T-79.159 Cryptography and Data Security **Spring 2003**
Tutorial 2: Solutions
28.–29.1.2003

1. We are asked to develop a general encoding for permutations from 64-bit input values to 64-bit output values. The most trivial solution would be to encode the permutation as a table of 2^{64} 64-bit values. Such a table clearly requires $64 * 2^{64} = 2^{70}$ bits of storage.

   However, a slight improvement is possible. Clearly the first value of the permutation must be chosen from 2^{64} possibilities. We remove this value from the index, and hence there are $2^{64} - 1, 2^{64} - 2, \cdots$ possibilities for the second, third, etc value of the permutation. The last value requires no space at all, since all other values have already been used.

   We may use a "mixed radix" encoding, where encode the permutation as an integer x, $0 \leq x \leq 2^{64}!$. We will describe only the decoding procedure. To get the first value, check $y_0 = x \bmod 2^{64}$, and set $x \leftarrow \frac{x-y_0}{2^{64}}$. Now remove y_0 from the set of possible values, and get the next one as $y_1 = x \bmod 2^{64} - 1$ and set $x \leftarrow \frac{x-y_1}{2^{64}-1}$. Remove y_1 from the permutation, and repeat. This way the permutation can be encoded in $\lg 2^{64}! \approx 0.977 * 2^{70}$ bits.

2. The device should have an internal clock, which produces a never-repeating minute number t. The device also contains the secret key k. k should be of sufficient length (say 80 bits), and would preferably be stored in a tamper-resistant way.

   We may use a cryptographic hash function H to produce the access number $n = H(k \mid t)$. One good selection of H is SHA-1.

   When checking the access number, the computer system may search in a small range around current t for a match. A PIN code (to be remembered by the user) may be embedded into the system as well for additional security.

3. The probability that a key K satisfies $DES(K,P) = C$ for given P and C is clearly 2^{-64}. Since there are 2^{56} keys, the probability of even a single key satisfying the condition is approximately 0.0038986, a number close to 2^{-8}.

   This illustrates that most ciphertext/plaintext pairs are in fact impossible when the key size is shorter than the block size. On the other hand, when key is longer than the block, multiple blocks are required to uniquely identify the correct key in exhaustive search.

4. Two-round Feistel cipher works as follows. We split the n-bit plaintext block into two halves $P = (L_0, R_0)$. Now we set $L_1 = L_0 \oplus f_1(R_0)$ and $R_1 = R_0 \oplus f_2(L_1)$, where f_1 and f_2 are unknown $n/2$-bit permutations ("round functions"). The ciphertext block is $C = (L_1, R_1)$.

   We encrypt a pair of plaintexts $P = (L_0, R_0)$ and $P' = (L_0 \oplus \Delta, R_0)$, where Δ is some "differential". We see that in the first round the differential is maintained: $L_1' = L_1 \oplus \Delta$. In the second round the differential stays in the left halve, but the right side is changed into an unknown value. Therefore we may easily distinguish the Feistel cipher by observing that the ciphertext blocks are related by $C \oplus C' = (\Delta, x)$, for some unknown x.

5. Stream ciphers encrypt data by XOR'ing each key-dependant keystream bit $z(t)$ with the plaintext bit $P(t)$ to produce a ciphertext bit $C(t)$.

   $$C(t) = P(t) \oplus z(t)$$

   Decryption is equivalent to encryption:

   $$P(t) = C(t) \oplus z(t)$$

   If the same keystream $z(t)$ is used to encrypt two plaintexts $P(t)$ and $P'(t)$, we see that the corresponding ciphertexts are related by $C(t) \oplus C'(t) = (P(t) \oplus z(t)) \oplus (P'(t) \oplus z(t)) = P(t) \oplus P'(t)$. In other words, the keystream $z(t)$ is canceled out and we get the XOR of the plaintexts! This greatly helps cryptanalysis (e.g. we can in many cases directly "see" the plaintext in one stream if the other stream has a run of zeroes).

6. Assume that the "XOR checksum" word is 32 bits long. It is easy to see that flipping bit n in the message will change bit $n \bmod 32$ in the checksum. This can be repeated any number of times.

   This may be a significant problem if the attacker has information about the data being encrypted (but not necessarily the data itself). For example, if the attacker knows that a certain position in the data stream contains the number of Euros being transferred, he may modify that amount.