Kryptographie II

Seminar: Elementare Zahlentheorie und Algebra

Dr. Stephan Ehlen

Anna Diermann 25. Juni 2019

---

1 RSA-Kryptosystem

Bei RSA handelt es sich um ein **Public-Key-Verschlüsselungsverfahren**, d.h., dass ein Klartext mit Hilfe eines öffentlichen Schlüssels durch den Sender in einen verschlüsselten Text umgewandelt werden kann, welcher mit Hilfe eines privaten Schlüssels durch den Empfänger wieder entschlüsselt werden kann.

1.1 Idee hinter RSA

Die Verschlüsselung geschieht durch Anwenden einer invertierbaren Funktion $E\colon X \to X$ auf die zu verschlüsselnde Nachricht, sodass die Entschlüsselung durch Einsetzen in E^{-1} erfolgen kann.
E soll so konstruiert werden, dass E^{-1} für den Empfänger leicht, aber für jede weitere Person schwer zu berechnen ist.

1.2 Konstruktion der Funktion E

Folgende Schritte werden vom Empfänger ausgeführt:

1. Wähle zwei große Primzahlen p, q und berechne $n := p * q$.
2. Berechne $\varphi(n) = \varphi(p) * \varphi(q) = (p-1) * (q-1)$.
3. Wähle zufällig $e \in \mathbb{Z}$ mit $1 < e < \varphi(n)$ und $ggT(e, \varphi(n)) = 1$.
4. Finde d, welches die Gleichung $e * d \equiv 1 \mod \varphi(n)$ löst. (Algorithmus 1)
5. Teile dem Sender den *public key* (n, e) mit; die Funktion E ist definiert durch $E\colon \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ mit $E(x) = x^e$.

1.3 Erweiterter Euklidischer Algorithmus

Algorithmus 1 Seien $a, b \in \mathbb{Z}$ mit $a > b \geq 0$. Der Algorithmus findet g, v, x, sodass $ggT(a, b) = g$ und $av + bx = g$.

1. Setze $v = 1, x = 0, r = 0, s = 1$.
2. Falls $b = 0$, setze $g = a$ und terminiere.
3. Finde $q, c \in \mathbb{Z}$ mit $0 \leq c < b$, sodass $a = qb + c$.
4. Setze $(a, b, r, s, v, x) = (b, c, v - qr, x - qs, r, s)$ und gehe zu Schritt 2.

Anwendung für RSA: Sei $a := \varphi(n)$ und $b := e$. Da $g := ggT(\varphi(n), e) = 1$ ist, erfüllt das x, welches dieser Algorithmus berechnet, die Gleichung $e * x \equiv 1 \mod \varphi(n)$ und kann somit als Dechiffrierschlüssel d verwendet werden.

1.4 Berechnung von x^e mod n

Algorithmus 2

1. Schreibe e in Binärdarstellung; seien $\varepsilon_i \in \{0, 1\}$ die Ziffern dieser Binärrzahl
   $\Rightarrow x^e = \prod_{\varepsilon_i=1} x^{2^i} \; mod \; n$
2. Berechne $x,\ x^2,\ x^{2^2} = (x^2)^2,\ x^{2^3} = (x^{2^2})^2$,..., $x^{2^r} \; mod \; n$, wobei $r + 1$ die Anzahl der Ziffern der Binärdarstellung von e ist.
3. Multipliziere die x^{2^i} zusammen ($mod\ n$), für die $\varepsilon_i = 1$ gilt.

1.5 Senden/Empfangen einer Nachricht

Vorgehensweise

1. Sender codiert die Nachricht als Sequenz von Elementen aus $\mathbb{Z}/n\mathbb{Z}$, also m_1,...,$m_r \in \mathbb{Z}/n\mathbb{Z}$.
2. Sender berechnet $E(m_1) = m_1^e$,...,$E(m_r) = m_r^e$ und sendet diese an den Empfänger. (Algorithmus 2)
3. Empfänger berechnet alle m_i durch $E^{-1}(E(m_i)) = (m_i^e)^d \underset{Proposition\ 1}{=} m_i$ und ermittelt aus diesen die decodierte Nachricht.

