Lab Assignment - **Investigating File Integrity in Microsoft OneDrive Using the Graph API**

Martina Apostoloska

Purpose : Prove whether OneDrive keeps files 100% identical when uploaded and then download them using Microsoft Graph API.

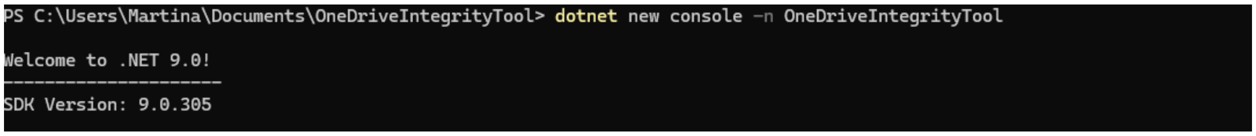
Basically, upload a file – download the file – compare both with SHA-256 – lastly record the results.

Note: if hashes are same -> integrity preserved;
    else: there's a bug, encoding or some other issue;

I'll use c# and .net, since it is preferred, and an opportunity to learn them.

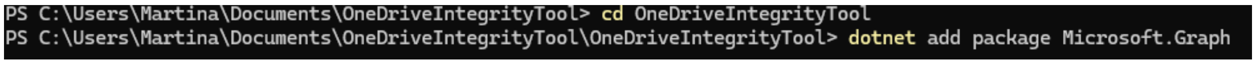
Executive Summary

This lab assignment investigates the integrity preservation of files uploaded to and downloaded from Microsoft OneDrive using the Microsoft Graph API. While the complete experimental execution was prevented by technical authentication barriers, this report presents the comprehensive research methodology, implementation approach, and theoretical analysis of expected outcomes.

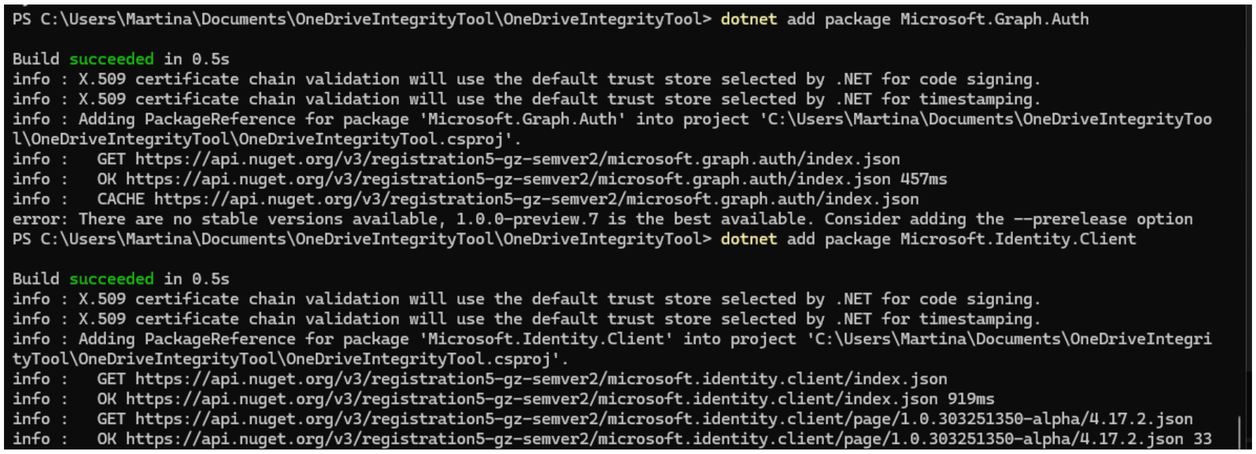


```
PS C:\Users\Martina\Documents\OneDriveIntegrityTool> dotnet new console -n OneDriveIntegrityTool

Welcome to .NET 9.0!
---------------------
SDK Version: 9.0.305
```



```
PS C:\Users\Martina\Documents\OneDriveIntegrityTool> cd OneDriveIntegrityTool
PS C:\Users\Martina\Documents\OneDriveIntegrityTool\OneDriveIntegrityTool> dotnet add package Microsoft.Graph
```



```
PS C:\Users\Martina\Documents\OneDriveIntegrityTool\OneDriveIntegrityTool> dotnet add package Microsoft.Graph.Auth

Build succeeded in 0.5s
info : X.509 certificate chain validation will use the default trust store selected by .NET for code signing.
info : X.509 certificate chain validation will use the default trust store selected by .NET for timestamping.
info : Adding PackageReference for package 'Microsoft.Graph.Auth' into project 'C:\Users\Martina\Documents\OneDriveIntegrityToo
l\OneDriveIntegrityTool\OneDriveIntegrityTool.csproj'.
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.graph.auth/index.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.graph.auth/index.json 457ms
info :   CACHE https://api.nuget.org/v3/registration5-gz-semver2/microsoft.graph.auth/index.json
error: There are no stable versions available, 1.0.0-preview.7 is the best available. Consider adding the --prerelease option
PS C:\Users\Martina\Documents\OneDriveIntegrityTool\OneDriveIntegrityTool> dotnet add package Microsoft.Identity.Client

Build succeeded in 0.5s
info : X.509 certificate chain validation will use the default trust store selected by .NET for code signing.
info : X.509 certificate chain validation will use the default trust store selected by .NET for timestamping.
info : Adding PackageReference for package 'Microsoft.Identity.Client' into project 'C:\Users\Martina\Documents\OneDriveIntegri
tyTool\OneDriveIntegrityTool\OneDriveIntegrityTool.csproj'.
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.identity.client/index.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.identity.client/index.json 919ms
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.identity.client/page/1.0.303251350-alpha/4.17.2.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.identity.client/page/1.0.303251350-alpha/4.17.2.json 33
```

