Chapter 2

Cryptanalysis of RSA

"Cryptanalysis of RSA" *doesn't break* the cipher—except in a few exceptional situations—but traces out the framework for applying it in a secure way according to our best judgment. In particular it helps avoiding some traps. We want answers to the questions:

- Do there exist sufficiently many keys to evade an exhaustion attack?
- Which mathematical results might lead to breaking an RSA ciphertext? Or to a computation of the private key?
- How to choose the parameters in order to avoid weaknesses?

There is a good overview in:

> D. Boneh: *Twenty years of attacks on the RSA cryptosystem.* Notices of the American Mathematical Society 46 (1999), 203–213.

2.1 The Prime Number Theorem

Let $\pi(x)$ be the number of primes $p \leq x$. Somewhat more generally let $\pi_{a,b}(x)$ be the number of primes $p \leq x$ of the form $p = ak + b$ (in other words: congruent to b modulo a). The prime number theorem states the asymptotic relation ()

$$\pi_{a,b}(x) \sim \frac{1}{\varphi(a)} \cdot \frac{x}{\ln(x)}$$

provided a and b are coprime. The special case $a = 1$, $b = 0$, is:

$$\pi(x) \sim \frac{x}{\ln(x)}.$$

There are many theoretical and empirical results concerning the quality of this approximation. An instance is a formula by Rosser and Schoenfeld:

$$\frac{x}{\ln(x)} \cdot \left(1 + \frac{1}{2\ln(x)}\right) < \pi(x) < \frac{x}{\ln(x)} \cdot \left(1 + \frac{3}{2\ln(x)}\right) \quad \text{for } x \geq 59.$$

The prime number theorem helps for answering the following questions (albeit not completely exactly):

How many prime numbers $< 2^k$ do exist?

Answer: $\pi(2^k)$, that is about

$$\frac{2^k}{k \cdot \ln(2)},$$

at least (for $k \geq 6$)

$$\frac{2^k}{k \cdot \ln(2)} \cdot \left(1 + \frac{1}{2k\ln(2)}\right).$$

For $k = 128$ this number is about $3.8 \cdot 10^{36}$, for $k = 256$, about $6.5 \cdot 10^{74}$.

How many k-bit primes do exist?

Answer: $\pi(2^k) - \pi(2^{k-1})$, that is about

$$\frac{2^k}{k \cdot \ln(2)} - \frac{2^{k-1}}{(k-1) \cdot \ln(2)} = \frac{2^{k-1}}{\ln(2)} \cdot \frac{k-2}{k(k-1)} \approx \frac{1}{2} \cdot \pi(2^k).$$

For $k = 128$ this amounts to about $1.9 \cdot 10^{36}$, for $k = 256$, to about $3.2 \cdot 10^{74}$. In other words, a randomly chosen k-bit integer is prime with probability

$$\frac{\pi(2^k) - \pi(2^{k-1})}{2^{k-1}} \approx \frac{\pi(2^k)}{2^k} \approx \frac{1}{k \cdot \ln(2)} \approx \frac{1.44}{k}.$$

For $k = 256$ this is about 0.0056.

The inequality

$$\pi(2^k) - \pi(2^{k-1}) > 0.71867 \cdot \frac{2^k}{k} \qquad \text{for } k \geq 21.$$

gives a reliable lower bound.

In any case the number of primes of size relevant for RSA is huge and makes an exhaustion attack completely obsolete.

Special Primes

Often cryptologists want their primes to have special properties:

Definition A **special prime** (or **safe prime**) is a prime of the form $p = 2p' + 1$ where p' is an odd prime (then p' is also called a GERMAIN prime).

Remark Let p be special. Then $p \equiv 3 \pmod 4$, for $p = 2p' + 1 \equiv 2 \cdot a + 1$ where $a = 1$ or 3.

Definition A **superspecial prime** is a prime of the form $p = 2p'+1$ where $p' = 2p'' + 1$ is a special prime.

Examples The two smallest superspecial primes are $p = 23$ (with $p' = 11$, $p'' = 5$) and $q = 47$ (with $q' = 23$, $q'' = 11$).

Are there enough primes to fulfill these special or superspecial requests?

Frankly speaking, there is no exact answer. However we can give (unproven!) fairly exact estimates for these numbers:

- As we saw, a (positive) k-bit integer is prime with probability $\frac{\alpha}{k}$ where $\alpha \approx 1.44$.
- If $p = 2p' + 1$ is special, then p' is a $k/2$-bit integer, and is prime (heuristically, but in fact unknown) with probability $\frac{2\alpha}{k}$.
- Thus we estimate that a random k-bit integer is a special prime with probability $\frac{\alpha}{k} \cdot \frac{2\alpha}{k} = \frac{2\alpha^2}{k^2}$, and we expect that $\frac{\alpha^2}{k^2} \cdot 2^k$ of the 2^{k-1} k-bit integers are special primes (assuming that the "events" p prime and $(p-1)/2$ prime are independent).
- Moreover $p'' = (p' - 1)/2$ is a $k/4$-bit integer, hence prime with probability $\frac{4\alpha}{k}$.

- This makes up for a probability of

$$\frac{\alpha}{k} \cdot \frac{2\alpha}{k} \cdot \frac{4\alpha}{k} = \frac{8\alpha^3}{k^3}$$

  for a k-bit integer to be a superspecial prime.

- By this consideration—although we have no mathematical proof for it—we expect that

$$\frac{\alpha^3}{k^3} \cdot 2^{k+2}$$

  of the 2^{k-1} k-bit integers are superspecial primes.

