Introduction to cryptology
TD#6

2018-W13/14

Exercise 1: The Menezes-Okamoto-Vanstone attack on the elliptic curve discrete logarithm problem

The discrete logarithm problem in a prime-order group of size N of the points of an elliptic curve E is generally considered to be "maximally hard", in the sense that the best known algorithms to solve it are generic, and have complexity $\sqrt{N}$. Yet, there are some cases where this problem becomes much easier.

Let $E/\mathbb{F}_q$ be an elliptic curve defined over the field $\mathbb{F}_q$. The points of E can be added together and form a group. The neutral element of this group is written O. The points P of E such that P added to itself r times (for some r, noted $[r]P$) is equal to O forms a subrgoup of E, noted $E[r]$, which is isomorphic to $\mathbb{Z}/r\mathbb{Z} \times \mathbb{Z}/r\mathbb{Z}$. One can define a pairing $e : E[r] \times E[r] \to \mu_r$, where μ_r is the group of r-th roots of unity. The smallest d such that $\mu_r \subseteq \mathbb{F}_{q^d}^\times$ is called the *embedding degree* of r in $\mathbb{F}_q$.

We want to study the hardness of the DLP in a subgroup $\langle P \rangle$ when $P \in E[r]$. That is, given $Q = [k]P$, we wish to recover k.

Q. 1: Show that if $T \in E$ is such that P and T generate the entire group $E[r]$, then $\omega := e(P, T)$ is a generator of μ_r.

Q. 2: Give an expression of $e(Q, T) = e([k]P, T)$ in function of k and ω.

Q. 3: Using the previous expression, show how to retrieve k by solving a DLP in $\mathbb{F}_{q^d}^\times$.

Q. 4: Conclude on the importance of the embedding degree for the hardness of the DLP in $\langle P \rangle$.

Note: In most cases, this attack is not a concern, as the embedding degree is usually expected to be proportional to r. However, some settings require it to be reasonably small (e.g. 12), for instance when the pairing is used to perform a three-party Diffie-Hellman key exchange.

Exercise 2: Big number arithmetic

Assume that we have a CPU featuring a $64 \times 64 \to 128$ bit multiplier `mul`. Any two integers $a, b < 2^{64}$ can then be multiplied (without loss of precision) with one multiplication instruction, i.e. a cost of `1M`. This CPU also features a $64 \times 64 \to 64$ bit integer adder `add` that adds two numbers $a, b < 2^{64}$ and reduces the result mod 2^{64}. If such a reduction was performed, a *carry flag* `CF` is set to 1; otherwise, it is set to 0. A second addition instruction `addc` is similar to `add`, except that it additionally adds the value previously contained in `CF`. Both of these have a cost of one addition instruction, i.e. `1A`. Finally, we suppose that

the instructions do not destroy their input operands, and adopt the syntax `mul a, b, c, d` to say that `c` (resp. `d`) stores the *high bits* (resp. *low bits*) of the multiplication of `a` by `b`, and `add a, b, c` to say that `c` stores the result of the addition of `a` and `b`.

Q. 1: Let `r0`, `r1`, `r2`, `r3` be registers that hold the numbers 2^{63}, $2^{63}+2^{25}$, 2^{12}, 2^{60}. What are the values of `r4` and `r5` after the execution of `add r0, r1, r4; add r2, r3, r5`? What about after the execution of `add r0, r1, r4; addc r2, r3, r5`?

Q. 2: Give a procedure to add two 256-bit numbers together, modulo 2^{256}. What is its complexity (in terms of `A`)?

Q. 3: We now wish to compute the exact product of two 128-bit numbers f, g. Let $f = 2^{64}f_1 + f_0$, $g = 2^{64}g_1 + g_0$ with f_i, g_i less than 2^{64}; give an algorithm to compute $h := fg = (2^{64}f_1 + f_0)(2^{64}g_1 + g_0) = 2^{192}h_3 + 2^{128}h_2 + 2^{64}h_1 + h_0$. What is the complexity (in terms of `M` and `A`) of your algorithm?

In the following exercises, we let $N = pq$ be the product of two prime numbers, $e \in (\mathbb{Z}/\varphi(N)\mathbb{Z})^\times$, $d = e^{-1}$. We define the RSA permutation RSA-P with parameters N and e as RSA-P : $\mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$, $m \mapsto m^e$. Its inverse is given by $c \mapsto c^d$.

Exercise 3: Domain of an RSA permutation

Q. 1: Using the extended Euclid algorithm, show that if $0 < \alpha < N$ is such that $\gcd(\alpha, N) = 1$, then α has a multiplicative inverse modulo N. Show then that for any $e > 0$, α^e is invertible modulo N.

Q. 2: Consider now $0 < \alpha < N$ with $\gcd(\alpha, N) = p$. What is the value of $\alpha \mod p$? Does α have an inverse modulo N? What is $\gcd(\alpha, q)$? Using the CRT, how many such elements are there in $\mathbb{Z}/N\mathbb{Z}$? What is $\alpha^{q-1} \mod q$?

Q. 3: Let $0 < u < N$ be the unique number modulo N that verifies $u = 0 \mod p$, $u = 1 \mod q$. How can you compute u using inversion modulo q? Let α be as in the above question; what are $\alpha^{q-1} \mod N$ and $\alpha^{k(q-1)} \mod N$ (for any k)? Give a necessary condition on e for the map $x \mapsto x^e$ to be invertible on α.

Q. 4: Let $e \in (\mathbb{Z}/\varphi(N)\mathbb{Z})^\times$, $d = e^{-1}$. What is $\alpha^{ed} \mod q$? What is $\alpha^{ed} \mod N$? Are there any elements not invertible by $x \mapsto x^e$? What is the domain of an RSA permutation?

Exercise 4: Semi-homomorphic property of an RSA permutation

Q. 1: Let m, $m' \in \mathbb{Z}/N\mathbb{Z}$, $c = \text{RSA-P}(m)$, $c' = \text{RSA-P}(m')$. Give an expression for cc' of the form x^e (for some x). Use this expression to compute the value $\text{RSA-P}^{-1}(cc')$.

Q. 2: Explain how the above property allows to multiply two numbers without decrypting them.

Q. 3: Note that the above procedure is deterministic. Does a modified procedure that works with encrypted numbers of the form $x || \text{pad}(\cdot)$ (where pad is a non-deterministic function) still allow to multiply numbers in encrypted form?