NTK ULTRA-COMPRESSION

Logiciel de compression par réduction maximale indépendant du temps

Amélioration des Techniques de Compression de Données par Disruption de la Contrainte Temporelle

Exploration de stratégies avancées pour réduire considérablement la taille d'un fichier

Nathan Pelletti, Thomas Demesse, Keany Vy Khun et Litissia Ben Mohand

Réalisation dans le cadre d'un projet libre en Rust à :
L'École pour l'informatique et les techniques avancées

NTK Ultra-Compression, 24 janvier 2025

Résumé

Le logiciel NTK Ultra-Compression répond à une problématique essentielle de l'ère numérique : comment réduire la taille des données à leur minimum absolu sans compromis sur leur intégrité, tout en s'affranchissant des contraintes temporelles ?

Dans un monde où les volumes de données explosent et où les solutions actuelles de compression restent limitées par des compromis entre rapidité, efficacité et complexité, nous proposons une approche disruptive. Les solutions actuelles, telles que le format ZIP ou 7z, bien qu'efficaces, atteignent leurs limites en termes de taux de compression et d'efficacité pour les fichiers volumineux ou complexes. Ces outils sont souvent contraints par des compromis entre rapidité d'exécution et efficacité de stockage.

Cette approche disruptive s'adresse notamment aux utilisateurs ayant besoin de manipuler des fichiers volumineux, comme les logiciels, les jeux vidéo ou les bases de données, où chaque octet économisé compte.

En remettant en question les paradigmes traditionnels, ce projet vise à atteindre une compression maximale qui ne dépend ni du temps de traitement ni des ressources disponibles, mais uniquement de l'optimisation algorithmique.

Conçu en Rust pour garantir performance et sécurité, et étant distribué sous licence BSD 3-Clause, nous ambitionnons de devenir une solution universelle et intemporelle pour répondre aux défis croissants de la gestion des données dans un monde numérique en perpétuelle expansion.

Table des matières

Table des figures

1 Introduction

Version actuelle : 1.0
Licence : BSD 3-Clause
Dépôt officiel : GitHub Repository

À l'ère numérique, la compression de données est devenue une nécessité incontournable en raison de l'explosion des volumes d'informations générés et stockés. La compression consiste à réduire le nombre de bits nécessaires pour représenter des données, ce qui permet d'optimiser l'espace de stockage et d'améliorer l'efficacité des transmissions sur les réseaux. Dans un monde où les capacités de traitement des processeurs évoluent plus rapidement que celles des systèmes de stockage, il est crucial de développer des solutions qui permettent de gérer efficacement cette disparité.

Les enjeux liés à la compression de données sont multiples. D'une part, elle permet de diminuer les coûts liés au stockage matériel et à la bande passante, facilitant ainsi le transfert d'informations entre utilisateurs et systèmes. Par exemple, sans compression, une chanson ou une vidéo peut occuper un espace considérable, rendant leur stockage et leur partage peu pratiques. D'autre part, la compression joue un rôle essentiel dans l'amélioration des performances des applications, notamment dans le cadre du streaming vidéo ou audio, où la rapidité et la fluidité sont primordiales pour l'utilisateur.

Il existe deux grandes catégories de compression : **avec perte** et **sans perte**. La compression sans perte permet de restaurer les données originales sans aucune altération, ce qui est vital pour les fichiers texte ou exécutables. En revanche, la compression avec perte supprime certaines informations jugées non essentielles, ce qui est acceptable dans le cas d'images ou de fichiers audio où une certaine dégradation de qualité peut être tolérée. Ce choix entre les méthodes dépend largement des besoins spécifiques des utilisateurs et des types de données traitées.

Ainsi, la compression de données ne se limite pas à un simple gain d'espace ; elle est au cœur des défis technologiques actuels, permettant une gestion optimale des ressources numériques dans un environnement en constante évolution. Les avancées dans ce domaine sont essentielles pour garantir que les infrastructures informatiques puissent suivre le rythme croissant des demandes en matière de stockage et de transmission d'informations.

