

UNIVERSITÉ DE YAOUNDE I
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

DÉVELOPPEMENT D'UN OUTIL DE CHIFFREMENT/DÉCHIFFREMENT MULTI-UTILISATEUR

Cahier des Charges

Travaux Pratiques INF 4167 (Cryptographie)

NOM	MATRICULE
TAHUE TCHOUTCHOUA GEMAEL DIMITRI	25G2032
MAHACHU FONGANG AURELIE GRACIANE	22T2924

Yaoundé, 20 Novembre 2025

Table des matières

INTRODUCTION	3
CONTEXTE ET PROBLÉMATIQUE	4
Contexte général	4
Problématique	4
Enjeux du projet	4
PÉRIMÈTRE ET LIMITES DU PROJET	5
Périmètre inclus	5
Périmètre exclu (hors scope)	5
Contraintes	5
HISTORIQUE DU CHIFFREMENT	7
OBJECTIFS DU PROJET	9
1. Gestion multi-utilisateur	9
2. Chiffrement des données	9
3. Déchiffrement contrôlé et permissions d'accès	9
4. Gestion des permissions et partage sécurisé	10
5. Authentification robuste des utilisateurs	10
6. Journalisation et traçabilité des actions	10
7. Interface ergonomique (CLI ou GUI)	10
8. Architecture modulaire et sécurisée	11
SPÉCIFICATIONS FONCTIONNELLES	12
Acteurs du système	12
Cas d'utilisation	12
CU01 : Inscription et création de compte	12
CU02 : Authentification	12
CU03 : Chiffrement d'un fichier	13
CU04 : Déchiffrement d'un fichier	13
CU05 : Partage d'un fichier	13
CU06 : Révocation d'accès	14
EXIGENCES NON FONCTIONNELLES	15
Performance	15
Sécurité	15
Disponibilité et fiabilité	15
Maintenabilité	15
Conformité	15
PRINCIPE DE FONCTIONNEMENT	17
3.1 Génération et gestion des clés	17
3.2 Processus de chiffrement	17
3.3 Processus de déchiffrement	18
3.4 Vérification d'intégrité et signature	18
3.5 Partage sécurisé entre utilisateurs	18

3.6 Journalisation et audit	19
MÉTHODOLOGIE ET ARCHITECTURE DU PROJET	20
5.1 Méthodologie générale	20
1. Analyse des besoins et conception fonctionnelle	20
2. Conception technique et choix des algorithmes	20
3. Implémentation et validation	20
5.2 Architecture fonctionnelle	21
5.2.1 Composants principaux	21
5.2.2 Flux de données	21
5.3 Architecture technique	22
5.4 Sécurité et conformité	22
5.5 Avantages de cette architecture	23
ANALYSE DES RISQUES	24
Risques techniques	24
Risques organisationnels	24
Risques liés à la sécurité	24
PLANNING ET LIVRABLES	25
Phases du projet	25
Livrables	26
CRITÈRES DE VALIDATION ET RECETTE	27
Critères de validation fonctionnelle	27
Critères de validation technique	27
Procédure de recette	27
GLOSSAIRE	28
CONCLUSION	29
BIBLIOGRAPHIE ET RÉFÉRENCES	30

INTRODUCTION

Depuis que les sociétés humaines échangent de l'information, la nécessité de la protéger n'a jamais cessé d'être centrale. Que ce soit pour transmettre des ordres militaires, préserver des secrets diplomatiques ou protéger des données personnelles, le besoin d'assurer **la confidentialité, l'intégrité et l'authenticité** des messages constitue un enjeu fondamental du transport et du partage de l'information.

Dans l'Antiquité déjà, les civilisations recherchaient des moyens d'envoyer des messages compréhensibles uniquement par leurs destinataires. Les Grecs utilisaient la *scytale*, les Romains le chiffre de César ; autant de techniques rudimentaires mais révélatrices d'une préoccupation universelle : empêcher qu'un tiers non autorisé ne comprenne le contenu d'un message intercepté. Avec le temps, ces approches ont évolué pour faire face à des adversaires toujours plus compétents, surtout lorsque les guerres, la diplomatie et les réseaux de communication se sont complexifiés.

L'avènement de l'ère numérique a transformé cette problématique en un défi d'une ampleur sans précédent. Les informations ne circulent plus dans des lettres scellées ou des parchemins codés, mais transitent à travers des réseaux mondiaux où elles peuvent être copiées, analysées ou altérées en une fraction de seconde. Les données sensibles—dossiers médicaux, transactions bancaires, identités numériques, documents confidentiels—sont désormais partagées massivement, souvent entre plusieurs utilisateurs et via des plateformes connectées.

Dans ce contexte, la cryptographie moderne est devenue l'un des piliers essentiels de la cybersécurité, garantissant que seules les bonnes personnes puissent accéder à la bonne information, au bon moment.

C'est dans cette dynamique que s'inscrit le projet de **développement d'un outil de chiffrement/déchiffrement multi-utilisateur**. Il s'agit non seulement de permettre à différents utilisateurs de chiffrer et déchiffrer des données, mais également d'assurer une gestion rigoureuse des permissions, une authentification fiable et une traçabilité complète des actions effectuées.

La multiplication des utilisateurs et des niveaux d'accès complexifie considérablement les enjeux de sécurité, ce qui impose une conception méthodique, l'usage de mécanismes cryptographiques robustes, et une architecture pensée pour résister aux menaces modernes.