- For $k = 256 = 2^8$ (and $\alpha^2 \approx 2$, $\alpha^3 \approx 3$) we may hope for

$$\begin{aligned} 2 \cdot 2^{256} \cdot 2^{-16} &\approx 3.5 \cdot 10^{72} && \text{special primes,} \\ 3 \cdot 2^{258} \cdot 2^{-24} &\approx 8.3 \cdot 10^{70} && \text{superspecial primes.} \end{aligned}$$

Extensions

Let p_n be the n-th prime, thus $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, ... Let $\vartheta(x)$ be the sum of the logarithms of the primes $\leq x$,

$$\vartheta(x) = \sum_{p \leq x,\, p \text{ prime}} \ln(p).$$

Then we have the asymyptotic formulas

$$\begin{aligned} p_n &\sim n \cdot \ln(n), \\ \vartheta(x) &\sim x, \end{aligned}$$

and the error bounds due to Rosser/Schoenfeld:
(1)

$$n \cdot \left(\ln(n) + \ln\ln(n) - \frac{3}{2} \right) < p_n < n \cdot \left(\ln(n) + \ln\ln(n) - \frac{1}{2} \right) \quad \text{for } n \geq 20,$$

(2)

$$x \cdot \left(1 - \frac{1}{\ln(x)} \right) < \vartheta(x) < x \cdot \left(1 - \frac{1}{2\ln(x)} \right) \quad \text{for } n \geq 41.$$

For a proof of the prime number theorem see any textbook on analytic number theory, for example

> Apostol, T. M. *Introduction to Analytic Number Theory.* Springer-Verlag, New York 1976.

2.2 Computing the Key and Factorization

Question: *How to compute the private RSA exponent d, given the public exponent e and the module n?*

Answer: Each of the following tasks (A) – (D) is efficiently reducible to each of the other ones:

(A) Computing the private key d.

(B) Computing $\lambda(n)$ (CARMICHAEL function).

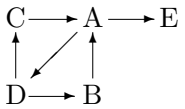
(C) Computing $\varphi(n)$ (EULER function).

(D) Factoring n.

Breaking RSA is the (possibly properly) easier task:

(E) Computing e-th roots in $\mathbb{Z}/n\mathbb{Z}$.

The "proof" (not an exact proof in the mathematical sense) follows the roadmap:



We always assume that n and the public exponent e are known, and $n = p_1 \cdots p_r$ with different primes $p_1, \ldots, p_r$.

Clearly "A $\longrightarrow$ E": Taking an e-th root means raising to the d-th power. So if d is known, computing e-th roots is easy.

Note that the converse implication is unknown: *Breaking RSA could be easier than factoring.*

"D $\longrightarrow$ C": $\varphi(n) = (p_1 - 1) \cdots (p_r - 1)$.

"D $\longrightarrow$ B": $\lambda(n) = \mathrm{kgV}(p_1 - 1, \ldots, p_r - 1)$.

"B $\longrightarrow$ A": Compute d by congruence division from $de \equiv 1 \pmod{\lambda(n)}$.

"C $\longrightarrow$ A": Since $\varphi(n)$ has exactly the same prime factors as $\lambda(n)$, also $\varphi(n)$ is coprime with e. From $de \equiv 1 \pmod{\varphi(n)}$ we get a solution for d by congruence division. This might not be the "true" exponent, but works in the same way as private key since a forteriori $de \equiv 1 \pmod{\lambda(n)}$.

"A $\longrightarrow$ D" is significantly more involved. Moreover we only construct a probabilistic algorithm.

Preliminary Remarks

1. *It suffices to decompose n into two proper factors.*

   (a) Let $n = n_1 n_2$ be a proper decomposition, and assume for simplicity that $n_1 = p_1 \cdots p_s$ with $1 < s < r$. Then

   $$\lambda(n_1) = \mathrm{kgV}(p_1 - 1, \ldots, p_s - 1) \mid \mathrm{kgV}(p_1 - 1, \ldots, p_r - 1) = \lambda(n),$$

   thus also $de \equiv 1 \pmod{\lambda(n_1)}$. This reduces the problem to the analoguous ones for n_1 and n_2.

   (b) Since the number of prime factors of n is at most $\log_2(n)$ the recursive reduction suggested by (a) is efficient.

2. *How can a residue class $w \in \mathbb{Z}/n\mathbb{Z}$ help with factoring n?*

   (a) Finding a $w \in [1 \ldots n-1]$ with $\gcd(w, n) > 1$ decomposes n since $\gcd(w, n)$ is a proper divisor of n.

   (b) Finding a $w \in [2 \ldots n-2]$ with $w^2 \equiv 1 \pmod{n}$ (that is a non-trivial square root of 1 in $\mathbb{Z}/n\mathbb{Z}$) likewise decomposes n:
   Since $n \mid w^2 - 1 = (w+1)(w-1)$ and $n \nmid w \pm 1$ we have $\gcd(n, w+1) > 1$, and this decomposes n by (a).

Now let (d, e) be a pair of RSA exponents. Then also $u := ed - 1 = k \cdot \lambda(n)$ is known (with unknown k and $\lambda(n)$). Since $\lambda(n)$ is even we may write

$$u = r \cdot 2^s \qquad \text{with } s \geq 1 \text{ and } r \text{ odd.}$$

If we choose a random $w \in [1 \ldots n-1]$, then we have to deal with two possibilities:

- $\gcd(w, n) > 1$—then n is decomposed.
- $\gcd(w, n) = 1$—then $w^{r2^s} \equiv 1 \pmod{n}$.

In the second case we efficiently find the minimal $t \geq 0$ with

$$w^{r2^t} \equiv 1 \pmod{n}.$$

Again we distinguish two cases:

- $t = 0$—bad luck, choose another w.
- $t > 0$—then $w^{r2^{t-1}}$ is a square root $\neq 1$ of 1 in $\mathbb{Z}/n\mathbb{Z}$.

In the second case we distiguish:

- $w^{r2^{t-1}} \equiv -1 \pmod{n}$—bad luck, choose another w.

- $w^{r2^{t-1}} \not\equiv -1 \pmod{n}$—then n is decomposed by preliminary remark 2.

