NLock 1.0

Technical documentation

Contents

1. Introduction

NLock is a simple application which uses Neurotechnology Biometric SDK (more specifically VeriLook SDK) to lock files on your personal computer. Using conventional PC webcam NLock can lock your files so that only you can unlock it again.

1.2. About This Guide

This document describes all the technical details of the NLock application.

2. About

Most of the Biometric operations starts with a capturing process and this process will make a person biometric details available on digital realm. Thereafter a template extraction process will be carried out to filter out extract information needed to identify a person again by matching the extracted template to a template created from same person at a future time.

NLock uses face biometric of a person which is provided by Neurotechnology VeriLook SDK and all the biometric operations are done via the Neurotechnology biometric client. NLock uses this face template representation of a person in digital realm to lock the files supplied to its locking process by the user so at a later time this file can be unlocked by a user after the VeriLook SDK identifies the user.

2.1 System Requirements

Microsoft .NET framework 4.5+
Ram usage - Currently high due to current in-memory locking implementation
Basically you will need ram as bigger as the size of the (.nlk) file you are preparing

3. Technical Implementation

This section is describing the overall high level implementation of the NLock.

3.1 Locking Mechanism

NLock does not use a good encryption scheme. More like a steganography or a obfuscation. Current simple locking mechanism used AES - Rijndael for the locking but the way it was used is poor that current locking strength depend on the programme logic.

$$\text{lockedData} = E_{(\text{template})}(\text{Data})$$

$$\text{lockedTemplate} = E_{(\text{lockedData})}(\text{template})$$

Locking process is simple as mentioned above and after the process both **lockedData** and **LockedTemplate** will be saved as a one .nlk file.

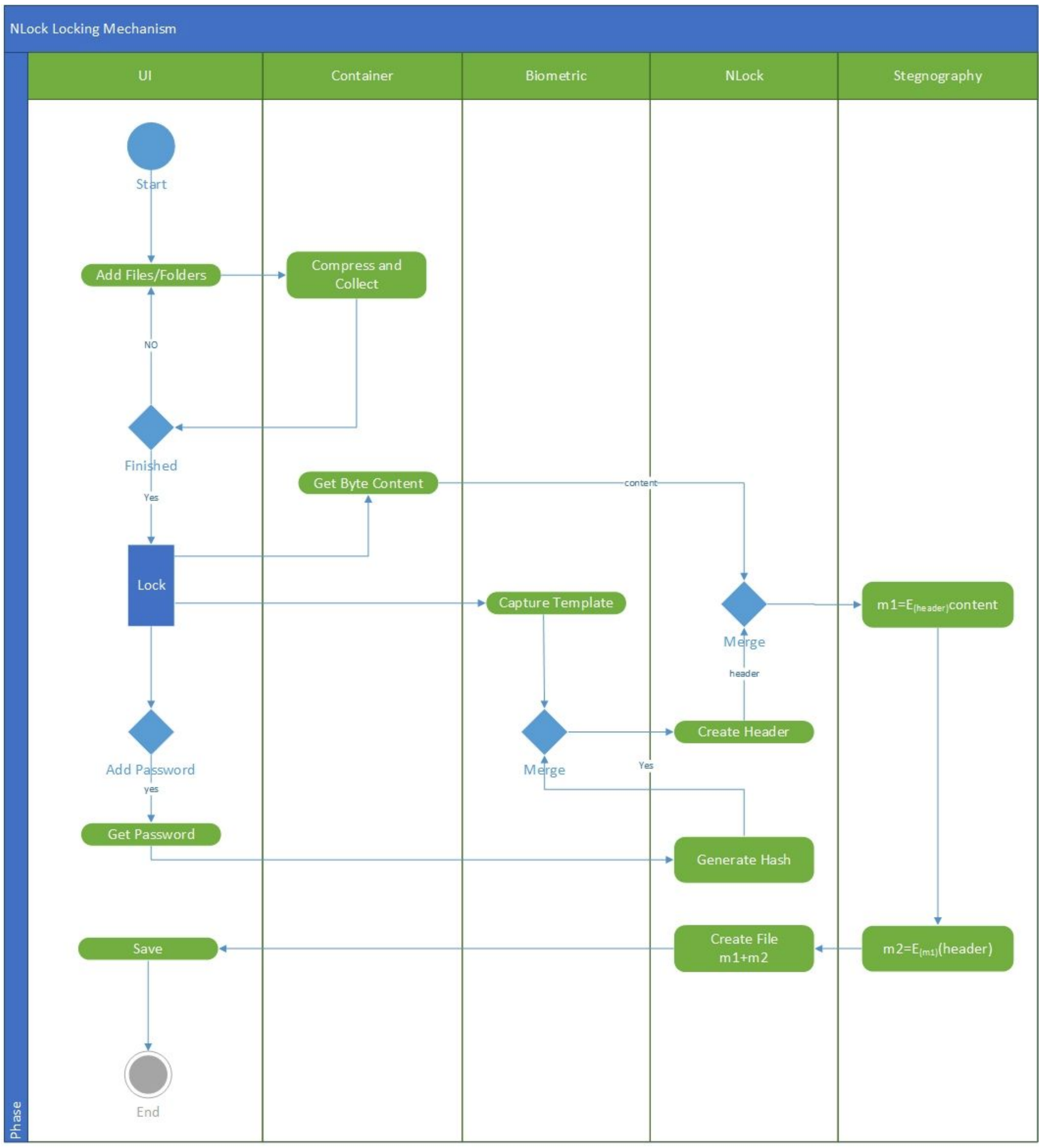
Current implementation uses AES - Rijndael for this Encryption algorithm and security depends on the user identifying how file is saved in the .nlk file and can be decrypted following the same steps backward using decryption. Internal Class diagram used the encryption algorithm as a interface which can be instantiated any other algorithm as well.

