UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Physical Sciences

by Rishabh Arora, Finley Atherton, Conor Fitch, Sam Gardner, Alex Pakeman, Alistair Sirman

September 21, 2025

Secure Data Management of a CCTV System

Supervisor: Son Hoang
Examiner: Andrea Lecchini-Visintini

A group design project report submitted for the award of
Master of Engineering

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

A group design project report submitted for the award of Master of Engineering

by Rishabh Arora, Finley Atherton, Conor Fitch, Sam Gardner, Alex Pakeman, Alistair Sirman

This report tackles the interesting problem of how to create, manage and operate a proof-of-concept tamper-resistant CCTV system in a hostile environment, that can only be managed externally and must promote trust between the observers and the observed. It looks at the implementation of one such system and describes its advantages and pitfalls. Its aim is to help political bodies verify treaties between one another, by balancing security and trust, to produce a robust product that benefits all parties involved. The system has been designed from the ground up around these pillars, designing both the hardware and software to mitigate and report attempts to tamper, to give control to treaty partners, and to emphasise the treaty agreements these exist for. This report lays out the design choices made and the arguments for them, along with how these choices were made, and the result of said choices. The resulting system met goals, exceeded expectations, and fulfilled the root requirements that led to its development.

Contents

Acknowledgements

We would like to thank our supervisor, Son Hoang, for his support throughout the project. We would also like to thank Neil Evans and Neil Grant from AWE for their guidance as well as their selection of a project that was interesting to be a part of.

Chapter 1

Introduction

The brief for this project was to design and build a secure CCTV system capable of recording video footage and transmitting it to the owner of the system whilst verifying its integrity. The applications of this are focused on treaty verification, and the example given when discussing the project with the external partner (AWE) was a possibly hostile compound with our cameras installed at the entrances to verify a treaty of some kind. It is easy to see that a camera in an untrustworthy compound sending data through an untrustworthy network has many attack vectors to exploit. For this reason, the owner of the system needs to be able to verify any data it receives to check for tampering. The system must also be transparent so that the other side of the treaty will have no reason to object to the system installation.

This product is necessary to help with research into treaty verification between parties that have mutual distrust. There are many applications for this technology, including treaties between nation-states, corporate companies, or larger intergovernmental organizations like NATO. Beyond treaty enforcement, broader applications of this technology could include ensuring compliance in corporate partnerships, environmental monitoring agreements, and international inspections where mutual accountability is critical.

Currently, competing technologies include traditional CCTV systems, which lack built-in mechanisms for ensuring data integrity, and blockchain-based or cryptographic solutions that provide transparency but may not integrate seamlessly with video systems. The prototype developed here combines both secure data transmission and system transparency, bridging the gap between conventional CCTV systems and emerging verification technologies.

The external partner, AWE, is a large company that conducts work in several fields. The company's focuses that relate to this project are in nuclear disarmament and treaty verification. This project will likely be used by researchers to determine the viability of CCTV systems in treaty verification.

The final product we created is a prototype for a system that can be deployed to any site, compound, or location to enforce a treaty between two parties that do not trust each other. The reasons that the system is a prototype and not a fully functional, ready-to-be-used system are mainly budgetary. The budget for this project was £400, and for reasons that we will discuss in the report, we needed to produce two cameras for the system. Although the system works as intended, we recommend that hardware upgrades are investigated before the system is fully deployed.

1.1 Background

CCTV is not a new invention, and therefore, this project is superseded by many existing solutions. For example, you can buy a wireless CCTV system at most home appliance stores. These cameras work well, provide good quality video footage and when working on a network you own, are likely very secure. Despite this, they don't tackle the mutual distrust provided by treaty verification, making these existing systems unsuitable for the context we have here.

In addition to the many CCTV systems out there, there are also lots of technologies that can be used to tackle a mutual distrust between two parties. One example of this would be blockchain which uses transparency and decentralisation through an immutable ledger to create trust between two parties. This solution provides trust but does not seamlessly integrate with camera and footage capture technology so it is also unsuitable for this task.

Our solution to the problem is to build a camera that will connect to an untrustworthy network and send data through said network that has already been signed on the camera. The home network will then verify the data once it is received using a system of master and derived secrets. With the camera performing the signing and encryption and the home server decrypting and verifying the data, all the untrusted server needs to do is act as a proxy node in the system and pass the data along. This solution requires the cameras to house slightly more than you would typically need but it means that the untrusted party can remain untrusted and the data still be verifiable.

Alongside the obvious security problems with this project, there is a social element to consider, due to the treaty agreement that the camera system will enforce. For the untrusted party to allow these cameras to be set up at their compound/location, the camera system will have to be transparent. To achieve this all of the code for the project should be open source so the untrusted party can view the code and make sure it only works within the scope of the treaty. This is especially important because the cameras will be wirelessly connected to the host's network, which opens up a possible attack vector against the host.

Chapter 2

Project Management

2.1 Planning and Responsibilities

To begin the project we met with our supervisor and external partner (AWE). From this meeting, we recorded the first draft of the specification for the project based on the requirements we discussed with them. The first week of the project was primarily focused on refining the specification and project requirements, and organising how the project would run through to its completion. Once we had completed the first draft of our project brief, we sent it to AWE and set up another meeting to make any alterations based on their feedback. We also organised a weekly meeting to keep them up-to-date with our progress and make sure the requirements of the project hadn't changed. There were several times during these meetings that requirements had shifted slightly or new ones had emerged, these could then be addressed over the course of the next week.

The next thing we did to manage the project was select roles for each group member, this allowed us to all work on different parts of the project but also have a clear lead in each area to direct questions or ideas towards. Alistair Sirman was the designated project manager, his role was to oversee the project as a whole and give the final decision if the group couldn't agree on an idea democratically. Conor Fitch was the designated communications lead, meaning he was the go-between for our group and AWE. Sam Gardner was given software lead and Alex Pakeman was assigned as hardware lead, these roles meant the core technical elements of the project had someone overseeing them, allowing us to keep track of how each section was progressing in relation to our Gantt chart. Rishabh Arora was assigned as security lead to oversee tasks like threat analysis and Finley Atherton was designated documentation lead to ensure decisions were recorded both for ourselves and AWE to understand the final product and how we got there. Once we had these roles we also organised a weekly meeting among the group in addition to the update meeting with the external partner. This allowed us to discuss ideas among ourselves before bringing them to AWE, as well as update each other and

our supervisor on the progress we had made and any issues we had faced so they could be resolved.

Despite the clear roles that give each team member oversight over one specific area, our weekly team meetings allowed the entire team to contribute ideas to all areas of the project. This meant that fresh eyes could provide feedback and point out any possible areas that may need to be rethought. This worked well, especially in the early stages of the project when brainstorming lots of ideas helped to shape possible stretch goals as well as decide on which goals we thought were essential for project completion. After deciding on the goals for the project we finalised the Gantt chart and assigned each goal a time period we thought would be an appropriate time frame for goal completion. This original Gantt chart is displayed below in Figure 2.1.

FIGURE 2.1: Original Gantt Chart

This Gantt chart was a fairly accurate plan for the project, but as goals shifted and unforeseen circumstances came up we had to alter it slightly. Some of the alterations made include two of the stretch goals being deemed less important to the external partner (Camera OTP authentication and Encrypted Storage and Permissions) and have

therefore been marked red in the updated 'Actual' Gantt Chart in Figure 2.2. We also had to alter the Hardware development section of the chart. This was due to the parts order taking longer than expected to arrive, this pushed some of the tasks back a week as they couldn't be started without the necessary hardware. We compensated for this lost time by making additional progress on the software development and testing. This shift of focus made little difference to the overall plan, as we had originally planned to make progress on the hardware and software in parallel but the hardware delay allowed us to put more time into developing software and making progress in other areas.

The main deviation from the original plan was due to the 'Redacting Footage' section. This was due to a significant change in the way we chose to redact footage on the cameras. Originally, we discussed with AWE giving the distrusted party access to the footage so that they could delete sections of the footage if they deemed that it showed information outside the scope of a treaty. This changed a few times over the course of the project, first moving towards a toggle switch for the distrusted party to turn the cameras off and on, with a delay in footage transmission to give them time to react. We then pivoted again to allowing all footage through but redacting sections of individual frames, this would keep the video streaming but black out a section of video for a period of time, obscuring the material that the other party did not want seen. This redaction feature became a much bigger focus than we originally intended due to AWE's interest in making the system more attractive to the untrusted party. For this reason, the time spent on this feature increased significantly, as it took us much longer to develop and test. An updated Gantt chart reflecting the changes discussed, as well as a few other minor changes due to over- or underestimating the time required for a task is shown below.

2.2 Ethics and GDPR Management

Throughout the project, we made sure to mitigate any ethical problems by maintaining good communication with the external partner as well as each other. One of the main ways we did this was by ensuring transparency between AWE and ourselves via regular updates and strong documentation. We also upheld ethical project management practises internally, within the team, by ensuring fairness through task allocation which leveraged each team member's strengths. This ethical approach to project management helped ensure that the team operated effectively and maintained trust between all parties involved.

At the start of the project, the group signed a general risk assessment for lab usage; however, this did not cover some GDPR issues relating to camera testing that presented themselves later in the project. To comply with regulations, we wrote another risk assessment for continued camera operation in the labs. This did not cause any problems

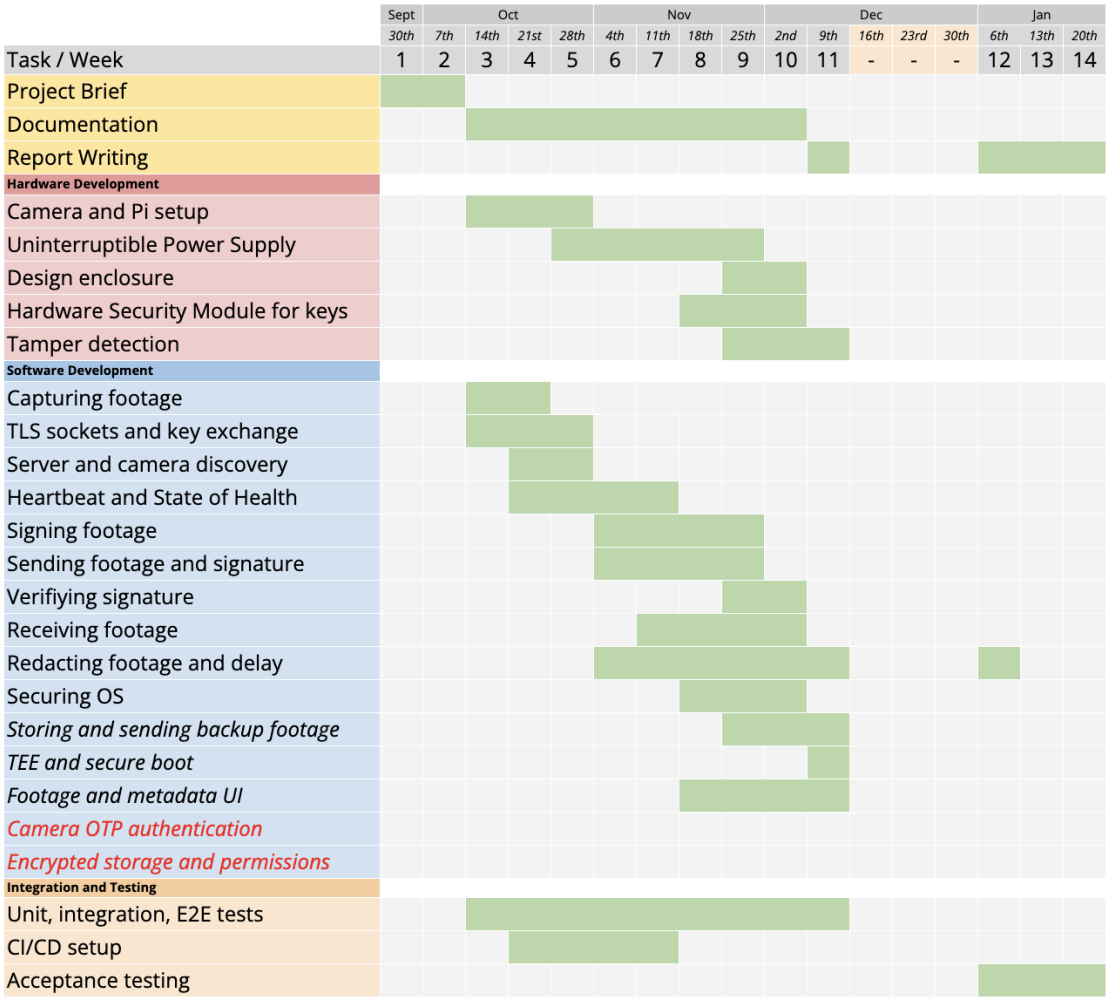


FIGURE 2.2: 'Actual' Gantt Chart at project completion

with project management as it fell nicely into the documentation side of the project and the risk assessment was completed, by Finley, prior to the Christmas break so we could keep the cameras running and test their longevity. A copy of this risk assessment can be found in the Appendix.

2.3 Technical Approach

During the project, we used several techniques to ensure the end result was a success. These ranged from different programming techniques to project management techniques and are detailed in this section.

The main techniques we used in this project were from the agile software development methodology. Agile is a development strategy agreed upon by a group of software development practitioners in 2001. This includes the practice of iterative development to ensure the end result is in line with the client's (or in this case external partner's)

vision. To do this we met with AWE weekly to ensure the work we had completed and the ideas that we had discussed during the week matched what they expected from the project. This also allowed us to ask any questions about where they would like us to go with certain elements, for example, with the footage redaction which evolved quite regularly in these meetings. Within the agile development principles, we used a scrum framework with 'stand-up' meetings to discuss what the group would work on over the next 'sprint'. A 'sprint' is a set amount of time to complete the goals discussed in the scrum meeting, while this set amount of time is meant to be short we found that two meetings every week and a week-long sprint was the best way to approach the project and balance it alongside our other studies. This framework is reflected in our Gantt charts in the section above which assign each task a set number of weeks for it to be completed in.

To ensure our management was effective throughout the project, we had to maintain efficient and regular face-to-face communication. As mentioned we did this twice each week in our 'stand-up' meetings with our supervisor and external client. Face-to-face communication is the sixth principle of agile software development agi and following this principle helped us not only to keep track of our progress but also to make sure that each team member was happy with their workload to reduce the likelihood of conflict in the future.

One of the other approaches we used to manage the technical side of the project was source control via the University's git lab. This had several benefits that helped us to manage the project effectively. The first benefit was risk management, as with lots of people working on different elements there is always the possibility of an unforeseen negative impact on another area but Git's versioning features allow these changes to be rolled back to preserve everyone's work. The commit history provided another benefit by maintaining a transparent record of contributions, helping to identify areas of progress and ensuring accountability for all team members. Finally, the collaboration and remote working features allow all members of the group to not only contribute but to stay up-to-date on the progress made by the team as a whole without having to wait for the next meeting.

Using git source control for the project also allowed us to automate tests when new commits were made to the project. This helped us to follow a test-driven development approach. To do this we wrote unit tests as well as end-to-end tests which would automatically be run on any new code. This practice ensures all code is of a high quality and helps to identify any errors early on in the process.

To summarise our technical approach to team management, we used lots of techniques including practices taken from agile frameworks and software tools like git. The combination of these helped the team to work together throughout the project and mitigated the risks of conflict within the team. All of the techniques we used were effective alone

but when combined formed the basis for a group that was able to work effectively across the different personalities and the different degrees and disciplines within the group.

