follows:

1. Create C random cylinder IDs, and assign each cylinder to be AT REST at a random facility (the simulator currently has a list of 11 facilities)
2. Set T=0 (i.e. simulated time)
3. Repeat until E events are produced
    3.1. With some small probability (e.g. 10%), pick a random AT REST cylinder, and pick a random destination facility. Produce a shipping event indicating SHIPPED, and mark the cylinder IN TRANSIT. Estimate the amount of time the transit will take (based on a naïve distance calculation between facilities), and add to a receiving queue data structure the scheduled arrival time
    3.2. Check the receiving queue data structure and for any cylinder IN TRANSIT for which arrival time has been reached, mark cylinder AT REST at destination facility, and produce a shipping event indicated RECEIVED
    3.3. Increment T+=1 (advance simulated time)

This simulated global routing of UF_6 cylinders is used as the input into our private ethereum DLT.

3.4. The DLT Agents

To support automated, realistic, and repeatable submission of shipping events to a deployed contract, we created a pair of scripts, `boss.py` and `worker.py`, that automatically deploy a contract, and coordinate submission of events across a population of Ethereum nodes. This is achieved by running `boss.py` on a single node, which registers event queues with a `rabbitmq` server (an open-source messaging broker which in this case is just used for coordination between boss and workers), and then waits for the population of workers to check in. Then, `worker.py` is run on each node, resulting in a check-in message per node. Once all nodes are checked in, `boss.py` deploys an instance of `UF6InventoryRepo`, and distributes the resulting contract address to all workers. Then `boss.py` program proceeds to distribute each shipping event to the appropriate worker in sequence (and waiting for confirmation before proceeding to the next), for which `worker.py` in turn takes and submits the corresponding `insertEvent` Ethereum transaction. For example, if we have 11 instances of `worker.py`, each of which corresponds to the 11 facilities in our event simulator as nodes, `boss.py` can direct shipping and receiving events involving a specific facility to arise specifically from the Ethereum node playing the role of that facility.

3.5. FIREWHEEL

FIREWHEEL is a Sandia-developed experiment orchestration framework that assists an experimenter in building, controlling, observing, and analyzing repeatable experiments of distributed systems at any scale. Typically, experiments are instantiated via emulation, but fundamentally

FIREWHEEL is an experiment orchestration tool that is agnostic as to how the experimental definition is converted into a running experiment and on to which execution platform(s) an experiment actually runs (i.e. bare metal, emulation, simulation, or a combination). Most users of FIREWHEEL will be using emulation to launch experiments; however it is important to note that is only one way in which FIREWHEEL can provide value in an Emulytics experiment.

FIREWHEEL is a collection of tools that build upon several mature, open source virtualization and data management technologies. FIREWHEEL enables a researcher to:

- Define an experimental topology programmatically.
- Manipulate an experimental topology programmatically.
- Scalably "compile" the topology to a configured and running set of virtual machines and associated networks in a cluster.
- Manage the execution of in-experiment events and their data inputs and outputs.
- Pervasively instrument the virtual machines and networks involved in an experiment.
- Centrally collect, analyze, and display experimental data.
- Archive an experimental description and reliably repeat an experiment.

FIREWHEEL also includes a growing library of model component objects that ease construction of common topology components like Linux and Windows hosts, routers, Internet services, windows services, client and server applications, and a variety of example topologies ranging from small and simple to large and complicated that can be adapted or extended.

An emulation-based experiment consists of a collection of virtual machines running operating system and application software connected by communications networks. The configuration of machines, software, and networks is driven by the design of the experiment, which in turn, is driven by the question the experiment has been designed to answer. FIREWHEEL enables users to define models of network topologies, as well as any time-scheduled actions they want performed on them. At run-time, FIREWHEEL first represents a topology abstractly, using a graph data-structure, and then instantiates the topology across a Firewheel Cluster of network-connected servers and triggers the execution of scheduled actions at their appointed times.

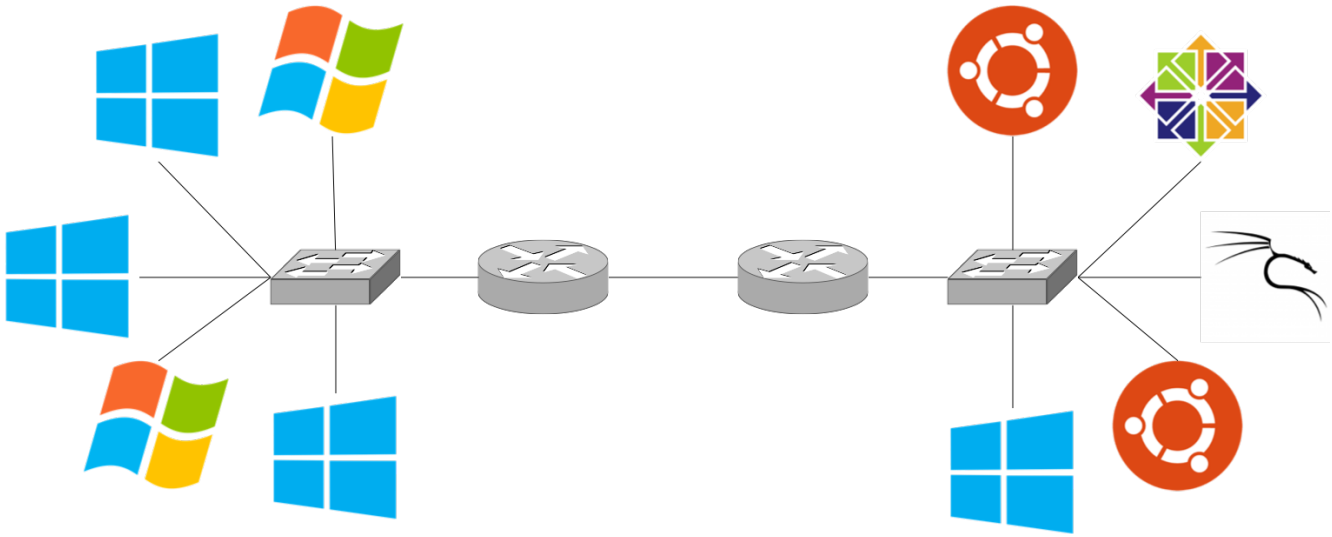

Figure 11. Simple example topology for emulytics across 10 PCs.

