Cryptography and Security — Midterm Exam
Solution

Serge Vaudenay

23.11.2017

- 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 RSA over $\mathbf{Z}_{p^\alpha q^\beta}$

We consider a variant of RSA in which the modulus is selected of form $n = p^\alpha q^\beta$ and two different large prime numbers p and q. As usual, the public key is a pair (n, e) and the secret key is a pair (n, d). We assume that α and β are two constants in this cryptosystem.

Q.1 Explain the **encryption algorithm** Enc, the **decryption algorithm** Dec, and the **key generation algorithm** Gen. What **relation** must exist between the public key and the secret key?

Encryption of x with public key (e, n): compute $x^e \bmod n$ using the square-and-multiply algorithm.
Decryption of y with secret key (d, n): compute $x^d \bmod n$ using the square-and-multiply algorithm.
Key generation algorithm: generate two different large prime numbers p and q by trials using a primality test, compute $n = p^\alpha q^\beta$, $\varphi(n) = (p-1)p^{\alpha-1}(q-1)q^{\beta-1}$, pick a random e until it is coprime with $\varphi(n)$ (use the Euclid algorithm to check), and compute the inverse $d = e^{-1} \bmod \varphi(n)$ (use the extended Euclid algorithm for that).
We must have $ed \bmod \varphi(n) = 1$.

Q.2 Prove that for all $x \in \mathbf{Z}_n^*$ we have $\mathsf{Dec}_{n,d}(\mathsf{Enc}_{n,e}(x)) = x$ but that there exist some (rare) $x \in \mathbf{Z}_n$ such that $\mathsf{Dec}_{n,d}(\mathsf{Enc}_{n,e}(x)) \neq x$ when $e > 1$.

We have $(x^e \bmod n)^d \bmod n = x^{ed} \bmod n = x^{1+k\varphi(n)} \bmod n$ for some integer k. If $x \in \mathbf{Z}_n^*$, we have $x^{\varphi(n)} \bmod n = 1$ due to the Lagrange theorem. So, $(x^e \bmod n)^d \bmod n = x$.
For $x = p^{\alpha-1}$ and $e > 1$, we have $x^e \bmod p^\alpha = 0$. So, we cannot have $(x^e \bmod n)^d \equiv x \pmod{p^\alpha}$. So, we cannot have $(x^e \bmod n)^d \bmod n = x$.

Q.3 If either $\alpha = 0$ or $\beta = 0$, prove that this cryptosystem is insecure. Describe an attack and show it has a low complexity.

Assume for instance that $\beta = 0$. So, $n = p^\alpha$. We can thus compute $\sqrt[\alpha]{n} = p$. This computation can be done in time $\mathcal{O}((\log n)^3)$.
Even though α was not known, it must be less than the size of n, so it can be found by exhaustive search (until $\sqrt[\alpha]{n}$ is an integer) in $\mathcal{O}(\log n)$ trials.
Once we have p and α, we can compute $\varphi(n) = p^{\alpha-1}(p-1)$ and the secret key by $d = e^{-1} \bmod \varphi(n)$ in time $\mathcal{O}((\log n)^2)$.
Overall, the complexity is $\mathcal{O}((\log n)^4)$ if α is unknown and $\mathcal{O}((\log n)^3)$ otherwise. This is a low complexity.

Q.4 If $x^2 \equiv y^2 \pmod n$, $x \not\equiv y \pmod n$, and $x \not\equiv -y \pmod n$, formally prove that $\mathsf{gcd}(x-y, n)$ is a nontrivial factor of n.

$\mathsf{gcd}(x-y, n)$ is always a factor of n, but it can be a trivial one (i.e. 1 or n).
If this gcd is n, this means that $x \equiv y \pmod n$ which contradicts the assumption. So, it remains to show that this factor cannot be 1. If it was 1, then $x - y$ would be invertible. But multiplying its inverse by $0 = x^2 - y^2 = (x-y)(x+y) \pmod n$ would imply that $0 = x + y \pmod n$ which contradicts $x \not\equiv -y \pmod n$.

Q.5 Explain that the ability to compute square roots in $\mathbf{Z}_n^*$ allows to factor n efficiently. (Note that we must find p and q, not only a non-trivial factor.)

