Advanced Cryptography — Midterm Exam
Solution

Serge Vaudenay

10.5.2015

- duration: 3h
- any document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **<u>not</u>** 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 Recovering a Secret RSA Modulus

Some people use RSA signature with exponent $e = 2^{16} + 1$ but they use too small prime numbers p and q to be secure. So, to prevent n from being factored, they decide to keep $n = pq$ secret. Only legitimate verifiers will receive n.

Q.1 Given a message $m \in \mathbf{Z}_n$ and a valid signature s, show that we can easily recover a multiple of n.

> *Since $s^e \bmod n = m$, the integer $s^e - m$ is a multiple of n. It is quite big though.*

Q.2 What is the complexity?

> *For $i = 1$ to 16, we have to iteratively square a number of size originally $\log n$. So, the complexity of the ith iteration is $\mathcal{O}((2^i \log n)^2)$. Hence, the final complexity is $\mathcal{O}((e \log n)^2)$.*

Q.3 Given a prime number r, what is roughly the probability that r divides the multiple of n recovered in Q.1? (Assume that m is random.)

> *We have a random multiple x of n. So, r is a factor of it if and only if $x \bmod r = 0$, so with probability $\frac{1}{r}$.*

Q.4 With two message/signature (m_i, s_i) pairs, show that we can recover n with high probability.

Given one pair, we recover a random multiple of n. We can easily remove the small prime factors of this multiple. Finding really small factors can be done by trial division. Other small factors can be found by the ECM method.
Then, we are left with hard-to-find random prime factors r which are different from p and q. The number of prime factor r which are at least B and divide both numbers is bounded by $\sum r^{-2}$ when we sum over all prime numbers larger than B. So, this is bounded by $\frac{1}{B}$. This means that as B is large enough, we should have no prime factor $r > B$ is common.
As the probability that the same big random prime factor appears in the two computations, the gcd of two recovered random multiples of n will yield n once the small factors are removed.

2 Finding Four-Term Zero Sums

The exercise is inspired by A Generalized Birthday Problem *by Wagner. Published in the proceedings of Crypto'02 pp. 288–303, LNCS vol. 2442, Springer 2002.*

Looking for collisions is frequent in cryptography. A collision of bitstrings is nothing but a two-term zero sum, using the XOR (denoted by $\oplus$) to define addition. A variant of this problem is to find four-term zero sums. For instance, if we define the signature of a pair of strings (x_1, x_2) of specific format to be the signature of $x_1 \oplus x_2$, we have a forgery attack by looking for a four-term zero sum $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$ with strings x_1, x_2, x_3, x_4 taken from lists of strings of a specific format.

In what follows, we call *a random list of ℓ-bit strings* the sequence $L = (x_1, \ldots, x_n)$ obtained by picking all x_i independently uniformly at random in $\{0,1\}^\ell$. We call n the *length* of the list. We denote by $\oplus$ the bitwise XOR operation between bitstrings.

Q.1 Given two lists L_1 and L_2 of length n_1 and n_2, respectively, in the following subquestions, we consider algorithms to find all (i, j) pairs such that the ith element of L_1 and the jth element of L_2 give a XOR of zero.

Q.1a Compute n_3, the expected number of such pairs (i, j).

The number of (i, j) pairs is $n_1 n_2$. Each pair is valid with probability $2^{-\ell}$. So, the expected number of pairs is $n_1 n_2 2^{-\ell}$. More precisely,

$$E(n_3) = E\left(\#\{(i,j); x_i = y_j\}\right) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \Pr[x_i = y_j] = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} 2^{-\ell} = n_1 n_2 2^{-\ell}$$

Q.1b Give an algorithm with complexity $\mathcal{O}(n_1\ell + n_2\ell + n_3 \log \max(n_1, n_2))$ to find these n_3 pairs.