For example, imagine an extremely simple experiment consisting of a collection of 10 PCs, running a variety of operating systems and applications, connected via a simple routed network as shown in Figure 11. To implement this experiment, we can run a set of virtual machines (VMs) distributed across three physical hosts as shown in Figure 12. Lastly, we can execute some actions on those VMs which enable answering our research question. For this simple experiment, it would be feasible to manually create and configure the virtual machine disk images, distribute them to the appropriate physical host machines, create the necessary virtual networks, start the virtual machines, and install, configure, and run software applications, but FIRWHEEL provides automations and optimizations easily that can save the manual user's time.

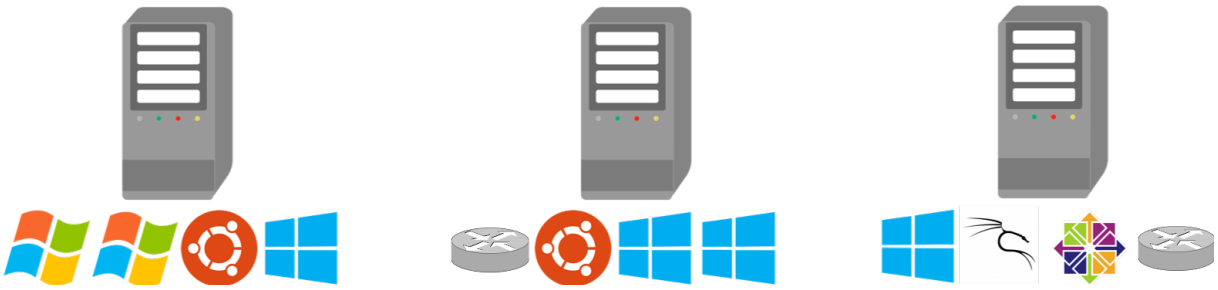

Figure 12. Virtual machines scheduled across a cluster for the simple topology example of Figure 11.

FIREWHEEL was designed to overcome several important shortcomings of manually configuring experiments, such as:

Scalability: While configuring 10 virtual machines connected to a few networks manually is no great chore, if an experiment required hundreds or thousands of virtual machines connected to a similar number of virtual networks, the task would quickly become overwhelming. Such a large number of nodes could easily be the case for a DLT network.

Diversity: If our question of interest requires us to test many variants of our initial configuration, for example to test the sensitivity of a system to a range of environmental parameters, manually managing the experimental complexity quickly becomes untenable.

Repeatability: If, months or years after we perform our experiments, we or other researchers wish to recreate our suite of experiments, a manual approach requires us to expend almost as much energy as during our original effort, even if excellent documentation is available.

Coordination: Experiments rarely consist of configuring and observing collections of idle virtual machines. Arranging the experimental events of interest to occur repeatably at precisely defined times across all the components of a manually defined experiment, especially if that experiment is large, is a challenge.

Instrumentation and data collection: Typically, the reason we run an experiment is to observe a system by collecting, storing, and analyzing detailed measurements. Reinventing this process anew for every manually constructed experiment is wasteful and error-prone. FIREWHEEL enables users

to conduct rigorous research to answer compelling questions about cyber environments while resolving the previously identified shortcomings of existing approaches.

For our UF_6 prototyping environment we leveraged a previously developed model component for deploying an arbitrarily sized Ethereum network. As such our work was limited to the three components previously described: the `UF6InventoryRepo.sol` contract, a notional web application, and a means for simulating and submitting data to the contract in a realistic fashion across the population of Ethereum nodes. Figure 13 displays one of the attributes that you get 'for free' when using FIREWHEEL, specifically an experiment-wide view of events occurring in virtual machines (in this case transaction events that are being submitted to the contract from various nodes).

