Chapter 06

Mingjia Huo

Problem 6.1.

1. First 10 bits are $1, 1, 1, 1, 1, 1, 0, 1, 0, 1$
2. Compute the First 69 bits:
   1,1,1,1,1,1,0,1,0,1,0,1,1,0,0,
   1,1,0,1,1,1,0,1,1,0,1,0,0,1,0,
   0,1,1,1,0,0,0,1,0,1,1,1,1,0,0,
   1,0,1,0,0,0,1,1,0,0,0,0,1,0,0,
   0,0,0,1,1,1,1,1,1
   After 63 bits, the block becomes the same with initial block $(1, 1, 1, 1, 1, 1)$. Thus it's maximal length.

Problem 6.6(b). Assume the key of the first and third round is k_1, and the key of second round is k_2. Giving a single input/output pair (x, y), fixing k_1:

- Compute $x_1 = x \oplus k_1$, this is the output of the first key-mixing step.
- Since we know the details of S-box substitution and mixing permutation, we can compute the output of the first round, denoted as x_1'.
- Using k_1, we can also compute $y \oplus k_1$ to be the value before the third key-mixing, denoted as x_2'.
- Given x_2', compute the inverse of S-box substitution and mixing permutation in round two, denoted as x_2.
- Thus $k_1 = x_1' \oplus x_2$.

Thus, for each choice of k_1, there is only one possible k_2. So the attack can use time no more than 2^{64}, much less than 2^{128}.

Problem 6.13. (a).

Giving input/output pairs $(x, y), k_1 \in \{0, 1\}^n$, we have $y = F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$.

First, if k_1 is the valid key, we can compute $F_{k_1}^{-1}(y) = y_1$ in constant time. Then compute $x_1 = F_{k_1}(x)$. Thus, with $y_1 = F_{k_2}^{-1}(x_1)$, we can get all k_2 in constant time. Since there are totally three pairs, we can use the other two pairs of input/output values to verify whether (k_1, k_2) is valid.

Given x, the output y has 2^n choices, thus the error rate is aproximately 2^{-2n}.

There are 2^n choices of k_1, thus the probability that we find the right keys is $(1 - 2^{-2n})^{2^n} \approx e^{2^{-n}}$, which is negligble to n.

Thus with high probability, we can recover the entire key.

(b).

Denote that K_m is the set of k such that $F_k^{-1}(0^n) = m$. All K_m forms the set $\mathcal{K}$. In preprocessing, We can construct a table of $\{0, 1\}^n \to \mathcal{K}$ by simply enumerate all k, which takes 2^n time.

$\forall k \in \{0, 1\}^n$, compute $m = F_k^{-1}(0^n)$, then add k to K_m, which saves in the table described above.

After preprocessing, given m_2, we can look up the table, find the line of m_2, and get the all keys which satisfies the condition in constant time.

(c).

1. Compute $F_{k_1}^{-1}(0^n)$ and denote as x.

2. Choose x as input, get access to the encryption oracle, then we get y.
3. Compute $F_{k_1}^{-1}(y)$ and denote as y'.
4. Thus, $y' = F_{k_2}^{-1}(0^n)$. Use the method in problem (b) to get all k_2 in constant time.

The procedure above takes constant time.

(d).

1. Preprocessing as the method in problem (b). (roughly 2^n time.)
2. Fixing k_1, run as the process in problem (c). (When k_1 is fixed, the time of (c) is constant, and only need a single chosen inputs. Thus, this step needs roughly 2^n time and 2^n chosen inputs.)
3. When we get a valid (k_1, k_2), use another two input/output pairs to verify it. (The probability that a key pair is valid is roughly 2^{-n}. Thus this step takes constant time and inputs.)

To sum up, this attack needs roughly 2^n time and 2^n chosen inputs.

Problem 6.19. (a).

In ideal-cipher model, F is a permutation. Attack:

1. Randomly choose k_1, k_2.
2. Ask the oracle $F_{k_1}^{-1}(0^n), F_{k_2}^{-1}(0^n)$, and get x_1, x_2.
3. Then $(k_1, x_1) = (k_2, x_2)$ is a collision. That is $H(k_1, x_1) = F_{k_1}(x_1) = 0^n = F_{k_2}(x_2) = H(k_2, x_2)$.

(b).

Proof. Assume there are totally $q(n)$ queries asked by the adversary $\mathcal{A}$, and the length of hash value is n.

If (k, x) are asked, then a hash value $h = F(k, x) \oplus x \oplus k$ can be computed. And if $F^{-1}(k, y)$ is asked, then the answer x is returned, and $\mathcal{A}$ can compute $x \oplus y \oplus k$ and get the hash value h.

Denote the hash values involved in the $q(n)$ queries as $h_1, h_2, \cdots, h_{q(n)}$, while the key/input/output are denoted as $(k_i, x_i, y_i), 1 \le i \le q(n)$.

A collision is there is $1 \le j < i \le q(n)$, such that $h_i = h_j$.

- Fix $i > j$, consider the probability that $h_i = h_j$.
- Since j is asked earlier, we first get $h_j = F(k_j, x_j) \oplus x_j \oplus k_j$.
- In ith query, there are two cases when a collision happens:
  - $F(k_i, x_i)$ is asked: Since F is ideal-cipher, $F(k_i, x_i)$ can uniformly set to any $y \in \{0,1\}^n$, expect for values answered by $F(k_i, \cdot)$. Thus $F(k_i, x_i)$ equals to $h_j \oplus k_i \oplus x_i$ holds with probability no more than $1/(2^n - i_1)$.
  - $F(k_i, y_i)^{-1}$ is asked: Similarly, x_i equals to $h_j \oplus k_i \oplus y_i$ with probability no more than $1/(2^n - i_1)$.

  Thus a collision $h_i = h_j$ happens with probability $< 1/(2^n - (i-1))$. Since $i \le q(n)$, the probability is less than $\frac{1}{2^{n-1}}$.

Taking a union bound of all pairs of $(i, j), 1 \le j < i \le q(n)$, the collision rate $< \frac{q(n)^2}{2^n}$, which is negligible. □

(c).

Attact:

1. Randomly choose k_1, k_2.

2. Ask the oracle $F_{k_1}^{-1}(k_1), F_{k_2}^{-1}(k_2)$, and get x_1, x_2.

3. Then $(k_1, x_1) = (k_2, x_2)$ is a collision. That is $H(k_1, x_1) = F_{k_1}(x_1) \oplus k_1 = k_1 \oplus k_1 = 0^n = H(k_2, x_2)$.

Problem 6.21. Assume $l = |x| \geq 2|k|$.

A brute search takes roughly $2^{l/2}$ time when there is $\frac{1}{2}$ probability to find a collision.

We can simply enumerate k, until we find the key that it's easy for it to find inputs x for which $F_k(x) = x$. If $F_k(x_1) = x_1, F_k(x_2) = x_2$, then $h(k, x_1) = F_k(x_1) \oplus x_1 = F_k(x_2) \oplus x_2 = h(k, x_2)$, a collision. And it takes roughly $2^{|k|-1}$ time when there is $\frac{1}{2}$ probability to find a collision.

Thus it's better than brute force approach.