3.2 File Container

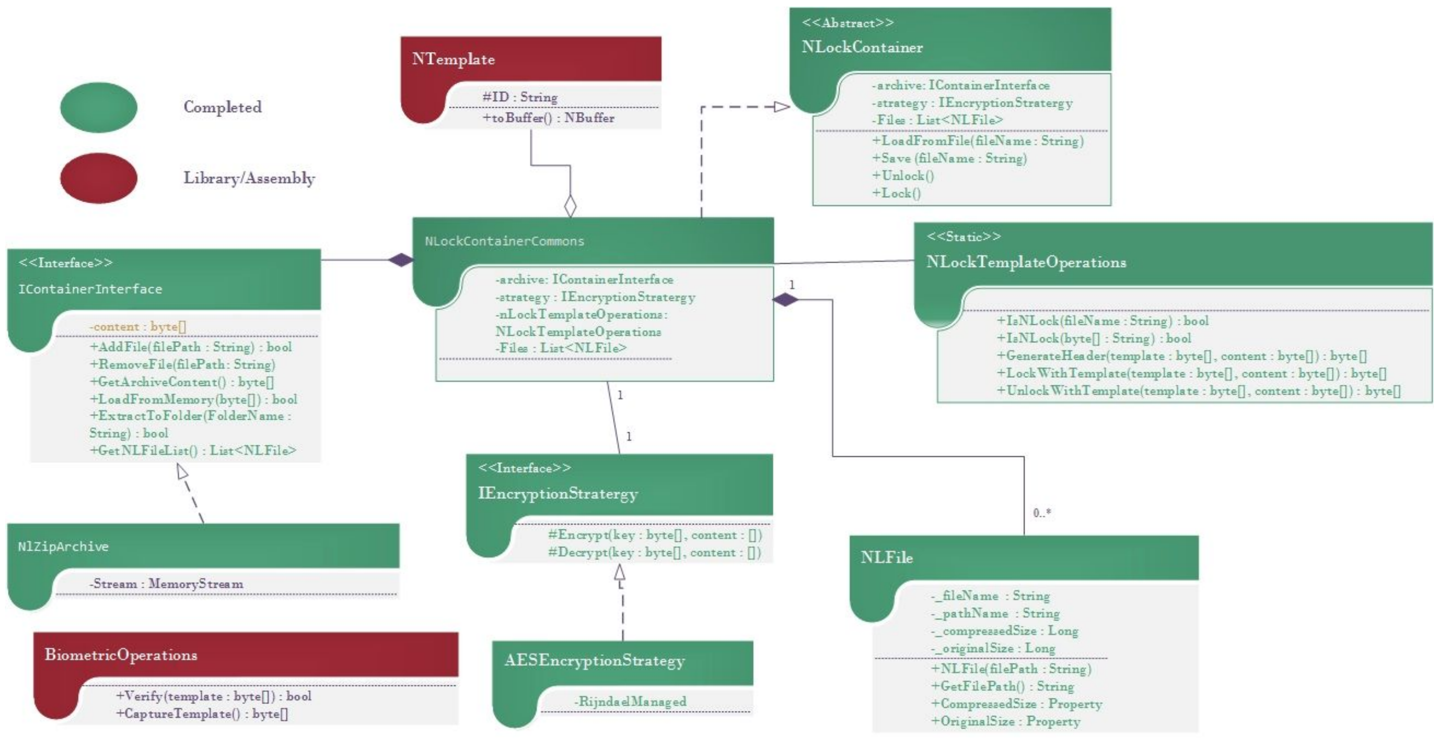
Internal file structure uses a zip archive to contain the files and exploit the compression capability of the zip archive and file management. This is also implemented by instantiating an interface to decouple the architecture thus making it easy to implement it using another good container.

