1 Attacken auf das RSA-Verfahren

1.1 Faktorisierungsattacken

Versuche p und q prim zu berechnen mit $N = pq$, denn dann ist
$\varphi(N) = (p-1)(q-1)$ und d kann durch erweiterten Euklidischen Algorithmus berechnet werden.
Soll wohl die effizienteste Attacke sein?

1.1.1 Pollard ϱ

Die Idee bei Pollard ϱ ist es zwei Zahlen x und y zu finden, sodass

$$x \not\equiv y \mod N \text{und } x \equiv y \mod p$$

Dann ist nämlich $n = \text{ggT}(x - y, N)$ ein nichttrivialer ($\neq 1$) Teiler von N. Hierbei ist zu beachten, dass die einzigen nichttrivialen Teiler von N, p, q und N selber sind. p kann dabei auch erstmal unbekannt sein und ist ein Primfaktor von N. Der Algorithmus lautet wie folgt

- Wähle $x_0 \in \{0, \ldots, N-1\}$ und $c \in \mathbb{Z}\backslash\{0, -2\}$.
- Berechne $x_1 = f(x_0)$ und $x_2 = f(f(x_0)) \mod N$, wobei $f(x) = x^2 + c$ ist.
- Falls $1 < n = \text{ggT}(x_2 - x_1, N) \rightarrow$ Stop!
- Sonst $x_2 = f(x_1)$ und $x_4 = f(f(x_2)) \mod N$
- Falls $1 < n = \text{ggT}(x_4 - x_2, N) \rightarrow$ Stop!
- Sonst $x_3 = f(x_2)$ und $x_6 = f(f(x_4)) \mod N$
- Falls $1 < n = \text{ggT}(x_6 - x_3, N) \rightarrow$ Stop!

usw.
Der Fall $n = N$ tritt selten auf, aber falls, soll man wohl ein neues c wählen.
Es gibt noch die sogenannte Pollard $p - 1$ Methode, die aber eher nicht so bei RSA funktionieren soll.

1.1.2 KNJ-Factorization

Das ist eine Methode die ich in einem Paper von 2015 gefunden habe. Der dahinterstehende Algorithmus ist nicht probabilistisch, gibt uns also immer einen bzw. beide Primfaktoren aus. Im Großen und Ganzen ist es eine verbesserte Ausprobiermethode (sprich teste alle Zahlen von 2 bis $\lfloor\sqrt{N}\rfloor$.) Hier werden nur die Zahlen die ungerade und prim sind getestet. Der Algorithmus lautet wie folgt

- Berechne $X = \lfloor\sqrt{N}\rfloor$
- Prüfe ob X gerade, wenn ja, dann $X \to X + 1$
- Prüfe ob X prim, wenn ja, dann
- Berechne Y = N/X
- Falls Y ganzzahlig $\to$ Stop!
- Sonst $X \to X - 2$
- Wenn nein, dann $X \to X - 2$
- Wiederhole ab Primzahlcheck solange bis Y ganzzahlig.

Ich weiß aber nicht wie gut der tatsächlich ist, die Autoren des Papers meinen jedenfalls das er recht gut ist, also vielleicht einen Versuch wert.

1.1.3 Sieb

Das allgemeine Verfahren hat folgenden Algorithmus

- Wähle $x \in \mathbb{Z}$ sodass ggT$(x, N) = 1$ und berechne $x^2 \mod N$
- Suche $y \in \mathbb{Z}$ mit $x^2 \equiv y^2 \mod N$. (*)
- Bestimme $a =$ ggT$(y - x, N)$. Falls $a \in \{1, N\}$, wähle neues x sonst gib a aus.

Die große Schwierigkeit dieses Algorithmus ist (*). Weiterhin ist dieses Verfahren probabilistisch. Hier müsste ich ein wenig in die Theorie gehen, wie man y findet. Es soll aber erstmal ein kurzer Überblick sein, jedenfalls kann man y durch folgenden Algorithmus bestimmen

- Definiere $f(X) = (X + \lfloor\sqrt{N}\rfloor)^2 - N$
- Wähle Schranke B und bestimme alle Primzahlen $r \leq B$ sodass es ein $k \in \mathbb{Z}$ gibt mit $k^2 \equiv N \mod r$ (Faktorbasis $p_1, \ldots, p_n$) und eine Zahl $S \in \mathbb{Z}$ und $I = \{-S, -S+1, \ldots, S-1, S\}$.
- Berechne $f(a)$ für alle $a \in I$ und notiere das Vorzeichen in $v_0(a)$. Setze weiterhin $U(a) = |f(a)|$.
- Für alle Primzahlpotenzen $p_i^j \leq B$ mit p_i in der Faktorbasis bestimme Lösungen $a_{p_i^j}$ von $f(X) \equiv 0 \mod p_i^j$.
- Ersetze für alle $k \in \mathbb{Z}$ mit $|a_{p_i} + kp_i^j| \leq S$ den Eintrag $U(a_{p_i} + kp_i^j)$ durch $U(a_{p_i} + kp_i^j)/p_i$ und erhöhe $v_i(a_{p_i} + kp_i^j)$ um 1. Diesen Schritt der Reihe nach durchführen, also erst für $j = 1$ dann $j = 2$ usw.
- Für jedes $a \in I$ mit $U(a) = 1$ bilde Vektoren $(v_0(a), \ldots, v_n(a))$. Schreibe diese Vektoren in Matrix