If we sample a random $x \in \mathbf{Z}_n^*$ and compute $x^2 \bmod n$, we obtain a random quadratic residue of $\mathbf{Z}_n^*$ and one of its square roots x at random. There are (at least) four square roots (we have actually exactly four, we only need to know we have at least four) and we know x and $-x$. So, with probability at least $\frac{1}{2}$, the square root subroutine will return a new square root. With the previous question, we deduce a nontrivial factor n' of n. If n' is not enough to fully factor n, we can repeat the procedure modulo n' to find a nontrivial factor of n'. Eventually, we recover either p or q and it is enough to recover p^α and q^β from n, then deduce the other prime factor using Q.3.

Q.6 Further prove that the knowledge of any multiple of $\lambda(n)$ allows to factor n efficiently.

If m is a multiple of $\lambda(n)$, we have $x^m \bmod n = 1$ for any $x \in \mathbf{Z}_n^*$. We write $m = 2^k t$ with t odd. Note that $\lambda(n) = \mathsf{lcm}(\lambda(p^\alpha), \lambda(q^\beta))$ is even because $\lambda(p^\alpha)$ is even, because $(-1)^{\lambda(p^\alpha)} \bmod p^\alpha$ must be equal to 1. We deduce that x^t has one of its iterative squares $x^t, (x^t)^2, (x^t)^4, \ldots$ equal to 1. The last value which is not 1 in this iterative squares is a square root of 1 which it not 1. With some constant probability, it is not -1 either. So, we deduce a nontrivial factor of n using Q.4.

2 A New Meet-in-the-Middle

Let $x \mapsto C_k(x)$ be a block cipher encryption using a key k and $y \mapsto C_k^{-1}(y)$ be the corresponding block cipher decryption. We assume that k, x, and y live in the space $\{0,1\}^\ell$, for a given integer ℓ. The key k is uniformly distributed. In this exercise, we define new block ciphers using longer keys. We do analysis in a known plaintext attack setting: we assume the adversary gets m pairs (x_i, y_i) consisting of a random x_i and its encryption y_i. The purpose is to do a key recovery with probability of success close to 1. We say that the attack succeeds if it stops with the correct secret key (and only the correct one) as output. In this exercise, we will look for attacks with a variable m and study how to select m for a high success probability in the last question.

Q.1 For the block cipher $x \mapsto C_k(x)$, describe the best possible **known plaintext attack** and its **complexity**. Describe the attack with a pseudocode.

Note: in all questions of the exercise, the complexity is the average time complexity. It is given in terms of number of C or C^{-1} evaluations to make, or equivalent computation.

For all a, we iteratively check for all (x_i, y_i) if $C_a(x_i) = y_i$. If any check fails, we do not check other pairs and iterate. If all checks succeed, the algorithm stops and yields a.

```
1: for all $a$ do
2:     set $i = 0$ and pass = true
3:     while $i < m$ and pass do
4:         $i \leftarrow i + 1$
5:         pass ← $C_a(x_i) = y_i$
6:     end while
7:     if pass then
8:         stop and yield $a$
9:     end if
10: end for
```

For sure, if this algorithm tries the right a, it stops and yields it after m tests. For nearly all wrong a, the first check $C_a(x_i) = y_i$ fails, so ***the complexity is of order*** $2^\ell + m$.

Q.2 We define $C'_{k_1,k_2}(x) = C_{k_2}(C_{k_1}(x))$.
Find **two functions** f and g such that

$$\forall k_1, k_2, x \quad f(k_1, x) = g(k_2, C'_{k_1,k_2}(x))$$
$$\Pr_{a,b,k_1,k_2,x}[f(a,x) = g(b, C'_{k_1,k_2}(x))] = 2^{-\ell}$$

prove **these properties**, and describe the best possible **known plaintext attack** and its **complexity**. Describe the attack with a pseudocode.

We define $f(a,x) = C_a(x)$ and $g(b,y) = C_b^{-1}(y)$. We have

$$g(k_2, C'_{k_1,k_2}(x)) = C_{k_2}^{-1}(C_{k_2}(C_{k_1}(x))) = C_{k_1}(x) = f(k_1, x)$$

so the first property is satisfied. We also have

$$g(b, C'_{k_1,k_2}(x)) = C_b^{-1}(C_{k_2}(C_{k_1}(x)))$$

so, this is equal to $f(a,x) = C_a(x)$ if and only if we have

$$C_b(C_a(x)) = C_{k_2}(C_{k_1}(x))$$

This happens with probability $2^{-\ell}$. So, the second property is satisfied.
Hence, we can store in a hash table all values of a with key $(f(a,x_1), f(a,x_2))$. Then, for all b, we look at key $(g(b,y_1), g(b,y_2))$. If there is any a, we check the pair (a,b) on other (x_i, y_i) pairs as in the first question.

```
1: set all list $H(\cdot)$ to empty
2: for all $a$ do
3:     insert $a$ in list $H(f(a,x_1), f(a,x_2))$
4: end for
5: for all $b$ do
6:     for all $a$ in list $H(g(b,y_1), g(b,y_2))$ do
7:         test the key $(a,b)$ like in Q.1
8:     end for
9: end for
```

