

ECE 371: Introduction to Security Engineering - Fall 2025  
**Lab 3: Secure Communication with RSA, DES, and Raspberry Pi  
GPIO**

**Due Date: 10/31/2025 11:59 PM**  
**Upload Submissions to Canvas**

### **Learning Goals**

By the end of this lab, students will be able to:

- Implement **RSA** for key generation, encryption, and decryption.
- Implement **DES** for symmetric encryption, including S-Box operations.
- Use **lgpio** for hardware feedback (LED and buzzer).
- Understand how **asymmetric + symmetric encryption** combine to form secure channels.

### **Introduction**

In this lab, you will build a small, but **secure communication system** using both **RSA** (asymmetric) and **DES**(symmetric) encryption on the Raspberry Pi 5.

You've seen in lecture how secure communication relies on two stages:

1. Using a **public-key algorithm** (like RSA) to share a secret key securely.
2. Using a **symmetric algorithm** (like DES) to encrypt actual data with that shared key.

Here, you'll replicate that process on the **Raspberry Pi 5**.

The client uses RSA to send an encrypted DES key to the server.

Then both sides use that key to encrypt or decrypt messages and images.

In this lab you will use GPIO hardware: the **LED** flashes when a message or image is received, and the **buzzer** sounds when one is sent.

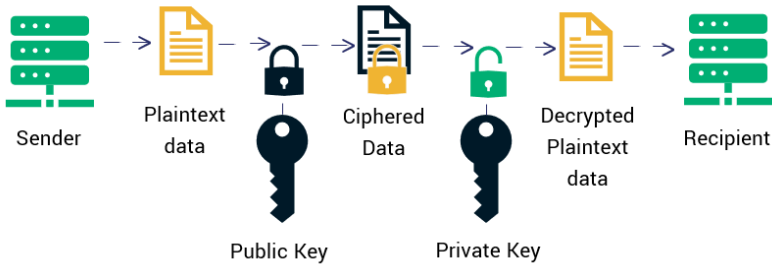
Before beginning, **review the lecture slides on Secure Communication Methods**

They summarize exactly how public and private keys are exchanged and how that enables secure data transfer.

## System Overview

### RSA:

#### How RSA Encryption Works



### DES:

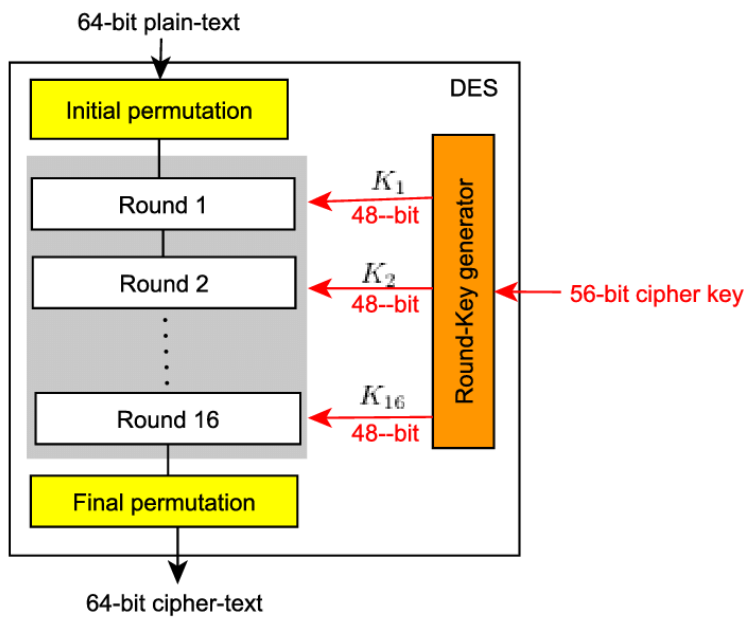


Figure 1. RSA/DES algorithm recap (check lecture slides for further details)

## Hardware Setup

### Required Components

- Raspberry Pi 5
- 1 × LED + 220  $\Omega$  resistor
- 1 × Active buzzer
- Jumper wires + breadboard
- **NOTE: DO NOT forget to ground your components**

## Software Installation

Run the following commands before starting:

```
sudo apt update
sudo apt upgrade -y
sudo apt install python3-igpio
```

### Note:

In previous labs you used **RPi.GPIO**.

This lab uses **igpio**, the supported library on Raspberry Pi 5.

