Technische Berichte in Digitaler Forensik

Herausgegeben vom Lehrstuhl für Informatik 1 der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Kooperation mit dem Masterstudiengang Digitale Forensik (Hochschule Albstadt-Sigmaringen, FAU, Goethe-Universität Frankfurt am Main)

Registry-Spuren verursacht durch die Quantenprogrammiersprache Q#

Michael Terörde

03.03.2019

Technischer Bericht Nr. 17

Zusammenfassung

Quantencomputer könnten zukünftig für kriminelle Zwecke eingesetzt werden. Da konventionelle Speichersysteme immer Teil eines solchen Quantencomputers sind, können auch herkömmliche Methoden der Digitalen Forensik bei Untersuchungen angewandt werden. Die Quantenprogrammiersprache Q# soll das Programmieren von Quantencomputern ermöglichen und die somit erstellten Quantenprogramme sollen später mittels einen Quantencompilers auch tatsächlich auf Quantencomputer laufen. Eine Spurenmenge der Registry, die durch Q# erzeugt wird, wird erstmals in diesem Berichts vorgestellt. Ebenso wird auf forensische Untersuchungsmöglichkeiten an Quantencomputer selbst eingegangen.

Der technische Bericht ist im Rahmen des Studiengangs Digitale Forensik unter der Anleitung von Prof. Dr. Martin Rieger und Prof. Holger Morgenstern entstanden.

E-Mail-Adresse des Autors: michaelteroerde@web.de

Inhaltsverzeichnis

1 Einführung

Die fortschreitende Miniaturisierung der Bauteile in konventionellen Rechnern wird demnächst in Größenordnungen vorstoßen, bei denen Quanteneffekte nicht vernachlässigt werden können [Sch17], [Sof08]. Weiterhin ist es unterhalb einer Größenordnung von etwa 7 nm [Qutl7] beim derzeitigen Stand der Technik nicht mehr möglich, die durch Informationsverlust entstehende Wärme aus den integrierten Schaltungen abzuleiten [Sch17]. Diese beiden Herausforderungen können durch Quantencomputer gelöst werden. Darüber hinaus sollen Quantencomputer mittels neuer Algorithmen viele derzeit von konventionellen Rechnern nicht lösbare Aufgaben effizient lösen.

Erste funktionsfähige Quantencomputer mit einer geringen Anzahl von Qubits sind bereits als Prototypen erfolgreich getestet worden und der Anbieter D-Wave Systems bietet bereits einen adiabatischen Quantenannealer zum Verkauf an. Quantencomputer sollen zukünftig viele Berechnungen parallel durchführen und könnten mittels des Shor-Algorithmus derzeit verbreitete asymmetrische Verschlüsselungsverfahren brechen. Ebenso eröffnen sich unzählige Möglichkeiten in vielen Forschungsbereichen wie der Chemie, Physik, Logistik und Arzneimittelforschung.

Der Bezug zur digitalen Forensik ergibt sich aus der Tatsache, dass jeder Quantencomputer immer auch aus einem konventionellen Computer bestehen muss, der die Steuerung und das Auswerten des quantenmechanischen Experiments durchführt.

Ein Problem bei der Entwicklung des Quantencomputers ist, dass Kenntnisse über Quantenmechanik vorliegen müssen und dies häufig nur bei Physikern der Fall ist. Aus diesem Grund stammen die Wissenschaftler, die an Quantencomputern und am Quantum Computing arbeiten aus dem Physikbereich. Um diesen Bereich weiter voranzutreiben, sollten auch Wissenschaftler aus dem Bereich der Informatik an der Problemlösung arbeiten. Diese Arbeit versucht eine Brücke zwischen der Physik und der Informatik zu schlagen.

Der Schwerpunkt dieses Berichts besteht in der forensischen Analyse der Registry-Spuren, die durch die Quantenprogrammiersprache Q# erzeugt werden. Es sollen forensische Spuren dieser Software, die auf eine Installation, eine durchgeführte Deinstallation, sowie Verwendung hindeuten, detailliert untersucht werden. Es werden konkret die Fragen beantwortet, wo man die Spuren finden und wie man diese auslesen kann. Dazu wird die Zustandsmethode mit dem Programm RegShot verwendet. Die Erstellung und Analyse der Spurenmenge kann in realen forensischen Untersuchungen helfen, die persistenten und flüchtigen Spuren von Q# zu identifizieren und zu interpretieren.

Dieser Bericht untersucht erstmalig die Quantenprogrammiersprache Q# auf forensische Spuren in der Registry und verknüpft somit das Thema Quantencomputer mit der digitalen Forensik.

2 Quantencomputer und Quantenprogrammierung

Die Unternehmen IBM, Intel und Google forschen an frei programmierbaren Quantencomputern mit Quantengattern und haben laut eigenen Angeboten bereits Quantenchips mit 49 (IBM), 50 (Intel) und 72 supraleitenden Qubits (Google), siehe **Bild 2.1**.

