Cryptography and Security — Midterm Exam
Solution

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

The exam grade follows a linear scale in which each question has the same weight.

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?

In the generalized Vernam cipher, k must be uniformly distributed in $\mathbf{Z}_{12}$. Here, k is a number from 2 to 12. It is not a big deal as it is equivalent to use $k \bmod 12$, but the distribution of $k \bmod 12$ we obtain is far from being uniform in $\mathbf{Z}_{12}$. For instance, $\Pr[k \bmod 12 = 2] = \frac{1}{12}$ *and* $\Pr[k \bmod 12 = 7] = \frac{1}{6}$.

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.)

We just have to say for which n is $k \bmod n$ uniformly distributed. Since $k = k_1 + k_2$, the sum of the values k_1 and k_2 of the two dice, and since k_1 and k_2 are independent and uniformly distributed modulo 6, the scheme is secure when n is a factor of 6: $n \in \{1, 2, 3, 6\}$. For $n = 12$, we have seen it is not secure. What remains is $n = 4$.
$k_1 \bmod 4$ *and* $k_2 \bmod 4$ *have distribution* $\Pr[k_i \bmod 4 = i] = \frac{1}{6}$ *for $i \in \{0, 3\}$ and* $\Pr[k_i \bmod 4 = i] = \frac{1}{3}$ *for $i \in \{1, 2\}$. So,* $\Pr[k \bmod 4 = 0] = \frac{1}{4}$, $\Pr[k \bmod 4 = 1] = \frac{2}{9}$, $\Pr[k \bmod 4 = 2] = \frac{1}{4}$, *and* $\Pr[k \bmod 4 = 3] = \frac{5}{18}$. *So, it is not uniform and the scheme is not secure for $n = 4$.*

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 $\Pr[x=1|y=c]$.

Using the Bayes formula, we have

$$\Pr[x=b|y=c] = \frac{\Pr[y=c|x=b]\Pr[x=b]}{\sum_{b'}\Pr[y=c|x=b']\Pr[x=b']}$$

Clearly, $\Pr[y=c|x=b'] = \Pr[k \equiv c-b' \pmod 4]$ *due to the independence between* x *and* k*. Since* x *is uniformly distributed, we obtain*

$$\Pr[x=b|y=c] = \frac{\Pr[k \equiv c-b]}{\sum_{b'}\Pr[k \equiv c-b']} = \frac{\Pr[k \equiv c-b]}{\Pr[k \in \{c, c-1\}]}$$

where values of k *are taken modulo 4. Using the distribution that we computed in the previous question, we can fill the following table:*

c	$\Pr[x=0\|y=c]$	$\Pr[x=1\|y=c]$
0	9/19	10/19
1	8/17	9/17
2	9/17	8/17
3	10/19	9/19

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 $\{0,1\}$).

We have $\tilde{x} = 1$ *for* $c = 0$, $\tilde{x} = 1$ *for* $c = 1$, $\tilde{x} = 0$ *for* $c = 2$, *and* $\tilde{x} = 0$ *for* $c = 3$. *For* $x = 0$, $x \neq \tilde{x}$ *when* $c \in \{0,1\}$ *so* $k \bmod 4 \in \{0,1\}$. *For* $x = 1$, $x \neq \tilde{x}$ *when* $c \in \{2,3\}$ *so* $k \bmod 4 \in \{1,2\}$. *So,* $P_e = \frac{1}{2}\left(\frac{1}{4}+\frac{2}{9}\right) + \frac{1}{2}\left(\frac{2}{9}+\frac{1}{4}\right) = \frac{17}{36} = \frac{1}{2} - \frac{1}{36}$.

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.)

$\mathsf{Proc1}(p)$

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 $\mathsf{Add1}(E_{a,b}(p), P, Q)$
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 $\mathsf{Proc1}$ terminates.

Due to the Lagrange Theorem, $q.S = \mathcal{O}$ for any initial point S. So, if i is large enough, q divides $i!$ and for sure, $i!.S = \mathcal{O}$. Hence, $\mathsf{Proc1}$ 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, $\mathsf{Proc1}$ 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.

Let $q = \prod j_k^{\alpha_{j_k}}$ be the factorization into primes of q, with $\alpha_{j_k} > 1$ and $j_1 < j_1 < \cdots$ primes. Due to the assumption, we have $\alpha_{j_k} \leq \left\lfloor \frac{M(q)}{j_k} \right\rfloor$. So, q divides $\prod j_k^{\left\lfloor \frac{M(q)}{j_k} \right\rfloor}$.
We have $\lfloor M(q)/j \rfloor$ integers multiple of j between 1 and $M(q)$ so $j_k^{\left\lfloor \frac{M(q)}{j_k} \right\rfloor}$ divides $M(q)!$ for all k. Since all j_k are different primes, $\prod j_k^{\left\lfloor \frac{M(q)}{j_k} \right\rfloor}$ divides $M(q)!$ as well. Hence, q divides $M(q)!$. We deduce that $i = M(q)$ makes the algorithm terminate.
As $M(q)$ is prime, $(M(q)-1)!$ is not divisible by $M(q)$ so when the order of P is a multiple of $M(q)$, $i = M(q) - 1$ does not terminate.
So, $i = M(q)$ is the smallest i making the algorithm terminate.

