ECMM462: Fundamentals of Security

Continuous Assessment (v21.3)

Diego Marmsoler

Wednesday 17th November 2021

The CA is worth 40% of your final mark and intended to last for 40 hours. You can get up to 100 marks in total split into four exercises:

Topic	Marks	Tasks	Target Time	Target date
Symmetric Encryption	25	4	10 h	October 13
Asymmetric Encryption	25	4	10 h	October 27
Protocol Verification	25	4	10 h	November 10
Access Control	25	4	10 h	November 17

Format of Submission Please prepare a folder for every exercise (E1-E4) and a file for every task (t1-t4). For example, the solution for the first task of the second exercise should be in `E2/t1.txt`. Then, submit your solutions via the electronic submission system BART (`https://bart.exeter.ac.uk/`) by Wednesday 17th November 2021 noon GMT.

Python Libraries For the exercises requiring you to implement something in Python you can use only basic Python libraries (in particular you must not use any encryption libraries).

Questions If you have any questions regarding the assessment brief please post them to the corresponding topic in the discussion forum: `https://vle.exeter.ac.uk/mod/forum/discuss.php?d=206680`. If you don't want to disclose your identity then the forum has the option to post anonymously.

1 Symmetric Encryption

In the lecture we discussed the Data Encryption Standard (DES) as an example of a modern symmetric encryption cipher. In the following, we briefly introduce a simplified version of the DES.

1.1 Key Generation

In simplified DES, one master key is used to generate multiple sub-keys (so called round keys). Figure 1 depicts the algorithm to generate round keys in our simplified version of DES: It takes a 10-bit master key as input and produces two 8-bit round keys using permutations and shifts.

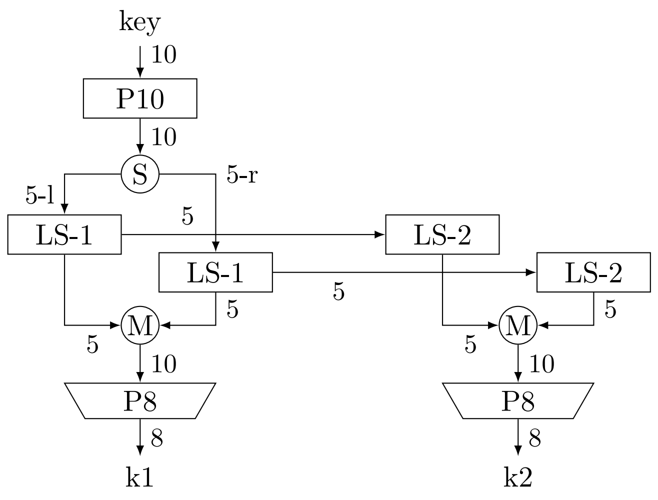


Figure 1: Round Key Generation

The operator S splits a bit sequence of length 10 into two bit sequences of length 5: 5-l (for left) denote the first 5 bits and 5-r (for right) denote the second 5 bits. The operator M concatenates two bit sequences of length 5 to produce a bit sequence of length 10 (again, the left input denotes the first 5 bits and the right input the second 5 bits). The functioning of the permutation tables P10 and P8 as well as the shifts LS-1 and LS-2 are described in Tab. 1. The table shows the position of each input bit in the output bit sequence after applying the corresponding permutation. Using P10, for example, the 3^{th} input bit becomes the 1^{st} output bit, the 5^{th} input bit becomes the 2^{nd} output bit, etc.

1.2 Encryption and Decryption

Encryption in our simplified version of DES is described by the Feistel structure depicted in Fig. 2. The internal functioning of the f function is shown in Fig. 3. The permutation tables IP, IP^{-1}, E/P, and P4 are described in Tab. 2.