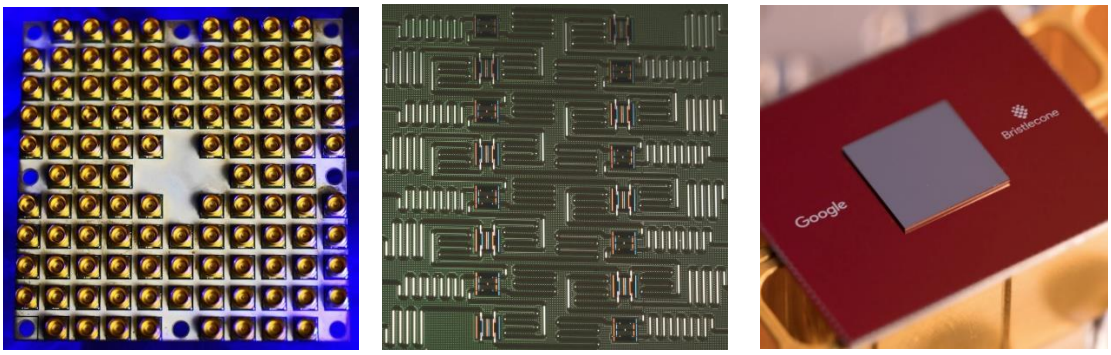

Bild 2.1 Quantenchips von IBM[1] (links, 49 Qubits), Intel[2] (Mitte, 16 Qubits) und Google[3] (rechts, 72 Qubits)

Bereits 1982 hat der Physiker Richard Feynman dargelegt, dass die zeitliche Entwicklung eines Quantensystems nicht effizient mit konventionellen Computern simuliert werden kann, da die benötigte Informationsmenge exponentiell zunimmt. Dies bedeutet, dass es umfangreiche Berechnungen erfordert um ein Mehrteilchen-Interferenz-Experiment auf Quantenniveau zu simulieren. Die Idee des Quantenrechnens ist es, diese Tatsache umzudrehen und zu schlussfolgern, dass die Durchführung des Experiments mit anschließender Messung eine komplexe Berechnung ersetzt [Sch17].

Quantencomputer und herkömmliche Computer konkurrieren im Lösen derselben Aufgaben, aber sie nutzen dazu verschiedene Naturgesetze [Cirl6]. Mit Quantencomputern können im Prinzip auch alle herkömmlichen Aufgaben konventioneller Rechner gelöst werden, dabei sind sie aber heutigen Computern nicht überlegen [Spe12]. Im Gegenteil wären Quantencomputer langsamer als herkömmliche Rechner, bezogen auf die Geschwindigkeit einzelner Rechenschritte. Nur bei Verwendung der besonderen quantenmechanischen Phänomene der Superposition und der Verschränkung ergeben sich Vorteile [Bez07].

Aufgaben, die Quantencomputer besser lösen können sollen, umfassen das Faktorisieren großer Zahlen, das Lösen von Optimierungsproblemen im Zusammenhang mit Mustererkennung und maschinellem Lernen, das Designen chemischer Verbindungen, wodurch neue Materialien wie

[1] Bildquelle: https://www.rankred.com/quantum-processors/ (Abruf am 31.07.2018)

[2] Bildquelle: https://www.nextplatform.com/2018/01/10/quantum-computing-enters-2018-like-1968/ (Abruf am 31.07.2018)

[3] Bildquelle: https://www.technologyreview.com/the-download/610423/google-has-built-the-worlds-most-advanced-quantum-chip/ (Abruf am 31.07.2018)

Hochtemperatur-Supraleiter ermöglicht werden sollen und die Suche in unstrukturierten Datenbanken.

Erstmals fand Peter Shor 1994 einen Quantenalgorithmus der ein praktisches Problem löst. Er zeigte wie eine n-stellige Zahl faktorisiert werden kann und die Anzahl der Rechenschritte nur mit Polynomialzeit zunimmt. Mit hoher Wahrscheinlichkeit ist das Faktorisieren von Zahlen aber kein NP-vollständiges Problem. Somit kann man nicht davon ausgehen, dass Quantencomputer NP-vollständige Probleme lösen werden können.

Ein Quantencomputer kann die drei bekannten Probleme

- das Integer-Faktorisierungsproblem (IFP),
- das diskrete Logarithmus-Problem (DLP) und
- das Elliptische-Kurven-Diskrete-Logarithmus-Problem (ECDLP)

effizient in Polynomialzeit lösen [Yan13], wohingegen diese Probleme bei binär arbeitenden Rechnern als praktisch nicht lösbar gelten. Daraus lässt sich aber nicht folgern, dass Quantencomputer alle Probleme effizient lösen und es kein sicheres kryptographisches Verfahren gibt. Bei anderen NP-schweren Problemen wie dem Problem des Handlungsreisenden (Traveling Salesman Problem) und beim kürzesten Gitter-Problem (Shortest Lattice Problem) ist die Leistungsfähigkeit eines Quantencomputers wahrscheinlich dieselbe wie bei klassischen Computern, da kein Quantenalgorithmus bekannt ist, der NP-Vollständige Probleme effizient lösen könnte [Yan13], [Spe12].

2.1 Gefährdete Verschlüsselungs-Verfahren

Die am meisten verwendeten Public-Key Verfahren heutzutage sind Rivest–Shamir–Adleman (RSA), Diffie-Hellman (DH) und Elliptische-Kurven-Kryptografie (Elliptic Curve Cryptography, ECC). Die Sicherheit dieser Verfahren basiert auf der Annahme, dass das Integer-Faktorisierungsproblem und das diskrete Logarithmus-Problem nicht effizient gelöst werden können [Jod14]. Da diese Annahme durch die Existenz eines Quantencomputers mit ausreichend Quantenbits ungültig wird, sind diese Verfahren zukünftig unsicher. Damit einhergehend ist auch die Kommunikation mittels HTTPS, SSH und VPNs nicht sicher. Ebenso können mit PGP und GPG verschlüsselte E-Mails entschlüsselt werden.