2.4 Team Contribution

As mentioned earlier in this section, each team member was given a specific role relating to a different part of the project for them to oversee. These roles meant that each group member's expected contribution area was clear and well-defined. Rishabh, as the Security Lead, spearheaded research into potential attack vectors and identified vulnerabilities that required mitigation. He contributed to the hardware aspect by designing and 3D-printing the enclosure model. Additionally, he authored the Security section of the report and poster and prepared the slides for his portion of the presentation. Finley led documentation throughout the project which involved taking the majority of meeting minutes and ensuring anything important was recorded in an accessible format to aid with report writing. This then translated into starting the report, specifically the introduction and project management sections ahead of time to ensure deadlines were met. He also completed these sections on the poster and in the slides for the presentation and wrote the report conclusions at the end of the project. Conor was in charge of any communication between the group and our supervisor or external partner. He also led the software testing, setting up automated CI/CD (continuous integration and continuous delivery) testing pipelines with unit and end-to-end tests. Sam was the software lead and he and Alistair wrote the majority of the software for the networking and hashing algorithms. These were a major part of the project, which is why two team members shared the majority of the workload. Sam and Alistair also wrote the software and results sections of the report, poster and presentation. Alex, as the only electronics student, took responsibility for the hardware and the physical side of the system, which included ordering the parts and writing the code to get the footage from the camera to the Raspberry Pi used to control it. He also wrote the corresponding section of the report with help from Rishabh.

Chapter 3

Threat Research

3.1 General Review of Security Risks

The main goal of this project is to build a strong and safe CCTV system capable of functioning in environments marked by inherent distrust between parties. This was achieved through a comprehensive threat analysis aimed at identifying potential vulnerabilities and implementing appropriate countermeasures. The focus of this study is on ensuring resilience against physical manipulation, data breaches, denial-of-service (DoS) attacks, unauthorised access, and maintaining the integrity of recorded footage. By addressing these risks, the system can reliably meet its objectives in high-stakes scenarios, such as treaty verification.

3.2 Threat Categories and Countermeasures

The following sections detail the primary categories of threats and the corresponding countermeasures implemented to mitigate them.

3.2.1 Physical Tampering

This poses a significant threat, especially in hostile environments where unauthorised individuals may attempt to access or damage the system hardware.

Key Threats

- Forced entry into the enclosure using tools or brute force.
- Drilling or cutting into the enclosure to disrupt internal components.

- Physical interference with the system's power supply or data storage.

Countermeasures

- The 3D-printed enclosure features robust construction.
- A copper strip within the enclosure acts as an active tamper seal. Any attempt to access the internal electronics breaks the circuit, triggering an alert.
- The Zymkey hardware security module includes an accelerometer capable of detecting physical vibrations, such as drilling or cutting.
- A heartbeat mechanism monitors the operational status of all cameras. Any disruption is flagged as a potential tampering attempt.

3.2.2 Data Interception

Intercepting data transmitted from the cameras to the home server could compromise privacy or allow data manipulation.

Key Threats

- Man-in-the-middle (MITM) attacks.
- Eavesdropping on data flows.

Countermeasures

- RSA-based encryption ensures all data transmissions remain secure. Intercepted data remains unintelligible without decryption keys.
- Transport Layer Security (TLS) verifies the identities of communicating parties, protecting against MITM attacks.
- Sensitive sections of video footage can be redacted while preserving the integrity of the remaining data, satisfying transparency requirements without compromising security.

3.2.3 DoS Attacks: Denial of Service

DoS attacks aim to overwhelm the system, rendering it non-functional and disrupting data collection or transmission.

Key Threats

- Resource depletion on camera nodes.
- Network flooding that overwhelms the server.

Countermeasures

- Rate-limiting mechanisms on the server restrict the number of requests from a single source, reducing the impact of network flooding.
- Network traffic is distributed across multiple server instances to maintain system functionality under high demand.
- Local storage on each camera allows temporary data storage during network outages. The data is transmitted to the server once the connection is restored.

3.2.4 Access Control and Verification

Maintaining security relies on the system permitting only authorised devices and users.

Key Threats

- Unauthorised access to captured video.
- Spoofing of camera nodes.

Countermeasures

- Each node is equipped with a unique private key for identification by the home server, preventing unauthorised devices from impersonating legitimate nodes.
- Derived secrets, generated from a master secret, provide unique keys for each device, ensuring system-wide security even if one key is compromised.
- Granular access control policies restrict user and device actions based on their roles, minimising insider threats.

3.2.5 Integrity of Video Recording

Ensuring the authenticity and tamper-resistance of recorded footage is critical to the system's credibility.

Key Threats

- Data loss due to storage failures.
- Manipulation of recorded footage.

Countermeasures

- Cryptographic hashing is applied to recorded footage. Any modifications trigger tampering alerts due to hash mismatches.
- Immutable logging of all footage-related events provides a tamper-proof audit trail.
- Redundant storage across multiple locations ensures data integrity even if one storage location is compromised.

3.3 Challenges Encountered and Solutions

The following subsections outline the primary challenges faced during the project and the strategies employed to address them effectively.

3.3.1 Hardware Integration

Compatibility issues and firmware limitations hindered the integration of the Zymkey module with the Raspberry Pi. Collaborative efforts between hardware and software teams resolved these issues through iterative testing.

3.3.2 Balancing Security with Transparency

Achieving transparency for treaty verification while maintaining robust security posed unique challenges. Redactable signatures allowed sensitive video areas to be obscured without compromising authenticity.

3.3.3 Detecting Subtle Tampering

Identifying subtle tampering attempts, such as vibrations or drilling, required highly sensitive detection mechanisms. Future versions will fully integrate the Zymkey module's accelerometer to enhance detection capabilities.

3.4 Testing and Validation

The system underwent rigorous testing to validate its resilience and security features:

1. **Physical Tampering Tests:** Simulated attempts assessed the effectiveness of tamper detection mechanisms.
2. **Network Stress Tests:** Simulated high-traffic scenarios and DoS attacks ensured system reliability under stress.
3. **Data Integrity Tests:** Cryptographic hashing and immutable logging were tested by deliberately altering footage.
4. **Key Authentication Verification:** Spoofing attempts confirmed the robustness of asymmetric key authentication.

3.5 Future Improvements

Based on the lessons learned, several improvements are recommended:

- Full integration of the Zymkey module's features, including secure key management and accelerometer-based tamper detection.
- Advanced anomaly detection using machine learning to identify subtle system deviations.
- Blockchain-based logging to enhance transparency and immutability.
- Incorporating additional sensors, such as pressure and temperature sensors, to improve tamper detection.

Chapter 4

Hardware

In this chapter, we will explore the hardware solutions to the problems posed in this project, namely the cameras. The cameras must be able to collect footage, sign and hash it, and send it to the foreign server. Raspberry Pis were chosen as the core of the camera. Three different models were tested: Zero 2 W, 3B+, and 4. Although these models all are capable of using ARM TrustZone, this is mostly for testing purposes and does not provide the supposed security features. For this reason, it was decided to use an external Hardware Security Module (HSM): the Zymkey 4. The cameras are to be deployed in a hostile environment, it is imperative that hardware integrity can be ensured.

Two cameras were produced for this project to work in tandem. Firstly, this demonstrates the scalability of the design. In a final implementation of this system, there would likely be many cameras to ensure any relevant footage is collected pertaining to the verification in question. In addition, multiple cameras were produced to increase the security of the system through camera overwatch. By having multiple cameras as part of the system, we can ensure that each camera is in the frame of another one. This provides a huge benefit as it allows the cameras to monitor each other. This means that any physical attack on the system would have to affect multiple cameras simultaneously because otherwise it will be captured by another camera. This can be used together with tamper detection to greatly increase the security of the system.

4.1 Overview of the Enclosure

This project relied heavily on 3D printing since it offers the adaptability and efficiency required to design, test, and fabricate the unique enclosures for the CCTV system. The enclosures were made especially to hold the hardware tightly and guarantee functionality and simplicity of integration. Rapid iterations and affordable manufacturing made possible by 3D printing made it a great tool for meeting hardware needs for the project.

Emphasising its characteristics, materials, difficulties, and testing, this section addresses the design and execution of the enclosure. It also emphasises the vital part 3D printing plays in producing the intended results.

4.1.1 Enclosure Design

Three main goals directed the design of the enclosure:

1. Ensuring the internal components were protected from physical and environmental threats.
2. Offering a framework that fits all required parts, including the Zymkey 4 hardware security module, Raspberry Pi, and camera module.
3. Using elements that stop unwanted physical access helps to create tamper resistance.

4.1.2 Main Design Characteristics

Two main elements comprised the 3D-printed enclosure: the **base** and the **detachable lid**. The design incorporated the following features:

- **Custom Standoffs and Screw Holes:** Included in the base to firmly mount the Raspberry Pi, Zymkey 4, and camera module. These mounting points guaranteed against vibrations and ensured component stability during operation.
- **Ventilation Grid:** The lid included a perforated grid meant to efficiently dissipate heat, preventing overheating during continuous operation.
- **Tamper Prevention Copper Strip:** A copper strip was integrated around the interior to act as a Faraday cage, preventing direct contact with internal electronics and triggering alarms if tampering, such as tool insertion, was detected.
- **Camera Access Port:** A precisely cut window for the camera lens was slightly recessed to provide an unobstructed view while protecting the lens from damage.

4.1.3 Iterative Design Methodologies

Multiple iterations of the enclosure's design addressed security and functional needs. Early prototypes revealed areas for improvement, such as enhanced ventilation and better alignment of mounting points. Integration tests guided each successive iteration, ensuring the final design satisfied all project requirements.

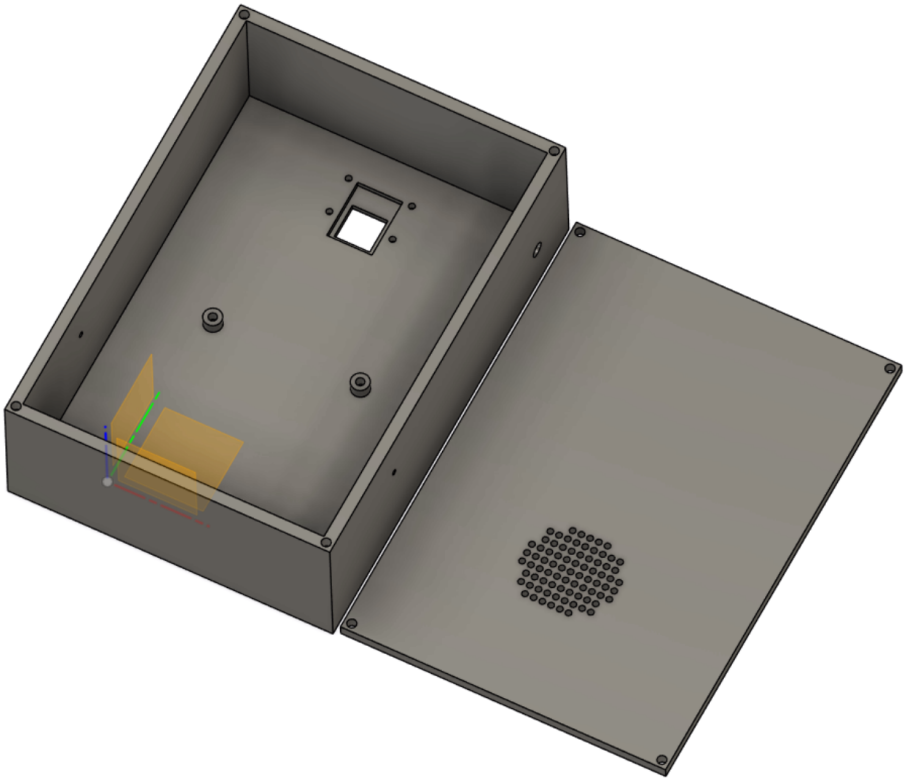


FIGURE 4.1: 3D Model of the Enclosure

4.2 Printing Methodology and Material Selection

PLA (polylactic acid) was selected as the sole material for the enclosures due to its simplicity, dimensional accuracy, and cost-effectiveness. Its characteristics made it ideal for prototyping and final production within the project's budget constraints.

4.2.1 Advantages of PLA

- **Low Warping Tendency:** Ensures consistent prints with minimal adjustments.
- **Affordability:** Allowed multiple iterations without exceeding budget.
- **Sustainability:** Being biodegradable and made from renewable resources aligns with sustainable practices.

4.2.2 Printing Parameters

The enclosures were fabricated using an FDM (Fused Deposition Modelling) 3D printer with the following key parameters:

- **Infill Density:** 30% to minimise material use while maintaining sufficient strength.
- **Orientation:** The lid was oriented to optimise the ventilation grid, while the base was printed flat to ensure accurate mounting points.
- **Supports:** Enabled for overhanging elements, such as internal standoffs and the ventilation grid.
- **Post-Processing:** Minor sanding was performed to remove support marks, edges were polished, and mounting holes were checked for accuracy.

4.2.3 Key Features of the Enclosure

The enclosure was designed to securely house the following components:

- **Raspberry Pi:** Mounted on standoffs, ensuring GPIO pins were easily accessible for additional connections.
- **Zymkey 4 Hardware Security Module:** Although not fully implemented, its potential features, such as accelerometer-based drilling detection and secure key management, were accounted for in the design.
- **Camera Module:** Positioned behind the access port using a custom bracket to ensure alignment and minimise vibration.

4.2.4 Challenges and Iterative Improvements

Several challenges were encountered during the design and manufacturing process, leading to iterative enhancements:

- **Alignment Issues:** Early prototypes revealed minor misalignments, which were corrected in the CAD model for a perfect fit.
- **Ventilation Efficiency:** The initial designs lacked sufficient airflow, leading to higher operating temperatures. Ventilation grids were expanded and optimised.
- **Structural Weaknesses:** Cracks around screw holes were addressed by increasing wall thickness.
- **Tamper Resistance:** Enhanced through tamper-resistant screws, copper strips, and better integration of the Zymkey 4 module.

4.2.5 Validation and Testing

The final enclosure design underwent extensive testing:

- **Drop Tests:** Confirmed the enclosure's ability to protect internal components from impacts.
- **Thermal Performance Tests:** Demonstrated the ventilation grid's effectiveness in maintaining safe operating temperatures.
- **Environmental Tests:** Simulated humidity and temperature fluctuations validated PLA's durability.
- **Tamper Resistance:** Attempts to breach the enclosure without proper tools highlighted the effectiveness of the design's security features.

4.3 Availability Protection

The availability of the CCTV footage is critical to maintaining trust and accountability. Any disruptions to the normal operation of the cameras could result in loss of footage which would compromise the integrity of the verification process. This in turn could lead to disputes between the inspectors and the host nation or undermine the trust and credibility of this system for verification. It is thus of vital importance that the cameras can remain functioning as autonomously as possible, even under adverse conditions.

This section outlines the measures implemented to safeguard the continuous availability of the system. In this project we have focused on protecting the functionality of the cameras in two circumstances which could prevent loss of footage: network failure and power failure. A connection failure could prevent gathered footage from being transmitted from the cameras to the host nation or from the host nation to the inspector. Instability, or total failure, of the power supply to the cameras could prevent the cameras from collecting footage in the first place.

