# Security Project:  Detecting Modified Footage

Name: Joshua Klotzkin
BNumber: B00791509
Advisor Name: Leslie Lander, Yu Chen
Completion Date: November 24, 2025

Abstract: The project involves a camera connecting to a receiver that's resistant to replay attacks.

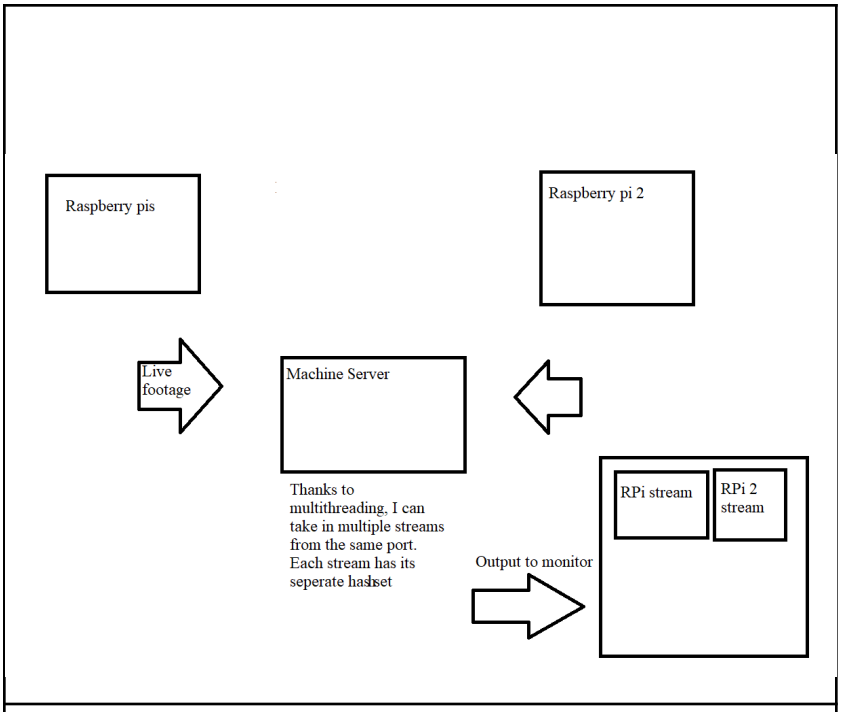



Figure 1: Block diagram of system, showing two clients, and a central monitoring computer. The clients can simulate replay attacks and the central monitoring computer will detect the attack and warn the user.

## Introduction:

The overall objective of this project was to develop a surveillance system that is resistant to replay attacks. A replay attack is when a video stream is interrupted and previously recorded video is inserted. This masks what is currently happening in the view of the camera. Such attacks can compromise security monitoring. This is used in the movie “Oceans Eleven.”

A previous goal of the project is to stream video footage from multiple clients onto a single machine, and detect replay attacks and repeated footage, so as to be a more accurate security system. In order to test the system, the clients must simulate replay attacks for the machine server to detect.

In the initial phase of this project, a method was developed to detect identical repeated footage by hashing each frame, and having the receiver keep a record of each hash. Multithreading was implemented to the stream so the hashing method of loop detection will only work if the looped footage is something the camera has seen before, which it won't be if the footage is mirrored.

The objective of the second phase is to detect modified footage that is not exactly identical, such as mirrored footage or other slight modifications. Replay attacks can be detected successfully, however the program also needs to be able to detect if the footage has been modified in any way such as the footage being mirrored. To accomplish that the clients encode the timecode in the image using steganography, which the receiver can detect and decode.

Steganography is a technique in which data is hidden inside other data, such as an image. In this case the least significant bit of the pixels of the image.