Die Kryptowährung Bitcoin ist ebenfalls durch Quantencomputer gefährdet [Gie16]. Herkömmliche Rechner benötigen 2^{128} Operationen (etwa $3{,}4 \cdot 10^{38}$) um aus dem öffentlichen Bitcoin-Schlüssel den privaten Bitcoin-Schlüssel zu berechnen, wodurch ein Angriff praktisch ausgeschlossen ist. Ein Quantencomputer würde mittels des Shor-Algorithmus nur 128^3 Operationen (etwa 2,1 Mio.) benötigen [Bitl8]. Die Berechnung einer Zahl dessen Hashwert SHA-256 vorgegeben ist, benötigt 2^{256} Operationen auf konventionellen Computern. Ein Quantencomputer kann mittels des Grover-Algorithmus schneller in 2^{128} Operationen berechnen, was allerdings ebenfalls unpraktisch viele Operationen sind. Das Bitcoin-Mining, welches im Wesentlichen ein Angriff gegen symmetrische Kryptographie darstellt, kann also mit weniger Operationen ausgeführt werden.

2.2 Quantenprogrammierung

Es sind derzeit etwa 50 Quantenalgorithmen bekannt, die quantenmechanische Effekte verwenden und eine geringere Komplexität nach der 0-Notation aufweisen als entsprechende klassische Algorithmen [Jor18]. Bisher konnten bei drei Quantenalgorithmen nachgewiesen werden, dass sie Probleme in Polynomialzeit lösen, die klassisch exponentielle Rechenzeit erfordern [Köh01]. Von diesen drei Algorithmen von Deutsch, Simon und Shor scheint nur der Shor-Algorithmus von praktischer Bedeutung zu sein. Mit diesem Algorithmus kann ein Quantencomputer also die heute verwendeten Public-Key-Kryptographieverfahren in Minuten entschlüsseln [Köh01]. Da der Shor-Algorithmus das Quanten-Parallelrechnen voll ausnutzt, ist ein Quantencomputer mit einem Register von 10 logischen Qubits 2^{10} = (etwa)10^3-mal schneller als ein klassischer Computer.

Die Simulation eines Quantenrechners mit *n* Qubits bedeutet im Wesentlichen den Umgang mit 2^nx2^n Matrizen über den komplexen Zahlenraum $\mathbb{C}$ [Rüd05]. In Verbindung mit Simulatoren kann man somit unter zu Hilfenahme von Quantenprogrammiersprachen bereits heutzutage Programme für zukünftige Quantenrechner entwickeln und Quantenalgorithmen zumindest mit kleinen Inputmengen testen.

Um die Hardware korrekt zu modellieren, wird eine Master-Slave-Architektur zugrunde gelegt. Der Master ist ein konventioneller Rechner der über einen Quantencompiler, auch Quantum Device Driver genannt, das parametrisierte physikalische Quantenexperiment, steuert [Rüd05]. Das in einer Quantenprogrammiersprache formulierte Programm läuft auf dem Master ab, der die am Experiment durchgeführte Messung als Messergebnis zurückgesendet bekommt.

Physikalisch betrachtet ist der Ablauf eines Algorithmus auf dem Quantenanteil eines Quantencomputers ein Experiment an einem Quantensystem mit einer initialen Zustandspräparation und einer abschließenden Messung des Zustandes [Rüd05]. Die zeitliche Zustandsentwicklung zwischen Anfang und Messung wird durch unitäre Transformationen im Raum der Zustände, beschrieben. Die Quantenprogrammiersprachen müssen diese theoretischen Grundlagen bei der Implementierung beachten.

Nach Ömer müssen nützliche Quanten-Programmiersprachen folgende Charakteristika aufweisen [Aka15]:

1. Unabhängig von der Hardware-Realisierung
2. Beliebige Abstraktionsebenen sind verfügbar
3. Nicht-klassische Eigenschaften werden auf semantischer Ebene integriert

Während die ersten beiden Forderungen allgemein für Programmiersprachen gelten, gilt die dritte Forderung speziell für Quanten-Programmiersprachen. Danach sollen Quantenberechnungen in der uns bekannten konzeptionellen Weise ausgedrückt werden, um die Programmierung zu ermöglichen.

Weitere Besonderheiten von Quanten-Programmiersprachen im Quantenanteil sind [Aka15]:

- Umkehrbarkeit von unitären Operationen

- Nicht-Lokalität von Quantenbits
- Nicht-Beobachtbarkeit von Zuständen
- Destruktive Natur von Messungen
- Das Fehlen von Löschoperationen

Microsoft hat im Dezember 2017 die neue Programmiersprache Q# der Öffentlichkeit vor- und zur Verfügung gestellt. Diese ist Teil des Quantum Development Kits, das auf Visual Studio 2017 läuft und neben Q# auch einen Quantencomputer-Simulator sowie ergänzende Ressourcen für die Programmierung bereitstellt [Par17]. Es handelt sich um eine .NET Sprache eingebettet in Visual Studio 2017.

3 Forensische Untersuchungsmethoden

Die Untersuchungen wurden auf einem Laptop unter zu Hilfenahme der Virtualisierungssoftware von VirtualBox durchgeführt. Vor Beginn der Untersuchungen wurde mittels des Anti-Viren-Programms *Symantec Endpoint Protection* sichergestellt, dass das System virenfrei ist.
Auf dem Notebook wurde mittels Oracle VirtualBox eine Virtuelle Maschine (VM) mit dem Betriebssystem Windows 10 64-Bit errichtet, sodass das Notebook als Host agiert. Diese virtuelle Maschine, das sogenannte Gastsystem, wird genutzt um zu untersuchen, welche persistenten Spuren der Einsatz von Q# in der Registry erzeugt.

Die verwendeten Programme sind in **Tabelle 1** zusammengefasst.

