University of St. Gallen - Cyber Security
Fall semester 2023
Prof. Katerina Mitrokotsa, Dr. Novak Kaluderovic

Assignment 1

Yasmin Lützelschwab

Exercise 1. (8 points) Consider a linear block cipher E that takes as input plaintext of size 256 bits and generates 256 bits of ciphertext. Let $E_k(M)$ denote the encryption of a 256-bit message M under a key k (length of the key is irrelevant). The linearity property of the cipher is as follows:

$$E_k(M_1 \oplus M_2) = E_k(M_1) \oplus E_k(M_2) \text{ for all 256-bit plaintexts } M_1 \text{ and } M_2.$$

There exists a black-box that will provide the corresponding plaintext when a ciphertext query is submitted. The adversary can have 256 such queries, *i.e.* adversary can choose 256 ciphertexts and get the corresponding plaintexts. Would it be possible for the adversary to decrypt any ciphertext without the knowledge of the key? If yes, explain the process. If no, provide a proof.

Given the linearity property of the cipher for plaintexts M_1 and M_2 is as follows: $E_k(M_1 \oplus M_2) = E_k(M_1) \oplus E_k(M_2)$

Given access to a black-box, we can make 256 queries to obtain the corresponding plaintext for a given ciphertext.

Given linear block cipher *(E,D):*
E (K,m) is invertible and it holds: $D(K, c) = E^{-1}(K, m)$

$k =$	Key	$E:$	$\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
$M =$	Message	$K \in K =$	$\{0,1\}^k$
$C =$	Cipher Text	$m \in M =$	$\{0,1\}^n$
$E_k(M):$	$\rightarrow M \times k\ C$		

Question: Would it be possible for the adversary to decrypt any ciphertext without knowing the key?

Answer: Yes! By exploiting the linearity property of the block cipher to decrypt a ciphertext $E_k(M) = C \in \{0,1\}^{256}$. It is crucial to recognize that each cipher $C \in \{0,1\}^{256}$ can be expressed as a linear combination of the binary space. Standard basis as a set of vectors:

$$B = \{e_i \in \{0,1\}^{256} : i \in \{1, 2, ..., 256\}\}$$

At position *i* has the value '1' and '0' in other places. To receive the encoded Ciphertext $E_k(e_i)$ for each element, the black box has to loop 256 times through the basis vector. Which will look like:

$$E_k(B) = \{ E_k(e_i): i \in \{1, 2, ..., 256\}\}$$

Now we can look for the linear combination of encrypted basis vectors represented by $C = (x_1, x_2, \ldots, x_{256})$:

$$C = x_1E_k(e_1) \oplus x_2E_k(e_2) \oplus ... \oplus x_{256}E_k(e_{256})$$

'x_i' is the coefficient of $E_k(e_i)$ which means the encrypted basis vector $E_k(e_i)$ can be removed from $x_i = 0$ and kept when $x_i = 1$. Out of that the linear propery for the plaintext M is:

$$M = x_1e_1 \oplus x_2e_2 \oplus ... x_{256}e_{256}$$

Exercise 2. (6 points (3+3)) Nowadays computers and processors work at a frequency of around 3 GHz, meaning that they can execute $3 \cdot 10^9$ operations per second. Suppose you have access to a supercomputer which has 100000 processors, each working at 3 GHz. Suppose further that each of these cores can compute a single evaluation of DES in 50 operations. You are given $C = \mathrm{DES}_K(00\ldots0)$ which is the encryption of the string $\underbrace{00\ldots0}_{64 \text{ times}}$ with DES under a random key K.

1. Describe a brute-force strategy for obtaining K. How much time would you need to do it?
2. What if AES with a 128 bit key was used instead, assuming identical evaluation cost?

Given supercomputer with 100'000 processors, each operating at a frequency of 3 Ghz, which means they can execute 3 x 10^9 operations per second.

Given each processor can compute a single evaluation of the DES encryption algorithm in 50 operations.

Given you are provided with a cipher $C = DES_k(M)$, which is the result of encrypting a 64-bit Message string of zeros using DES with an unknown random key *K*.

1. Brute-Force strategy for DES

= DES uses a 56-bit key. A Brute-Force attack on DES involves trying every possible key 2^{56} until the correct one is found.

For the brute-force strategy, every possible key *k* would be needed to try to get the correct decryption for $D_k(E_k(M)) = M$. Since there are 2^{56} keys, divide this number by the total number of evaluations per second to get the total time required for a Brute-Force attack.

DES can be evaluated *n* times per processor:

$$\mathrm{n} = \frac{3\times10^9}{50} = 6 \times 10^6$$

DES can be evaluated *n* times per second:

$$\mathrm{n} = 100000 \times \frac{3\times10^9}{50} = 6 \times 10^{12}$$

Now we add the 2^{56} possibilities:

$$t = \frac{2^{56}}{6 \times 10^{12}} = \frac{72{,}057{,}594{,}037{,}927{,}936}{6 \times 10^{12}} \approx 12,\ 009,\ 599s \approx 200,\ 160min \approx 3,\ 336h \approx 139\ days$$

In other words, the Brute-Force strategy would need 139 days to find the right key *k* for DES given the 100'000 core super computer.

2. Brute-Force strategy for AES

= AES uses a 128-bit key. A Brute-Force attack on DES involves trying every possible key 2^{128} until the correct one is found.