4.3.1 Uninterruptible Power Supply

Power failures can occur due to a range of factors and, if unaccounted for, pose a threat to the continuous operation of the cameras. These can consist of complete loss of power (blackouts) or a reduction in voltage supply (brownouts) Savage (2023). Many of these events may occur incidentally due to failure of power infrastructure Atputharajah and Saha (2009) or excessive demand on the grid Wu et al. (2017). However, it would be remiss not to consider the possibility of the host nation intentionally sabotaging the power supply to the cameras directly to disrupt their operation and mask malicious

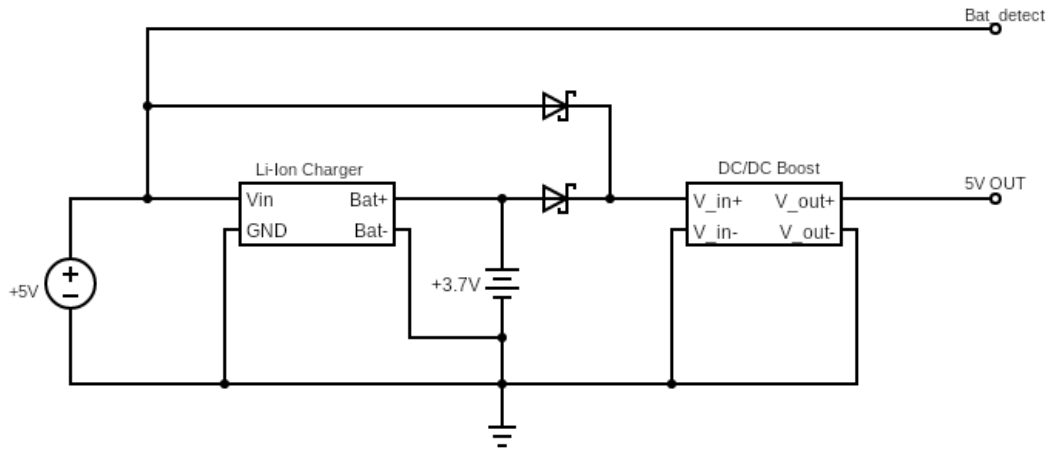


FIGURE 4.2: Block diagram of the UPS.

activity. To mitigate this threat, the cameras were equipped with an Uninterruptible Power Supply (UPS) to maintain stable operation in the event of power loss or instability.

A UPS provides clean, uninterrupted power to sensitive loads to protect them from outages and other adverse power conditions Bekiarov and Emadi (2002). These range in size and implementation depending on the requirements of the system. For this project, a simple implementation is used based on a "standby" or "offline" UPS. This provides outage and undervoltage protection but lacks many of the more advanced features offered by large-scale UPSes.

The cameras are based on Raspberry Pis which require 5V DC to operate, normally from a USB charger. The current requirements depend on the model used. The most power-hungry Pi tested in this project is the Pi 4, which requires a stable input of up to 2.5A when no USB peripherals are used. As a result, the UPS is designed to provide a stable 5V 3A DC supply. As decided in the specification, it should be able to keep the cameras running through a power outage for a minimum of one hour. The UPS designed in this project consists of four main parts: a battery, a charging circuit, a switching mechanism, and a voltage regulator. A block diagram of the system can be found in Figure 4.2.

A Lithium-Ion battery was chosen for this project due to their high energy density and discharge rate. The battery used consists of two 18650 cells in parallel with a capacity of 4500mAh. This supplies a nominal voltage of 3.7V and can provide a constant discharge current of 2.25A up to a peak discharge current of 4.5A. This provides ample capacity and current supply to power a Pi 4 for two hours or more.

An off the shelf charging circuit was used based on the TP4056 IC. This is a widely used IC for Li-Ion charging which protects the battery from over or under charging to

help prevent damage to the battery. The breakout board chosen also includes a DW01A battery protector IC. This further protects the battery by providing overcurrent and deep discharge protection when the TP4056 is not powered.

It was decided to use an off the shelf charging circuit for several reasons. Firstly, safety and reliability are paramount in this subsystem. Due to their highly reactive chemical composition, Li-Ion batteries can pose serious hazards if not handled correctly. An excessive charging current or overcharging of a cell can result in thermal runaway which in turn can cause combustion of the cell Chen et al. (2021). This poses a serious risk to people in the vicinity as well as the system itself. By using a commercially available, proven design, these risks can be minimised. Secondly, this project is majorly time-constrained with only a single member working on the hardware. As a result, it was necessary to simplify designs where possible to ensure a working system could be produced within the available time. Finally, using an off the shelf breakout board proved to be the most cost-effective solution. Due to mass manufacturing, purchasing a premade charging circuit costs less than the individual components required to produce a similar circuit ourselves.

Initially, a charging current of 1A was used. This is the standard charge current of our chosen battery and would ensure that the battery returns to a charged state as quickly as possible throughout a sustained period of power instability. However, it was found that this charging current drew too much power away from the Raspberry Pi and camera itself and caused instability, especially when it was under high computational loads. To mitigate this, the charging current was reduced to 250mA. This is well within the supply overheads so as no to affect the normal operation of the camera whilst charging the battery. Under normal operating conditions, the power supply will be constantly present, and the battery will only be used in exceptional circumstances thus a slower charging current should not hinder the effectiveness of the system.

The next key element of the UPS is the switching method. In the event of power failure, the UPS must seamlessly switch to providing battery power without interruption to the devices supply, allowing it to continue functioning as normal. Due to the difference between battery and supply voltage, diode "ORing" was an appropriate method to achieve this. This consists of the pair of Schottky diodes seen in Figure 3.1. Schottky diodes were chosen as they have a low forward voltage drop and fast switching time. The low forward voltage increases energy efficiency over a conventional p-n diode by reducing the energy lost as heat. The fast switching time is vital to maintain as constant a supply to the Raspberry Pi as possible.

The first diode has its anode connected to the power supply and the other has its anode connected to the positive terminal of the battery. Both cathodes are connected to form the output. When the 5V power supply is available, the first diode is forward-biased and allows current to flow. As the battery voltage is lower at 3.7V, the second diode is

reversed-biased and blocks reverse currents from flowing to, or from, the battery. When the 5V supply is removed, the second diode is now forward biased and current can flow from the battery to the load whilst the first diode prevents reverse current. This mechanism not only provides fast and efficient switching but also further protects the battery. The battery diode prevents reverse currents, which could cause damage, whilst the 5V supply is available. In addition, the battery will be disconnected from the load when charging. This is important as the falling charge current is used to determine when to stop charging the battery. If the battery is being loaded and charged simultaneously, it is possible that the charging circuit may never terminate and overcharge the battery. The charging circuit is powered by the 5V supply so, whenever the charging circuit is powered, the battery diode is reversed biased thus disconnecting it from the load.

The final part of the UPS is the voltage regulator. It is important that the Raspberry Pis receive a stable 5V supply. However, the battery only provides 3.7V and there is a voltage drop across the switching diodes. For this reason, it is necessary to boost and regulate the signal to ensure a stable 5V. This is achieved by using a boost converter. This is a DC-to-DC converter which outputs a higher voltage than its input. In this case, it was tuned to output 5.2V to account for line losses. This circuit outputs a constant 5.2V as long as the input voltage is lower than this (which is always the case due to the voltage drop over the diode). This is the final output that is used to power the Raspberry Pis. Time constraints necessitated that an off-the-shelf component was used once again.

4.3.2 Backup Storage

The other availability scenario considered for in this project is a disruption to the network. Each camera is equipped with 256GB of onboard storage with the goal of storing signed frames and heartbeat messages in the event of a local network failure. If a wider network failure occurs, signed frames should instead be stored on the foreign server. When a connection is re-established, these frames should then be transmitted in bulk. Unfortunately, this was not able to be successfully implemented within the available time and thus is left as future work. This is discussed further in Section 7.

4.4 Tamper Detection

The integrity and reliability of the surveillance system are paramount. Ensuring that the cameras remain not only functional but unaltered is vital to maintaining the credibility of the monitoring process. If a camera's hardware is compromised, it could be possible to deliver illegitimate footage and mask malicious activity. Tamper detection serves as a safeguard against unauthorised access to or modification of the cameras therefore helping to ensure the integrity of the gathered footage.

Three methods of tamper detection have been implemented in this project and are described in the following subsections. These are: perimeter detection, battery detection, and tap detection. Whilst these methods provide a good basis for detecting tampering, they are by no means exhaustive of possible attack vectors. Future work should consider improving upon these designs and accounting for more possible types of tampering such as a temperature attack.

A series of single-bit flags are used to record whenever tampering is detected. These flags are communicated periodically to the home server as part of a heartbeat message, which is discussed further in Section 5. The heartbeat acts as a state-of-health report for the system, it confirms a connection between the camera and home server separately from the video being sent. It contains the aforementioned tamper flags as well as a timestamp. The timestamp can be used to identify exactly when a tampering event is detected. When a tampering event is detected on one camera, the timestamp can be used to cross-examine with the footage from the other camera to check for the cause of the alert. The inclusion of a timestamp also makes each heartbeat message unique making it resistant to replay attacks.

4.4.1 Perimeter Detection

Seals are one of the most widely used methods for detecting tampering. They provide a method of identifying if an enclosure has been accessed. These usually take the form of a passive seal where a device or material becomes damaged or shows changes when they are manipulated. Examples of this include adhesive labels, foils, and optical fibres Johnston (2001).

Although simple to implement, there are several major issues that make passive seals unsuited for our purposes. They require an inspector to visit the premises and check the physical seals in person to determine if any tampering has occurred. This takes time and likely requires authorisation from the host. It may be possible to repair or replace a seal before this can occur. A simple passive seal provides no information other than whether tampering has occurred or not. It is important to know when the tampering happens since footage delivered by that camera can not necessarily be trusted once it has been tampered with. For these reasons, we decided to use an active, electronic seal.

The goal of the electronic seal is to detect if the camera's housing has been accessed and inform the inspector as soon as this happens. This was implemented by making a continuous circuit around the interior perimeter of the two-part housing. A series of thin copper contacts are attached to both the edge of the enclosure and the matching backplate in an alternating pattern, as seen in Figure 3.2, such that affixing the backplate results in one continuous loop of copper. At one edge of the enclosure, a single copper contact is connected to the 3.3V pin of the Raspberry Pi. At the opposite edge, another

contact is connected to a GPIO pin. The result of this setup is that the GPIO pin is held at high when the enclosure is screwed shut. As soon as the enclosure is unscrewed, the pressure holding the contacts together is lost and the circuit is broken leading to low being read on the GPIO pin thus detecting that the perimeter has been breached. Whenever low is read on this pin a flag included in the heartbeat message is set. The inspector can then see exactly when a tamper event has occurred.

This electronic seal has several advantages over a passive one. It provides immediate notification to the inspector with a timestamp so the exact moment that tampering occurred can be detected and reacted to. It is also reusable, resealing the housing will reconnect the circuit allowing multiple tamper events to be detected. However, it is best that this is used in conjunction with a passive, destructive tamper seal in order to account for any bypass of the electronic system. This was not implemented in this project due to the need to regularly open the enclosures for testing and development but should be kept in mind for an actual implementation or future work.

4.4.2 Battery Detection

It is important for the system to be able to detect when it is on battery power. Detecting a switch to battery power allows the inspector to log this event, providing a clear timeline of power-related issues. This is essential for troubleshooting and maintaining trust in the system's reliability. In addition, sudden transitions to battery power may indicate intentional tampering, such as someone attempting to disable the main power supply. Alerting operators to this change can prompt immediate investigation and response. Battery detection can be easily implemented by measuring the 5V input to the UPS. This is connected directly to a GPIO pin of the Raspberry Pi as indicated by "Bat_detect" in Figure 4.2. When the main power supply is available, this holds the pin high. If the power supply fails and the UPS switches to battery power this pin will now be low. When this happens a flag is set for transmission in the heartbeat message.

4.4.3 Tap Detection

Although useful for simple intrusions, it is possible to bypass the perimeter detection. Therefore, an alternative and more general method of detecting physical interference with the camera is required. This layered approach improves overall security by ensuring multiple mechanisms detect and report tampering attempts. Accelerometers can detect physical motion and vibrations which enables them to be used to identify potential tampering events that might compromise the integrity of the system. This could include moving the camera or vibrations caused by using tools to sabotage or bypass the enclosure. For this, we can use the accelerometer provided by the Zymkey HSM.

The Zymkey API stores and returns accelerometer data as a structure, `accelData`, containing the g force and tap direction in each axis. A tap is defined as an event where a sudden movement or physical impact is detected. The sensitivity of tap detection is configurable using the function setTapSensitivity(pct), where pct is a percentage from 0-100. By setting the tap sensitivity to a higher value (e.g., 100%), the device becomes more responsive to lighter taps or subtle movements. Conversely, lowering the sensitivity makes it less responsive, requiring more forceful impacts to detect a tap.

Due to this tunable sensitivity, it was decided to use tap detection to define tamper events. Initially, this was implemented by reading accelerometer data and checking if any of the tap values were non-zero. However, once a tap is detected, the direction remains set until a new tap is detected. This invalidates this method as, once a tap event occurs, tampering will be reported constantly. Instead, a tamper event was defined as a change in tap direction. Whenever the tap direction is different to the last time it was read, the corresponding tamper flag, to be sent in the heartbeat message, is set.

It was necessary to fine-tune the tap sensitivity value to optimize for a balance between detecting genuine tampering events and minimizing false positives caused by environmental vibrations or minor disturbances. To achieve this, six simulated scenarios were chosen, three representing tamper events (positive results) and the other three representing noise (negative results). The chosen tamper scenarios were drilling (simulated by placing a rotary tool, running at a constant speed, on top of the enclosure), hand tools (simulated by constant tapping of the enclosure), and removal of the camera (simulated by lightly shaking the enclosure). The chosen noise scenarios were ambient room vibrations, traffic vibrations, and footsteps. However, the accelerometer measurements taken for traffic vibrations proved to be highly unrepresentative due to the irregularity of traffic. Ten sensitivity levels were tested for each scenario, ranging from 10% to 100% in increments of 10%. For each sensitivity, ten samples were gathered at one-second intervals. This data was used to calculate accuracy, precision, recall, and F1 score at each sensitivity interval.

Figure 4.3 presents the evaluation metrics relative to tap sensitivity. The accuracy remains relatively constant across sensitivity levels and varies between 0.4 and 0.6. This highlights major issues with the methods of this experiment as random guessing with two outcomes would provide an accuracy of 0.5. It can be seen that the precision remains high throughout all sensitivity values, only dropping from one at the highest and lowest sensitivities. This means that the system is extremely resistant to false positives. This can be contrasted with the exceptionally low recall, performing worse than any other metric. This is a huge problem as it demonstrates that a large portion of tamper events go undetected. F1 score represents a balance between precision and recall. Whilst the high precision raises this metric above the poor performance of recall, it still demonstrates poor performance for correctly identifying tampering. Overall, a sensitivity of 70% seems to provide the best performance so this was selected.