Tabelle 1 Verwendete Programme und deren Einsatzzweck

Software	**Verwendet für**
Oracle VirtualBox Version 5.1.14	Virtuelle Maschine mit dem Betriebssystem Windows 10 für die Untersuchungen
PowerGui Script Editor v3.8.0.129	Für PowerShell-Skript zur Erstellung von Snapshots
Visual Studio Community 2017 Version 15.7.4	Die zu analysierende Programmierumgebung
Microsoft Edge	Browser zum Herunterladen der Installations-Dateien
RegShot 1.9.0 x64 ANSI	Zur Analyse der Registry-Änderungen
Registrierungs-Editor (Regedit) v6.1	Zur Untersuchung der Registry
HashMyFiles v2.30	Zur Protokollierung von SHA-256-Hashwerten

Der Ablauf der forensischen Untersuchungen wird, wie in **Tabelle 2** beschrieben, in unterschiedliche Phasen unterteilt.

Tabelle 2 Übersicht der zu untersuchenden Phasen

Lfd. Nr.	**Aktion/Phase**	**Kurz-Beschreibung**
1	Installation von Visual Studio 2017 Community und .NET Core SDK	Installation von Visual Studio 2017 Community mittels des Installationsprogramms *vs_community__ 1527698340. 1530114608*. Es wird als spezielle Workload *.NET Core cross-platform development* ausgewählt und ansonsten werden die Standardeinstellungen verwendet.
2	Installation des Quantum Development Kits	Die Installation des Quantum Development Kits erfolgt über das Installationsprogramm *QsharpVSIX*
3	Bibliotheken und Beispiele von GitHub laden	Bibliotheken und Beispiele des Microsoft Quantum Development Kit werden von GitHub[4] in Visual Studio integriert. Dazu wird die URL *https://github.com/Microsoft/Quantum.git* bei Local Git Repositories in Visual Studio eingetragen. Damit ist es auf der lokalen Festplatte geklont.
4	Ausführung eines Quantenprogrammes	Das Programm *Teleportation* wird als Startprojekt festgelegt und gestartet.
5	Deinstallation von Visual Studio	Deinstallation von Visual Studio 2017, Visual Studio Installer sowie Microsoft .NET Core SDK – 2.1.201 mittels der Windows-Funktion *Programme deinstallieren* in der Systemsteuerung

Es wird RegShot für die Zustandsmethode verwendet, um Veränderungen in der Registry zu erkennen. Hier wird vor und nach der Aktion mittels RegShot ein Registry-Abbild erstellt und anschließend auf Änderungen untersucht. Der entstehende Bericht wird als Textdatei (.txt) gespeichert.

Die Hive-Kopien, die von RegShot angelegt werden, werden alle gespeichert und später miteinander verglichen, um Änderungen zu identifizieren. Dadurch kann auch die Registry vor der Installation von Visual Studio mit der Registry nach der Deinstallation-Phase verglichen werden. Ebenfalls können zusätzlich zu den Registry-Änderungen, die der Deinstallations-Prozess selbst auslöst, auch Spuren gefunden werden, die auf eine frühere Installation deuten. Es wird *RegShot 1.9.0 x64 Ansi* verwendet, wobei der standardmäßig eingestellte Ausgabepfad *C:\Users\besitzer\AppData\Local\Temp* geändert werden musste, um ein fehlerfreies Erstellen des Reports zu ermöglichen.

[4] Vollständige Internetseite: https://github.com/Microsoft/Quantum

4 Spuren in der Registrierungs-Datenbank

Die Registrierungs-Datenbank (kurz: Registry) ist die zentrale hierarchische Konfigurations-datenbank von Windows und enthält Informationen zu installierten Programmen und zu Windows selbst. Da eine Manipulation dieser Spuren PC-Kenntnisse voraussetzt, gelten diese Spuren als robuste und persistente Spuren und sind von forensischem Interesse. Die Software RegShot Version 1.9.0 x64 ANSI kann ein Abbild der Windows-Registrierungs-Datenbank (kurz: Registry) erstellen und ermöglicht das Abspeichern als .hiv oder .hivu Datei. Zusätzlich berechnet RegShot alle Änderungen von zwei Registry-Abbildern durch den Befehl *compare* und speichert diese als Text- oder HTML-Bericht (siehe **Bild 4.1**).

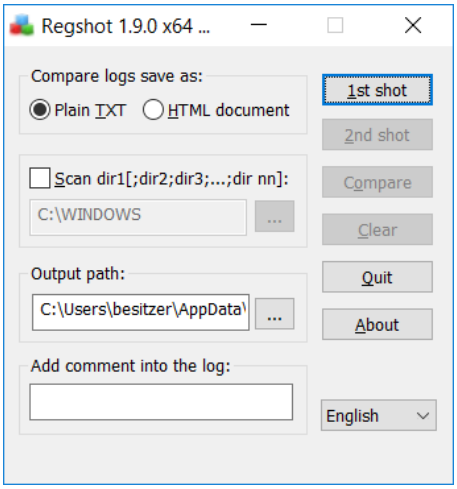

Bild 4.1 Benutzeroberfläche von RegShot 1.9.0 x64 ANSI

Der Vergleichs-Bericht erzeugt eine Auflistung nach dem Schema:

- Schlüssel entfernt (Keys deleted)
- Schlüssel hinzugefügt (Keys added)
- Werte entfernt (Values deleted)
- Werte hinzugefügt (Values added)
- Werte modifiziert (Values modified)
- Anzahl Änderungen (Total changes)

Die hier verwendeten Spuren sind entweder erzeugte Schlüssel (Keys added), erzeugte Werte (Values added) oder modifizierte Werte (Values modified). Damit Registry-Änderungen, die vom Betriebssystem erzeugt wurden, nicht in der Spurenmenge von Q# erfasst werden, wird ein RegShot-Differenzbild erstellt, bei dem keine Aktion ausgeführt wurde. Die durch dieses Leerlauf-Bild erkannten Spuren werden aus der Spurenmenge von Q# herausgenommen.