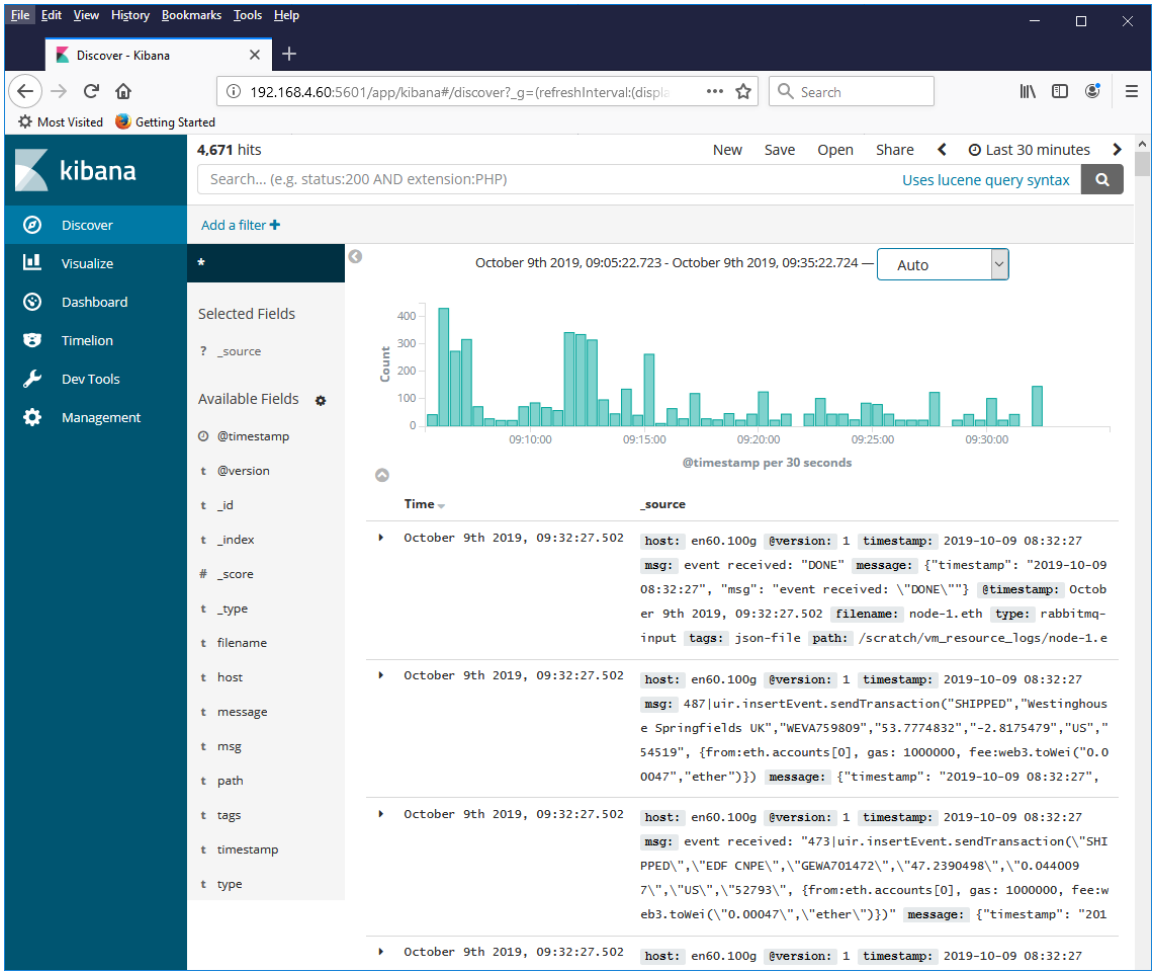

Figure 13. Screenshot of the Kibana elastic search visualization tool, here displaying a global view of events coming from a deployed FIREWHEEL instance simulating UF_6 movement.

3.6. More Detailed Walkthrough

Our UF_6 cylinder tracking DLT prototype consists of the following steps:

1) Any node on the private Ethereum network deploys a compiled `UF6InventoryRepo` contract to the blockchain, as illustrated in Figure 14. The contract script listed in Appendix A compiles to under 5,000 bytes (~5Kbytes). Once the contract is deployed to the blockchain, it can be interacted with by any node on the network via its contract address, e.g. `0xbdcfcf211a273b67e7cad537f17501667958442c`

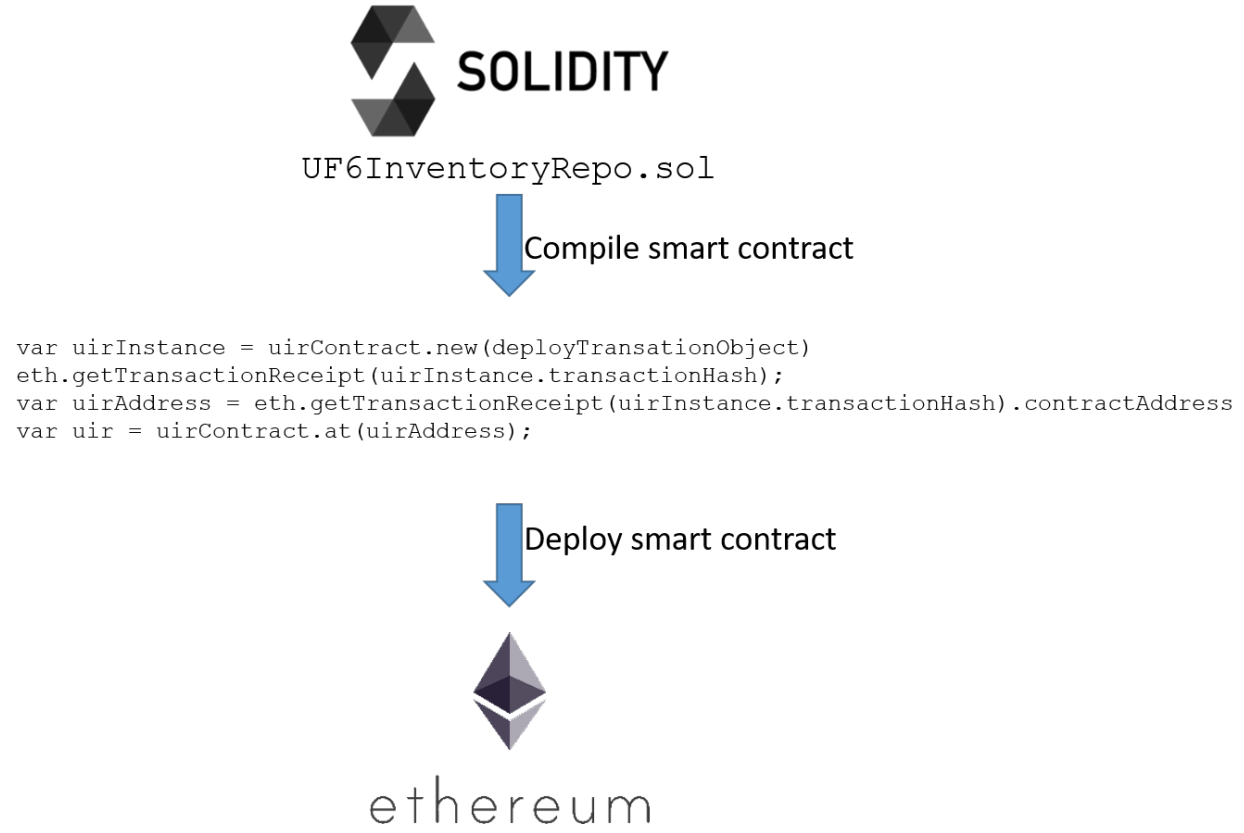

Figure 14. A depiction of the contract compile and deploy process.

