**Project Report: Bitcoin Address Generator**

**Abstract**

The Bitcoin Address Generator project is a Python-based application that allows users to generate secure Bitcoin addresses using Elliptic Curve Cryptography (ECC). This project demonstrates the core concepts of cryptographic key generation, public-private key derivation, and address encoding in Base58Check format. Through a graphical user interface (GUI) built using Tkinter, users can input their details and generate unique Bitcoin addresses securely and intuitively.

## ****Introduction****

Bitcoin is a decentralized digital currency that uses public-key cryptography for secure transactions. Each Bitcoin user has a pair of cryptographic keys: a public key and a private key. A Bitcoin address is derived from the public key and is used to receive funds.

This project explores how a Bitcoin address is generated from scratch, covering all essential cryptographic steps. The project integrates core cryptographic operations with a user-friendly GUI.

## ****Objectives****

* To understand and implement the principles of Elliptic Curve Cryptography (ECC).
* To generate Bitcoin public-private key pairs using the secp256k1 curve.
* To convert public keys into Bitcoin addresses using the Base58Check encoding scheme.
* To design an intuitive GUI using Tkinter.
* To ensure secure handling of private keys and user inputs.

**Tools and Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| Python | Main programming language |
| Tkinter | GUI development |
| hashlib | SHA-256 and RIPEMD-160 hashing |
| secrets | Secure random number generation |
| base58 | Base58Check encoding |
| tinyec | Elliptic Curve operations |
| PIL (Pillow) | Image display in GUI |

## ****System Architecture****

**Input:**

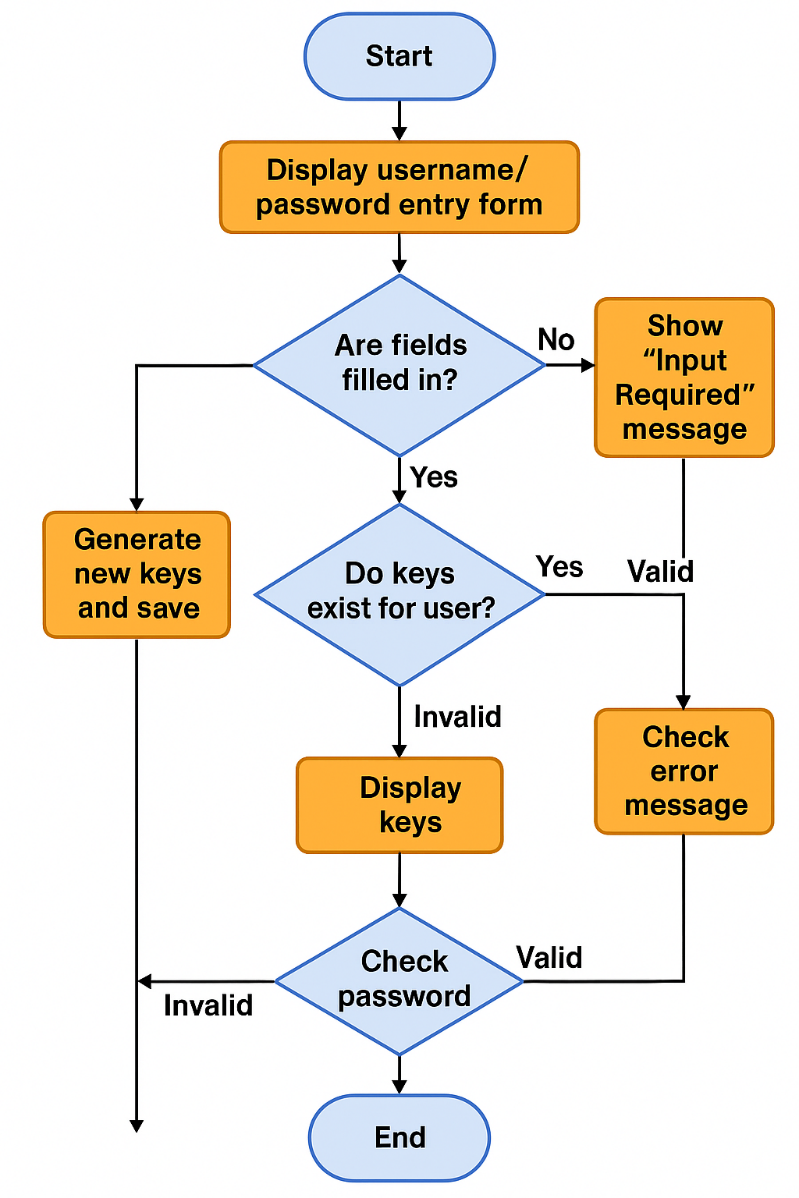
* Username (used to trigger address generation)
* Optional: Password (not stored for security)

**Process:**

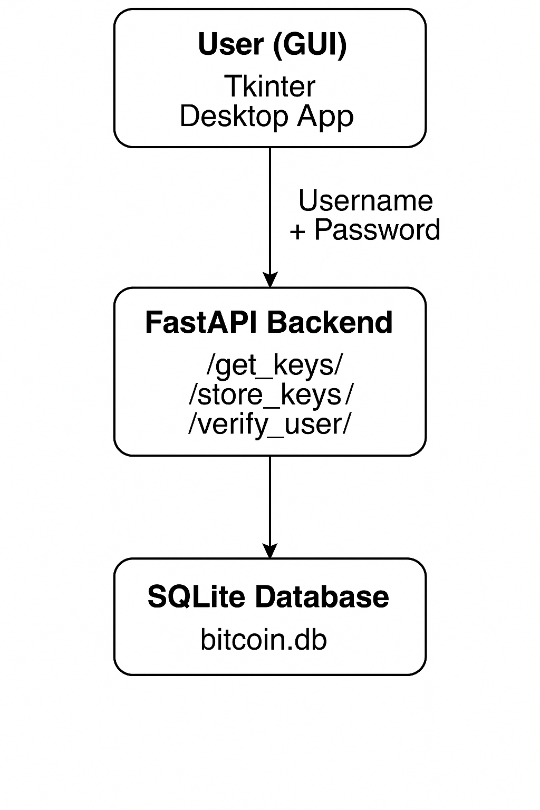
1. Generate a 256-bit random number.
2. Hash it using SHA-256 to get the private key.
3. Use ECC with the secp256k1 curve to generate a public key.
4. Hash the public key with SHA-256, then RIPEMD-160.
5. Add version byte and checksum.
6. Encode using Base58 to produce the Bitcoin address.

**Output:**

* Display private key
* Display public key
* Display Bitcoin address

****

**Flowchart**

****

## ****Implementation****

### **Core Logic (bitAddress.py)**

* Uses secrets to generate a secure 256-bit random number.
* Applies SHA-256 to generate the private key.
* Utilizes ECC (secp256k1) to compute the public key.
* Encodes the address using SHA-256 → RIPEMD-160 → Base58Check.

### **GUI (main.py)**

* Built using Tkinter.
* Accepts user input for the name and password.
* Displays private key, public key, and Bitcoin address.
* Includes a copy-to-clipboard feature for convenience.
* Displays relevant images for branding.

**Output Screenshots**