Die entstehenden RegShot-Berichte im Textformat sind zwischen 932 kB und 70 MB groß[5] und enthalten bis zu 300.000 ermittelte Änderungen. Damit aus diesen riesigen Datenmengen Spuren extrahiert werden können, wird eine strukturierte Stringsuche nach bestimmten Schlüsselbegriffen verwendet. Dieses einheitliche Vorgehen für alle Phasen bei der Analyse der Registry-Berichte berücksichtigt die unterschiedlichen notwendigen Programme für Q#. Es wird für jede Phase jeweils eine Auswahl von Spuren vorgestellt. Die Suchbegriffe für den jeweiligen RegShot-Report sind:

- Visual Studio, Visualstudio
- .NET Core SDK, .NETCore, dotnet
- F#, Fsharp
- Quantum, Q#, Qsharp, VSIX
- Github, git.exe
- Teleportation

4.1 Phase 1: Installation Visual Studio und .NET Core SDK

Die Installation von Visual Studio in Verbindung mit .NET Core SDK erzeugt persistente Spuren in der Registry. Es sind eindeutige Spuren auf die Installationen von Visual Studio, .NET Core SDK, F# und dotnet.exe zu finden. In **Tabelle 3** sind hinzugefügte Schlüssel und Werte aufgelistet, die durch eine Deinstallation entfernt werden und somit eine installierte Visual Studio Umgebung nachweisen.

Tabelle 3 Charakteristische Registry-Spuren für Visual Studio und .NET Core SDK

Auswahl hinzugefügter Schlüssel	• HKLM\SOFTWARE\Classes\AppID\devenv.exe\: "Microsoft Visual Studio" • HKLM\SOFTWARE\Classes\Installer\Dependencies\DotNet.CLI.SharedFramework.Microsoft.NETCore.App_2.0.7_x64 • HKLM\SOFTWARE\Classes\Installer\Products\36CB2C782C280E647A78B5E8279EAC67\ProductName: "Visual F# 10.1 SDK"
Auswahl hinzugefügter Werte	• HKLM\SOFTWARE\Classes\.vsix\: "VisualStudio.Launcher.vsix" • HKLM\SOFTWARE\Classes\Installer\Dependencies\Microsoft.VisualStudio.Community.Msi,v15\Version: "15.7.27617" • HKLM\SOFTWARE\Classes\Installer\Products\C7737F95A21003D46A85C3B379BEBE5E\SourceList\PackageName: "dotnet-runtime-2.0.7-win-x64.msi" • HKLM\SOFTWARE\Classes\Microsoft.VisualStudio.Setup.Configuration\: "Microsoft Visual Studio Setup Configuration" • HKLM\SOFTWARE\dotnet\Setup\InstalledVersions\x64\sharedhost\Version: "2.0.7" • HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folders\C:\Program Files\dotnet\sdk\2.1.300\FSharp\: ""

[5] RA.txt 70 MB; RB.txt 4 MB; RC.txt 932 kB; RD.txt 1.740 kB, RE.txt 19 MB, RF.txt 63 MB; RG.txt 25 MB

4.2 Phase 2: Quantum Development Kit

Während der Phase 2 Installation des Quantum Development Kit wurden in der Registry 3.693 Schlüssel gelöscht und 4.709 neue erstellt. Es wurden 8.542 Werte entfernt, 14.480 neue erzeugt und 1.471 Werte modifiziert. Die meisten dieser Registry-Änderungen wurden durch Hintergrundprozesse des Betriebssystems ausgelöst. Die Installation des QDK erzeugt wenig neue Schlüssel und Werte im Hinblick auf die Suchbegriffe. Charakteristische Spuren wurden durch den String VSIX gefunden. Eine Stringsuche nach den Begriffen Quantum, Q# und Qsharp ergab keine Treffer. Die in **Tabelle 4** gezeigten Spuren sind alle auch nach einer Deinstallation vorhanden.

Tabelle 4 Charakteristische Registry-Spuren durch QDK-Installation

Auswahl hinzugefügter Schlüssel	• HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.vsix
Auswahl hinzugefügter Werte	• HKLM\SYSTEM\ControlSet001\Services\bam\UserSettings\S-1-5-21-111732601-3564288977-3518119001-1001\\Device\HarddiskVolume2\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\VSIXInstaller.exe: A7 B9 10 DF 37 0E D4 01 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 • HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\VisualStudio\Telemetry\PersistentPropertyBag\vsixinstaller\VS.TelemetryApi.ChannelsDisposeLatency: 0x0000000E • HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\Windows\CurrentVersion\ApplicationAssociationToasts\VisualStudio.Launcher.vsix_.vsix: 0x00000000

4.3 Phase 3: GitHub laden

Durch die Phase 3 werden zwei Spuren in Form von hinzugefügten Werten in der Registry erzeugt. Dabei wird das Argument *C:\Users\MT\source\repos\Quantum* als Eintrag beim Visual Studio Start hinzugefügt sowie der erscheinende Name *Quantum* (siehe **Tabelle 5**). Diese Änderungen führen zu Einträgen auf der Startseite von Visual Studio wie in **Bild 4.2** zu sehen.

Tabelle 5 Charakteristische Registry-Spuren

Auswahl hinzugefügter Werte	• HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{E1AFAD38-CD28-4A67-95BE-787DA75CF53D}\RecentItems\{CD5D2D2D-6BC3-43DF-ACD1-94C69FF775E7}\Arguments: ""C:\Users\MT\source\repos\Quantum" source:JumpList" • HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{E1AFAD38-CD28-4A67-95BE-787DA75CF53D}\RecentItems\{CD5D2D2D-6BC3-43DF-ACD1-94C69FF775E7}\DisplayName: "Quantum"

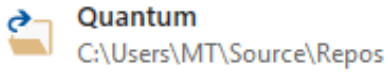

Bild 4.2 Neuer Eintrag auf der Startseite von Visual Studio

4.4 Phase 4: Quantenalgorithmus

Das Ausführen des Programms TeleportationSample löscht und erzeugt keine Q#-spezifischen Schlüssel oder Werte. Somit erzeugt das Ausführen von Quantenprogrammen keine Spuren in der Registry.