Due to the first property, the iteration with $b = k_2$ with yield $a = k_1$. Due to the second property, the probability of another (a,b) pair to pass before checking the other (x_i, y_i) pairs is $2^{-2\ell}$. So, we may have one wrong pair but it will be eliminated by checking the other (x_i, y_i) pairs.
This algorithm has ***complexity*** *$\mathcal{O}(2^\ell + m)$ times the cost of a C evaluation.*

Q.3 We let $\star$ designate a group operation in $\{0,1\}^\ell$. We define $C''_{k_1,k_2,k_3}(x) = C_{k_3}(k_2 \star C_{k_1}(x))$.

Find **two functions** f and g such that

$$\forall k_1, k_2, k_3, x, x' \quad f(k_1, x, x') = g(k_3, C''_{k_1,k_2,k_3}(x), C''_{k_1,k_2,k_3}(x'))$$
$$\Pr_{a,b,k_1,k_2,k_3,x,x'}[f(a,x,x') = g(b, C''_{k_1,k_2,k_3}(x), C''_{k_1,k_2,k_3}(x'))] = 2^{-\ell}$$

and show **the first property**. Describe the best possible **known plaintext attack** and its **complexity**. Describe the attack with a pseudocode.

We define $f(a,x,x') = C_a(x')^{\mathsf{inv}} \star C_a(x)$ and $g(b,y,y') = C_b^{-1}(y')^{\mathsf{inv}} \star C_b^{-1}(y)$, where z^{inv} is the inverse of z in the sense of the $(\{0,1\}^\ell, \star)$ group. We have

$$\begin{aligned} g(k_3, C''_{k_1,k_2,k_3}(x), C''_{k_1,k_2,k_3}(x')) &= C_{k_3}^{-1}(C_{k_3}(k_2 \star C_{k_1}(x')))^{\mathsf{inv}} \star C_{k_3}^{-1}(C_{k_3}(k_2 \star C_{k_1}(x))) \\ &= (k_2 \star C_{k_1}(x'))^{\mathsf{inv}} \star (k_2 \star C_{k_1}(x)) \\ &= C_{k_1}(x')^{\mathsf{inv}} \star C_{k_1}(x) \\ &= f(k_1, x, x') \end{aligned}$$

Using the same algorithm as in the previous question, we obtain a candidate value for (k_1, k_3) with complexity $\mathcal{O}(2^\ell)$. We expect to get 2^ℓ such candidates. Once we get it, we recover k_2 by

$$k_2 = C_{k_3}^{-1}(y_1) \star C_{k_1}(x_1)^{\mathsf{inv}}$$

Then, we can test (k_1, k_2, k_3) with the other (x_i, y_i) pairs. So, the algorithm is

```
 1: set all list $H(\cdot)$ to empty
 2: for all $a$ do
 3:     insert $a$ in list $H(f(a,x_1), f(a,x_2))$
 4: end for
 5: for all $b$ do
 6:     for all $a$ in list $H(g(b,y_1), g(b,y_2))$ do
 7:         set $c = C_b^{-1}(y_1) \star C_a(x_1)^{\mathsf{inv}}$
 8:         test the key $(a,c,b)$ like in Q.1
 9:     end for
10: end for
```

This gives an algorithm with complexity $\mathcal{O}(2^\ell + m)$.

Q.4 Assume a construction $x \mapsto \mathsf{Enc}_{k_1,\ldots,k_n}(x)$ like the ones in this exercise, i.e. based on the block cipher $x \mapsto C_k(x)$ and using n keys $k_1, \ldots, k_n \in \{0,1\}^\ell$. Given m pairs (x_i, y_i), **estimate** the probability that there exists no tuple $(k'_1, \ldots, k'_n)$ different from $(k_1, \ldots, k_n)$ such that for all i, we have $\mathsf{Enc}_{k'_1,\ldots,k'_n}(x_i) = y_i$. **Explain** how to select a minimal m so that this probability is high (i.e. very close to 1).
NOTE: We consider "reasonable" constructions $\mathsf{Enc}_{k_1,\ldots,k_n}(x)$. I.e., there may be weird counterexamples for which the results of this question are different.