1.1 Problématique : atteindre une compression maximale sans dépendance temporelle

La problématique centrale du projet se focalise sur l'atteinte d'une compression maximale sans dépendance temporelle, un défi qui remet en question les paradigmes traditionnels de la compression de données.

Actuellement, la plupart des algorithmes de compression font face à un compromis entre le taux de compression et le temps de traitement. Plus le taux de compression visé est élevé, plus le temps nécessaire pour compresser et décompresser les données augmente. Cette contrainte temporelle limite souvent l'efficacité des solutions existantes, particulièrement pour les fichiers volumineux ou dans des contextes où la rapidité d'accès aux données est cruciale.

Ce projet propose ainsi de briser ce paradigme en se concentrant exclusivement sur l'optimisation du taux de compression, sans tenir compte du temps nécessaire pour effectuer ces opérations. Cette approche soulève plusieurs questions fondamentales :

— Comment concevoir un algorithme qui priorise uniquement l'efficacité de la compression, sans se soucier des contraintes de temps ?
— Quelles sont les limites théoriques de la compression de données lorsqu'on s'affranchit des contraintes temporelles ?
— Comment gérer et justifier des temps de compression/décompression potentiellement très longs auprès des utilisateurs ?
— Quels types de structures de données et d'algorithmes peuvent permettre d'atteindre des taux de compression jusqu'alors inatteignables ?
— Comment garantir l'intégrité et la récupération des données compressées, étant donné la complexité accrue du processus ?

En abordant ces questions, notre projet vise à repousser les frontières de ce qui est possible en matière de compression de données. L'objectif est de créer une solution qui, bien que potentiellement lente à l'exécution, offre des taux de compression inégalés, ouvrant ainsi de nouvelles possibilités pour le stockage et le partage de fichiers extrêmement volumineux.

Cette approche novatrice pourrait révolutionner des domaines tels que l'archivage à long

terme, la gestion de grandes bases de données, ou encore le partage de logiciels et de jeux vidéo de taille importante, où la priorité est donnée à l'optimisation maximale de l'espace de stockage plutôt qu'à la vitesse de traitement.

1.2 Objectifs du projet

Notre projet vise à atteindre plusieurs objectifs ambitieux dans le domaine de la compression de données :

1. **Développer un algorithme de compression ultra-performant :** Créer un algorithme capable d'atteindre des taux de compression supérieurs à ceux des solutions existantes, sans se soucier du temps de traitement.
2. **Concevoir le format de fichier .ntk :** Élaborer un nouveau format de fichier optimisé pour stocker les données compressées de manière efficace et sécurisée.
3. **Maximiser le taux de compression :** Viser à devenir la solution de compression offrant le meilleur ratio de réduction de taille de fichier au monde, surpassant les standards actuels comme ZIP ou 7z.
4. **Assurer l'intégrité des données :** Garantir que les fichiers compressés puissent être décompressés sans perte d'information, malgré la complexité accrue du processus.
5. **Implémenter une solution multiplateforme :** Développer un logiciel compatible avec les principaux systèmes d'exploitation (Windows, macOS, Linux) pour une utilisation large et flexible.
6. **Optimiser pour les fichiers volumineux :** Se concentrer particulièrement sur la compression efficace de fichiers de grande taille, tels que les logiciels, les jeux vidéo ou les bases de données.
7. **Faciliter le partage de fichiers :** Créer une solution qui simplifie le partage de fichiers volumineux en réduisant considérablement leur taille.
8. **Explorer les limites théoriques :** Pousser la recherche sur les limites théoriques de la compression de données en s'affranchissant des contraintes temporelles habituelles.
9. **Développer en Rust :** Utiliser le langage de programmation Rust pour bénéficier de ses avantages en termes de performance et de sécurité.
10. **Distribuer le logiciel en Open Source :** Publier le projet sous licence BSD 3-Clause pour encourager la collaboration et l'amélioration continue de la solution.

Ces objectifs visent à nous positionner comme une solution révolutionnaire dans le domaine de la compression de données, ouvrant de nouvelles possibilités pour le stockage et le partage d'informations numériques.