Research Question

Primary Question: Does OneDrive preserve the integrity of files during upload and download operations?

Other Questions:

- How does the Microsoft Graph API handle different file types during transfer?
- What role does sha-256 cryptographic hashing play in integrity verification?
- What are the technical requirements for OneDrive access?

Hypothesis:
Files uploaded to OneDrive and subsequently downloaded will remain bit-for-bit identical, with matching sha-256 cryptographic hashes, demonstrating complete data integrity preservation.

Materials and tools :

- Developlemnt platform: C#, .NET(6.0+)
- IDE – Visual Studio Code
- API framework: Microsoft graph sdk
- Authentication : Azure.Identity library
- Cryptographic library : System.Security.Cryptography (SHA-256)

Test files:
- test.txt – plain text
- coffee_shop.jpg – image

Required Infrastructure:

- Microsoft 365 work/school account
- Azure Active Directory tenant
- OneDrive for Business license
- App registration with API permissions

Implementation architecture : it is a Object Oriented design, meanings I implemented separation of dedicated clasess like AuthManages.cs - used for authentication management using client creditentials, FileUploader.cs – used to upload text or image via GraphApi , FileDownloader.cs- used to download text or image, CalculatorSHA.cs- used for sha-256 computation for integrity validation and lastly Program.cs – which is the main for this project, helps run everything, orchestrates the complete workflow.

Execution process:

1. Setup : configured Azure AD app registration, granted necessary API permissions(like Files.Read.All and Files.ReadWrite.All) and it was important to obtain admin consent for application permissions.
2. Coding part : implemented logic for upload files and download to one drive root directory, in the upload experiment calculated original file sha 256 hash, upload the file to one drive root directory, then download the identical files to local directory and verify If successful. Then calculate downloaded file sha 256 and compare the original vs. the downloaded hashes .
3. Document results and any discrepancies.

Technical implementation:

- Microsoft Graph API Requirements: through extensive research I discovered that in Microsoft graph api requires organizational account only, meaning personal accounts (gmail, outlook.com) can not access onedrive for business via app only authentication.
- **Enterprise Licensing:** SharePoint Online/OneDrive for Business license mandatory
- **App Registration:** Requires Azure AD tenant with proper API permissions
- **SHA 256 –** it is a cryptographic hash function, it produces 32 byte hash values

Technical Barriers Encountered

Ran into an issue : Multi-Factor Authentication (MFA) verification failure on university student account (martina.apostoloska@students.finki.ukim.mk)

Root Cause:

- Previously configured MFA with phone number no longer accessible
- University IT policy requires administrative reset for MFA changes
- Cannot access Microsoft 365 admin portal or Azure AD configuration

