Solution Sheet 7

Cryptography and Security 2025

Solution 1 Elliptic Curve Factoring Method

1. 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, Proc1 terminates.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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 $b = y^2 - x^3 - ax$

7. 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:

Table 1: Weak keys of DES

C	D	k
$\{0\}^{28}$	$\{0\}^{28}$	$\mathrm{PC1}^{-1}(\{0\}^{28}, \{0\}^{28})$
$\{0\}^{28}$	$\{1\}^{28}$	$\mathrm{PC1}^{-1}(\{0\}^{28}, \{1\}^{28})$
$\{1\}^{28}$	$\{0\}^{28}$	$\mathrm{PC1}^{-1}(\{1\}^{28}, \{0\}^{28})$
$\{1\}^{28}$	$\{1\}^{28}$	$\mathrm{PC1}^{-1}(\{1\}^{28}, \{1\}^{28})$

Add3$(E_{a,b}(n), P, Q)$
1: **if** $x_P \equiv x_Q \pmod{n}$ and $y_P \equiv -y_Q \pmod{n}$ **then**
2: return $\mathcal{O}$
3: **end if**
4: **if** $x_P \equiv x_Q \pmod{n}$ and $y_P \equiv y_Q \pmod{n}$ **then**
5: set $(u, d) =$ eEuclid$(n, 2y_P)$
6: if $d > 1$, abort and yield d
7: set $\lambda = ((3x_P^2 + a) \times u) \bmod n$
8: **else**
9: set $(u, d) =$ eEuclid$(n, x_Q - x_P)$
10: if $d > 1$, abort and yield d
11: set $\lambda = ((y_Q - y_P) \times u) \bmod n$
12: **end if**
13: set $x_R = (\lambda^2 - x_P - x_Q) \bmod n$
14: set $y_R = ((x_P - x_R)\lambda - y_P) \bmod n$
15: return $R = (x_R, y_R)$

ECM(n)
1: pick $S = (x, y) \in \mathbf{Z}_n^2$ at random
2: pick $a \in \mathbf{Z}_n$ at random
3: set $b = y^2 - x^3 - ax \bmod n$
4: set $i = 1$
5: **while** $S \neq \mathcal{O}$ **do**
6: $i \leftarrow i + 1$
7: $S \leftarrow i.S$ with the double-and-add algorithm using Add3$(E_{a,b}(n), P, Q)$
8: **end while**
9: 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 $e^{\sqrt{(1+o(1))\ln p \ln\ln p}}$.

Solution 2 Weak Keys of DES

If the subkeys k_1 to k_{16} are equal, then the reversed and original key schedules are identical. In that case, DES_k clearly is an involution. The sixteen subkeys will be equal when the registers C and D are all-zero or all-one bit vectors, as the rotation of such bitstrings has no effect on them. Therefore, the four weak keys of DES can easily be computed by applying $\mathrm{PC1}^{-1}$ to the four possible combinations of these C and D values. We have represented the weak keys of DES on Table 1, where $\{b\}^n$ denotes a sequence of n bits all equal to b.

Solution 3 Complementation Property of DES

1. First note that $\overline{x} \oplus y = \overline{x \oplus y}$ and that $\overline{x} \oplus \overline{y} = x \oplus y$. The initial and final permutations (IP and IP^{-1}) do not have any influence on our computations, so we will not consider them. We

can write one round of DES as

$$(C_L, C_R) \leftarrow (P_R, P_L \oplus F(P_R, K))$$

where P_L and P_R denote the left and right half of the plaintext, respectively, where C_L and C_R denote the left and right half of the ciphertext and where K denotes the key. From the definition of the key schedule algorithm, we see that if we take the bitwise complement of the key, then each subkey will turn into its bitwise complement as well. Furthermore, from DES F-function definition, we can see that if we complement its input and the subkey, then the input of the S-boxes and thus the output will remain the same. We can thus write

$$(C_L, C_R) \leftarrow (\overline{P_R}, \overline{P_L} \oplus F(\overline{P_R}, \overline{K})) = \overline{(P_R, P_L \oplus F(P_R, K))}$$

If we extend this to the whole Feistel scheme, then we can conclude that $\mathsf{DES}_{\overline{K}}(\overline{x}) = \overline{\mathsf{DES}_K(x)}$.

2. The following algorithm describes a brute force attack that exploits the complementation property of DES. Note that in this algorithm, $\overline{c}$ corresponds to $\overline{\mathsf{DES}_k(x)} = \mathsf{DES}_{\overline{k}}(\overline{x})$. Therefore, if the condition of line 6 is true, we almost surely have $K = \overline{k}$. In the loop, the only *heavy* computation is the computation of $\mathsf{DES}_k(x)$, and we expect to perform 2^{54} such computations.

Input: a plaintext x and two ciphertexts $\mathsf{DES}_K(x)$ and $\mathsf{DES}_K(\overline{x})$
Output: the key candidate for K
Processing:
1: **for all** non-tested key k **do**
2: $c \leftarrow \mathsf{DES}_k(x)$
3: **if** $c = \mathsf{DES}_K(x)$ **then**
4: output k and stop.
5: **end if**
6: **if** $\overline{c} = \mathsf{DES}_K(\overline{x})$ **then**
7: output $\overline{k}$ and stop.
8: **end if**
9: **end for**

Solution 4 A Weird Mode of Operation

1. It is equivalent to the ECB mode. Namely, a passive adversary can compute t_i and then $y_i \oplus t_i$ for every i. This gives the ECB encryption of $x_1, \ldots, x_n$.

2. No.

3. Like the ECB mode, if the entropy of a block x_i is low, then $y_i \oplus t_i$ repeats. For instance, $x_i = x_j$ is equivalent to $y_i \oplus t_i = y_j \oplus t_j$ which can be observed with values which are sent over the insecure channel.