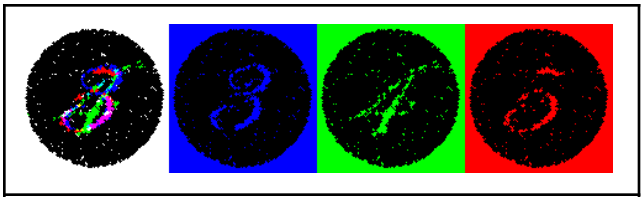


Figure 2: Example of Steganography, taken from wikipedia

The encoding is invisible to humans. If the receiver detects an inaccurate time code or fails to detect one at all will display an error message noting that the stream has been tampered with. This is a second security measure on top of hashing.

The clients also have a database of faces included to increase its attack capabilities. When a face from the program's database is detected by the sender, looped footage will be sent to the server.

## Project Methods:

This was run in python. The intention of this program is to eventually run it on a raspberry pi, so the program should really be low requirementsThe popular extension

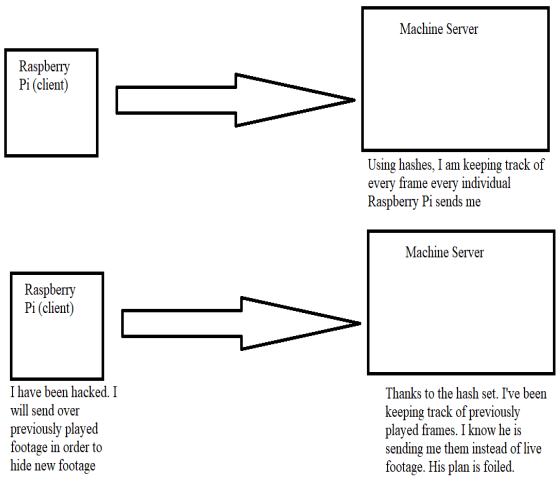



Figure 3: The system works as follows. As the client sends video, the server creates a hash of every video frame that will identify it hopefully uniquely. If the server detects ten frames with a hash already in the database in a row, it flags the footage as a loop.

OpenCV was used in order to recognize faces and stream footage from the client to the server.

For all examples shown below the server and client were run on the same machine.The user will run the server first, and the server will wait until a client is run and connected. Once the sender starts running the user is prompted to type in the ip address of the server, if none is entered it runs on localhost. They are also prompted if they want to add modifications to the pictures in the facial database, as more pictures allow the facial detection to work better.

Once they are connected the client will stream footage from the camera. The handle clients function allows for multithreading through the same port so it can handle multiple streams at once, by wrapping each stream in a file object for easier reading and streaming of incoming frames. The file also stores the hash sets of that individual camera.

The receiver is designed to prevent replay attacks by creating a hash set per thread and if it detects a certain amount of frames in a row with an identical hash set it will display a Loop Detected text on the screen. The hash set doesn't take up much data, as running it for 10 minutes only generated 1.3 kB of Data

In order to test this, the current clients can record footage with the space bar and stream the recorded footage to the server with the press of the L button. This will send the recorded footage instead of the current footage and should be flagged as a loop in the server. As a more advanced version of this there is an option to automatically loop recorded footage into the server when it detects a face in the database.

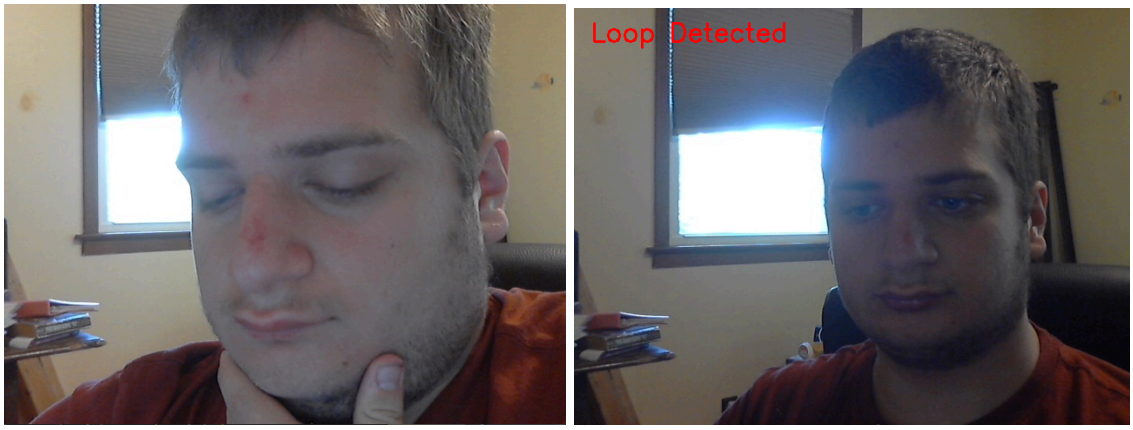
The next goal was to detect if the footage had been modified in any way and my idea was to hide the time code using a color coded string. This worked pretty well but wasn't hidden at all. Steganography, a process to hide the time code in the images Least Significant Bit was used instead.

