technische universität
dortmund

Seminararbeit

CrypTool 2 – Visuelle Programmierung / Visualisierung von Algorithmen

Jan Goclik

30. Januar 2014

Gutachter:

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund
http://www-jj.cs.uni-dortmund.de/secse

Jan Goclik
jan.goclik@udo.edu
Matrikelnummer: 147756
Studiengang: Bachelor Informatik

Werkzeugunterstützung für sichere Software
Thema: CrypTool 2 – Visuelle Programmierung / Visualisierung von Algorithmen

Eingereicht: 30. Januar 2014

Betreuer: Sebastian Pape

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dortmund, den 30. Januar 2014

Jan Goclik

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Was ist CrypTool 2 ?

CrypTool 2 ist ein Open-Source-Programm mit dem man kryptographische Verfahren simulieren und untersuchen kann. Dafür stellt CrypTool 2 verschiedene Funktionen zur Verfügung. In dieser Ausarbeitung beschäftige ich mich mit zwei dieser Funktionen und erläutere aus meiner Sicht inwieweit diese zur Werkzeugunterstützung für sichere Software beitragen.

Die erste Funktion ist die Visuelle Programmierung mit der man in CrypTool 2 verschiedene kryptographische Verfahren simulieren und auch miteinander kombinieren kann. Dafür stehen dem Benutzer in CrypTool 2 eine Reihe von verschiedenen Bausteinen zur Verfügung, auf die ich in Kapitel 2 näher eingehe.

Die zweite Funktion mit der ich mich hier beschäftige ist die Visualisierung von Algorithmen. CrypTool 2 ermöglicht es dem Benutzer bei einigen Bausteinen eines Algorithmus, wie zum Beispiel Schieberegistern, die inneren Abläufe einzusehen. Dies ist sozusagen 'ein Blick hinter die Kulissen', durch den der Benutzer sehen kann, was bei der Ausführung von Algorithmen im Hintergrund abläuft. Diese Funktion erläutere ich in Kapitel 3 näher.

2 Visuelle Programmierung in Cryptool 2

2.1 Visuelle Programmierung

Visuelle Programmierung findet nicht auf Code-Ebene statt. Vielmehr werden in einer visuellen Programmierumgebung vorgefertigte Bausteine eingesetzt und miteinander verbunden, um so Algorithmen darzustellen.[1] Der Vorteil hierbei ist, dass sich so in sehr kurzer Zeit ein ausführbarer Algorithmus visualisieren lässt. Allerdings ist man, anders als bei der Programmierung auf Code-Ebene, an die Bausteine gebunden, die man zur Verfügung hat.
In CrypTool 2 ist dies durch eine Reihe von klassischen und modernen Kryptographieverfahren gegeben, die als Bausteine zur Verfügung stehen. Diese kann man mit Ein- und Ausgabebausteinen verbinden, um einfache kryptographische Verfahren zu simulieren.

2.2 Bausteine in CrypTool 2

Abbildung 2.1: Vigenere Baustein (links)und Texteingabe Baustein (rechts), Screenshots aus CrypTool 2

Die meisten Bausteine in CrypTool 2, die ein kryptographisches Verfahren darstellen, sehen aus wie der Vigenere-Baustein in Abbildung 2.1(links).
Auf der linken Seite befinden sich Datenstromeingänge. Bei der Vigenere-Verschlüsselung wird ein Text mit Hilfe eines Schlüsselwortes verschlüsselt. Daher hat der Vigenere-Baustein einen Eingang für den Textstring, den man verschlüsseln möchte (Oben), einen Eingang für ein Eingabe-Alphabet (Mitte) und einen Eingang für das Schlüsselwort als String (Unten).