4.5 Phase 5: Deinstallation

Die Deinstallation entfernt viele Schlüssel und Werte, die sich auf Visual Studio, .NET Core und F# beziehen, wie in **Tabelle 6** zu sehen. Es werden keine neuen Schlüssel oder Werte erzeugt, die Spuren darstellen.

Tabelle 6 Auswahl gelöschter Schlüssel und gelöschter Werte

Auswahl gelöschter Schlüssel	• HKLM\SOFTWARE\Classes\.vsix\OpenWithProgids • HKLM\SOFTWARE\Classes\Applications\VSLauncher.exe\Shell\Open\Command • HKLM\SOFTWARE\Classes\Installer\Assemblies\C:|Program Files (x86)|ReferenceAssemblies|Microsoft|Framework|.NETFramework|v4.5|Microsoft.VisualC.STLCLR.dll • HKLM\SOFTWARE\Classes\Installer\Dependencies\DotNet.CLI.SharedFramework.Microsoft.NETCore.App_2.0.7_x64 • HKLM\SOFTWARE\Microsoft\.NETFramework\v2.0.50727\NGenService\Roots\C:/Program Files (x86)/Microsoft SDKs/F#/10.1/Framework/v4.0/FSharp.Build.dll • HKLM\SOFTWARE\Classes\.vsix\: "VisualStudio.Launcher.vsix"
Auswahl gelöschter Werte	• HKLM\SOFTWARE\Classes\.androidproj\OpenWithProgids\VisualStudio.Launcher.androidproj.90c3ce82: "" • HKLM\SOFTWARE\Classes\.svclog\: "VisualStudio.SvcLog.10"

	• HKLM\SOFTWARE\Classes\Installer\Products\57E5A60D2DEC91B4E980EE1B8CF4420D\ProductName: "Visual F# 10.1 SDK" • HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{C5C91AA6-3E83-430E-8B7A-6B790083F28D}\URLUpdateInfo: "http://www.microsoft.com/windowsazure/sdk" • HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{d27a4039-4055-49f4-931d-8373d9449e3d}\URLInfoAbout: "http://dotnet.github.io/"

Um nach einer Deinstallation nachzuweisen, dass eine vorherige Installation von Visual Studio und .NET Core SDK vorlag, sind die hinzugefügten Schlüssel und Werte entscheidend, die durch eine Deinstallation nicht entfernt werden. Mittels RegShot wird das Abbild vor der Installation (Registry-Dateien R0) mit dem Abbild nach der Deinstallation (Registry-Dateien R5) verglichen. Die Analyse zeigt, dass mehrere Registry-Keys, die bei der Installation erstellt werden auch nach der Deinstallation erhalten bleiben (siehe **Tabelle 7**).

Tabelle 7 Charakteristische Registry-Spuren für eine Visual Studio-Installation

Auswahl hinzugefügter Schlüssel	• HKLM\SOFTWARE\Classes\.vsixmanifest • HKLM\SOFTWARE\Classes\Installer\Dependencies\Microsoft.AspNet.DiagnosticPack_VisualStudio15_enu,v15 • HKLM\SOFTWARE\Classes\Installer\Dependencies\Microsoft.FSharp.SDK.Core,v10.1 • HKLM\SOFTWARE\Classes\Installer\Dependencies\Microsoft.VS.NETCoreSDK,v15 • HKLM\SOFTWARE\Microsoft\.NETFramework\v2.0.50727\NGenService\Roots\C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/Common7/IDE/CommonExtensions/Microsoft/GettingStartedTemplates/Resources/Microsoft.VisualStudio.GettingStartedTemplates.Resources.dll
Auswahl hinzugefügter Werte	• HKLM\SOFTWARE\Classes\.settings\: "VisualStudio.settings.2063b37f" • HKLM\SOFTWARE\Classes\CLSID\{26933B26-DA32-49FC-B31F-02BACE3A497D}\InprocServer32\: "C:\Program Files\Common Files\Microsoft Shared\VS7Debug\pdm.dll" • HKLM\SOFTWARE\Classes\Installer\Dependencies\Microsoft.FSharp.SDK.Core,v10.1\Version: "10.1" • HKLM\SOFTWARE\Classes\WOW6432Node\CLSID\{27E23E49-0BAD-437C-9F4A-F1F8682535CA}\LocalServer32\: ""C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\common7\ide\devenv.exe"" • HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folders\C:\Program Files\dotnet\sdk\: ""

Spuren, die auf eine Installation und Verwendung von QDK hindeuten, sind durch das Differenzbild zwischen den Registry-Dateien R1 und R5 detektierbar. Charakteristische

Registry-Spuren, die auch nach der Deinstallations-Phase auf eine frühere Verwendung von QDK hindeuten, sind in **Tabelle 8** aufgeführt. Dabei handelt es sich um die zwei Spuren, die bereits in **Unterabschnitt 0** erwähnt wurden.

