BackupAes256

Benutzerhandbuch

Ein Programm für gespiegelte, unverschlüsselte Sicherungskopien.

- quelloffen und kostenlos
- einfach und übersichtlich
- synchronisiert jeweils ein Verzeichnis und dessen Unterverzeichnisse

Das ist das Handbuch zu Programmversion 1.0.4 vom 26.12.2020.

Aktualisierungen sind unter der Adresse https://github.com/dasSubjekt/BackupAes256 in längeren und unregelmäßigen Abständen geplant.

Kontakt: praxis@matthias-gloeckner.de
https://twitter.com/dasSubjekt

Inhaltsverzeichnis

1. Einleitung

Der Name dieser Software entstand zu einer Zeit, als ich noch nicht wusste, dass

1. man verschlüsselte Sicherungskopien am einfachsten anlegt, indem man die Daten auf einen hardware-verschlüsselten externen Datenträger kopiert und

2. es nicht ganz einfach ist, den Schlüssel bzw. das Passwort zu den verschlüsselten Dateien möglichst wirksam vor Trojanern, unbefugtem Zugriff und dem Durchprobieren aller Zeichenkombinationen zu schützen.

Aus diesen Gründen habe ich die Verschlüsselungsfunktion wieder entfernt und arbeite jetzt an anderen quelloffenen Projekten für Verschlüsselung.

Weil es nicht ausgeschlossen ist, dass ich zu einem späteren Zeitpunkt diesem Programm wieder Verschlüsselung hinzufüge und weil eine Programmversion aus dem Februar 2020 als Teil des Projekts »Arctic Code Vault«[1] jetzt unter dem Eis von Spitzbergen liegt, bleibt der Name BackupAes256 bestehen. Der Programmkode enthält noch Zeilen, die erst mit Verschlüsselung wieder einen Sinn erhalten würden. Sie konsequent zu entfernen, werde ich mir bis auf Weiteres nicht die Zeit nehmen.

In der vorliegenden Version spiegelt das Programm alle Dateien eines Quellverzeichnisses in ein Zielverzeichnis. Ich benutze es zum Anfertigen der Sicherungskopien in meiner psychotherapeutischen Praxis und ich meine, dass es zuverlässig arbeitet. Garantie oder Haftung übernehme ich dafür trotzdem nicht.

Über Rückmeldungen bin ich dankbar, ob dieses Programm genutzt wird, ob Fehler auftreten und ob zusätzliche Funktionen gewünscht werden.

2. Das Programm zum Laufen bringen

2.1. Warum so kompliziert?

Als Benutzer des Betriebssystems Windows sind wir es gewohnt, eine Installationsdatei herunter zu laden, sie mit Doppelklick zu starten und die Nachfrage, ob wir wirklich wissen, was wir tun, mit dem gewohnten Klick auf »ja« zu beantworten. Für das Programm BackupAes256 gelten andere Regeln.

Wollte man wirklich nachvollziehen, was so eine Installationsdatei und das von ihr installierte Programm auf unserem Computer tun, müsste man großen technischen Aufwand betreiben. Funktioniert das Programm tatsächlich so, wie seine Dokumentation es beschreibt? Schickt es Daten in das Internet, ohne mich darüber zu informieren? Enthält es einen Computervirus, den der Virenscanner nicht erkennt? Für diese Zwecke gibt es spezialisierte Computerprogramme, sogenannte Disassembler und Decompiler. Jedoch braucht man zu

1 https://archiveprogram.github.com/arctic-vault/

ihrer Benutzung eine fortgeschrittene Programmiererausbildung. Außerdem hat die Anwendung solcher Programme auf kommerzielle Software rechtliche Hürden.

Um Transparenz und Sicherheit den Vorrang zu geben, habe ich für mein Programm eine Entscheidung getroffen, die es wesentlich einfacher macht, die eben genannten Fragen zu beantworten. Der Preis dafür ist ein etwas komplizierter und langwieriger Weg, bis das Programm zum ersten Mal auf dem eigenen Computer läuft. Als willkommenen Nebeneffekt dieser Umstände können auch Benutzer*innen ohne Programmierkenntnisse einige Programmfunktionen selbst anpassen. Beispielsweise, ob die Benutzeroberfläche auf Deutsch oder Englisch beschriftet ist.

Aus den Begriffen Disassembler und Decompiler lässt sich erahnen, dass es sich um die Gegenteile von Assemblern und Compilern handelt. Assembler sind sehr nah an der Maschinensprache des Computerprozessors angesiedelt und wandeln eine so genannte Assemblersprache in Prozessorbefehle um. Mit jedem Jahrzehnt der Computer- und Softwareentwicklung grenzte sich ihre Verwendung immer weiter auf Spezialbereiche ein und sie wurden durch Compiler für höhere Programmiersprachen ersetzt. Wer BackupAes256 nutzen möchte, braucht also einen Compiler, und zwar einen für die Programmiersprache C# (gesprochen wie »see sharp« - sieh scharf). Anstelle einer Installationsdatei muss der Quelltext des Programms aus dem Internet heruntergeladen werden. Der Compiler übersetzt diesen dann in Prozessorbefehle.

