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

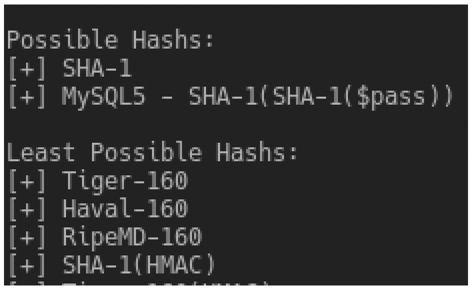
Name	Title/Role	Where (Location where document is stored)
Min Zhan	Developer	Team’s Google Drive
Wei Chern	Developer	Team’s Google Drive
Mervyn	Developer	Team’s Google Drive
Daryl	Developer	Team’s Google Drive
Terence	Developer	Team’s Google Drive

Record Revision

Revision Date	Description	Section Affected	Changes Made By	Version After Revision
3 Aug 2022	Start of documentation	All	Team	0.1
19 Sep 2022	**Revisions after Mid-Point Presentation**	**All**	**Team**	**0.2**

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.

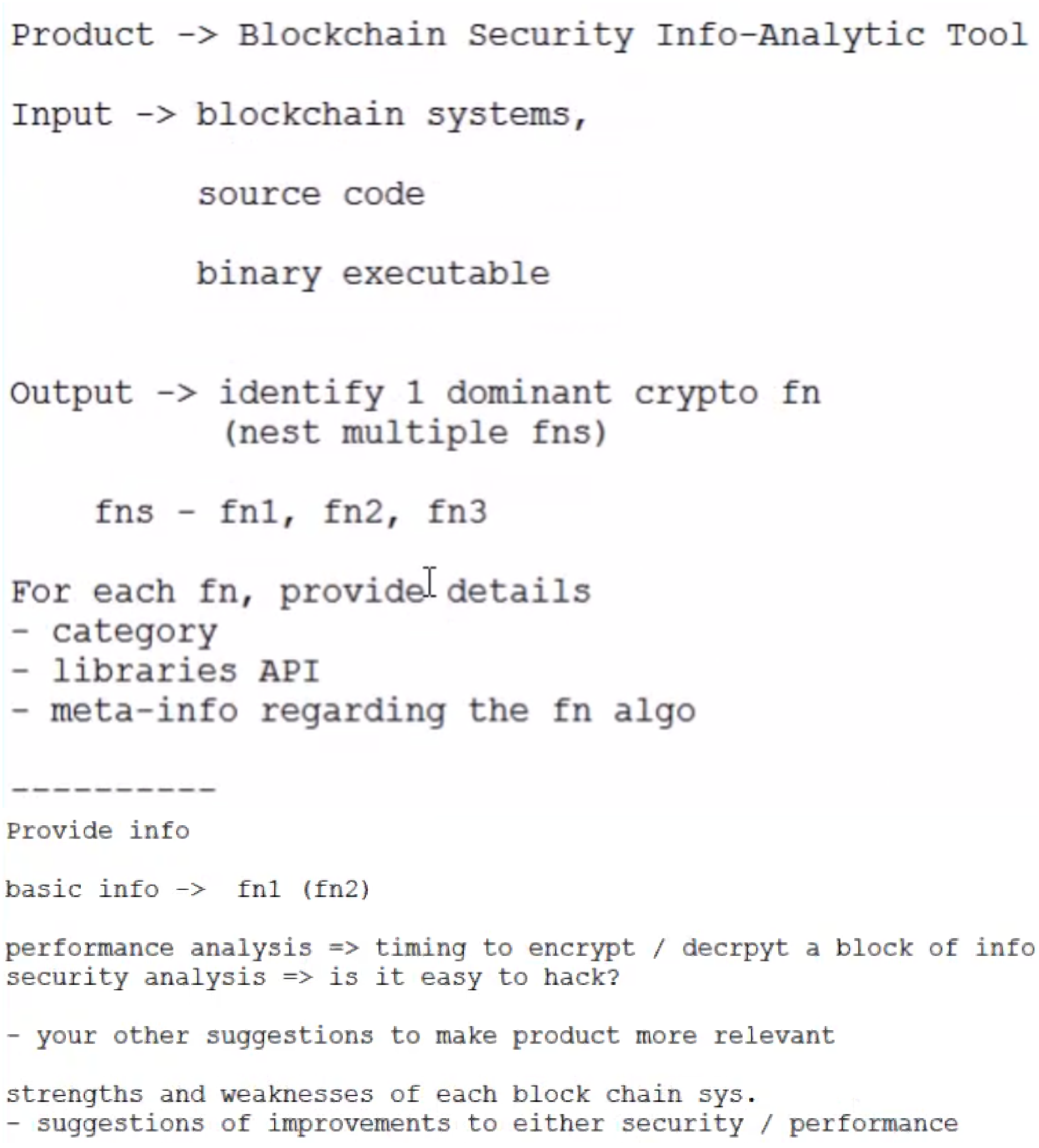
The blocks are similar to databases but these data are digitally chained together, with a hash pointer pointing from one block to another, forming the chain.

