盗まれても復元できないだと！
～分散型セキュアストレージの試作と比較検証～

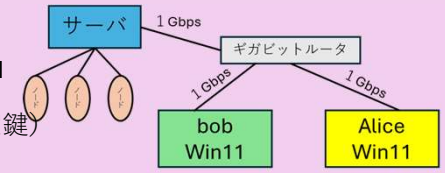
HGの情報漏洩対策Team　柴原 光希　村崎 玄昇

1．背景・目的

近年、ランサムウェアなどによるデータ被害が増加し、**データ保護の重要性**が高まっている。
一般的なNASは手軽で便利だが、暗号化やアクセス制御の強度に限界がある。
そこで、本研究では**分散保存と暗号化を組み合わせたセキュアなファイル共有システム**を開発し、NASと比較して**安全性・性能・信頼性を検証する**ことを目的とした。

2．構成

サーバー環境
Intel Core i7-3770K／ 16GB／ Ubuntu Desktop 24.04 ／ SMBサーバ／Python 3.12（ChatGPTを活用） ＋ FastAPI
ノード構成
サーバー内に3ノード（node1～node3）を仮想化し、ラウンドロビン方式で分散保存
クライアント環境
Windows 11
通信環境
1000BASE-T 有線LAN
暗号化方式
AES-256-GCM（固定鍵）



3．プログラム概要・検証方法

サーバーは **FastAPI** によりREST APIを構築し、ファイル送受信する。アップロード時にファイルを**1MiB単位で分割・暗号化**（1MiB未満は無分割）し、都合上サーバー内部の3つのノードへ順番に保存する。
クライアントは転送経過時間を自動表示し、**JWT認証を30分ごとに更新する。**
今回は、2アカウント（alice／bob）を登録し、ユーザーごとの独立領域を作成し、検証した。ダウンロード方式は、**MargeDLモード**:サーバー側で復号結合後に送信(高速・安定)
単純DLモード:各ノードから個別DLし、クライアントで結合(負荷大きめ)　を比較した。

4．結果

ここでは、安全性・性能・信頼性の3つの視点から評価した。

安全性

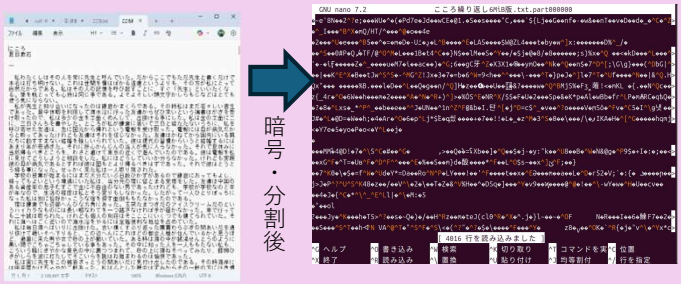
・ノードから抜き出した断片（便宜上TXTファイルを用いた）を開いた結果、意味のない文字列で**復号不能**に。（図１）
・AVIファイル断片は再生不能であり、**暗号化が有効に機能**していることの確認。
・別PCにおいてすべての断片が結合できたが、暗号鍵がないため、**復号不可。**

性能

・10MB～1GBのファイルで転送時間を測定（図２）。
・16GiBファイルでは分割数が多くなり、**処理落ち**が発生計測不可であった。
・1GiB以下では安定動作し、処理も軽快であった。

信頼性

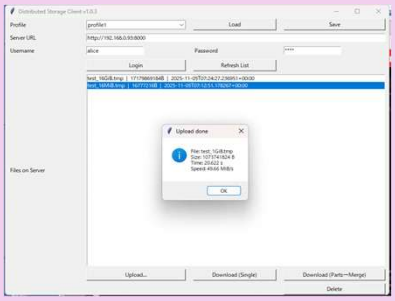
・ノード1つ欠損で結合後ファイルが**破損・再生不能**に。
・同一ファイルを複数回アップロードしても暗号結果が毎回異なり、**動的暗号処理を確認。**
・アカウントごとに領域が分離され、データ干渉は発生せず、意図したプログラム作成が成功。



（図１）TXTテキストファイルの例

	NAS		分散型セキュアストレージ		
ファイルサイズ	アップロード(s)	ダウンロード(s)	アップロード(s)	単純ダウンロード(s)	Merge ダウンロード(s)
16MiB	0.142	0.143	0.262	0.255	0.449
128MiB	1.091	1.094	1.782	1.742	3.847
1GiB	8.375	8.670	19.734	12.538	43.69
4GiB	34.949	34.644	110.125	55.626	237.437
16GiB	139.874	139.284	計測できず	計測できず	計測できず

（図２）各方式における速度比較（3回平均）



制作したクライアントの例

5．考察

16GiBの実験で、**分割単位が細かすぎるため、処理落ち**が発生することを確認した。対策としては、ファイルサイズに応じて**分割単位を自動調整**すれば、大容量にも対応できるのではないか。
一方で、**1GiB以下では、MargeDL時でも1分以下**で処理が終わり、安定して動作した。よって、現構成は中容量ファイル向けとして実用的であることがわかる。また、**単純DL方式**のほうが**軽く安定**していた。ファイルサイズや通信環境に応じて、最適な**結合方式を自動で切り替える仕組み**が有効と考えられる。
現在は3ノードが同一サーバー内で動作しており、実際の分散環境での影響は**未検証**である。今後はLAN内の別PCをノードとして構築し、**遅延・同期・通信最適化を含めた検証**を行う必要がある。

6．まとめ・展望

分散保存と暗号化を組み合わせることで、**NASより高い安全性を実現**した。ノード単体では内容を復元できず、**機密性の高い構造**であることを確認した。
1GiB以下のファイルでは安定・実用的速度を維持したが、大容量ファイルでは分割処理が負荷となり、**最適化が必要**である。
今後は、
・分割単位の自動調整による大容量対応
・サーバー／クライアント処理の効率化
・実ノード分散による通信検証
・暗号化の高速化（GPU利用・並列化）
を進め、より実用的な方法を実装・検証していきたい。