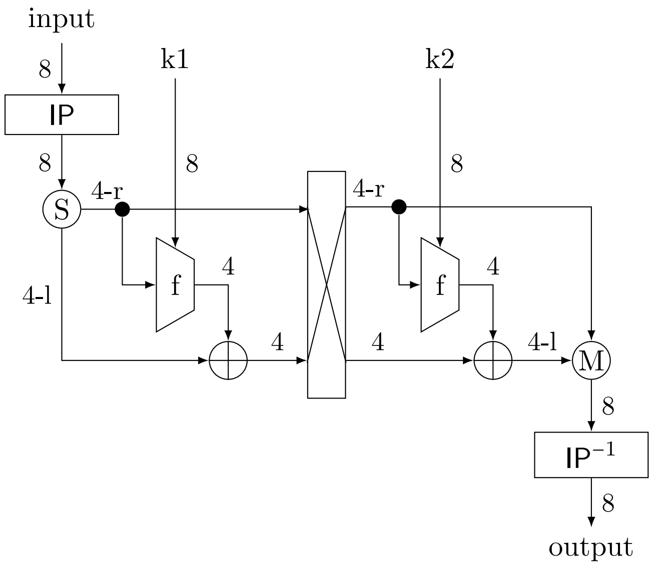


Figure 2: Encryption

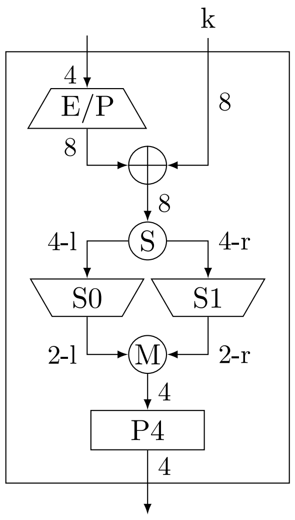


Figure 3: Function f

The substitution tables S0 and S1 are depicted in Tab. 3 and Tab. 4. The tables get 4 bits as input: the first and the last bit specify the row, and the second and the third specify the column of the corresponding output. For example, 0101 denotes the row 01 (which corresponds to 1 in decimal notation) and the column 10 (which corresponds to 2 in decimal notation). Thus, the corresponding output for table S0, for example, can be found in row 1 and column 2 of the table, which is 1 (which corresponds to 01 in binary notation).

Decryption in our simplified version of DES is similar to encryption except that we swap the round keys k1 and k2.

1.3 Tasks

In the following you will encrypt and decrypt the following text using our simplified version of DES:

EXETER

Table 1: Permutation Tables.

	1	2	3	4	5	6	7	8	9	10
P10	3	5	2	7	4	10	1	9	8	6
LS-1	2	3	4	5	1	-	-	-	-	-
LS-2	3	4	5	1	2	-	-	-	-	-
P8	6	3	7	4	8	5	10	9	-	-

To this end you should use the following key:

0110100111

T1.1 Compute the round keys (including intermediate results).

T1.2 Convert the above text to bit representation in ASCII and encrypt the first letter using the algorithm (including intermediate results).

T1.3 Decrypt the first letter using the algorithm (including intermediate results).

T1.4 Implement our simplified version of DES in Python. The program should be called `myDes` and take three input parameters:

- the first parameter is either `enc` or `dec` to encrypt or decrypt.
- the second parameter is a string representing the 10-bit key.
- the third parameter is a string representing the 8-bit input.

The program should return only the 8-bit result. For example:

```
>myDes enc "1010101010" "00110011"
>01010101
```

Table 2: Permutation Tables.

	1	2	3	4	5	6	7	8
IP	2	6	3	1	4	8	5	7
IP^{-1}	4	1	3	5	7	2	8	6
E/P	4	1	2	3	2	3	4	1
P4	2	4	3	1	-	-	-	-

Table 3: S_0

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

Table 4: S_1,

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

2 Asymmetric Encryption

In the following we describe a simple description of an asymmetric encryption mechanism. The idea is to represent encryption and decryption with table lookups. To this end, three types of tables are used:

M1 is just a sequence of N elements and contains a random permutation of all integers between 1 and N. For example $m1 = (4, 3, 2, 5, 1)$ could be an example for $N = 5$. It is used to construct a public key from a private key. For example, if we assume that our private key is 2, then the corresponding public key, according to our example table $m1$, is given by $m1(2) = 3$.

M2 is an $N \times N$ matrix such that each row represents a random permutation of all integers between 1 and N. For example, for $N = 5$ we may have:

$$m2 = \begin{matrix} 3 & 1 & 2 & 5 & 4 \\ 1 & 4 & 3 & 2 & 5 \\ 4 & 5 & 2 & 3 & 1 \\ 3 & 2 & 1 & 4 & 5 \\ 2 & 3 & 5 & 1 & 4 \end{matrix}$$

It is used for encryption. For example, to encrypt 3 using our table $m2$ and key 2, we get $m2(2, 3) = 3$.