[1] http://de.wikipedia.org/wiki/Visuelle_Programmierumgebung

Auf der rechten Seite befinden sich die Ausgänge. Bei dem Vigenere-Baustein ist das nur der Ausgang für den codierten Textstring.
Abgesehen von den Eingängen kann man die Funktion eines Bausteins durch erweiterte Einstellungen manipulieren. Auf diese kann man per Doppelklick auf den Baustein zugreifen. Hier kann man beispielsweise einstellen, ob der Baustein ver- oder entschlüsselt und ob er dabei Groß- und Kleinschreibung beachtet oder vernachlässigt.
Die Prozentzahl am unteren Rand gibt später bei der Ausführung an, ob der Baustein vollständig ausgeführt wurde (100 %) oder falls nicht, wie weit er gekommen ist. Werden die 100% nicht erreicht liegt das daran, dass der Baustein seine Berechnungen nicht fortsetzen kann. In den meisten Fällen liegt das an einer fehlenden oder fehlerhaften Eingabe.

In Abbildung 2.1(rechts) ist ein Texteingabe-Baustein dargestellt. Dieser wird benutzt, um dem Benutzer Texteingaben in Form von Strings zu ermöglichen. Der gewünschte Text kann einfach in dem Baustein abgetippt werden und wird später beim Ausführen als String über den Ausgang auf der rechten Seite weitergegeben. Analog gibt es diesen Baustein auch als Textausgabe mit einem String-Eingang anstatt einem String-Ausgang.

2.3 Vigenere-Verschlüsselung mit CrypTool 2

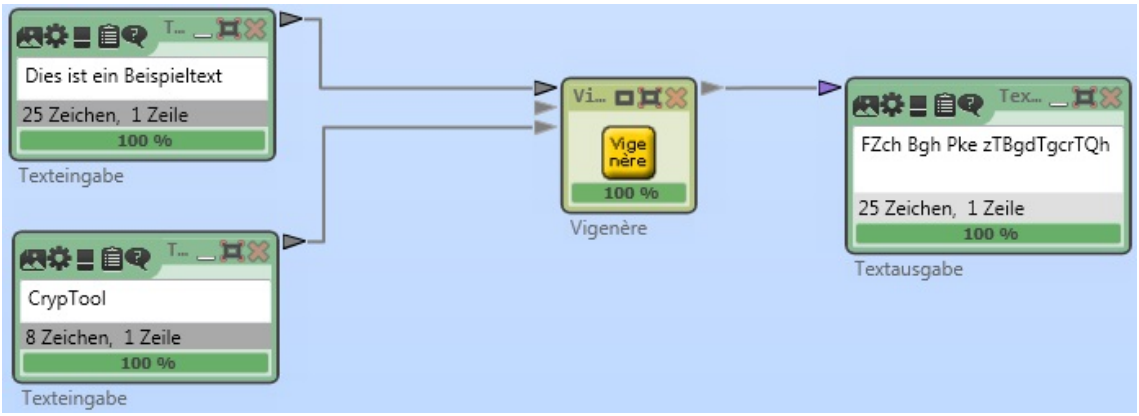

Abbildung 2.2: Vigenere-Verschlüsselung , Screenshot aus CrypTool 2

In Abbildung 2.2 sieht man, wie sich die Vigenere-Verschlüsselung mit CrypTool 2 simulieren lässt. Der Kern ist der Vigenere-Baustein. Von links bekommt er als Eingaben einen Text (”Dies ist ein Beispieltext”) und ein Schlüsselwort (”CrypTool”) mit Hilfe der Texteingabe-Bausteine. Da die Alphabet-Eingabe nicht benutzt wird, verwendet der Baustein hier automatisch das Standardalphabet. Auf der rechten Seite ist ein Textausgabe-Baustein, in dem der codierte Text angezeigt wird. Die Ein- und Ausgänge der Bausteine lassen sich einfach per drag-and-drop miteinander verbinden.

Hierbei ist praktisch, dass man den selben Aufbau auch zur Entschlüsselung benutzen kann. Dazu muss man nur in den Einstellungen des Vigenere-Bausteins die Aktion Entschlüsselung auswählen.
Auf diese Art kann man mit der visuellen Programmierung von CrypTool 2 verschiedene kryptographische Verfahren oder auch Kombinationen aus mehreren Verfahren simulieren, wie im nächsten Abschnitt näher erklärt wird.