1.3 Avancement et planification

1.3.1 Répartition du développement des fonctionnalités

Catégorie	**Détails du Projet**	
	Tâches / Fonctions	**Responsable**
Algorithme de compression	- Développer l'algorithme de compression NTK - Implémenter la compression adaptative - Créer des méthodes d'analyse spectrale avancée - Concevoir des algorithmes de segmentation adaptative - Intégrer des techniques de compression contextuelle profonde	Toute l'équipe
Format de fichier .ntk	- Concevoir la structure du format .ntk - Implémenter les fonctions d'écriture et de lecture du format - Développer un système de vérification d'intégrité des données	keany-vy.khun
Interface utilisateur	- Créer une interface en ligne de commande - Développer une interface graphique multiplateforme - Implémenter des assistants guidés pour les opérations complexes et le site web	keany-vy.khun
Optimisation	- Implémenter la parallélisation massive des algorithmes - Optimiser les performances pour les fichiers volumineux - Développer des techniques de compression asymétrique	nathan.pelletti
Sécurité	- Implémenter le chiffrement AES-256 en mode GCM - Développer les fonctionnalités de stéganographie - Créer un système de gestion des clés sécurisé	keany-vy.khun et litissia.ben-mohand
Tests et évaluation	- Développer des outils de benchmarking - Créer un corpus de test diversifié - Implémenter des fonctions de calcul du taux de compression - Concevoir des tests de performance comparatifs	thomas.demesse et litissia.ben-mohand
Compatibilité	- Assurer la compatibilité multiplateforme (Windows, macOS, Linux) - Développer des plugins pour l'intégration avec d'autres logiciels	nathan.pelletti et litissia.ben-mohand
Documentation	- Rédiger la documentation technique - Créer un guide utilisateur - Préparer la documentation pour les contributeurs open source	thomas.demesse
Distribution	- Mettre en place le système de versionnage - Configurer l'intégration continue et le déploiement continu (CI/CD) - Préparer les packages de distribution pour différentes plateformes	nathan.pelletti

Figure 1.1 – *Tâches et responsabilités du logiciel de compression*

1.3.2 Tableau de planification d'avancement

Catégorie	**État de finalisation**		
	Soutenance 1	**Soutenance 2**	**Soutenance 3**
Algorithme de compression	Pas optimal	Avancé	Finalisé
Format de fichier .ntk	Fonctionnel	En amélioration	Testé et approuvé
Interface utilisateur	Concept	Prototype	Opérationnelle
Optimisation	Préliminaire	Optimisations majeures	Performance validée
Sécurité	Stéganographie	Verrouillage par clef	Pentest de l'implémentation
Tests et évaluation	Planification	Premiers tests	Résultats complets
Compatibilité	Linux, Unix	Linux, Unix et Windows	Multi-plateforme
Documentation	Version intermédiaire	Documentation finale	Documentation avancée
Distribution	Préparation initiale	Configuration CI/CD	Prêt pour release

Figure 1.2 – *Progression du développement du logiciel par soutenance*

2
État de l'Art

2.1 Techniques de compression actuelles

Les méthodes de compression de données ont considérablement évolué ces dernières années, avec des avancées significatives dans plusieurs domaines :

— **Algorithmes classiques** : Les algorithmes comme Huffman, LZ77, et leurs variantes (DEFLATE, LZMA) restent largement utilisés pour leur efficacité et leur rapidité. Le codage arithmétique, similaire au codage de Huffman mais capable de produire des codes plus courts, offre des taux de compression supérieurs.

Algorithme	**Spécifications**			
	Ratio de Compression	**Vitesse**	**Sans Perte**	**Utilisation Typique**
Huffman	60%	Rapide (100 MB/s)	✓	Texte, usage général
LZ77	65%	Rapide (80 MB/s)	✓	Texte, usage général
DEFLATE	70%	Moyen (60 MB/s)	✓	Fichiers ZIP, HTTP
LZMA	80%	Lent (20 MB/s)	✓	Archives 7z, gros fichiers
Codage Arithmétique	75%	Lent (30 MB/s)	✓	Multimédia
JPEG	90%	Rapide	-	Images
MPEG	85%	Moyen	-	Vidéo

Figure 2.1 – *Comparaison des algorithmes de compression*

— **Compression adaptative** : Des compresseurs adaptatifs construisent dynamiquement leur dictionnaire à l'encodage, s'adaptant à n'importe quelles données d'entrée pour obtenir le meilleur taux de compression possible.