You must edit the GPIO sections marked # TODO in the code to:

- Import igpio
- Assign your own BCM pin numbers
  - [https://vilros.com/pages/raspberry-pi-5-pinout?srsltid=AfmBOooosE6tT2OeL\\_nS8x6JA2NzUYmWLg4gZYd4pnAatIT6LMvXA9XA](https://vilros.com/pages/raspberry-pi-5-pinout?srsltid=AfmBOooosE6tT2OeL_nS8x6JA2NzUYmWLg4gZYd4pnAatIT6LMvXA9XA)
  - Refer to this webpage for GPIO pinouts
- Initialize the LED and buzzer

## Starter Files Provided

- RSA.py – RSA key generation, encryption, decryption
- des.py – DES algorithm implementation (with TODOs)
- chat\_client.py – Encrypts and sends text messages
- chat\_server.py – Receives and decrypts messages
- image\_client.py – Encrypts and sends an image file
- image\_server.py – Receives and decrypts the image

## Task 1 – Cryptographic Core (30 pts)

### Objective

Implement the cryptographic foundation: RSA and DES.

### Instructions

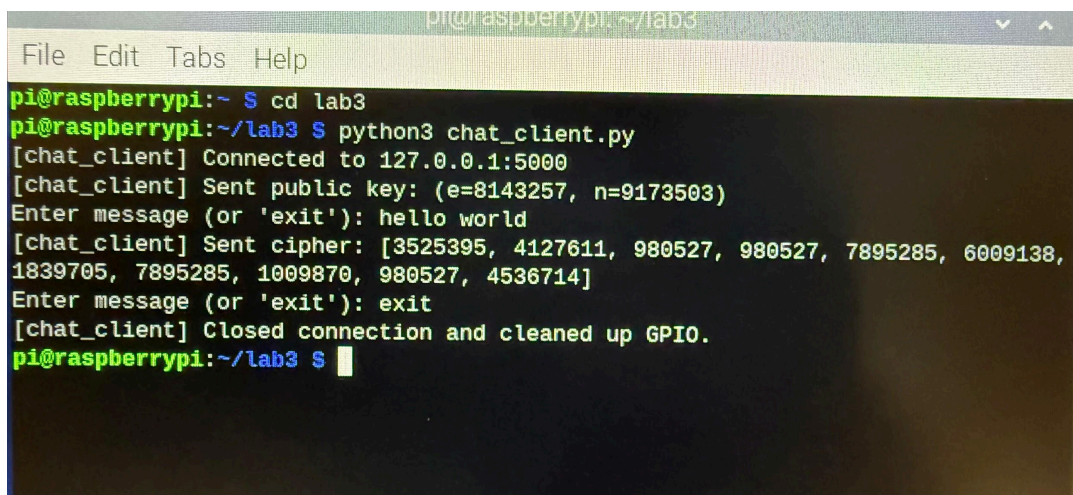
- Complete all missing functions in:
  - RSA.py

- `generate_keypair(p, q)` → Generate (e, n) and (d, n)
- `encrypt(key, plaintext)` → Encrypt text with RSA
- `decrypt(key, ciphertext)` → Decrypt text with RSA
- `des.py`
  - `string_to_bit_array` – Convert string → bits
  - `bit_array_to_string` – Convert bits → string
  - `xor` – Bitwise XOR
  - `generatekeys` – Complete DES key scheduling (16 round keys)
  - `compute_s_box` – Implement S-Box substitution logic

## Task 2 – Chat Communication System (20 pts)

Files: `chat_client.py`, `chat_server.py`

1. Implement GPIO sections (# TODO) to control the **buzzer** (client) and **LED** (server).
2. Encrypt and send messages using RSA.
3. Server flashes LED upon decryption; client buzzes after sending. (This will be tested in the demo)
4. Run the programs in separate terminals:
  - a. Run the server first: `python3 chat_server.py`
  - b. Once server is running, run client in a different terminal window:  
`python3 chat_client.py`



```

pi@raspberrypi: ~/lab3
File Edit Tabs Help
pi@raspberrypi:~$ cd lab3
pi@raspberrypi:~/lab3$ python3 chat_client.py
[chat_client] Connected to 127.0.0.1:5000
[chat_client] Sent public key: (e=8143257, n=9173503)
Enter message (or 'exit'): hello world
[chat_client] Sent cipher: [3525395, 4127611, 980527, 980527, 7895285, 6009138,
1839705, 7895285, 1009870, 980527, 4536714]
Enter message (or 'exit'): exit
[chat_client] Closed connection and cleaned up GPIO.
pi@raspberrypi:~/lab3$

