School of Computer Science and Software Engineering
Bachelor of Computer Science
(Digital System Security & Cybersecurity)

CSIT321 – Project
SIM Session 3-4, 2022
Group: FYP-22-S3-11

Project Requirement Specification

Project Members	UOW ID
Daryl Low Ze Lin	7349026
Goh En Wei Mervyn	7233292
Low Wei Chern	6656250
Terence Tay Jia Hao	6859136
Foo Min Zhan	7058810

Project ID: CSIT-22-S3-01

Project Title: Identifying Cryptographic Functions in Blockchain Systems

Supervisor: Sionggo Japit
Assessor: Tian Sion Hui

Document Control

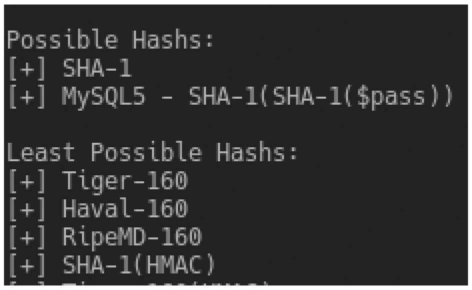
Name	Title/Role	Where (Location where document is stored)
Min Zhan	Developer	Team’s Google Drive
Wei Chern	Developer	Team’s Google Drive
Mervyn	Developer	Team’s Google Drive
Daryl	Developer	Team’s Google Drive
Terence	Developer	Team’s Google Drive

Record Revision

Revision Date	Description	Section Affected	Changes Made By	Version After Revision
3 Aug 2022	Start of documentation	All	Team	0.1

Project Objective

The overview of this project is to develop a tool that automatically detects cryptographic functions in Blockchain systems and compares them to known open-source crypto-libraries or other Blockchain systems.



```
Possible Hashs:
[+] SHA-1
[+] MySQL5 - SHA-1(SHA-1($pass))

Least Possible Hashs:
[+] Tiger-160
[+] Haval-160
[+] RipeMD-160
[+] SHA-1(HMAC)
```

1.1 A binary showing possible hash function used in a certain hash

Blockchains

Blockchains are **distributed ledgers** that store data as **Blocks** on its networks. The networks create a shared, **immutable ledger** that helps in the process of recording transactions. The computers on these networks are called the **Nodes** of the Blockchain.

The blocks are similar to databases but these data are digitally chained together, with a hash pointer pointing from one block to another, forming the chain.

Each block contains a hash (digital fingerprint or unique identifier), timestamped batches of recent valid transactions, and the hash of the previous block. The previous block hash links the blocks together and prevents any block from being altered or a block being inserted between two existing blocks. In this way, each subsequent block strengthens the verification of the previous block and hence the entire blockchain. This is the method that renders the blockchain tamper-evident, lending to the key attribute of immutability.

Blockchains can be public or private.
Public blockchains are open to anyone to interact and participate in. Public blockchains are usually developed in by an **Open-Source Community** and have broad decentralisation.
Private blockchains are closed networks for designated parties to participate in consensus verification and are partially decentralised.

Blockchains are valuable at increasing the level of trust among network participants because it provides cryptographic proof over a set of transactions.

Requirement Definition

User Stories

#1, As a User, I want to be able to analyse the hash functions from a block in a blockchain.

Name: Analyse Hash Functions as a User
Stakeholder and goals: User, Retrieve hash function used
Description: The user enters a block hash/.txt file and chooses to view the cryptographic hash functions from it
Actor: User
Trigger: User selects Analyse Hash Function from the CLI/GUI/API
Normal Flow: 1. User enters a block hash/.txt file onto the command line prompt 2. System will acknowledge the user input 3. System will identify the cryptographic functions on the block and display it to the user 4. User reads the information displayed, end.
Sub-flows: None
Alternative flows: 1. User enters a block hash/.txt file that is not identifiable 2. System will acknowledge the user input 3. System will cross reference the block with its stored information and identify the origin of its Blockchain 4. System will display the error message to the user

Requirement Definition

User Stories (Cont.)

#2, As a User, I want to be able to identify which Blockchain a block originates from.