M3 is an $N \times N$ matrix such that each column represents a random permutation of all integers between 1 and N. It is used for decryption.

The tables must be constructed in a way such that for all k and p, with $1 \leq k, p \leq N$ the following property holds:

$$M3(M2(M1(k), p), k) = p \tag{1}$$

2.1 Tasks

T2.1 Construct an example of M1, M2, and M3 for $N = 5$. Hint: first, randomly create M1 and M2 and then construct M3 such that property Eq. (1) holds.

T2.2 Encrypt the number 3 and then decrypt it again.

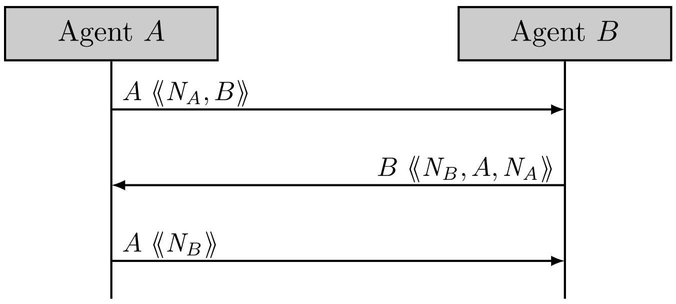
T2.3 Is the scheme secure? Explain why/why not.

T2.4 Implement the key generation scheme in Python. It should be called `myPKE` and take one input parameter which represents N. It should then generate three random tables $m1$, $m2$, and $m3$ which satisfy Eq. 1. For example:

```
>myPKE 5
>m1:
>4,3,2,5,1
>m2:
>3,1,2,5,4
>1,4,3,2,5
>4,5,2,3,1
>3,2,1,4,5
>2,3,5,1,4
>m3:
>........
>........
>........
>........
>........
```

3 Protocol Verification

Consider the following protocol where $X \langle\!\langle M \rangle\!\rangle$ denotes a message M digitally signed by agent X:


Agent A
Agent B
$A \langle\!\langle N_A, B \rangle\!\rangle$
$B \langle\!\langle N_B, A, N_A \rangle\!\rangle$
$A \langle\!\langle N_B \rangle\!\rangle$

The aim of the protocol is to establish authentication between two agents. In particular, at the end of the protocol, agent B needs to be sure that nonce N_A was indeed send by agent A and also not replayed by someone else.

3.1 Tasks

T3.1 Formalize the protocol in OFMC.

T3.2 State the security property required and formalize it as a goal in OFMC.

T3.3 The protocol does not meet its goal. Provide a counterexample.

T3.4 One simple fix is to encrypt all the messages. Provide an alternative fix of the protocol and verify in OFMC that the property now holds.

4 Access Control

Assume you are developing an access control policy for a university according to the Bell-LaPadula model. To this end, lecturers are assigned security clearance “high” for the modules they teach and students clearance “low” for the modules they take. Moreover, exams are classified as “high” and homeworks as well as assignments as “low” for the corresponding modules.

4.1 Tasks

T4.1 Define a starting state $z_0 = (b_0, m_0, f_0)$ in which the following holds:

- *Alice* is a lecturer for module *Security*. *Bob* is a student of *Security* and *Eve* a student of *Logics*.
- *Ex1* is an exam for module *Logics*. *Hw1* is a homework for *Security* and *A1* an assignment for *Logics*.
- *Alice* has given edit (read/write) rights for *Ex1*, read rights for *A1*, and write rights for *Hw1*. *Bob* has read/write rights for *Hw1* and *Eve* for *A1*.
- Currently *Bob* is editing (reading and writing) *Hw1* whereas *Alice* is reading *A1*.
- The current security level of all subjects to an object is initialized with their maximum security level for this object.

T4.2 Argue whether or not the state described above is secure.

T4.3 Describe the new state arising when *Bob* stops writing to *Hw1* and *Alice* changes the exam (i.e., executes read/write rights on the exam), and use the security theorem to argue whether or not the new state is secure.

T4.4 Assume *Alice* wants to comment on *Bob*’s homework *Hw1*, i.e., execute write rights on it. (i) Explain how this is possible, (ii) define the corresponding protection state $z_1 = (b_1, m_1, f_1)$, such that it fulfils the security conditions, and (iii) show that it is secure using the security theorem.