— **Compression semi-adaptative** : Cette méthode effectue un premier passage sur les données pour construire le dictionnaire optimal avant l'encodage, offrant un compromis entre adaptabilité et efficacité.

— **Approches basées sur l'IA** : L'utilisation de réseaux neuronaux profonds pour la compression d'images a surpassé les approches traditionnelles, bien que leur complexité limite parfois leur utilisation pratique.

2.2 Compromis entre performance et complexité

La plupart des solutions actuelles font face à un compromis entre :

— Le taux de compression atteint
— La vitesse de compression et de décompression
— La consommation de ressources (CPU, mémoire)

Ce compromis est particulièrement visible dans la compression d'images médicales volumiques, où l'équilibre entre la qualité de l'image, la résolution, et l'efficacité de la compression est crucial.

2.3 Nouvelles approches en recherche

Plusieurs pistes de recherche visent à dépasser les limites actuelles :

— **Compression asymétrique** : Optimisation du taux de compression au détriment du temps de traitement pour des applications spécifiques, comme l'archivage de données rarement consultées.

Caractéristique	Type de Compression	
	Asymétrique	**Symétrique**
Temps de compression	Plus long	Équilibré
Temps de décompression	Plus rapide	Équilibré
Utilisation typique	Archivage de données peu consultées	Transmissions de données
Ressources requises	Déséquilibrées entre compression/décompression	Équilibrées
Optimisation	Favorise la décompression rapide	Compromis compression/décompression
Taux de compression	Potentiellement plus élevé	Variable selon l'algorithme
Flexibilité	Adaptée à des cas d'usage spécifiques	Plus polyvalente
Complexité d'implémentation	Plus élevée	Généralement plus simple

Figure 2.2 – *Comparaison entre compression asymétrique et symétrique*

— **Algorithmes adaptatifs** : Développement d'algorithmes capables de s'adapter dynamiquement au type de données pour maximiser l'efficacité de la compression.
— **Compression presque sans perte** : Cette approche permet de réduire la taille des fichiers compressés de 20% à plus de 60% avec une perte d'information contrôlée et raisonnable.
— **Stockage ADN** : Exploration de l'utilisation de l'ADN synthétique comme support de stockage ultra-dense et durable, avec des projets visant à commercialiser des "DNA drives" d'ici 2030.

2.4 Défis actuels et perspectives

Malgré ces avancées, plusieurs défis persistent :

— Atteindre des taux de compression significativement supérieurs aux méthodes actuelles sans perte de données, notamment pour les fichiers volumineux (supérieurs à 1 To).
— Gérer l'explosion continue des volumes de données, prévue pour tripler d'ici 2025.

— Développer des solutions de compression écologiques pour réduire l'empreinte carbone des centres de données.
— Intégrer les avancées en informatique quantique pour révolutionner l'analyse et la compression du Big Data.

Ces défis ouvrent la voie à des innovations futures dans le traitement et l'analyse des informations numériques, avec des implications majeures pour la gestion des données à grande échelle.

3
Concept et Innovation

3.1 Présentation du format .ntk

Le format .ntk est un nouveau format de fichier conçu spécifiquement pour le projet NTK Ultra-Compression. Bien que les détails techniques ne soient pas encore développés, ce format vise à offrir des taux de compression supérieurs aux formats existants comme ZIP ou RAR, tout en maintenant l'intégrité des données.