Let k designate the right key tuple and k' be a bad one (i.e., $k' \neq k$). The probability that $\mathsf{Enc}_{k'}(x_i) = y_i$ for all i is $2^{-m\ell}$. The probability p that there is one such k' is bounded by the sum over all possible k' that k' is such, namely $p \leq 2^{n\ell} \times 2^{-m\ell}$. So, the probability that there is no such k' is at least $1 - 2^{-(m-n)\ell}$. Hence, by taking $m = n+1$, the probability is at least $1 - 2^{-\ell}$, which is very high.

3 A Variant of Diffie-Hellman in $\mathbf{Z}_n^*$

In this exercise, we want to adapt the Diffie-Hellman protocol in the group $\mathbf{Z}_n^*$ for $n = pq$ where p and q are two different odd prime numbers which are not known by anyone.

We consider the following protocol:

$$\begin{array}{rcl}
\textbf{Alice} & & \textbf{Bob} \\
\text{pick } g \in \mathbf{Z}_n^* \text{ at random} & & \\
\text{pick } x \in \{1, \ldots, B\} \text{ at random} & & \\
X \leftarrow g^x \bmod n & \xrightarrow{\quad g,X \quad} & \\
 & \xleftarrow{\quad Y \quad} & \text{pick } y \in \{1, \ldots, B\} \text{ at random} \\
 & & Y \leftarrow g^y \bmod n \\
K \leftarrow Y^x \bmod n & & K \leftarrow X^y \bmod n
\end{array}$$

where B is an integer which is at least n^2. We will see that using this protocol is not a good idea.

Q.1 What is missing in the above variant of the Diffie-Hellman protocol? (Compare to what was explained in class.)

– *We did not check that $X, Y \in \langle g \rangle$.*
– *We did not use a KDF either to fix the bad distribution of g^{xy}.*
– *We did not check low orders, e.g. $X \neq 1$, $X^2 \neq 1$, etc.*

Ideally, we should work in a group of prime order, but this is not the case in $\mathbf{Z}_n^*$. This would limit the last problem to checking $X \neq 1$.

Q.2 **Prove** that $\mathbf{Z}_n^*$ is not cyclic. Then, **explain** why the protocol added g in the communication.

HINT: Show that $\mathbf{Z}_n^*$ is isomorphic to $\mathbf{Z}_{p-1} \times \mathbf{Z}_{q-1}$ and find elements of order two.

HINT^2: It is not necessary to follow the previous hint.

We have a direct proof here: We use the Legendre symbols (x/p) and (x/q).
For any $\varepsilon_p = \pm 1$ and $\varepsilon_q = \pm 1$, there exist x_p and x_q such that $(x_p/p) = \varepsilon_p$ and $(x_q/q) = \varepsilon_q$. Using the Chinese Remainder Theorem, $x = \mathsf{CRT}(x_p \bmod p, x_q \bmod q)$ is such that $x \bmod p = x_p$ and $x \bmod q = x_q$, so $(x/p) = (x_p/p) = \varepsilon_p$ and $(x/q) = (x_q/q) = \varepsilon_q$. Hence, for any $(\varepsilon_p, \varepsilon_q)$, there are values $x \in \mathbf{Z}_n^*$ such that $(x/p) = (x_p/p) = \varepsilon_p$ and $(x/q) = (x_q/q) = \varepsilon_q$. A generator g would have to generate elements x for all these combinations.
Let g be in $\mathbf{Z}_n^*$.

- *If $(g/p)(g/q) = +1$, g only generates values x such that $\varepsilon_p = \varepsilon_q$, so not the entire $\mathbf{Z}_n^*$.*
- *If $(g/p) = +1$, g only generates values x such that $\varepsilon_p = +1$, so not the entire $\mathbf{Z}_n^*$.*
- *If $(g/q) = +1$, g only generates values x such that $\varepsilon_q = +1$, so not the entire $\mathbf{Z}_n^*$.*

