Family Name: . . . .

First Name: . . . .

Section: . . . .

Advanced Cryptography

Midterm Exam
Solution

May 22th, 2007

Duration: 3 hours 45 minutes

This document consists of **??** pages.

Instructions

Electronic devices are *not* allowed.

Answers must be written on the exercises sheet.

This exam contains 2 *independent* exercises.

Answers can be either in French or English.

Questions of any kind will certainly *not* be answered. Potential errors in these sheets are part of the exam.

You have to put your full name on the first page and have all pages *stapled*.

1 Substitution-Permutation Networks

1. For the AES, we have $n = 128, k = 128, b = 16$

2. The linear transformation of the AES is the $\mathsf{MicColumns}$ subroutine.

3. Since the input x is different from 0, $\mathsf{hw}(x) \geq 1$. Each round is a permutation implies that an input difference will produces an output difference and thus $\mathsf{hw}(y) \geq 1$. So, $B \geq 2$.

   Now, we can limit the input diffence to one block, i.e. $\mathsf{hw}(x) = 1$. The maximal output difference is when all blocks are differents, i.e. $\mathsf{hw}(y) \leq b$. So, $B \leq b + 1$.

4. Consider $x = (x_1, ldots, x_b)$ and $x' = (x'_1, ldots, x'_b)$ where each x_i, x'_j belongs to the set W. Let $y = (y_1, ldots, y_b) \leftarrow f(x_1, ldots, x_b)$ and $y' = (y'_1, ldots, y'_b) \leftarrow f(x'_1, ldots, x'_b)$.

   f is a multipermutation if and only if a difference of r elements between x and x' implies a difference of $b - r + 1$ elements between y and y'.

5. We consider the tuples $(x_1, ldots, x_b, y_1, ldots, y_b)$ and $(x'_1, ldots, x'_b, y'_1, ldots, y'_b)$ form the previous question. f is a multipermutation if and only if there is at least $b + 1$ different elements between these two tuples which means that the branch number B of f is $b + 1$.

6. In question 1, we saw that the linear transformation is $\mathsf{MixColumns}$. Thus, L (in this case) is a multipermutation with 16 elements of 8 bits to 16 elements of 8 bits. So, the branch number is 17.

7. **When $r = 1$:** We consider one round of C. Thus, there is no linear transformation. There is at least 1 active s-box since the input is different. So, $\ell = 1$.

   When $r = 2$: We consider two rounds of C. There is at least one different block at the input of C, so at least one active s-box at the first round. This implies that there is at least one different element at the input of L.

   Around L, there is at least B active s-boxes. In other words, there is at least B actives s-boxes in this round and in the next one. So, $\ell = B$.

   For any $r \geq 3$: There is B active s-boxes around each linear transformation. So, the $r-1$ first rounds, contains at least B active s-boxes around it. The last round contains only one active s-box. Let n_A, resp. n_B, the number of actives s-boxes at the input, resp. output. We have

   $$2\ell = n_A + B + B + B + \ldots + B + n_B$$

   Here, we counted two times each active s-box. Finally, we have $\ell = \frac{(r-1)\cdot B}{2} + 1$.

8. Let $\Omega^* = (\Delta_1^*, \Delta_2^*, \ldots, \Delta_r^* + 1)$ be the differential charaterisitc which maximizes $\mathsf{P}(\Omega^*)$. A differential charaterisitc uses two distinct input for C and so we know that there is at least ℓ actives s-boxes. Let ℓ_i be the number of actives s-boxes at round i for $1 \leq i \leq r$. Naturally, we have $\sum_{i=1}^{r} \ell_i = \ell$.

   We can write:

   $$\begin{aligned}
   \mathsf{P}_{\max} &= \max_{\Delta_1,\Delta_2,\ldots,\Delta_r+1} \mathsf{DP}^{C_1}(\Delta_1, \Delta_2) \cdot \ldots \cdot \mathsf{DP}^{C_r}(\Delta_r, \Delta_{r+1}) \\
   &= \mathsf{DP}^{C_1}(\Delta_1^*, \Delta_2^*) \cdot \ldots \cdot \mathsf{DP}^{C_r}(\Delta_r^*, \Delta_{r+1}^*) \\
   &\leq (\mathsf{DP}^S_{\max})^{\ell_1} \cdot \ldots \cdot (\mathsf{DP}^S_{\max})^{\ell_r} \\
   &= (\mathsf{DP}^S_{\max})^{\ell}
   \end{aligned}$$

9. Using the previous questions, we obtain:

$$\mathsf{P}_{\max} = (2^{-6})^{\frac{(r-1)B}{2}+1}$$

So,

	$\mathsf{P}_{\max}$
$r = 4$	2^{-129}
$r = 6$	2^{-251}
$r = 8$	2^{-363}

10. We have to enumerate the possiblities... The best solution is 4-1-4 and the branch number is 8.

11. Let B' the branch number of G, i.e. two rounds. There is B' active s-boxes on each two-rounds. So,

$$\mathsf{P}_{\max} = (\mathsf{DP}^{S}_{\max})^{\frac{r}{2}B'}$$

12. $B' = 8$. So,

	$\mathsf{P}_{\max}$
$r = 4$	2^{-96}
$r = 6$	2^{-144}
$r = 8$	2^{-192}

2 Finding Collisions

1. As $\mathcal{D}$ is finite then after at most $N = \#\mathcal{D}$ iteration of f, we must obtain some point twice. Starting from this point the sequence is periodic.