3.2 Principe de compression sans contrainte temporelle

NTK Ultra-Compression adopte une approche inédite en se concentrant exclusivement sur l'optimisation du taux de compression, sans se soucier du temps nécessaire pour effectuer ces opérations. Cette méthode s'inspire des récentes avancées en compression asymétrique, où le temps de compression peut être plus long, mais la décompression reste rapide. Cette approche permet d'explorer de nouvelles techniques de compression qui seraient normalement écartées en raison de leur complexité ou de leur temps de traitement élevé. Par exemple, l'algorithme pourrait utiliser des méthodes d'apprentissage automatique avancées ou des techniques d'analyse spectrale inspirées du domaine du traitement du signal, comme suggéré par les recherches sur le noyau tangent neuronal (NTK).

3.3 Avantages pour le stockage et le partage de fichiers volumineux

Les avantages de NTK Ultra-Compression pour le stockage et le partage de fichiers volumineux sont nombreux :

1. **Réduction significative de l'espace de stockage :** Avec des taux de compression potentiellement supérieurs aux méthodes actuelles, NTK pourrait permettre de stocker beaucoup plus de données sur le même espace disque.

2. **Économies sur les coûts de stockage :** Pour les entreprises gérant de grandes quantités de données, cela pourrait se traduire par des économies substantielles sur les infrastructures de stockage.
3. **Partage de fichiers facilité :** La réduction de la taille des fichiers rend leur partage plus rapide et moins coûteux en termes de bande passante.
4. **Archivage optimisé :** Pour les données rarement consultées mais devant être conservées à long terme, nous offrons une solution idéale en minimisant l'espace occupé.
5. **Gestion améliorée des big data :** Dans le contexte de l'explosion des volumes de données prévue d'ici 2025, notre projet pourrait jouer un rôle crucial dans la gestion efficace de ces masses d'informations.

Bien que le temps de compression puisse être plus long, les avantages en termes d'économie d'espace et d'efficacité de partage font de NTK une innovation prometteuse pour l'avenir de la gestion des données volumineuses.

4

Méthodologie et Développement

4.1 Choix du langage de programmation (Rust) et justification

Le projet NTK Ultra-Compression a opté pour le langage de programmation Rust, un choix stratégique justifié par plusieurs facteurs clés :

- **Performance** : Rust offre des performances comparables au C++, cruciales pour l'implémentation d'algorithmes de compression complexes.
- **Sécurité mémoire** : Le système de propriété de Rust élimine de nombreuses classes de bugs liés à la gestion de la mémoire, assurant une meilleure stabilité du logiciel.
- **Concurrence sans data races** : Les fonctionnalités de Rust pour la programmation concurrente sécurisée sont particulièrement utiles pour optimiser les processus de compression sur des systèmes multi-cœurs.
- **Écosystème moderne** : L'écosystème Rust, avec son gestionnaire de paquets Cargo, facilite la gestion des dépendances et le déploiement du projet.
- **Interopérabilité** : Rust peut facilement s'interfacer avec du code C, permettant l'utilisation de bibliothèques existantes si nécessaire.

4.2 Approches algorithmiques innovantes

Le projet pourra explorer plusieurs approches algorithmiques innovantes :

- **Compression par dictionnaire avancée** : Utilisation de dictionnaires dynamiques et adaptatifs, s'inspirant des méthodes LZ77 et LZ78, mais avec des optimisations pour les grands volumes de données.
- **Analyse spectrale avancée** : Utilisation de techniques d'analyse de Fourier et d'ondelettes pour identifier des motifs récurrents dans les données à grande échelle.
- **Compression contextuelle profonde** : Développement de modèles de prédiction basés sur l'apprentissage automatique, incluant des réseaux neuronaux profonds, pour anticiper les séquences de données.

— **Segmentation adaptative** : Mise en œuvre d'une segmentation dynamique des données basée sur leur structure intrinsèque pour optimiser la compression.
— **Parallélisation massive** : Conception d'algorithmes hautement parallélisables pour exploiter pleinement les architectures multi-cœurs modernes.
— **Codage arithmétique optimisé** : Implémentation de versions avancées du codage arithmétique, potentiellement avec des modèles de contexte adaptatifs.
— **Compression différentielle** : Utilisation de techniques de compression différentielle pour les données séquentielles ou les versions successives de fichiers.
— **Transformées réversibles** : Exploration de diverses transformées réversibles (comme la transformée en cosinus discrète ou la transformée de Burrows-Wheeler) pour prétraiter les données avant la compression.
— **Compression symbolique** : Utilisation de techniques de compression symbolique pour les données structurées ou semi-structurées.
— **Méthodes hybrides** : Combinaison intelligente de plusieurs techniques de compression pour s'adapter optimalement à différents types de données.