Thus every choice of $w \in [1 \ldots n-1]$ has one of four possible outcomes, two of them decompose n, and the other two flop. Denote the last two events by

$$\begin{aligned} &(\mathrm{E}_{n,u}(w)/\mathrm{I}) \qquad w^r \equiv 1 \pmod{n} \\ &(\mathrm{E}_{n,u}(w)/\mathrm{II}) \quad w^{r2^{t-1}} \equiv -1 \pmod{n} \quad \text{for a } t \text{ with } 1 \le t \le s. \end{aligned}$$

Altogether this yields a tree-like structure:

$w \in [1 \ldots n-1] \longrightarrow$
  $\gcd(w,n) > 1 \longrightarrow n$ decomposed SUCCESS
  $w \in \mathbb{M}_n \longrightarrow$
    $w^r \equiv 1 \pmod{n} \longrightarrow (\mathrm{E}_{n,u}(w)/\mathrm{I})$ FLOP
    $w^r \not\equiv 1 \pmod{n} \longrightarrow$
      $w^{r2^{t-1}} \equiv -1 \pmod{n} \longrightarrow (\mathrm{E}_{n,u}(w)/\mathrm{II})$ FLOP
      $w^{r2^{t-1}} \not\equiv -1 \pmod{n} \longrightarrow n$ decomposed SUCCESS

Thus our procedure decomposes n "with high probability" if there are only "few" "bad" integers w with $(\mathrm{E}_{n,u}(w)/\mathrm{I,II})$. The next section will provide an upper bound for their number.

2.3 The Probability of Flops

Let $n \in \mathbb{N}_3$. Furthermore assume that $u \in \mathbb{N}_2$ is even, $u = r \cdot 2^s$ with odd r and $s \geq 1$. We introduce the sets:

$$\begin{aligned}
A_u^{(0)} &= B_u^{(0)} := \{w \in \mathbb{M}_n \mid w^r = 1\} \quad [\text{case } (\mathrm{E}_{n,u}/\mathrm{I})], \\
A_u^{(t)} &:= \{w \in \mathbb{M}_n \mid w^{r \cdot 2^t} = 1, w^{r \cdot 2^{t-1}} \neq 1\} \quad \text{for } 1 \leq t \leq s, \\
B_u^{(t)} &:= \{w \in A_u^{(t)} \mid w^{r \cdot 2^{t-1}} = -1\} \quad [\text{case } (\mathrm{E}_{n,u}/\mathrm{II})], \\
A_u &:= \bigcup_{t=0}^{s} A_u^{(t)} = \{w \in \mathbb{M}_n \mid w^u = 1\}, \\
B_u &:= \bigcup_{t=0}^{s} B_u^{(t)} \quad [\text{case } (\mathrm{E}_{n,u}) \text{ (I or II)}]. \\
C_0 &:= \{w \in \mathbb{M}_n \mid \operatorname{ord} w \text{ odd}\}, \\
C_1 &:= \{w \in \mathbb{M}_n \mid -1 \in \langle w \rangle\}, \\
C &:= C_0 \cup C_1.
\end{aligned}$$

Remarks

- $A_u^0 \leq A_u \leq \mathbb{M}_n$ are subgroups, as are $A_u^0 \leq C_0 \leq \mathbb{M}_n$.
- $B_u^{(t)} = A_u^{(t)} \cap C$ for $t = 0, \ldots, s$, since a cyclic group $\langle w \rangle$ can contain at most one square root of 1 in addition to 1 itself.
- Hence also $B_u = A_u \cap C$.
- B_u is the exceptional set of "bad" integers with $(\mathrm{E}_{n,u})$ from Section 2.2 that flop with factoring n. The following proposition upper bounds by $\frac{1}{2}$ the probability of hitting an element of this set by pure chance. If we try k random candidate integers the probability of not factoring n is $< 1/2^k$, hence *extremely* small even for moderate sizes of k

Proposition 4 *Let n be odd and not a prime power. Let $u = r \cdot 2^s$ be a multiple of $\lambda(n)$ with odd r. Then*

$$\#B_u \leq \frac{1}{2} \cdot \varphi(n).$$

Proof. By the following lemma C, and a forteriori B_u, is contained in a proper subgroup of $\mathbb{M}_n$. $\diamond$

Lemma 1 (Dixon, AMM 1984) *Let $n \in \mathbb{N}_3$. Assume $\langle C \rangle = \mathbb{M}_n$. Then n is a prime power or even.*

Proof. For this proof let $\lambda(n) = r \cdot 2^s$ with odd r. (Since $n \geq 3$, we have $s \geq 1$. The "old" meanings of r and s don't occur in this proof.) Consider the map

$$h : \mathbb{M}_n \longrightarrow \mathbb{M}_n, \qquad w \mapsto w^{r \cdot 2^{s-1}}.$$

This h is a group homomorphism with $h(\mathbb{M}_n) \subseteq \{v \in \mathbb{M}_n \mid v^2 = 1\}$ (group of square roots of 1 mod$n$). Since the $w \in C_0$ have odd order $h(C_0) \subseteq \{1\}$.

For $w \in C_1$ we have $h(w) \in \langle w \rangle$ and $h(w)^2 = 1$, hence $h(w)$ is one of the two roots of unity $\pm 1 \in \langle w \rangle$.

Together we have $h(C) \subseteq \{\pm 1\}$.

If n is not a prime power (and a forteriori not a prime) there is a decomposition $n = pq$ into coprime factors $p, q \in \mathbb{N}_2$. Since $2^s | \lambda(n) = \operatorname{lcm}(\lambda(p), \lambda(q))$ we may assume $2^s | \lambda(p)$. The chinese remainder theorem provides a $w \in \mathbb{M}_n$ with $w \equiv 1 \pmod{q}$ such that $w \bmod p$ has order 2^s. Then $h(w) \not\equiv 1 \pmod{p}$, a forteriori $h(w) \neq 1$. Since $h(w) \equiv 1 \pmod{q}$ we also have $h(w) \neq -1$—except when $q = 2$.