2.4 Kombination mehrerer Verfahren

Durch den Einsatz von kryptographischen Verfahren soll die Sicherheit von Programmen erhöht werden, indem sie vor Fremdeingriffen geschützt werden. Aber gerade in sicherheitskritischen Bereichen reicht ein kryptographisches Verfahren möglicherweise nicht aus, da es zu leicht zu entschlüsseln ist. CrypTool 2 bietet die Möglichkeit Kombinationen aus mehreren Verfahren zu simulieren.

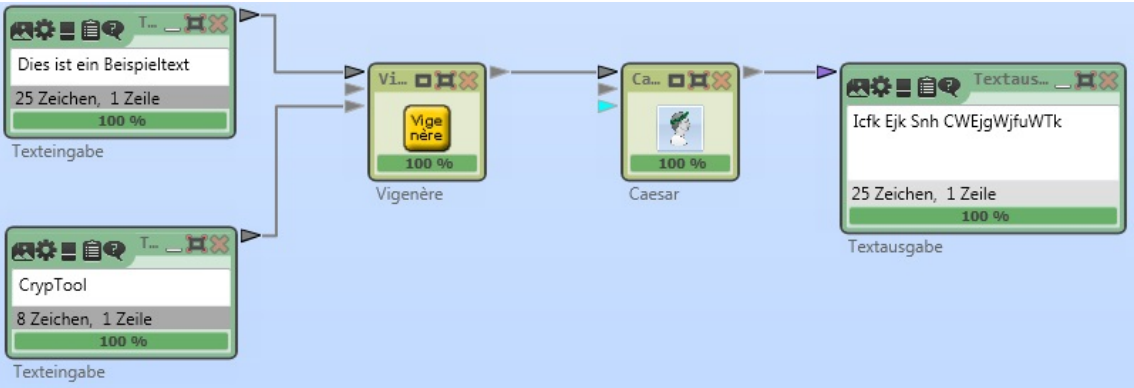

Abbildung 2.3: Kombination mehrerer Verfahren, Screenshot aus CrypTool 2

In Abbildung 2.3 sieht man das Vigenere-Verschlüsselung-Beispiel, welches um einen zusätzlichen Baustein erweitert wurde, welcher nun die Ausgabe des Vigenere-Bausteins als Eingabe erhält.
Dieser zusätzliche Baustein stellt das Caesar-Verschlüsselungsverfahren dar. Dabei handelt es sich um ein einfaches Verfahren welches jedes Symbol im Ursprungstext um einen bestimmten Wert im Alphabet verschiebt. In diesem Beispiel bekommt er den schon durch den Vigenere-Baustein verschlüsselten Text als Eingabe und verschiebt die Zeichen um den Wert 3. Der Wert der Verschiebung ist hier allerdings nicht sichtbar, sondern wird über die internen Einstellungen des Bausteins festgelegt, auf die man per Rechtsklick zugriff hat.
Bei der Kombination mehrerer kryptographischer Bausteine ist zu beachten, dass die Datentypen der Ausgabe des ersten und Eingabe des zweiten Bausteins übereinstimmen. Ist dies nicht der Fall, kann der Datentyp in einigen Fällen, z.B. von String zu Byte[ ], mit Hilfe eines zusätzlichen Bausteins angepasst werden. Ein Beispiel dazu befindet sich im nächsten Kapitel.
Auf diese Weise lassen sich mehrere kryptographische Bausteine kombinieren und man kann durch Simulationen herausfinden, ob sich die entsprechenden Verfahren miteinander kombinieren lassen.

3 Visualisierung von Algorithmen mit CrypTool 2

3.1 Visualisierung von Algorithmen

Wie in Kapitel 2 gezeigt lassen sich kryptographische Verfahren schnell und einfach mit CrypTool 2 simulieren. Dabei werden die Algorithmen, die für das gewünschte Verfahren benutzt werden sollen, durch Bausteine dargestellt. Dadurch erhält man jedoch nur einen stark begrenzten Einblick in die eigentliche Funktionsweise des Algorithmus.
An diesem Punkt kommt die Visualisierung von Algorithmen ins Spiel. Viele Bausteine in CrypTool 2 verfügen über eine Funktion, die Präsentation heißt. Mit dieser Funktion kann der Benutzer die Arbeitsweise dieses Bausteins genauer betrachten. Wie diese Visualisierung aussieht hängt von dem Baustein ab, den man gerade betrachtet. Bei einem Schieberegister kann man sich beispielsweise den Inhalt des Registers bei jedem Ausführungsschritt anschauen. Im nächsten Abschnitt werde ich diese Funktion an einem anderen Beispiel genauer erläutern.

