Cryptography and Security — Midterm Exam

Serge Vaudenay

25.11.2015

- duration: 1h45
- no documents allowed, except one 2-sided sheet of handwritten notes
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **not** answer any technical question during the exam
- readability and style of writing will be part of the grade

1 Vernam with Two Dice

Our crypto apprentice decided to encrypt messages $x \in \mathbf{Z}_{12}$ (instead of bits) using the generalized Vernam cipher in the group $\mathbf{Z}_{12}$. As he did not fully understand the course, he decided to pick a key k (for each x) by rolling two dice (with 6 faces numbered from 1 to 6) and setting $k = k_1 + k_2$ to the sum of the two faces up k_1 and k_2. The encryption of x with key k is then $y = (x + k) \bmod 12$.

Q.1 Why is this encryption scheme insecure?

Q.2 We still use $k = k_1 + k_2$. Given a factor n of 12, we now take $x \in \mathbf{Z}_n$ and $y = (x + k) \bmod n$. Show that for some values n, this provides perfect secrecy but for others, this does not. (Consider *all* factors n of 12.)

Q.3 Finally, the crypto apprentice decides to encrypt a bit $x \in \{0, 1\}$ into $y = (x + k) \bmod 4$, still with $k = k_1 + k_2$ from rolling the two 6-face dice. We assume that x is uniformly distributed in $\{0, 1\}$. For each c, compute the probabilities $\Pr[x = 0|y = c]$ and $\Pr[x = 1|y = c]$.

Q.4 By taking $\tilde{x} \in \{0, 1\}$ as a function of c such that $\Pr[x = \tilde{x}|y = c]$ is maximal, compute the probability $P_e = \Pr[x \neq \tilde{x}]$ (still when x is uniform in $\{0, 1\}$).

2 Elliptic Curve Factoring Method

In this exercise, we want to recover the smallest prime factor p of an integer n.

Given an elliptic curve $E_{a,b}(p)$ over $\mathbf{Z}_p$, we denote by $\mathcal{O}$ the point at infinity. The procedure to add two points P and Q which has been seen in class can be implemented as follows:

$\mathsf{Add1}(E_{a,b}(p), P, Q)$
if $x_P \equiv x_Q \pmod p$ and $y_P \equiv -y_Q \pmod p$ (equivalent to $P = -Q$) **then**
  return $\mathcal{O}$
end if
if $x_P \equiv x_Q \pmod p$ and $y_P \equiv y_Q \pmod p$ (equivalent to $P = Q$) **then**
  set $u = (2y_P)^{-1} \bmod p$
  set $\lambda = ((3x_P^2 + a) \times u) \bmod p$
else

set $u = (x_Q - x_P)^{-1} \bmod p$
set $\lambda = ((y_Q - y_P) \times u) \bmod p$
end if
set $x_R = (\lambda^2 - x_P - x_Q) \bmod p$
set $y_R = ((x_P - x_R)\lambda - y_P) \bmod p$
return $R = (x_R, y_R)$

We first consider the following algorithm. (Yes, it uses p but we will later build on it another algorithm ignoring p.)

Proc1(p)

pick some random parameters $a, b \in \mathbf{Z}_p$, define the elliptic curve $E_{a,b}(p)$ over $\mathbf{Z}_p$ by $y^2 = x^3 + ax + b$ and pick a random point S on $E_{a,b}(p)$
set $i = 1$
while $S \neq \mathcal{O}$ **do**
$i \leftarrow i + 1$
$S \leftarrow i.S$ with the double-and-add algorithm using Add1($E_{a,b}(p), P, Q$)
end while

We let q denote the order of $E_{a,b}(p)$ over $\mathbf{Z}_p$. We assume that, due to selecting a and b at random, q is a random number between $p - 2\sqrt{p}$ and $p + 2\sqrt{p}$.

Q.1 Show that Proc1 terminates.

Q.2 Let $M(q)$ be the largest prime factor of q and α_j be the largest integer such that j^{α_j} divides q. We assume that the probability that q is such that we have $\alpha_j \leq \left\lfloor \frac{M(q)}{j} \right\rfloor$ for all prime j is "very high", and that the probability that a random point P in $E_{a,b}(p)$ has an order multiple of $M(q)$ is also "very high".
Show that when these two conditions are met, Proc1 terminates with the value $i = M(q)$.
HINT: Show that when the first condition is met, then q divides $M(q)!$.
HINT^2: This question may be a bit harder than the next ones.
In what follows, we assume that this implies that the average number of iterations in Proc1 is $e^{\sqrt{(1+o(1)) \ln p \ln \ln p}}$.

Q.3 We change Proc1 into Proc2 by making computations modulo n instead of modulo p. When adding two points P and Q, the test $P = Q$ and the test $P = -Q$ are still done modulo p. We temporarily assume that we can easily pick an element in the curve at random in the first step of Proc2. Below, we underline what was changed.

Add2($E_{a,b}(p, n), P, Q$)

if $x_P \equiv x_Q \pmod{p}$ and $y_P \equiv -y_Q \pmod{p}$ **then**
return $\mathcal{O}$
end if
if $x_P \equiv x_Q \pmod{p}$ and $y_P \equiv y_Q \pmod{p}$ **then**
set $u = (2y_P)^{-1} \bmod n$ (abort with an error message if non invertible)
set $\lambda = ((3x_P^2 + a) \times u) \bmod n$
else
set $u = (x_Q - x_P)^{-1} \bmod n$ (abort with an error message if non invertible)
set $\lambda = ((y_Q - y_P) \times u) \bmod n$
end if
set $x_R = (\lambda^2 - x_P - x_Q) \bmod n$
set $y_R = ((x_P - x_R)\lambda - y_P) \bmod n$

13: return $R = (x_R, y_R)$

$\mathsf{Proc2}(p, n)$

1: pick some random parameters $a, b \in \underline{\mathbf{Z}}_n$, define the curve $\underline{E_{a,b}(p, n)}$ over $\mathbf{Z}_n$ by $y^2 = x^3 + ax + b$, and pick a random point S on $\underline{E_{a,b}(p, n)}$

2: set $i = 1$

3: **while** $S \neq \mathcal{O}$ **do**

4: $\quad i \leftarrow i + 1$

5: $\quad S \leftarrow i.S$ with the double-and-add algorithm using $\mathsf{Add2}(\underline{E_{a,b}(p, n)}, P, Q)$

6: **end while**

We execute in parallel Proc1 and Proc2 with the same random seed. We let S_1 (resp. S_2) designate the value of the register S in Proc1 (resp. Proc2). Show that at every step, $x_{S_1} \equiv x_{S_2} \pmod p$ and $y_{S_1} \equiv y_{S_2} \pmod p$ until Proc2 aborts with an error or terminates.

Q.4 Transform Add2 so that any abortion yields a non-trivial factor of n instead of an error.

Q.5 Further transform Add2 so that it does not need p any longer.
HINT: look at what can go wrong if we do the comparisons modulo n.

Q.6 Observe that the first step of Proc2 cannot be done efficiently. Transform this step to make it doable efficiently and without using p.
HINT: pick S first!

Q.7 Show that the probability that Proc2 terminates with an abortion is "very high" based on the assumptions from Q.2. Deduce that we can find the smallest prime factor p of n with complexity $e^{\sqrt{(1+o(1)) \ln p \ln \ln p}}$.
HINT: we do not expect any probability computation, just identify cases when the algorithm does not abort and heuristicaly justify that this is unlikely to happen.