DOCUMENTATION POUR LE CHIFFREMENT

Architecture technique proposée

plaintext
CopierModifier
EasySave.Core
│
├── BackupManager.cs
│   └── Gère les tâches de sauvegarde
│   └── Appelle CryptoSoftWrapper.cs si un fichier doit être chiffré
│
├── CryptoSoftWrapper.cs
│   └── Encapsule les appels à CryptoSoft.exe via ProcessStartInfo
│   └── Gère les erreurs, les timings et les fuites potentielles

Logique métier : Quand chiffrer ?

L'utilisateur doit pouvoir définir :

- Si le chiffrement est activé (UseEncryption = true)
- Les **extensions concernées** (par ex. .docx, .xlsx)

Exemple :

```
{
  "UseEncryption": true,
  "ExtensionsToEncrypt": [".docx", ".xlsx"]
}
```

Gestion mémoire et sécurité

Comme dans le PROSIT :

- Process est un **objet système coûteux**. D'où le using pour forcer sa libération via le pattern IDisposable.
- Le système .NET n'a pas besoin de libération manuelle du heap, mais les ressources **non managées** comme les processus, handles de fichiers ou flux doivent être **explicitement libérées**.

- Le wrapper est **robuste** face aux plantages de CryptoSoft.

Analyse mémoire et MMU :

- Chaque appel à CryptoSoft génère un nouveau **processus** dans l'espace mémoire virtuel du système.
- Si ces processus s'accumulent (non terminés), on observe une **fragmentation mémoire** et un risque de **thrashing**.
- Le wrapper assure un WaitForExit() + using pour éviter cette fuite de contexte.

Calcul du temps de chiffrement

Ajoute dans BackupManager.cs :

```
var sw = Stopwatch.StartNew();
bool result = CryptoSoftWrapper.EncryptFile(filePath);
sw.Stop();
LogEncryptionTime(filePath, sw.ElapsedMilliseconds);
```

Et dans le log :

```
{
  "File": "C:/docs/facture.docx",
  "Encrypted": true,
  "EncryptionTimeMs": 212
}
```

Exécution dynamique : nombre et ordre variable

Ce mécanisme te permet de traiter **une séquence dynamique de fichiers**, comme demandé dans le PROSIT :

```
foreach (string file in filesToBackup)
{
    if (ShouldEncrypt(file))
    {
        CryptoSoftWrapper.EncryptFile(file);
    }

    BackupFile(file);
}
```