The sender has been giving new code to encode the time the footage was recorded using the least significant bits of the streamed image using steganography. This is invisible if you are not a computer. The receiver can decode the steganography and see if it matches the current time.

Obviously, if the sender sends looped footage the time will be wildly off, and the receiver will give a warning if that's the case. However the sender has the option to mirror the sent footage horizontally. If the receiver is sent that footage it won't be able to read the time at all and will also give a warning.

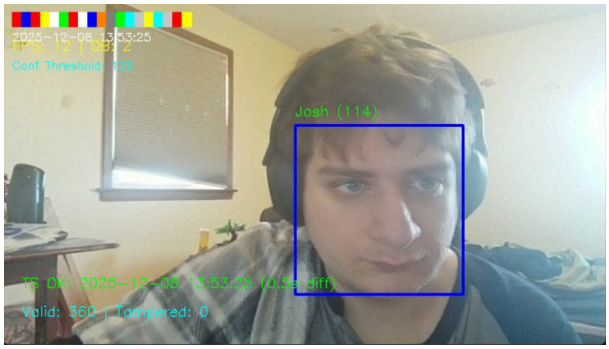
## Pictures and Figures:

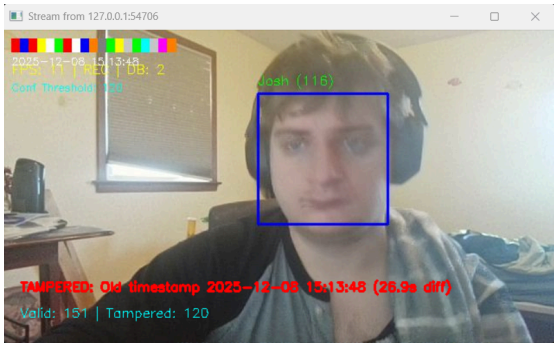
Below shows the results of just adding hashing into the receiver code. Normally it would have displayed nothing, as the picture on the left, however, when the receiver detects footage being played a second time it will display "Loop Detected" on the screen as it shows on the right.



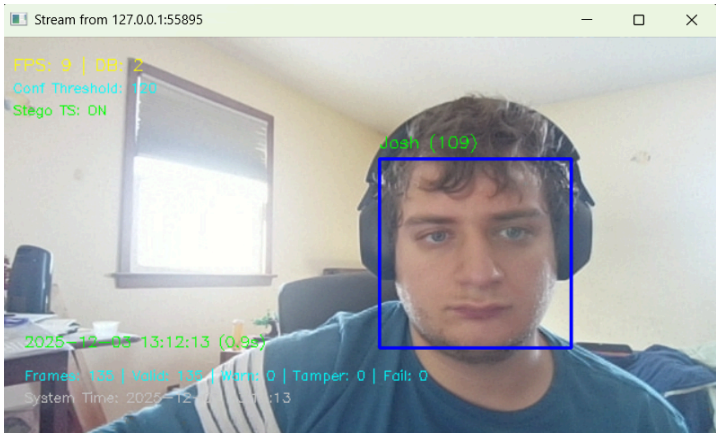

While the code in order to detect if the footage in a loop was mirrored or otherwise tampered with, my first idea was displaying a timecode in the top left corner of the screen encoded, using colors to represent characters of it. This required adding code to the receiver and the sender. The left picture shows everything being fine and the right picture shows a difference between the current time and the time being displayed to the receiver. Facial detection was added at this point in the project. A rectangle surrounds

