Cryptography and Security — Midterm Exam
Solution

Serge Vaudenay

5.12.2018

- 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
- answers should not be written with a pencil

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

1 Design Challenges with Bad Requirements

This exercise proposes two totally independent design challenges. Students with the best answer will get the full points. Others will have partial points. (To compare the answers, we check if all requirements are satisfied and we then look at the complexity of the algorithms.)

Q.1 (Design challenger 1.) You must design a block cipher $(\{0,1\}^k, \{0,1\}^n, \mathsf{Enc}, \mathsf{Dec})$ (following the definition from the course in which k is the key length and n is the block length) which is secure against key recovery under chosen plaintext and ciphertext attacks. More precisely, it must be $(q, t, 2^{-k})$-secure for any q and t, i.e. whatever the number of queries q and the time complexity t, the probability of success should not be better than the probability of success 2^{-k} of an attack which guesses the key at random.

We can propose $\mathsf{Enc}(K, \mathsf{pt}) = \mathsf{pt}$ *and* $\mathsf{Dec}(K, \mathsf{ct}) = \mathsf{ct}$. *We obtain a correct block cipher, i.e.* $\mathsf{Dec}(K, \mathsf{Enc}(K, \mathsf{pt})) = \mathsf{pt}$ *all the time. Furthermore, it is perfectly secure against key recovery:* K *is actually never used so nothing even leaks about* K.

Q.2 (Design challenger 2.) Given some parameters k and n, you must design two algorithms Enc and Dec with the following interface:
- Enc takes two inputs $K \in \{0,1\}^k$ and $\mathsf{pt} \in \{0,1\}^n$ and produce one output ct.
- Dec takes two inputs $K \in \{0,1\}^k$ and ct and produce one output pt'.

The obtained pair $(\mathsf{Enc}, \mathsf{Dec})$ must be secure against decryption under chosen plaintext attack. More precisely, it must be $(q, t, 2^{-n})$-secure for any q and t, i.e. whatever the number of queries q and the time complexity t, the probability of success should not be better than the probability of success 2^{-n} of an attack which guesses pt at random.

We can propose algorithms doing nothing: $\mathsf{Enc}(K, \mathsf{pt}) = \perp$ *and* $\mathsf{Dec}(K, \mathsf{ct}) = \perp$. *This is perfectly secure against decryption attacks. Actually, nothing ever leaks about* pt. *What must be missing in the requirements is the correctness notion of the cryptosystem, i.e.* $\mathsf{Dec}(K, \mathsf{Enc}(K, \mathsf{pt})) = \mathsf{pt}$ *all the time. The proposed cryptosystem is not correct.*

2 RSA with Carmichael Numbers

We recall that by definition, a Carmichael number is an integer n which is the product of several (at least two) pairwise different prime numbers p_i such that $p_i - 1$ divides $n - 1$. We know that a positive integer n is a Carmichael number if and only if it is not prime and for any $b \in \mathbf{Z}_n^*$, we have $b^{n-1} \bmod n = 1$.

In what follows, we consider two Carmichael numbers p and q, and $n = pq$. We denote $\delta = (p-1) \times (q-1)$.

Q.1 Assuming that p and q are coprime, formally prove that $x^\delta \bmod n = 1$ for all $x \in \mathbf{Z}_n^*$? In what follows, we assume that p and q are coprime.

We have

$$x^\delta \bmod p = (x^{p-1} \bmod p)^{q-1} \bmod p = 1^{q-1} \bmod p = 1$$

and similarly, $x^\delta \bmod q = 1$.
With the condition $\mathsf{gcd}(p, q) = 1$*, we can apply the Chinese Remainder Theorem and obtain* $x^\delta \bmod n = 1$.

Q.2 If e and d are integers such that $ed \bmod \delta = 1$, show that for all $x \in \mathbf{Z}_n^*$, we have $x^{ed} \bmod n = x$.

We write the Euclidean division $ed = 1 + k\delta$*. We have*

$$x^{ed} \equiv x^{1+k\delta} \equiv x \times (x^\delta)^k \equiv x \pmod{n}$$

Q.3 With the same notations, prove that $x^{ed} \bmod n = x$ for all $x \in \mathbf{Z}_n$.

If k *is a prime factor of* p *or* q*, for* $x \in \mathbf{Z}_k^*$*, we have* $x^\delta \bmod k = 1$ *because* $k - 1$ *divides* $p - 1$ *or* $q - 1$ *and both divide* δ*. Hence,* $x^{ed} \bmod k = x$*. For* $x = 0$*, we also have* $x^{ed} \bmod k = x$*. Therefore, for all* $x \in \mathbf{Z}$*, we have* $x^{ed} \equiv x \pmod{k}$*. Since* p *and* q *are coprime and product of pairwise different primes,* n *is also a product of pairwise different primes. We have* $x^{eq} \equiv x$ *modulo every prime factor of* n*. So, we have it as well modulo* n*, due to the Chinese Remainder Theorem.*

Q.4 Is it a good idea to use such variant of RSA?