2) At this point nodes may start submitting shipping/receiving events to the contract, which will result in network-wide updates to the state. For example, if the contract has not been used, issuing a transaction with the following data will result in the subsequently shown data.

```
>
uir.insertEvent.sendTransaction("SHIPPED","URENCO","TNXA452809","
32.4330195","-103.0797433","US","8587", {from:eth.accounts[0],
gas: 1000000, fee:web3.toWei("0.00047","ether")})
"0x10d0e237e14f87edddc00896e3c1a27b042974678b54bbcac84fe39b1b3e29
7b"
> uir.getNumEvents()
1

> uir.getNumCylinders()
1
```

```
> uir.getLatestEvent("TNXA452809")
"SHIPPED,URENCO,32.4330195,-103.0797433,US,8587"

> uir.getNumCylindersInTransit()
1
```

The above results indicate that a single cylinder has been shipped, and is IN TRANSIT. Subsequently a second "received" event is entered per below:

```
>
uir.insertEvent.sendTransaction("RECEIVED","AREVA","TNXA452809","
43.497161","-112.0755061","US","3305", {from:eth.accounts[0],
gas: 1000000, fee:web3.toWei("0.00047","ether")})
> uir.getNumEvents()
2

> uir.getNumCylinders()
1

> uir.getLatestEvent("TNXA452809")
"RECEIVED,AREVA,43.497161",-112.0755061,12234"

> uir.getNumCylindersInTransit()
0
```

3) As an example of more realistic data conditions, we executed steps 1 and 2 above on a simulated dataset consisting of 20,000 unique cylinder identifiers, and 75,000 distinct shipping/receiving events (which is roughly six months of events per today's system load). The process for generating the, e.g. 75,000 `insertObservation` events, is depicted in Figure 15. After the simulation has completed, the user can query any aspect of the various transactions as stored in the DLT, such as

```
> uir.getNumEvents()
75000
> uir.getNumCylinders()
16957
> uir.getLatestEvent ("TNXA452809")
"RECEIVED,Westinghouse Columbia USA,33.882558,-
80.921731,US,7261131"
> uir.getNumCylindersInTransit()
106
>
```

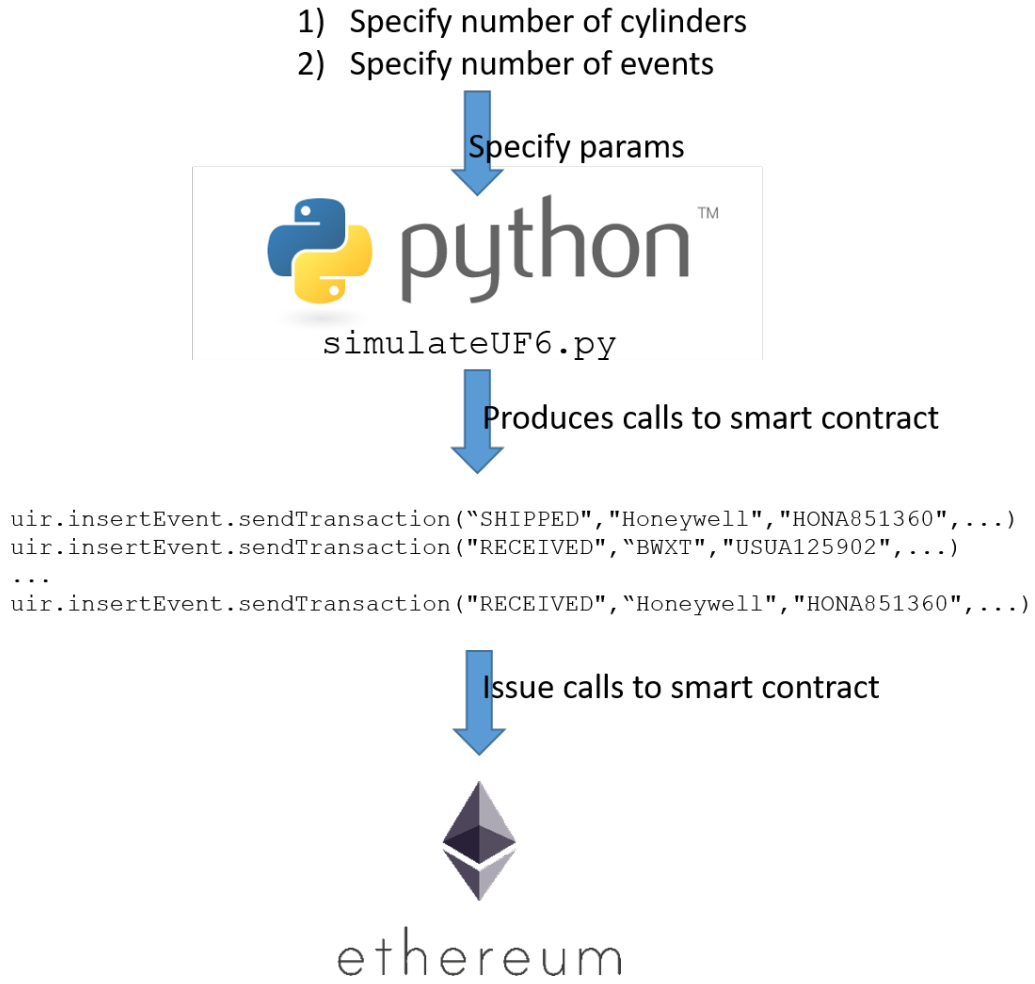

Figure 15. A depiction of the process to generate simulated UF_6 movement events, and submit them to a deployed `UF6InventoryRepo` contract.