4.3 Architecture du logiciel et processus de développement

L'architecture du logiciel sera conçue pour être modulaire et extensible :

— **Core Engine** : Le moteur central de compression/décompression, implémentant les algorithmes principaux.
— **Plugin System** : Une architecture de plugins permettant l'ajout facile de nouveaux algorithmes ou optimisations.
— **API Layer** : Une couche d'API bien définie pour faciliter l'intégration avec d'autres logiciels ou services.
— **UI Module** : Une interface utilisateur séparée, permettant différentes implémentations (CLI, GUI, Web).

Le processus de développement suivra une approche agile, avec les pratiques suivantes :

— **Intégration continue** : Utilisation de CI/CD pour automatiser les tests et le déploiement.
— **Code review** : Revues de code systématiques pour maintenir la qualité et la cohérence du code.
— **Benchmarking continu** : Tests de performance réguliers pour suivre les améliorations et détecter les régressions.
— **Documentation exhaustive** : Maintien d'une documentation technique détaillée pour faciliter la collaboration et la maintenance.

Cette méthodologie visera à assurer le développement efficace et robuste du logiciel, en tirant parti des forces de Rust et en implémentant des approches algorithmiques innovantes dans une architecture logicielle bien structurée.

5 Performances et Résultats

Pour évaluer rigoureusement les performances du logiciel, nous mettrons en place les méthodes de mesure suivantes :

5.1 Évaluation du taux de compression

- Implémentation d'une fonction de calcul du taux de compression.
- Comparaison automatisée avec les taux obtenus par des algorithmes standards (ZIP, 7z, RAR) sur un ensemble de fichiers de test.
- Utilisation de crates Rust comme `zip`, `sevenz-rust`, et `unrar` pour implémenter ces algorithmes de référence.

5.2 Tests sur différents types de fichiers

- Création d'un corpus de test diversifié :
  - Logiciels : exécutables, bibliothèques, code source
  - Jeux : assets graphiques, fichiers de configuration, sauvegardes
  - Bases de données : fichiers SQL, dumps de grandes bases NoSQL
- Développement d'un script Rust d'automatisation des tests sur l'ensemble du corpus.
- Stockage des résultats dans une base de données SQLite via le crate `rusqlite` pour analyse ultérieure.

5.3 Analyse des temps de compression/décompression

- Utilisation de la bibliothèque standard pour des mesures précises :

```
use std::time::Instant;
```

- Implémentation d'une macro pour faciliter la mesure du temps d'exécution.

— Comparaison des temps avec ceux des algorithmes standards sur les mêmes fichiers.
— Calcul d'un ratio performance/gain d'espace pour évaluer l'efficacité globale.

Ces mesures permettront d'obtenir des données quantitatives précises sur les performances de notre logiciel, essentielles pour son développement et son optimisation.

6
Applications et Cas d'Usage

Le projet NTK Ultra-Compression offre des solutions innovantes pour divers domaines d'application, en particulier pour la gestion de données volumineuses. Voici les principaux cas d'usage envisagés :

6.1 Partage de fichiers volumineux

Le projet améliore le partage de fichiers volumineux :

- **Réduction drastique de la taille des fichiers** : Permet le transfert rapide de fichiers auparavant difficiles à partager en raison de leur taille.
- **Économie de bande passante** : Réduit significativement les coûts et le temps de transfert pour les utilisateurs et les fournisseurs de services.
- **Amélioration de l'expérience utilisateur** : Facilite le partage de contenus haute définition, de logiciels complexes ou de bases de données importantes.