3.2 Beispiel: Passwortstärke

CrypTool 2 verfügt über einen Baustein, der einen Algorithmus beinhaltet, mit dem man die Stärke eines Passwortes überprüfen kann.

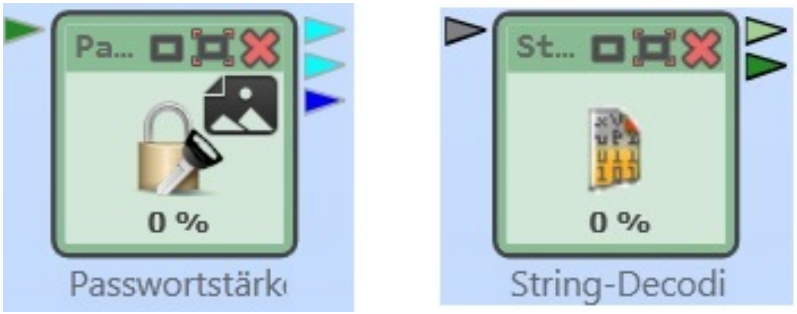

Abbildung 3.1: Baustein Passwortstärke (links) und Baustein String-Decodierer (rechts), Screenshots aus CrypTool 2

Dieser Baustein (Abbildung 3.1 links) erhält als Eingabe ein Passwort vom Typ Byte[ ] und bewertet beim Ausführen dessen Stärke. Da die Texteingabe Bausteine aus Kapitel 2 nur einen Datenstrom vom Typ String ausgeben, wird hier ein weiterer Baustein benötigt.

In Abbildung 3.1 (rechts) ist ein String-Decodierer-Baustein zu sehen. Mit diesem Baustein kann man einen String in einen ICryptoolStream (spezieller Datentyp von CrypTool 2, oberer Datenausgang) oder in eine Byte[ ] (unterer Datenausgang) umwandeln.
Nun ist alles bekannt, was man benötigt, um den Passwortstärke-Baustein zu benutzen.

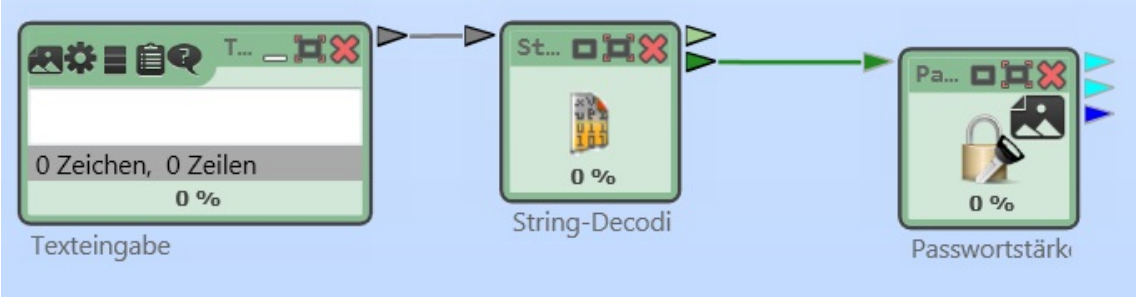

Abbildung 3.2: Aufbau zur Prüfung der Passwortstärke , Screenshot aus CrypTool 2

In Abbildung 3.2 sieht man den kompletten Aufbau, den man benötigt, um den Passwortstärke-Baustein zu benutzen. Allerdings sieht man auch hier nicht wie der Algorithmus arbeitet. Hier lässt sich jetzt die Präsentation-Funktion von CrypTool 2 nutzen, um den Algorithmus zu visualisieren. Dazu ruft man die Einstellungen des Passwortstärke-Bausteins mit einem Doppelklick auf und wählt den Reiter 'Präsentation' aus.