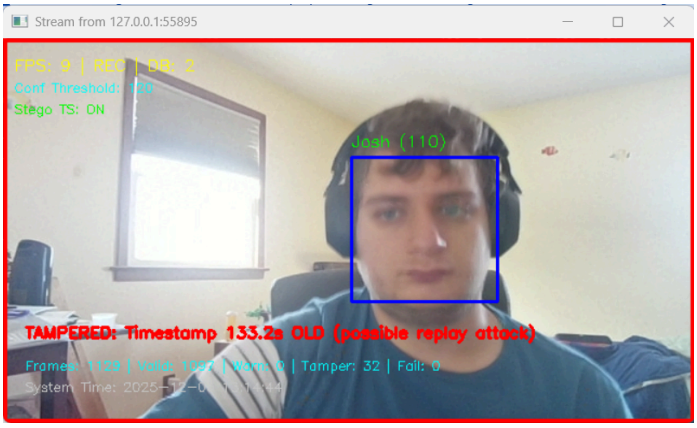
every face it detects in the frame and displays a confidence above it. If confidence is less than 120, it will recognize that person as someone from their database and apply their name above it as well.

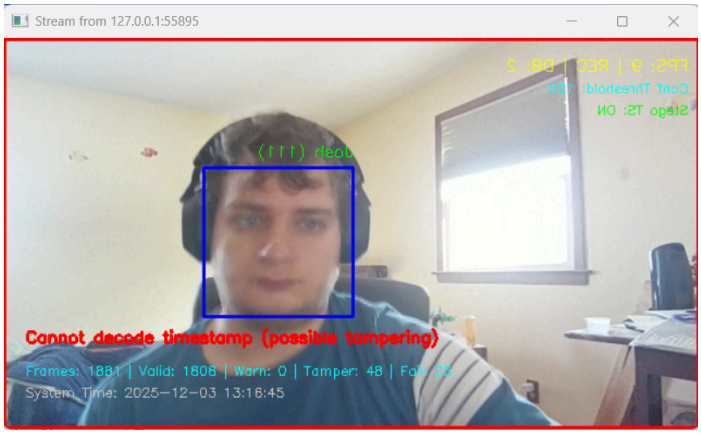



This approach was not subtle and have decided to look up a different way of implementing the timestamp in an image and came across steganography, which would hide the timestamp in the least significant bit and is virtually invisible to humans. The top image displays the receive getting normal ,untampered with footage, the second picture shows one receiving looped footage, noting the detected timestamp is about 2 minutes off from the actual time. The third image was mirrored and no timestamp could be detected, also triggering a warning from the receiver.






## Conclusion:

A security system was created by connecting a server to multiple clients. The server has other security measures to prevent anyone from tampering with the clients and the clients have ways of sending tampered footage to the server in order to test the functionality.

The server and client are connected via sockets as the user is prompted which IP to connect to. Multithreading has been added into the server in order to connect multiple clients at once. first implemented hashsharing on the server. For every client the server will keep a list of hashes of  frames it has seen before, and give a warning if it sees one. The hashing takes up a minimal amount of memory.

Facial detection was added to the client code. If the program is confident enough that the face it's seeing is one in its facial database it will automatically loop the footage into the server, if the option is selected.

Steganography was added in the final step of the process. The client will encode the current time code and the least significant bit of the frame being shown and the server will decode it and compare it with the current time. If it's incorrect or the time can't be shown, an error report will be triggered.

Code: https://github.com/JoshuaKlot/YOLOPersonDetection

## References/Resources:

https://www.geeksforgeeks.org/python/opencv-python-program-face-detection/

https://www.geeksforgeeks.org/computer-networks/image-steganography-in-cryptography/

https://www.geeksforgeeks.org/python/socket-programming-python/

Programs used:
Python
Visual Studio Code