CODING
Programme zur direkten Chiffrierung

Projekt CODING

von

Dipl.-Math. Jürgen Müller

D-64289 Darmstadt

Stand: 29.11.2013

Motivation

"Aha, noch so ein symmetrisches Verschlüsselungssystem, wie langweilig !"
Nicht ganz !

- Sowohl beim Ver- als auch beim Entschlüsseln werden **2-stellige Megabyte-Werte pro Sekunde** als Datendurchsatz erreicht.
- Es gibt **keine "Falltür-Funktionen"**, d.h. der CODING-Algorithmus ist garantiert absolut "sauber" umgesetzt. Sie können sich im Zweifelsfall den Quellcode selbst herunter laden und mit dem Free-Pascal-Comiler (http://www.freepascal.org/) bzw. mit dem Lazarus-Entwicklungssystem (http://sourceforge.net/projects/lazarus/) Ihre eigenen ausführbaren Module erstellen. Der Autor nicht, und auch keine sonstigen ggf. noch so potenten Organisationen sind auch nur ansatzweise in der Lage, einen nicht trivialen Schlüssel zu erschließen, ohne alle in Frage kommenden Schlüssel auszuprobieren.
- In CODING können Sie eine **besondere Methode zur Eingabe des Schlüssels** verwenden, das sogenannte **"Überladen"** (auch „überlagern" genannt).
  Sie kennen das: Man erklärt Ihnen, Sie sollen einen möglichst komplexen Schlüssel verwenden und diesen bitte schön auch noch alle 3 Monate wechseln ! Wie kommen Sie dazu, sich der Technik anpassen zu müssen ? Wieso kann sich die Technik nicht Ihnen anpassen und Sie "griffige" und damit für Sie "merkbare" Schlüssel verwenden ?

  Da haben Sie völlig recht ! CODING macht das für Sie !

  Wie's geht ? Ganz einfach:
  Sie geben zu Ihrem gewöhnlichen Schlüssel noch eine oder mehrere Dateien an, aus denen Informationen dazu verwendet werden, Ihren Schlüssel zu "modifizieren". Aber kann das nicht ein "Bösewicht" nachbilden und bei seinen Entschlüsselungsversuchen mit berücksichtigen ?
  Ja und nein ! Von einer so genannten Überladedatei können bis zu allen Informationen zum Überladen des Ausgangs-Schlüssels berücksichtigt werden. Welche Informationen und wie hängt von weiteren Parametern und dem jeweiligen Schlüsselwert an angesprochenen Stellen ab. Damit aber bleibt dem "Bösewicht", **selbst wenn er Ihre Überladedateien und die damit zusammenhängenden Parameter kennt, nichts anderes übrig, als für jeden Schlüssel**, mit dem er einen Entschlüsselungs-Versuch startet, die Überladedateien erneut durchzugehen und jede Information einer Überladedatei anhand des gerade vorliegenden Schlüssels zu überprüfen !
  Diesen Aufwand zur Entdeckung eines selbst relativ einfachen Schlüssels durchzuführen, reduziert die Anzahl der möglichen Entschlüsselungs-Versuche pro Zeiteinheit so entscheidend, dass **Ihr einfacher Schlüssel mit Überladedateien es allemal mit dem Entdeckungsaufwand von komplexen Schlüsseln ohne Überladedatei aufnehmen kann !**
  Sind jedoch Ihre Überladedateien und die damit zusammenhängenden Parameter nicht bekannt, so besteht Ihr eigentlicher Schlüssel auf jeden Fall aus Ihrem auf die maximale Schlüssellänge duplizierten "einfachen" Schlüssel, der "zufällig" an einzelnen oder mehreren Stellen mit Informationen aus den Überladedateien überschrieben wurde. **D.h. Ihr eigentlicher Schlüssel ist ein unsystematischer Schlüssel maximaler Länge und damit müsste ein "Bösewicht" alle solche Schlüssel prüfen !**

  Sie sehen, das Merken von komplizierten Passworten (die man sich nicht merken kann, weshalb man diese dann doch aufschreibt, was wiederum "schlecht" ist) hat ein Ende. Vergessen Sie komplizierte Regeln, Ihr Passwort gehört in Ihren Kopf und zwar so, dass Sie es sich einfach merken können !

Kurzexposé

1 Beschreibung des Verfahrens

Bei dem durch die Programme CODINGx (x=1,2,3,...) verwendeten Verfahren handelt es sich um ein Verfahren zur Chiffrierung von beliebigen Daten „in place“. Das Verfahren ist charakterisiert durch

- die Eigenschaft, dass Ausgangsdaten und chiffrierte Daten die **gleiche Länge** haben,
- eine **hohe Operationsgeschwindigkeit** (z.B. zweistelliger MB-Wert / sec. bei einer Blocklänge von 512 Bytes , Dimension t=1 [siehe 1.4 unten], PC des Autors),
- **unabhängig** voneinander bearbeitete **Datenblöcke** (fast) **beliebiger Blocklänge** (d.h. falls ein Block nicht entschlüsselbar ist, so sind alle übrigen Blöcke davon nicht betroffen; weiter kann mit variabler Blocklänge gearbeitet werden),
- die Möglichkeit der **Stromchiffrierung** sowohl als Blockchiffrierung als auch **mit Blöcken variabler Länge** und **weitgehendem Kompromittierungsschutz**,
- eine **sehr hohe Sicherheit**, die durch die Dimensionierung des Verfahrens (Dimension t) in beliebigen Regionen angesiedelt werden kann (**à** http://eprint.iacr.org/2013/742.pdf),
- die Möglichkeit, eine „große Anzahl“ von **Authentisierungs- und Hash-Verfahren** mit (fast) **beliebigen Ergebnis-Längen** erzeugen zu können.

