

# A GUI-Based Tool for Automated Steganography Detection in Digital Images

## Members:

## Muhammad Huzaifa(22K-4641)

## Jaswant Lal(22K-4473)

## Usman Tanveer(22K-4478)

### Instructor: Sir Muhammad Khalid Khan

**Introduction**

Steganography, the practice of concealing data within digital files like images, poses significant risks to cybersecurity. Traditional detection methods are labor-intensive and lack scalability. StegGuard addresses this challenge by automating steganography detection through machine learning (ML) and providing an intuitive GUI for users with minimal technical expertise. This tool empowers cybersecurity professionals, digital forensics teams, and researchers to efficiently identify hidden content in images using a Random Forest classifier trained on color histogram features.

**Objectives**

* Develop a user-friendly GUI application for training ML models and detecting steganography.
* Achieve >85% accuracy in distinguishing clean images from steganographic ones.
* Enable real-time predictions and visualization of model performance.

**Technical Approach**

Data Pipeline

* Dataset: Kaggle’s StegoImagesDataset (clean vs. steganographic images).
* Feature Extraction:
  + - * 1. Extract RGB color histograms (256 bins per channel) using OpenCV.
        2. Generate feature vectors for model training.

Machine Learning Model

* Algorithm: Random Forest Classifier (Scikit-learn).
* Advantages: Robust to overfitting, handles high-dimensional data.
* Training: 70% dataset for training, 30% for testing.
* Evaluation Metrics: Accuracy, precision, recall, F1-score, confusion matrix.

GUI Development

* Framework: Tkinter (Python) for cross-platform compatibility.
* Core Features:

1. Model Training Interface:

* Upload directories of clean/steganographic images.
* Track training progress with a progress bar (TQDM).

1. Prediction Interface:

* Analyze single images for hidden content.
* Display results (e.g., “Steganography Detected: Yes/No”).

1. Visualization Dashboard:

* Plot accuracy curves, and confusion matrices (Matplotlib/Seaborn).
* Compare histograms of clean vs. altered images.

**Implementation Plan**

Phase 1: Setup & Data Preparation (1 Week)

* Collect and preprocess the Kaggle dataset.
* Develop scripts for histogram extraction and dataset splitting.

Phase 2: Model Development (2 Weeks)

* Train and validate the Random Forest classifier.
* Optimize hyperparameters (grid search).

Phase 3: GUI Integration (2 Weeks)

* Build Tkinter interfaces for training, prediction, and visualization.
* Link ML pipeline to GUI buttons (e.g., “Train Model”, “Predict”).

Phase 4: Testing & Deployment (1 Weeks)

* Conduct end-to-end testing (usability, accuracy).
* Deploy on GitHub with requirements.txt for dependencies.

**Conclusion**

StegGuard bridges the gap between advanced ML techniques and practical cybersecurity needs. Simplifying steganography detection, it equips users to combat covert data threats effectively. This project seeks collaboration and funding to achieve its vision of a secure digital ecosystem.