2. For $\theta = \lambda$, the point x is always λ steps ahead compared to y. Thus, in the loop we eventually obtain $f(x) = f(y)$ (more precisely, after μ iteration of the loop). The obtain this collision, we perform $\mu + \lambda$ evaluation of f to obtain $x_{\mu+\lambda}$ and μ to obtain x_μ, so that we perform $2\mu + \lambda$ evaluations in total.

3. Algorithm 2 terminates when $x = y$, where y is the *smallest* value obtained in the sequence. If the smallest value belongs to the tail of the sequence, the algorithm never reaches it twice so that it loops forever. If this smallest value belongs to the cycle, the algorithms stops. Among the $\mu + \lambda$ distinct values of the sequence, the probability that the algorithm stops is thus $p = \frac{\lambda}{\mu+\lambda}$, as the smallest value is uniformly distributed.

4. Assuming that the algorithm terminates, the smallest value (which belongs to the cycle) is on average at index $\frac{\lambda}{2}$. Thus, the algorithm performs $\mu + \lambda + \frac{\lambda}{2} = \mu + \frac{3}{2}\lambda$ iterations of f before it reaches this value a second time.

5. Adding the number of iterations needed by both algorithms (and considering that we need to run $\frac{1}{p}$ instances of Algorithm 2 on average to obtain a collision), we get a total of

$$\frac{1}{p} \cdot \left(\mu + \frac{3}{2}\lambda\right) + 2\mu + \lambda = 8 \cdot \sqrt{\pi N/8}$$

   iterations of f to obtain a collision. The memory requirements of this method is negligible, since it is $\Theta(\log N)$.

6. The difference compared to the previous method is that we do not focus on one single smallest value here. We have divided the space in k equivalence classes, and compute the smallest value obtained for each of these k classes. As soon as one of these smallest equivalence class values occurs twice, it is detected and the value of λ is returned. Consequently, the algorithm terminates if at least one of the smallest values (among those obtained for the different equivalence classes) belongs to the cycle. Considering that f is uniformly distributed, so that the x_i's are mutually independent and uniformly distributed, each smallest equivalence class value belongs to the cycle with probability $\frac{\lambda}{\mu+\lambda}$ and the algorithm terminates with probability

$$p' = 1 - \left(1 - \frac{\lambda}{\mu + \lambda}\right)^k .$$

7. We denote by S the number of smallest equivalence class values that belong to the cycle. As we assumed being in the case where the algorithm succeeds, we know that at least one smallest equivalence class value is on the cycle. Considering $\mu = \lambda$, each other equivalence class value is on the cycle with probability $1/2$.

   So, the probability that $S = u$ for $0 < u \leq k$ is equal to the probability that $u - 1$ values (among the $k - 1$ remainings) are on the cycle. We have

$$\Pr[S = u] = k \cdot \binom{k-1}{u-1} \cdot \frac{1}{2^{k-1}}.$$

8. We denote by L the number of evaluations of f. As we assume that the algorithm succeeds, we have $0 < S \leq k$. As the expected value of L is equal to $\mu + \lambda + \frac{\lambda}{u+1}$ when $S = u$, we have

$$\mathrm{E}(L) = \sum_{u=1}^{k} \left(\mu + \lambda + \frac{\lambda}{u+1}\right) \cdot \Pr[S = u] = \mu + \lambda + \lambda \cdot \sum_{u=1}^{k} \frac{1}{u+1} \cdot \Pr[S = u].$$

Using the preivous question, we have

$$\mathrm{E}(L) = \mu + \lambda + \frac{\lambda}{2^k - 1} \cdot \sum_{u=1}^{k} \frac{\binom{k}{u}}{u+1} = \mu + \lambda + \frac{\lambda}{2^k - 1} \cdot \sum_{u=1}^{k} \frac{\binom{k+1}{u+1}}{k+1}.$$

As $\sum_{u=1}^{k} \binom{k+1}{u+1} = \sum_{u=2}^{k+1} \binom{k+1}{u} = 2^{k+1} - k - 2$, We finally obtain

$$\mathrm{E}(L) = \mu + \lambda + \frac{\lambda}{k+1} \left(2 - \frac{k}{2^k - 1}\right).$$

9. From the previous question, we obtain that approximatively $2 \cdot \sqrt{\pi N/8}$ are needed to find λ. Adding the number of iterations needed by Algorithm 1 leads to the announced result. This method has a space complexity equal to $\Theta(k \cdot \log(N))$.