Secure and Auditable Academic Collections Storage via Hyperledger Fabric-Based Smart Contracts

Thomas Atkins

B.S in Computer Engineering

Submitted to the graduate degree program in Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master Of Computer Engineering.

Committee members

Drew Davidson, Chairperson

Bo Luo

Fengjun Li

Date defended: December 11, 2023

The Thesis Committee for Thomas Atkins certifies
that this is the approved version of the following thesis:

Secure and Auditable Academic Collections Storage via Hyperledger Fabric-Based Smart Contracts

____
Drew Davidson, Chairperson

Date approved: ______________________________

Abstract

This paper introduces a novel approach to manage collections of artifacts through smart contract access control, rooted in on-chain role-based property-level access control. This smart contract facilitates the lifecycle of these artifacts including allowing for the creation, modification, removal, and historical auditing of the artifacts through both direct and suggested actions. This method introduces a collection object designed to store role privileges concerning state object properties. User roles are defined within an on-chain entity that maps users' signed identities to roles across different collections, enabling a single user to assume varying roles in distinct collections. Unlike existing key-level endorsement mechanisms, this approach offers finer-grained privileges by defining them on a per-property basis, not at the key level. The outcome is a more flexible and fine-grained access control system seamlessly integrated into the smart contract itself, empowering administrators to manage access with precision and adaptability across diverse organizational contexts. This has the added benefit of allowing for the auditing of not only the history of the artifacts, but also for the permissions granted to the users.

Acknowledgements

Acknowledgements go here.

Contents

1 Introduction **1**

1.1 Motivation 1

1.1.1 Summary of Resutls 1

1.2 Background 1

1.2.1 How Hyperledger Fabric Works 1

2 Chapter 2 Design and Implementation **2**

2.1 Smart Contract Object Model 2

2.1.1 Types of Items 2

2.1.2 short 2

3 Chapter 2 Title **4**

3.1 Example Section 4

3.1.1 Code Example 4

3.1.2 Subfigure Example. 5

4 Conclusions **6**

4.1 Summary of Results 6

4.1.1 Implications 6

4.2 Future Work 6

A Misc stuff **8**

B Misc Stuff 2 **9**

List of Figures

List of Tables

2.1 Functions In Generic Collection Smart Contract 3

Chapter 1

Introduction

1.1 Motivation

Blockchain are a form of distributed immutable databases

1.1.1 Summary of Resutls

1.2 Background

1.2.1 How Hyperledger Fabric Works

(prefix Kuzlu et al., 2019, postfix)

Chapter 2

Chapter 2 Design and Implementation

2.1 Smart Contract Object Model

2.1.1 Types of Items

The data model for the smart contract is designed around the key-value interface of the ledger and the fabric stubs ability to store the state of object under composite keys. The composite keys support querying the world state by levels of hierarchal keys that are appended onto each other. The definition of the objects are done though a protobuf definition file, where the objects are messages that have been annotated with the KeySchema option message.

The objects that are managed by the smart contract are grouped into two sets primary objects and sub-item types. The primary objects are the objects types that are directly managed by the access control and the sub-items are used to manage metadata and auxiliary state of the primary objects. In the proof of concept implementation the two sub-item that are defined are the Hidden Transaction list and the Suggested Updates.

All of primary objects have to have a collection Id property and then have the paths to the properties that make up the other attributes of the key listed in the key FieldMask property of the annotation.

2.1.2 Stages of Generic Object Contract

The smart contract is broken up into three stages. The first stage is responsible for defining functions that are exposed by the smart contract. This layer is responsible for initial validation and

Function	Transaction Type	Action
Get	Query	View
List	Query	View
ListByCollection	Query	View
ListByAttrs	Query	View
Create	Invoke	Create
Update	Invoke	Update
Delete	Invoke	Delete
GetHistory	Query	View History
GetHiddenTx	Query	View Hidden Txs
HideTx	Invoke	Hide Tx
UnHideTx	Invoke	UnHide Tx
GetSuggestion	Query	Suggest View
SuggestionListByCollection	Query	Suggest View
SuggestionByPartialKey	Query	Suggest View
SuggestionCreate	Invoke	Suggest Create
SuggestionDelete	Invoke	Suggest Delete
SuggestionApprove	Invoke	Suggest Create

Table 2.1: Functions In Generic Collection Smart Contract

unpacking the requests arguments provided to the invocation of the smart contract. It is also responsible for calling the second stage function and then packing the item into its return format. The second stage is a wrapper around the fabric shim interface to the world state that defines defines the operations that can happen on primary objects and their sub-object domains. This layer is responsible for building the operation data structure by setting the action field biased on the function and then extracting the collection id, the type of the object, and potentially the paths that the action takes place on from the object and arguments passed into the function. This operation is then handed of to the third and final stage where the action is authorized. If the operation is authorized the second layer then calls the required chaincode stub functions to interact with the ledger.

Chapter 3

Chapter 2 Title

Abstract

Chapter 2 Abstract: Code and figure example.

3.1 Example Section

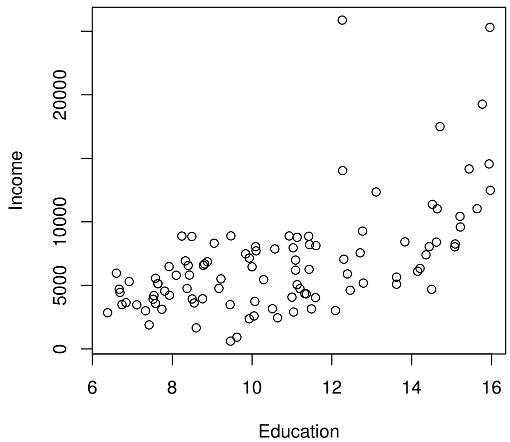
3.1.1 Code Example

Check on conventions on inputting code just to be safe. This is likely field dependent, so this is worth considering. This is the default style, which is aberrant to look at. In any case, this is how subimport works for nested files to organize your document with each chapter self contained in it's own folder.

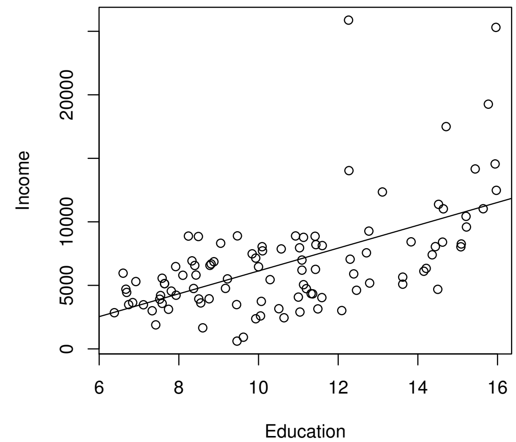
```
library(car)


pdf(file="car.inc.ed.pdf", height=5,width=5, onefile=F,
    paper="special")
plot(income~education, xlab="Education", ylab="Income",
    main="", data=Prestige)
dev.off()
```

3.1.2 Subfigure Example.

This is a true subfigure example.



(a) Caption A



(b) Caption B

Chapter 4

Conclusions

Abstract

Chapter 2 Abstract: Code and figure example.

4.1 Summary of Results

4.1.1 Implications

4.2 Future Work

References

Kuzlu, M., Pipattanasomporn, M., Gurses, L., & Rahman, S. (2019). Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability. In *2019 IEEE International Conference on Blockchain (Blockchain)* (pp. 536–540). Atlanta, GA, USA: IEEE.

Appendix A

Misc stuff

Appendix B

Misc Stuff 2