Ainsi, ce projet vise à proposer un outil à la fois pratique, sécurisé et adapté aux réalités actuelles de la gestion des données sensibles en environnement multi-utilisateur, tout en s'appuyant sur les fondements historiques et techniques qui ont façonné la cryptographie contemporaine.

CONTEXTE ET PROBLÉMATIQUE

Contexte général

La transformation numérique des organisations et des échanges personnels a engendré une explosion du volume de données sensibles transitant sur les réseaux. Les entreprises, administrations et particuliers manipulent quotidiennement des informations confidentielles : données financières, dossiers médicaux, propriété intellectuelle, communications stratégiques.

Cette réalité s'accompagne de menaces croissantes :

- **Cyberattaques en hausse** : ransomwares, vols de données, espionnage industriel
- **Réglementations strictes** : RGPD en Europe, exigences de conformité ISO 27001
- **Environnements collaboratifs** : nécessité de partager des données sensibles entre plusieurs acteurs tout en contrôlant les accès

Problématique

Dans ce contexte, la question centrale est :

Comment concevoir un système permettant à plusieurs utilisateurs de chiffrer, déchiffrer et partager des données sensibles de manière sécurisée, tout en garantissant :

- La confidentialité des données (seuls les utilisateurs autorisés y accèdent)
- L'intégrité des fichiers (détection de toute altération)
- L'authenticité des sources (garantie de l'origine des données)
- La traçabilité des opérations (audit complet des actions)
- La conformité réglementaire (RGPD, ISO 27001)

Enjeux du projet

1. **Enjeu technique** : implémenter un chiffrement hybride (AES + RSA/ECC) robuste et performant
2. **Enjeu organisationnel** : gérer efficacement les droits d'accès multi-utilisateurs
3. **Enjeu de conformité** : respecter les normes de sécurité et de protection des données
4. **Enjeu d'utilisabilité** : proposer une interface accessible aux utilisateurs non experts

PÉRIMÈTRE ET LIMITES DU PROJET

Périmètre inclus

Le projet couvre les fonctionnalités suivantes :

- **Gestion des utilisateurs**
 - Création de comptes sécurisés
 - Authentification par mot de passe dérivé cryptographiquement
 - Génération automatique de paires de clés RSA par utilisateur
- **Chiffrement et déchiffrement**
 - Chiffrement hybride (AES-256 + RSA)
 - Support des fichiers de toute taille
 - Signature numérique pour l'intégrité
- **Partage sécurisé**
 - Attribution de droits d'accès à d'autres utilisateurs
 - Révocation des accès
 - Rechiffrement de la clé AES pour chaque destinataire
- **Journalisation**
 - Traçage de toutes les opérations
 - Horodatage et identification des acteurs

Périmètre exclu (hors scope)

Les éléments suivants ne sont pas couverts par ce projet :

- Chiffrement de flux en temps réel (streaming)
- Intégration avec des services cloud tiers (AWS KMS, Azure Key Vault)
- Authentification biométrique ou par carte à puce
- Chiffrement homomorphe
- Application mobile native
- Haute disponibilité et réplication multi-sites

Contraintes

- **Technologiques** : utilisation du framework Laravel (PHP) et des bibliothèques phpseclib/OpenSSL

- **Sécurité** : conformité aux bonnes pratiques OWASP et aux recommandations de l'ANSSI
- **Performance** : temps de chiffrement/déchiffrement acceptable pour des fichiers jusqu'à 100 Mo

HISTORIQUE DU CHIFFREMENT

La cryptographie est née du besoin fondamental de protéger l'information, d'abord pour des raisons militaires et diplomatiques, puis pour les échanges commerciaux et numériques. Chaque époque a apporté de nouvelles méthodes visant à garantir la confidentialité, l'intégrité et l'authenticité des messages, des concepts qui restent essentiels pour tout outil de chiffrement et déchiffrement multi-utilisateur.

Dans l'Antiquité, les méthodes étaient simples mais ingénieuses. Le chiffre de César consistait à décaler chaque lettre d'un message d'un certain nombre dans l'alphabet. Par exemple, avec un décalage de trois lettres, « A » devenait « D ». Cette technique permettait de protéger les messages contre les tiers, même si elle était très vulnérable aux attaques par analyse de fréquence. La scytale spartiate, utilisée autour de 500 av. J.-C., consistait à enrouler un ruban sur un bâton de diamètre spécifique pour rendre le message lisible uniquement sur un bâton identique. Ces premières méthodes illustrent le principe fondamental de la cryptographie : la sécurité repose sur la possession d'un élément secret.

À la Renaissance, la cryptographie progresse avec des méthodes plus complexes. Le chiffre de Vigenère, introduit en 1553, utilisait un alphabet différent pour chaque lettre selon une clé secrète. Cette approche poly alphabétique rendait l'analyse statistique du texte chiffré beaucoup plus difficile. L'importance de cette innovation est capitale : elle introduit l'idée que la clé, et non seulement l'algorithme, devient centrale pour la sécurité d'un message. Cette notion est directement réutilisée aujourd'hui dans les algorithmes symétriques modernes comme AES.

