Summerschool crypt@b-it 2006

JOACHIM VON ZUR GATHEN, MICHAEL NÜSKEN, DANIEL LOEBENBERGER

1. Exercise sheet RSA/AES

Exercise 1.1 (Playing with RSA). (7 points)

ALICE has set up RSA with $n = 8051 = 83 \cdot 97$ and published her public key $K = (n, e) = (8051, 3149)$.

(i) Find her private key $S = (n, d)$. [3]

(ii) Encrypt the message $x = 101$ with RSA using the key K. [2]

(iii) BOB has sent ALICE the ciphertext 694. Compute the plaintext. [2]

Exercise 1.2 (Blind signatures). (5 points)

BOB is a public notary on the net and signs hashes of messages on behalf of EVE, but for reasons of efficiency he does not require to see the explicit message he is signing.

(i) Show that EVE can misuse BOB's service to decrypt an arbitrary message that was encrypted with BOB's public key without BOB ever learning what he did. [4]

(ii) How can BOB avoid this attack in practice? [1]

Exercise 1.3 (φ-asco). (3 points)

Let $p, q \in \mathbb{N}$ distinct primes and $n = p \cdot q$ and assume you know $\varphi(n)$.

(i) How can you compute p and q with this knowledge? [2]
Hint: Consider the quadratic polynomial $(x - p)(x - q) \in \mathbb{Z}[x]$.

(ii) What impact on the security of RSA does this have? [1]

Exercise 1.4 (RSA bad choice). (4 points)

Show why the 35-bit integer $23\,360\,947\,609$ is a particularly bad choice for $N = pq$. (And not only because it is too small in practice.)

We claim that two prime numbers which are really close to each other are bad choices for the RSA system. To show this we use Fermat's factorization method based on the fact: Suppose $p > q$ are odd integers. Then:

$$N = pq \quad \Longleftrightarrow \quad N = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2.$$

3 (i) Explain how you can use this fact to find prime factors of N. (Hint: let $s := \left\lceil \sqrt{N} \right\rceil$, and consider s, $s+1$, $s+2$, ...)

1 (ii) Do it for $N = 23\,360\,947\,609$.

Exercise 1.5 (The finite field $\mathbb{F}_{2^8}$). (8+4 points)

The elements of the finite field $\mathbb{F}_{2^8}$ are polynomials of degree less than 8 with coefficients in the two-element field $\mathbb{F}_2$. Each element is of course given by eight bits, which we can also read as a hexadecimally written byte, so that, for example, `A1` corresponds to $x^7 + x^5 + 1$. Addition and multiplication are executed 'as usual' but the result is reduced modulo the polynomial $x^8 + x^4 + x^3 + x + 1$. Calculate in this field:

1 (i) Add $x^5 + x + 1$ and $x^7 + x^6 + 1$.

1 (ii) Add `23` and `C1`.

1 (iii) Multiply $x^5 + x + 1$ and $x^7 + x^6 + 1$.

1 (iv) Multiply `23` and `C1`.

2 (v) Calculate the inverse of $x^5 + x + 1$.

2 (vi) Calculate the inverse of `23`.

+4 (vii*) Describe an algorithm to calculate the inverse of a non-zero element.

Exercise 1.6 (The finite ring $\mathbb{F}_{2^8}[y]/\langle y^4+1\rangle$). (10 points)

Calculate in the finite ring $S = \mathbb{F}_{2^8}[y]/\langle y^4+1\rangle$:

(i) Multiply $c = \texttt{02} + \texttt{01}y + \texttt{01}y^2 + \texttt{03}y^3$ by $d = \texttt{0E} + \texttt{09}y + \texttt{0D}y^2 + \texttt{0B}y^3$. 4

(ii) Multiply the column of values `00`, `7A`, `01`, `00` with the polynomial c and write it again as a column. 2

(iii) Try to compute an inverse for $\texttt{01} + \texttt{01}y^2$. 2

(iv) Try to compute an inverse for $\texttt{11} + \texttt{01}y^2$. 2

Exercise 1.7 (Chinese remaindering). (0+6 points)

For RSA we consider a lot of numbers modulo $N = p \cdot q$, where p, q are distinct primes each having, say, $\lfloor n/2 \rfloor$ bits. Since p and q are required to be different primes they are clearly coprime. Thus by the Chinese Remainder Theorem we know that $\mathbb{Z}_N$ is *isomorphic* to $\mathbb{Z}_p \times \mathbb{Z}_q$.

(i) Prove that we can compute $z := y^d \text{ rem } N$ by computing +2

$$z_1 := y^{d \text{ rem } (p-1)} \text{ rem } p \text{ and } z_2 := y^{d \text{ rem } (q-1)} \text{ rem } q$$

and combining these into $z = (z_1 tq + z_2 sp) \text{ rem } N$ where $sp + tq = 1$.

Suppose that in a given implementation a multiplication modulo a k-bit number takes $\mathcal{O}(k^2)$ bit operations. Let p and q be both $n/2$-bit numbers.

(ii) How much time do we need to compute $z = y^d \text{ rem } N$? +1

(iii) How much time do we need to compute z as in (i)? (Do not count the computation of s and t because this can be done in a precomputation. So assume that sp and tq are given.) +1

(iv*) Bob encrypts x and sends $y = x^e \text{ rem } N$ to Alice. Now, say, she actually does the decryption of y using (i). What do you think about the security of this approach? (Consider what happens if Alice, or her computing device, does an error in computing z_2 and gets z_2' instead of the correct value. How do x and $z' = z_1 tq + z_2' sp$ differ?) +2