# DD2448 Foundations of cryptography

## Homework 1

**Manuel Osvaldo Olguin Muñoz**
921223-4679
molguin@dcc.uchile.cl

## krypto15

,
,

**1**

**1a** (5T) NOT SOLVED

**1b** (1T) NOT SOLVED

**2** (2T) NOT SOLVED

**3**

*Handed in through Kattis.* "Borrowed" ideas:

- Implementation of the S-Box and R-Con as static arrays, since they are input independent and only take 256 different values.
- Multiplication in $GF(2^8)$ done through bit arithmetic:
  - Multiplication by two: if the high bit of the input $X$ is 1, shift one bit to the left and $XOR$ with 00011011. Otherwise, only shift one bit left.
  - Multiplication by three: multiply by two and $XOR$ with the original input.

**4**

**4a** The information is sufficient to calculate the entropy of $Y$. Since its clear that $X_i, ..., X_n$ are mutually independent, one can conclude that $Y$ distributes uniformly over $(0, 1)^n$, and thus its entropy is easily calculated.

**4b** (1T) NOT SOLVED

**4c** (1T) NOT SOLVED

**4d** (2T) NOT SOLVED

**4e** (2T) NOT SOLVED

**4f** (2T) NOT SOLVED

**5** The Hill cipher is vulnerable to a known-plaintext attack, since the encryption is completly linear. For this attack, the adversary needs to have captured at least $m$ distinct plaintext-ciphertext pairs, where $m \times m$ is the size of the key used for the encryption (if $m$ is unknown to the attacker, he or she can continue trying with different values until the plaintext-ciphertext pairs agree - assuming $m$ is not too big). The time complexity of this algorithm is aproximately similar to that of matrix multiplication, which can be done using the Strassen algorithm in $O(n^{2.8074})$.

**6** NOT COMPLETE

First of all, given that these are all students with no previous knowledge of cryptography and none of them is a genious, I would assume all of the ciphers fall into one of the following categories:

- Substitution ciphers.
- Transposition ciphers.
- Very simple symmetric ciphers.

The program written to break these would then work in the following manner:

1. Feed two known, and different, plaintexts to the cipher, and wait for the outputs.
2. Analyze the outputs, comparing them to the inputs:
   (a) If input and output have exactly the same set of characters (excluding spaces), but in different order, mark as transposition cipher.
   (b) If input and output have different sets of characters, but the frequencies of characters are the same (for example, input has 3 A's, 2 B's, 5 E's and so on, and output has the same frequencies but with 3 W's, 2 R's, and so on), mark as substition cipher.

**7**

**7a** Basically, an uniform adversary is a probabilistic polynomial-time Turing Machine, which operates in an "uniform" way over all values of the security parameter of a scheme. On the other hand, a non-uniform adversary is a "family of circuits". For each value of the security parameter of a scheme, a different program in said family is executed - in a way, it adapts to the security parameter of the scheme.

**7b** Yes, it matters. Non-uniform adversaries are stronger than uniform ones, since the former can do everything the latter can, and more. Thus, security measures against non-uniform adversaries are always **stronger**, but often require stronger computational hardness assumptions.

**8**

**8a** Handed in on a separate sheet of paper.

**8b** For $t = 1$, we have

$$L_1 = R_0$$

$$R_1 = L_0 \oplus f_1(R_0)$$

Thus, the whole left half of the ciphertext ALWAYS corresponds to exactly the whole right half of the plaintext - an ocurrence which is extremely unlikely in any pseudorandom permutation.

**8c** For $t = 2$, we have

$$L_2 = R_1 = L_0 \oplus f_1(R_0)$$

$$R_2 = L_1 \oplus f_2(R_1) = R_0 \oplus f_2(L_0 \oplus f_1(R_0))$$

If we now take two plaintexts with the same right half, $L_0^a R$ and $L_0^b R$, note that the left sides of their respective ciphertexts are

$$L_2^a = L_0^a \oplus f_1(R)$$

$$L_2^b = L_0^b \oplus f_1(R)$$

And thus the following is always true,

$$L_2^a \oplus L_2^b = L_0^a \oplus L_0^b$$

which is very unlikely in a pseudo-random permutation.

**8d**

For $t = 3$, we have

$$L_3 = R_2 = R_0 \oplus f_2(L_0 \oplus f_1(R_0))$$
$$R_3 = L_2 \oplus f_3(R_2) = (L_0 \oplus f_1(R_0)) \oplus f_3(R_0 \oplus f_2(L_0 \oplus f_1(R_0)))$$

Now, if let's modify the left side of the message. We end up thus with $L_0 \oplus \xi || R_0$, and the ciphertext:

$$L'_3 = R_0 \oplus f_2(L_0 \oplus \xi \oplus f_1(R_0))$$
$$R'_3 = (L_0 \oplus \xi \oplus f_1(R_0)) \oplus f_3(R_0 \oplus f_2(L_0 \oplus \xi \oplus f_1(R_0)))$$

Working in the other direction, we define $L''_0 || R''_0$ such that its ciphertext has the same left side as $L'_3 || R'_3$ and its right side is the same right side $XOR$ our modification $\xi$;

$$L''_3 = L'_3$$
$$R''_3 = R'_3 \oplus \xi$$

Then, we can derive as follows:

$$L''_2 = R''_3 \oplus f_3(L''_3 = L'_3) = L_2$$
$$L''_1 = R''_2 \oplus f_2(L''_2 = L_2) = L''_3 \oplus f_2(L_2) = R''_0$$

Also

$$R_0 = L_3 \oplus f_2(L_2)$$

$$\Rightarrow R''_0 \oplus L''_3 = R_0 \oplus L_3 = f_2(L_2)$$

Which always hold! This couldn't be the case if this was a pseudorandom permutation.
$\therefore$ A 3-round Feistel network is not a pseudorandom permutation.