Therefore if n is not even nor a prime power we have the contradiction $h(\mathbb{M}_n) \not\subseteq \{\pm 1\}$. $\diamond$

This also completes the missing step of Section 2.2: Who knows the private RSA key is able to factor the module n.

2.4 Factoring Algorithms

A crucial question for the security of RSA is: *How fast can we factorize large integers?*

- There are "fast" factoring algorithms for integers of the form $a^b \pm c$ with "small" values a and c, the most prominent examples are the MERSENNE and FERMAT primes $2^b \pm 1$. The probability that the generation of RSA keys from random primes yields such a module is extremely low and usually neglected.

- FERMAT factoring of n: Find an integer $a \geq \sqrt{n}$ such that $a^2 - n$ is a square $= b^2$. This yields a decomposition

$$n = a^2 - b^2 = (a + b)(a - b).$$

  Example: $n = 97343$, $\sqrt{n} \approx 311.998$, $312^2 - n = 1$, $n = 313 \cdot 311$. This method is efficient provided that we find an a close to $\sqrt{n}$, or $a^2 \approx n$. In the case $n = pq$ of two factors this means a small difference $|p - q|$. (Un-) fortunately finding a seems to be hard.

- The fastest general purpose factoring algorithms
  - number field sieve (SILVERMAN 1987, POMERANCE 1988, A. K. LENSTRA/ H. W. LENSTRA/ MANASSE/ POLLARD 1990),
  - elliptic curve factoring (H. W. LENSTRA 1987, ATKIN/ MORAIN 1993),

  need time of size

$$L_n := e^{\sqrt[3]{\ln n \cdot (\ln \ln n)^2}},$$

  hence are "subexponential", but also "superpolynomial". Anyway they show that *factoring is a significantly more efficient attack on RSA than exhaustion ("brute force").*

This results in the following estimates for factoring times:

integer	bits	decimal places	expense (MIPS years)	status
rsa120	399	120	100	< 1 weak on a PC
rsa154	512	154	100 000	TE RIELE 1999
rsa200	665	200	$(*)$	FRANKE 2005
	1024	308	10^{11}	insecure
	2048	616	10^{15}	for short-term security

$(*)$ 80 CPUs à 2.2 GHz in 4.5 months

When we extrapolate these estimates we should note:

- they are rough approximations only,
- they hold only as long as nobody finds significantly faster factoring algorithms.

Remember that the existence of a polynomial factoring algorithm is an *open problem.*

Some recent developments are already incorporated into the table:

- A paper by A. K. Lenstra/ E. Verheul, *Selecting cryptographic key sizes* summarizes the state of the art in the year 2000 and extrapolates it.
- A proposal by Bernstein, *Circuits for integer factorization* triples (!) the length of integers that can be factorized with a given expense, using the fastest factoring algorithms.
- Special-purpose hardware designs by Shamir and his collaborators:
  - TWINKLE (The Weizmann Institute Key Locating Machine) (1999) is the realization in hardware of an idea by Lehmer from the 1930s that accelerates factoring 100–1000 times,
  - TWIRL (The Weizmann Institute Relation Locator) (2003) accelerates factoring another 1000–10000 times following Bernstein's idea.

  Taken together these approaches make factoring 10^6 (or 2^{20}) times faster using the number field sieve. However the order of magnitude L_n of the complexity is unaffected.

This progress lets the Lenstra/ Verheul estimates look overly optimistic. 1024-*bit keys should no longer be used.* 2048-bit keys might be secure enough to protect information for a few years.

Recommendation: Construct your RSA module $n = pq$ from primes p and q that have bit lengths of at least 2048 bits, and choose them such that also their difference $|p - q|$ has a bit length of about 2048 bits.

2.5 Iteration Attack

Consider a bijective map $E\colon M \longrightarrow M$ of a finite set M onto itself and its inverse $D = E^{-1}$ (think of E as an encryption function). Then E is an element of the full symmetric group $\mathfrak{S}(M)$ that has the (huge) order $\#\mathfrak{S}(M) = (\#M)!$. Nevertheless this group is finite, thus there is an $s \in \mathbb{N}_1$ with $E^s = \mathbf{1}_M$, hence

$$D = E^{s-1}.$$

As a consequence an attacker can compute D from E by sufficiently many iterations. This attack is relevant only for asymmetric ciphers where the attacker knows E. The only protection against it is *to choose the order of E, the smallest $s \geq 1$ with $E^s = \mathbf{1}_M$, as large as possible.*

The Example of RSA

Let $M = \mathbb{Z}/n\mathbb{Z}$, then $\#\mathfrak{S}(M) = n!$, where n itself is a very large integer. The attacker could compute $E^{n!-1}$, but even the fastest power algorithm is not fast enough to accomplish this task in this universe. So the attack doesn't seem to put RSA into immediate danger.

However, as a closer look reveals, RSA encryption functions are contained in a significantly smaller subgroup of $\mathfrak{S}(M)$—fortunately the attacker doesn't know its order. To see this consider the map

$$\Phi\colon \mathbb{N} \longrightarrow \operatorname{map}(M, M), \quad e \mapsto E_e \quad \text{with } E_e(a) = a^e \bmod n.$$

Here are some of its properties:

1. For $e, f \in \mathbb{N}$ we have $E_{ef} = E_e \circ E_f$ since $a^{ef} \equiv (a^f)^e \pmod{n}$ for all $a \in M$. Hence Φ is a homomorphism of the multiplicative semigroup $\mathbb{N}$.

2. If $e \equiv f \pmod{\lambda(n)}$, then $E_e = E_f$: Assume $f = e + k\lambda(n)$, then $a^f = a^{e+k\lambda(n)} \equiv a^e \pmod{n}$ for all $a \in M$.