Since the needed operations for each AES remains the same as for DES, the total number of AES evaluations per second is:

$$\mathrm{n} = 100000 \times \frac{3\times10^9}{50} = 6 \times 10^{12}$$

AES has a 128-bit key which will be added in the calculation as followed:

$$t = \frac{2^{128}}{6 \times 10^{12}} = \frac{3.4 \times 10^{38}}{6 \times 10^{12}} = 5,\ 671372782 \times 10^{25}\mathrm{s} \approx 6,\ 56 \times 10^{20}\ days \approx 1,\ 79 \times 10^{18}\ years$$

In other words, the Brute-Force strategy would need $1,\ 79 \times 10^{18}$ years to find the right key *k* for AES given the 100'000 core super computer.

Exercise 3. (8 points (3+3+2) In this exercise we analyse a new mode of operation. Assume that we have a block cipher E, a key k, and we wish to encrypt a message M divided into blocks $M = (M_1, M_2, \ldots, M_l)$. In this mode of operation we do that by initialising a counter T to IV, and computing

$$C_i = T_i \oplus E_k(M_i)$$

for each $i = 1, \ldots, l$, where $T_i = IV + i$. The ciphertext is

$$C = (IV, C_1, C_2, \ldots, C_l).$$

1. Is this mode of operation equivalent to another one? If yes, prove it. If no, explain it.
2. Does the IV need to be unique?
3. Does this mode of operation suffer from some security problems?

Given a message M with blocks $M = (M_1, M_2, \ldots, M_l)$

Given a block cipher (E, D) with the key k.

Given initialising a counter T to IV, and computing $C_i = T_i \oplus E_k(M_i)$.

1. Is this mode of operation equivalent to another one?

For each block i of the message M, it computes $C_i = T_i \oplus E_k(M_i)$, *where* $T_i = IV + l$ *and* E_k is the encryption function under key k. This mode describes a variant of CTR (deterministic counter mode) as follows:

$$C_i = E_k(T_i) \oplus M_i$$

$$T_i = IV + i$$

$$i \in \{0, ..., l\}$$

$$C = (IV \,|C_0|C_1|...|C_l)$$

The difference is that in standard CTR mode, the nonce N_i is incremented for each block, while in the described mode, the counter T_i is directly derived from the initial vector IV and the block index i.

2. Does the IV need to be unique?

YES! The Initial Vector (IV) needs to be unique for each encryption session inside the mode. As well as in the CTR mode of operation. It is crucial for several reasons:

- To prevent reuse of key-streams
- The security against certain types of attacks
- To ensure unpredictability

3. Does this mode of operation suffer from some security problems?

Yes, as already mentioned using the same IV for the encryption of messages $M_1 = M_2$ would produce the same ciphertext $C_1 = C_2$, which has the following certain security considerations:

- IV Uniqueness and non-repetition
- Integrity and Authentication
- Predictability of IV
- Side-Channel attacks
- Key management

This is the reason behind the XOR used directly on the counter T_i, for the plaintext M_i before encrypting it.

Exercise 4. (8 points (1+1+1+5)) A block cipher on messages of 64 bits acts as a permutation. It takes as input an element of $\{0,1\}^{64}$ and outputs an element of $\{0,1\}^{64}$. In addition to that, it is invertible, and therefore a permutation.

1. How many different permutations are there from $\{0,1\}^{64}$ to $\{0,1\}^{64}$?
2. How many different functions are there from $\{0,1\}^{64}$ to $\{0,1\}^{64}$.
3. What is the probability that a random function from $\{0,1\}^{64}$ to $\{0,1\}^{64}$ is a permutation? Is it large or small? Explain.
4. Let K be a random DES key, M a message of 64 bits and $C = \mathrm{DES}_K(M)$ the corresponding ciphertext. Suppose an adversary is given (M, C) and they do an exhaustive key search attack to locate the first key K' such that $\mathrm{DES}_{K'}(M) = C$. Estimate the probability that $K' = K$? Explain your reasoning.

Given a block cipher to go from an input $M \in \{0, 1\}^{64}$ to an output $C \in \{0, 1\}^{64}$.

1. **How many different permutations are there from $\{0, 1\}^{64}$ to $\{0, 1\}^{64}$?**

Since there are 2^{64} different 64-bit strings, each permutation will map each element of message M to a unique element ciphertext C in a one-to-one fashion. The Total of $n_{permutations}$ is:

$$n_{permutations} = (2^{64}) \times (2^{64} - 1) \times (2^{64} - 2) \times \ldots \times 2 \times 1 = (2^{64})!$$

2. **How many different functions are there from $\{0, 1\}^{64}$ to $\{0, 1\}^{64}$?**

The input $M_i \in \{0, 1\}^{64}$ can be mapped to any output of $C_i \in \{0, 1\}^{64}$ and for that reason the total of $n_{functions}$ is:

$$n_{functions} = (2^{64})^{(2^{64})}$$

3. **What is the probability that a random function from $\{0, 1\}^{64}$ to $\{0, 1\}^{64}$ is a permutation? (large/small)**

The probability P that a random function is a permutation is:

$$P = \frac{n_{permutations}}{n_{functions}} = \frac{(2^{64})!}{(2^{64})^{(2^{64})}}$$

Considering the nature of the factorial function vs. the exponential function, P is extremely small, almost negligible.

4. **Estimate the probability that $K' = K$?**

The probability that $K' = K$ in an exhaustive key search attack
>> where K' is the first key found such that $DES_{K'}(M) = C$

The probability that the first key K' found is actually the key K used to encrypt the message is:

$$P(K' = K) = \frac{1}{2^{56}}$$