25. OBERSCHULE „AM POHLANDPLATZ“ DRESDEN

Dokumentation zum „RSA-Tool“

eine Informatikabschlussarbeit der 25. Oberschule „Am Pohlandplatz“ Dresden

von Alec Schitzkat und Karl R. Jahn

Dresden, der 25.02.2022

Das RSA-Tool ist eine kleine Windowsanwendung, welche zur Visualisierung der RSA-Verschlüsselung dient, und hierbei gleichzeitig heutigen Sicherheitsstandards entsprechen soll. Umgesetzt wurde dies in C++ (ISO C++ 20, MSV) und C# (Windows Forms, .NET Framework), welche über eine Kommandozeilenschnittstelle einseitig kommunizieren können. C# übernimmt hierbei die Darstellung um den Benutzer das Prinzip verständlich herüberzubringen. Dies ist die Dokumentation zu beiden Anwendungen und den Projektablauf/Schaffungsprozess.

1 INHALTSVERZEICHNIS

2 IDEE

Im Zeitalter der Digitalisierung wird die Sicherheit dieser digitalen Daten zunehmend wichtiger. Deshalb ist es sinnvoll, sich mit ebendieser zu beschäftigen und so kam uns die Idee zum einen anderen dabei behilflich zu sein und zum anderen selber natürlich dazuzulernen, indem wir dies mit dem „RSA-Tool“ ermöglichen.

Das RSA-Tool soll in der Lage sein, das RSA-Verfahren durchzuführen (und dies nach heutigen Standard) und dies gleichzeitig zu visualisieren, sodass der Benutzer in der Lage ist dieses augenscheinlich trockene Verschlüsslungsverfahren nachvollziehen und verstehen zu können. Und als letztes möchten wir uns in der Kryptographie ausspielen und einige Verfahren zu unseren „eigenen“ vereinen.

3 DAS RSA-VERFAHREN

Das RSA-Verfahren, welches nach dessen Entwicklern Rivest, Shamir und Adlema benannt wurde basiert auf Zahlentheoretischen Problemen (die Ineffizienz der Faktorisierung großer Zahlen) und ist ein asymmetrisches Verfahren, d.h. es gibt einen privaten und einen öffentlichen Schlüssel (oder in den Fall sogar zwei).

3.1 SCHLÜSSELERZEUGUNG

Ablauf:

1. Wähle zwei unterschiedliche riesige (min. 600 Dezimalstellen, also 2048 Bytes) Primzahlen p und q.

   $p \in \mathbb{P}$
   $q \in \mathbb{P};\ p \neq q$

2. Berechne das RSA-Modul aus p und q.

   $N \coloneqq p * q;\ N \in \mathbb{N}$

3. Berechne die Eulersche φ-Funktion aus N.

   Eulersche φ-Funktion: $\varphi(m) = |\{ x \in \mathbb{N} \mid 1 \leq x \leq m \ \wedge\ \mathrm{ggT}(x, m) = 1 \}|$
   $\varphi(N) = (p-1)(q-1)$
   da $\varphi(p) = p - 1$, $\varphi(q) = q - 1$ und $N = p * q$

4. Wähle eine zu $\varphi(\mathrm{N})$ teilerfremde Zahl e.

   $e \in \mathbb{N};\ 1 < e < \varphi(N) \ \wedge\ \mathrm{ggT}\big(e, \varphi(N)\big) = 1$

5. Berechne d, welches das multiplikative Inverse zu e bezüglich $\mathrm{mod}\ N$ darstellt. Dies geht über den erweiterten euklidischen Algorithmus.

   $d \in \mathbb{N};\ d * e \bmod N = 1$

6. Fertig! Die öffentlichen Schlüssel sind nun (e, N) und das private Schlüsselpaar besteht aus (d, N). $\varphi(N)$, p und q werden nun nicht mehr gebraucht.

Der C++ - Quellcode herfür befindet sich in der Datei TODO.cpp, sowie ein kleiner Auszug dessen ist im Anhang 2 „Code“, Abschnitt 3 zu finden.

3.2 VERSCHLÜSSELUNG

Zur Verschlüsslung der Nachricht wird das öffentliche Schlüsselpaar (also $(e,\ N)$) benötigt. Dieses muss sich derjenige, der die Nachricht versenden möchte zuerst vom Empfänger anfordern. Nun muss der zu verschlüsselnde Text, bzw. die Nachricht als Zahl Codiert werden, dies erfolgt zumeist über den ASCII-Code. Da die Nachricht aber einer Häufigkeitsanalyse zu Opfer fallen kann, werden nun die Codierten Zeichen zu Blöcken zusammengefasst, meist werden hierfür zwei bis vier Zeichen zu einen Block zusammengefasst. Zu beachten ist hierbei, dass die Primzahl auch wirklich groß genug ist, da die Zahl kleiner als das RSA-Modul, N, sein muss.

Nach dieser kleinen Vorarbeit kann nun mit der Verschlüsselung begonnen werden. Hierfür wird jeder Zahlenblock verschlüsselt, indem man diesen Block mit den Exponenten e Potenziert und das N-te Modul daraus zieht. Dieses Verfahren lässt sich durch die binäre Exponentiation beschleunigen.

$m \in \mathbb{N};\ \ m \coloneqq \text{Nachrichtenblockkodierung}$
$c = c^e \bmod N$

Der C++ - Quellcode herfür befindet sich in der Datei TODO.cpp, sowie ein kleiner Auszug dessen ist im Anhang 2 „Code“, Abschnitt 2 zu finden.

3.3 ENTSCHLÜSSELUNG

Zur Entschlüsselung braucht der Empfänger logischer Weise seinen privaten Schlüssel. Nun wird das Verschlüsselungsverfahren umgedreht wiederholt. D. h. zuerst wird die Nachricht, bzw. der Nachrichtenkodierungsblock als Basis zum Exponenten d potenziert und das N-te Modul hieraus gezogen. Danach muss der der Block nur noch „auseinandergestückelt“ und decodiert werden und das war‘s.

$c \in \mathbb{N};\ \ c \coloneqq \text{verschlüsselter Nachrichtenblock}$
$m = c^d \bmod N$

Der C++ - Quellcode herfür befindet sich in der Datei TODO.cpp, sowie ein kleiner Auszug dessen ist im Anhang 2 „Code“, Abschnitt 3 zu finden.

4 VERSCHLÜSSELUNGSPROGRAMM (C++, CMD-LINE)

5 UI UND VISUALISIERUNG (C#, WINDOWSFORMS)

6 INTERFACE DER BEIDEN ANWENDUNG

Die Schnittstelle beider Anwendung, findet, wie bereits schon in den beiden vorherigen Abschnitten angedeutet über die „Kommandozeile“ und Dateien statt. Dies hat den Vorteil, dass das C++-Tool auch für andere Programme verwendet werden kann und so auch die Benutzung dessen von anderen Programmen für realistischere Anwendungsfälle in Betracht gezogen werden kann. Über den Programmnamen und eines der Argumenten „--h“, „--help", oder auch „/h“ oder „/help“, erhält der Benutzer des C++-Tools im CMD die in Bild TODO zu sehende Übersicht valider Argumente und der Benutzung des C++-Tools.

7 SCHAFFUNGSPROZESS

8 CODEAUSSCHNITTE

9 QUELLEN