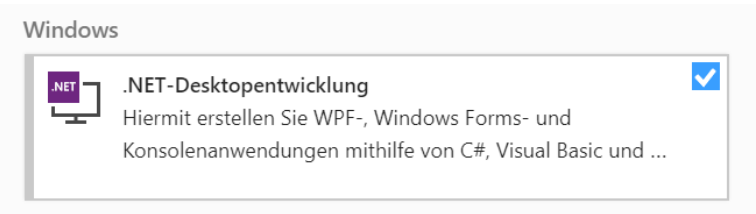
Bis vor einigen Jahren war das ungefähr so kompliziert, wie es klingt. Dank Microsoft (das muss hier einmal lobend gesagt werden) ist es einfacher geworden.

2.2. Herunterladen von Visual Studio Community

Um den Quellkode auf übersichtliche Weise anzuzeigen, bei Bedarf daran Änderungen vorzunehmen und schließlich ein lauffähiges Programm daraus zu erzeugen, wird eine sogenannte Integrierte Entwicklungsumgebung benötigt. Die Abkürzung dafür ist »IDE«, von englisch »Integrated Development Environment«.

In diesem Fall ist das »Visual Studio Community 2019«. Diese Entwicklungsumgebung kann von der Adresse https://visualstudio.microsoft.com/de/vs/ heruntergeladen werden. Bitte achten Sie darauf, nicht versehentlich die kostenpflichtige Professional- oder Enterprise-Version herunterzuladen. Da es sich tatsächlich um eine »Umgebung« handelt, nicht nur um ein Programm, kann die Installation bis zu einer Stunde oder länger dauern. Manchmal kommt es dabei zu kleineren Fehlern, die ignoriert werden können.

Nicht alle der zahlreichen Bestandteile müssen installiert werden. Benötigt wird die ».NET-Desktopentwicklung«, bei der ein Häkchen zu setzen ist:



Wer Visual Studio Community länger als 30 Tage nutzen möchte, muss sich mit einer E-Mail-Adresse bei Microsoft registrieren. Microsoft hält sich bedeckt, welche Nutzerdaten es aus Visual Studio ausliest und in den Vereinigten Staaten oder an dritten Orten speichert. Deshalb empfiehlt es sich, zeitnah die ausführbare Programmdatei zu erzeugen.

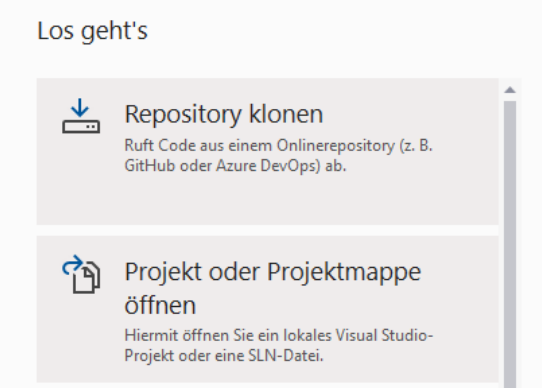
Wenn das Programm nicht nur ausprobiert wird, sondern mit vertraulichen Daten gearbeitet werden soll, ist es sicherer, die ausführbare Datei »BackupAes256.exe« immer direkt zu starten, nicht aus Visual Studio heraus.

2.3. Herunterladen des Quellkode

Linus Torvalds ist der Ideen- und Namensgeber des Betriebssystems Linux. Er ist jedoch auch Ideen- und Namensgeber einer Versionsverwaltung für Software namens Git. Zumindest scherzte er, dass Git (Englisch für Depp) nach ihm benannt sei.[2]

Ein Unternehmen aus San Francisco entwickelte Git zu einem Speicherdienst für Softwareprojekte namens Github weiter. BackupAes256 steht dort zum Herunterladen bereit. Die Adresse ist

https://github.com/dasSubjekt/BackupAes256



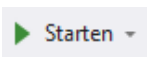
Am einfachsten geht das Herunterladen, indem man auf der Startseite von Visual Studio Community »Repository klonen« wählt, die obige Adresse einträgt und auf »Klonen« klickt.

Alternativ können alle benötigten Dateien in Form eines Zip-Archivs von Github heruntergeladen werden. Anschließend lässt sich dieses mit der rechten Maustaste anklicken, auf der Festplatte entpacken und mit »Projekt oder Projektmappe öffnen« anzeigen.

Oder man sucht im entpackten Verzeichnis die Datei »BackupAes256.sln« und öffnet sie per Doppelklick.