Name: Identify Origin Blockchain as a User
Stakeholder and goals: User, Identify Origin of block presented
Description: The user enters a block hash/.txt file and chooses the option to have its Origin Blockchain identified
Actor: User
Trigger: User selects Identify Blockchain Function from the CLI/GUI/API
Normal Flow: 1. User enters a block hash/.txt file onto the program 2. System will acknowledge the user input 3. System will cross reference the block with its stored information and identify the origin of its Blockchain 4. System will display the information to the user
Sub-flows: None
Alternative flows: 1. User enters a block hash/.txt file that is not identifiable 2. System will acknowledge the user input 3. System will cross reference the block with its stored information and identify the origin of its Blockchain 4. System will display the error message to the user

Requirement Definition

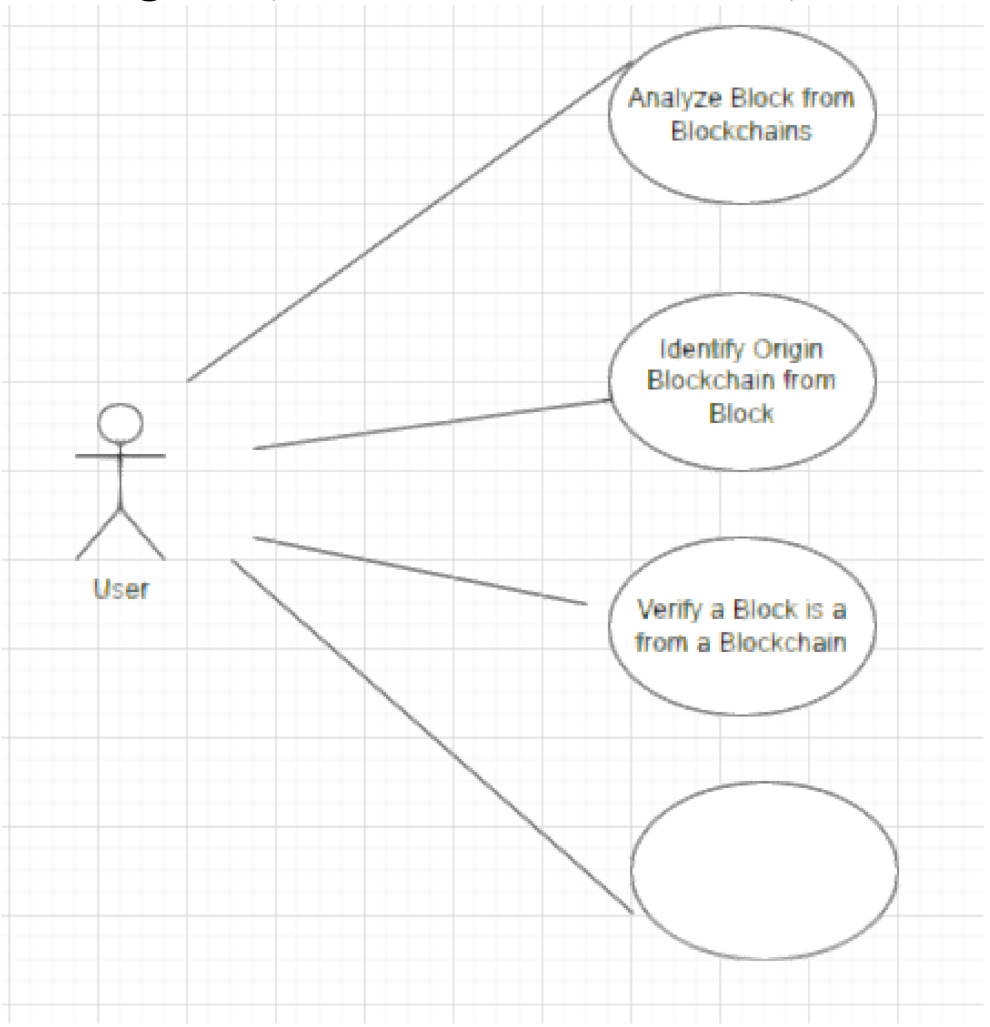
User Stories (Cont.)

#3, As a User, I want to be able to verify if a block is from a Blockchain

Name: Validity verification on a Block
Stakeholder and goals: User, Verification of block presented
Description: The user enters a block hash/.txt file and chooses the option to have its verified
Actor: User
Trigger: User selects Identify Blockchain Function from the CLI/GUI/API
Normal Flow: 1. User enters a block hash/.txt file onto the program 2. System will acknowledge the user input 3. System will check the hash of the block to verify it 4. System will display validity to the user
Sub-flows: None
Alternative flows: 1. User enters a block hash/.txt file onto the program 2. System will acknowledge the user input 3. System checks the hash and is unable to verify it 4. System will display that the block cannot be verified

Requirement Definition

Use Case Diagram (Placeholder also, for now)


Analyze Block from Blockchains
Identify Origin Blockchain from Block
Verify a Block is a from a Blockchain
User

1.2 User diagram of possible use cases in this project.

Requirement Definition

Functional Requirements

1) The system must be able to scrape blocks from a Blockchain

2) The system must read blocks from Blockchain. This process may require using Language Processing techniques in order to:
    i) Identify encryption used
    ii) Identify hash function used

3) The system must compare cryptographic function to known open-source cryptolibraries

4) The system must be able to store identifying information to be used to determine the hash function and present it to the user.