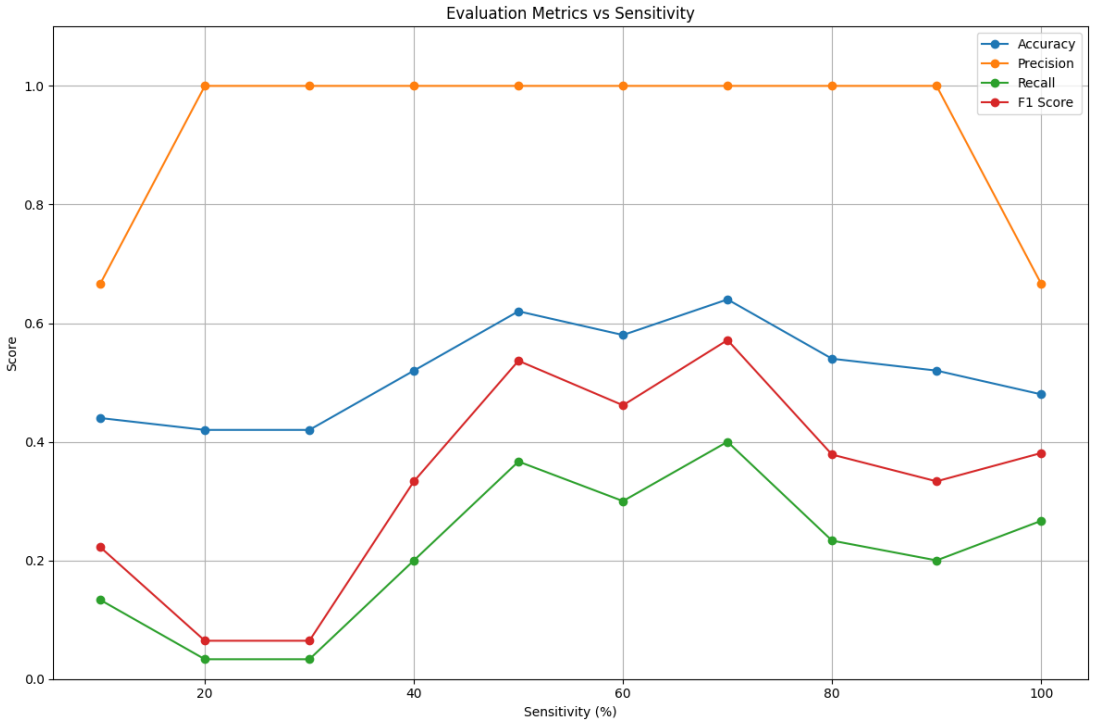


FIGURE 4.3: Accelerometer detection performance using change in tap direction at different sensitivity levels.

The data shown in Figure 4.3 demonstrates that the tap detection method used performs poorly. For this system, recall should be considered more important than precision. This is because unnoticed tamper events pose the biggest threat to the integrity of the hardware and thus the system as a whole whilst false negatives cause very little issue. The high precision but low recall at all sensitivities indicates that the method used is inherently insensitive. This may be due, in part, to how a detection is defined. By looking for a change in tap direction, this method disregards if a tap occurs in the same direction. Future work should explore other, more sensitive methods for defining a detection based on the raw acceleration data or using interrupt-based detection rather than solely a change in tap direction. Collecting more data over a longer time period would also be helpful with the development of a more accurate detection mechanism.

Chapter 5

Software

5.1 System Architecture

The system architecture encompasses all the nodes in the system, spanning across the foreign party network and the home network. This section outlines the different types of nodes that make up the system, and the changes that were made in response to the requirements.

At the start of the design, there were two types of nodes: a camera node and a home server. This would allow data to be sent from a camera node back to a home server for storage. However, given the key aspects of this project are trust and transparency, a new node was added to the architecture: the foreign server. This is a node that sits between the camera nodes and the home server, acting as a basic proxy service with some additional functionality.

This provides several benefits, including the other party being able to see exactly what data is being sent out their network, improving trust and transparency; a single place for them to control data to the internet; the ability for them to redact video within the bounds of the treaty; the camera can be "offline" (connects over the local network only). The addition of the foreign server means that the logical network architecture was restructured, forcing all requests and responses to go through the foreign server.

The order of node startup does not matter, but the camera nodes will not send any data until they are connected to the foreign server. The foreign server will not accept connections from the camera node until it is connected to the home server. Therefore, for the three nodes to connect, the home server must be online, allowing the foreign server to connect and subsequently accept connections from camera nodes. Back-end foreign server code (such as connecting to the home server) cannot run until the localhost is connected via a browser, initializing the web server. This means that the foreign server GUI must be active for the node system to be fully connected.

5.2 Cryptography

The cryptography model required a protocol that allowed an authenticated connection to be established, with all data flowing through the foreign server. Using end-to-end encryption between the camera and the home server was not an option, as this prevented transparency. The following is a short list of candidate models.

- Static public/private key pair for the home server and camera nodes. The camera node would send a handshake message to the home server, which would respond with a signed challenge. The camera node would then sign the challenge in response, to prove it is indeed the camera node.
- Certificate-based authentication system, where the home server would also act as a certificate authority and would provide and renew certificates to the cameras as required.
- The home server maintains a "master secret", and each camera node contains a "derived secret" - a hash of the camera identifier and master secret. This would give the camera nodes a unique piece of information that no one else could own, authenticating themselves. The derived secret would be sent under asymmetric encryption with the home server's public key.

5.2.1 General Model

Between these options, the "master secret" option was chosen. This is because it allowed the public/private key pairs to be rotated at will, by attaching the public key into the handshake message and reconnecting. As the message contains the derived secret and is encrypted under the home server's public key, it cannot be even partially restructured to contain the correct derived secret and a different public key. Therefore, it is safe to include a new public key in the handshake, and the home server can authenticate it by checking the derived secret.

The home server can reconstruct derived secrets by hashing the stored master key with the camera identifier that is in the handshake. If the secrets match, then the party communicating must be the camera. This provided a simple and secure authentication model for the cameras and home server.

5.2.2 Asymmetric Cryptography

RSA is the selected asymmetric algorithm. This is a well-tested and robust algorithm and has signature/verification and encryption/decryption properties. It is important

that the algorithm had the encryption property, as this is required to mask the derived secret when it is sent to the home server, using the home server's public key. This meant elliptic curves were not an option for the home server key, and as such it was decided to keep a consistent algorithm for all asymmetric operations. Therefore, RSA was selected.

A Diffie-Hellman (DH) or Elliptic-Curve-Diffie-Hellman (ECDH) key exchange followed by a symmetric encryption for the masking could have been done, but it was simpler to utilize the known RSA keys for a simple encapsulation. The existence of the derived secret provides the authentication inside the encapsulation.

A problem with the RSA encapsulation was with the maximum message size being linked to the key size; a 1024-bit key can encrypt a message of a maximum size of 1024 bits. As the handshake contains the camera's public key (1024 bit), the entire buffer was taken up, with no space for the derived secret and other associated data. The new camera node public key could not be encrypted separately, as the message would not contain the derived secret, opening it up to man-in-the-middle attacks. This was solved by using 2048-bit keys for the home server, and 1024-bit keys for the camera nodes, allowing a maximum 2048-bit message to be encrypted, meaning that a 1024-bit camera node public key could fit inside the buffer, and other data could be stored alongside it.

The cryptography functions, in particular key and certificate generation, were further utilized for the TLS socket implementations. This allows for one unified cryptography interface to be used for both TLS sockets and providing signatures for heartbeat and video messages.

5.2.3 TLS Sockets

Custom TCP socket classes (`TCPServer` and `TCPClient`) were implemented for two reasons. Firstly, the classes provide a simple object-oriented interface, matching the rest of the program. This created an easily maintained codebase, and also reduced the reliance on free-function C function calls. This reduced global namespace pollution across the codebase, as C++'s `#include` system automatically includes any previously included symbols. Secondly, it means TLS functionality can seamlessly be used with the sockets, all in one place, by using functions in the OpenSSL library to secure the connections.

This provides point-to-point encryption, between the camera nodes and the foreign server, and the foreign server and the home server. It is important to note that there is no end-to-end encryption between the camera nodes and the home server, as this would violate the trust; the foreign server could not be certain of what data was leaving the network. As such, confidentiality from third parties is maintained by using point-to-point encryption, with end-to-to end authentication.

5.2.4 Serialization

Both the DER and PEM formats are used for storing keys. The PEM format is used to store keys in the filesystem, such as the home server's key pair. The DER format is used to include keys in messages being sent over sockets.

PEM is used for file storage because it allows for easier debugging, given the base64 encoding used. DER encoding converts the keys into a `unsigned char*` type, which is more optimal to send over a socket, as no extra encoding is required; the socket buffer type is also `unsigned char*`.

5.2.5 Memory Security

The OpenSSL library provides a number of macros and functions to ensure safety at the memory level. This includes safe allocation and de-allocation, and constant-time memory comparison operations, helping provide side-channel security against certain memory attacks within the software.

The allocation links into a "secure heap" that the OpenSSL library designates, meaning that all signature, plaintext, ciphertext, and key buffers are stored securely when in memory. The functions `CRYPTO_secure_malloc`, `CRYPTO_secure_free` and `CRYPTO_memcmp` are used for operations involving sensitive data. Each node initializes its secure heap on program startup. The heaps are cleaned up on program destruction.

We utilized the secure heap, as it has many security features:

- Prevents memory swapping: data in the secure heap is locked into the RAM and is never swapped to the disk by the operating system. This reduces the chance of exposure from memory.
- Automatic memory zeroing: once a pointer in the secure heap is freed, the memory that was pointed to is automatically zeroed, preventing residual data from staying in memory for longer than required.
- Buffer overflow protection: OpenSSL prevents buffer overflows by protecting areas around valid pointers in the secure heap, preventing data from being exposed out of a secure allocation.

File security is increased by using the `fopen_s` function rather than the deprecated `fopen` function. This is used for serializing and loading keys and is oriented towards preventing any attacks that might try to exploit pointer logic in running binaries.

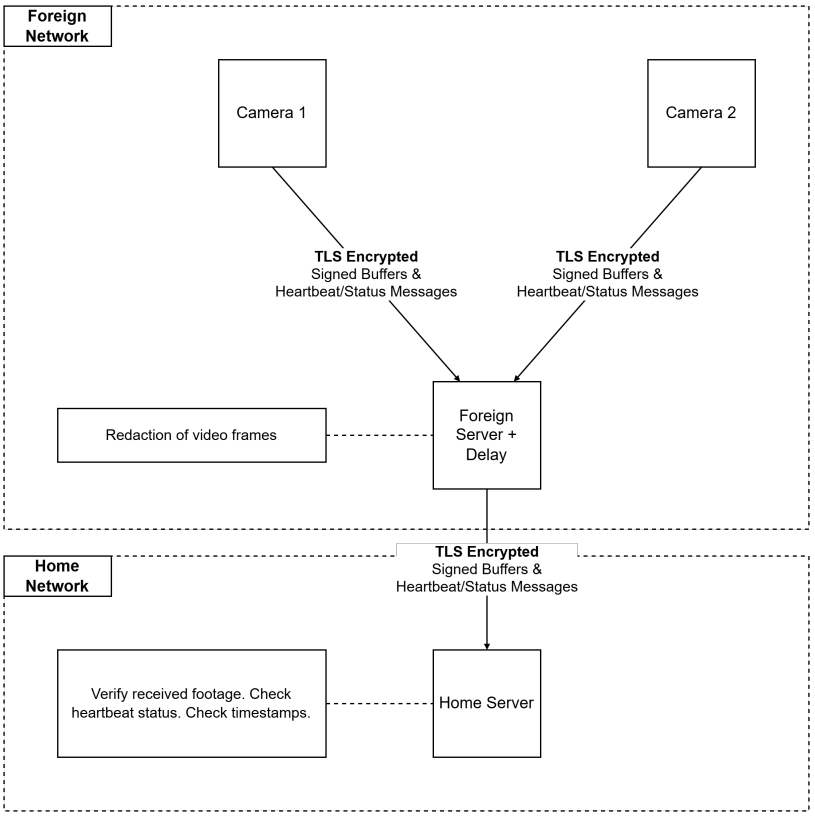


FIGURE 5.1: Network Model

5.3 Network Model

The network model describes how requests and responses flow between the three types of nodes. The camera nodes use three types of requests: handshake initialization; heartbeat information; and video frame information. As there are many camera nodes and only one home server node, the foreign server attaches the camera's socket identifier to the outgoing camera messages destined for the home server. The home server attaches the same identifier to the responses, so the foreign server can direct the response to the correct camera node.

In this model, the foreign server acts as a proxy node, forwarding messages between the home server and camera nodes. The foreign server withholds video frame messages for a short time interval, allowing backtrack redactions from the front end (this is described later on). Other request types are immediately forwarded to the destination node.

5.3.1 Request Types

This section provides an overview to the four request types used with the networking protocol; the three mentioned previously are exclusive to the camera node - the home

server also uses the handshake response. The foreign server can add to the lookup map of the video info request.

Each specific request type is wrapped inside a `NetworkRequest` type, with information for the target node on how to cast the internally stored request back into its source type.

5.3.1.1 Camera Handshake Request

```
struct CameraHandshakeRequest {

    Handshake timestamp.
    std::array<unsigned char, TIMESTAMP_LENGTH> timestamp;

    Camera identifier initiating the handshake.
    std::array<unsigned char, CAMERA_IDENTIFIER_LENGTH> identifier;

    Camera's public key for this session.
    std::array<unsigned char, RSA_CAMERA_KEY_LENGTH_BITS / 8 + 12> public_key;

    Camera's derived secret.
    std::array<unsigned char, DERIVED_SECRET_LENGTH> derived_secret;
};
```

FIGURE 5.2: Camera Handshake Request Struct

The handshake struct contains 4 fields: the timestamp, identifier, public key, and derived secret. The identifier is required for the home server to know which derived secret to compute, to check for a match. As the entire struct is serialized and encrypted, the derived secret is mixed with the other information into the ciphertext; its existence authenticates the entire struct.

All handshakes are timestamped, meaning that if an already-connected node attempts a new handshake, only if the timestamp in the handshake is newer than the timestamp that was part of the previous handshake, will the home server accept the camera node's new public key.

A malformed message or stale timestamp will result in the connection not being refreshed, the connection being terminated, and a message logged to the home server that there was potential tampering. This means that replay attacks are not feasible. If the connection is rejected then the camera will error, triggering the program to re-run, and a proper handshake with the home server will be performed.

Unique to this type of request, it is encrypted under the home server's public RSA key. This means that it is never sent in a raw serialized form; it is serialized, encrypted, and stored inside a network request, which is then in turn serialized and sent.

5.3.1.2 Home Server Handshake Response

```
struct HomeServerHandshakeResponse {

    Hash of the derived secret.
    std::array<unsigned char, SHA256_DIGEST_LENGTH> hashed_derived_secret;

    Server signature for authentication.
    std::array<unsigned char, RSA_SERVER_SIGNATURE_LENGTH> server_signature;
};
```

FIGURE 5.3: Home Server Handshake Response Struct

The home server will respond to the cameras' handshake requests, given the authentication and other checks pass, so the cameras know they can start sending heartbeat and video frame messages. The home server responds with a signature of the hash of the derived secret so that the camera knows the home server is online. An oversight here is the lack of a timestamp in the response. Unstable development branches have the timestamp included, preventing replay attacks of a home server handshake response.