Tabelle 8 Charakteristische Registry-Spuren für eine QDK-Installation

Auswahl hinzugefügter Werte	• HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\ Windows\CurrentVersion\Search\ RecentApps\{E1AFAD38-CD28-4A67-95BE-787DA75CF53D}\RecentItems\{CD5D2D2D-6BC3-43DF-ACD1-94C69FF775E7}\Arguments: ""C:\Users\MT\source\repos\Quantum" source:JumpList" • HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\ Windows\CurrentVersion\Search\ RecentApps\{E1AFAD38-CD28-4A67-95BE-787DA75CF53D}\RecentItems\{CD5D2D2D-6BC3-43DF-ACD1-94C69FF775E7}\DisplayName: "Quantum"

Liegt z.B. der Registry-Wert *HKU\S-1-5-21-111732601-3564288977-3518119001-1001\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{E1AFAD38-CD28-4A67-95BE-787DA75CF53D}\RecentItems\{CD5D2D2D-6BC3-43DF-ACD1-94C69FF775E7}\Arguments: ""C:\Users\MT\source\repos\Quantum" source:JumpList"* vor, hat man eine robuste Spur für das Herunterladen des quantum.git von der GitHub.Seite. Eine weitere eindeutige Spur ist der Registry-Schlüssel *HKLM\SOFTWARE\Classes\.vsixmanifest* (siehe **Bild 4.3**).

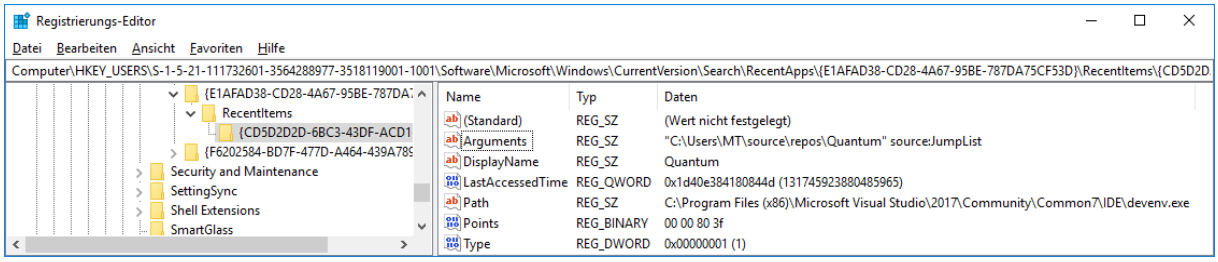

Bild 4.3 Robuste Spuren in der Registry für die Deinstallation

5 Forensische Spuren bei Quantencomputern

Nach der Vorstellung der Ergebnisse für digitale Spuren werden in diesem Kapitel andere forensische Spuren bei Quantencomputern diskutiert, die für Forensiker in bestimmten Szenarien hilfreich sein könnten. Der quantenmechanische Anteil eines Quantencomputers liefert keine persistenten Spuren während der Laufzeit, da jeder Versuch, Informationen aus dem System zu erhalten, eine Wechselwirkung erfordert und somit die Berechnungen verfälscht. Denkbar ist, dass das Endergebnis vor Ablauf der Dekohärenzzeit ausgelesen wird, da dieses sich durch eine bereits durchgeführte Messung nicht mehr verändert.

Bei einer Implementierung eines Quantencomputers mittels Photonen müssen die Quantengates physisch in Form von optischen Bauteilen wie Spiegeln und Strahlteilern aufgebaut werden. Eine Untersuchung des Aufbaus könnte Hinweise auf den beabsichtigten Quantenalgorithmus liefern. Ein solcher Aufbau mit dynamischen Qubits mittels Photonen für zukünftige frei programmierbare Quantencomputer ist aber unwahrscheinlich. Die wahrscheinlicheren Realisierungen mit stationären Qubits aus Supraleitern, Ionenfallen und Fehlstellen in Diamanten sind universell für Quantenalgorithmen einsetzbar, weshalb der Aufbau keine Spuren auf bestimmte Programme liefern kann.

6 Fazit

Es wird derzeit in vielen Ländern viel Forschungsgeld für die Entwicklung von skalierbaren Quantenbits investiert. Wird eine einfach zu skalierende Technik gefunden, ist es wahrscheinlich, dass universelle, frei programmierbare Quantencomputer eingesetzt werden können. Erreichen diese eine Größe von 60 logischen Quantenbits, befinden wir uns im Zeitalter des Quanten-Vorteils, da die Rechenfähigkeit dieser Quantencomputer für bestimmte Problemklassen dann allen herkömmlichen Computern überlegen ist. In diesem Szenario werden Quantenprogrammiersprachen entwickelt und verwendet, um die Quantencomputer zu programmieren. Die derzeit verfügbare Quantenprogrammiersprache Q# soll laut Microsoft für zukünftige Quantencomputer verwendet werden können.

Aus physikalischer Sicht ist der Ablauf eines Algorithmus auf einem Quantenrechner ein Experiment an einem Quantensystem. Ein Quantencomputer rechnet aber nicht pauschal schneller als ein herkömmlicher Rechner [Mei15]. Jeder Quantencomputer beinhaltet einen klassischen Rechner, der das quantenmechanische Experiment an den Qubits steuert und das Quantenprogramm beinhaltet. Somit hinterlässt auch ein Quantencomputer digitale Spuren in den Speichern des Rechners.

Die Quantenprogrammiersprache Q# erzeugt robuste und persistente Spuren in der Registry bei der Installation des Quantum Development Kits, des Ladens der Programmbibliotheken von GitHub und bei der Deinstallation. Das Ausführen von Quantenprogrammen erzeugte keine Spuren in der Registry.

Der quantenmechanische Teil eines Quantencomputers wird keine persistenten Spuren hinterlassen. Als flüchtige Spur kann das Ergebnis vorhanden sein, bei den meisten Technologien ist dieses jedoch nur für Zeiträume unter 1 s stabil und im Anschluss nicht mehr auslesbar. Sollte ein Quantencomputer basierend auf Photonen verwendet werden, müssen diese eine Strecke durchlaufen, wo sie ihre Quantengatter-Manipulationen erfahren. In einem solchen Fall können durch Reverse Engineering der Strecke Hinweise auf das Quantenprogramm gewonnen werden.