Proposition 1 (Dechiffrierschlüssel)
Sei $n \in \mathbb{Z}$ das Produkt von paarweise verschiedenen Primzahlen und seien $d, e \in \mathbb{N}$ so, dass $p - 1 | de - 1$ für jede Primzahl p mit $p|n$. Dann gilt $a^{d*e} \equiv a \mod n$ für alle $a \in \mathbb{Z}$.

2 RSA angreifen

Ein Angriff auf RSA ist unter Verwendung von Faktorisierungsalgorithmen[1] möglich, da man mit der Primfaktorzerlegung $n = p * q$ den Wert von $\varphi(n) = (p - 1) * (q - 1)$ berechnen und dadurch den Dechiffrierschlüssel d, der die Gleichung $e * d \equiv 1 \mod \varphi(n)$ löst, finden kann.

2.1 Berechnung der Faktorisierung von n mit bekanntem φ(n)

Es gilt: $\varphi(n) = (p - 1) * (q - 1) = pq - (p + q) + 1 \underset{n=p*q}{\Leftrightarrow} \varphi(n) = n - (p + q) + 1$
$\Leftrightarrow p + q = n + 1 - \varphi(n)$.

[1] siehe hierfür den Vortrag Faktorisierungsalgorithmen von Julian Sander

Die Nullstellen des Polynoms $f(x) = (x-p)(x-q) = x^2 - (p+q)x + pq$
$= x^2 - (n+1-\varphi(n))x + n$ sind also p und q, welche sich mit Hilfe der pq-Formel berechnen lassen. Es folgt also, dass die Berechnung von $\varphi(n)$ mindestens so schwer ist, wie die Bestimmung der Faktorisierung von n.

2.2 Sonderfall: |p − q| ist klein

Nutzung der Faktorisierungsmethode von Fermat. Sei $n = pq$ und OBdA $p > q$.
Es gilt: $n = (\frac{p+q}{2})^2 - (\frac{p-q}{2})^2$. Sei $s := \frac{p-q}{2}$ und $t := \frac{p+q}{2}$. Somit ist $s^2 = t^2 - n$ ein Quadrat.

1. Versuche $t = \lceil \sqrt{n} \rceil + i$ mit $i = 0, 1, ...$ solange, bis $t^2 - n = s^2$ ein Quadrat ist.
2. Berechne anschließend $s = \sqrt{t^2 - n} \in \mathbb{Z}$ und hiermit $p = t + s$ und $q = t - s$.

Fazit: p und q so zu wählen, dass $|p - q|$ klein ist, sollte vermieden werden, da sie sonst einfach zu berechnen sind.

3 Das diskrete Logarithmusproblem (Angriff auf Diffie-Hellman)

Sei G eine endliche zyklische Gruppe mit $ord(G) = \eta$. Sei des Weiteren $g \in G$ ein Erzeuger dieser Gruppe und $\alpha \in G$ ein beliebiges Element. Gesucht ist der diskrete Logarithmus von α zur Basis g, d.h. das kleinste, nicht negative $n \in \mathbb{Z}$, für das gilt: $\alpha = g^n$.
Zur Erinnerung: Der Diffie-Hellman Schlüsselaustausch[2] lässt sich hiermit angreifen, da wir p, g und g^n mod p (bzw. g^m mod p) abfangen können und uns die Werte n und m ermöglichen würden, den Schlüssel s zu berechnen.

3.1 Shanks Babystep-Giantstep Algorithmus

Algorithmus 3

1. Setze $t := \lceil \sqrt{ord(G)} \rceil$.
2. Berechne die Menge der "Babysteps" $B = \{(\alpha * g^{-r}, r) : 0 \leq r < t\}$. Wird hierbei ein Paar $(1, r)$ gefunden: Setze $n := r$ und terminiere.
3. Wenn $(1, r) \notin B$: Bestimme $\delta := g^t$ und prüfe $q = 1, 2, 3, ...$ solange bis $(\delta^q, r) \in B$. ("Giantsteps")
4. Nutze q und r aus dem gefundenen Paar zur Berechnung von $n := q * t + r$ und terminiere.

[2]Für die genaue Funktionsweise des Diffie-Hellman Schlüsselaustauschs siehe den Vortrag Kryptographie I von Sebastian Haines.