Le XX^e siècle voit apparaître la mécanisation du chiffrement. La machine Enigma, utilisée par l'armée allemande de 1918 à 1945, combinait plusieurs rotors pour changer l'alphabet à chaque frappe, produisant ainsi des milliards de combinaisons possibles. La cryptanalyse menée par Alan Turing et son équipe montre qu'un algorithme complexe ne suffit pas si la gestion des clés ou les procédures sont faibles. Ce principe reste fondamental dans les systèmes modernes : même un bon algorithme ne garantit pas la sécurité si la gestion des clés est déficiente.

L'arrivée des ordinateurs dans les années 1970 transforme radicalement la cryptographie. DES (Data Encryption Standard), adopté en 1977, est un premier standard de chiffrement symétrique utilisant une clé de 56 bits. Bien que maintenant considéré comme faible, il a montré que la sécurité pouvait être formalisée et évaluée. AES (Advanced Encryption Standard), adopté en 2001, est devenu la norme symétrique moderne, sécurisée et rapide, et constitue aujourd'hui la base du chiffrement des fichiers dans les systèmes hybrides.

La véritable révolution arrive avec la cryptographie asymétrique. Diffie-Hellman, en 1976, permet à deux utilisateurs de générer une clé secrète commune sur un canal non sécurisé. L'année suivante, RSA introduit la possibilité de chiffrer et de signer des messages en utilisant des clés publiques et privées distinctes. Plus tard, les courbes elliptiques (ECC) ont permis d'obtenir le même niveau de sécurité que RSA avec des clés beaucoup plus courtes et des performances supérieures. Ces innovations sont à la base des systèmes modernes de chiffrement hybride et de signatures numériques, essentiels pour des outils multi-utilisateurs sécurisés.

Aujourd'hui, la cryptographie est omniprésente. SSL/TLS sécurise les communications web, les messageries comme Signal et WhatsApp utilisent des protocoles chiffrés de bout

en bout, et les blockchains reposent sur des signatures numériques avancées pour garantir l'authenticité et l'intégrité des transactions. Les concepts historiques sont donc toujours réutilisés et combinés : chiffrement symétrique pour la performance, chiffrement asymétrique pour l'échange sécurisé des clés et signatures pour la traçabilité et la non-répudiation.

En résumé, l'histoire du chiffrement montre trois grandes évolutions. D'abord, les méthodes simples de substitution et de scytale, ensuite la cryptographie mécanique et symétrique formalisée, puis la cryptographie asymétrique moderne et les systèmes distribués. Ces étapes illustrent que la sécurité repose non seulement sur des algorithmes solides mais aussi sur une gestion rigoureuse des clés et des processus. L'outil multi-utilisateur à proposer s'inscrit dans cette continuité, en combinant chiffrement symétrique, chiffrement asymétrique et signatures numériques pour garantir la confidentialité, l'intégrité et l'authenticité des données échangées.

OBJECTIFS DU PROJET

L'objectif global du projet est de concevoir un **outil de chiffrement/déchiffrement multi-utilisateur** qui pourra répondre aux exigences modernes de sécurité dans le partage et le stockage des informations. Dans un contexte où les données circulent massivement entre plusieurs acteurs (collaborateurs d'une entreprise, utilisateurs d'un service numérique, membres d'une organisation), il devient essentiel de garantir que seules les personnes autorisées puissent accéder aux contenus sensibles.

L'idée du projet repose sur la mise en place d'un système capable d'assurer non seulement la confidentialité des données, mais également un contrôle rigoureux des accès, une authentification fiable et une traçabilité complète des opérations effectuées.

Pour répondre à ces besoins, l'outil proposé doit offrir plusieurs fonctionnalités fondamentales, chacune répondant à une exigence de sécurité ou d'usage spécifique :

1. Gestion multi-utilisateur

Chaque utilisateur dispose d'un compte personnel, séparé et identifiable.

Pertinence :

- Permet de différencier clairement les droits, les responsabilités et les actions ;
- Indispensable dans un contexte collaboratif ou organisationnel ;
- Empêche qu'un utilisateur accède par erreur ou accident aux données d'un autre.

2. Chiffrement des données

L'outil permet de chiffrer des fichiers ou messages à l'aide d'algorithmes modernes.

Pertinence :

- Assure la confidentialité quelles que soient les conditions de transmission ou de stockage ;
- Protège les données même en cas d'interception, vol de support ou défaut de sécurité externe ;
- Constitue la base de toute solution de sécurité numérique moderne.

3. Déchiffrement contrôlé et permissions d'accès

Chaque utilisateur ne peut déchiffrer que les données auxquelles il a été explicitement autorisé.

Pertinence :

- Empêche les fuites de données internes, responsables de plus de 30% des incidents de sécurité modernes ;
- Garantit une granularité fine du contrôle d'accès ;
- Permet d'adopter un modèle *least privilege* (principe du moindre privilège).

4. Gestion des permissions et partage sécurisé

Le système permet à un utilisateur autorisé de partager une clé de déchiffrement ou d'accorder des droits à d'autres utilisateurs.

Pertinence :

- Rend le système réellement collaboratif ;
- Évite le partage non sécurisé des clés (par email, messagerie, etc.) ;
- Permet de tracer précisément qui a accès à quoi et pourquoi.

5. Authentification robuste des utilisateurs

L'accès à l'outil doit être protégé par un mécanisme d'authentification fiable (mot de passe dérivé cryptographiquement, authentification à deux facteurs).