Damit ist CODING das ideale Werkzeug

- zur Chiffrierung von „**Voice over IP**“, „Bluetooth“-/„Wireless“-Daten,
- zur Chiffrierung von **Multimedia**-Informationen (z.B. MP3-Dateien, DVD-Filmen usw.),
- zur Chiffrierung von beliebigen **Austauschdaten** (z.B. zwischen Auftraggebern/ Abnehmern und Zulieferern/Lieferanten in **der Industrie**) auch über hochgradig unsichere Verbindungen (z.B. **via Internet**).

1.1 Performance

Eine besondere Herausforderung stellt die zeitkritische Chiffrierung von Massendaten dar. Dies ist u. a. in einem Realtime-Szenario der Fall, bei dem ein kontinuierlicher Datenstrom in Echtzeit verarbeitet werden muss. Jede Verzögerung durch eine zu langsame Verarbeitung führt zwangsläufig zum Kollaps und damit zu einer nicht Anwendbarkeit, d.h. dem Ausschluss des Chiffrierverfahrens.

Das Verfahren CODING reduziert die Anzahl der Multiplikationen, die im Endeffekt notwendig sind, auf ein notwendiges Minimum, das zur Aufrechterhaltung der vollen Sicherheit selbst bei homogensten Daten (geprüfter Testfall: 1 GB binäre Nullen) erforderlich ist. Dadurch erst wird die gegenüber anderen Verfahren sehr hohe Ausführungs-geschwindigkeit erreicht, wobei das Verfahren neben einer höheren Flexibilität gleichzeitig auch eine höhere Sicherheit (speziell für t>1) für sich in Anspruch nehmen kann.

1.2 Operations-Sicherheit

Die bisherigen Chiffriersysteme wie DES, Tripple-DES, AES arbeiten zum einen mit festen Blocklängen, so dass kürzere Daten zunächst auf diese Längen erweitert werden müssen. Zum anderen gewährleisten ggf. nur verschränkte Blöcke solcher Systeme ausreichend Sicherheit, d.h. dass ein fehlerhafter Block (z.B. durch Übermittlungsfehler) die Dechiffrierung aller weiteren Blöcke unmöglich macht.

Bei CODING existieren prinzipiell keine definierten Blocklängen. D.h. kürzere Informationen am Ende einer Datenmenge müssen nicht erweitert werden, was die Größe der chiffrierten Datenmenge gegenüber der der Ausgangsdaten unverändert lässt.

Wesentlicher für die Operationssicherheit jedoch ist, dass ein beschädigter Block nicht die Dechiffrierung der weiteren Blöcke verhindert. Dies betrifft speziell z.B. Telefongespräche (Voice over IP), bei denen eine Störung im Datenfluss nicht automatisch dazu führen muss, das Gespräch wieder völlig neu aufzubauen. Auch bei sonstiger Datenübermittlung ist es mit CODING lediglich notwendig, den/die gestörten Datenblöcke erneut zu übertragen. Mit entsprechenden Modifikationen des hier vorgestellten Algorithmus (z.B. Wegschreiben der Wiederaufsetzpunkt-Parameter für einen gestörten Block) können diese auch nachträglich schnell und effizient bearbeitet werden.

1.3 Chiffrier-Sicherheit

Selbst die neueren Kodierungssysteme wie AES verwenden nur Schlüssel mit maximaler Länge von 2048 Bits. Schon die einfachste Version von CODING (Dimension t=1) verwendet einen Schlüssel der maximalen Länge von 2048 Bits (für Dimension t=2 sind es bereits 1.048.576 Bits maximaler Schlüssellänge, für Dimension t=3 sind es 402.653.184 Bits). Dabei umfasst CODING eine Methode, durch Schlüsselüberladung Dateien in den Schlüsselerzeugungsprozess einzubeziehen und damit erst die Erzeugung sinnvoller „zufälliger“ Schlüssel genannten Umfangs zu ermöglichen.

Nach Untersuchungen des Autors (http://eprint.iacr.org/2013/742.pdf) kann das Verfahren nicht gebrochen werden, ohne einen Großteil aller möglichen Schlüssel durchzuprobieren. Eine genauere Beurteilung der „Sicherheit“ speziell in informationstheoretischer Sicht (d.h. müssen für nicht triviale Schlüssel tatsächlich immer alle Schlüssel überprüft werden oder lässt sich die Anzahl zu prüfender Schlüssel einschränken) muss einer eingehenderen Untersuchung vorbehalten bleiben.