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 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.

Mid-Point Presentation



```
Product -> Blockchain Security Info-Analytic Tool

Input -> blockchain systems,

         source code

         binary executable


Output -> identify 1 dominant crypto fn
          (nest multiple fns)

    fns - fn1, fn2, fn3

For each fn, provide details
- category
- libraries API
- meta-info regarding the fn algo

----------
Provide info

basic info ->  fn1 (fn2)

performance analysis => timing to encrypt / decrpyt a block of info
security analysis => is it easy to hack?

- your other suggestions to make product more relevant

strengths and weaknesses of each block chain sys.
- suggestions of improvements to either security / performance
```

1.2 Project Assessor Feedback.

Revision

After the team has been through the Mid-Point presentation, feedback has been received and much needed clarification was given to us from the Project Assessor, to guide us towards achieving the expected project benefits.

However, this has also exposed the inadequacy and failures of the group's initial Requirement Specification. The initial proposed Requirement Specification has both **over promised** and **under promised** with our proposed User Stories and Requirement Specification.

The initial proposed Requirement Specification has **too many wrong** Glossary and Terms regarding Blocks and Blockchains written on it. **Blatant misuse** of these terms have **severely distorted** how the team wished to convey our initial goals and promises.

Our initial prototype idea also shifted fundamentally from a simple CLI program to a web application and has received a warm reception from the Project Assessor.

Our foray into cryptocurrency blockchains was not heavily penalised by the Project Assessor, this also clarified one of our major concerns of being too involved with Cryptocurrency Blockchains.

Revisions were made to our Requirement Specification to be **more aligned** with **achieving the expected project goals**, **correct the misuse** of the glossary and terms of Blockchains and to further **guide the project to achieving its goals**.

Requirement Definition

User Stories

#1, As a User, I want to view the Hash function used on a Cryptocurrency Blockchain.

Name: View Hash Function used on a Cryptocurrency Blockchain
Stakeholder and goals: User, obtain information on the Hash Function used on a Blockchain
Description: The user enters the name of the Cryptocurrency Blockchain and the Product displays the main Hash Function used on it to the User
Actor: User
Trigger: User enters Cryptocurrency Blockchain into our Product CLI/GUI/API/WebApp
Normal Flow: 1. User enters the Cryptocurrency Blockchain into the Product 2. Product retrieves the main hash function used 3. Product displays the main hash function used to the user 4. User reads the information displayed to them
Sub-flows: None
Alternative flows: 1. User enters a wrong Cryptocurrency Blockchain 2. Product informs User of a wrong input 3. Product shows that no information was found

Requirement Definition

User Stories (Cont.)

#2, As a User, I want to view the Hash Function used from a given Block Hash

Name: Identify Hash Function on a Block Hash as a User
Stakeholder and goals: User, Identify Possible Hash Functioned used
Description: The user enters a block hash into the product. Product analyses it and displays the possible Hash Function used
Actor: User
Trigger: User selects view the Hash Function from the CLI/GUI/API/WebApp
Normal Flow: 1. User enters a block hash into the product 2. Product performs analysis 3. Product displays possible hash functions used based on input 4. User reads the information displayed by the product
Sub-flows: None
Alternative flows: 1. User enters a block hash file that is not identifiable 2. System fails to perform analysis 3. System shows that no Hash Functions can be identified

Requirement Definition

User Stories (Cont.)

#3, As a User, I want to be able to perform analysis on a Cryptocurrency Blockchain source file for dominant cryptographic functions.