Pertinence :

- Renforce considérablement la sécurité contre l'usurpation d'identité ;
- Protège les clés privées, qui sont au cœur du modèle de sécurité ;
- Évite qu'un attaquant accède à toute l'infrastructure en compromettant un seul terminal.

6. Journalisation et traçabilité des actions

Chaque opération (connexion, chiffrement, déchiffrement, partage, modification de permissions) est enregistrée dans un journal sécurisé.

Pertinence :

- Permet un audit complet en cas d'incident ;
- Renforce la transparence et la responsabilisation des utilisateurs ;
- L'outil doit respecter les règles essentielles du RGPD, notamment la confidentialité, la minimisation des données, la sécurité dès la conception, la limitation des accès et la journalisation.

Il doit également s'aligner sur les exigences de la norme ISO 27001, en particulier les mécanismes de contrôle d'accès, la gestion cryptographique, la traçabilité et la gestion des incidents ;

- Décourage les abus internes.

7. Interface ergonomique (CLI ou GUI)

L'outil doit offrir un usage simple, clair et accessible, qu'il s'agisse d'une interface en ligne de commande ou graphique.

Pertinence :

- Améliore l'adoption de l'outil par les utilisateurs ;

- Réduit les erreurs de manipulation, souvent sources de failles ;
- Permet un usage adapté selon les besoins (développeurs, administrateurs, utilisateurs non techniques).

8. Architecture modulaire et sécurisée

La solution repose sur une conception indépendante des modules (authentification, gestion des clés, chiffrement, permissions, logs).

Pertinence :

- Facilite l'évolution, la maintenance et l'ajout de nouvelles fonctionnalités ;
- Rend les audits de sécurité plus simples et plus précis ;
- Réduit l'impact d'une éventuelle faille localisée.

SPÉCIFICATIONS FONCTIONNELLES

Cette section décrit les cas d'utilisation principaux et les acteurs du système.

Acteurs du système

- **Utilisateur standard** : peut créer un compte, chiffrer/déchiffrer ses fichiers, partager l'accès à d'autres utilisateurs
- **Propriétaire de fichier** : utilisateur ayant chiffré un fichier, il détient tous les droits sur celui-ci
- **Utilisateur autorisé** : utilisateur ayant reçu un droit d'accès à un fichier partagé
- **Administrateur** : gère les utilisateurs, consulte les journaux d'audit, peut révoquer des accès

Cas d'utilisation

CU01 : Inscription et création de compte

- **Acteur** : Utilisateur non inscrit
- **Préconditions** : Aucune
- **Scénario principal** :
 1. L'utilisateur saisit ses informations (nom, email, mot de passe)
 2. Le système valide les données et vérifie l'unicité de l'email
 3. Le système génère une paire de clés RSA pour l'utilisateur
 4. La clé privée est chiffrée avec le mot de passe dérivé (PBKDF2/Argon2)
 5. Le compte est créé et l'utilisateur est connecté
- **Postconditions** : L'utilisateur possède un compte avec sa paire de clés

CU02 : Authentification

- **Acteur** : Utilisateur inscrit
- **Préconditions** : Compte existant
- **Scénario principal** :
 1. L'utilisateur saisit son email et mot de passe
 2. Le système vérifie les identifiants
 3. Le système déchiffre la clé privée avec le mot de passe
 4. L'utilisateur accède à son espace personnel

- **Postconditions** : Session authentifiée active

CU03 : Chiffrement d'un fichier

- **Acteur** : Utilisateur authentifié
- **Préconditions** : Utilisateur connecté
- **Scénario principal** :
 1. L'utilisateur sélectionne un fichier à chiffrer
 2. Le système génère une clé AES-256 aléatoire
 3. Le fichier est chiffré avec AES-256
 4. La clé AES est chiffrée avec la clé publique de l'utilisateur
 5. Une signature numérique est générée (hash SHA-256 signé)
 6. Le fichier chiffré est stocké
- **Postconditions** : Fichier chiffré stocké, action journalisée

CU04 : Déchiffrement d'un fichier

- **Acteur** : Utilisateur autorisé
- **Préconditions** : Utilisateur ayant accès au fichier
- **Scénario principal** :
 1. L'utilisateur demande le déchiffrement d'un fichier
 2. Le système récupère la clé AES chiffrée correspondant à l'utilisateur
 3. L'utilisateur déchiffre la clé AES avec sa clé privée
 4. Le fichier est déchiffré avec la clé AES
 5. La signature est vérifiée pour garantir l'intégrité
- **Postconditions** : Fichier déchiffré accessible, action journalisée

CU05 : Partage d'un fichier

- **Acteur** : Propriétaire du fichier
- **Préconditions** : Fichier chiffré existant, destinataire inscrit
- **Scénario principal** :
 1. Le propriétaire sélectionne un fichier et un destinataire
 2. Le système récupère la clé publique du destinataire
 3. La clé AES est rechiffrée avec la clé publique du destinataire

4. Le destinataire obtient l'accès au fichier

- **Postconditions** : Accès accordé, action journalisée

CU06 : Révocation d'accès

- **Acteur** : Propriétaire du fichier

- **Préconditions** : Accès partagé existant

- **Scénario principal** :

1. Le propriétaire sélectionne un utilisateur à révoquer
2. Le système supprime la clé AES chiffrée de l'utilisateur révoqué
3. Optionnellement, une nouvelle clé AES est générée et le fichier est rechiffré