Unabhängig davon wurden vom Autor bereits umfangreiche Tests vorgenommen, die nachweisen, dass selbst bei homogensten Daten (nur binäre Nullen) das Verfahren gegen die Gleichverteilung der binären Nullen und Einsen sowie der Bytes bzw. Byte-Gruppen konvergiert.

Des weiteren konnten „near-key“-Untersuchungen zeigen, dass selbst Versuchsschlüssel, die nur ein oder wenige Bits vom tatsächlichen Schlüssel entfernt waren, zu keinen statistisch relevanten Änderungen in den genannten Verteilungen gegenüber den verschlüsselten Daten führten. Auch Blockdifferenz-Betrachtungen und deren Verteilungen konnten keine relevanten Unterschiede zu zufälligen Blockinhalten aufdecken.

Vielmehr kann das hier vorgestellte Verfahren auch dann als sicher eingestuft werden, wenn neben dem Chiffrier-Algorithmus selbst sowohl die Ausgangsdaten als auch die kodierten Daten bekannt sind. Der ursprüngliche Schlüssel selbst kann daraus nicht (bzw. nur durch Testen aller möglichen Schlüssel) ermittelt werden.

In der höchsten Sicherheitsstufe führt selbst die Kompromittierung eines Block-Abschnitts nicht automatisch zur Kompromittierung vorangehender oder nachfolgender Block-Abschnitte.

1.4 Zukunfts-Sicherheit

Hinter dem Verfahren CODING steht eine (unendliche) Reihe von konkreten Chiffrierverfahren. Die Dimension t eines Verfahrens ist dabei definiert als die Anzahl von Bytes, die zu einer Byte-Gruppe zusammengefasst wird. Für t=1 werden daher einzelne Bytes betrachtet und es finden 1-Byte-Substitutionstabellen ihre Anwendung. Für t=2 werden jeweils 2 Bytes als eine Byte-Gruppe aufgefasst, für die 2-Byte-Substitutionstabellen Anwendung finden, usw.

Das Verfahren CODING selbst ist bezüglich t **nicht begrenzt**, bis t=3 existieren zur Zeit vom Autor entwickelte einsatzfähige PC-/Unix-Programme.

1.5 Mächtigkeit der Schlüsselräume

Falls man „nur“ die Mächtigkeit des Schlüsselraums betrachtet (allgemein (256**t)**(256**t) Elemente), so ergeben sich für t=1 ca. 3,2317*10**616, für t=2 ca. 6,74114*10**315.652 bzw. für t=3 ca. 10**121.210.686 mögliche Schlüsselraum-Elemente (x**t, sprich „x hoch t“, ist definiert als x*x*...*x, wobei t-mal der Wert x auftritt). Letztere Zahl ist z.B. darstellbar durch eine 1 mit Nullen, die mehr als 33 Bände á 1000 Seiten jeweils gefüllt mit 3600 Nullen belegen (zum Vergleich: Atome im Universum: ca. 10**80). Es lässt sich leicht nachweisen, dass mindestens (256**t)! Elemente des jeweiligen Schlüsselraums durch den hier angesprochenen Algorithmus auch tatsächlich wirksam werden können (n!, sprich „n Fakultät“, ist definiert als 1*2*3*...*n), tatsächlich sind alle Elemente jeweils eines Schlüsselraums wirksam.

Unter diesen Umständen muss t nur „groß genug“ gewählt werden, um niemandem selbst mit zukünftig denkbaren (Quanten-)Rechnern bzw. Rechner-Farmen nur irgend eine realistische Möglichkeit zu eröffnen, mit CODING chiffrierte Daten in akzeptabler Zeit ohne entsprechenden Schlüssel bzw. Chiffrier-Parameter zu dekodieren. Zur Zeit dürfte das spätestens für t=2 der Fall sein.

2 Alleinstellungsmerkmale

- CODING ist ein „in place“ –Verfahren, d.h. keine Änderung der Datenlänge.
- CODING ist mit seiner hohen Operationsgeschwindigkeit auch als Stromchiffrierungs-Verfahren für Realtime-Prozesse geeignet.
- CODING ist für (fast) beliebige (auch variable) Blocklängen einsetzbar.
- CODING besitzt
  - eine sehr hohe Operations-Sicherheit,
  - eine sehr hohe Chiffrier-Sicherheit,
  - eine sehr hohe Kompromittier-Sicherheit und
  - eine sehr hohe Zukunfts-Sicherheit.

3 Entwicklungsstand

Das hier als CODING bezeichnete Verfahren hat seine Anfänge in Voruntersuchungen des Autors, die vor mehr als 20 Jahren stattfanden. Die konsequente Weiterentwicklung, Verallgemeinerung und Umsetzung in Free-Pascal-Programme bisher für die Dimensionen t=1, t=2 und t=3 wurde in jüngster Vergangenheit auch auf Grund der geänderten Anforderungen auf der potentiellen Anwenderseite und den Prozessor-Möglichkeiten auf der Durchführungsseite voran getrieben.

Genannte CODING-Programme sind für t=1, t=2 und t=3 **voll ausgetestet** und **einsatzfähig**.