3. If $e \bmod \lambda(n)$ is invertible, then E_e is bijective: Assume $de \equiv 1 \pmod{\lambda(n)}$, then $E_d \circ E_e = E_1 = \mathbf{1}_M$. Hence the map

   $$\bar{\Phi}\colon \mathbb{M}_{\lambda(n)} \longrightarrow \mathfrak{S}(M)$$

   induced by Φ is a group homomorphism.

4. $\bar{\Phi}$ is injective: For if $\Phi(e) = E_e = \mathbf{1}_M$, then $a^e \equiv a \pmod{n}$ for all $a \in M$, hence $a^{e-1} \equiv 1 \pmod{n}$ for all $a \in \mathbb{M}_n$, hence $\lambda(n) | e-1$, thus $e \equiv 1 \pmod{\lambda(n)}$.

These remarks prove:

Proposition 5 *The RSA encryption functions E_e form a subgroup $H_n \leq \mathfrak{S}(M)$ that is isomorphic with $\mathbb{M}_{\lambda(n)}$ and has order $\varphi(\lambda(n))$ and exponent $\lambda(\lambda(n))$.*

Of course the order of a single encryption function E_e could be even much smaller: All we can say is that the cyclic subgroup $\langle e \rangle \leq \mathbb{M}_{\lambda(n)}$ has order $s := \operatorname{ord}(e) \mid \lambda(\lambda(n))$.

This observation raises two problems:

1. How large is $\lambda(\lambda(n))$?
2. Under what conditions is $\operatorname{ord}(e) = \lambda(\lambda(n))$? Or at least not significantly smaller?

Answer to 1 (without proof): "In general" $\lambda(\lambda(n)) \approx \frac{n}{8}$.

If we want to be sure about this we should choose p, q as special primes $p = 2p' + 1$, $q = 2q' + 1$ with different primes $p', q' \geq 3$. Then for $n = pq$ we have

$$\lambda(n) = \operatorname{kgV}(2p', 2q') = 2p'q' = \frac{(p-1)(q-1)}{2} \approx \frac{n}{2}.$$

If moreover p and q are superspecial primes, that is, $p' = 2p'' + 1$ and $q' = 2q'' + 1$ are special primes too, then

$$\lambda(\lambda(n)) = 2p''q'' = \frac{(p-3)(q-3)}{8} \approx \frac{n}{8}.$$

By the prime number theorem, see Section 2.1, we may expect that superspecial primes exist in astronomic quantities.

Answer to 2: in most cases (also without general proof).

Again, if we want to be sure, we should confine our choices to special or even superspecial primes. We use some elementary results on finite groups, see Lemmas 21, 22, and 23 of Appendix A.10.

Let p be an odd prime number. In the additive cyclic group $\mathbb{Z}/2p\mathbb{Z}$ we consider the subsets:

$$\begin{aligned} E_p &= \{a \bmod 2p \mid 0 \leq a < p,\ a \text{ even}\} - \{0\}, \\ O_p &= \{a \bmod 2p \mid 0 \leq a < p,\ a \text{ odd}\} - \{p\}. \end{aligned}$$

Clearly, $\mathbb{Z}/2p\mathbb{Z} = \{0, p\} \cup E_p \cup O_p$, and

$$\#E_p = \#O_p = p - 1.$$

The order of an element $x \in \mathbb{Z}/2p\mathbb{Z}$ is

$$\operatorname{ord} x = \begin{cases} 1 & \iff x = 0, \\ 2 & \iff x = p, \\ p & \iff x \in E_p, \\ 2p & \iff x \in O_p. \end{cases}$$

We transfer this result to an abstract cyclic group $\mathcal{Z}_{2p}$ with generating element g via the isomorphism

$$\tau : \mathbb{Z}/2p\mathbb{Z} \longrightarrow \mathcal{Z}_{2p}, \quad x \mapsto g^x.$$

Let $\mathcal{E}_p = \tau E_P$ and $\mathcal{O}_p = \tau O_P$. Then the result is:

Lemma 2 *The order of an element $h \in \mathcal{Z}_{2p}$ is*

$$\operatorname{ord} h = \begin{cases} 1 & \Longleftrightarrow h = \mathbf{1}, \\ 2 & \Longleftrightarrow h = g^p, \\ p & \Longleftrightarrow h \in \mathcal{E}_p, \\ 2p & \Longleftrightarrow h \in \mathcal{O}_p. \end{cases}$$

Next we study the orders of the elements of the direct product $\mathcal{Z}_{2p} \times \mathcal{Z}_{2q}$ for two different odd primes p and q. Applying Lemma 21 we see that the order of a pair (g, h) for $g \in \mathcal{Z}_{2p}$ and $h \in \mathcal{Z}_{2q}$ is given by the following table:

		$\operatorname{ord} g =$			
		1	2	p	$2p$
$\operatorname{ord} h =$	1	1	2	p	$2p$
	2	2	2	$2p$	$2p$
	q	q	$2q$	pq	$2pq$
	$2q$	$2q$	$2q$	$2pq$	$2pq$

An obvious count yields:

Proposition 6 *Let p and q be two different odd primes. Then the direct product group $\mathcal{Z}_{2p} \times \mathcal{Z}_{2q}$ has*

(i) 1 *element of order* 1,

(ii) 3 *elements of order* 2,

(iii) $p - 1$ *elements of order* p,

(iv) $3 \cdot (p - 1)$ *elements of order* $2p$,

(v) $q - 1$ *elements of order* q,

(vi) $3 \cdot (q - 1)$ *elements of order* $2q$,

(vii) $(p - 1) \cdot (q - 1)$ *elements of order* pq,

(viii) $3 \cdot (p - 1) \cdot (q - 1)$ *elements of order* $2pq$.

