BIRMINGHAM CITY University

CMP6200/DIG6200

INDIVIDUAL UNDERGRADUATE PROJECT

2023–2024

A1: Proposal

A PLUGGABLE, CRYPTOGRAPHIC RECEIPT GENERATOR FOR PERSONAL AND ENTERPRISE USE (NAME STILL IN DEVELOPMENT)

Course: MSci Computer Science
Student Name: Matthew Potts
Student Number: 21124021
Supervisor Name: Shadi Basurra

Contents

1 Introduction

1.1 Background and Rationale

My project, in a simplified manner, will consist of developing an open-source, cryptographically verifiable log of transactions performed on different files in a personal or enterprise space. This is to ensure, especially in an enterprise environment, that different staff can be held accountable for any action performed on a file – to ensure that any malicious foul play, or errors (to name two examples), can be easily traversed to its source. This consistently maintains policy regulation throughout entire business operations, without requiring timely independent investigations when issues arise.

Modern-day enterprise requires data to be intricate, and correct – malicious behaviour within, and outside, a firm can occur, and files can be tampered with. These files may be distributed to customers, used within business operations, or be constantly modified further as part of an ongoing project – as such these files must be logged for changes in order to maintain consistency, and ensure policy is implemented. Personal users may also choose to use this for independent software they're distributing, and can be useful in open-source software – to check for any changes made by a community, and ensure changes can be made answerable directly to a user.

Blockchain technologies exist already, predominantly for enterprise, on cloud architecture – including the likes of AWS QLDB (Amazon Web Services, .n.d.), and Trillian (transparency.dev, .n.d.), both use API's that present customers an 'immutable' log of 'cryptographically verifiable' changes. However, they're limited by quotas and a few loopholes I aim to amend – for instance, QLDB is limited by quotas – a maximum of 5 ledgers in one 'Region' (enterprise), 5 Kinesis Streams, and only 20 tables (docs.aws.amazon.com, n.d.). Trillian does also note a few limitations, one being that malicious code could still be injected, it doesn't know what is malicious and what isn't. Secondly, a hacker could tamper with the verifiable log, albeit this can be undetected if the log and verifiers are simultaneously modified (transparency.dev, .n.d.).

I'm aware these software's are far more advanced than the tool I'm planning to create, and my interest in Cyber Security leads me to believe I can create a similar software, without any numerical quotas – a more lenient framework that can be expanded to a user's or enterprise's will. Similarly, I plan on developing basic anti-malicious frameworks, by using 'keywords' as definitions – this can be expanded in future updates.

1.2 Key Themes/Topics

An enormous aspect of this project could be considered a blockchain, acting in very similar methods – and likewise, out of curiosity from QLDB, I'm looking into researching how databases can be used to store cryptographically verifiable data, the algorithms that can be used to encrypt and decrypt data, and the methods that can be used to constantly stream data into a software. I'd also like to research case studies as to where similar software's have operated successfully, and which didn't – taking key inspiration from both sides of the spectrum in determining what features should be included or omitted in my final release. Key components as to what details are taken during a logging process, their effectiveness, and

whether other details could be taken for further security purposes – could be another topic I choose to look into during this project.

Although databases could play an intricate part in this software, I'd like to research what other forms of storage could be used – and their advantages and disadvantages in comparison to a database.

2 Aim and Objectives

2.1 Project Aim

I aim to craft an open-source framework, excluding numerical quotas, ledging software that can be tailored by personal or enterprise customers – that maintains a cryptographically verifiable receipt log of data being modified or transferred. This aims to maintain policy, data repudiation, and aid disciplinary investigations.