This covers all possible g.
We can also follow the hint to have an alternate proof. Due to the Chinese Remainder Theorem, the $\mathbf{Z}_n$ ring is isomorphic to the $\mathbf{Z}_p \times \mathbf{Z}_q$ ring. Their group of units are isomorphic groups. So, $\mathbf{Z}_n^*$ is isomorphic to $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$, which is itself isomorphic to $\mathbf{Z}_{p-1} \times \mathbf{Z}_{q-1}$. Then $\mathbf{Z}_{p-1} \times \mathbf{Z}_{q-1}$ has two elements $(\frac{p-1}{2}, 0)$ and $(0, \frac{q-1}{2})$ which have order two. So, the group cannot be isomorphic to any $\mathbf{Z}_m$ which has at most one element of order two (namely, $m/2$ if integer). Hence, it cannot be cyclic.
Since $\mathbf{Z}_n^*$ is not cyclic, there is no generator. So, we have to specify which g generates X and Y. This is why it was added in the protocol. Namely, it is easy to pick a random element g in $\mathbf{Z}_n^*$ and work in the subgroup $\langle g \rangle$ generated by g.
Finally, we could also prove that $\mathbf{Z}_n^*$ is not cyclic be observing that $\lambda(n) < \varphi(n)$. Indeed, since both p and q are odd, we have $\lambda(n) = \mathsf{lcm}(p^{\alpha-1}(p-1), q^{\beta-1}(q-1))$ and $\varphi(n) = p^{\alpha-1}(p-1) \times q^{\beta-1}(q-1)$. Since both $p-1$ and $q-1$ are even, the lcm is lower than the product. So, we have no element of order $\varphi(n)$ as orders are always bounded by $\lambda(n)$.

Q.3 **Explain** why can't we pick $x, y \in \mathbf{Z}_m$ where m is the order of g? Instead, we pick $x, y \in \{1, \ldots, B\}$ uniformly, where B is an integer such that $B \geq n^2$. Let m be an integer such that $m < n$. **Prove** that for any fixed value X,

$$\left| \Pr[x = X] - \Pr[y \bmod m = X] \leq \frac{1}{B} \right|$$

where x is uniformly distributed in $\mathbf{Z}_m$ and y is uniformly distributed in $\{1, \ldots, B\}$.

We cannot pick $x, y \in \mathbf{Z}_m$ because we do not know m.
For x uniformly distributed in $\mathbf{Z}_m$, we have $\Pr[x = X] = \frac{1}{m}$. Let $B = mt + r$ be the Euclidean division of B by m, so with $t = \lfloor \frac{B}{m} \rfloor$. For y uniformly distributed in $\{1, \ldots, B\}$, we have $\Pr[y = X] = \frac{t}{B}$ or $\Pr[y = X] = \frac{t+1}{B}$, depending on the value of X. We have $\frac{t}{B} \leq \frac{1}{m} \leq \frac{t+1}{B}$. So, $|\Pr[x = X] - \Pr[y = X]| \leq \frac{1}{B}$.

Q.4 Prove that there is a polynomial-time algorithm $\mathcal{D}$ such that

$$\Pr[\mathcal{D}(g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(g, g^x, g^y, g^z) = 1] \geq \frac{1}{4}$$

where $g \in \mathbf{Z}_n^*$ is uniform and $x, y, z \in \mathbf{Z}_m$ are uniform, with m equal to the order of g in $\mathbf{Z}_n^*$.
HINT: use the Jacobi symbol $(\cdot/n)$ and reconstruct a distinguisher like the one seen in class. Consider the cases $(g/n) = +1$ and $(g/n) = -1$.

We define $\mathcal{D}(g, X, Y, Z)$. If $(g/n) = +1$, the algorithm answers 1. If $(g/n) = -1$, the algorithm computes $a = 1_{(Z/n)=-1}$ and $b = 1_{(X/n)=(Y/n)=-1}$ and returns $1_{a=b}$. We can easily see that for $X = g^x$, $Y = g^y$, and $Z = g^z$, we have $a = z \bmod 2$ and $b = xy \bmod 2$. So, we have

$$\mathcal{D}(g, g^x, g^y, g^z) = 1_{(g/n)=+1 \text{ or } xy \equiv z \pmod 2}$$

Clearly, $\Pr[\mathcal{D}(g, g^x, g^y, g^{xy}) = 1] = 1$. Similarly, $\Pr[\mathcal{D}(g, g^x, g^y, g^z) = 1 | (g/n) = +1] = 1$ and $\Pr[\mathcal{D}(g, g^x, g^y, g^z) = 1 | (g/n) = -1] = \frac{1}{2}$. So, $\Pr[\mathcal{D}(g, g^x, g^y, g^z) = 1] = \frac{3}{4}$. Hence

$$\Pr[\mathcal{D}(g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(g, g^x, g^y, g^z) = 1] = \frac{1}{4}$$