We note that the above steps and terminal script snippets may not be convenient for an IAEA analyst. However, it is also straightforward to interact with the contract as a back-end datastore for a GUI approach as was illustrated in Figs. 7-10.

4. DISCUSSION

We have developed a distributed ledger prototype for the UF_6 cylinder tracking safeguards use-case utilizing Ethereum DLT technology in permissioned (private) mode that creates a simulated random sequence of UF_6 cylinder events (e.g. shipping and receiving), stores all events in our DLT, and we provide a GUI mechanism to query all aspects of the blockchain data. We have written source code using the Solidity framework for DLT contracts, specifically the `UF6InventoryRepo.sol` code (see Appendix A). Further, we have used emulytics technology developed at Sandia to connect nodes of virtual machines of the DLT network utilizing the FIREWHEEL framework. We show examples of our DLT in action using the simulated data of 75,000 UF_6 cylinder transactions, and find that the IAEA should easily be able to handle such a permissioned DLT for the UF_6 cylinder tracking safeguards use-case, both from a code sustainability perspective (i.e. the code implementation utilizes user-friendly GUI interfaces) and from a data storage perspective (tracking 150,000 UF_6 cylinders is not an overwhelming data load for modest server requirements).

4.1. UF_6 Use-Case Value Proposition

It is important and clarifying to realize that from a functionality perspective, DLT in the UF_6 setting is essentially no different than what is achievable with a standard or distributed database. The major value of DLT is in the security and integrity properties of the stored data, specifically in reducing the opportunity for facilities to manipulate a shared database to hide cylinder diversion [2]. Additionally, DLT may aid in making progress on unifying UF_6 data practices by a community of mutually distrusting data providers/writers. All Member States involved in the IAEA-managed DLT could have access to the DLT-stored immutable data, and the IAEA can restrict access as the Central Authority for any privacy concerns.

In the nomenclature of [10], our prototype is a "DLT-backed database", which is broader in scope than a traditional SQL database as the users cannot just use SQL tools but must also abide by DLT rules for writing data to the data-store and consensus mechanisms. If a DLT solution must interact with a legacy SQL database, then complications arise in the implementation, one solution for which is to create a cloud-based "DLT-backed table". As this demo did not require integration with any known legacy solution, our DLT-backed database approach for this prototype is an acceptable choice. However, depending on the nature of the IAEA's existing database for UF_6 cylinder tracking, or other databases for different safeguards use-cases, this prototype may need further work to incorporate legacy data systems into a working DLT-backed database.

Another system worth further exploring in this use case is TradeLens[2], a partnership between IBM and Maersk, which has made headway in a DLT-backed approach to international shipping data provenance. An argument for UF_6 relevance of TradeLens is argued in [11]. As of July 2019, Hapag-Lloyd, Singapore-based Ocean Network Express (ONE), CMA CGM, and MSC Mediterranean Shipping Company have announced they are joining TradeLens. With these additions, the scope of the platform now extends to more than half of the world's ocean container cargo. Members of TradeLens will have a comprehensive view of their data and can collaborate as cargo moves around the world, helping create a transparent, secured, immutable record of transactions. While the IAEA must have its own DLT for its purposes, it may be beneficial in the future to create a system for the IAEA that integrates with TradeLens.

[2] http://www.tradelens.com

The prototype described in this report is agnostic to whether the UF_6 community adopts the concept of a global identifier, such as Global ID [7]. As long as cylinders have some unique identifier, our prototype applies. Further, as alluded before, the DLT data may provide means to extract cylinder identity when an ID is unavailable, since the DLT should immutable and all transactions are recorded.

4.2. Pitfalls, and barriers to integration

There are risks to deploying a DLT solution in any domain at the current point in time. For UF_6 cylinder tracking we assess the largest risks having to do with talent, development tools, regulation and teamwork, as described below:

- Talent: There is a dearth of blockchain technical specialists, especially as compared with the relative abundance of database engineers and administrators
- Standards: There is no DLT standard that would compel DLTs to work together
- Development tools: it is still very early days for DLT and frameworks are varied such that it is very much a user choice of which technology to use, and how well these tools are supported
- Regulation: while likely mostly applicable to other DLT use-cases (e.g. cryptocurrency), there is uncertainty regarding regulatory requirements for businesses leveraging DLT, and it is even less clear of the policy requirements for the IAEA to be allowed to implement DLT with Member States
- Teamwork: one of the upsides of DLT is in not having to fully trust the various data writers to the network/ledger. That said, organizations will need to work together to the extent required to keep the peer to peer network operational. A good DLT has a consensus mechanism that works even when nodes are inoperable or malicious, but the IAEA needs to put in place a procedure for Member States to become part of its DLT and how to maintain their nodes.

4.3. Data Privacy

The prototype described in this report focuses on proof-of-concept and technical feasibility/scalability of a UF_6 shipping/receiving tracker that leverages a distributed ledger. There is another important design requirement that is not explored, but which likely must be addressed as a prerequisite for adoption – specifically, *data privacy/confidentiality*. In other words, facilities worry about outsiders' ability to infer proprietary information from shipping/receiving information. Alas, the current situation regarding UF_6 shipping/receiving [2] is in large part due to facility reluctance to trust any single party with protecting this information. While we have not addressed this design dimension in our prototype, it is likely that this information is more straightforward to protect than it would be with a more traditional database approach. We now describe a few approaches for achieving this vision, of which further exploring is left as future work.

