2 RSA and Probabilistic Prime Number Tests

2.1 General Considerations and the RSA System

The RSA cryptosystem (named after R. Rivest, A. Shamir, and L. Adleman, who published it in the 1970s) is one of the best-known so-called public key cryptosystems. The idea is the following: Every participant chooses two different big primes \( p \) and \( q \) "at random" and calculates their product \( n = pq \). Then he chooses some arbitrary natural number \( e \) that is relatively prime to the Euler totient function \( \varphi(n) \) (which denotes the number of relative primes to \( n \) that are smaller than \( n \) or - in other words - the number of invertible elements mod.\( n \)). In our situation, we have \( \varphi(n) = (p - 1)(q - 1) \). So for \( e \) one can take, e.g., any prime larger than \( (p - 1)(q - 1) \) or, what makes the decoding and encryption in the binary system especially simple, the 4th Fermat Number \( F_4 := 2^{2^4} + 1 = 65'537 (= 1'0000'0000'0000'0001 \text{ in the binary system}) \). The pair \((n,e)\) is the so-called public key of the participant, which he publishes and will be known to everybody. As his secret key, he keeps the solution \( d < \varphi(n) \) of the equation

\[
ed = 1 (mod. (p - 1)(q - 1)). \tag{2.1}
\]

This solution can be found rapidly by the Euclidean algorithm if \( p \) and \( q \) are known. But factorizing numbers \( n \) seems to be computationally hard in the sense that there seems to exist no algorithm that is faster than exhaustive search. Moreover, there is no known algorithm to solve (2.1) faster than by finding \( p \) and \( q \). But the actual equivalence has not been proved up to now. See also Boneh, Venkatesan (1998). There are similar systems (however, with other disadvantages) where breaking the system is provably equivalent to finding the secret key, for example the Rabin system (Kranakis (1986)) or the Williams (1980) algorithm. For convenience, we will now write \((n_A,e_A)\) and \((n_B,e_B)\) for the public key of Alice and Bob resp., and \(d_A\) and \(d_B\) for their respective secret keys. Assume Alice wants to send a message \( x \) (w.l.o.g. in the form of a natural number mod.\( n_B \)) to Bob. For that, she calculates the ciphertext \( y \) (which will also be a natural number mod.\( n_B \)) by

\[
y := x^{e_B} (mod. n_B) \tag{2.2}
\]
and sends this to Bob. Bob will make the decoding
\[ x = y^{d_B} \pmod{n_B} \] (2.3)
(which follows from (2.2) by (2.1) and Fermat’s Little Theorem). So the RSA system seems to ensure confidentiality. The system can also be used to ensure authenticity: For that, Alice sends, in addition to the encrypted message \( x \), her "electronic signature" \( m \), encrypted by
\[ u := m^{d_A} \pmod{n_A}, \] (2.4)
to Bob. Finding \( d_A \) from \( u \) is the so-called discrete logarithm problem, which is also believed to be hard. So by signing, Alice does not reveal her private key \( d_A \). Since \( d_A \) is only known to Alice, she alone can have produced \( u \), so \( u \) has really the role of a "signature". On the other hand, Bob can verify that this is really Alice’s signature by checking if
\[ u^{e_A} \equiv m \pmod{n_A}. \] (2.5)

A probabilistic (or so-called Monte Carlo) primality test is an algorithm \( A_P(n) \) that, for the input \( n \), gives one of the two answers "prime" or "composite" such that if it yields "composite", then \( n \) is composite and if it yields "prime", then \( n \) is indeed prime with high probability. It seems to be a general fact in prime number testing that if in the case of the output "prime" one is satisfied that this answer is correct only up to some small error probability, then the test runs much faster, or - in other words - what costs most effort is to obtain absolute security in improbable cases. At least theoretically, a major breakthrough has been achieved recently by Agrawal et al. (2003), who gave an unconditional (i.e., not depending on any unproven assumption as, e.g., the Extended Riemann Hypothesis (see Section 2.2) deterministic polynomial-time algorithm to decide whether or not a number is prime (see also Bornemann (2002), Bernstein (2002), and New York Times 8/8/2002).

In detail, for a probabilistic primality test one defines a so-called primality sequence \( P = \{P_n\}_{n \geq 1} \) of sets of natural numbers with the following properties:

(i) \( P_n \subset \mathbb{Z}^*_n \) (= group of integers mod.\( n \) relatively prime to \( n \)).
(ii) Given \( b \in \mathbb{Z}^*_n \) one may check in time polynomial in the length of the binary expansion of \( n \) if \( b \in P_n \).
(iii) If \( n \) is prime, then \( P_n = \emptyset \).
(Iv) There exists a so-called primality constant \( \varepsilon \in [0, 1[ \) (independent of \( n \)) such that for all sufficiently large composite odd \( n \geq 1 \) one has
\[ P(x \in \mathbb{Z}^*_n : x \notin P_n) \leq \varepsilon. \]

Now the test algorithm works as follows:

- Input: \( n \geq 2 \).
- Choose an integer \( x \in \mathbb{Z}^*_n \) at random.
- Output: \( A_P(n) = "prime" \) if \( x \notin P_n \) and \( A_P(n) = "composite" \) if \( x \in P_n \).