```

```
pi@raspberrypi: ~/lab3
File Edit Tabs Help
pi@raspberrypi:~$ cd lab3
pi@raspberrypi:~/lab3$ python3 chat_server.py
[chat_server] Listening on 0.0.0.0:5000 ...
[chat_server] Connected by ('127.0.0.1', 38222)
[chat_server] Received client PUBLIC key: (8143257, 9173503)
[chat_server] Received cipher: [3525395, 4127611, 980527, 980527, 7895285, 6009138, 1839705, 7895285, 1009870, 980527, 4536714]
[chat_server] Decrypted plaintext: hello world
[chat_server] Client disconnected.
[chat_server] Closed sockets and cleaned up GPIO.
pi@raspberrypi:~/lab3$
```

### Task 3 – Image Encryption System (20 pts)

Files: `image_client.py`, `image_server.py`

1. Encrypt `penguin.jpg` using your DES implementation.
2. Encrypt the DES key with RSA and send it securely.
3. Server decrypts the key and then decrypts the image, the decrypted image should be saved as `"penguin_decrypted.jpg"`.
4. Client buzzes on send; server LED flashes on receive. (This will be tested in the demo)
5. Same as Chat Communication, run:
  - a. `python3 image_server.py`
  - b. `python3 image_client.py`

```
pi@raspberrypi: ~/lab3
File Edit Tabs Help
pi@raspberrypi:~/lab3$ python3 image_client.py
[image_client] Connected to 127.0.0.1:6000
[image_client] Sent public key: (2634159, 9173503)
[image_client] Sent encrypted DES key
[image_client] Sent encrypted image
[image_client] Closed connection and cleaned up GPIO.
pi@raspberrypi:~/lab3$
```

```
pi@raspberrypi: ~/lab3
File Edit Tabs Help
pi@raspberrypi:~/lab3 $ python3 image_server.py
[image_server] Listening on 0.0.0.0:6000 ...
[image_server] Connected by ('127.0.0.1', 48714)
[image_server] Received client PUBLIC key: (2634159, 9173503)
[image_server] Received encrypted DES key (8 vals)
[image_server] Decrypted DES key: '8bytekey'
[image_server] Received encrypted image (3584 values)
[image_server] Image decrypted and saved as penguin_decrypted.jpg
[image_server] Closed sockets and cleaned up GPIO.
pi@raspberrypi:~/lab3 $
```

## Verification

If you attempt to open the **encrypted image**, it will **NOT** open.

You may see an error such as:

*“Not a JPEG file”*

This is **expected**, it means your encryption worked.

Only the decrypted image should open successfully (it will be identical to the original penguin.jpg).

## Expected Behavior

Condition	Expected Output
Encrypted file opened	“Not a JPEG file” error – normal
Decrypted file opened	Displays original penguin image
Client send	Buzzer sounds



Server receive	LED flashes
----------------	-------------

### Answer Questions (10 pts)

Include short responses (2–3 sentences each) in your lab report:

1. What is the difference between RSA and DES encryption?
2. Why is DES used for bulk data instead of RSA?
3. How does padding ensure DES can handle arbitrary file sizes?
4. Why is CBC mode more secure than ECB mode?
5. Describe the hardware feedback (LED / buzzer) logic flow in your implementation.

### Deliverables & Grading (100 pts Total)

Component	Points	Description
<b>Code Implementation</b>	<b>60</b>	RSA (15) DES (20) Chat Client (5) Chat Server (5) Image Client (7.5) Image Server (7.5)
<b>Lab Report</b>	<b>10</b>	Written answers to Questions 1–5 + Screenshots of working files
<b>Live Demo</b>	<b>30</b>	Working chat and image demo with hardware feedback

### Submission Instructions

- Upload a .zip file containing all completed Python files, penguin\_decrypted.jpg, and your completed Lab Report (PDF) to Canvas.
- In your Lab Report, add screenshots showing **chat/image\_client/server.py** files running and executing correctly, along with answers to the written questions.

**Each task must be completed sequentially.**

**If you are working in Teams please make sure to add the names and SPIRE IDs of BOTH the team members in the Lab Report.**

**Start your Lab early. Do NOT try to rush everything the night before.**

**Good Luck!! 😊**