$$A = \begin{pmatrix} v_0(a_1) & \ldots & v_0(a_m) \\ v_1(a_1) & \ldots & v_1(a_m) \\ & \vdots & \\ v_n(a_1) & \ldots & v_n(a_m) \end{pmatrix} = \begin{pmatrix} v_{01} & \ldots & v_{0m} \\ v_{11} & \ldots & v_{1m} \\ & \vdots & \\ v_{n1} & \ldots & v_{nm} \end{pmatrix}$$

- Löse $A\mu = 0 \mod 2$ mit $\mu_j \neq 0$ für wenigstens ein j.
- Wenn jetzt

$$x = \prod_{i=1}^{m} x_i^{\mu_i} \text{ und } y = \prod_{i=0}^{m} p_i^{\frac{1}{2}\sum_{j=1}^{n} \mu_j v_{ij}},$$

dann ist $x^2 \equiv y^2 \mod N$.

1.2 Wiener-Attacke

Die Wiener-Attacke beruht auf der Idee, das wenn der geheime Schlüssel $d < \frac{1}{3}N^{1/4}$ ist, so kann er eventuell als Nenner eines Näherungsbruches von $\frac{e}{N}$ auftreten. Soll heißen, wenn die Bedingung an d erfüllt ist, ist $\frac{r}{d}$ ein Näherungsbruch von $\frac{e}{n}$, wobei $r = \frac{ed-1}{\varphi(N)}$. Die Wiener-Attacke folgt dann folgendem Schema

- Berechne alle Näherungsbrüche $\frac{k}{l}$ von $\frac{e}{N}$.
- Berechne $\varphi_N = \frac{el-1}{k}$
- Ermittle Nullstellen p_0, q_0 des Polynoms

$$X^2 - (n - \varphi_N + 1)X + n$$

- Falls $N = p_0 q_0$ dann ist $l = d$ und $k = r$. Sonst wähle nächsten Näherungsbruch und beginne von vorne.

Dabei ist anzumerken das die Attacke auch funktionieren kann, selbst wenn d obige Bedingung nicht erfüllt, es ist dann halt nur nicht beweisbar.
Die Näherungsbrüche bestimmt man wie folgt

- Bestimme Kettenbruch

$$[a_0, \ldots, a_m] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \frac{1}{a_n}}}}$$

  von $\frac{e}{N}$. Da das eine rationale Zahl ist, kann es effizient mittels euklidischem Algorithmus berechnet werden durch

$$\left\{\begin{array}{cc} e = a_0 N + r_1, & 0 \leq r_1 < N \\ N = a_1 r_1 + r_2, & 0 < r_2 < r_1 \\ r_1 = a_2 r_2 + r_3, & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{m-2} = a_{m-1} r_{m-1} + r_m & 0 < r_n < r_{n-1} \\ r_{m-1} = a_m r_m & \end{array}\right\}.$$

  Dabei sind a_i und r_i ganze Zahlen.

- Jetzt können die Näherungsbrüche durch Rekursion berechnet werden

$$k_0 = a_0; l_0 = 1; k_1 = a_1 a_0 + 1; l_1 = a_1$$

$$k_i = a_i k_{i-1} + k_{i-2}; l_i = a_i l_{i-1} + l_{i-2}$$

1.3 Gitterbasierte Attacken

Diese Art von Attacken basieren im Großen und Ganzen auf der LLL-Reduktion zum finden von einer kurzen Gitterbasis um damit in einem bestimmten, durch Polynomkoeffizienten definierten, Gitter ein Polynom mit kleinenm Koeffizienten zu bestimmen, sodass man von diesem dann durch Nullstellenbestimmungung den privaten Schlüssel d zu finden. "Leider"funktionieren sie nur, wenn $d < N^{0.292}$, soweit ich es bisher rausfinden konnte, aber dann auch unglaublich schnell. Hierzu gibt es aber eine ganze Menge Material was ich erstmal durcharbeiten und verstehen muss. Fortsetzung folgt.