5.3.1.3 Camera Heartbeat Info

```
struct CameraHeartbeatInfo {

    Heartbeat flags.
    std::array<unsigned char, FLAGS_LENGTH> flags;

    Camera identifier for the heartbeat.
    std::array<unsigned char, CAMERA_IDENTIFIER_LENGTH> identifier;

    Heartbeat timestamp.
    std::array<unsigned char, TIMESTAMP_LENGTH> timestamp;

    Camera signature for authentication.
    std::array<unsigned char, RSA_CAMERA_SIGNATURE_LENGTH> signature;
};
```

FIGURE 5.4: Camera Heartbeat Info Struct

The heartbeat information includes information about the camera itself, including its status inside the foreign network. This status information is stored inside the 1-byte "flag" byte. The flags include power source information, network information, and a range of tamper flags such as accelerometer and copper contact information. A timestamp is included before signing so that heartbeats cannot be faked with old messages.

5.3.1.4 Camera Video Info

```
struct CameraVideoInfo {
    Camera signature for authentication.
    std::array<unsigned char, TIMESTAMP_LENGTH> timestamp;

    Camera identifier for the video frame.
    std::array<unsigned char, CAMERA_IDENTIFIER_LENGTH> identifier;

    Video frame buffer.
    unsigned char *video_stream;

    Camera signature for authentication.
    std::array<unsigned char, RSA_CAMERA_SIGNATURE_LENGTH> signature;

    Frame index so the home server can place the frames in order to reconstruct the video.
    size_t frame_index;

    The root hash of the merkle tree.
    std::array<unsigned char, REDACTABLE_SIGNATURES_HASH_SIZE> merkle_hash;

    The hashed piel redactions for reconstruction at the home server.
    std::map<size_t, std::array<unsigned char, REDACTABLE_SIGNATURES_HASH_SIZE>> redacted_pixels_mapping;
};
```

FIGURE 5.5: Camera Video Info Struct

The video information struct represents a single frame of the video from one camera. The camera attaches the timestamp, identifier, and video-frame. The video frame is a still image from the video, and its length is $w * h * 3$ - it has no special encoding or compression. The Merkle hash is the root hash created from the redactable signature algorithm described below. The signature includes the Merkle hash, timestamp, and identifier. The redacted pixel mapping is left empty by the camera, and is set by the foreign server, if any redactions take place. This is also described later.

5.4 Camera Node

The camera node is the software component for the Raspberry Pi that records footage. It has three main functions: handshakes, heartbeat messages, and video frame messages. The first thing it does when coming online is to obtain the foreign server address. This is done in one of two ways: if a fixed server IP address has been given as a command line argument, then this is the fixed address used; otherwise the camera attaches itself to a pre-configured multicast group and waits for an address to arrive - the foreign server advertising itself. An oversight here is that there is no way to confirm it is truly the foreign server sending the address. However, as this is on the foreign server's network, it is their responsibility that they don't have an intrusion on the network over that multi-cast group.

Once the address is received, the camera node can send messages to the home server, using the foreign server as a proxy. The first message is the handshake, to tell the home server that a camera node is ready to send messages. The handshake message is encrypted under the home server public key, as described above, and sent to the home server via the foreign server. Once the handshake response is received and authenticated, (confirmed to have come from the home server), the camera spawns two threads - one for sending heartbeat messages, and the other for sending video frames.

The heartbeat information is sent to the home server for inspection. These messages pulse on a fixed time interval and include information that allows the home server to detect potential tampering. Status flags included are the power source (grid or switched to the battery), copper contact integrity, and accelerometer data. Anomalies can be logged and handled by the home server.

The video frame function uses the webcam of the Pi, which in this case is a hardware-specific module, to record frames at 30 fps. Each frame is tagged with an index (to maintain order on sockets), and wrapped in a request object. Metadata includes a timestamp and camera identifier, and again the request is signed.

As each of the two described messages can be sent simultaneously, mutexes are used to lock sockets, so that there is no merging of data when two socket write operations occur very close together.

5.5 Foreign Server Node

The foreign server code is run on a server that sits on the foreign network. It acts as a proxy server, and can, by design, read all incoming and outgoing messages between the camera nodes and home server. The exception is for the camera node handshake, which is required to be encrypted to maintain the confidentiality of the derived secret.

For messages from camera nodes, the foreign server attaches the camera's socket ID to the network request object. This is so that when the home server creates a response, it can also attach the same socket identifier to the message, allowing the foreign server to be able to direct a single stream of responses, from the home server to the correct camera nodes.

An improvement here would be for the camera to determine its own socket identifier as the foreign server sees it, so that it can be included in the signature, and the home server can be certain that the foreign server isn't switching socket identifiers tagged to messages.

Aside from video frame messages, all network requests are immediately forwarded to the destination node. For video frame messages, the request is withheld for a fixed interval,

allowing backtrack redaction to be applied to these frames, and then sent onward to the home server. Because all messages are timestamped, and the home server will know its round-trip time to the foreign server, the timestamps can be checked to ensure the foreign server is not delaying messages for any period. The timestamps are included in the signature too, meaning they cannot be forged.

5.5.1 Front end

To increase ease of use on the foreign server, which has the dedicated functionality of "redacting", there is a front-end graphical user interface that integrates directly with the received video frames, allowing video manipulation within an agreement, before sending them onto the home server. The front end contains two specific classes: a *RedactionApplication*, and a *RedactionSingleStream*. The redaction application holds multiple single streams; one for each connected camera, indexed by the socket identifier.

This allows for seamless additions of cameras as they come online, and because the socket identifiers are used to index streams, they integrate into the back-end very easily. Members of the foreign party can redact footage as it arrives on their streams.

5.5.2 Redacting Footage

The GUI displays each of the camera streams, with a selection of tool buttons below. There are buttons for: rectangle/circular/polygon redactions; removing all redactions; undoing and redoing the last action; and confirming all redactions. Upon the confirmation of redactions, the application converts the redaction shapes into a vector of pixel positions, which are registered against identifiers. The back-end application has a pointer to these lists, which are read from and used to blackout parts of the frames currently being held.

Because frames are held for an interval, "backtrack redaction" is supported. This means that the redaction of the current frame and onwards, is retroactively applied to the previous n frames, n being the number of frames held in a time interval - for 5 seconds at 30 fps, this would be 150 frames.

The back-end application maintains a pointer to the front-end application instance, allowing for the redacted pixel map to be viewed from the back end, with a mutex lock to prevent multithreading errors. For every frame about to be released, the redaction algorithm is run, which gets the redaction pixel positions for the socket identifier who sent the video frame, and applies the blackout - each stream is individually redactable. For an empty redaction pixel list, nothing changes. This redacted frame is then sent back to the home server.

Technical details on the redaction are discussed in the section 5.7.

5.5.3 Localhost

The `Wt` library is used to display the streams and redaction tools on a website accessible by localhost. In order for the foreign server to run, the localhost website must be opened, as this fires off the `Wt` events that unblock the back-end application. This means that for the camera nodes and home server to connect, the localhost must be active. This was a design decision made to ensure the foreign server is ready to begin redacting at any time, and only once the foreign party is ready, will any other nodes else connect and send information.

5.6 Home Server Node

The home server is the server that sits on the local home network. It receives requests from the foreign server, proxying camera node requests/information. These requests are processed and responses are generated if needed (specifically for handshake responses). The home server is also responsible for handling heartbeat messages and video frames.

Heartbeat message flags can be checked for anomalies and logged if there is anything abnormal. These messages are received constantly over a short time interval, allowing for fine-grained checking. All heartbeats are timestamped and signed, providing replay-safe authentication.

Video frames are also signed, and their unique information comes in the form of a frame index. This is required because frames can reorder themselves over the socket or when released from the foreign server, as each frame is released by a separate thread (after withholding for redaction). The frames are reconstructed, in order, to form chunks of a video that form fixed time chunks. These chunks are then stored under a folder (named by the camera ID) and a file (the timestamp of the first frame in the chunk).

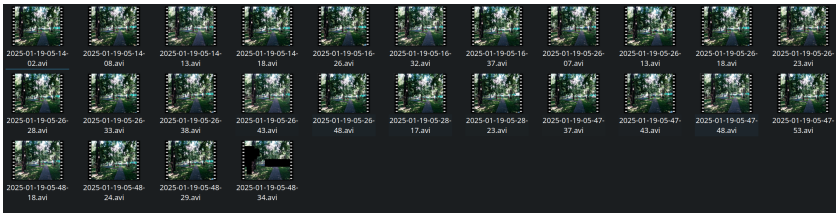


FIGURE 5.6: Chunked Storage of a Video

The video format chosen was .avi, because is can be formed from raw frame information, ie 3-byte pixel colour and a long string of colours, all being joined together. This meant that no extra encoding was needed, and was the simplest option which didn't compromise the video quality.

The home server is assumed, for the purposes of this project, to be completely secure. This means that the master secret and static private key are unobtainable, as the leaking of this information would compromise the entire system, and would require an entire camera recall.

5.7 Redactable Signatures

Originally, before redaction was implemented, video frames were signed and sent. When redaction was implemented, this invalidated the signatures on the frames, because the frame being sent (containing blacked-out regions) would no longer match the signature of the original frame that the camera recorded. To combat this, we introduced "redactable signatures". This is a type of signature that allows for redactions, whilst still being able to authenticate the message (video frame).

The general idea is to create a Merkle tree of the hashes of pixels, from individual elements to pairs of elements, all the way up to a "root hash". The root hash is signed by the camera and included in the video frame message. The term "element" refers to a pixel or a grouping of pixels, depending on the "redaction density", explained below.

The foreign server then performs any redactions desired, and blacks out pixels. These blacked-out pixels are stored in a lookup map with a structure of { position: $Hash(colour|x|y)$ }. The home server receives the redacted image, and recomputes the hash tree, using substitute hashes from the lookup map where required.

This allows for the computation of the root hash, without knowing what the original colours that have been blacked out were. The home server then checks the root hashes match and verifies the camera node's signature on the root hash. If both of these checks pass, then it is confirmed that the redacted frame is authentic. This frame is then allowed to be stored. The lookup map is stored in the video request.

The Redactable Signatures algorithm relies upon 4 constants and sets made up of the natural numbers up to but not including their values.

$$
\begin{aligned}
w &\in \mathbb{N}^+ \\
W \subset \mathbb{N}, W &= [0, w) \\
h &\in \mathbb{N}^+ \\
H \subset \mathbb{N}, H &= [0, H) \\
d_x \in \mathbb{N}^+ \text{ s.t. } & d_x | w \\
D_x \subset \mathbb{N}, D_x &= [0, d_x) \\
d_y \in \mathbb{N}^+ \text{ s.t. } & d_y | h \\
D_y \subset \mathbb{N}, D_y &= [0, d_y)
\end{aligned}
$$

Where w represents the width of a frame, h is the height of the frame, and d_x and d_y are the *Redaction Density* constants. Redaction Density is the measure of how many pixels are grouped together for each hash. d_x and d_y refer to a rectangle of pixels grouped together in a redaction, meaning that if any individual pixel in the area is redacted, the entire area is redacted.

A colour is defined using 3 bytes, in RGB format.

$$C = \{(r \in [0, 255], g \in [0, 255], b \in [0, 255])\}$$

An image is defined as its pixels' coordinates and colours.

$$I = \{(x \in [0, w), y \in [0, h), c \in C)\}$$

The image I can take another form, where instead of being indexed by $x \cdot y$, it is instead indexed by an integer index i, and treats the image as if it was laid out in a single row, going row by row from the original row/column form. This new image I' and original I are isomorphic, as shown below.

$$I' = \{(i \in [0, w \cdot h), c \in C)\}$$

$$f : I \to I'$$
$$f((x, y, c)) = (x + y \cdot w, c)$$

$$f^{-1} : I' \to I$$
$$f^{-1}((i, c)) = (i \bmod w, \lfloor i/w \rfloor, c)$$

I' can also be considered a function of indices to pixels, for all indices $0 \leq i < w \cdot h$.

The first of a sequence of hash functions can be defined. The first in the sequence calculates all the hashes of the image, in blocks of size d_x by d_y.

$$L_0 \subseteq \mathbb{N}, L_0 = \left\{ x \in \mathbb{N} \middle| 0 \leq x < \frac{w \cdot h}{d_x \cdot d_y} \right\}$$
$$H_0 : L_0 \to \mathbb{N}$$
$$H_0(i) = \text{hash}\left(\left\{i + x + w \cdot y, I'(i + x + w \cdot y) | x \in D_x, y \in D_y\right\}\right)$$

This new function H_0 is the first of a sequence, defined as such.

$$L_{i+1} \subset L_i, L_{i+1} = \{x \in L_i | \exists k \in L_i, 2k - 1 = x\}$$
$$H_{i+1}(x) = \begin{cases} \text{hash}(H_i(2x), H_i(2x + 1)) & \text{if } 2x + 1 \in L_i \\ H_i(2x) & \text{otherwise} \end{cases}$$

Note that $|L_{i+1}|$ is strictly smaller than $|L_i|$, unless $L_i = L_{i+1} = \{l\}$. The *root hash* is the first $L_i = \{l\}$ to appear. This root hash is signed and will remain the same for any fixed frame.

To redact, a middleman takes the frame, decides which pixels it would like to redact, and redacts all chunks containing redacted pixels.

$$R \subset \mathbb{N} \times \mathbb{N}$$

$$R = \{(i, H_0(i)) \text{ where chunk } i \text{ contains at least 1 redacted pixel}\}$$

Now the image has changed, the hash tree and therefore root hash has changed. To restore the original hash tree, H_0 can be altered to replace the redacted chunks' incorrect hashes with the correct hashes provided by R.

$$H_0' : L_0 \to \mathbb{N}$$

$$H_0'(i) = \begin{cases} h, & \exists (i,h) \in R, \\ H_0(i), & \text{otherwise.} \end{cases}$$

With the correct R, $H_0' = H_0$, therefore the hash tree and root hash are identical, and the signature is valid.

Because there are only 16777216 different colours in RGB, for a known pixel location, only 16777216 hashes have to be made to find a match, making the redaction seem breakable; that is, redactions could be inverted by brute-forcing hashes to match in the lookup map. However, by utilizing the redaction density and setting $d > 1$, pixels are combined before hashing, making it exponentially harder to brute force the hashing: for every increase of 1 in the redaction size, the number of hashes required increases at a polynomial rate, even faster so if both dimensions' density is increased.

5.7.1 Redaction Video Frame Flow

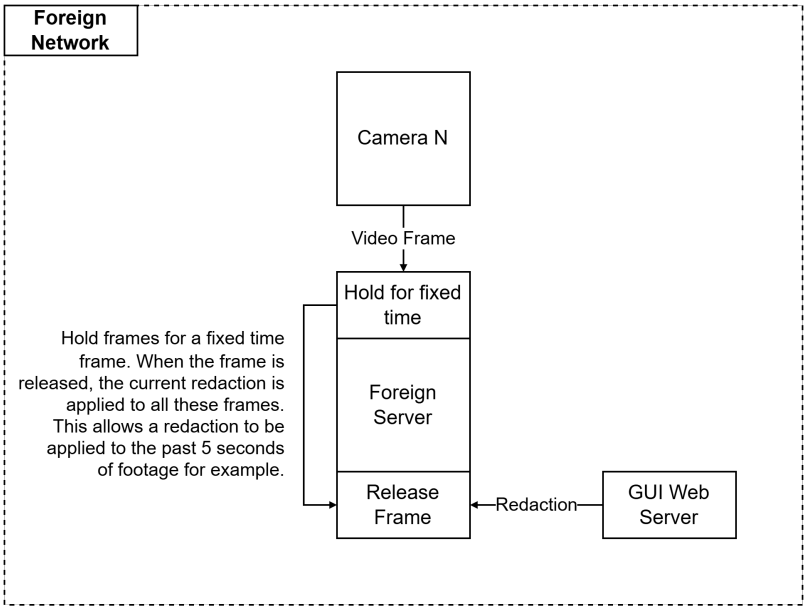


FIGURE 5.7: Video Frame Flow for Redaction

5.7.2 Example

The following two images show the redaction as it's drawn onto the video, and the redaction with the redaction density set, showing the groupings of pixels that are being redacted.