Proposition 2: Shanks Babystep-Giantstep Algorithmus benötigt $\mathcal{O}(\sqrt{(|G|)})$ Multiplikationen und Vergleiche in G und muss $\mathcal{O}(\sqrt{(|G|)})$ Elemente speichern.

3.2 Pollard-ρ-Algorithmus

Algorithmus 4

1. Definiere drei paarweise disjunkte Teilmengen $G_1,\ G_2,\ G_3 \subset$ G mit $G = \bigcup_{i=1}^{3} G_{\mathrm{i}}$. Sei die Funktion $f\colon G \to G$ definiert durch

$$f(\beta) = \begin{cases} g\beta, & \text{falls } \beta \in G_1, \\ \beta^2, & \text{falls } \beta \in G_2, \\ \alpha\beta, & \text{falls } \beta \in G_3, \end{cases}$$

2. Wähle $x_0 \in \{1, ..., \eta\}$ zufällig, setze $y_0 = 0$, berechne $\beta_0 = g^{x_0}$ und definiere die Folge (β_{i}) durch $\beta_{\mathrm{i+1}} = f(\beta_{\mathrm{i}})$.
3. Sei

$$x_{\mathrm{i+1}} = \begin{cases} x_{\mathrm{i}} + 1 \mod \eta, & \text{falls } \beta_{\mathrm{i}} \in G_1, \\ 2x_{\mathrm{i}}, \mod \eta & \text{falls } \beta_{\mathrm{i}} \in G_2, \\ x_{\mathrm{i}}, & \text{falls } \beta_{\mathrm{i}} \in G_3, \end{cases}$$

   und

$$y_{\mathrm{i+1}} = \begin{cases} y_{\mathrm{i}}, & \text{falls } \beta_{\mathrm{i}} \in G_1, \\ 2y_{\mathrm{i}}, \mod \eta & \text{falls } \beta_{\mathrm{i}} \in G_2, \\ y_{\mathrm{i}} + 1 \mod \eta, & \text{falls } \beta_{\mathrm{i}} \in G_3, \end{cases}$$

   So lassen sich alle Glieder der Folge (β_{i}) darstellen durch $\beta_{\mathrm{i}} = g^{x_{\mathrm{i}}}\alpha^{y_{\mathrm{i}}},\ i \geq 0$.
4. Da $ord(G) < \infty \Rightarrow \exists\ i \geq 0, k \geq 1\colon \beta_{\mathrm{i}} = \beta_{\mathrm{i+k}}$. Löse die folgende Kongruenz: $(x_{\mathrm{i}} - x_{\mathrm{i+k}}) \equiv n(y_{\mathrm{i+k}} - y_{\mathrm{i}}))\mod \eta$. Ist die Lösung hiervon nicht eindeutig *mod* η, finde das n, welches $\alpha = g^n$ erfüllt, durch Einsetzen der gefundenen Lösungen. Ist dies zu ineffizient, wiederhole den Algorithmus mit einem anderen Startwert x_0.

Vorgehensweise bei der Speicherung von $(\boldsymbol{\beta_{\mathrm{i}}}, \mathbf{x_{\mathrm{i}}}, \mathbf{y_{\mathrm{i}}})$

1. Setze $i := 1$.
2. Speichere $(\beta_{\mathrm{i}}, x_{\mathrm{i}}, y_{\mathrm{i}})$.
3. Berechne $(\beta_{\mathrm{j}}, x_{\mathrm{j}}, y_{\mathrm{j}})$ für $j = i+1, i+2, ...$ bis es ein Match (i, j) gibt, dann terminiere, oder bis $j := 2i$.
4. Lösche $(\beta_{\mathrm{i}}, x_{\mathrm{i}}, y_{\mathrm{i}})$, setze $i := 2i$ und gehe zu Schritt 2.

Proposition 3

Da die Vorperioden- und Periodenlänge $\mathcal{O}(\sqrt{(|G|)})$ ist, muss der Pollard - ρ - Algorithmus $\mathcal{O}(\sqrt{(|G|)})$ Folgenglieder berechnen, bis ein Match gefunden wird. Des Weiteren hat der Algorithmus konstanten Speicherbedarf.