6.2 Archivage à long terme

L'archivage bénéficiera grandement des capacités de NTK Ultra-Compression :

- **Optimisation de l'espace de stockage** : Permet de stocker beaucoup plus de données sur le même espace physique.
- **Réduction des coûts d'archivage** : Diminue les investissements en infrastructure de stockage pour les entreprises et les institutions.
- **Préservation améliorée** : Facilite la conservation à long terme de grandes quantités de données historiques, scientifiques ou culturelles.

6.3 Potentiel pour diverses industries

Le projet présente un potentiel significatif pour plusieurs secteurs :

— **Industrie du jeu vidéo** :
  — Réduction de la taille des jeux, facilitant le téléchargement et l'installation.
  — Optimisation du stockage des assets graphiques et sonores de haute qualité.
  — Amélioration des temps de chargement en jeu.
— **Développement logiciel** :
  — Compression efficace des packages de distribution de logiciels.
  — Optimisation du stockage et de la distribution des mises à jour.
  — Gestion améliorée des dépôts de code source et des systèmes de contrôle de version.
— **Big Data et analyse** :
  — Réduction significative de l'espace de stockage requis pour les ensembles de données massifs.
  — Accélération des processus d'analyse en réduisant les temps de lecture/écriture des données.
  — Optimisation des coûts de stockage et de traitement dans le cloud.

Ces applications démontrent le potentiel de NTK Ultra-Compression à transformer la gestion et l'utilisation des données volumineuses dans de nombreux secteurs, offrant des avantages en termes d'efficacité, de coûts et de performances.

7

Options de Sécurité Avancées

7.1 Chiffrement des Archives .ntk

Le projet intègrera un système de chiffrement robuste pour sécuriser les archives .ntk :

1. Algorithme de chiffrement :
   - Utilisation de AES-256 en mode GCM (Galois/Counter Mode)
   - Choix de GCM pour son authentification intégrée et ses performances
2. Gestion des clés :
   - Dérivation de clé via Argon2id avec paramètres configurables
   - Génération d'un sel unique de 16 octets pour chaque archive
   - Stockage sécurisé du sel et des paramètres de dérivation dans l'en-tête de l'archive
3. Processus de chiffrement :
   - Chiffrement du contenu de l'archive par blocs de 1 Mo
   - Utilisation d'un vecteur d'initialisation (IV) unique pour chaque bloc
   - Stockage des IVs dans l'en-tête de l'archive
4. Protection des métadonnées :
   - Chiffrement des noms de fichiers et de la structure de l'archive
   - Option pour masquer la taille réelle des fichiers
5. Intégrité des données :
   - Utilisation du tag d'authentification GCM pour vérifier l'intégrité
   - Calcul d'un HMAC-SHA256 sur l'ensemble de l'archive chiffrée

7.2 Stéganographie

Il offrira aussi des fonctionnalités de stéganographie pour dissimuler les archives .ntk :

1. Dissimulation dans des images :

   - Support des formats PNG et BMP (sans perte)
   - Utilisation de la technique LSB (Least Significant Bit) adaptative
   - Répartition aléatoire des données dans l'image pour résister à l'analyse statistique
   - Capacité de stockage : jusqu'à 12.5% de la taille de l'image
2. Intégration dans des fichiers audio :
   - Support du format WAV non compressé
   - Utilisation de l'insertion de phase pour une meilleure imperceptibilité
   - Capacité de stockage : environ 16 kbps pour un fichier audio 44.1 kHz
3. Processus de dissimulation :
   - Chiffrement préalable de l'archive .ntk
   - Insertion d'un en-tête stéganographique contenant la taille et un checksum
   - Dispersion des données selon un schéma pseudo-aléatoire basé sur une clé
4. Extraction des données cachées :
   - Détection automatique de la présence de données cachées
   - Extraction et vérification de l'intégrité via le checksum
   - Déchiffrement de l'archive extraite avec la clé fournie par l'utilisateur

7.3 Interface Utilisateur et Utilisation

1. Interface en ligne de commande :

```
ntk encrypt <input_file> <output_file> --password <password>
ntk decrypt <input_file> <output_file> --password <password>
ntk hide <ntk_file> <cover_file> <output_file> --password <password>
ntk extract <stego_file> <output_file> --password <password>
```

