Introduction to cryptology
TD#3

2022-W09,…

Exercise 1: A random sequence *(Exam '18)*

Q. 1: Let $\mathcal{S}$ be a set of size N; let $(u_n)_{n\in\mathbb{N}}$ be a sequence whose elements are drawn independently and uniformly at random from $\mathcal{S}$, i.e. for all i, $u_i \twoheadleftarrow \mathcal{S}$. Suppose that you do not initially know $\mathcal{S}$,[1] nor N.

1. Give an algorithm that takes as input a finite number of elements of (u_n) and that returns an approximation of N.

2. What is the time and memory complexity of your algorithm (be careful to specify the data structures you may use)?

Exercise 2: Hash functions *(Exam '19)*

In the following questions, $\mathcal{H} : \mathcal{I} \to \{0,1\}^n$ is a cryptographic hash function, where $\mathcal{I} = \bigcup_{\ell=0}^{2^N} \{0,1\}^\ell$. We recall the two following definitions:

— A *second preimage attack* on $\mathcal{H}$ is an algorithm that on input $m \in \mathcal{I}$ returns $m' \neq m \in \mathcal{I}$ s.t. $\mathcal{H}(m') = \mathcal{H}(m)$.

— A *collision attack* on $\mathcal{H}$ is an algorithm that on input $\varnothing$ returns m, $m' \neq m \in \mathcal{I}$ s.t. $\mathcal{H}(m) = \mathcal{H}(m')$.

Q. 1:

1. Give an algorithm for a second preimage attack. What is its expected running time (in function of n) for a perfectly random function $\mathcal{H}$?

2. What is the average complexity of a collision attack for a perfectly random function $\mathcal{H}$?

3. Give the specifications of a hash function $\mathcal{H}' : \mathcal{I} \to \{0,1\}^n$ for which every pair of distinct messages forms a collision. Is it possible to efficiently find second preimages for this function?

We informally call a hash function $\mathcal{H}$ *preimage-resistant* (resp. *collision-resistant*) if there is no "efficient" (first or second) preimage attack (resp. collision attack) on $\mathcal{H}$.

[1] Be careful that the elements of $\mathcal{S}$ need not be integers. For instance $\mathcal{S}$ could be equal to $\{\textit{martes martes}, \textit{martes foina}, \textit{martes zibellina}\}$.

Q. 2:

1. Show that an adversary having a black box access to an efficient second preimage attack can perform a "similarly efficient" collision attack[2]. Is the converse true?

2. Is it possible for a hash function to be collision-resistant but not preimage-resistant?

3. Let $\mathcal{H}$ be such that the best collision attack on it is a generic attack. What can you say about the complexity of preimage attacks on $\mathcal{H}$?

Exercise 3: A compression function *(Exam '19)*

In the following questions, $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a compression function, and $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher with n-bit keys and blocks. We give the following definitions:

— A *first preimage attack with known chaining value* on F is an algorithm that on input $(h, t) \in \{0,1\}^n \times \{0,1\}^n$ returns $m \in \{0,1\}^n$ s.t. $F(h, m) = t$.

— A *key-recovery attack with one known plaintext* on E is an algorithm that on input $(m, c = E(k, m)) \in \{0,1\}^n \times \{0,1\}^n$ (where k is drawn uniformly at random in $\{0,1\}^n$) returns k' s.t. $E(k', m) = c$.

Q. 1:

1. Give an example of block cipher for which in the above key-recovery attacks, one always has $k' = k$. Give another example where with high probability $k' \neq k$.

We now make the simplifying assumption that E is "ideal" in the sense that any algorithm for a key-recovery attack with one plaintext has an expected running time of 2^n.

Q. 2: A designer proposes to build a compression function F from a block cipher E as $F(h, m) := E(h, m)$.

1. Give an efficient preimage attack on F.

Q. 3: Another designer proposes to build a compression function F from a block cipher E as $F(h, m) := E(m, h)$.

1. Let $A \Rightarrow B$ be a logical proposition. Give its contrapositive.

2. Explain why there is no efficient preimage attack on F if E is ideal in the above sense.

[2] If this statement were expressed formally, what we want would be a reduction whose time cost is (for instance) polynomial in the input size.