FIGURE 5.8: Redaction: Raw

FIGURE 5.9: Redaction: With Density 32px

5.7.3 Performance

The following graph shows the performance of redactable signatures over the different nodes that a camera instance was running on; the Pi3B+ and the Pi4:

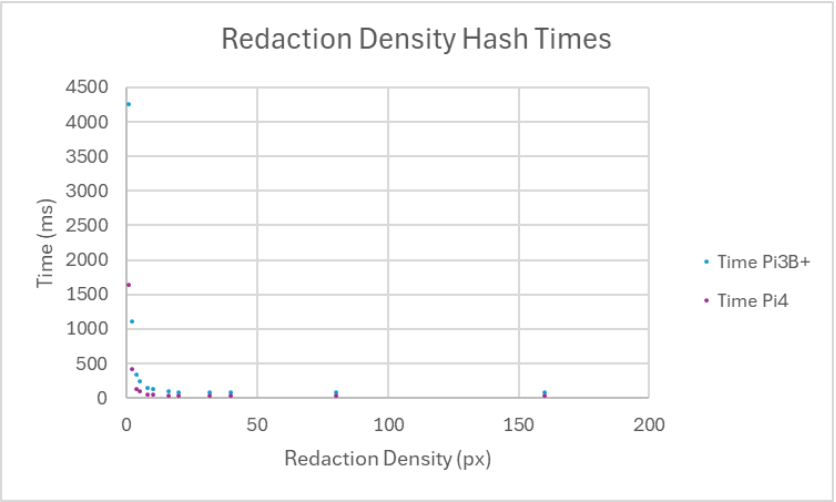


FIGURE 5.10: Redaction Density Graph

Redaction Density	Avg. Time Pi3B+ (ms)	Avg. Time Pi4 (ms)
1	4251.4417	1642.9856
2	1108.2125	423.96205
4	334.51523	127.68029
5	243.85074	92.850347
8	148.28016	56.760692
10	124.74309	47.581565
16	98.591087	37.414992
20	92.098572	34.804669
32	85.778274	32.097158
40	84.044408	31.411503
80	81.020455	30.545887
160	80.340906	30.342333

TABLE 5.1: Comparison of Merkle-Hash Time for Pi3B+ and Pi4

This is an exhaustive redaction density check for the camera resolution, showing the extremes at both ends of redactions - very fine-grained and slow, or too big and very quick. The redaction density must be a factor of the resolution so that the pixel groups can stack adjacently with no pixels left at the edge of the image.

Using a 1px redaction density is unreasonable because whilst initially it appears that there would just be a 4.25-second delay (on the Pi3B+) before receiving footage due to the Merkle tree generation, the bigger problem is the memory limitations; at 30fps, there is 4.25 seconds or approximately 128 frames being Merkle-hashed at once, requiring huge amounts of memory allocation (relative to the memory of the Pis). Therefore, using a larger density meant that fewer frames were processed simultaneously, and the camera wouldn't crash. Memory constraints were one of the most important constraints as the camera nodes are microcontrollers.

The large memory allocation comes from the number of hashes that happen; for a 1px redaction, with the current resolution, $1,843,204$ hash operations were executed. With frames being processed concurrently, the allocation built up too quickly to process, causing allocation errors and the camera nodes to freeze.

While it would be optimal to have the Merkle-hashing done in under 1/30 of a second, such that only 1 frame is processed at a time, this would require a redaction density $d \geq 32$ on the Pi4, which is fine, but on the Pi3B+ is was unreachable, even with 160px redaction density. A compromise between redaction density and performance had to be made, as it would be preferable for the home party if the redaction density was low, so minimal redactions are made, whereas this would cause the camera nodes to crash with memory allocation errors. A 32px redaction was chosen for testing and deployment.

5.8 Multithreading

All 3 of the nodes use threads to handle requests and command processing in parallel. This led to a number of issues around data races, inconsistent data reading and deadlocks. In order to combat this, we made use of mutexes for two key object types: locking on vectors that needed reading/writing from/to, and locking on sockets that needed exclusive access to read/write from/to.

Any functions that use loops and mutexes have small thread-sleep function calls at the end of the loop, to prevent tight loops from constantly locking the mutexes and causing deadlocks.

Another issue with the threading was the GUI library Wt. Fortunately, there is a method on the application instance that allows us to use threading; `application->attachThread();`. Combined with mutexes, the front-end and back-end execute in parallel.

5.9 Libraries

5.9.1 Internal Libraries

The GUI code is written as a separate library that is linked to the main codebase. This allows individual development and testing of the library separate from the back-end code. The Wt and OpenCV (and Unix-specific X11) libraries are linked to the front-end library.

5.9.2 External Libraries

OpenSSL was chosen as the cryptography library, as it has support for all the different cryptographic primitives that are required for the cryptography model, and had simple integration. We used the envelope family of functions for all the asymmetric operations (key generation, singing, verifying, encrypting and decrypting, and hashing). This is a flexible high-level class of functions for variety of cryptographic operations.

OpenCV is used for all image and video-related code. It is used to capture webcam footage, encode frames into jpeg, and reconstruct videos from frames. The front-end code also uses OpenCV, to display the images on the image widget. OpenCV was chosen because it has a wide variety of functionality, allowing us to expand other image-related areas of the project without having to import more library code.

Wt is a web-based GUI framework, similar in structure to Qt, but for web applications. We use Wt to display an interface to all camera streams that the foreign server is receiving, with tool buttons for redacting parts of the video. The Wt library is used only in the GUI code, which is linked only to the foreign server application.

X11 is a library used only by Unix builds, as it provides support for the X Window System, which is the foundational software system used by the majority of Unix systems to manage graphical displays.

5.10 Cross Platform

Development of the program was done mainly on Windows, with Linux support being added features as each update was subsequently tested on Linux. Both a Linux machine and WSL2 (with cross-compilation) were used to test Linux builds. Towards the end of the project, development switched to Linux-driven, as the target platform was confirmed to be a Linux-oriented machine (Raspberry Pi running 64-bit Raspberry Pi OS).

We used conditional compilation in C++ for code that had to be different on different platforms. This was for the OpenSSL configuration file loader, the webcam initialisation, and the socket code. Following the conditional compilation, the CMakeLists.txt was updated for platform-specific libraries such as X11. Whilst Windows builds are now unsupported, the conditional compilation has been left in so that Windows supported can be re-added without too many changes in the future.

Chapter 6

Testing and Deployment

6.1 CI/CD Pipeline

At the beginning of the development stage, we set up the Raspberry Pis to function as the camera nodes in the university labs. In order to target these devices for testing purposes, we needed to employ a remote deployment method that would allow what we had developed on our local machine to run on the Pis. As we were already using GitLab for source control, we decided to utilise GitLab's CI/CD tools that could automate build, test, and deploy scripts on push and merge events. This pipeline was configured in a YAML file and required self-managed runners for the university's self-hosted instance of GitLab. Another Raspberry Pi was set up in the labs to function as this build server. This also enabled us to run instances of the home and foreign server on the same local network as the Pis. This way, the cameras and foreign server could be tested in the same setup as a production environment.

6.2 Building and Testing

The build scripts were configured to run in Docker containers that were based on Debian Bookworm. This is the same as what the target OS is based on, Raspberry Pi OS, preventing issues arising from cross-platform compilation. The first stage of the pipeline was to build the Docker images, using a declarative list of dependencies defined in a Dockerfile. These dependencies include build tools, such as CMake, and open-source libraries such as OpenSSL and OpenCV that are needed to build our executables and libraries. Running each build job in an isolated Docker container ensured that the same build was produced each time and not affected by the external state of the system. The build results produced by each job were uploaded as artifacts to GitLab and shared with any other jobs that used them as dependencies.

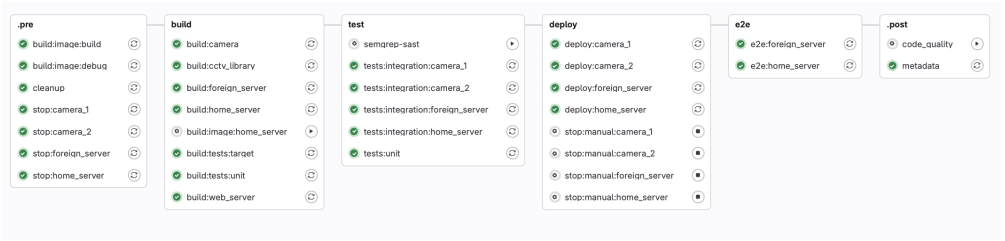


FIGURE 6.1: Build, Test and Deploy Pipeline

After the build stage is completed, the resulting test executables are run. We structured our tests into unit tests, integration tests and end-to-end tests. Each of these were configured with the Google Test library for C++ which produced a test report visible on GitLab. The following test definitions were decided upon and implemented:

- **Unit Tests** - These are run on push to any branch and merge to main. They test the smallest individual parts of the code, using mocking where necessary.
- **Integration Tests** - These are run on merge to main. They test that the hardware working as expected and components are working together. For example, there is a test to check that the hardware security modules on the Pis are all functioning correctly.
- **End-to-end Tests** - These are run on merge to main, after the system has been deployed. They test the workflow of the system as a whole from the perspective of users of the foreign and home servers.

We followed a test-driven development approach by writing some initial tests to confirm that they failed. Regression testing by confirming that previously passed tests continued to pass before the code was merged into the main branch. Additional tests such as code coverage, code quality, and static application security testing were configured to run, with their respective reports visible in the merge requests.

6.3 Remote Deployment

Whilst some open-source libraries, such as Wt, were statically linked into our executables, they also relied on shared libraries. To ensure that the targets we were deploying to had the necessary dependencies to run the programs, a Debian package was created after building them. This was integrated into the CMake configuration using CPack. Shared library dependencies were automatically detected and added to the package metadata using the `GENERATE_SHLIBDEPS` option. The apt package manager could then install these dependencies at the same time as installing the package. Due to the possibility

of multiple pipelines running at the same time, the deploy jobs were configured to prevent simultaneous deploys of the same program from happening on the same system, using the `resource_groups` option. Programs could also be stopped or re-deployed on GitLab's environments view. We installed Cloudflare Tunnel on the server in order to expose the program's web server to the internet and view it remotely. For security reasons, authorisation access was configured to only allow group members and partners to view it.

Through the installation of a package, the executables could be set up as a `systemd` service by copying a `.service` file into the `/lib/systemd/system` directory. As a service, the programs were configured to automatically restart if they crashed. This minimises the need for manual intervention if an error occurs on the cameras during operation. Furthermore, the reliability of the system was improved by installing a watchdog to reboot it if was under a very high load or the system stopped responding to a hardware timer. As we developed the cameras to be connected wirelessly, another script was set up to act if no traffic was detected on the wireless interface. As the cameras were designed to store backup footage in the case of a lost connection, the script attempts to restart the network manager instead of rebooting, so that the camera program can continue to record. Both of these recovery features were tested independently to confirm they were working as expected. For example, if the network was manually disconnected, it would be automatically reconnected again within one minute.

Chapter 7

Results

Overall, we achieved many of the goals we set out, including some stretch goals.

Capturing footage

Footage could be captured from the device's webcam, using the OpenCV library. We successfully verified the footage being recorded was correct by visual inspection (purely on the camera node) before integrating this into the video frame requests being sent to the home server. We ensured a steady frame rate by pushing all image processing into a separate thread per frame, and then adding a fixed time thread-sleep into the image capture loop.

TLS sockets

TLS sockets are successfully used, by using the OpenSSL socket primitives inside the normal socket primitive. These were wrapped into a single class and linked to our cryptography code, for certificates and keys. Using TLS sockets ensured point-to-point confidentiality and authenticity.

Server and camera discovery

Multi-casting was successfully deployed by the foreign server, to advertise itself to camera nodes over the network. This meant that it didn't require a static internal IP address. If the server does have a static IP address, certain arguments can be passed into the command line execution, to bypass the multicast system. Cameras are able to successfully connect to the foreign server, over TLS.

Heartbeat and state of health

Heartbeat messages successfully reached the home server from the camera nodes. Request serialization and parsing were tested here too, and was determined to all be working as expected. The signature was verifiable, and all information was received in the same

way it was sent. Currently, the stable branch has a dummy flag byte, but another development branch has the Pi status flags linking into this heartbeat message, for things such as power and network information.

Signing footage

The RSA code to sign and verify footage was tested in the handshake, heartbeat, and video frame data being sent. After being used for the handshake initially, and then deliberately broken with modified data, we verified the signing and verifying was performing as expected. This formed the foundation of our authenticity model.

Sending footage and signature, verifying signature, receiving footage

Once the signing was ready, we introduced proper video frames into the video requests, as opposed to dummy data, and signed the entire frame. This was tested by saving still frames that had been received. Once these were successfully appearing in the filesystem and uncorrupted, it was determined they were ready for video reconstruction. The still images were converted into jpgs, exclusively for testing.

Securing OS

One of the advantages of the Raspberry Pi 4 is that it provides a secure boot implementation. This allows a boot image to be signed so that only trusted programs can be loaded at boot. There a number of quick configurations that can be made to the Raspberry Pi OS to lock it down and greatly decrease the attack vector; for example, only accept certain traffic with a firewall configured through `iptables`, disable the ability for peripherals to be connected, turn off shell access, remove executable permissions and lock all user accounts. Additionally, the hardware security module we have chosen (Zymkey 4) has a production mode to permanently bind to the specific Raspberry Pi hardware by measuring certain attributes, such as the SD card. As a result, if the Pi or SD card were to be swapped out for a different one then the Zymkey would not recognize it and its secure storage component containing a private cryptographic key could not be read from. Each of these safeguards can be used together to follow a defense-in-depth strategy, meaning that the attacker can not gain control of the system even if they breach some of the security layers in place. Although we designed and developed the programs with these security measures in mind, we did not implement them on the devices during development, as doing so would greatly impede our ability to deploy and test on the devices. It is recommended to only take these actions when the system is ready for production as many of them are irreversible, as appropriate, protecting the system in cases of remote and physical access.

Redacting footage

We successfully created a front-end web server for the redaction to be performed manually. This worked as expected and was deployed on the foreign server. The back-end

was separately tested with hardcoded redactions before combining with the front-end, with the blacked-out pixels deliberately breaking the signature verification on the home server; this test was to ensure the image didn't get corrupted following pixel edits.

The back-end redactable signature scheme was tested separately from the implementation with dummy images. The scheme underwent a few iterations, for performance reasons, due to the large number of hashed required. Once we could verify root hashes with redactions taking place, and verifying all redactions (changes to the image) were black pixels, we inserted the root hash generation into the image processing steps. The integration of the redactable signature scheme completed the back-end redaction implementation.