2.2 Project Objectives

- Create cryptographically-verifiable receipts for every data transaction on a system. A mixture of SHA hashing and AES encryption algorithms can be utilised to create these.
- Ensure every data transaction can be made directly answerable to a user.
- Ensure data policy is regulated across an entire system.
- Ensure every receipt can be utilised to maximum effect in a disciplinary investigation.
- Create a framework that can be expanded in future updates, this includes anti-malicious definitions.
- Ensure new data is constantly streamed into the software, can be operated on in the background.
- Flag new developments when they occur, and whether they hold any malicious intent.
- Develop in a manner so that software can be used by other users, for instance a computer can act as a server, for others to connect to.
- Create a GUI that can be used to display changes to files, the answerable users, and a log of any changes they made, this will only be created once the main framework has been entirely developed.
- Ensure receipts contain accurate details of file(s) operated on, changes made, and the user behind them.
- If such details can't be found, or inconsistences are spotted (i.e. user can't be detected), raise for further investigation.
- Store logs in an organised structure, such as .csv, or a database. Ensure they're stored in a manner to which they can be quickly searched, retrieved, and visualised for use in investigations.
- Frequently overview logs for any malicious keywords, or peculiarities, which can then be flagged.

3 Project Planning

3.1 Initial Project Plan

- **Create a cryptographically verifiable log of data transaction receipts.**
  The log becomes immutable once file editing has finalised, it should be encrypted so it can be only displayed to a figure of authority, and each receipt should contain intricate details of changes made, the user behind them, and other metadata such as time and place.
    - Each receipt will undergo modernised AES encryption/decryption, and be stored collectively into a blockchain-like framework utilising SHA hashing.
    - These receipts can only be viewed by specific users, receipts could also be further password locked, settings can also be modified to ensure receipts take more parameters when being stored.
    - Receipts can also be backdated, in the sense that older changes can also be displayed. In comparison with newer changes, this can further aid in discrepancy issues.
    - Different policies cam also be implemented into this software, providing parameters that can be used to examine receipts for any policy discrepancies.
- **Create a data recorder that constantly fetches data into software**
  To save manual input, and implement background operation – a directory-wide, or system-wide constant data reader should be implemented to fetch data constantly, when files are opened and modified – they should be passed through the software as soon as possible.
    - Users can customise how this operates as this does have a potential to use a significantly high amount of computer resources, full scans can be made to start, and instead of watching all programs – specific programs that require receipts can be chosen.
- **Create a utility that can port this tool on a server-side machine, or on a user-side machine**
  Depending on the environment this tool is used in, a customer might prefer to install this framework on a centralised server, so it can be accessed by everyone. In a smaller environment, this might not be necessary.
    - I might divide this software into two. During the installation, the user can choose one of two program variants; a predominantly server-built one (e.g. with added network settings, and more intense operations).
- **Integrate maintenance into framework**
  To update anti-virus definitions, or install any new updates to this project (being open-source), a utility to update this software with new features can also be integrated, saving a user having to download files themselves from a repositor, and eliminating the chance of a corrupted installation.
    - Users can also customise as to whether they want to receive updates automatically, or install them themselves off the host repository. Anti-virus definitions can also be listed to show what they can protect against. Logs of future updates in the making can also be shown. This does depend on the user having an internet connection.
- **Create a GUI for this tool**
  Once the entire framework has been completely implemented, a GUI can be constructed on top of this, displaying clearly to a user new events in the log, flagged events, and easily accessible receipts of changes made to files, or to a

specified given file. Preferences can also be adjusted for a number of different factors, such as text side, and log storage.
    - Other preferences could include changing visual colours (e.g. contrast), how images are displayed, boot on start-up, and easily-clickable options.

3.2 Resources

Due to the nature of a software like this, I don't believe it is entirely paramount that stronger hardware, for testing purposes, is completely required for a tool like this. Although this software will operate differently on different hardware, preferences can be configured to change operation, and resource intensity. There are numerous tools online that can be used to emulate a server-side environment, and I can utilise other computers I own to test different capabilities on the server-side variant of this software.

Depending on how I choose to organise storage, .csv files can be easily modified with a tool such as Excel, Microsoft also offers DBMS programs such as Access, albeit I would like a majority of my storage management to be included within this software, for user-ease. In regards to research materials, I'd like to examine past case studies of similar software on Google Scholar and IEEE Xplore. To research further about the anatomies of blockchain-like software, cryptography, and server-side environments to name a few examples, I intend on using textbook repositories such as BCU Library Search. Transparency.dev does also contain an entirely open-source Github repository – that can serve as inspiration for future development (GitHub, n.d.).