2. Options avancées :
   - Choix des paramètres de dérivation de clé pour Argon2id
   - Sélection du niveau de remplissage pour la stéganographie
   - Mode de compatibilité pour les anciennes versions
3. Intégration avec l'interface graphique existante :
   - Ajout d'onglets dédiés au chiffrement et à la stéganographie
   - Assistants guidés pour les opérations de sécurité

8 Considérations Éthiques et Légales

Le développement et l'utilisation de NTK Ultra-Compression soulève plusieurs questions éthiques et légales importantes qui doivent être soigneusement examinées et traitées.

8.1 Utilisation responsable de la technologie

- **Transparence algorithmique** :
  - Engagement à documenter et expliquer les principes de fonctionnement de l'algorithme.
  - Publication de rapports techniques détaillés pour permettre l'examen par la communauté scientifique.
- **Équité d'accès** :
  - Veiller à ce que la technologie ne crée pas de disparités d'accès aux données entre différents groupes d'utilisateurs.
  - Considérer des modèles de licence qui permettent un accès équitable, y compris pour les organisations à but non lucratif et les institutions éducatives.
- **Impact environnemental** :
  - Évaluer et minimiser l'empreinte carbone liée à l'utilisation intensive de ressources de calcul pour la compression.
  - Promouvoir l'utilisation de la technologie pour réduire globalement la consommation d'énergie liée au stockage et au transfert de données.
- **Prévention des utilisations abusives** :
  - Développer des lignes directrices claires sur l'utilisation éthique de la technologie.
  - Mettre en place des mécanismes pour prévenir l'utilisation de la technologie à des fins malveillantes ou illégales.

8.2 Aspects de sécurité et de confidentialité des données

- **Protection des données personnelles** :
  - Assurer la conformité avec les réglementations sur la protection des données (comme le RGPD en Europe).
  - Implémenter des mesures de chiffrement robustes pour protéger les données pendant la compression et le stockage.
- **Intégrité des données** :
  - Garantir que le processus de compression et de décompression préserve l'intégrité des données sans altération.
  - Mettre en place des mécanismes de vérification pour détecter toute corruption potentielle des données.
- **Sécurité du processus de compression** :
  - Protéger l'algorithme contre les attaques potentielles visant à exploiter le processus de compression pour accéder aux données.
  - Réaliser des audits de sécurité réguliers et des tests de pénétration pour identifier et corriger les vulnérabilités.
- **Contrôle d'accès et traçabilité** :
  - Implémenter des systèmes robustes de contrôle d'accès pour garantir que seuls les utilisateurs autorisés peuvent accéder aux données compressées.
  - Mettre en place des mécanismes de journalisation pour tracer toutes les opérations de compression et décompression.

En abordant proactivement ces considérations éthiques et légales, NTK Ultra-Compression vise à établir un standard élevé de responsabilité et de confiance dans l'utilisation de technologies de compression avancées, tout en garantissant la sécurité et la confidentialité des données des utilisateurs.

Bibliographie

[1] Pelletti, N., Demesse, T., Khun, K. V., & Ben Mohand, L. (2025). Amélioration des Techniques de Compression de Données par Disruption de la Contrainte Temporelle.

[2] Rust Programming Language. *Official Rust Website*. https://www.rust-lang.org/

[3] Open Source Initiative. The 3-Clause BSD License. https://opensource.org/licenses/BSD-3-Clause

[4] Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3), 379-423.

[5] Salomon, D., & Motta, G. (2010). Handbook of Data Compression (5th ed.). Springer.

[6] Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9), 1098-1101.

[7] Ziv, J., & Lempel, A. (1977). A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3), 337-343.

[8] Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic Coding for Data Compression. *Communications of the ACM*, 30(6), 520-540.

[9] Toderici, G., et al. (2017). Full Resolution Image Compression with Recurrent Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Church, G. M., Gao, Y., & Kosuri, S. (2012). Next-Generation Digital Information Storage in DNA. *Science*, 337(6102), 1628.