Cryptography, winter term 16/17: Sample solution to assignment 12

Cornelius Brand, Marc Roth

Exercise 12.1 (Be nice to your tutors and TAs, 1 Bonus Point)
Write the name and matriculation number of every author as well as number, time slot and the name of the tutor of your tutorial group on the first page of your solution. Furthermore write in english and staple *all* sheets of your solution.

Exercise 12.2 (Grading, 2 + 2 Bonus Points) The cryptography lecture is graded as follows: Every student scored a percentage p of points in the midterm exam (no-shows scored zero percent). The number of bonus points b_{endterm} for the endterm exam is computed by multiplying p to a quarter of the total number of points achievable in the endterm exam and rounding up. b_{reexam} is computed likewise. Now let N_{endterm} and N_{reexam} be the number of points a student achieved in the endterm and reexam, respectively. Furthermore let T_{endterm} and T_{reexam} be the total number of points achievable in the endterm and reexam, respectively. The final score S of a student is calculated as follows

$$S := \max\left\{\frac{N_{\text{endterm}} + b_{\text{endterm}}}{T_{\text{endterm}}}, \frac{N_{\text{reexam}} + b_{\text{reexam}}}{T_{\text{reexam}}}\right\}$$

The course is passed if $S \geq \frac{1}{2}$.

(a) A student got 18 out of 42 points in the midterm exam. The total number of points in the endterm is 120 and the total number of points in the reexam is 180[1]. In the endterm the student scored 45 points. Is the course already passed? If not: Calculate the minimum number of points the student has to achieve in the reexam to pass the course.

(b) Let T_{endterm} be as above. Furthermore, the total number of points in the midterm exam is 42. Assume a student scored N_{midterm} points in the midterm exam. Compute the minimum number N_{endterm} of points the student has to score in the endterm exam such that the course is passed without taking the reexam.

Solution 12.2 (Grading, 2 + 2 Bonus Points)

(a) It holds that

$$\frac{N_{\text{endterm}} + b_{\text{endterm}}}{T_{\text{endterm}}} = \frac{45 + \lceil \frac{18}{42}\frac{120}{4} \rceil}{120} = \frac{29}{60} < \frac{1}{2}\,.$$

Therefore the student has to take the reexam. It is passed if

$$\frac{N_{\text{reexam}} + \lceil \frac{18}{42}\frac{180}{4} \rceil}{180} \geq \frac{1}{2}\,.$$

Solving this inequality yields $N_{\text{reexam}} \geq 70$.

[1] Those numbers are only for the purpose of this exercise. The actual number of points in the exams may differ.

(b) The course is passed without taking the reexam if

$$\frac{N_{\text{endterm}} + \lceil \frac{N_{\text{midterm}}}{42} \cdot \frac{T_{\text{endterm}}}{4} \rceil}{T_{\text{endterm}}} \geq \frac{1}{2} .$$

Solving this inequality yields

$$N_{\text{endterm}} \geq \frac{T_{\text{endterm}}}{2} - \lceil \frac{N_{\text{midterm}} \cdot T_{\text{endterm}}}{168} \rceil .$$

Exercise 12.3 (Breaking RSA with a parity-oracle, 5 points) Let (N, e) be a fixed RSA public-key, and suppose you are given access to an oracle $\oplus(\cdot)$ that computes the *parity* of x, i.e., the remainder of x after division by two, when given $x^e \bmod N$ as input.

Give an algorithm that, given $y = x^e \bmod N$, reconstructs x. Prove that your algorithm is correct and performs a polynomial (in $\log N$) number of operations and oracle queries.

Solution 12.3 (Breaking RSA with a parity-oracle, 5 points) We will show a different approach as the one given in the book. Instead of reconstructing bits from least to most significant, we will show how to iteratively obtain better approximate values for x.

W.l.o.g. we can assume N is odd (if N were even, we could factor it as $N = 2 \cdot (N/2)$ and henceforth break RSA). Of course, $0 \leq [x \bmod N] < N$, which is equivalent to saying that

$$x \in \left[\frac{0 \cdot N}{2^0}, \frac{1 \cdot N}{2^0} \right) .$$

Now, assume that for some i and t, we know that

$$x \in \left[\frac{t \cdot N}{2^i}, \frac{(t+1) \cdot N}{2^i} \right) .$$

We will show how to use the parity oracle to find t' such that

$$x \in \left[\frac{t' \cdot N}{2^{i+1}}, \frac{(t'+1) \cdot N}{2^{i+1}} \right) .$$

That is, we divide $[0, N]$ into 2^i parts of length $N/2^i$, and we know the part in which x is contained. Then, we want to say in which half of this subinterval x is contained, using the parity oracle.

We will first make a slight detour: Let $0 \leq s < N$ be arbitrary. Consider the integer number $2s$. If $2s < N$, then $[2s \bmod N] = 2s$, and the parity of $2s$ is obviously 0. If $2s \geq N$, then because $0 \leq s < N$, $0 \leq 2s < 2N$, so that $[2s \bmod N] = 2s - N$. We assumed that N is odd, and $2s$ is obviously even, such that $2s - N$ is odd, and hence as parity 1. Therefore, by computing the partiy of $[2s \bmod N]$, we can tell where $s < N/2$ or $s \geq N/2$. Note that this works for *any* s as above.

