Controlled Item Inventory using Ethereum Smart Contracts

Preprint · May 2019

DOI: 10.13140/RG.2.2.17455.38562/1

1 author:

Justin Vermillion
New Mexico Institute of Mining and Technology

Controlled Item Inventory Using Ethereum Smart Contracts

Justin Vermillion
justin.vermillion@student.nmt.edu
New Mexico Institute of Mining and Technology
Socorro, NM

Abstract

In order to modernize the existing inventory process for maintaining accountability of controlled items, paper and wet signature processes should be augmented or replaced with digital means. While there are reasonable security concerns surrounding adoption of new methods for this time of inventory control, modern blockchain technologies have undoubtedly demonstrated their ability to mitigate chain-of-custody and tampering/forgery risks. This paper presents a proof-of-concept system for digitally maintaining controlled item inventory, and sets forth a direction for potential future research and development.

Keywords

blockchain, smart contracts, asset inventory, chain-of-custody

1 Introduction

The United States Government maintains stockpiles of thousands of nuclear weapons, and hundreds of thousands of pounds of weapons-ready nuclear materials. Other governments, such as China and Russia, also maintain their own stockpiles. [1, 10] One very prominent risk associated with such stockpiles is that fissile materials and their derivatives could potentially fall into the hands of terrorists or adversarial nations. [2, 6] Further, many nations have signed on to Nuclear Non-Proliferation Agreements, which are essentially contracts committing those nations to reducing and entirely eliminating their nuclear stockpiles. These treaties are difficult to enforce because of the lack of a secure and verifiable inventory system. [8]

This project aims to create a prototype inventory system for such materials, which will hopefully enable responsible agencies to maintain chain-of-custody and materials security without the need for paper and wet signature and the additional overhead (such as storage, auditing, etc.) that come with those. The prototype demonstrates a minimal level of functionality. However, the development was limited in scope, primarily due to restrictions around time– the project had to be completed within the span of a single semester, and the proposal was not approved until nearly mid-way through the course.

2 Approach

2.1 Requirements

As discussed in the introduction, the subject matter of this project could be especially sensitive, from a national and international security perspective. Therefore, it was critical to identify the key needs of the nuclear industry, and establish a set of requirements to meet those needs as closely as possible. In order to produce a reasonable set of requirements, it was necessary to meet with experts in the area and gather their input. These discussions led to the enumeration of four key areas of focus, which because the requirements that the prototype was built toward.

Auditability The first key requirement was auditability. Most controlled items have some sort of oversight around their use and possession, and this is especially true for fissile materials. Various organizations, such as the Department of Defense, the Department of Energy, or the Office of the Inspector General, are interested in knowing every detail about each sample. A high level of auditability means that all transactions are recorded and stored in a database of some type, and made available for auditors to review upon request.

Accountability Because of the concern around the risk of fissile materials falling into the hands of adversaries, it is necessary to track the movement of each sample with a high level of detail. This type of of accountability is often referred to as *chain of custody*. Every time the asset changes hands, that transfer must be documented. In a given experiment, a single sample could be handled by multiple researchers, each of which needs to be recorded in the transfer log, to ensure that the sample hasn't been tampered with, replaced, or outright stolen.

Ease of Use Fulfilling the previous two requirements can be complex, and require quite a bit of documentation. The concept of *ease of use* means that those processes should be as transparent to the user as possible, so that their interaction with the tool is as uncomplicated as possible. Any potential replacement for paper and wet signature processes must be less intrusive, allowing the users to work more efficiently and with fewer interruptions to their workflow. Maintaining auditability and accountability should be as easy for the user as simply scanning a barcode or tapping a button.

Integrity Finally, transactions should be secure from forgery or modification. Every single transaction for every sample should be publicly and independently verifiable as being authorized and authentic. Transactions should not be able to be modified either in-transit or after submission, and new transactions should only be able to be created by authorized users.

2.2 Proposed Solution

Based on the requirements listed above, it was easy to find a basic solution to the overall problem. Blockchain technology allows for complete auditability, accountability, and integrity of transactions. To address ease of use, it was decided to implement a modern, reflexive web application that can be used on any network-connected device. During development, it became apparent that plain blockchain transactions were not going to be acceptable to represent tangible goods. To resolve this issue, the Ethereum network's concept of *Smart Contracts* was adopted.