- **Postconditions** : Accès révoqué, action journalisée

EXIGENCES NON FONCTIONNELLES

Performance

- **ENF01** : Le chiffrement d'un fichier de 10 Mo doit s'effectuer en moins de 5 secondes
- **ENF02** : Le déchiffrement doit avoir des performances équivalentes au chiffrement
- **ENF03** : Le système doit supporter au moins 100 utilisateurs simultanés
- **ENF04** : Le temps de réponse pour l'authentification doit être inférieur à 2 secondes

Sécurité

- **ENF05** : Les clés RSA doivent avoir une taille minimale de 2048 bits
- **ENF06** : Le chiffrement symétrique doit utiliser AES-256 en mode GCM ou CBC avec HMAC
- **ENF07** : Les mots de passe doivent être dérivés avec PBKDF2 (100 000 itérations minimum) ou Argon2
- **ENF08** : Les clés privées ne doivent jamais être stockées en clair
- **ENF09** : Toutes les communications doivent être chiffrées (HTTPS/TLS 1.3)
- **ENF10** : Protection contre les injections SQL, XSS, CSRF (conformité OWASP Top 10)

Disponibilité et fiabilité

- **ENF11** : Disponibilité cible de 99% (hors maintenance planifiée)
- **ENF12** : Sauvegarde quotidienne des données chiffrées
- **ENF13** : Procédure de récupération en cas de perte de mot de passe (via question secrète ou email)

Maintenabilité

- **ENF14** : Code source documenté et commenté
- **ENF15** : Architecture modulaire permettant le remplacement d'algorithmes cryptographiques
- **ENF16** : Tests unitaires couvrant au moins 80% du code critique

Conformité

- **ENF17** : Conformité RGPD (droit à l'oubli, portabilité, consentement)

- **ENF18** : Alignement sur ISO 27001 (gestion des accès, journalisation, gestion des incidents)
- **ENF19** : Journalisation de toutes les opérations sensibles avec horodatage

PRINCIPE DE FONCTIONNEMENT

Nous nous attarderons ici sur la mécanique de fonctionnement de l'**outil de chiffrement/déchiffrement multi-utilisateur**, étape par étape, depuis la génération des clés jusqu'au partage sécurisé entre utilisateurs. L'idée étant de comprendre le **flux complet**, à la fois conceptuellement et dans une implémentation typique (ex : backend Laravel).

3.1 Génération et gestion des clés

Chaque utilisateur possède un couple de clés cryptographiques :

- **Clé privée** (secrète, utilisée pour déchiffrer ou signer)
- **Clé publique** (partageable, utilisée pour chiffrer ou vérifier une signature)

Objectifs :

- Associer chaque action à un utilisateur identifiable
- Permettre le chiffrement asymétrique
- Empêcher tout accès non autorisé aux données

Fonctionnement :

1. Lors de la création du compte, le système génère une paire RSA pour l'utilisateur.
2. La **clé privée** est stockée chiffrée (via Laravel Crypt, Vault, AWS KMS...).
3. La **clé publique** est stockée en clair et servira aux autres utilisateurs pour chiffrer des fichiers à destination du propriétaire.

3.2 Processus de chiffrement

Lorsque l'utilisateur souhaite chiffrer un fichier ou un message :

1. Le backend génère une **clé symétrique aléatoire (AES-256)**.
2. Cette clé symétrique sert à chiffrer le fichier (rapide & efficace).
3. La clé symétrique elle-même est ensuite **chiffrée asymétriquement** avec :
 - La clé publique du propriétaire,
 - Et éventuellement la clé publique de chaque utilisateur autorisé à accéder au fichier.

Pourquoi deux niveaux ?

- AES est très performant pour les gros fichiers
- RSA sécurise la distribution des clés

C'est le modèle standard appelé **hybrid encryption**.

Résultat :

Le fichier stocké contient :

- Le fichier AES chiffré
- La clé AES chiffrée pour chaque utilisateur autorisé
- La signature du fichier (voir partie 3.3, si intégrité incluse)

3.3 Processus de déchiffrement

Pour lire un fichier :

1. L'utilisateur demande le fichier au backend.
2. Le backend lui envoie :
 - Le fichier chiffré
 - La clé AES chiffrée correspondant à SON identité
3. L'utilisateur déchiffre la clé AES grâce à sa **clé privée**.
4. Il utilise ensuite cette clé AES pour déchiffrer le fichier.

Garanties :

- Seul l'utilisateur ayant la clé privée peut récupérer la clé AES
- Même le serveur ne doit pas avoir la clé privée

3.4 Vérification d'intégrité et signature

Cette étape intervient pour garantir :

- Que le fichier n'a pas été altéré
- Que la source est authentique

Le système compare :

- Le hash déchiffré depuis la signature
- Le hash recalculé du fichier (Le fichier est authentique si les deux hashes sont identiques)

3.5 Partage sécurisé entre utilisateurs

Dans un système multi-utilisateur, le partage sécurisé se fait ainsi :

1. Le propriétaire autorise un autre utilisateur à accéder au fichier.
2. Le backend récupère sa clé publique.
3. Il **rechiffre la clé AES** pour le nouvel utilisateur.
4. Le fichier chiffré lui-même n'est jamais modifié : seule la clé AES est ajoutée pour ce nouvel utilisateur.