Name: Conduct Analysis on Cryptocurrency Blockchain source files
Stakeholder and goals: User, Analysis of Cryptocurrency Blockchain source file
Description: The user uploads a source file onto the program, program performs analysis on this source file
Actor: User
Trigger: User selects Source File analysis from the Product, User uploads a file onto the product
Normal Flow: 1. User enters source file onto the Product 2. Product accepts the file 3. Product performs analysis on the file given 4. Product outputs the analysis to be displayed to the user 5. User views the analysis done by the program and view the dominant cryptographic function
Sub-flows: None
Alternative flows: 1. User uploads an invalid file (not .zip file) 2. Product accepts the file 3. Product detects that an invalid file type was uploaded 4. Product displays error message to the user

User Stories (Cont.)

#4, As a User, I want to be able to compare the performance between different Cryptographic Functions on different Cryptocurrency Blockchains.

Name: Performance Comparison between different Cryptographic Functions on different Cryptocurrency Blockchains
Stakeholder and goals: User, Performance Analysis comparison between different cryptographic functions
Description: The user selects the Compare Crypto feature on our Product
Actor: User
Trigger: User selects the Compare Crypto feature on our Product .
Normal Flow: 1. User selects the Compare Crypto Feature 2. User selects 2 different Cryptocurrency Blockchains to compare 3. Product perform analysis 4. Product displays comparison to the user
Sub-flows: None
Alternative flows: 1. User selects the Compare Crypto feature 2. User makes an invalid selection 3. Product registers the invalid selection 4. Product informs user of the invalid selection

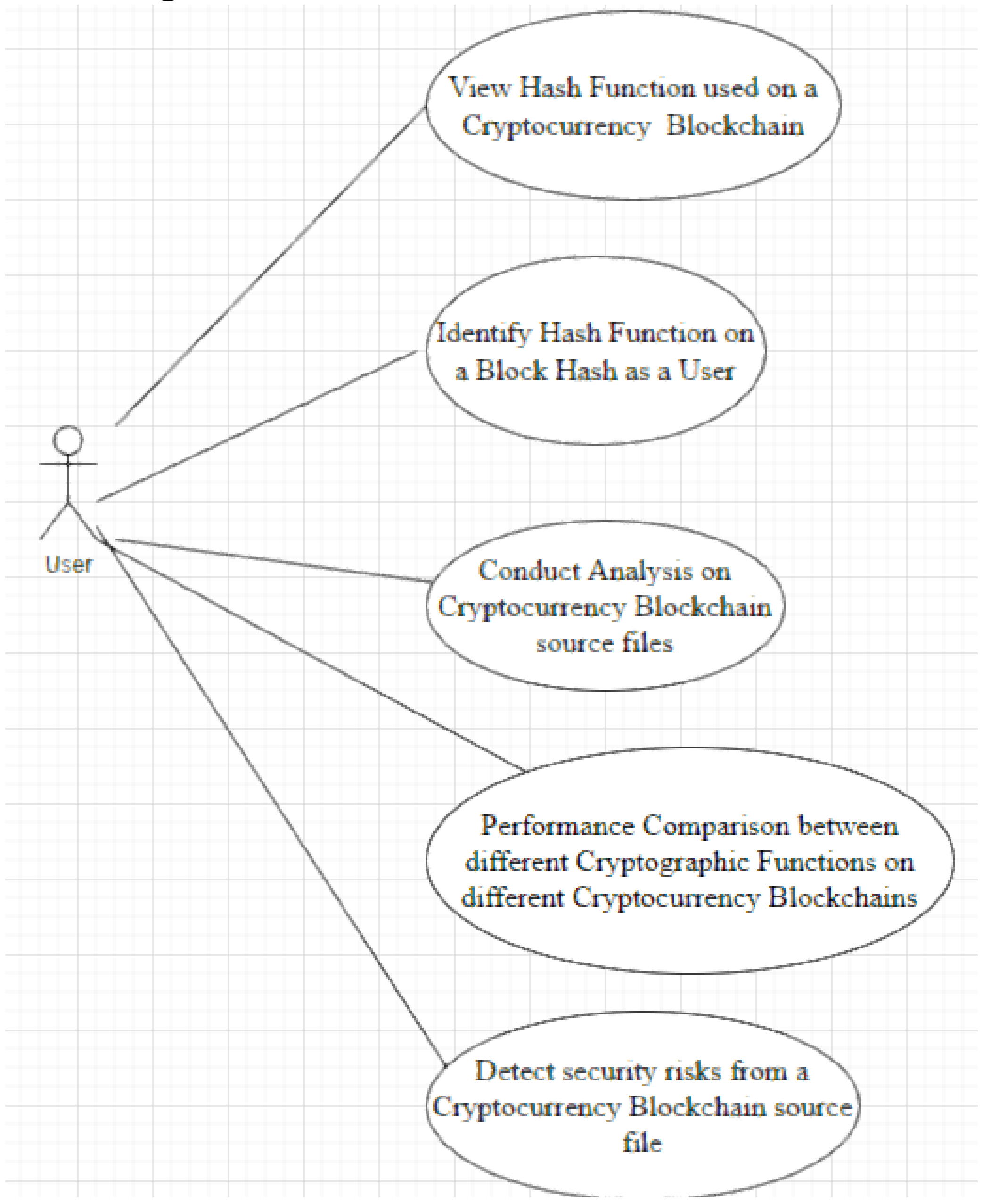
User Stories (Cont.)

