HIT — Cryptography — Solutions 3

1603202-1150810613-Qiuhao Li

December 31, 2018

Problem 1. In our attack on a two-round substitution-permutation network, we considered a block length of 64 bits and a network with 16 S-boxes that each take a 4-bit input.

1. Repeat the analysis for the case of 8 S-boxes, each taking an 8-bit input. What is the complexity of the attack now?
2. Repeat the analysis again with a 128-bit block length and 16 S-boxes that each take an 8-bit input.
3. Does the S-boxes length make any difference? Does the block length make any difference?

Solution 1.
Actually, the solution on the slides for 64-bits block with 16-Sboxes network is not optimal, we should enumerate on k_1 instead of k_2, and the total time for adversary to try should be $16 * 2^4 = 2^8$, please refer to the answer of the first question (or the errata of the textbook) for more information.

1. Given some input/output pairs (x, y) as before. First, the adversary can enumerate over all possible values for the first byte of k_1. It can XOR each such value with the first byte of x to obtain a candidate value for the input of the first S-box. Evaluating this S-box, the attacker learns a candidate value for the output of that S-box. Since the output of that S-box is $XORd$ with 8 bits of k_2 to give 8 bits of y (where the positions of those bits depend on the mixing permutation and are known to the attacker), this yields a candidate value for 8 bits of k_2.

   To summarize: for each candidate value of the first byte of k_1, there is a unique possible corresponding value for some 8 bits of k_2. Put differently, this means that for some 16 bits of the master key, the attacker has reduced the number of possible values for those bits from 2^{16} to 2^8 . So the total time to do this, is $8 * 2^8 = 2^{11}$.

   The attacker knows that the correct value from thatlist must be consistent with any additional input/output pairs the attacker learns. Heuristically, any incorrect value from the list is consistent with some additional input/output pair (x0, y0) with probability no better than random guessing. Let's say, the adversary has l pairs of (x, y), then we can conclude that after $8 * 2^8 = 2^{11}$ times enumerating he can win with posibility of $(1 - 2^{-8*(l-1)})^8$.

2. As said above, using the same method, now we can conclude the adversary can win after $16 * 2^8 = 2^{12}$ times enumerating with posibility of $(1 - 2^{-8*(l-1)})^{16}$.

3. They both "make a difference". Assuming that the length of the S-box and Block is S and B (S is less than B and divisible to B). As said above, the adversary can win after $\frac{B}{S} * 2^S$ times enumerating with posibility of $(1 - 2^{-S*(l-1)})^{\frac{B}{S}}$.

Problem 2. Show that DES has the property that $DES_k(x) = \overline{DES_{\overline{k}}(\overline{x})}$ for every key k and input x (where $\overline{z}$ denotes the bitwise complement of z). This is called the complementarity property of DES.

Solution 2.

Proof. Let f be the DES mangler function. Since first step of f is XOR the 48-bit sub-key and 48-bit intermediate expanded from 32-bit input, and rest steps have nothing to do about the key and input, we can conclude that for every subkey k and message x, it holds that:

$$f(k, x) = f(\overline{k}, \overline{x}) \tag{1}$$

The heart of DES is the Feistel network, whose one stage algorithm is described by:

$$L_{i+1} = R_i \tag{2}$$

$$R_{i+1} = L_i \oplus f(R_i, K_i) \tag{3}$$

Let c be the "flip bit function", For $DES(L_0R_0, K)$, define $L'_0 = c(L_0)$, $R'_0 = c(R_0)$ and $K'_i = c(K_i)$, which leads to another instance $DES(L'_0R'_0, K')$. Now I will show that for any stage of the Feistel network, $L'_i = c(L_i)$ and $K'_i = c(K_i)$.