Carmichael numbers are harder to generate than prime numbers, so such variant would be terrible to implement. Additionally, we may end up with n *having small prime factors. So, it would be easy to find, at least a partial factoring of* n*. The hardness of RSA would degrade substantially.*

3 Invalid Curve Attack over ECDH

The following exercise is inspired from Breaking the Bluetooth Pairing — Fixed Coordinate Invalid Curve Attack *by Biham and Neumann.*

The ECDH protocol uses an elliptic curve defined by some domain parameters $D = (q, a, b, P, n)$, where q and n are prime, $a, b \in \mathbf{Z}_q$, and P is a point of order n on the elliptic curve over $\mathbf{Z}_q$ defined by the equation $y^2 = x^3 + ax + b$. In ECDH, each participant U generates his ephemeral secret key $\mathsf{sk}_U \in \mathbf{Z}_n^*$ and its ephemeral public key $\mathsf{pk}_U = \mathsf{sk}_U \times P$. Two participants A and B end up with a secret $\mathsf{sk}_A \times \mathsf{pk}_B = \mathsf{sk}_B \times \mathsf{pk}_A$ from which they extract some key DHKey. We focus on Bluetooth using the P256 curve. In Bluetooth, two devices which want to communicate without knowing each other first run the ECDH protocol to share a secret DHKey, then authenticate the x-coordinates of exchanged public keys by using an alternate communication channel. So, the objective of an adversary could be to interfere with the communication without modifying the x-coordinates so that one or both participants end up with some DHKey which is known by the adversary. The goal of this exercise is to mount such attack for the Bluetooth implementation of ECDH.

In Bluetooth, ECDH is run as follows: each participant U interacting with his counterpart $\bar{U}$ does what follows:

- U sets up the parameters D of P256.
- U picks $\mathsf{sk}_U \in \mathbf{Z}_n^*$ at random and computes $\mathsf{pk}_U = \mathsf{sk}_U \times P$. We denote by (x_U, y_U) the coordinates of pk_U. The computation is done with the double-and-add algorithm.
- U sends (x_U, y_U) to $\bar{U}$.
- U receives $(x_{\bar{U}}, y_{\bar{U}})$ from $\bar{U}$ and verifies that it is not the point at infinity.
- U computes $(K_x, K_y) = \mathsf{sk}_U \times (x_{\bar{U}}, y_{\bar{U}})$. This computation is done with the double-and-add algorithm.
- U throws away K_y and computes a KDF function on K_x to obtain DHKey.
- After that, U authenticates x_U, $x_{\bar{U}}$ and some other values by some ad-hoc means.

We recall how point addition and point doubling is done. We denote $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$, $R = (x_R, y_R)$. For $P \neq Q$ and $x_P = x_Q$, $P + Q$ is the point at infinity. Otherwise, $R = P + Q$ is computed by first computing

$$\lambda = \begin{cases} \frac{y_P - y_Q}{x_P - x_Q} \bmod q \text{ if } P \neq Q \\ \frac{3x_P^2 + a}{2y_P} \bmod q \text{ if } P = Q \end{cases} \qquad x_R = \lambda^2 - 2x_P \bmod q \qquad y_R = -y_P - \lambda(x_R - x_P) \bmod q$$

Q.1 Based on what you have learned on the Diffie-Hellman protocol, tell what is missing and how we should have implemented it.

What misses is the verification that the received public key lies on the group generated by P in the curve. It could have been implemented by first checking that the received key is not the point at infinity, that its coordinates satisfy the equation of the curve, and that when multiplied by n we obtain the point at infinity.

Q.2 Given an attack in which the adversary replaces $y_{\bar{U}}$ by some value $y'_{\bar{U}}$ such that $(x_{\bar{U}}, y'_{\bar{U}})$ lies on a curve of equation $y^2 = x^3 + ax + b'$ and $(x_{\bar{U}}, y'_{\bar{U}})$ has order 2 on this curve. Give the method to compute b' and $y'_{\bar{U}}$ from D and $x_{\bar{U}}$.

An element had order 2 if its y-coordinate is 0*. Hence,* $y'_{\bar{U}} = 0$*. The curve containing the obtained point has parameter* $b' = -x_{\bar{U}}^3 - ax_{\bar{U}}$*.*

Q.3 Prove that the algorithm mapping an input point Q to $\mathsf{sk}_U \times Q$ on the curve $y^2 = x^3 + ax + b$ and on the curve $y^2 = x^3 + ax + b'$ by using the double-and-add algorithm are exactly the same algorithm. Deduce that by applying the previous attack, the adversary can easily compute DHKey which is obtained by U.

Point addition and point doubling never use the b *parameter. So, the computation on both curve imply the same computations. We deduce that during the attack,* U *will in fact compute* $\mathsf{sk}_U \times (x_{\bar{U}}, y'_{\bar{U}})$ *on the invalid curve. Since* U *receives a point of order 2 in a new curve and that* U *applies the double-and-add algorithm in this curve (as it does not make the difference between* b *and* b'*), we obtain* (K_x, K_y) *in a group of order 2, hence either* (K_x, K_y) *equal to the point at infinity, or* $(K_x, K_y) = (x_{\bar{U}}, y_{\bar{U}})$*. So, the extracted* x*-coordinate is either* ∞ *or* $x_{\bar{U}}$*. This only depends on the parity of* sk_U*. The adversary cannot predict this parity but can deduce which one it correct by looking at the forthcoming encrypted communication. It will use KDF applied to either value.*

Q.4 Applying the previous attack in both directions, prove that both participants obtain the same DHKey with probability $\frac{1}{4}$.

The extracted x*-coordinate is* ∞ *with probability* $\frac{1}{2}$*, for each participant. So, with probability* $\frac{1}{4}$*, both are equal to* ∞ *and are henceforth the same.*