#5, As a User, I want to be able to see if there are any security risks from a Cryptocurrency Blockchain source file

Name: Detect security risks from a Cryptocurrency Blockchain source file
Stakeholder and goals: User, Detect any security risks or weakness
Description: The user uploads a source file onto the program, program performs analysis on this source file
Actor: User
Trigger: User selects Source File analysis from the Product, User uploads a file onto the product
Normal Flow: 1. User enters source file onto the Product 2. Product accepts the file 3. Product performs analysis on the file given 4. Product outputs the analysis to be displayed to the user 5. User views the analysis done by the program and view the dominant cryptographic function
Sub-flows: None
Alternative flows: 1. User uploads an invalid file (not .zip file) 2. Product accepts the file 3. Product detects that an invalid file type was uploaded 4. Product displays error message to the user

Requirement Definition

Use Case Diagram


View Hash Function used on a Cryptocurrency Blockchain
Identify Hash Function on a Block Hash as a User
User
Conduct Analysis on Cryptocurrency Blockchain source files
Performance Comparison between different Cryptographic Functions on different Cryptocurrency Blockchains
Detect security risks from a Cryptocurrency Blockchain source file

1.3 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

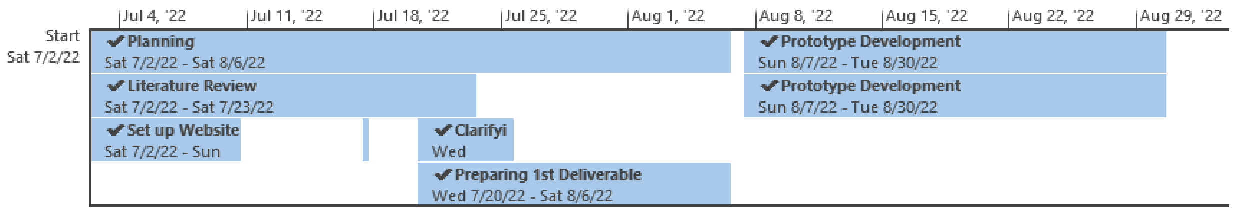
1) CLI/GUI/API to display the relevant information based on user's actions.
2) Security of a single block in a Blockchain
3) Learnability (How easy for users to use)
4) Error Occurrences (How often will user cause the program to throw errors)
5) Integrating Database to store identifying information on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


Jul 4, '22
Jul 11, '22
Jul 18, '22
Jul 25, '22
Aug 1, '22
Aug 8, '22
Aug 15, '22
Aug 22, '22
Aug 29, '22
Start
Sat 7/2/22
✓ Planning
Sat 7/2/22 - Sat 8/6/22
✓ Literature Review
Sat 7/2/22 - Sat 7/23/22
✓ Set up Website
Sat 7/2/22 - Sun
✓ Clarifyi
Wed
✓ Preparing 1st Deliverable
Wed 7/20/22 - Sat 8/6/22
✓ Prototype Development
Sun 8/7/22 - Tue 8/30/22
✓ Prototype Development
Sun 8/7/22 - Tue 8/30/22

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.