2.4. Ein lauffähiges Programm erzeugen

In Visual Studio Community startet man das Programm mit der Taste F5 oder mit einem Klick auf den grünen Pfeil.



Erläuterungen zum Unterschied zwischen Debug- und Releasekonfiguration und eine ausführliche Anleitung zum Umgang mit der erstellten ausführbaren Datei folgen hier

2 https://de.wikipedia.org/wiki/Git#Name

vielleicht zu einem späteren Zeitpunkt. Sie können jede der beiden Konfigurationen verwenden.

Unter Projekt => BackupAes256-Eigenschaften => Build => Ausgabe => Ausgabepfad ist der relative Speicherort innerhalb des Quellkode-Verzeichnisses zu finden, an dem die ausführbare Datei »BackupAes256.exe« erzeugt wurde. Innerhalb des Quellkode-Verzeichnis ist das in der Regel der Pfad »\BackupAes256\BackupAes256\bin\x64\Release«. Es genügt, diese ausführbare Datei und das Unterverzeichnis »Assets« mit seinem Inhalt an einen beliebigen Ort zu kopieren. Dort kann das Programm nun immer direkt gestartet werden, Visual Studio wird nicht mehr benötigt.

Zur Ausführung unter Windows benötigt BackupAes256 das NetFramework 4.7.2 oder höher. Falls es nicht schon auf dem Computer vorhanden war, wird es automatisch als ein Teil von Visual Studio Community installiert. Um BackupAes256 auf weiteren Windows-Computern ausführen zu können, muss zuvor gegebenenfalls das NetFramework 4.7.2 von der entsprechenden Microsoft-Seite aus der Spalte »Runtime«[3] heruntergeladen und auf dem Computer installiert werden. Fehlt es, erscheint beim Versuch des Programmstarts der entsprechende Hinweis.

Wer ein anderes Programmsymbol als das voreingestellte wünscht, kann eine Symboldatei z.B. von http://www.iconarchive.com/ herunterladen und in Visual Studio Community unter Projekt => BackupAes256-Eigenschaften => Anwendung => Symbol festlegen. Die Festlegung für das kleine Symbol am Programmfenster oben links erfolgt separat in der Datei »MainWindow.xaml«.

Als Sprache der Benutzeroberfläche kann entweder Deutsch oder Englisch gewählt werden. Dazu muss unter Projekt => BackupAes256-Eigenschaften => Build => Symbole für bedingte Kompilierung als Schlüsselwort entweder DEUTSCH oder ENGLISH eingetragen werden. Während die Benutzeroberfläche vollständig in beiden Sprachen vorliegt, sind noch nicht alle Fehlermeldungen übersetzt.

2.5. Lizenzen

Die Lizenz für Visual Studio Community[4] räumt Nutzern weitreichende Rechte ein. Dafür »zahlen« diese möglicherweise mit nicht genau bezeichneten Nutzerdaten.

Ein Teil der in BackupAes256 verwendeten Bilder und Symbole stammt aus der Sammlung »My Secret Icons«[5] der Designerin https://artdesigner.me, der andere Teil aus der Sammlung »Dock Icons«[6] des Designers cemagraphics. Beide Autoren erlauben die nichtkommerzielle Nutzung ihrer Werke unter der Bedingung einer legalen Verwendung und der Verlinkung auf sie als Urheber.