5) The system must be able to ensure Block Validity. Blocks have to be verified before they can be approved.

Non-functional Requirements

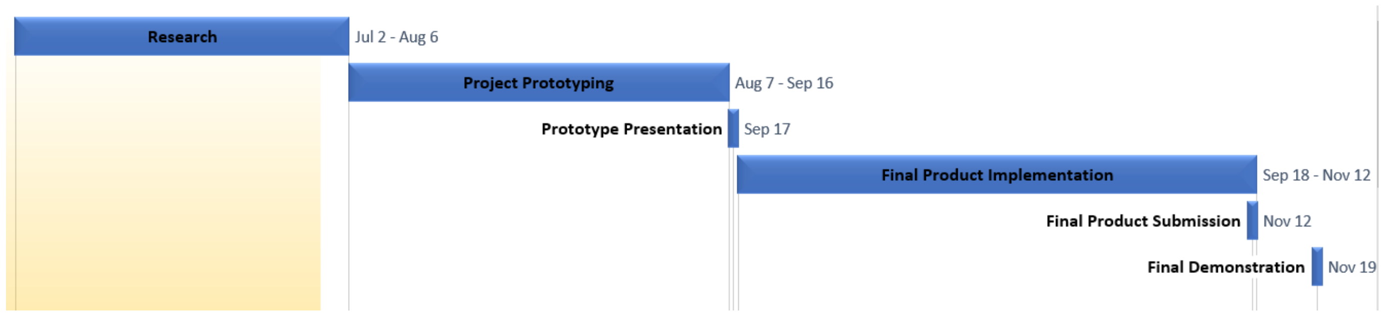
1) CLI/GUI/API to display the relevant information based on user’s actions.
2) Security of a single block in a Blockchain
3) Learnability (How easy for users to use)
4) Error Occurrences (How often will user cause the program to throw errors)
5) Integrating Database to store identifying information on Blockchains

Other Requirements

Cross Platform compatibility

- With the program being developed in the Python language, this will allow Windows, MacOS and Linux users to be able to run the program.

Timeline


Research
Jul 2 - Aug 6
Project Prototyping
Aug 7 - Sep 16
Prototype Presentation
Sep 17
Final Product Implementation
Sep 18 - Nov 12
Final Product Submission
Nov 12
Final Demonstration
Nov 19

Project Milestones

Date	**Deliverables**
6th August 2022	Project’s Requirement Specification document
17th September 2022	Prototype Demonstration
17th September 2022	Progress Report submission
12th November 2022	Final product & documentation submission
12th November 2022	Reflective Diary submission
19th November 2022	Final Presentation

Appendix

Language processing:

The ability of a program that breaks down human language for machines to understand how human language as it is spoken and written.

Tokenization - Breaking down of text into single clauses.
Part-of-speech-tagging - Process of correctly marking words as nouns, verbs, adjectives, adverbs, pronouns

Stemming and Lemmatization - Reducing words to their root forms

Stop word removal - Filter common words that add no meaning or provide no information

Open-Source Development

This refers to software with their **source codes available to the public** which allows for communities to help and participate in continuous development.
The end product of these software might not be free but the source codes are still made available freely for communities to develop their own versions of the program.

Open-Source developments are published under **Open-Source Licences** for all to have read and write access and distribute the source code. This results in a form of **Decentralised Collaboration.**

Command Line Interface

A text-based user interface which users use it to interact with a machine. This provides a way for users to update setting parameters for environment, invoking executables or issuing commands, providing information to users as to what actions they perform

Graphical User Interface

Another form of user interface which allow users to interact with executables, programs through graphical icons and/or audio indicators such as primary notation, instead of text-based interface as mentioned above.

Application Programming Interface

A software intermediary that allows applications/products to communicate with each other without the need to know how they're implemented. API gives developers flexibility; simplifying design, administration and usage, hence allowing developers to develop applications faster and simpler, saving time and cost of developing applications.