Bénéfices :

- Partage instantané sans rechiffrer le fichier complet
- Chaque utilisateur possède sa propre clé AES chiffrée
- Le propriétaire peut révoquer l'accès (et régénérer une nouvelle clé AES si nécessaire)

3.6 Journalisation et audit

Chaque opération est tracée :

- Génération des clés
- Chiffrement
- Déchiffrement
- Partage
- Révocation
- Vérification d'intégrité

Objectif :

Garantir transparence, traçabilité, conformité RGPD/ISO 27001, et faciliter les audits de sécurité.

MÉTHODOLOGIE ET ARCHITECTURE DU PROJET

La méthodologie adoptée pour le développement de l'outil de chiffrement/déchiffrement multi-utilisateur repose sur une combinaison de **bonnes pratiques de développement logiciel**, de **principes cryptographiques modernes** et de **gestion sécurisée des utilisateurs et des clés**. Elle permet d'assurer à la fois la **confidentialité**, l'**intégrité**, l'**authenticité** des données, et la **facilité d'utilisation pour plusieurs utilisateurs**.

5.1 Méthodologie générale

La méthodologie suit trois grands axes :

1. Analyse des besoins et conception fonctionnelle

- Identifier les types d'utilisateurs (propriétaire d'un fichier, utilisateur autorisé, administrateur).
- Définir les opérations principales : chiffrement, déchiffrement, partage, révocation et vérification d'intégrité.
- Déterminer les exigences de sécurité : gestion des clés, signatures numériques, traçabilité, conformité RGPD et normes ISO 27001.

2. Conception technique et choix des algorithmes

- **Chiffrement symétrique** : AES-256 pour le chiffrement rapide des fichiers volumineux.
- **Chiffrement asymétrique** : RSA ou ECC pour le chiffrement des clés symétriques et les signatures numériques.
- **Hashing sécurisé** : SHA-256 pour la vérification d'intégrité.
- **Gestion sécurisée des clés** : stockage chiffré de la clé privée, clé publique partagée pour le chiffrement et les signatures.

3. Implémentation et validation

- Développement du backend Laravel pour gérer les utilisateurs, le chiffrement/déchiffrement et la vérification des signatures.
- Utilisation de bibliothèques cryptographiques fiables (ex : phpseclib pour RSA/ECC, OpenSSL pour AES).
- Tests unitaires et fonctionnels pour vérifier le bon fonctionnement des processus cryptographiques et des permissions utilisateurs.
- Audit et journalisation : chaque action (upload, déchiffrement, partage, révocation) est enregistrée pour assurer traçabilité et conformité.

5.2 Architecture fonctionnelle

L'architecture repose sur un **modèle client-serveur**, où le serveur Laravel centralise les fichiers et gère la sécurité, et les clients (utilisateurs) interagissent via une interface web ou une application.

5.2.1 Composants principaux

- **Utilisateur**

Chaque utilisateur possède un **compte sécurisé** avec une paire de clés asymétriques (publique/privée) et peut réaliser les opérations autorisées par le système.

- **Backend Laravel**

- Gestion des utilisateurs et des droits d'accès.
- Génération et stockage sécurisé des clés.
- Chiffrement et déchiffrement des fichiers (hybride AES + RSA/ECC).
- Vérification de l'intégrité via signature numérique.
- Journalisation et audit pour la traçabilité.

- **Stockage des fichiers**

Les fichiers chiffrés et les clés AES chiffrées pour chaque utilisateur sont stockés de manière sécurisée, par exemple dans le système de fichiers Laravel ou un service cloud sécurisé.

- **Interface client**

Permet à l'utilisateur de :

- Upload un fichier et le chiffrer pour lui-même ou pour d'autres utilisateurs,
- Déchiffrer les fichiers auxquels il a accès,
- Vérifier l'intégrité et la signature des fichiers,
- Gérer les permissions de partage.

5.2.2 Flux de données

Le flux typique d'un fichier sécurisé se déroule ainsi :

1. L'utilisateur génère un fichier à uploader.
2. Le backend crée une **clé AES aléatoire** pour chiffrer le fichier.
3. La clé AES est ensuite **chiffrée avec la clé publique** de chaque utilisateur autorisé.
4. Le fichier chiffré et les clés AES chiffrées sont stockés.
5. Lorsqu'un utilisateur souhaite accéder au fichier :

- Il récupère la clé AES chiffrée correspondant à son compte,
 - Déchiffre cette clé avec sa **clé privée**,
 - Puis déchiffre le fichier avec la clé AES.
6. Une signature numérique peut être vérifiée pour confirmer que le fichier n'a pas été modifié.

5.3 Architecture technique

L'architecture technique du projet peut être décrite comme suit :

- **Backend Laravel**
 - Routes sécurisées pour toutes les opérations cryptographiques.
 - Middleware pour vérifier l'authentification et les permissions.
 - Utilisation de phpseclib ou OpenSSL pour :
 - AES pour chiffrement symétrique,
 - RSA/ECC pour chiffrement asymétrique et signatures.
- **Base de données**
 - Stocke les utilisateurs, les clés publiques, les métadonnées des fichiers et les permissions.
 - Les clés privées sont chiffrées avant stockage pour éviter toute compromission.
- **Stockage sécurisé des fichiers**
 - Contient uniquement les fichiers chiffrés et les clés AES chiffrées.
 - Les fichiers originaux en clair ne sont jamais stockés côté serveur.
- **Journalisation**
 - Chaque action est enregistrée pour audit et conformité : upload, déchiffrement, partage, suppression.