We first scan $L_1 = (x_1, \ldots, x_{n_1})$. For each $i = 1, \ldots, n_1$, we insert i in a hash table at position $h(x_i)$. This takes n_1 iterations where we essentially have to read x_i and hash it. Assuming that hashing an ℓ-bit string takes $\mathcal{O}(\ell)$, we obtain $\mathcal{O}(n_1\ell)$. Then, we scan $L_2 = (y_1, \ldots, y_{n_2})$. For each $j = 1, \ldots, n_2$, we look at position $h(y_j)$. Each i that we find produce the output (i, j). This takes n_2 iterations where we essentially have to read y_j and hash it, so $\mathcal{O}(n_2\ell)$. We further have to enumerate all matching i. This sums to n_3 iterations for showing (i, j). Assuming that printing (i, j) takes $\mathcal{O}(\log \max(n_1, n_2))$ (the bit-sze of i and j), we obtain $\mathcal{O}(n_3 \log \max(n_1, n_2))$.
Here, we neglected the cost of storing many i's at the same place. These are collisions in the hash function. If collisions are rare, this approximation is valid.
An alternate approach is to sort L_1 and L_2 then scan both lists at the same time to see collisions and copy them in the final list. Sorting L_1 takes $\mathcal{O}(n_1 \log n_1)$ string comparison, so $\mathcal{O}(n_1\ell \log n_1)$. The same holds for sorting L_2. The complexity for scanning and comparing is $\mathcal{O}(n_1\ell + n_2\ell)$ so absorved by the complexity of sorting. The complexity of copying the result is $\mathcal{O}(n_3 \log \max(n_1, n_2))$.

In the following questions, we discard the ℓ factors from the complexities for simplicity. I.e., the cost of copying or comparing ℓ-bit strings is $\mathcal{O}(1)$. Similarly, copying an index i or j is assumed to take $\mathcal{O}(1)$.

Q.1c What is the optimal value for n_1 and n_2 to make $n_3 = 1$ and minimize the complexity at the same time? What is the complexity with these parameters?

We let n_1 and n_2 be such that $n_1 n_2 = 2^\ell$. We want to minimize $n_1 + n_2 = n_1 + \frac{2^\ell}{n_1}$. The function $x \mapsto x + \frac{2^\ell}{x}$ has a derivative which vanish on $x = 2^{\frac{\ell}{2}}$. It corresponds to the minimum of the function. So, the optimal value is $n_1 = n_2 = 2^{\frac{\ell}{2}}$. The complexity is $\mathcal{O}(2^{\frac{\ell}{2}})$.

Q.2 We denote $L_j = (x_{j,1}, \ldots, x_{j,n})$. Given four lists L_1, L_2, L_3, L_4 of same length n, we want to find tuples (i_1, i_2, i_3, i_4) such that $x_{1,i_1} \oplus x_{2,i_2} \oplus x_{3,i_3} \oplus x_{4,i_4} = 0$.

Q.2a What is the expected number of solutions?
Give an efficient algorithm to find them all and its complexity.

It is $n^4 2^{-\ell}$.
To find them, we can first enumerate all (i_1, i_2) pairs and store (i_1, i_2) at position $h(x_{i_1} \oplus x_{i_2})$ for $L_1 \times L_2$. Then, for all (i_3, i_4) we can check the hash table at position $h(x_{i_3} \oplus x_{i_4})$ and show (i_1, i_2, i_3, i_4) for each (i_1, i_2) found. This works with complexity $\mathcal{O}(n^2 + n^4 2^{-\ell})$.

Q.2b We now want to find all tuples (i_1, i_2, i_3, i_4) such that $x_{1,i_1} \oplus x_{2,i_2}$ and $x_{3,i_3} \oplus x_{4,i_4}$ have both their b most significant bits equal to zero and $x_{1,i_1} \oplus x_{2,i_2} = x_{3,i_3} \oplus x_{4,i_4}$.
What is the expected number of solutions?
Give an algorithm to find them all with complexity $\mathcal{O}(n + n^2 2^{-b} + n^4 2^{-\ell-b})$.