A distributed ledger is replicated, and thus there is not reliance/trust placed on any single entity, i.e. a community need not decide/agree on an organization/platform to host/aggregate data. This attribute does not directly address facility worries about being able to infer proprietary information, but, rather than one external entity (the database provider) having sole control over the aggregated view of many facilities' data, now many external entities have a common view over the aggregated view. An upside, however, is that all parties can see precisely how their own data is manifested in the datastore, i.e. it is not behind/managed by an opaque database provider. This enhances confidence,

on the part of each organization, in the integrity of the data regarding their facility, which represents a step forward in addressing data modification threats.

Addressing the proprietary data inference threat, even under the replicated data (DLT) approach, is likely achievable via creative applications of cryptography, specifically via a combination of storing mostly encrypted data (for which only the facility owning the data can decrypt), along with select unencrypted derived metadata, of which the metadata is aggregated to achieve the improved global situational awareness covered by the rest of this report. The metadata is cryptographically reliant upon the encrypted facility data, such that facilities are effectively attesting to the accuracy of the metadata. An example may help illustrate this idea:

For example, a facility ships a cylinder, and

1. The facility encrypts the shipment information (details about contents of cylinder, weight, shipping destination, etc.);
2. The facility publishes the encrypted data to the DLT;
3. The facility cryptographically signs metadata consisting of the hash of the shipment data, along with e.g. an unecrypted statement such as 'this is a shipment,'
4. The facility publishes the signed metadata to the DLT.

Now all parties can confidently update their awareness of the number of cylinders in transit worldwide. Further, upon an incident, the shipping organization could selectively reveal underlying data to support audit, with the added benefit of strong integrity and authenticity properties of the reveald data. A draft standard, called encrypted data vaults (cite https://digitalbazaar.github.io/encrypted-data-vaults/), is one scheme which could likely support the approach described in this paragraph.

Another data security avenue to consider is the addition of cryptographic secret-sharing protocols, such as secure Multi-Party Computation (MPC) [12] [13]. With MPC, the underlying raw data of parties is never revealed to any other party. Rather, each parties' data is converted into random number "tokens" that represent the raw data, and these tokens are used in an agreed function to produce a result that all parties can see. The "function" can be any function that represents an outcome or goal, such as "do these ID's match?", and such an agreed-upon function can then be converted to a digital circuit for use with above-mentioned tokens in MPC protocols. Note that the function and the mechanics of the cryptographic procedures are completely transparent to all parties, keeping in-line with the transparency intent of a DLT. The blockchain and privacy-preservation issue is a known issue to the commercial sector, so efforts are already ongoing to develop hybrid MPC-DLT solutions that may find applicability to IAEA use-cases [14].

In closing, achieving some of the ideas discussed in this section, specifically those addressing facility concerns about the inference of proprietary information, are likely achievable whether or not DLT is used as the datastore. For example, the encrypted but attested to scheme, could also be used in conjunction with a traditional database as the datastore. That said, using a traditional database would likely compromise some of the information integrity and authenticity benefits described above. Exploring this tradeoff space is also left as future work.

4.4. Next Steps

We believe a useful step forward will be to demonstrate this DLT prototype to the IAEA. We could provide a GUI for the IAEA to try and walk them through the code, which we believe will relax

many concerns regarding the sustainability of the DLT and code infrastructure. Pending the outcome of this demonstration, we could create other prototypes for other safeguards use-cases.

We believe a logical next step if a working UF_6 cylinder tracking DLT is deemed to be worthy of further pursuit is to engage TradeLens representatives on how to integrate an IAEA DLT with their DLT framework. This decision also depends on how much Member States rely on the international shipping industry to transport UF_6 cylinders. Another avenue for in-country UF_6 cylinder transport could be to demonstrate our DLT integration with the existing tracking mechanism of a willing Member State participant. This solution may require a DLT-backed table to bridge our DLT with legacy SQL databases that may be in use by the Member State.

In short, we have developed a working private DLT applicable to IAEA needs and see no computational roadblocks to further utilization by the IAEA and Member States. Other DLT approaches of course can and should be considered beyond our specific implementation using Ethereum/Solidity, but we believe all approaches will arrive at a similar conclusion that a DLT solution for the IAEA makes sense and is practical. The next steps should involve the IAEA or at least a real-world scenario such as integration with the TradeLens DLT.

5. REFERENCES

[1] S. L. Frazar, K. D. Jarman, C. A. Joslyn, S. J. Kreyling, A. M. Sayre, M. J. Schanfein, C. L. West and S. T. Winters, "Exploratory study on potential safeguards applications for shared ledger technology," Pacific Northwest National Laboratory, Richland, 2017.

[2] S. L. Frazar, C. A. Joslyn, R. K. Singh and A. M. Sayre, "Evaluating Safeguards Use Cases for Blockchain Applications," Pacific Northwest National Laboratory, Richland, 2018.

[3] J. Whitaker, J. White-Horton and J. Morgan, "Preliminary Concept of Operations for a Global Cylinder Identification and Monitoring System," Oak Ridge National Laboratory, Oak Ridge, 2013.

[4] O. Heinonen, "Why the Monitoring of Movements of UF6 Cylinders Matters.," in *Global Cylinder Identification and Monitoring System Stakeholder Meeting*, Washington, DC, 2014.

[5] C. Vestergaard, "Better than a floppy: The potential of Distributed Ledger Technology for nuclear safeguards information management," *Policy Analysis Brief,* pp. 1-8, October 2018.

[6] American National Standard for Nuclear Materials, "Uranium Hexafluoride - Packaging for Transport," American National Standards Institute, New York, 2001.

[7] WNTI, "UF6 Cylinder Identification Standard," London, 2017.

[8] M. M. Curtis, "NGSI: IAEA Verification of UF6 Cylinders," PNNL-SA-88399, Richland, WA, 2001.

[9] A. Antonopoulos and G. Wood, Mastering Ethereum: Building Smart Contracts and DApps, O'Reilly, 2018.

