Design Document for Backup and Restore Scripts

## 1. Overview

The backup and restore tool is a bash script-based solution designed to automate the process of backing up directories, encrypting them, and transferring the backup to a remote server for safekeeping. It also facilitates restoring backups from the remote server. The tool is split into two main parts: **backup.sh** and **restore.sh**, along with a shared library of functions in **backup\_restore\_lib.sh.**

## 2. Assumptions

1. **Environment Assumptions:**

- The scripts are intended to run in a Linux environment.

- The backup and restore scripts rely on the `tar` and `gpg` commands for creating and encrypting archives.

- The remote server is an EC2 instance on AWS with SSH access using a private key for secure file transfer.

2. **User Input Assumptions:**

- The user is prompted interactively for input parameters such as source directory, destination directory, encryption key, and the number of days to consider for modified files.

- All user-provided directories are assumed to be valid and accessible.

3. **Encryption Assumptions:**

- Files are encrypted using symmetric encryption with a passphrase provided by the user through GPG.

- The passphrase is not hardcoded but entered by the user at runtime.

4. **Backup Assumptions:**

- Only directories that have been modified within the specified number of days are included in the backup.

- The tool handles errors gracefully, including issues with invalid directories, permissions, or the absence of the encrypted backup file.

5. **Security Assumptions:**

- The encryption key is securely provided by the user and used only in the session. It is not stored in any permanent logs.

- SSH access to the remote server is secured using a private key.

## 3. Functional Flow

### 3.1 Backup Process

1. **User Input:**

The user is prompted to provide:

- Source directory to back up.

- Destination directory where the backup will be stored.

- Encryption key for securing the backup file.

- Number of days to consider for modified files.

2. **Validation:**

Each parameter provided by the user is validated:

- Source directory must exist.

- Destination directory must exist or be created.

- Encryption key cannot be empty.

- Days parameter must be a positive integer.

3.  **Backup Creation:**

- A timestamp is generated to uniquely identify the backup.

- Sub-directories within the source directory are checked for modified files within the specified number of days.

- Each modified directory is archived using `tar` and encrypted using `gpg`.

- A final backup archive containing all encrypted directories is created**.**

4. **Transfer to Remote Server:**

- The final encrypted backup file is transferred to the specified EC2 remote server using `scp` over SSH.

- The EC2 public IP is hard-coded in the script for simplicity, but it can be customized as per user requirements.

**Note:** The remote server used in this script is hosted on an EC2 instance, which has a dynamic public IP address. If you wish to test the script, please reach out to me, and I will provide you with the current IP address.

5.  **Completion:**

The backup process ends with a message confirming successful completion or failure.

### 3.2 Restore Process

1. **User Input:**

The user is prompted to provide:

- The encrypted backup file to restore from.

- The destination directory where the backup should be restored.

2. **Validation:**

Each parameter is validated:

- The encrypted backup file must exist.

- The destination directory must exist or be created.

3**. Restoration Process:**

- The encrypted backup file is decrypted using the user-provided encryption key.

- The contents are extracted using `tar` into the specified destination directory.

4. **Completion:**

The restoration process ends with a message confirming successful restoration or failure.

## 4. Design Decisions

1. **Modularization:**

- The core logic of both backup and restore functionalities is encapsulated in the  **backup\_restore\_lib.sh** file, which contains the following functions:

- **validate\_backup\_params**: Validates the backup-related parameters.

- **backup**: Performs the backup process.

- **validate\_restore\_params**: Validates the restore-related parameters.

- **restore**: Performs the restoration process.

2. **Parameter Validation:**

- Input parameters are validated during runtime to ensure that no invalid or empty values are used, minimizing errors during execution.

- The use of `sed` is implemented to replace spaces and colons in directory names with underscores, preventing issues when creating filenames or directory names.

3. **Encryption:**

- AES-256 encryption is used for secure backup storage.

- The encryption key is passed securely at runtime and is never hard-coded into the scripts to protect sensitive information.

4. **Remote Server Integration:**

- SCP (Secure Copy Protocol) is used to transfer encrypted backups to a remote EC2 server.

- The script ensures secure transfer of the backup to the specified remote directory.

**Note:** The remote server used in this script is hosted on an EC2 instance, which has a dynamic public IP address. If you wish to test the script, please reach out to me, and I will provide you with the current IP address.

## 6. Conclusion

This backup and restore tool is a simple yet powerful script-based solution for creating encrypted backups and restoring them from a remote server. By validating inputs, securing data with encryption, and integrating with a remote server, it provides a robust solution for users needing a secure and automated backup process.