5.4 Sécurité et conformité

L'architecture assure :

- **Confidentialité** : Les fichiers ne sont jamais accessibles en clair sans la clé privée correspondante.
- **Intégrité** : Chaque fichier peut être vérifié via sa signature.
- **Authenticité** : Les signatures numériques garantissent l'origine des fichiers.
- **Contrôle d'accès multi-utilisateur** : Possibilité de partager ou révoquer l'accès à un fichier de manière sécurisée.

- **Audit et traçabilité** : Toutes les actions sont historisées pour répondre aux exigences RGPD et ISO 27001.

5.5 Avantages de cette architecture

- **Scalabilité** : possibilité de gérer de nombreux utilisateurs et fichiers sans dégrader la performance.
- **Flexibilité** : ajout de nouveaux algorithmes cryptographiques ou mise à jour des clés possible sans modifier le flux global.
- **Sécurité renforcée** : séparation nette entre stockage de fichiers et gestion des clés, chiffrement hybride pour combiner sécurité et performance.
- **Compatibilité multi-utilisateur** : chaque utilisateur possède sa propre clé asymétrique et peut partager des fichiers sans révéler ses clés privées.

ANALYSE DES RISQUES

Cette section identifie les risques potentiels du projet et propose des mesures de mitigation.

Risques techniques

Risque	Probabilité	Impact	Mitigation
Faible dans l'implémentation cryptographique	Moyenne	Critique	Utiliser des bibliothèques éprouvées (OpenSSL, phpseclib), audits de code, tests de pénétration
Perte de clé privée par un utilisateur	Haute	Élevé	Procédure de récupération, possibilité de clé de secours chiffrée
Compromission du serveur	Faible	Critique	Clés privées chiffrées côté client, séparation des environnements, monitoring
Performance insuffisante	Moyenne	Moyen	Tests de charge, optimisation du code, mise en cache
Incompatibilité navigateur/système	Faible	Moyen	Tests multi-plateformes, utilisation de standards web

Risques organisationnels

Risque	Probabilité	Impact	Mitigation
Mauvaise adoption par les utilisateurs	Moyenne	Élevé	Interface intuitive, documentation utilisateur, formation
Non-conformité réglementaire	Faible	Critique	Veille juridique, consultation d'experts RGPD
Manque de ressources	Moyenne	Moyen	Planification réaliste, priorisation des fonctionnalités

Risques liés à la sécurité

Risque	Probabilité	Impact	Mitigation
Attaque par force brute	Moyenne	Élevé	Limitation des tentatives, CAPTCHA, 2FA
Attaque man-in-the-middle	Faible	Critique	TLS 1.3 obligatoire, certificate pinning
Fuite de données via logs	Moyenne	Élevé	Ne jamais journaliser de données sensibles en clair
Ingénierie sociale	Moyenne	Élevé	Sensibilisation des utilisateurs, alertes de sécurité

PLANNING ET LIVRABLES

Phases du projet

Le développement du projet est organisé en phases successives :

1. **Phase 1 : Analyse et conception**
 - Rédaction du cahier des charges
 - Conception de l'architecture technique
 - Modélisation de la base de données
 - Choix des algorithmes cryptographiques
2. **Phase 2 : Développement du noyau cryptographique**
 - Implémentation de la génération de clés RSA
 - Implémentation du chiffrement/déchiffrement AES-256
 - Implémentation des signatures numériques
 - Tests unitaires cryptographiques
3. **Phase 3 : Développement du backend**
 - Gestion des utilisateurs et authentification
 - API de chiffrement/déchiffrement
 - Gestion des permissions et partage
 - Système de journalisation
4. **Phase 4 : Développement du frontend**
 - Interface d'inscription/connexion
 - Interface de gestion des fichiers
 - Interface de partage et permissions
 - Tableau de bord utilisateur
5. **Phase 5 : Tests et validation**
 - Tests d'intégration
 - Tests de sécurité (pentest)
 - Tests de performance
 - Correction des anomalies
6. **Phase 6 : Documentation et livraison**

- Documentation technique
- Guide utilisateur
- Déploiement en production
- Formation des utilisateurs

Livrables

Livrable	Description
Cahier des charges	Document présent, définissant les spécifications du projet
Code source	Application Laravel complète avec modules cryptographiques
Base de données	Scripts de création et de migration
Documentation technique	Architecture, API, procédures de déploiement
Guide utilisateur	Manuel d'utilisation de l'application
Rapport de tests	Résultats des tests unitaires, d'intégration et de sécurité
Application déployée	Instance fonctionnelle de l'outil

CRITÈRES DE VALIDATION ET RECETTE

Critères de validation fonctionnelle

Chaque fonctionnalité doit satisfaire les critères suivants pour être validée :

Fonctionnalité	Critères de validation
Inscription	Compte créé avec paire de clés générée, email unique vérifié
Authentification	Connexion réussie avec identifiants valides, rejet des identifiants invalides
Chiffrement	Fichier chiffré non lisible sans clé, taille du fichier chiffré cohérente
Déchiffrement	Fichier restauré identique à l'original (vérification par hash)
Partage	Destinataire peut déchiffrer, non-destinataire ne peut pas
Révocation	Utilisateur révoqué ne peut plus déchiffrer
Journalisation	Toutes les actions sont tracées avec horodatage et identité