The number of solutions is now $n^4 2^{-\ell-b}$ as we have a constraint on b more bits.
We can first find all (i_1, i_2) pairs such that $x_{1,i_1} \oplus x_{2,i_2}$ start with b zero bits in complexity $\mathcal{O}(n + n^2 2^{-b})$. For each of these pairs, we can store (i_1, i_2) in a hash table at position $h(x_{1,i_1} \oplus x_{2,i_2})$. We can do the same for (i_3, i_4), then find all tuples.

Q.2c Give an optimal b and n such that we can find one expected tuple with zero XOR. Give the corresponding complexity.
NOTE: to simplify the computations, allow b to take any real value.

As we need one solution we can lower n so that $n^4 2^{-\ell-b} = 1$. The complexity is thus $\mathcal{O}(n + n^2 2^{-b})$. Since $n = 2^{\frac{\ell}{4}+\frac{b}{4}}$, the complexity is $\mathcal{O}(2^{\frac{\ell}{4}+\frac{b}{4}} + 2^{\frac{\ell}{2}-\frac{b}{2}})$. To minimize it, we must have $\frac{\ell}{4} + \frac{b}{4} = \frac{\ell}{2} - \frac{b}{2}$ so $b = \frac{\ell}{3}$ thus $n = 2^{\frac{\ell}{3}}$. The final complexity is $\mathcal{O}(2^{\frac{\ell}{3}})$.

Q.2d What is the complexity to obtain $\alpha \le n$ solutions instead of just one?
As an application, give n, b, and the complexity for $\alpha = n$.

We redo the previous computation with $n = \alpha^{\frac{1}{4}} 2^{\frac{\ell}{4}+\frac{b}{4}}$, the complexity is $\mathcal{O}(\alpha^{\frac{1}{4}} 2^{\frac{\ell}{4}+\frac{b}{4}} + \alpha^{\frac{1}{2}} 2^{\frac{\ell}{2}-\frac{b}{2}} + \alpha)$. To minimize it, we must have $\frac{\ell}{4} + \frac{b}{4} + \frac{1}{4}\log_2 \alpha = \frac{\ell}{2} - \frac{b}{2} + \frac{1}{2}\log_2 \alpha$ so $b = \frac{\ell}{3} + \frac{1}{3}\log_2 \alpha$ thus $n = \alpha^{\frac{1}{3}} 2^{\frac{\ell}{3}}$. The final complexity is $\mathcal{O}(\alpha^{\frac{1}{3}} 2^{\frac{\ell}{3}} + \alpha)$.
For $\alpha = n$, we have $\alpha = 2^{\frac{\ell}{2}}$ so $n = 2^{\frac{\ell}{2}}$, $b = \frac{\ell}{2}$, and the complexity is $\mathcal{O}(2^{\frac{\ell}{2}})$.

3 Number of Samples to Distinguish Two Distributions

Given two distributions P_0 and P_1, we recall that the statistical distance $d(P_0, P_1)$ is defined by

$$d(P_0, P_1) = \frac{1}{2} \sum_z |P_0(z) - P_1(z)|$$

We define the Hellinger distance $H(P_0, P_1)$ by

$$H(P_0, P_1) = \sqrt{\frac{1}{2} \sum_z \left(\sqrt{P_0(z)} - \sqrt{P_1(z)} \right)^2}$$

If P is a distribution, we denote by $P^{\otimes n}$ the distribution of the tuple $(X_1, \ldots, X_n)$ where all X_i are independent random variables following the distribution P.

Q.1 Show that

$$H(P_0, P_1) = \sqrt{1 - \sum_z \sqrt{P_0(z)P_1(z)}}$$

By expanding the square in the sum inside the square root, we have

$$H(P_0, P_1) = \sqrt{\frac{1}{2} \sum_z \left(P_0(z) + P_1(z) - 2\sqrt{P_0(z)P_1(z)} \right)}$$

As $\sum_z P_0(z) = \sum_z P_1(z) = 1$, we obtain the result.

