Organigramme détaillé des étapes principales

1. Enregistrement des utilisateurs

- **But :** Permettre à chaque utilisateur de créer un compte sécurisé.
- **Tâches :**
    1. Création d'un **répertoire utilisateur** pour stocker ses données.
    2. Génération automatique d'un **couple de clés publique/privée** RSA :
        - **Clé publique** : utilisée pour le chiffrement et partagée sans chiffrement.
        - **Clé privée** : protégée et accessible uniquement via un mot de passe.
    3. Sauvegarde sécurisée de la clé publique dans un emplacement non chiffré.

2. Dérivation de la clé (Key Derivation Function - KDF)

- **But :** Transformer un mot de passe en une clé cryptographique robuste.
- **Tâches :**
    1. L'utilisateur entre un **mot de passe**.
    2. Transformation du mot de passe via un **algorithme de hashage sécurisé** :
        - Utilisation d'une fonction éponge avec plusieurs cycles (essorage) pour augmenter la complexité.
    3. Génération d'une **clé privée dérivée** suffisamment grande pour le chiffrement (e.g., 256 bits).

3. Authentification double sens

- **But :** Vérifier mutuellement l'identité du coffre-fort et de l'utilisateur.
- **Tâches :**
    1. Le coffre-fort envoie un **certificat** à l'utilisateur.
    2. L'utilisateur valide ce certificat auprès d'une **autorité de certification**.
    3. Implémentation d'une **preuve de connaissance à divulgation nulle (Zero-Knowledge Proof - ZKP)** :
        - Exemple avec **Schnorr** :
            - L'utilisateur prouve qu'il possède la clé privée sans la divulguer en répondant à un défi.
            - Vérification : $M = \alpha^{\text{preuve}} \times \text{pub}^{r}$.

4. Échange de clés

- **But :** Établir une clé de session partagée pour sécuriser les échanges entre client et serveur.
- **Tâches :**
    1. Utilisation du protocole **Diffie-Hellman** :
        - Les deux parties échangent des valeurs publiques.

        - Une clé de session commune est calculée localement de manière sécurisée.
    2. La clé de session est ensuite utilisée pour chiffrer les communications.

5. Développement des fonctionnalités principales

5.1 Stockage sécurisé des fichiers

- **But :** Protéger les fichiers sensibles de l'utilisateur.
- **Tâches :**
    1. Chiffrement asymétrique avec **RSA** :
        - Chaque fichier est chiffré avec la **clé publique** de l'utilisateur.
    2. Sauvegarde des fichiers chiffrés sur le serveur.

5.2 Consultation et déchiffrement des fichiers

- **But :** Permettre à l'utilisateur de récupérer ses fichiers.
- **Tâches :**
    1. L'utilisateur récupère les fichiers chiffrés depuis le serveur.
    2. Déchiffrement des fichiers avec la **clé privée** protégée par son mot de passe.

6. Chiffrement symétrique COBRA

- **But :** Implémenter un chiffrement rapide et sécurisé pour les échanges.
- **Tâches :**
    1. Basé sur l'algorithme **Serpent** avec des modifications :
        - **XOR** avec une clé d'itération.
        - **Substitution** avec 4 types de S-Boxes (utilisées par blocs selon l'itération).
        - **Fonction Feistel** personnalisée :
            - Transformation des blocs en 64 bits.
            - Utilisation de fonctions pseudo-aléatoires et mélanges spécifiques.
        - **Transformation linéaire** pour combiner les blocs.
    2. Réduction à 16 ou 12 itérations si le code est trop lent.

7. Fonctionnalités avancées (optionnelles)

- **But :** Ajouter des améliorations pour valoriser le projet.
- **Suggestions :**
    1. **Coffre-fort partagé** entre plusieurs utilisateurs.
    2. **Journalisation et audit** des actions (non-répudiation).
    3. Intégration d'un mécanisme de versionning pour les fichiers.

Résumé :

1. **Enregistrement des utilisateurs** → Répertoire + Clés RSA.
2. **Dérivation de la clé (KDF)** → Mot de passe → Hashage → Clé privée.
3. **Authentification double sens** → Certificat + ZKP.
4. **Échange de clés** → Diffie-Hellman → Clé de session.
5. **Stockage et consultation** → RSA pour fichiers.
6. **Chiffrement COBRA** → Serpent modifié.
7. **Options avancées** → Partage, journalisation, versionning.