Ethereum and Smart Contracts Ethereum is an extended version of blockchain technology. While it does have its own currency, called ETH, it is also programmable, which makes it much more flexible than other blockchain technologies for non-currency use cases.[5] One aspect of this programmability is the *Smart Contract*. A *Smart Contract* is a piece of code that can be executed on the blockchain, which can establish and enforce arbitrary rules (and resulting actions) on digital assets. [3, 9]

ERC-721 Standard In order to translate physical assets, such as fissile material samples, into digital assets that can be managed by *Smart Contracts*, a tokenization scheme needed to be adopted. A key feature of this possible scheme is the concept of non-fungibility. An asset is considered fungible if it can be interchanged with any other asset of the same class without issue. A good example of a fungible asset is a dollar bill. Any one dollar bill can be exchanged for any other dollar bill, and the owner will still have an asset worth exactly one dollar, that can be exchanged for goods or services. Controlled items are, however, non-fungible. Sample 1 could not reasonably be substituted with Sample 2 without issue.

Luckily, quite a bit of work has been done in the space of tokenizing physical assets, and a non-fungible token standard was created by a team of developers, called ERC-721. [4] This standard allows for creating new tokens, transferring ownership of tokens, and decommissioning existing tokens. This is perfect for representing unique physical assets as equally unique digital assets, and is the standard that was adopted for this project.

2.3 Technical Implementation and Challenges

Smart Contracts are implemented in a proprietary programming language called Solidity. Having no experience in writing in Solidity became quite a challenge during the development of this prototype. Luckily, some base templates for a wide variety of token implementations have been made available by a group called OpenZeppelin (Appendix A). This made development of the token relatively straightforward–a sample can be found in Appendix B.

The web application itself was written and deployed in Python 3, using the Flask framework. Flask makes development of web apps fairly easy by breaking down the application into *routes*, each of which represents a specific URI path, and can be programmed to perform behavior independent of each other. The results of the code execution for each route are displayed to the user via HTML templates.

In order for the web application to interact with the blockchain, two third-party libraries were necessary: **Web3.py** and **py-solc-x**. Web3 is an Ethereum API wrapper developed specifically for Python applications. It exposes all of the Ethereum API to your Python application without having to make complicated websocket calls. This is accomplished by instatiating the blockchain as a Python object, which possesses methods for each potential API call.

For compiling the Solidity source code and deploying it to the blockchain, the py-solc-x library was invaluable. This obviates the need to maintain a separate store of compiled bytecode. The library also allows the raw Solidity code to be defined as a simple text block within the body of the Python application.

From within the application, users can onboard new assets by creating a new token and providing some basic metadata about the asset (currently, only an Asset ID and the owner's wallet address). They can also manage existing assets, either transferring them to a new owner, or decommissioning the asset entirely. The interface is modern and familiar–thanks in part to **bootstrap.js**–and maintains ease of use by reducing the overall number of interactions. There are currently only three buttons, two screens, and a single list of token links.

3 Discussion

3.1 Future Work

While the prototype application implements the specified requirements at a very basic level, there is quite a bit of work to do in order to make the application fully ready for production use. First, a full transaction history for each token should be made available within the application. As it stands, in order to get information about an individual token, then blockchain has to be queried directly, and the relevant information must be manually extracted. Second, users are currently hard- coded in the application. A production-ready system must have a dynamic user database that supports both user onboarding and user offboarding.

Additionally, the application only collects the minimum amount of metadata required to create and assign ownership to a token. For full usability, more information must be stored for each asset. This information includes, but is not limited to, the type of isotope (if inventorying fissile material), acquisition date of the sample, its provenance, weight, storage location, and so on. The ERC-721 standard does provide support for collecting this metadata, but it was outside of the scope of this project.

Finally, the issue of authorization must be sorted before the application could ever be used in a production environment. Currently, the application connects to the blockchain and conducts transactions as the coinbase account, which means that any user can perform any action on any asset. Authorization needs to be tightened down so that users can only control their own assets. It may also make sense to designate a smaller subset of users who can onboard new assets, and perhaps add an approval process for decommissioning existing assets.