Q.2 We have a biased dice with faces numbered from 1 to 6. We consider the distribution P_0 such that $P_0(1) = \frac{1}{6} - \varepsilon$ and $P_0(x) = \frac{1}{6} + \frac{\varepsilon}{5}$ for $x = 2, \ldots, 6$. We consider the uniform distribution P_1.
Compute an asymptotic equivalent of $d(P_0, P_1)$ and $H(P_0, P_1)$ for $\varepsilon \to 0$.
HINT: $\sqrt{1+t} = 1 + \frac{1}{2}t - \frac{1}{8}t^2 + o(t^2)$ when $t \to 0$.

We have $d(P_0, P_1) = \varepsilon$ and $1 - H(P_0, P_1)^2 = \frac{1}{6}\sqrt{1 - 6\varepsilon} + \frac{5}{6}\sqrt{1 + \frac{6}{5}\varepsilon}$. Since $\sqrt{1+t} = 1 + \frac{1}{2}t - \frac{1}{8}t^2 + o(t^2)$, we obtain that $1 - H(P_0, P_1)^2 = 1 - \frac{9}{10}\varepsilon^2 + o(\varepsilon^2)$. So, $H(P_0, P_1) \sim \frac{3}{\sqrt{10}}\varepsilon$.

Q.3 Using an upper bound for $d(P_0^{\otimes n}, P_1^{\otimes n})$ in terms of $d(P_0, P_1)$, show that for $n \leq n_{0.5}$, the advantage of any distinguisher between P_0 and P_1 using n samples has an advantage lower than 0.5, where

$$n_{0.5} = \frac{0.5}{d(P_0, P_1)}$$

We have seen in the course that $d(P_0^{\otimes n}, P_1^{\otimes n}) \leq nd(P_0, P_1)$. So, for $n \leq n_{0.5}$, we have $d(P_0^{\otimes n}, P_1^{\otimes n}) \leq 0.5$. We have seen in the course that $d(P_0^{\otimes n}, P_1^{\otimes n})$ is the largest advantage we can obtain to distinguish P_0 from P_1 using n samples. Hence, any distinguisher using n samples has an advantage limited to 0.5.

Q.4 One problem with the previous approach is that we do not know what to say when $n \geq n_{0.5}$. Actually, the bound we obtain is very loose, as we will see.

In the following questions, we estimate $d(P_0^{\otimes n}, P_1^{\otimes n})$ in terms of $H(P_0, P_1)$.

Q.4a Show that $1 - H(P_0^{\otimes n}, P_1^{\otimes n})^2 = (1 - H(P_0, P_1)^2)^n$.

We have

$$\begin{aligned}
1 - H(P_0^{\otimes n}, P_1^{\otimes n})^2 &= \sum_{z_1,\ldots,z_n} \sqrt{P_0(z_1,\ldots,z_n)P_1(z_0,\ldots,z_n)} \\
&= \sum_{z_1,\ldots,z_n} \sqrt{P_0(z_1)P_1(z_1)\cdots P_0(z_n)P_1(z_n)} \\
&= \sum_{z_1} \sqrt{P_0(z_1)P_1(z_1)} \cdots \sum_{z_n} \sqrt{P_0(z_n)P_1(z_n)} \\
&= \left(\sum_{z} \sqrt{P_0(z)P_1(z)}\right)^n \\
&= \left(1 - H(P_0, P_1)^2\right)^n
\end{aligned}$$

So, as n grows, we can estimate $H(P_0^{\otimes n}, P_1^{\otimes n})$ using $H(P_0, P_1)$ with no loss at all.