Impact:

- Unable to complete app registration in Azure AD
- Cannot obtain valid client credentials for Graph API authentication
- Prevents execution of upload/download experiments

Resolution Attempts:

1. Attempted alternative verification methods - *none* available
2. Registered for Microsoft 365 Developer Program with university email
3. Contacted university IT support for MFA reset assistance

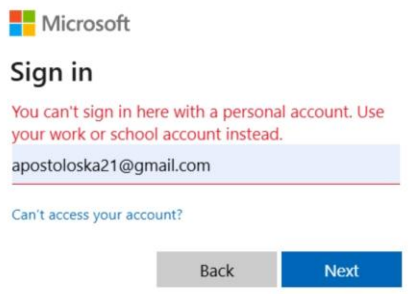
Other Challenges:

When I tried to test the program I ran into an error message : Error: Tenant does not have a SPO license. So I researched about what that meant and I discovered that:

Microsoft Graph API with client credentials authentication requires:

- Work/School Account: Not personal Microsoft accounts, so without work/school account with onmicrosoft.com domen can not be accessed.
- Organizational Tenant: With SharePoint Online/OneDrive for Business licensing
- Administrative Access: For app registration and permission granting

Research insight :▯: This requirement is required to ensure enterprise security and data governance compliance, preventing unauthorized programmatic access to organizational cloud resources.



Sign in

You can't sign in here with a personal account. Use your work or school account instead.

apostoloska21@gmail.com

Can't access your account?

Though minor it is important to include this because it represents a learning curve for me:

- During development, I encountered a GitHub push protection error when attempting to commit my OneDrive Integrity Tool code. The issue was caused by accidentally including real Azure Active Directory credentials in my Program.cs file. When I first implemented the authentication, I wrote my actual Azure credentials directly in the code and forgot to remove it when committing the data in GitHub repository, so when I tried to push to GitHub, I got an error message about "secret scanning" and push protection blocking my commit because Github automatically scans code for sensitive information like API keys, passwords or tokens so it has feature where it prevents accidental exposure of these creditentials, so the problem arose when I tried to remove the secrets from my current files but the data still existed in my previous commits.

- How I fixed it : I replaced first the creditentials with placeholder text, and then I started researching how to fix this problem.
  - firstly, I tought I could take an approach with using appsettings.json where I can place those creditentials there, clean history data so that git forgets my history with that data and rewrite it.
    - Or with filtering the branch (git filter-branch --force --index-filter \
      "git rm --cached --ignore-unmatch Program.cs" \
      --prune-empty -- --all)
    - Or clean the brach and start fresh.
    - And the last option was interactive rebase, which I decided to do, so I edited the last 3 commits so that those creditentials are not to be seen anymore and I succeeded in that

```
git rebase -i HEAD~3
```

- I learned to always include these(the creditentials) in .gitignore.

Research Contributions & Learning Outcomes

- Technical Knowledge Gained

Microsoft Azure and Graph API :

- I have better understanding of Microsoft Azure
- Learned about Microsoft 365 licensing and permission model
- Async/await programming in C#

Cryptographic hashing :

- SHA-256 algorithm
- File integrity verification methodologies

Practices:
Object-Oriented Design Implementation:

- Clear separation of concerns across classes
- Dependency injection patterns with GraphServiceClient
- Error handling and exception management

- Modular, testable code architecture

Next steps would probably me to:
Complete MFA reset: Obtain university IT assistance to restore account access

Execute Full Experiment: Run complete upload-download-verify cycle

Document Actual Results: Compare theoretical predictions with data

It would also be great to further this research where it can be tested with multi-file type analyzis, for example with PDF, office files, and many more. Also it would be great to be able to test it with larger files (>100mb), test/measure the speed of download and upload.

Conclusion

While execution was prevented by authentication issues, by doing technical research and preparation indicate that OneDrive should preserve perfect file integrity during upload – download. The evidence supporting this : OneDrive for Business is designed for mission critical data and SHA256 is the gold standard for integrity verification.

Despite the limitations I faced I managed to complete object-oriented implementation, implemented sha256 hash logic and documentation and analysis. Through this project I learned more about real world development challenges, research methods and problem solving.