The rest of the argument then goes as follows. Let return to the situation above, namely, we assume that for some i and t, we know that

$$x \in \left[\frac{t \cdot N}{2^i}, \frac{(t+1) \cdot N}{2^i} \right) .$$

By multiplying with 2^i, we may equivalently state this as

$$tN \leq 2^i x < (t+1)N .$$

This directly implies $[2^i x \bmod N] = 2^i x - tN$. Letting $s = 2^i x - tN$ as in the above argument, the parity of $[2s \bmod N]$ now allows us to decide which of $2s < N$ or $2s \geq N$ was the case. Now, $tN \leq 2^i x < (t+1)N$ is obviously equivalent to $2tN \leq 2^{i+1} x < (2t+2)N$. Equally clear is that $2s = 2 \cdot (2^i x - tN) < N$ is equivalent to $2^{i+1} x < (2t+1)N$, and similarly, $2s \geq N$ if and only if $2^{i+1} x \geq (2t+1)N$. Combining these observations gives: If $2s < N$, then $2tN \leq 2^{i+1} x < (2t+1)N$, if $2s \geq N$, then $(2t+1)N \leq 2^{i+1} x < (2t+2)N$. Letting $t' = 2t$ if $2s < N$ and $t' = 2t+1$ if $2s \geq N$, we can write this in the desired form:

$$x \in \left[\frac{t' \cdot N}{2^{i+1}}, \frac{(t'+1) \cdot N}{2^{i+1}} \right) .$$

Thus, repeating this procedure until $N < 2^{i+1}$ (i.e., $\lceil \log_2 N \rceil$ times) places x in an interval of length less than 1, and in our case, it then has to coincide with the left (non-strict) border, which is the sought number x.

As one might notice, we haven't even used the parity oracle so far. This will now change. Observe that when we compute the parity of $[2^i x \bmod N]$ in the above procedure, we actually don't know x and much less $2^i x$. However, we can query our parity oracle with the input $y^{(i)} = 2^{ei} y = (2^i x)^e$, such that $\oplus(y^{(i)})$ returns the parity of $[2^i x \bmod N]$. This can clearly be done in polynomial time and costs one oracle call for each i. As argued, we need $\lceil \log_2 N \rceil$ many values of i to reconstruct x, so that the algorithm has a running time polynomial in $||N||$.

Exercise 12.4 (Signature schemes and one-way functions, 6 Points)
Let $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ be a *secure* signature scheme for 1-bit messages. Without loss of generality, we can assume that there is a polynomial p such that Gen uses exactly $p(n)$ random bits on input 1^n. Now consider the following algorithm A_f:

(1) On input x, A_f searches n such that $p(n) \leq |x| \leq p(n+1)$[2].

(2) Then A_f simulates $\mathrm{Gen}(1^n)$ with random bits x and obtains a pair (pk, sk).

(3) A_f outputs pk.

Now let f be the function that is computed by A_f. Show that f is a one-way function.

[2]You can assume that this n exists for every x and can be found in polynomial time in $|x|$.

Solution 12.4 (Signature schemes and one-way functions, 6 Points)
First we observe that f can be computed in polynomial time (by A_f). We prove that f is one-way by reduction. Assuming f is not one-way, there exists a ppt adversary $\mathcal{A}$ and a polynomial q such that

$$\Pr[\text{Invert}_{\mathcal{A},f}(n) = 1] \geq \frac{1}{q(n)}$$

Next we construct an adversay $\mathcal{A}'$ for Π:

(1) On input pk, $\mathcal{A}'$ simulates $\mathcal{A}$ on pk to obtain x.

(2) Then $\mathcal{A}'$ simulates $\text{Gen}(1^n)$ with random bits x and obtains a pair of keys (pk', sk').

(3) Finally, $\mathcal{A}'$ outputs $(0, \text{Sign}_{sk'}(0))$.

Now let $\text{Valid}_{n,pk}(pk', sk')$ be the event that $pk = pk'$ and $(pk', sk') \leftarrow \text{Gen}(1^n)$. Then it holds that

$$\begin{aligned}
\Pr[\text{Sig-forge}_{\mathcal{A}',\Pi}(n) = 1] &\geq \Pr[\text{Valid}_{n,pk}(pk', sk')] \cdot \Pr[\text{Sig-forge}_{\mathcal{A}',\Pi}(n) = 1 | \text{Valid}_{n,pk}(pk', sk')] \\
&= \Pr[\text{Valid}_{n,pk}(pk', sk')] \cdot \Pr[\text{Vrfy}_{pk}(0, \text{Sign}_{sk'}(0)) = 1 | \text{Valid}_{n,pk}(pk', sk')] \\
&= \Pr[\text{Valid}_{n,pk}(pk', sk')] \cdot (1 - \text{negl}(n)) \\
&\geq \Pr[\mathcal{A}(pk) = f^{-1}(pk)] \cdot (1 - \text{negl}(n)) \\
&= \Pr[\text{Invert}_{\mathcal{A},f}(|f^{-1}(pk)|) = 1] \cdot (1 - \text{negl}(n)) \\
&\geq \frac{1}{q(p(n+1))} \cdot (1 - \text{negl}(n)) \\
&= \frac{1}{q(p(n+1))} - \frac{\text{negl}(n)}{q(p(n+1))}
\end{aligned}$$

which is not negligible.

Exercise 12.5 (From fixed- to arbitrary-length signatures, 5 Points) Adapt Construction 4.7 (see the lecture slides) for constructing variable-length MACs from fixed-length MACs so that it produces a signature scheme for arbitrary-length messages from any signature scheme for fixed-length messages of length $\ell(n) \geq n$. It may be advisable to go through the proof of Theorem 4.8 and to adapt it to the new requirements.

Solution 12.5 (From fixed- to arbitrary-length signatures, 5 Points) The proof that Construction 4.7 also works for digital signature schemes follows exactly along the same lines as the proof of Theorem 4.8.