kpsp Zwischenbericht

Manuel Jenny & Christian Glatthard

4. Semester (FS 2013)

Inhaltsverzeichnis

1 Abstract

2 Idee des Projektes

Ziel unseres Projektes ist es eine funktionierende Implementation nach der Funktionsweise von RSA zu programmieren. Diese muss nicht unbedingt den Sicherheitsstandards des echten RSA entsprechen, soll aber sämtliche Funktionen bereitstellen wie das Generieren von Keys sowie Ver- und Entschlüsselung von Strings. Soweit möglich versuchen wir dabei die im Modul Kryptografie kennengelernten Algorithmen zu verwenden.

3 Theoretischer Teil

RSA (benannt nach den Erfindern Ron Rivest, Adi Shamir, und Leonard Adleman) ist ein asymmetrisches kryptografisches Verschlüsselungsverfahren, welches sowohl zur Verschlüsselung, als auch zur digitalen Signatur verwendet werden kann.

Es wird ein privater und ein öffentlicher Schlüssel generiert. Der öffentliche wird zum Verschlüsseln und zum Prüfen von Signaturen verwendet und ist öffentlich zugänglich. Der private Schlüssel hingegen wird zum Entschlüsseln, sowie zum Signieren der Daten verwendet.

3.1 Vorgehen RSA

1. Wähle 2 Primzahlen p,q
2. n = p * q
3. Wähle natürliche Zahl e, teilerfremd zu $\phi(n)$, d.h. $gcd(e, \phi(n)) = 1$, für die gilt $1 < e < \phi(n)$
4. bestimme natürliche Zahl d mit $e * d * mod\phi(n) = 1$

3.2 Verwendete mathematische Formeln und Algorithmen

- ggt (grösster gemeinsamer Teiler)
- erweiterter Euklid
- inverer Modulo
- diverse Algorithmen zum Verifizieren von Primzahlen

4 Haskell-Programme

Listing 1: Modul Header

```
1 -- # Define RSA module and its public functions
2 module RSA
3 (  generatePrivateKey
4 ,  generatePublicKey
5 ,  encrypt
6 ,  decrypt
7 ) where
8
9 -- # import required external modules
```

```
10 import Prelude    -- contains gcd function(greatest common divisor)
11 import System.Random
12 import Control.Monad.Fix
13 import Data.Bits
```

---

5 Testfälle