Again let p be a prime number. Then the multiplicative group $\mathbb{M}_p = (\mathbb{Z}/p\mathbb{Z})^\times$ of the finite field $\mathbb{Z}/p\mathbb{Z}$ is cyclic of order $p-1$. Let q be a prime different from p and let $n = p \cdot q$. Then by the Chinese Remainder Theorem $\mathbb{M}_n \cong \mathbb{M}_p \times \mathbb{M}_q$ is (up to isomorphy) the direct product of two cyclic groups of orders $p-1$ and $q-1$. Hence:

Lemma 3 *Let $n = pq$ be the product of two different odd primes p and q. Then the multiplicative group $\mathbb{M}_n = (\mathbb{Z}/n\mathbb{Z})^\times$ of the quotient ring $\mathbb{Z}/n\mathbb{Z}$ has order $\varphi(n) = (p-1)(q-1)$ and exponent $\lambda(n) = \operatorname{lcm}(p-1, q-1)$. In particular $\mathbb{M}_n$ is not cyclic.*

The latter statement is due to the common divisor 2 of $p-1$ and $q-1$.

We now consider the case where $p = 2p'+1$ and $q = 2q'+1$ are special primes. Then

$$\varphi(n) = 4p'q' \quad \text{and} \quad \lambda(n) = 2p'q'.$$

By Proposition 5 the RSA encryption functions for the module $n = pq$ make up a group H_n isomorphic with $\mathbb{M}_{\lambda(n)}$. For special primes we therefore have by Theorem 2 in Appendix A.4:

Proposition 7 *Let $n = pq$ be the product of two different special primes $p = 2p'+1$ and $q = 2q'+1$. Then the RSA group*

$$H_n \cong \mathbb{M}_{\lambda(n)} \cong \mathcal{Z}_{p'-1} \times \mathcal{Z}_{q'-1}$$

is the product of two cyclic groups of orders $p'-1$ and $q'-1$.

In order to derive some more easy results we assume that p and q are superspecial primes, with $p' = 2p''+1$ and $q' = 2q''+1$. Then

$$H_n \cong \mathbb{M}_{\lambda(n)} \cong \mathcal{Z}_{2p''} \times \mathcal{Z}_{2q''},$$

and Proposition 6 applies for the primes p'' and q'':

Proposition 8 *Let $n = pq$ be the product of two different superspecial primes $p = 2p'+1$ and $q = 2q'+1$ with $p' = 2p''+1$ and $q' = 2q''+1$. Then the RSA group H_n consists of*

(i) *1 element of order 1,*

(ii) *3 elements of order 2,*

(iii) *$p''-1$ elements of order p'',*

(iv) *$3 \cdot (p''-1)$ elements of order $2p''$,*

(v) *$q''-1$ elements of order q'',*

(vi) $3 \cdot (q'' - 1)$ *elements of order* $2q''$,

(vii) $(p'' - 1) \cdot (q'' - 1)$ *elements of order* $p''q''$,

(viii) $3 \cdot (p'' - 1) \cdot (q'' - 1)$ *elements of order* $2p''q''$.

Since $2p''q'' = \lambda(\lambda(n))$ is the exponent of H_n we see that almost all of its elements have their orders near the maximum. More precisely the number of elements of order $< \frac{1}{2}\,\lambda(\lambda(n)) = p''q''$ is

$$1 + 3 + 4 \cdot (p'' - 1) + 4 \cdot (q'' - 1) = 4 \cdot (p'' + q'' - 1).$$

Corollary 1 *The number of elements of* H_n *with order* $< \frac{1}{2}\,\lambda(\lambda(n))$ *is* $p + q - 7$.

Proof. Note that $p'' = (p - 3)/4$. $\diamond$

Thus this number is $\approx 2 \cdot \sqrt{n}$ if p and q—as recommended in Section 2.4—are chosen near $\sqrt{n}$. Then the proportion of elements of "small" orders is $\approx 2/\sqrt{n}$, and this proportion asymptotically tends to 0 with growing values of n.

As a consequence we resume: *With negligeable exceptions* s *has the order of magnitude of* $n/8$. The best known general results are in Chapter 23 of Shparlinski's book, see the references for these lecture notes.

In addition to Section 2.2 we formulate the task

(F) Finding the order s of the encryption function.

In the sense of complexity theory we have the implication

$$\text{(F)} \longrightarrow \text{(A)}$$

but maybe not the reverse implication. If the order s is known, then $D = E^{s-1}$ and thus $d = e^{s-1}$ are efficiently computable. *Finding the order of the encryption function is at least as difficult as factoring the module.*

2.6 Breaking Single Ciphertexts

Breaking a single ciphertext (without necessarily computing the private key) could be even easier: For a given ciphertext c we could have $E_e^r(c) = c$ even if $E_e^r \neq \mathbf{1}_M$. If a is the corresponding plaintext, $c = E_e(a)$, then the cryptanalyst can compute:

$$E_e^{r-1}(c) = D_e(E_e^r(c)) = D_e(c) = a.$$

From a mathematical viewpoint we have the situation:

- The group $\mathbb{M}_{\lambda(n)}$ acts on the set $M = \mathbb{Z}/n\mathbb{Z}$, as does its cyclic subgroup $G := \langle e \rangle \leq \mathbb{M}_{\lambda(n)}$.
- For $a \in M$ the orbit is $G \cdot a = \{a^{e^k} \mid 0 \leq k < s\}$ (where s is the order of e in the multiplicative group $\mathbb{M}_{\lambda(n)}$).
- The stabilizer $G_a = \{f \in G \mid a^f \equiv a \pmod{n}\}$ is a subgroup of G. We have a natural bijective correspondence between the sets $G \cdot a$ and G/G_a.
- For the orbit length $t = \#G \cdot a$ we have

$$t = \frac{s}{\#G_a}, \qquad t|s|\lambda(\lambda(n))$$
$$E_e^r(c) = c \iff E_e^r(a) = a \iff t|r.$$

- $G \cdot c = G \cdot a$ and $\#G_c = \#G_a$. (The two stabilizers are conjugate.)
- Finding the orbit length t of a and c is at least as difficult as breaking the ciphertext c.