3.2 Unsolved Problems

One issue that was not addressed by this project, and still remains outstanding, is how to package the application in such a way that the security of the private blockchain cannot be compromised. With a network of limited size, any malicious actor who is able to add a node could potentially compromise the entire network. This stems from our knowledge that any network relying on consensus (such as blockchain) can be compromised if more than 1/3 of the nodes are malicious (this is a very reductionist view of the Byzantine Generals Problem). [7] Ensuring security in this space is going to be critical to gaining adoption, especially among the global nuclear security community.

4 Conclusions

There is no question that blockchain is a useful technology for maintaining digital records that, up to now, have had to utilize wet signature. Ethereum's concept of *Smart Contracts* extends this usefulness immensely. As the prototype created for this project demonstrates, it is a trivial task to implement a basic inventory control system using this technology. However, some issues do remain to be ironed out. The primary roadblock to implementing this a similar system in the nuclear industry is very likely going to be security of the private blockchain itself.

Acknowledgments

To Andrew Bowman and Kirk Thompson, for helping to brainstorm ideas for the project and dealing with my *prima donna* attitude.

To American Airlines, for delaying and canceling so many flights. Without their help, I would not have felt the crunch of the submission deadline so clearly.

To the pack rat that chewed up the wires in my car twice in one week, setting me back approximately $1400. Rest in peace.

References

[1] Graham T Allison, Owen R Cote Jr, Richard A Falkenrath, Steven E Miller, et al. 1996. *Avoiding Nuclear Anarchy: Containing the Threat of Loose Russian Nuclear Weapons and Fissile Material.* MIT Press, Cambridge, MA.

[2] Matthew G Bunn. 2000. *The Next Wave: Urgently needed new steps to control warheads and fissile material.* Carnegie Endowment for International Peace, Washington, DC.

[3] Vitalik Buterin. 2017. A Next-Generation Smart Contract and Decentralized Application Platform. Retrieved May 4, 2019 from https://github.com/ethereum/wiki/wiki/White-Paper/f18902f4e7fb21dc92b37e8a0963eec4b3f4793a

[4] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. 2018. ERC-721 Non-Fungible Token Standard. Retrieved April 16, 2019 from https://eips.ethereum.org/EIPS/eip-721

[5] Ethereum. 2019. Beginners: Welcome! Retrieved May 4, 2019 from https://www.ethereum.org/beginners/

[6] Siegfried S Hecker. 2006. Toward a comprehensive safeguards system: Keeping fissile materials out of terrorists' hands. *The Annals of the American Academy of Political and Social Science* 607, 1 (2006), 121–132.

[7] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.

[8] Zia Mian, Tamara Patton, and Alexander Glaser. 2017. Addressing Verification in the Nuclear Ban Treaty. *Arms Control Today* 47, 5 (2017), 14–22.

[9] Jeremy M Sklaroff. 2017. Smart contracts and the cost of inflexibility. *U. Pa. L. Rev.* 166 (2017), 263.

[10] Frank von Hippel and Oleg Bukharin. 2018. US-Russian Cooperation on Fissile Material Security and Disposition. In *Arms Control and Disarmament.* Springer, New York City, 255–272.

A Online Resources

- **Project Source Code Repository**
  - https://github.com/vermi/CSE589/tree/master/project
- **OpenZeppelin Smart Contract Templates**
  - https://github.com/OpenZeppelin/openzeppelin-solidity
- **Go Ethereum Distribution**
  - https://geth.ethereum.org/

B Smart Contract Example

```
pragma solidity ^0.5.7;

import 'openzeppelin-solidity/contracts/
    token/ERC721/ERC721Mintable.sol';
import 'openzeppelin-solidity/contracts/
    token/ERC721/ERC721Burnable.sol';
import 'openzeppelin-solidity/contracts/
    token/ERC721/ERC721Enumerable.sol';

contract NukeToken is ERC721Burnable,
    ERC721Mintable, ERC721Enumerable {
     uint8 public decimals = 18;
}
```