3.3 Risk Assessments

A few risks, to name a few examples, right from the start come to mind:

- **Unsecure cryptographic log**; regardless of ether is stored as a .csv file or a database, unless this file is not securely encrypted to AES standard, it can be modified to disguise malicious changes to a system, and remove receipts entirely (given the receipt is correctly modified). Relative to how significant this is to the entire functionality of this software, it's of paramount importance that this log is fully encrypted.

- **Incomplete blockchain**; if a discrepancy is identified in the hash of a blockchain, there is a potential for it to completely disrupt the receipt process, and render the update history of a file obsolete. This can cause major harm to internal disciplinary investigations, as a major source of credible evidence is lost.

- **Corrupt cryptographic receipts**; in the event a receipt is produced, and not all parameters can be retained for example – this can render a receipt obsolete, and disrupt a blockchain further. This is not as high-risk as others though, as a receipt will attempt to be created again, and if this fails – higher authority will overlook this with urgency. However, there is also a risk that individual receipts can be loaded with malicious data, or intentionally spoofed to cause harm – this is where anti-malicious definitions, as mentioned earlier, come into play. If any of this is detected, the receipt will not be appended to the log, and will be held in a standby state for further verification. The file(s) can't be modified until further verification is complete.

- **Corrupt installation**:
  The update facility in this software does come with a risk of corrupting installations on either a server or user-side configuration. The facility will only receive files from an official repository, and will only replace required files. This isn't as high a risk as others, due to the fact a customer could easily reinstall the entire software if any errors are found like this.

- **Incorrect anti-malicious definitions**:
  Similarly, these definitions are also packaged within updates. These are only ever tested beforehand until they're completely compatible with both the software, and the receipt structures. This could be classed as a somewhat high risk in comparison to others, but due to the rigorous nature of testing these definitions before they're dispatched, it's highly unlikely that any errors will transpire.

4 Project Review and Methodology

4.1 Critique of Past Final Year Projects

A past project here that intrigued me is one published by Amritpal Singh Sandu, titled "Using Blockchain as Secure and Immutable Storage" (Sandhu, A. "Using Blockchain as Secure and Immutable Storage", 2018). A key weakness that struck my eye right from the start was a lack of extensive Research Methodology, in Section 3.1, on Page 18, he goes on to solely discuss the Waterfall Model. Although this is done in clear, informative detail – it stuck me as odd that only one of these methodologies were examined, a system may have performed better if other models, such as the Agile Model, were also considered. A common theme in a majority of these pages is that a few explanations do lack in how clear they are. It's not as easy looking from a third party perspective how exactly certain technologies affect certain processes. For instance, on Page 21 – Sandhu describes "this analysis is done by applying the Blockchain technology to any digital transaction that is exchanged online". This statement is quite vague, what about the Blockchain technology does this?

Aside from that, I couldn't resource any more projects with an explicit link to either blockchains, or a ledging software of some sort. Therefore, another project I decided to examine was one titled "Speechless, a Virtual Personal Assistant" by Villiam Langsch (Langsch, V. "Speechless a Virtual Personal Assistant.", 2018). Right from the off, I can already see how the implementation aspect of this project has been described in far greater detail than Sandhu's project. It's clearly defined into segments, includes figures, and explains in extensive detail how different areas of code operate, in a manner that is easily comprehendible to a third-party audience. This manner of writing is similarly applied into other segments, like "Testing" for instance. Alternatively, an aspect that disappointed me slightly as the "Objectives" segment, on Page 13. Objectives like "Producing a report" are not objectives, I wanted to see what Langsch wanted to achieve with this project, and a clear set of objectives that would define the success of his project. A report is already a guaranteed segment of this project that does not determine how successful it is.

I take a lot of inspiration from these, particularly Langsch's report – it's very extensive, and it provides a clear skeleton structure as to how my final report should be structured.

4.2 Literature Search Methodology

As mentioned earlier, I intend on using a mixture of both Google Scholar, and other academic sources such as IEEE Explore – mainly to conduct research on how blockchains operate, how they intertwine with other forms of cryptography; similar case studies on similar software, and the advantages, disadvantages, and risks they produced. A few examples of search terms I've used on these platforms include "blockchain", "cryptographic log" and "cryptographic ledger", mainly because due to their straightforward nature – it provides me a basic platform to refine searches further, based on information found within these searched documents and the progress of the software itself.