Following the successful individual testing of the front- and back-end redaction logic, the two were combined successfully, by sharing pointers and callbacks between the Wt front-end instance and the foreign server instance. This allowed the front-end to directly control the redactions that appeared on the home server. However, upon testing at a high fps, we found issues with the amount of parallel allocation. This was mitigated with an increased redaction density. This completed the successful implementation of redacting footage.

Uninterruptible Power Supply

A UPS was designed and implemented with great success. In the event of power loss, the cameras seamlessly switch to a battery backup without any interruption to their normal operation. The cameras can continue to operate in these conditions for a minimum of two hours, more than double what was initially considered. The batteries trickle charge at 250mA whenever there is a power source available. This low charge current prevents any disruption to the Raspberry Pis that may be caused by putting too much demand on the USB charger being used as a power supply.

Tamper Detection

Three different methods of detecting tampering were implemented with varying success. Firstly, failure in the main power supply worked flawlessly. Whenever the supply voltage was removed and the battery backup was required, this would be reported in the heartbeat message. Secondly, perimeter detection via an electronic tamper seal functioned as intended but could use improvement in some regards. There was some instability in this feature on the second camera due to slight warping in the backplate of the 3D printed enclosure. This could lead to false positives as there wasn't always a firm connection between the copper contacts. Due to the change in Raspberry Pi used at a late stage of the project, it was necessary to add better ventilation to the enclosure. This created a point of access that could be used to bypass the perimeter detection. Finally, tap detection using the accelerometer of the HSM was functional, but far from perfect.

It was found that the method implemented to interpret accelerometer data never produced false positives (precision = 1) but would often miss actual tamper events (recall = 0.4). It is clear that this aspect of tamper detection requires further development. This could potentially use machine learning on a much larger set of collected data to form a significantly more accurate detection criteria. Overall, tamper detection was successfully implemented to protect the cameras but could require further development to be improved in the future.

7.1 Stretch goals not met

Backup storage

There was a feature theorized where if the foreign server couldn't connect to the home server, or it lost connection, then it would begin saving footage onto its own storage device. Because the home server would know there wasn't a connection, it would take this into account when checking the timestamps' tolerance.

If the cameras couldn't connect to the foreign server, or lost connection, then the cameras would begin storing footage on their SD card and then would release it all to the foreign server when the connection was available again. This was designed before the streaming protocol of video frames was introduced, so it would need more refining to work with redaction because the stream would be permanently delayed. A possible option would be to enable a fast forwards button if the footage had been queued on the camera. This would allow the foreign server to "catch up" to the live stream again.

HSM for key storage

The Zymkey HSM we have on the Pis has been used for the accelerometer, but the onboard crypto utilities were never utilized. This was mainly down to time constraints as a large amount of the cryptography library would need to be changed, and also we would need to switch from RSA to ECC to perform operations on the HSM. his would lose the asymmetric encryption property, leading to a partial refactor of the handshake protocol. We would also have liked to use the file-locking mechanism on the Zymkey to store keys or derived secrets more securely.

Encrypted home server storage

The home server stored the video chunks as raw .avi files. In the future, we would have liked to encrypt these files under a key that users with certain permissions had access to. There were a few ideas about this, such as a key-file being given read permissions to users in a certain admin group, or using a secret sharing scheme such as Shamir's, to allow n keys from many to be used. Using Shamir's secret sharing scheme would have

opened up interesting properties to the security such as requiring 3 keys from 10 admins to be used at once to see footage etc, enhancing the security.

Chapter 8

Conclusions

This project successfully designed and implemented a secure CCTV system tailored for treaty verification in a mutually distrustful environment. By addressing the root challenges of tamper detection, data integrity, and system transparency, the system achieved its primary objectives of providing secure footage transmission and ensuring data verifiability. The prototype demonstrated a combination of hardware and software solutions, including cryptographic methods, a secure enclosure design, and mechanisms to detect and report tampering events.

Project management played a key role in maintaining progress and adaptability. The use of agile methodologies, fixed roles, and weekly meetings with the external partner ensured efficient problem-solving and alignment with changing requirements. Despite limitations with time and budget, the team produced a system that exceeded the technical specifications and provided a solid foundation for future work on the project.

Looking forward, the project offers significant possibilities for continued research. Future improvements could include enhanced tamper detection through machine learning, increased system scalability, and integration of advanced technologies such as blockchain for transparent logging. In addition, improving the user interface and further optimising the redaction features could enhance the usability and appeal of the system.

In conclusion, this project not only met its technical goals, but also highlighted the importance of trust, transparency, and security in systems designed for high-stakes applications. The lessons learned and the solutions developed will be a valuable contribution to the ongoing research and development of treaty verification technologies.

Appendix A

Appendix

A.1 Parts

GDP Project 10 Budget

Part Name	Order code	Price per unit	Price (inc. VAT)	Quantity
Order 1				
Pi Zero 2 W	SC19061	£12.40	£14.88	4
256 GB MicroSD	MD01478	£16.48	£19.78	2
RPi Power Supply	SC19062	£6.30	£7.56	2
Pi Wide Angle Camera Module	4132320	£27.56	£33.07	2
Delivery		£0.00	£0.00	
Order 2				
Zymkey	2947716	£33.85	£40.62	2
Pi 4	3051887	£42.31	£50.77	1
Pi Heatsink	2319947	£2.02	£2.42	2
Battery	2401853	£25.36	£30.43	2
Schotkey diode	7278403	£0.08	£0.10	5
P MOSFET	2061526	£0.37	£0.45	5
SOT-23 Adapter	3549297	£1.92	£2.30	5
Delivery		£0.00	£0.00	

Total		VAT	Budget
		1.2	£400
£59.52			
£39.55			
£15.12			
£66.14			
£0.00			
Order 1 Subtotal:	£180.34		
£81.24			
£50.77			
£4.85			
£60.86			
£0.49			
£2.24			
£11.52			
£0.00			
Order 2 Subtotal:	£211.98		
Grand Total:	£392.32		

[h]

A.2 Gantt Charts

FIGURE A.1: Original Gantt Chart

A.3 Meeting Minutes

A.3.1 Supervisor Meeting 18th October 2024

- Discussed Part order
  - Waiting on financial approval
- Progress
  - TCP working with multiple sockets for heartbeat and video

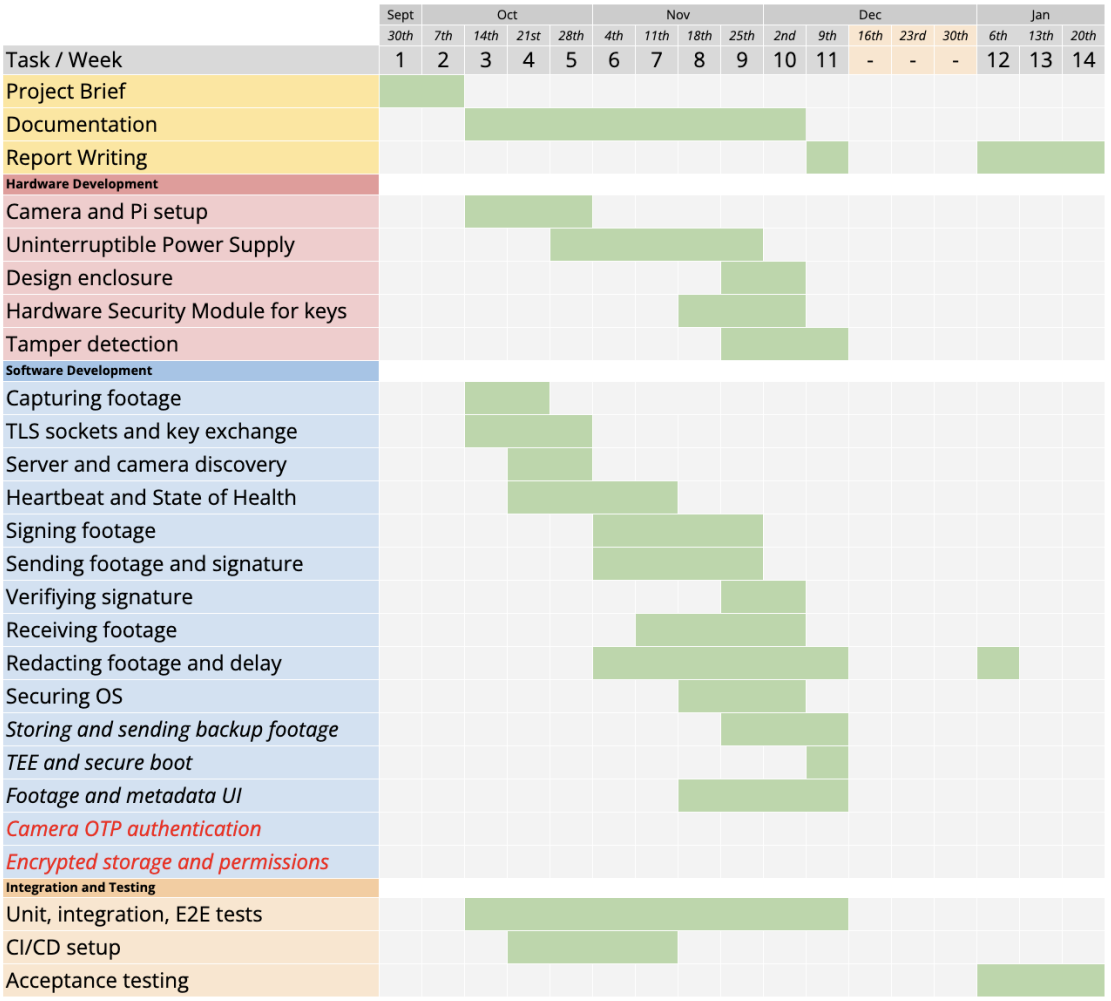


FIGURE A.2: Actual Gantt Chart

    - Simulating server-client using multiple threads

- To do next:
    - Host receiving and vetoing footage
        - Including actual video from Pi in communication
- Discussed heartbeat
    - Why is it detouring host server?
        - Can block packets
        - Signing and timestamping to stop replay attacks
- Progress on camera
    - It works with inbuilt Pi apps
        - Tested with RaspiCam

            - Provides unsigned char* buffer from camera using Raspicam.grab()
        - Issues with git
    - Look into GitHub student hub
- Reviewed progress against Gantt chart
- NEED TO ASSIGN MANAGEMENT ROLES ASAP
    - Project manager - ?
    - Communications - Connor
    - Budget - Alex
    - We will be marked on this
    - Will make things easier
- ARRANGE MEETING WITH AWE
    - 2pm Monday?
    - Weekly online
    - In person once per month
    - Plan agenda in advance
        - Manager?

A.3.2 Client Meeting 24th October 2024

- Updated client on current project progress
- Cameras
    - Using Pi camera module 3 (wide)
        - 12MP
        - Wide angle lens
- Backup specs
    - 1hr footage and battery
- Budget
    - Could design be improved with a higher budget?
        - Potentially more powerful microcontroller for camera but shouldn't be necessary
        - Custom ICs and SoC but out of scope for this project
- Heartbeat

  - Best to go through foreign server for security and trust

- Documentation
  - Improve README
    * Need to keep up-to-date documentation on GitLab
    * Need clear understanding of how the project is built
      · For clarity and production
      · Measure of success

- Need to come up with a way for the foreign authority to trust the system
- Discussed documents shared in Teams

A.3.3 Supervisor Meeting 25th October 2024

- Add meeting minutes to Git
- Adding documentation to Git
- Next integrating camera with network
  - Attempting today to potentially demonstrate on Monday
- Write agenda and share in advance of client meeting
- Discussed week 7 presentation
  - Progress
    * Diagrams
    * Gantt chart
      · Compare actual progress to planned progress
      · Indicate individual contribution
  - 10 mins
- Weekly reports also to be stored on Git
  - Logbook usage
- Discussed timings for remaining project
- Discussed tamper-proofing mechanisms
  - Drilling
    * Opening case
    * Camera overwatch

- Lab working
  - All get lab access
    - Store in locker
    - Connect with static IP for remote access
    - Ask about having cameras running

A.3.4 Client Meeting 28th October 2024

- End-to-end testing on GitLab
  - Results shown on each merge to main
- UI approach
  - Reaching minimum requirements
  - Minimal streaming service setup
  - Website or individual?
  - Inspector side is not a major concern
  - Will be hosted to a local network
  - Can add extra authentication if required
  - Both web interface and application are options
  - Weigh up security concerns
  - Investigate which is more appropriate for the application
  - Web is easier to distribute and can include authentication still
  - Web toolkit library can potentially be used
  - Public, must be shared with anyone using the software
  - Neil & Neil will discuss feasibility of this and get back to us
- Authentication protocol for the cameras
  - Not yet finalized but will be on GitLab in a few days
- Will improve README to explain build instructions and usage
- Started integrating camera task and network task
  - Few bug fixes remaining but well on the way
- Neil Evans can't make the next 2 Mondays
  - Can work out a sensible alternative if required

A.3.5 Supervisor Meeting 1st November 2024

- Pre-shared key/certificate. What's the standard?
  - Derived secrets instead using seed and deterministic number generator
- Camera knows identifier but not master secret
  - Sends derived secret and identifier to authenticate
  - If derived secrets match, then the camera is authenticated
  - One derived secret per identifier
- Attacker models
  - Rishab doing?
  - Fin can help if needed
- Camera derived secrets allow the server to be independent meaning the server doesn't need to know about cameras for the system to work
- Advantages and disadvantages of cameras not knowing the master secret
- Can't cycle master secrets
- One camera compromise doesn't compromise all
- Master derived secret vs. certificates with our server as authority
- Tamarin Prover for our authentication method
  - Tamarin Prover
- Master secret on our home server
- Cameras run disconnect, reconnect to renew the public-private key pair
- Standard of how often the camera renews
- Cameras store data and heartbeat when disconnected to avoid data loss
- Setting up a certificate authority?
  - Is it viable?
  - Write up why not for the report if it's not

A.3.6 Supervisor Meeting 29th November 2024

- Looking at Turnigy 2200mAh 3S 25C Lipo Pack
  - More than we would need
  - Have 2 working prototypes
  - Designed around 3.7 volt, standard for Lithium ion cells
  - Run for an hour per initial specification
- Upgrading to Pi 4
  - 2 more would use the remaining budget
- Current order - just below budget
  - Pi 3
  - Pi 4
  - Battery
  - Hardware security module (Zymkey) - contains accelerometer
  - Can afford heat sinks with £9 left over
- Camera risk assessment sent off
  - Hopefully sorted today
- Adapting 3D model to fit different hardware
  - Leave space for leeway
- 3D printing
  - Doesn't take too long but there is a queue
  - Not available over weekend
- Use existing Pi 3 so can get required battery
- No meeting with AWE on Monday
  - Update them with box design and the new order
- Redacting footage and signatures working separately
  - Will join together
  - Send screenshot
- Report started on Overleaf
  - Add literature as separate chapter
- Group report due same time as individual report and poster per recent email (w/c 20th Jan)

A.3.7 Supervisor 8th November 2024

- Using camera in lab
  - SoP
    - Why are we collecting data?
    - How long are we storing it?
  - Permission required from university
  - Put it in a box
    - For privacy
    - Allows remote connection/constant running
- Presentation - Wed 16:00
  - Son unavailable on Wed
    - Dry run on Monday 14:00
    - Potentially embedded video feed
  - Focus on project overview
    - Initial Breakdown:
      - Overview and outline of project - Fin
      - Problems with existing solutions - Rishab
      - Hardware countermeasures - Alex
      - Networking - Sam + Ali
- Redactable signatures would be ideal
  - Still have the option of cutting feed?
  - Or just full redaction option?
- Dynamic frame buffers
- Working on reducing frame size
- Look into more powerful processing options for the camera