3.3 Possible Improvements

Current zip archive uses a memory stream to store the file content and thus make NLock very poor on memory usage as a application. Possible future solutions would be to replace it by a file stream and have asynchronous operations for file handling.



Locking Mechanism


Completed
Library/Assembly
NTemplate
#ID : String
+toBuffer() : NBuffer
<<Abstract>>
NLockContainer
-archive: IContainerInterface
-strategy : IEncryptionStratergy
-Files : List<NLFile>
+LoadFromFile(fileName : String)
+Save (fileName : String)
+Unlock()
+Lock()
<<Interface>>
IContainerInterface
-content : byte[]
+AddFile(filePath : String) : bool
+RemoveFile(filePath: String)
+GetArchiveContent() : byte[]
+LoadFromMemory(byte[]) : bool
+ExtractToFolder(FolderName : String) : bool
+GetNLFileList() : List<NLFile>
NLockContainerCommons
-archive: IContainerInterface
-strategy : IEncryptionStratergy
-nLockTemplateOperations : NLockTemplateOperations
-Files : List<NLFile>
1
<<Static>>
NLockTemplateOperations
+IsNLock(fileName : String) : bool
+IsNLock(byte[] : String) : bool
+GenerateHeader(template : byte[], content : byte[]) : byte[]
+LockWithTemplate(template : byte[], content : byte[]) : byte[]
+UnlockWithTemplate(template : byte[], content : byte[]) : byte[]
1
1
<<Interface>>
IEncryptionStratergy
#Encrypt(key : byte[], content : [])
#Decrypt(key : byte[], content : [])
0..*
NlZipArchive
-Stream : MemoryStream
NLFile
-_fileName : String
-_pathName : String
-_compressedSize : Long
-_originalSize : Long
+NLFile(filePath : String)
+GetFilePath() : String
+CompressedSize : Property
+OriginalSize : Property
BiometricOperations
+Verify(template : byte[]) : bool
+CaptureTemplate() : byte[]
AESEncryptionStrategy
-RijndaelManaged

Class Diagram

4. Limitation

- Cannot lock large files
- Cannot lock large number of files

Refer section 3.3

5. Neurotechnology Biometric Configurations

Template Size	- Small template
Quality Threshold	- 50%
Matching Threshold	- 48%
Faces Matching Speed	- Low