However, quality control must also be assured to ensure that my content is relevant and from verifiable sources. For example, I'd stray away from content on newsletter articles – as intricate details may be overlooked for reader-ease, and sources might not be entirely accurate. Different books will include some form of bibliography that I can use to further verify data used.

4.3 Initial Literature Search Results

A few resources I have gathered as part of a brief literature search I have undertaken include the likes of "Blockchain" (Quiniou Matthieu, Blockchain), " and "Blockchain in Payment Card Systems" (Godfrey-Welch, Darlene; Lagrois, Remy; Law, Jared; Anderwald, Russell Scott; and Engels, Daniel W, 2018).

The former describes the general architecture of a blockchain, how it can be used, its advantages, disadvantages, and a number of case studies proving how blockchain theory can be applied into real environments. The predominant aim of a blockchain, and more so how a blockchain can be applied into a software like this, can be easily summarised into a single sentence, found on Page 1, underneath the "Non-Centralised Architecture" heading; "The blockchain and the distributed registry technology allow us to solve the problem of certification of the transaction chain, without using a centralised system, to a trusted third party".

Similarly, the latter describes, more specifically, how a blockchain can be applied to a real-world computational system, in this case – card payment networks, such as Visa and Mastercard. Banks often use ledgers as a form of transaction history, and they can be used to regulate their accounting, and ensure all financial transactions are correctly recorded. Godfrey-Welch, and others, state "the transaction ledger contains sensitive information that can be traced directly back to an individual or entity", on Page 2. They continue to state that "this traceability is multi-directional". This is so, in practice, suspicious transactions can be reported, and examined for any malicious intent; thus a similar concept can be applied into our software.

Both these sources are relevant in their own regard, Quintou's examination of the anatomy of a blockchain provides vital information into how they operate, how suitable they are in given environments, their advantages, disadvantages, and how they compare to other forms of cryptographic ledging technologies. Godfrey-Welch's work takes this theory, and showcases how it can be applied into real environments, providing vast inspiration as to how these blockchains can be constructed, and proving just how effective they can be on a wide range of scales.

5 Bibliography

Amazon Web Services, Inc. (n.d.). Amazon QLDB. [online] Available at: https://aws.amazon.com/qldb/?c=bl&sec=srv.

transparency.dev. (n.d.). An open-source append only ledger | Trillian. [online] Available at: https://transparency.dev [Accessed 4 Oct. 2023].

docs.aws.amazon.com. (n.d.). Quotas and limits in Amazon QLDB - Amazon Quantum Ledger Database (Amazon QLDB). [online] Available at: https://docs.aws.amazon.com/qldb/latest/developerguide/limits.html [Accessed 8 Oct. 2023].

transparency.dev. (n.d.). Trillan helps you reliably log all actions performed on your servers | Trillian. [online] Available at: https://transparency.dev/application/reliably-log-all-actions-performed-on-your-servers/#limitations [Accessed 8 Oct. 2023].

GitHub. (n.d.). transparency-dev. [online] Available at: https://github.com/transparency-dev [Accessed 10 Oct. 2023].

Quiniou, Matthieu. Blockchain : The Advent of Disintermediation, John Wiley & Sons, Incorporated, 2019. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/bcu/detail.action?docID=5781105.

Godfrey-Welch, Darlene; Lagrois, Remy; Law, Jared; Anderwald, Russell Scott; and Engels, Daniel W. (2018) "Blockchain in Payment Card Systems," SMU Data Science Review: Vol. 1: No. 1, Article 3.

Available at: https://scholar.smu.edu/datasciencereview/vol1/iss1/3

Sandhu, A. “Using Blockchain as Secure and Immutable Storage.” Faculty of Computing, Engineering and the Built Environment, 2018. Print.

Langsch, V. “Speechless a Virtual Personal Assistant.” Faculty of Computing, Engineering and the Built Environment, 2018. Print.