A.3.8 Client 18th November 2024

- Presentation
  - Standing on either side to face the audience
  - Feedback in the email
  - Presentation added to Teams channel

- Redactable signatures
  - Read through paper
  - A neat idea
  - Working implementation in Python
  - Computationally expensive
  - Don't have to do at pixel-level?
- Spending additional money
  - Considering upgrading Pis
  - Still to buy components for enclosure, circuits, UPS, and hardware security module
  - £220 remaining of £400 budget
  - Send an invoice to AWE if we need to spend more?
- Hardware security module
  - To securely store keys
  - Offload cryptographic operations
  - Zymkey can plug into Pi
    - Also has tamper detection, device binding, and real-time clock
- Looking at Gantt chart
  - TEE still a stretch goal
  - More for prototyping purposes on Pi?
  - Currently working on signatures, dynamic buffers, designing enclosure, and testing deployments
- Same time next week

A.3.9 Supervisor 22nd November 2024

- Gantt Chart update
  - Not working over Christmas
  - Follow up with Alex about:
    - Enclosure
    - Hardware Security Module
    - Tamper Detection
  - Up to Redacting footage on Software side

  - Redacting and Secure Boot are Priority
  - Basic UI is there and can be worked on if there's time
- Fin start on report ASAP
- Report due week commencing 6th of January
- Individual Report due two weeks later
- Testing needs about 10 pages in the report
- Acceptance testing should be a focus

A.3.10 Client Meeting 25th November 2024

- Report content
  - How secure is the system?
  - Any holes or weaknesses?
  - Data management aspects
  - Discuss, not necessarily provide answers
  - Not necessarily multiple docs, just the one is fine
- Possible presentation dry run in January at AWE
  - Before the actual presentation (24th January)
- Demonstration Methods
  - Any preferred demonstration methods?
    - Release on the GitLab with setup instructions?
    - Examples of tampering triggers
    - Example file and format
    - Example redactions
- Merge / Progress
  - Redaction implemented
    - Needs optimization
  - Merge is waiting on the final networking updates
  - More tests set up after merge
- Cameras need to be running in the lab for testing
  - Fin to write RA ASAP (Mon or Tue)

A.3.11 Supervisor 10th January 2025

- AWE presentation
  - 23rd Ideally
  - if not stay with 20th
- Evaluation metrics (for different pieces of hardware)
  - Redaction resolution vs frame time
  - Frame Rate
  - Frame resolution?
- Hardware testing
  - Accelerometer sensitivity
  - Structured testing
  - Realistic scenarios
  - Traffic
  - Drilling
  - Movement
  - Drop known weights for precise force?
  - Measure battery life?
- Security
  - Could utilise HSM more than we have
  - Secure boot
  - Key storage
  - Other options
  - Evaluation

A.4 Risk Assessment

Risk Assessment				
Task/Activity	GDP Group 10 Camera Testing		**Date**	27th Nov 2024
Unit/Faculty/Directorate	4th Year Group Design Project	**Assessor**	Finley Atherton	
Line Manager/Supervisor	Son Hoang	**Primary site/location**	Building 16 Lab (16.1003)	
Brief details/comments	We would like to test our cameras performance over extended periods by leaving it running in the lab for the 3 weeks over Christmas December 13th – January 6th.			

PART A

(1) Risk identification			(2) Risk assessment				(3) Risk management				
Hazard	**Potential Consequences**	**Who might be harmed** **(user; those nearby; those in the vicinity; members of the public)**	**Inherent**			**Control measures (use the risk hierarchy)**	**Residual**			**Further controls (use the risk hierarchy)**
			Likelihood	**Impact**	**Score**		**Likelihood**	**Impact**	**Score**	
The camera may inadvertently record identifiable individuals (e.g., lab staff or visitors) without their consent, breaching GDPR.	Possible legal and reputational consequences if personal data is mishandled.	Other lab users, staff, students etc.	3	3	9	Position the camera to avoid capturing people. Ensure it is focused solely on non-sensitive test environments preferably in a container. Disable audio recording unless necessary and justified under GDPR. Place clear signage in and around the camera, informing all individuals that the camera is operational and its purpose. Use encryption and secure storage for any recorded footage. Ensure the data is accessible only to authorised users. Delete all footage once it is no longer required for testing purposes (at latest 24th January, end of module).	1	3	3	

University of Southampton Health & Safety Risk Assessment

Version: ECS 1.0/2018

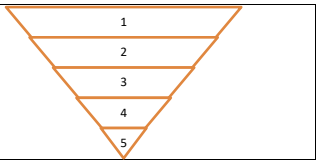
PART A												
(1) Risk identification			**(2) Risk assessment**				**(3) Risk management**					
Hazard	**Potential Consequences**	**Who might be harmed** **(user; those nearby; those in the vicinity; members of the public)**	**Inherent**			**Control measures (use the risk hierarchy)**	**Residual**			**Further controls (use the risk hierarchy)**
			Likelihood	**Impact**	**Score**		**Likelihood**	**Impact**	**Score**	
Increased fire risk due to camera coverage	Component damage due to overheating, fire	Lab users, staff, students etc.	2	4	8	Ensure containers used are flame retardant if possible. Ensure the container is well-ventilated to allow for cooling.	1	4	4	

University of Southampton Health & Safety Risk Assessment

Version: ECS 1.0/2018

Assessment Guidance

1. Eliminate	Remove the hazard wherever possible which negates the need for further controls	If this is not possible then explain why
2. Substitute	Replace the hazard with one less hazardous	If not possible then explain why
3. Physical controls	Examples: enclosure, fume cupboard, glove box	Likely to still require admin controls as well
4. Admin controls	Examples: training, supervision, signage	
5. Personal protection	Examples: respirators, safety specs, gloves	Last resort as it only protects the individual



Risk process

1. Identify the impact and likelihood using the tables above.
2. Identify the risk rating by multiplying the Impact by the likelihood using the coloured matrix.
3. If the risk is amber or red – identify control measures to reduce the risk to as low as is reasonably practicable.
4. If the residual risk is green, additional controls are not necessary.
5. If the residual risk is amber the activity can continue but you must identify and implement further controls to reduce the risk to as low as reasonably practicable.
6. If the residual risk is red do not continue with the activity until additional controls have been implemented and the risk is reduced.
7. Control measures should follow the risk hierarchy, where appropriate as per the pyramid above.
8. The cost of implementing control measures can be taken into account but should be proportional to the risk i.e. a control to reduce low risk may not need to be carried out if the cost is high but a control to manage high risk means that even at high cost the control would be necessary.

Impact		Health & Safety
1	Trivial - insignificant	Very minor injuries e.g. slight bruising
2	Minor	Injuries or illness e.g. small cut or abrasion which require basic first aid treatment even in self-administered.
3	Moderate	Injuries or illness e.g. strain or sprain requiring first aid or medical support.
4	Major	Injuries or illness e.g. broken bone requiring medical support >24 hours and time off work >4 weeks.
5	Severe – extremely significant	Fatality or multiple serious injuries or illness requiring hospital admission or significant time off work.

Likelihood	
1	Rare e.g. 1 in 100,000 chance or higher
2	Unlikely e.g. 1 in 10,000 chance or higher
3	Possible e.g. 1 in 1,000 chance or higher
4	Likely e.g. 1 in 100 chance or higher
5	Very Likely e.g. 1 in 10 chance or higher

A.5 Project Specification

School of Electronics and Computer Science

ELEC6200 MEng Group Design Project

GDP 10: Secure Data Management of a CCTV System

Rishabh Arora - ra5g21@soton.ac.uk
Finley Atherton - fa1g21@soton.ac.uk
Conor Fitch - cf2g21@soton.ac.uk
Samuel Gardner - sg5g21@soton.ac.uk
Alex Pakeman - adjp1g20@soton.ac.uk
Alistair Sirman - as9g21@soton.ac.uk

October 2024

Academic Supervisor: Son Hoang - t.s.hoang@soton.ac.uk

External Partner: AWE Nuclear Security Technologies

1 Problem Statement

The aim of this project is to develop a secure, authenticated and reliable CCTV transmission and storage system, allowing for the continuous monitoring and inspection of non-local facilities. It will feature a server-client model, an application-layer protocol over TCP, and small portable hardware for the camera. The key objectives of this project are:

- **CCTV System Development**: Building a functional CCTV camera on an embedded system that is capable of recording video footage, signing, performing encryption and transmitting the data.
- **Data Authenticity and Confidentiality**: Crucially, data authentication will be provided by signing the feed to assure the recipient it has not been manipulated. Encryption across the local network and internet will keep the data confidential and unable to be understood by anyone other than the sender and receiver.
- **Physical Security**: The system will include an uninterruptible power supply and storage to maintain surveillance in the case of power or connection being lost. The cameras will include tamper detection mechanisms and will communicate these in a 'state of health' report.

2 Project Plan

2.1 Technical Goals

- **Ensure authenticity of data**: Public key cryptography will be used to sign data immediately and directly from the camera feed, ensuring that data received is the data that was generated by the camera recording.
- **Tamper detection**: The cameras will include sensors to detect and report if they have had their enclosures opened or have been physically moved.
- **Status monitoring**: Regular heartbeat and 'state of health' messages will be sent even when the footage is not being sent. These will be monitored by the home server to verify conditions regarding the camera remain normal.
- **Camera overwatch**: Two cameras will be created and placed facing each other, either side of the monitored area. This ensures that any attempt to tamper with one camera will be recorded by the other.
- **Data encryption at rest**: Received data will be stored securely on the home server, encrypted by server-stored local keys. This ensures that the data remains secure from anyone without authorised access.
- **Data encryption in transit**: The foreign server and the home server will maintain a TLS connection, ensuring that all data exchanged will remain secure from any 3rd party trying to intercept it.
- **Footage delay and filtering**: Video transmission can be delayed by a certain time-frame to allow the sender to filter images they do not want to be sent.
- **Verify outgoing data**: Whilst the host cannot see the signing key, they do have access to the encryption key. Therefore, they can ensure the contents of the data leaving their network is what they expect.
- **Uninterruptible Power Supply**: This will ensure continuous monitoring through power outage, "brown out", or intentional disconnection of the power supply. The UPS should be able to provide no less than one hour of continued operation in the event of power disconnection.
- **Backup footage**: The camera is able to store footage in the event the server or CCTV network is down. The server can also retain footage when the connection between the two servers is down. In both these cases, the backlogged footage is sent upon restoration.
- **Scalability**: The network protocol will be flexible enough to "plug and play" new cameras in, where as long as the server is aware a new camera has been added, and its public key registered, it can be used immediately.

2.2 Stretch Objectives

- **Trusted Execution Environment**: Incorporating a TEE for key management, cryptographic operations, and secure boot to ensure the verified software is running and prevent signing keys from being exposed.
- **Permission system on data store**: The servers could alter the permissions in the folder of storage to only allow kernel and specific user access to the footage being stored. This could be connected to the file encryption being used.

- **User interface**: A user interface could be introduced that allows reviewing and analysing the footage that has been stored on the servers. This would integrate with accounts for levels of access.

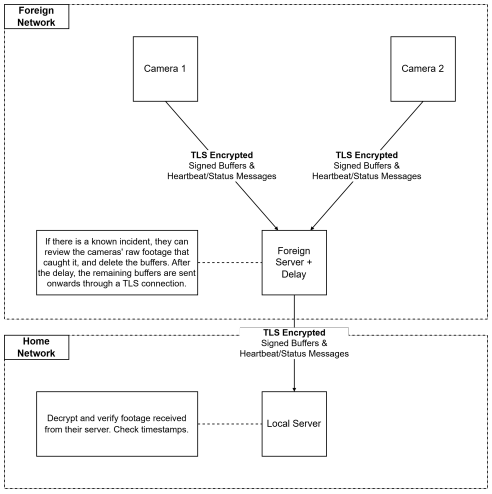


Figure 1: Network Diagram

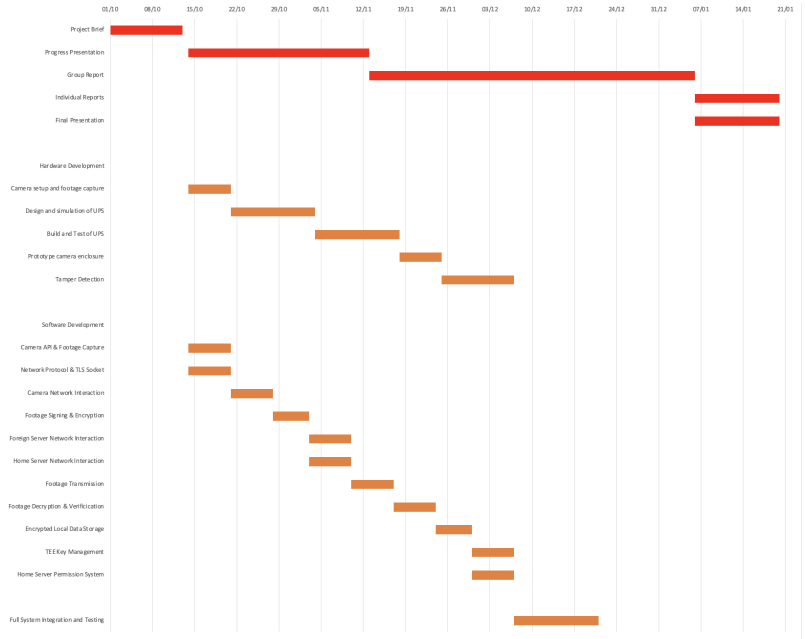


Figure 2: Gantt Chart

[h]

Bibliography

Principles behind the agile manifesto. Accessed: 16/01/2025.

Arulampalam Atputharajah and Tapan Kumar Saha. Power system blackouts - literature review. In *2009 International Conference on Industrial and Information Systems (ICIIS)*, pages 460–465, December 2009. ISSN: 2164-7011.

S.B. Bekiarov and A. Emadi. Uninterruptible power supplies: classification, operation, dynamics, and control. In *APEC. Seventeenth Annual IEEE Applied Power Electronics Conference and Exposition (Cat. No.02CH37335)*, volume 1, pages 597–604 vol.1, March 2002.

Yuqing Chen, Yuqiong Kang, Yun Zhao, Li Wang, Jilei Liu, Yanxi Li, Zheng Liang, Xiangming He, Xing Li, Naser Tavajohi, and Baohua Li. A review of lithium-ion battery safety concerns: The issues, strategies, and testing standards. *Journal of Energy Chemistry*, 59:83–99, August 2021. ISSN 2095-4956.

Roger G. Johnston. Tamper detection for safeguards and treaty monitoring: Fantasies, realities, and potentials. *The Nonproliferation Review*, 8(1):102–115, March 2001. ISSN 1073-6700, 1746-1766.

Mila Demchyk Savage. Blackouts and Brownouts or Power Outages. In *The Handbook of Homeland Security*. CRC Press, 2023. ISBN 978-1-315-14451-1. Num Pages: 5.

Yuan-Kang Wu, Shih Ming Chang, and Yi-Liang Hu. Literature Review of Power System Blackouts. *Energy Procedia*, 141:428–431, December 2017. ISSN 1876-6102.