Solution Sheet 10

Cryptography and Security 2022

Solution 1 CFB-MAC

1. We can simply query a message x of one block to the oracle $\mathcal{O}$. The oracle returns the value $y = x \oplus \mathsf{E}_k(\text{IV})$. Hence, $\mathsf{E}_k(\text{IV})$ is found by computing $x \oplus y$.

2. After querying n different messages to $\mathcal{O}$, we receive n MAC values for which we would like to get a collision. The probability to have at least one collision is given by the Birthday Paradox. Let $N = 2^{64}$. We know that the probability is approximated by $1 - e^{-\frac{\theta^2}{2}}$, where $n = \theta\sqrt{N}$. In our case, $\theta = 4$ since $\frac{\theta^2}{2} = 8$. Thus, n must at least be equal to $4 \cdot 2^{32} = 2^{34}$.

3. Let $m = x_1 \| x_2 \| \cdots \| x_n$ and $h = \text{CFB-MAC}_k(m)$. We take another message $m' = x_1 \| x_2 \| \cdots \| x_{n-1} \| x'_n$ where x'_n is any block of 64 bits. Since CFB mode is used with a fixed IV the output blocks of m' will be identical to those of m except the last one. Since $h' = \text{CFB-MAC}_k(m') = y_1 \oplus \cdots \oplus y_{n-1} \oplus y'_n$ and $h = y_1 \oplus \cdots \oplus y_n$, we have $h \oplus h' = y_n \oplus y'_n$. We also know that $y_n = \mathsf{E}_k(y_{n-1}) \oplus x_n$ and $y'_n = \mathsf{E}_k(y_{n-1}) \oplus x'_n$. Using these two relations, we finally deduce that $h' = h \oplus y_n \oplus y'_n = h \oplus x_n \oplus x'_n$.

4. Set $m = x_1 \| x_2$ and $x_1 = \mathsf{E}_k(\text{IV}) \oplus \text{IV}$. We then have $y_1 = \text{IV}$ and $y_2 = h \oplus \text{IV}$. Thus, $x_2 = \mathsf{E}_k(y_1) \oplus y_2 = \mathsf{E}_k(\text{IV}) \oplus \text{IV} \oplus h$.

5. Yes, this works in the same way! Set $m = x_1 \| x_2 \| \cdots \| x_n$ where $x_1 = x_2 = \cdots = x_{n-1} = \mathsf{E}_k(\text{IV}) \oplus \text{IV}$. Hence, $y_1 = y_2 = \cdots = y_{n-1} = \text{IV}$. If n is even, setting $x_n = h \oplus \text{IV} \oplus \mathsf{E}_k(\text{IV})$ gives $y_n = h \oplus \text{IV}$ and thus $y_1 \oplus \cdots \oplus y_n = h$. If n is odd, setting $x_n = h \oplus \mathsf{E}_k(\text{IV})$ gives $y_n = h$ and thus $y_1 \oplus \cdots \oplus y_n = h$.

Solution 2 Analysis of the Floyd Cycle Finding Algorithm

1. We simply store in the last loop the previous position.

   Input: an initial string x_0, a function $F : \{0,1\}^* \to \{0,1\}^n$.
   Output: Two elements a', b' such that $F(a') = F(b')$.

   $a \leftarrow x_0$ //(tortoise)
   $b \leftarrow x_0$ //(hare)
   repeat
   $\quad a \leftarrow F(a)$
   $\quad b \leftarrow F(F(b))$
   until $a = b$
   $a \leftarrow x_0$
   while $a \neq b$ **do**
   $\quad a' = a$
   $\quad b' = b$
   $\quad a \leftarrow F(a)$
   $\quad b \leftarrow F(b)$
   end while
   return a' and b'

The algorithm is failing when $\lambda = 0$. In this case, there is no collision. This happen with very small probability.

2. Let j be the iteration at which $a = b$. Note first that this can happen only in the loop and not in the tail since the hare is going twice faster. Hence $j \geq \lambda$. We have

$$2j - \lambda \equiv j - \lambda \pmod{\tau},$$

since we have to remove the tail part. This implies that $j \equiv 0 \pmod{\tau}$ and, hence, that $\tau | j$.

To prove the other direction, suppose we are at a step i such that $i \geq \lambda$ and $\tau | i$. From the first condition, we know that we are in the loop. From the second condition, we know that we can write $i = \tau k$ for some integer k. We also know that we did $2i - \lambda = 2\tau k - \lambda$ steps in the loop with the hare and $i - \lambda = \tau k - \lambda$ steps in the loop with the tortoise. If we look at their position in the loop, i.e., we take the number of steps modulo τ, we have $\tau k - \lambda \equiv 2\tau k - \lambda \pmod{\tau}$. Hence, both the tortoise and the hare are on the same point of the loop. This point is at the position $-\lambda \bmod \tau$ of the loop, i.e., λ steps before the end of the loop.

3. From the previous point, we know that the tortoise and the hare meet *always* when $\tau | i$ and $i \geq \lambda$. To see after how many iterations the loops stops, we are looking for the *smallest* such i. Hence, the number of iterations is the smallest multiple of τ greater than λ. This is obviously (when $\lambda \neq 0$) $\lceil \lambda/\tau \rceil \tau$ since there are $\lceil \lambda/\tau \rceil - 1$ multiples of τ that are smaller than λ.

4. Since we know from question 2 that $a = b$ at point $-\lambda \bmod \tau$ of the loop, we know that after λ steps, the tortoise (a) will have walked on the tail and reach the meeting point between the tail and the loop. On the other hand, the hare (b)(which is now going slowly) will have done λ steps as well and reach the same point.

5. For the first loop, we have $\tau \geq \lambda$ with probability $1/2$. In this case, we have to stop after τ iterations and we need $3\sqrt{\pi 2^n/8}$ evaluations of F in average. If $\tau < \lambda$, we stop after 2τ iterations and we need $6\sqrt{\pi 2^n/8}$ evaluations of F in average. Hence, we have an average complexity for the first loop of $4.5\sqrt{\pi 2^n/8}$

In the second loop, we do $2\sqrt{\pi 2^n/8}$ evaluations of F. Hence, for the whole algorithm, we do $6.5\sqrt{\pi 2^n/8}$ evaluations of F in average.

We need to store four n bit words (a, a', b, b') as well as x_0. Hence, in total, we need only $5n$ bits of memory.