This suggests yet another problem:

3. Under what conditions is $t = s$, in other words, which stabilizers G_a are trivial? Or at least quite small?

Answer once more (without proof): in most cases. For superspecial primes p and q where $\lambda(\lambda(n)) = 2p''q''$ we expect by similar considerations as in Section 2.5 that $t < p''q''$ only for a negligeable set of exceptions.

The following two papers show how low is the risk of hitting a small orbit length by pure chance, enabling an iteration attack:

- J. J. Brennan/ Bruce Geist, Analysis of iterated modular exponentiation: The orbits of x^α mod N. **Designs, Codes and Cryptography** 13 (1998), 229–245.
- John B. Friedlander/ Carl Pomerance/ Igor E. Shparlinski, Period of the power generator and small values of Carmichael's function. **Mathematics of Computation** 70 (2001), 1591–1606, + 71 (2002), 1803–1806.

2.7 Re-Use of a Module

Question: What happens if two different participants use the same RSA module n?

In other words, A and B use (n, e_A) and (n, e_B) as public keys.

Obviously A and B may read each other's messages since both can factorize n and hence compute the other's private key. Thus a common module makes sense only in a cooperative situation where A and B absolutely trust each other.

However it's even worse: A message a sent to both A and B is readable *by everyone*. The ciphertexts are:

$$c_\mathrm{A} = a^{e_\mathrm{A}} \bmod n, \quad c_\mathrm{B} = a^{e_\mathrm{B}} \bmod n.$$

Assuming e_A and e_B coprime is no significant loss of generality. Then the attacker, using the extended Euclidean algorithm, finds coefficients x and y with

$$xe_\mathrm{A} + ye_\mathrm{B} = 1.$$

Necessarily x and y have opposite signs, assume $x < 0$. If $\gcd(c_\mathrm{A}, n) > 1$, then the attacker can decompose n and is done. Otherwise she computes

$$g := c_\mathrm{A}^{-1} \bmod n$$

by congruence division and gets

$$g^{-x} \cdot c_\mathrm{B}^{y} \equiv (a^{e_\mathrm{A}})^x \cdot (a^{e_\mathrm{B}})^y \equiv a \pmod{n},$$

breaking the ciphertext without computing the private keys d_A and d_B.

Hence the common module n is secure only when A and B trust each other and moreover keep the module secret. But in this situation it makes much more sense to use a symmetric cipher.

2.8 Small Exponents

Question: Is RSA in danger if someone chooses a small public exponent e?

The exponent $e = 1$ is nonsensical since it leaves plaintexts unencrypted.

The exponent $e = 2$ doesn't work for RSA since it is even and thus not coprime with $\lambda(n)$. Nevertheless the related RABIN cipher uses $e = 2$. Here the receiver of the message must be able to take square roots $\bmod n$, and this works since he knows the prime factors of n (see later). (By the way he must also be able to recognize the true plaintext among several different square roots.)

Same Message for Several Receivers

For RSA the smallest potentially suited exponent is $e = 3$. However it enables an attack that applies as soon as someone sends the same message a to *three* different receivers A, B, and C. Let their public keys be $(n_\mathrm{A}, 3)$, $(n_\mathrm{B}, 3)$, and $(n_\mathrm{C}, 3)$. Assume the modules n_A, n_B, and n_C are pairwise coprime, otherwise the attacker factorizes at least two of them and reads a. Then (with some luck) she intercepts three ciphertexts

$$c_\mathrm{A} = a^3 \bmod n_\mathrm{A}, \quad c_\mathrm{B} = a^3 \bmod n_\mathrm{B}, \quad c_\mathrm{C} = a^3 \bmod n_\mathrm{C},$$

with $0 \le a < n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}$, thus $a^3 < n_\mathrm{A} n_\mathrm{B} n_\mathrm{C}$. Using the chinese remainder algorithm she constructs an integer $\tilde{c} \in \mathbb{Z}$ with

$$0 \le \tilde{c} < n_\mathrm{A} n_\mathrm{B} n_\mathrm{C}$$

such that

$$\tilde{c} \equiv c_\mathrm{X} \bmod n_\mathrm{X} \quad \text{for } \mathrm{X} = \mathrm{A}, \mathrm{B}, \mathrm{C}.$$

By uniqueness $\tilde{c} = a^3$ in $\mathbb{Z}$. So she computes $a = \sqrt[3]{\tilde{c}}$ by taking the integer root in $\mathbb{Z}$. This is an efficient procedure. (In this situation she doesn't succeed with computing the private exponents.)

This attack obviously generalizes to other "small" shared public exponents e: If the same message is sent to e different people, then everybody can read it. This attack is not completely unrealistic: Think for example of fixed "protocol information" at the beginning of a larger message. Even in classical cryptography an important maxim was: *Never encrypt the same plaintext with different keys.*

In practice the exponent $e = 2^{16} + 1 = 65537$ is considered as sufficiently secure for "normal" situations.

Stereotypical Message Parts

Consider the key parameters (n, e, d). Imagine an attack with known plaintext that reads:

`Der heutige Tagesschluessel ist:********`

("The master key for today is: ...", example by Julia Dietrichs) with known (stereotypical) 32 byte part "`Der heutige Tagesschluessel ist:`", and unknown 8 byte part "`********`".