Critères de validation technique

- **Sécurité** : Aucune vulnérabilité critique détectée lors de l'audit de sécurité
- **Performance** : Respect des seuils définis dans les exigences non fonctionnelles
- **Compatibilité** : Fonctionnement sur les navigateurs Chrome, Firefox, Edge (versions récentes)
- **Couverture de tests** : Minimum 80% du code critique couvert par des tests

Procédure de recette

1. Vérification de la conformité aux spécifications fonctionnelles
2. Exécution du plan de tests (unitaires, intégration, sécurité)
3. Validation des performances selon les critères ENF01 à ENF04
4. Vérification de la documentation livrée
5. Signature du procès-verbal de recette

GLOSSAIRE

Terme	Définition
AES	Advanced Encryption Standard : algorithme de chiffrement symétrique standardisé, utilisant des clés de 128, 192 ou 256 bits
RSA	Rivest-Shamir-Adleman : algorithme de chiffrement asymétrique basé sur la factorisation de grands nombres premiers
ECC	Elliptic Curve Cryptography : cryptographie basée sur les courbes elliptiques, offrant une sécurité équivalente à RSA avec des clés plus courtes
Chiffrement symétrique	Méthode de chiffrement utilisant la même clé pour chiffrer et déchiffrer
Chiffrement asymétrique	Méthode utilisant une paire de clés : publique (chiffrement) et privée (déchiffrement)
Chiffrement hybride	Combinaison du chiffrement symétrique (pour les données) et asymétrique (pour la clé symétrique)
Hash / Empreinte	Résultat d'une fonction de hachage, permettant de vérifier l'intégrité d'un fichier
SHA-256	Secure Hash Algorithm : fonction de hachage produisant une empreinte de 256 bits
Signature numérique	Mécanisme cryptographique garantissant l'authenticité et l'intégrité d'un document
PBKDF2	Password-Based Key Derivation Function 2 : fonction de dérivation de clé à partir d'un mot de passe
Argon2	Fonction de hachage de mots de passe moderne, résistante aux attaques GPU
RGPD	Règlement Général sur la Protection des Données (règlement européen)
ISO 27001	Norme internationale de gestion de la sécurité de l'information
OWASP	Open Web Application Security Project : organisation définissant les bonnes pratiques de sécurité web
TLS	Transport Layer Security : protocole de sécurisation des communications réseau
2FA	Two-Factor Authentication : authentification à deux facteurs
API	Application Programming Interface : interface de programmation
Backend	Partie serveur d'une application
Frontend	Partie client (interface utilisateur) d'une application
Laravel	Framework PHP pour le développement d'applications web

CONCLUSION

Ce cahier des charges définit les spécifications complètes pour le développement d'un **outil de chiffrement/déchiffrement multi-utilisateur**. Le projet répond à un besoin croissant de protection des données sensibles dans un contexte où la collaboration numérique est omniprésente.

Les principaux apports de ce projet sont :

- Une **architecture cryptographique robuste** combinant chiffrement symétrique (AES-256) et asymétrique (RSA) selon le modèle hybride éprouvé
- Une **gestion multi-utilisateur** avec contrôle fin des permissions et partage sécurisé
- Une **traçabilité complète** des opérations pour répondre aux exigences de conformité
- Une **conception modulaire** facilitant l'évolution et la maintenance du système

Le respect des normes de sécurité (OWASP, ANSSI) et de conformité (RGPD, ISO 27001) garantit que l'outil pourra être utilisé dans des contextes professionnels exigeants.

Ce document constitue la référence pour l'ensemble des parties prenantes du projet et servira de base pour la validation des livrables lors de la recette finale.

BIBLIOGRAPHIE ET RÉFÉRENCES

1. **Schneier, B.** (2015). *Applied Cryptography : Protocols, Algorithms and Source Code in C*. Wiley.
2. **Ferguson, N., Schneier, B., Kohno, T.** (2010). *Cryptography Engineering : Design Principles and Practical Applications*. Wiley.
3. **NIST** (2001). *Advanced Encryption Standard (AES)*. FIPS PUB 197. <https://csrc.nist.gov/publications/detail/fips/197/final>
4. **NIST** (2013). *Digital Signature Standard (DSS)*. FIPS PUB 186-4. <https://csrc.nist.gov/publications/detail/fips/186/4/final>
5. **IETF** (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. <https://www.rfc-editor.org/rfc/rfc8446>
6. **Union Européenne** (2016). *Règlement Général sur la Protection des Données (RGPD)*. Règlement (UE) 2016/679.
7. **ISO/IEC** (2022). *ISO/IEC 27001 :2022 - Systèmes de management de la sécurité de l'information*.
8. **OWASP** (2021). *OWASP Top 10 - 2021*. <https://owasp.org/Top10/>
9. **ANSSI** (2021). *Recommandations de sécurité relatives à TLS*. <https://www.ssi.gouv.fr/>
10. **Rivest, R., Shamir, A., Adleman, L.** (1978). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM.
11. **Diffie, W., Hellman, M.** (1976). *New Directions in Cryptography*. IEEE Transactions on Information Theory.
12. **Katz, J., Lindell, Y.** (2020). *Introduction to Modern Cryptography*. CRC Press, 3rd Edition.
13. **PHP Security Consortium**. *phpseclib Documentation*. <https://phpseclib.com/>
14. **Laravel Documentation**. <https://laravel.com/docs>