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

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 $u = (2y_P)^{-1} \bmod \underline{n}$ <u>(abort with an error message if non invertible)</u>
  set $\lambda = ((3x_P^2 + a) \times u) \bmod \underline{n}$
else
  set $u = (x_Q - x_P)^{-1} \bmod \underline{n}$ <u>(abort with an error message if non invertible)</u>
  set $\lambda = ((y_Q - y_P) \times u) \bmod \underline{n}$
end if
set $x_R = (\lambda^2 - x_P - x_Q) \bmod \underline{n}$
set $y_R = ((x_P - x_R)\lambda - y_P) \bmod \underline{n}$
return $R = (x_R, y_R)$

Proc2(p, n)

pick some random parameters $a, b \in \underline{\mathbf{Z}_n}$, define the curve $\underline{E_{a,b}(p,n) \text{ over } \mathbf{Z}_n}$ by $y^2 = x^3 + ax + b$, and pick a random point S on $\underline{E_{a,b}(p,n)}$
set $i = 1$
while $S \neq \mathcal{O}$ **do**
  $i \leftarrow i + 1$
  $S \leftarrow i.S$ with the double-and-add algorithm using Add2($\underline{E_{a,b}(p,n)}, P, Q$)
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.

For any polynomial function f, $f(x) \bmod n \bmod p = f(x) \bmod p$. This is also the case when we have divisions, except if we try to divide by something non invertible. So, by induction, the intermediate results are equal modulo p until we have an illegal division.

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

The original algorithm never tries to divide by something non-invertible. So, the new algorithm never tries to divide by a multiple of p. If it tries to divide by some value z which is not invertible modulo n, then $\mathsf{gcd}(n,z) > 1$ and p does not divide z. So, $\mathsf{gcd}(n,z)$ is a non-trivial factor of n. Hence, we can run the extended Euclid algorithm $(u,d) = \mathsf{eEuclid}(n,z)$ to obtain the $d = \mathsf{gcd}(n,z)$ and the inverse u of z modulo n (if $d = 1$). If $d > 1$, we can abort and yield d as a non-trivial factor of n.

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.

If two points are equal modulo n, they must be equal modulo p. However, two different points modulo n may become equal modulo p. What can go wrong is when we add two points P and Q such that $P \neq -Q$ modulo n but $P = -Q$ modulo p. In that case, $x_Q - x_P$ is a multiple of p but not a multiple of n and we are back in the previous case which will yield a non-trivial factor of n. If $P \neq Q$ modulo n but $P = Q$ modulo p, this is the same.

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!

We cannot try to solve $y^2 = x^3 + ax + b$ modulo n as we do not know how to extract roots modulo n. Instead, we pick $S = (x,y)$ at random in $\mathbf{Z}_n$ then $a \in \mathbf{Z}_n$ at random then set $b = y^2 - x^3 - ax$:

1: *pick $S = (x,y) \in \mathbf{Z}_n^2$ at random*
2: *pick $a \in \mathbf{Z}_n$ at random*
3: *set $b = y^2 - x^3 - ax$*

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.

We have seen that Proc1 *terminates with "very high" probability with a number of iterations equal to $M(q)$. If* Proc2 *terminates without any illegal division, it means that for each prime factor p' of n, the order q' of the curve modulo p' have all the same $M(q')$. Since these orders are random and independent, this is "highly unlikely" to happen.*
Here is the final algorithm:

Add3$(E_{a,b}(n), P, Q)$

```
if x_P ≡ x_Q (mod n) and y_P ≡ −y_Q (mod n) then
    return O
end if
if x_P ≡ x_Q (mod n) and y_P ≡ y_Q (mod n) then
    set (u, d) = eEuclid(n, 2y_P)
    if d > 1, abort and yield d
    set λ = ((3x_P^2 + a) × u) mod n
else
    set (u, d) = eEuclid(n, x_Q − x_P)
    if d > 1, abort and yield d
    set λ = ((y_Q − y_P) × u) mod n
end if
set x_R = (λ^2 − x_P − x_Q) mod n
set y_R = ((x_P − x_R)λ − y_P) mod n
return R = (x_R, y_R)
```

ECM(n)

```
pick S = (x, y) ∈ Z_n^2 at random
pick a ∈ Z_n at random
set b = y^2 − x^3 − ax mod n
set i = 1
while S ≠ O do
    i ← i + 1
    S ← i.S with the double-and-add algorithm using Add3(E_{a,b}(n), P, Q)
end while
stop (the algorithm failed)
```

Based on the previous questions, this algorithm is most likely to yield p, or at least a non-trivial factor but we can then run it recursively until we find p. Furthermore, its expected number of iterations is $e^{\sqrt{(1+o(1))\ln p \ln\ln p}}$.