[10] L. Allen, P. Antonopoulos, A. Arasu, J. Gehrke, J. Hammer, J. Hunter, R. Kaushik, D. Kossmann, J. Lee, R. Ramamurthy and S. Setty, "Veritas: shared verifiable databases and tables in the cloud," in *9th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2019.

[11] P. Gasser, "A Distributed Ledger Technology (DLT) Approach to Monitoring UF6 Cylinders: Lessons Learned from TradeLens," in *Institute of Nuclear Materials Management 60th Annual Meeting*, Palm Desert, 2019.

[12] A. Solodov, D. Farley, C. Brif, N. Pattengale, Y. Gao, J. Lin, M. Negus and R. Slaybaugh, "Development of Novel Approaches to Anomaly Detection and Surety for Safeguards Data," in *Institute of Nuclear Materials Management 60th Annual Meeting*, Palm Desert, 2019.

[13] A. C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science*, Toronto, 1986.

[14] A. A.-T. Innocent and G. Prakash, "Blockchain applications with privacy using efficient Multiparty Computation protocols," in *IEEE 2019 PhD Colloquium on Ethically Driven Innovation and Technology for Society (PhD EDITS)*, Bangalore, 2019.

[15] D. Yaga, P. Mell, N. Roby and K. Scarfone, "Blockchain technology overview," National Institute of Standards and Technology, 2018.

APPENDIX A. ETHEREUM SMART CONTRACT

```
contract UF6InventoryRepo {

    uint numEvents;

    mapping(bytes32=>uint) numEventsByCylinder;
    mapping(bytes32=>string[]) cylinderHistories;
    mapping(uint=>string) cylinderIDs;
    uint numCylinders;

    mapping(bytes32=>int) numCylsByFacility;
    mapping(uint=>string) facilityIDs;
    uint numFacilities;

    uint numInTransit;

    bytes32 shippedStatic = keccak256(abi.encodePacked("SHIPPED"));
    bytes32 receivedStatic = keccak256(abi.encodePacked("RECEIVED"));

    function insertEvent (
           string memory action, //e.g. SHIPPED or RECEIVED
           string memory reporter, //the organization reporting the event (the shipper/receiver)
           string memory cylinderID, //the UID for the cylinder
           string memory lat, //geographical latitude
           string memory long, //geographica longitude
           string memory flag, //state with accountability for cylinder
           string memory timestamp //unix timestamp for event
    ) public {
             bool seencyl = true;
             bytes32 cylinderBID = keccak256(abi.encodePacked(cylinderID));
             if(numEventsByCylinder[cylinderBID]==0){
               cylinderIDs[numCylinders] = cylinderID;
               numCylinders+=1;
               seencyl = false;
             }
             numEventsByCylinder[cylinderBID] += 1;
             numEvents += 1;

             //this may break if encodePacked changes in the future to not just do concatenation
             string memory h =
           string(abi.encodePacked(action,",",reporter,",",lat,",",long,",",flag,",",timestamp));

             cylinderHistories[cylinderBID].push(h);

             bytes32 reporterBID = keccak256(abi.encodePacked(reporter));
             bytes32 actionBID = keccak256(abi.encodePacked(action));
             if(numCylsByFacility[reporterBID] == 0){
                facilityIDs[numFacilities] = reporter;
                numFacilities += 1;
                //see note below about weirdness
                numCylsByFacility[reporterBID] = -1;
             //this is a little weird, we need to use zero to indicate
             //that we've not seen this location before, and so we use -1 as our zero
             if(actionBID == receivedStatic){
                //increment receiving facility count (-1 and 0 => 1, everything else i+1)
                if(numCylsByFacility[reporterBID] > 0){
                        numCylsByFacility[reporterBID] += 1;
                } else {
                        numCylsByFacility[reporterBID] = 1;
                }
                //decrement intransit
                numInTransit -= 1;
             } else if (actionBID == shippedStatic){
                //increment intransit
                numInTransit += 1;
                //decrement shipping facility count if we've seen it before
                //(1 and 0 => -1, everything else i-1)
                if (seencyl){
                        if(numCylsByFacility[reporterBID] > 1){
                                numCylsByFacility[reporterBID] -= 1;
```

```
                        } else {
                                numCylsByFacility[reporterBID] = -1;
                        }
                    }
                }
    }

    function getNumCylinders() public view returns (uint) {
             return numCylinders;
    }

    //index must be between 0 and getNumCylinders-1, and iterating
    //through those indices will return all cylinderIDs
    function getCylinder (uint index) public view returns (string memory){
        return cylinderIDs[index];
    }

    function getHistoryLength (string memory cylinderID) public view returns (uint){
             bytes32 cylinderBID = keccak256(abi.encodePacked(cylinderID));
             return cylinderHistories[cylinderBID].length;
    }

    //index must be between 0 and getHistoryLength-1, and iterating
    //through those indices will return all events for this cylinder
    function getHistoryEntry (string memory cylinderID, uint index) public view returns (string
       memory){
             bytes32 cylinderBID = keccak256(abi.encodePacked(cylinderID));
             return cylinderHistories[cylinderBID][index];
    }

    //convenience method to retrieve latest event for given cylinder
    function getLatestEvent(string memory cylinderID) public view returns (string memory){
             return getHistoryEntry(cylinderID, getHistoryLength(cylinderID)-1);
    }


    function getNumFacilities() public view returns (uint) {
             return numFacilities;
    }
    //index must be between 0 and getNumFacilities-1, and iterating
    //through those indices will return all facilities
    function getFacility (uint index) public view returns (string memory){
        return facilityIDs[index];
    }

    //total number of events received by system
    function getNumEvents () public view returns(uint){
             return numEvents;
    }

    function getNumCylindersAtFacility(string memory facility) public view returns (uint) {
             bytes32 facilityBID = keccak256(abi.encodePacked(facility));
             int n = numCylsByFacility[facilityBID];
             if(n < 0){
                return 0;
             } else {
                return uint(n);
             }
    }

    function getNumCylindersInTransit() public view returns (uint) {
             return numInTransit;
    }

}
```