Abbildung 3.3: Präsentation des Passwortstärke-Bausteins , Screenshot aus CrypTool 2

In Abbildung 3.3 ist die Tabelle abgebildet, die genutzt wird, um den Passwortstärke-Algorithmus zu visualisieren. Man erkennt nicht nur welche Kriterien mit ein berechnet werden, sondern auch die genaue Formel mit der berechnet wird, wie stark das jeweilige Kriterium den Gesamtwert beeinflusst. So geht zum Beispiel die Zeichenanzahl, die in der Berechnung durch Formel '+(n*4)' beschrieben wird, vierfach in den Gesamtwert mit ein.
Außerdem werden die Werte, abhängig davon, ob sie für das jeweilige Kriterium einen guten oder schlechten Wert haben, grün oder rot markiert. Beim Testen eines Passwortes kann man so sofort erkennen, wie man das Passwort verändern muss, um einen besseren Gesamtwert zu erzielen. Davon abgesehen befindet sich ganz oben eine Punkteskala, auf der die Gesamtpunkte dargestellt werden.
In diesem Beispiel wird die Eingabe 'Passwort123' verwendet. Man sieht, dass das Passwort 75 Punkte erreicht hat und somit relativ stark ist. Man erkennt auch auf den ersten Blick, wo die Schwächen des Passwortes liegen, da die entsprechenden Kriterien mit einem rot unterlegten X oder einem gelb unterlegten Ausrufezeichen gekennzeichnet sind. Dadurch ist es einfach das Passwort zu modifizieren.
Dieses Beispiel zeigt deutlich, dass die Visualisierung in CrypTool 2 sehr dabei hilft die Algorithmen, die hinter den Bausteinen stecken und damit auch die Funktion des Bausteins zu verstehen.

4 Fazit

Am Ende stellt sich nun die Frage, ob CrypTool 2 ein Werkzeug ist, welches bei der Erstellung sicherer Software nützlich ist.
Mit der visuellen Programmierung in CrypTool 2 kann man, wie in Kapitel 2 gezeigt, sehr schnell und einfach die bekanntesten kryptographischen Verfahren simulieren und diese auch miteinander kombinieren. Dadurch kann man verschiedene Möglichkeiten ausprobieren, noch bevor man das Verfahren in seine Software implementiert und anschließend genau den Algorithmus auswählen, welcher die Anforderungen an die Software am besten erfüllt.
Da CrypTool 2 insgesamt 39 verschiedene kryptographische Verfahren (15 klassische und 24 moderne) als Bausteine zur Verfügung stellt und man diese zusätzlich noch beliebig kombinieren kann, ist es fast sicher, dass man mindestens ein Verfahren findet, das die Anforderungen erfüllt. Außerdem lernt man, durch das Ausprobieren dieser Bausteine, auch neue kryptographische Verfahren kennen, wodurch sich die Chance einen möglichst gut zum Programm passenden Algorithmus zu finden erhöht. CrypTool 2 unterstützt dies durch eine Dokumentation und Vorlagen, die bereits zur Ausführung bereit sind.
Um die Arbeitsweise der einzelnen Algorithmen zu verstehen, fand ich die Visualisierung durch die Präsentation-Funktion, die ich in Kapitel 3 erläutert habe sehr gut und ich denke, dass durch diese unterstützende Funktion die Auswahl eines passenden Algorithmus erleichtert wird.
Deshalb ist die visuelle Programmierung von CrypTool 2 zusammen mit der Visualisierung von Algorithmen in meinen Augen ein nützliches Werkzeug für die Erstellung sicherer Software.

5 Literaturverzeichnis

- CrypTool 2 (Version 2.0.5751.1)
- http://www.cryptool.org/de/cryptool2 (Stand: 15.12.2013)
- https://www.cryptool.org/trac/CrypTool2/wiki/WikiStart (Stand: 15.12.2013)
- http://www.cryptool.org/de/ct2-dokumentation-de (Stand: 15.12.2013)
- http://de.wikipedia.org/wiki/Visuelle_Programmierumgebung (Stand: 29.01.2014)