Problem sheet 2 for Course 01410, 2023

These practice problems have the purpose of helping you understand the material better and learning the skills that are necessary to analyze cryptographic constructions, and sometimes to prepare you for the next class. All answers should be supported by a written justification. To gauge whether a justification is sufficient, ask yourself if your peers would be convinced by it without additional explanations.

Exercise 2.1 Denote by e^{OTP} and d^{OTP} the encryption and decryption algorithms of the one-time pad for n-bit strings, i.e. for $k, m, c \in \{0,1\}^n = \mathbb{K}^{\mathrm{OTP}} = \mathbb{M}^{\mathrm{OTP}} = \mathbb{C}^{\mathrm{OTP}}$ we have $e_k(m) = k \oplus m$ and $d_k(c) = k \oplus c$. Decide whether the following modifications of the one-time pad are secure. For insecure schemes, describe an adversary that wins the IND-PASS game with probability larger than $\frac{1}{2}$. For secure schemes, argue why they are secure.

For both schemes, $\mathbb{M} = \mathbb{M}^{\mathrm{OTP}}$ and $\mathbb{K} = \mathbb{C} = \{0,1\}^{2n}$, and the keys and ciphertexts consists of two parts of n bits each, $k = k_1 \| k_2$ and $c = c_1 \| c_2$ with $k_1, k_2, c_1, c_2 \in \{0,1\}^n$. For two strings $x, y \in \{0,1\}^n$, we denote by $x \odot y$ the bit-wise AND (or bit-wise product).

1. $e^1_k(m) = e^{\mathrm{OTP}}_{k_1}(m) \| (m \odot k_2)$
   $d^1_k(c) = d^{\mathrm{OTP}}_{k_1}(c_1)$.
2. $e^2_k(m) = e^{\mathrm{OTP}}_{k_1}(m) \| e^{\mathrm{OTP}}_{k_2}(m)$
   $d^2_k(c) = d^{\mathrm{OTP}}_{k_1}(c_1)$.

Note: The schemes in this exercise are of no real-world relevance and are designed for practicing the use of security definitions.

Exercise 2.2 We use the notation from last week's problem sheet for ASCII characters. Let $\mathrm{ord}(c)$ be the index of an ASCII character c, i.e. $\mathrm{ord}(c)$ is defined such that $c = y_{\mathrm{ord}(c)}$. Define the one-time pad encryption scheme for ASCII strings as follows. For a message string m and a key string k (of equal length ℓ), the one-time pad encryption outputs $e_k(m) = c = (c_1, ..., c_\ell)$ such that $c_i = \mathrm{ord}(m_i) \oplus \mathrm{ord}(k_i)$, and $\oplus$ is the XOR operation when the integer values are represented in binary.

Example: For $\ell = 2$ we have the message $m =$`Hi` and the key string $k =$ `B`y_5 (the ASCII character with index 5 is not printable). Using a subscript 2 to indicate binary numbers, we have $\mathrm{ord}($`H`$) = 72 = 1001000_2$, $\mathrm{ord}($`i`$) = 105 = 1101001_2$, $\mathrm{ord}($`B`$) = 66 = 1000010_2$ and $ord(y_5) = 6 = 0000101_2$. We therefore get $e_k(m) = (10, 108)$.

The following are the ciphertexts of encrypting with the same key the ASCII strings `grape`, `apple`, `pears`, in a different order:

$c = (17, 46, 118, 127, 77)$, $c' = (0, 59, 103, 97, 91)$, $c'' = (23, 44, 103, 99, 77)$

In addition, $\tilde{c} = (0, 41, 104, 50, 9)$ is the encryption of an unknown ASCII string, using the same key as before. Break the OTP and decrypt $\tilde{c}$!