APPENDIX B. DLT USE CASE CONSIDERATIONS

The National Institute of Standards & Technology (NIST) provides a flowchart for "Do you need a blockchain" [15]. From the NIST documentation, they state that blockchain technology solutions may be suitable if the activities or systems require features such as:

- Many participants
- Distributed participants
- Want or need for lack of trusted third party
- Workflow is transactional in nature (e.g., transfer of digital assets/information between parties)
- A need for a globally scarce digital identifier (i.e., digital art, digital land, digital property)
- A need for a decentralized naming service or ordered registry
- A need for a cryptographically secure system of ownership
- A need to reduce or eliminate manual efforts of reconciliation and dispute resolutions
- A need to enable real time monitoring of activity between regulators and regulated entities
- A need for full provenance of digital assets and a full transactional history to be shared amongst participants

We addressed the NIST criterion applied to the UF_6 cylinder tracking safeguards use-case below in Figure 16, where green-shaded blocks indicate affirmative utility of blockchain technology, and yellow indicates dubious justification. In our opinion, which also confirms the previous work by PNNL [1] [2], the UF_6 cylinder tracking safeguards use-case does warrant utilization of a DLT, or blockchain.

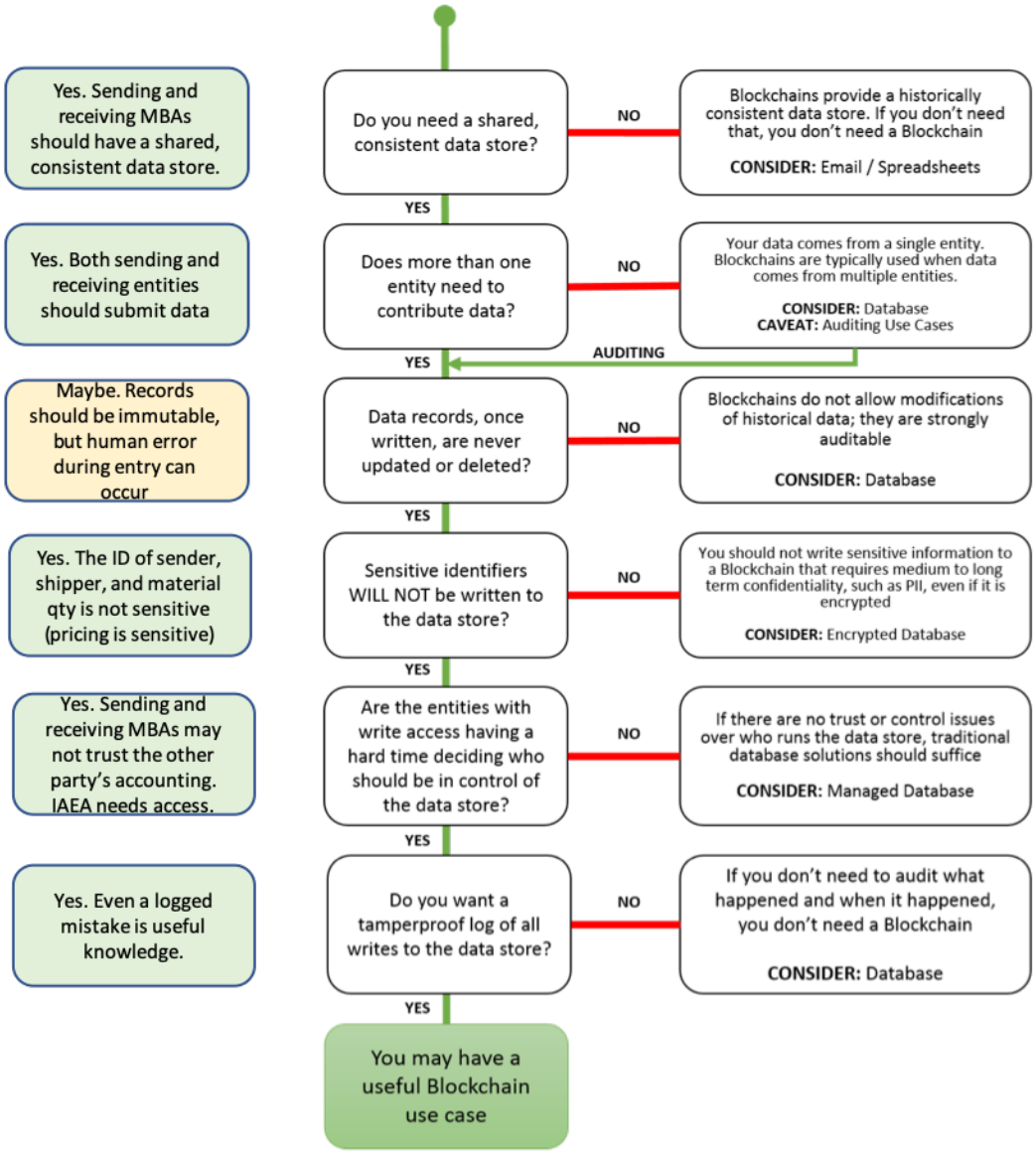

Figure 16. NIST flow chart for "Do you need a blockchain" as answered by the authors for the case of UF_6 cylinder tracking.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Tina Hernandez	06832	therna@sandia.gov
Zoe Gastelum	06832	zgastel@sandia.gov
Nick Pattengale	05882	ndpatte@sandia.gov
David Farley	08648	dfarley@sandia.gov
Technical Library	01177	libref@sandia.gov

Email—External [redacted]

Name	Company Email Address	Company Name
Sarah Frazier	Sarah.Frazar@pnnl.gov	PNNL
Cliff Joslyn	Cliff.Joslyn@pnnl.gov	PNNL

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.