---

3 https://dotnet.microsoft.com/download/visual-studio-sdks

4 https://visualstudio.microsoft.com/de/license-terms/mlt031819/

5 https://www.iconarchive.com/show/my-secret-icons-by-artdesigner.html

6 https://www.deviantart.com/cemagraphics/gallery/7888035/dock-icons

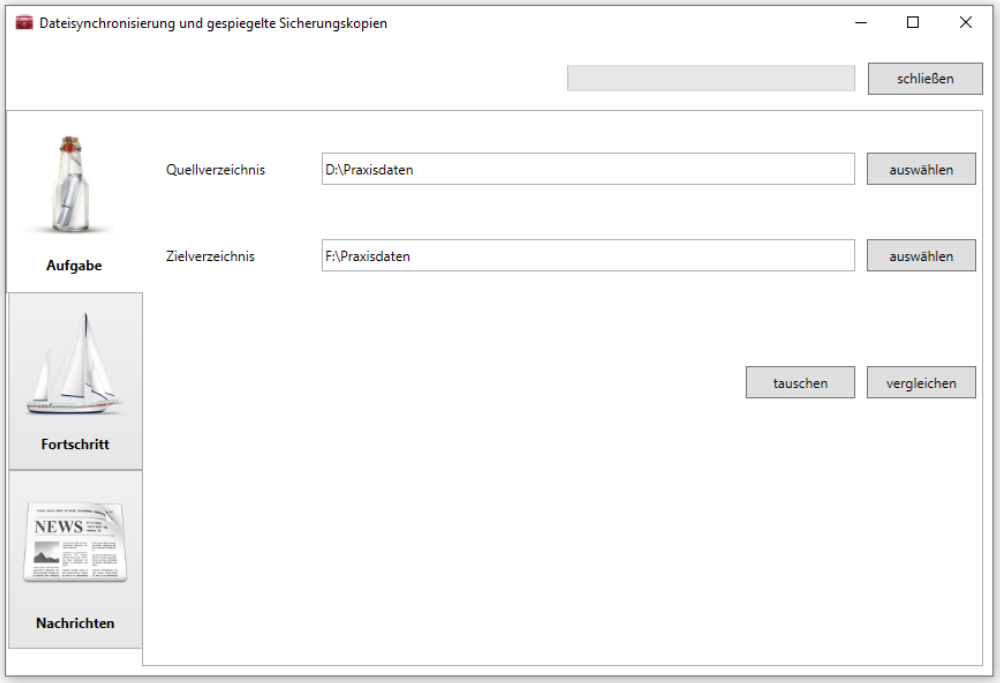
Inaktiven, aber nicht gelöschten Programmkode für die Darstellung kryptographischer Schlüssel in Dezimalschreibweise habe ich aus dem Projekt InfInt[7] übernommen. Grundlegende Funktionen des ViewModel stammen aus dem Projekt mvvmlight[8], das unter der MIT-Lizenz steht.

Auch BackupAes256 habe ich unter die MIT-Lizenz[9] gestellt.

3. Das Programm bedienen

3.1. Aufbau der Benutzeroberfläche

Am linken Rand des Programmfensters sind unter einander drei Register angeordnet: »Aufgabe«, »Fortschritt« und »Nachrichten«. Jedes dieser Register kann per Mausklick auf das entsprechende Symbol in den Vordergrund geholt werden.



Unter »Aufgabe« lassen sich Quelle und Ziel der Synchronisierung einstellen. Nur die Bearbeitung ganzer Verzeichnisse ist möglich, die Auswahl einzelner Dateien innerhalb eines Verzeichnisses ist nicht vorgesehen. Am Ende des Synchronisierungsvorgangs soll der Inhalt beider Verzeichnisse identisch sein soll.

Quell- und Zielverzeichnis können entweder über die Schaltflächen »auswählen« ausgewählt oder mit Ziehen und Ablegen aus dem Windows-Explorer übernommen werden. Sollte

7 https://github.com/sercantutar/infint

8 https://github.com/lbugnion/mvvmlight

9 https://opensource.org/licenses/MIT

beim Ziehen und Ablegen das Register »Aufgabe« nicht im Vordergrund sein, kann es aktiviert werden, indem man mit der Maus über das entsprechende Symbol zieht.

Existiert auf einem anderen angeschlossenen Datenträger ein Verzeichnis desselben Namens, wird es automatisch als zugehöriges Ziel- oder Quellverzeichnis eingetragen.

Ein Klick auf die Schaltfläche »vergleichen« startet zunächst den Vergleich beider Verzeichnisse. Dazu ermittelt BackupAes256 die im Quellverzeichnis neu dazu gekommenen Dateien sowie die Dateien mit einem neueren Bearbeitungsdatum. Dieser Schritt ändert noch keine Daten. Im Register »Fortschritt« ist zu sehen, welche Dateien synchronisiert werden sollen.

Für den Fall, dass im Quellverzeichnis Dateien gelöscht wurden, die im Zielverzeichnis noch existieren, kann hier gewählt werden, ob sie im Zielverzeichnis ebenfalls gelöscht (mit Löschen), ignoriert (kein Löschen) oder zurück in das Quellverzeichnis kopiert werden sollen (gegenseitig).

Ein Klick auf die Schaltfläche »synchronisieren« startet den Schreibvorgang. Das Register »Fortschritt« zeigt nun an, welche Dateien bisher verarbeitet wurden und für welche Dateien die Synchronisierung noch aussteht.

Das Register »Nachrichten« enthält die Versionsnummer des Programms und wird in nachfolgenden Programmversionen eventuell ausführlichere Status- und Fehlermeldungen enthalten.

3.2. Geplante Programmfunktionen

Ein verbessertes Abfangen von Fehlern und Übersetzen von Fehlerbeschreibungen zwischen Deutsch und Englisch werde ich bei Gelegenheit ergänzen. Auch die Fortschrittsanzeige ließe sich noch flüssiger gestalten.

Vielleicht werde ich das Anlegen von Synchronisierungsaufgaben ergänzen, um wiederkehrende Einstellungen zu speichern, auch für mehrere Quell- und Zielverzeichnisse gleichzeitig.

Ein nächster Aktualisierungstermin steht nicht fest.