This message is encoded by the 8-bit character code ISO-8859-1 (used for German texts) as a sequence of 40 bytes or 320 bits, and for encryption by RSA interpreted as an integer $a \in [0 \dots n-1]$ (assume n has more then 320 bits, and $e = 3$). Decompose a as $a = b + x$ where b corresponds to the known, and x, to the unknown part. Since the latter forms the end of the message and consists of 64 bits we know $x < 2^{64}$. Encryption yields the ciphertext

$$c = a^e \bmod n = (b + x)^e \bmod n.$$

Hence the secret x is a root of the polynomial

$$(T + b)^e - c \in (\mathbb{Z}/n\mathbb{Z})[T].$$

At first sight this observation doesn't seem alarming since we know of no general efficient algorithms that compute roots. However algorithms for certain special cases exist, for instance:

> COPPERSMITH'S **algorithm**
> Let $f \in (\mathbb{Z}/n\mathbb{Z})[T]$ be a polynomial of degree r. The algorithm finds all roots x of f with $0 \le x < \sqrt[r]{n}$ (or proves that there are none).
> The execution time is polynomial in $\log n$ and r.
> (The algorithm uses the "LLL algorithm" for reduction of lattice bases.)

In our example n has at least 321 bits, and $e = 3$. Thus the algorithm outputs x since $x^3 < 2^{192} < 2^{320} < n$.

This is only a simple example of a larger class of attacks for special situations that amount to a computation of roots $\bmod n$.

Exercise. Modify the attack for a situation where the unknown part of the plaintaxt consists of some contiguous letters surrounded by known plaintext sequences.

2.9 The Signature Trap

The signature trap doesn't challenge the security of RSA itself, but the frame conditions of its use: Since reversing the order of encryption and decryption is the basic mechanism of digital signatures the user has to take care that he doesn't inadvertently decrypt a ciphertext in the erroneous belief that he digitally signs a document. Would the standard input to the signature algorithm be a normal plaintext, the user would realize this situation at once. However for (at least) three reasons the situation is different:

1. To get acceptable performance usually a digital signature is applied to a (cryptographic) hash value of a document. This cannot be distiguished from a random bitstring.

2. Strong authentication requires digitally signing a random bitstring instead of entering a password to prove the user's identity. Even if the result was a decrypted plaintext—the user wouldn't see it at all since it is immediately sent to the communication partner (that might be a server, or a "man in the middle").

3. Moreover the attacker could present an arbitrary text that is "camouflaged" by some kind of encryption, and require the user to "sign" (i. e. decrypt) it. Even a close inspection of the result would not detect the fraud—see below. This is a otherwise very useful property of RSA: It is the basis for blind signatures and hence the generation of digital pseudonyms and anonymous transactions.

By the way this an instance of an *attack with chosen ciphertext*. To escape this attack in practice each of the three (or four) functions

- encryption,
- digital signature,
- strong authentication,
- (optionally) blind signature,

should use a different key pair.

Now for the "camouflage" that disguises the chosen ciphertext attack. Here is the procedure:

1. The attacker M ("Mallory") has an intercepted ciphertext $x = E_{\mathrm{A}}(a)$ and would like to read it. He encrypts it as $y = C(x)$ using a function C known only to him.

2. He presents y to his victim A ("Alice") and requires a digital signature. A generates $z = D_{\mathrm{A}}(y)$.

3. M removes the "camouflage" by a suitable inverse transformation C'. For this he needs a pair (C, C') of transformations such that

$$C' \circ D_{\mathrm{A}} \circ C = D_{\mathrm{A}}.$$

   Then $a = D_{\mathrm{A}}(x) = C'(z)$.

As a peculiarity of RSA such pairs (C, C') of transformations exist: Let $E_{\mathrm{A}}(a) = a^e \bmod n$, and take C as the shift by u^e on $\mathbb{M}_n = (\mathbb{Z}/n\mathbb{Z})^\times$, and C' as the multiplication by $u^{-1} \bmod n$ where $u \in \mathbb{M}_n$ is randomly chosen. Thus the attack runs through the steps:

1. M chooses u und computes $y = C(x) = u^e x \bmod n$.
2. A generates $z = y^d \bmod n$.
3. M computes

$$C'(z) = zu^{-1} = y^d u^{-1} = u^{ed} x^d u^{-1} = x^d = a$$

   in $\mathbb{Z}/n\mathbb{Z}$.

2.10 More Attacks

Finally we give a short overview over some other attacks on RSA. For a comprehensive treatment consult the paper by D. Boneh (see the introduction of this section):

1. **Small private exponent:** M. Wiener detected a way of efficiently computing the private key d from the public key (n, e) using continued fractions in the case $d < \frac{1}{3} \cdot \sqrt[4]{n}$.

2. **Related plaintexts** after Franklin/Reiter. Assume two different plaintexts a_1 and a_2 are related by an affine equation $a_2 = sa_1 + t$ with known coefficients s, $t \neq 0$. Then the corresponding plaintexts are efficiently computable from the public key (n, e), the coefficients s and t and the ciphertexts. Coppersmith found a situation that forces such an affine equation in the case where a_1 and a_2 originate from the same plaintext by "padding" differently.

3. **Partial leak** after Boneh/Durfee/Frankel/Coppersmith. If the last quarter of the bits of one of the integers d (the private exponent), p, or q (the prime factors of the module) are known, then n may be efficiently factorized.

4. **Timing and power attacks** after Kocher. The attacker observes the CPU during a decryption and measures the execution time or the power consumption. From this she gains informations about the bits of the private exponent. See the binary power algorithm, Section 1.2.

5. **Differential fault analysis** after Shamir et al. The attacker exposes the processor (for instance a smartcard) to environment conditions slightly outside the range where the specification guarantees a faultless operation, for instance by deforming, heating, radiation. Then the processor will produce single faulty bits that allow statistical inferences about the internal parameters.

Other attacks don't target the RSA algorithm itself but bugs in the implementation, faulty use in cryptographic protocols, flawed interaction with the system environment, and other mistakes.

In some situations using modules with more then two prime factors might even be advantageous, as the following paper suggests:

- M. Jason Hinek, Mo King Low, Edlyn Teske: On some attacks on multi-prime RSA. SAC 2002, 385–404.