- Base: the case when $i = 1$. For instance $DES(L_0R_0, K)$,

$$L_1 = R_0 \tag{4}$$

$$R_1 = L_0 \oplus f(R_0, K_0) \tag{5}$$

For instance $DES(L'_0R'_0, K')$,

$$L'_1 = R'_0 = c(R_0) = c(L_1) \tag{6}$$

$$R'_1 = L'_0 \oplus f(R'_0, K'_0) = c(L_0) \oplus f(c(R_0), c(K_0)) = c(L_0) \oplus f(R_0, K_0) = c(R_1) \tag{7}$$

- Induction: Assume the claim holds for all $i < n$, consider the case when $i = n$. For instance $DES(L_0R_0, K)$,

$$L_n = R_{n-1} \tag{8}$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_{n-1}) \tag{9}$$

For instance $DES(L'_0R'_0, K')$,

$$L'_n = R'_{n-1} = c(R_{n-1}) = c(L_n) \tag{10}$$

$$R'_n = L'_{n-1} \oplus f(R'_{n-1}, K'_{n-1}) = c(L_{n-1}) \oplus f(R_{n-1}, K_{n-1}) = c(R_n) \tag{11}$$

Therefore, after the Feistel network, we can get the output of $DES(L_0R_0, K)$ equals the reverse output of $DES(L'_0R'_0, K')$, which means:

$$DES_k(x) = \overline{DES_{\overline{k}}(\overline{x})} \tag{12}$$

□

Problem 3. Is the addition function $f(x, y) = x + y$ (where $|x| = |y|$ and x and y are interpreted as natural numbers) a one-way function?

Solution 3.

No, it isn't.

Simply speaking, given $f(x, y)$, the adversary have 2^{n-1} choices, which let him invert the function in polynomial time. For example, given $f(x, y) = 10111$ and $n = 4$, the adversary can choose from $(1000, 1111)$ to $(1111, 1000)$ for (x, y).

Problem 4. Let $f_1(x)$ and $f_2(x)$ be one-way functions. Is $f(x) = (f_1(x), f_2(x))$ necessarily a one-way function? Prove your answers.

Solution 4.

No, it isn't.

Proof. Let $g(x)$ be an one-way function, and

$$f_1(x_1||x_2) = g(x_1)||x_2 \tag{13}$$

$$f_2(x_1||x_2) = g(x_2)||x_1 \tag{14}$$

Obviously, $f_1(x)$ and $f_2(x)$ are both one-way functions, but for $f(x) = (f_1(x), f_2(x))$, which imply

$$f(x) = (f_1(x_1||x_2), f_2(x_1||x_2)) = (g(x_1)||x_2, g(x_2)||x_1) \tag{15}$$

So the adversary can get the $x = x_1||x_2$ with ease. □

Problem 5. Let f be a one-way function. Is $g(x) = f(f(x))$ necessarily a one-way function? What about $g(x) = (f(x), f(f(x)))$? Prove your answers.

Solution 5.

$g(x) = f(f(x))$ is not necessarily a one-way function.

Proof. Let $h(x)$ be an one-way function, and we can construct the one-way function $f(x_1||x_2) = h(x_2)||0^n$.

So, $g(x_1||x_2) = f(f(x_1||x_2)) = f(h(x_2)||0^n) = h(0^n)||0^n$, which indicates that f is a constant function and thus not a one-way function. □

As for $g(x) = (f(x), f(f(x)))$, it is. I will prove this by contradiction using reduction.

Proof. Truth: A function must be either one-way function or not.

Assuming that $g(x) = (f(x), f(f(x)))$ is not a one-way function, which means given $g(x) = (a, f(a))$, the adversary find the $x = c$ in poly time with posibility better than $negl(n)$.

Then we can find a way to attack $f(x)$: Given $f(x) = a$, the adversary can use the $invert - g(x)$ algorithm with $(a, f(a))$ as input. Obviously, we can break $f(x)$ in poly time with posibility better than $negl(n)$.

But, we already know $f(x)$ is an one-way function, so any adversary can't achieve finding the $x = c$ in poly time with posibility better than $negl(n)$. Thus our assumption is wrong, which means $g(x)$ is a one-way function. □

Happy  Hacking!