Q.4b Show that

$$H(P_0, P_1)^2 \leq d(P_0, P_1) \leq \sqrt{1 - (1 - H(P_0, P_1)^2)^2}$$

HINT: $\left(\sqrt{a} - \sqrt{b}\right)^2 \leq |a - b| = |\sqrt{a} - \sqrt{b}| \times (\sqrt{a} + \sqrt{b})$

Using the Cauchy-Schwarz inequality, we have

$$\begin{aligned}
d(P_0, P_1) &= \frac{1}{2}\sum_z |\sqrt{P_0(z)} - \sqrt{P_1(z)}| \times (\sqrt{P_0(z)} + \sqrt{P_1(z)}) \\
&\leq \frac{1}{2}\sqrt{\sum_z (\sqrt{P_0(z)} - \sqrt{P_1(z)})^2}\sqrt{\sum_z (\sqrt{P_0(z)} + \sqrt{P_1(z)})^2} \\
&\overset{(1)}{=} H(P_0, P_1)\sqrt{\frac{1}{2}\sum_z (\sqrt{P_0(z)} + \sqrt{P_1(z)})^2} \\
&\overset{(2)}{=} H(P_0, P_1)\sqrt{1 + \sum_z \sqrt{P_0(z)P_1(z)}} \\
&\overset{(3)}{=} H(P_0, P_1)\sqrt{2 - H(P_0, P_1)^2} \\
&= \sqrt{2H(P_0, P_1)^2 - H(P_0, P_1)^4} \\
&= \sqrt{1 - (1 - H(P_0, P_1)^2)^2}
\end{aligned}$$

where (1) *is by definition of* H, (2) *is by expanding the square, and* (3) *is by using Q.1.*
For the other inequality, we have

$$d(P_0, P_1) = \frac{1}{2}\sum_z |P_0(z) - P_1(z)| \geq \frac{1}{2}\sum_z \left(\sqrt{P_0(z)} - \sqrt{P_1(z)}\right)^2 = H(P_0, P_1)^2$$

Q.4c Show that

$$1 - (1 - H(P_0, P_1)^2)^n \leq d(P_0^{\otimes n}, P_1^{\otimes n}) \leq \sqrt{1 - (1 - H(P_0, P_1)^2)^{2n}}$$

This is a direct application of Q.4b with $P_0^{\otimes n}$ *and* $P_1^{\otimes n}$ *followed by a direct application of Q.4a.*

Q.4d Consider that the advantage of the best distinguisher using n samples is an incresing function of n that we extend over the real numbers. Let $n_{0.5}$ be the value of n for which the advantage is 0.5. Show that

$$\frac{0.20}{-\log_2(1 - H(P_0, P_1)^2)} \leq n_{0.5} \leq \frac{1}{-\log_2(1 - H(P_0, P_1)^2)}$$

HINT: $\log_2 \frac{3}{4} \approx -0.4150$.

We know that the best advantage is the statistical distance. So, by using Q.4c, we have

$$1 - (1 - H(P_0, P_1)^2)^{n_{0.5}} \leq 0.5$$

So, by unrolling $n_{0.5}$, we obtain

$$n_{0.5} \leq \frac{\log(0.5)}{\log(1 - H(P_0, P_1)^2)} = \frac{1}{-\log_2(1 - H(P_0, P_1)^2)}$$

By using Q.4c, we have

$$0.5 \leq \sqrt{1 - (1 - H(P_0, P_1)^2)^{2n_{0.5}}}$$

So, by unrolling $n_{0.5}$, we obtain

$$n_{0.5} \geq \frac{\log \frac{3}{4}}{2\log(1 - H(P_0, P_1)^2)} \geq \frac{0.20}{-\log_2(1 - H(P_0, P_1)^2)}$$

Q.5 Compare $n_{0.5}$ from Q.3 and Q.4d for the example of Q.2.

For Q.3, we have $n_{0.5} \sim \frac{0.5}{\varepsilon}$ and no idea about what happens for $n \geq n_{0.5}$. For Q.4d, we have $n_{0.5} \sim \frac{\lambda}{\varepsilon^2}$ with $\frac{0.20 \ln 2}{\frac{9}{10}} \leq \lambda \leq \frac{\ln 2}{\frac{9}{10}}$. So, $0.1540 \leq \lambda \leq 0.7702$. Clearly, we have a much more precise result.