Wichtige Aussagen dieses Berichts sind:

1. Frei programmierbare Quantencomputer können zukünftig für kriminelle Zwecke eingesetzt werden, wenn eine skalierbare Technik für Quantenbits gefunden wird. Um dafür gewappnet zu sein, werden im Rahmen dieser Arbeit erstmalig die Themen Quantencomputer und digitale Forensik verknüpft.

2. Es ist aufgrund des No-Cloning-Theorems nicht möglich, Informationen aus den Quantenbits während einer Berechnung eines Quantencomputers zu erhalten. Die einzige Möglichkeit, bei stationären Quantenbits Informationen zu erhalten, ist das Auslesen des Endergebnisses innerhalb der kurzen Kohärenzzeit. Bei dynamischen Quantenbits aus Photonen könnte der Aufbau selbst Hinweise auf das durchgeführte Programm liefern.

3. Ein Quantencomputer besteht neben den Quantenbits immer auch aus einem konventionellen Rechner und herkömmlichen Speichertechnologien. Somit können Methoden aus der digitalen Forensik wie die Datenträgerforensik und Live-Analyse angewendet werden.

4. Die Programmierung von Quantencomputern erfordert Programmierplattformen und Programmierumgebungen, die für Quantenprogrammiersprachen wie Q# ausgelegt sind. Diese Programme erzeugen digitale Spuren im Dateisystem, in der Registry und im Arbeitsspeicher.

7 Literaturverzeichnis

[Aka15] Akama, Seiki: Elements of Quantum Computing -History, Theories and Engineering Applications, Springer Verlag, ISBN 978-3-319-08283-7, 2015

[Bez07] Bezold, Matthias: Eine Einführung zum Thema Quantencomputer, 2007
Internetquelle: www.mbezold.de/quantencomputer/pdf/quantencomputer.pdf (Abruf am 20.06.2018)

[Bit18] Internetseite: https://en.bitcoinit/wiki/Quantum_computing_and_Bitcoin (Abruf am 20.06.2018)

[Cir16] Cirac, Juan: Quantencomputer wird es geben, Zeitschrift: Physik unserer Zeit, Wiley-VCH Verlag, Februar 2016

[Gie16] Giechaskiel, I., Cremers, C., Rasmussen, K.: On Bitcoin Security in the Presence of Broken Crypto Primitives, In Computer Security - ESORICS 2016, Lecture Notes in Computer Science, vol. 9879, Springer Verlag, ISBN 978-3-319-45741-3, veröffentlicht am 19.02.2016
Internetquelle: http://diyhpl.us/--bryan/papers2/bitcoin/On%20BitcoincY020security%20in%20the%20 presence% 20of%20broken%20crypto %20primitives%20-%202016.pdf (Abruf am 06.05.2018)

[Jod14] Jodoin, Eric: Straddling the Next Frontier Part 2: How Quantum Computing has already begun impacting the Cyber Security landscape, SANS Institute, veröffentlicht 09.08.2014

[Jor18] Jordan, Stephan: Algebraic and Number Theoretic Algorithms, 2018 Internetquelle: https://math.nist.gov/quantum/zoo/ (Abruf am 18.06.2018)

[Köh01] Köhler, Steffen: Sympathetisches Kühlen als Anwendung für den Ionenfallen-Quantencomputer, Dissertation der Ludwig-Maximillians-Universität München, 11.05.2011

[Mei15] Meier, Christian: Eine kurze Geschichte des Quantencomputers (TELEPOLIS) Taschenbuch — 2. April 2015

[Par17] Parbel, Matthias: Q#: Microsofts Development Kit für Quantencomputing mit eigener Programmiersprache und Simulator, veröffentlicht am 12.12.2017
Internetquelle: https://www.heise.de/developer/meldung/Q-Microsofts-Development-Kit-fuer-Quantencomputing-mit-eigener-Programmiersprache-und-Simulator-3915895.html (Abruf am 08.05.2018)

[Qut17] Qutega, BMBF: QUANTENTECHNOLOGIE Grundlagen und Anwendungen, Konzeptpapier der Nationalen Initiative zur Förderung
der Quantentechnologie von Grundlagen bis Anwendungen (QUTEGA)

[Rüd05] Rüdiger, R.: Quantenprogrammiersprachen, DOI 10.1007/s00287-003-0350-0, Springer Verlag, Gesellschaft für Informatik, veröffentlicht am 28.07.2005
Internetquelle: https://gi.de/informatiklexikon/quantenprogrammiersprachen/ (Abruf am 19.06.2018)
Siehe auch
https://link.springer.com/article/10.1007/s00287-003-0350-0

[Sof08] Sofge, D.: A Survey of Quantum Programming Languages: History, Methods, and Tools, Proceedings of the Second International Conference an Quantum, Nano, and Micro Technologies (ICQNM 2008), D01: 10.1109/ICQNM.2008.15, 2008, Seite 66-71m IEEE Computer Society, pp. 66-71, 2008, Internetquelle: https://pdfs.semanticscholar.org/ad58/9535382ac00d91e553d356618c312584 9603.pdf (Abruf am 14.06.2018)

[Spe12] Spektrum der Wissenschaft: Quanteninformation -Teleportation - Kryptografie - Quantencomputer (Spektrum Highlights) Broschiert, ISSN: 0947-7934, 1/2012, 28. Februar 2012

[Yan13] Yang, Song: Quantum Attacks on Public-Key Cryptosystems, Springer Verlag, ISBN 978-1-4419-7721-2, 2013