University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Application Security
Lab 1

4. November 2013

Inhaltsverzeichnis

1 Aufgabenstellung

Konstruieren Sie zwei Briefe an Alice, einen (original) für Bob und einen (gefälscht) für Alice, die aber den gleichen Hashwert haben. Da die Fälschung etwas für Sie einbringen soll, ersetzen Sie im Brief an Alice die Kontonummer 222-1101.461.12 durch Ihre eigene: 202-1201.262.10. Sie haben freilich bei der gleichen Bank ein Konto mit dieser Nummer eröffnet.

Ihre Aufgabe besteht darin, sogenannte Kollisionen im Hash-Verfahren zu suchen, d.h. Änderungen im Originaltext, die den gleichen Hashwert liefern: $h(m_{orig}) = h(m_{fake})$.

2 Softwareaufbau

Zur Lösung der Aufgabe wurden die Klassen ”HashAlgorithm” und ”Generator” implementiert.

Die Klasse ”HashAlgorithm” dient ausschliesslich zur Berechnung der Hash-Werte. In der Klasse ”Generator” werden die Briefe eingelesen, generiert und bei einer Kollision in einem Text-File ausgegeben.

2.1 Hashfunktion

Die Hashfunktion wurde gemäss Aufgabenstellung implementiert.

Um den Output mit der Länge von 64 Bit zu verkürzen, wird er in zwei 32 Bit Blöcke geteilt und durch die Verwendung von XOR zu einem 32 Bit Block ”verschmolzen”.

2.2 Generator

2.2.1 Variationserzeugung

Laut der Aufgabenstellung haben wir 2^{32} verschiedene Kombinationsmöglichkeiten pro Mail.
In unserem Projekt wird jeweils für jeden Platzhalter durch eine Zufallszahl eine der Möglichkeiten ausgewählt und dann ersetzt. Durch einfaches hinzufügen neuer Möglichkeiten im Replacements.txt file können so neue Kombinationen hinzugefügt werden.

2.3 Search Collisions

2.3.1 Strategien

Zur Generierung der Mails verwenden wir keinen linearen Ansatz, sondern einem Randomisierten. Dadurch wird die Performance erhöht und Memory Footprints werden verringert. Letzteres wird interessant, weil vor Start der Anwendung angegeben werden kann, nach wievielen Kollisionen man suchen möchte.

Erst werden 2^{11} (2048) Varianten der Original- und der Fake-Email generiert, anschliessend werden diese 2048 Varianten auf Kollisionen verglichen, bevor weitere generiert werden.

Zusätzlich zur gemeinsamen Generierung von Original- und Fake-Emails bietet unsere Implementation die Möglichkeit, der Definition des Geburtstagsparadoxons entsprechend, nur Fake-Emails zu generieren und diese mit dem Original des Aufgabenblattes zu vergleichen. Allerdings haben wir auf diese Art auch nach längerer Suche keine Kollisionen finden können, während bei generierten Originalfiles nach durchschnittlich je 80000 generierten Files eine Kollision entdeckt wurde.

2.3.2 Datenstrukturen

Unsere Lösung verwendet zwei Hashmaps (Integer, String). Eine für alle Variationen der Original-Mail und eine für die falschen Mails. Für den Key benutzen wir jeweils den Hashwert und als Value wird der generierte Text dieser Mail verwendet.