ELL305 Computer Architecture
Assignment 2
Implementation of Cryptographic Ciphers

September 11, 2019

In this assignment you have to implement two cryptographic ciphers - PRESENT and ESF using LogiSim. You have to design a **hardware architecture** for both the ciphers.

- Download the assignment and extract it.
- The folder "assignment2" contains three files: `mem_img.txt`, `PRESENT_test.txt` and `ESF_test.txt`. `mem_img.txt` shows a sample input file. Rest of the two files cointains test vectors for the verification of your circuits.

1 Description

Cryptography is the science of secret writing. In cryptography, unencrypted data (plaintext) is encrypted into ciphertext using a secret key, which might subsequently (usually) be decrypted back into usable plaintext. A cryptographic cipher is an algorithm used to perform this encryption and decryption. In this assignment, you will be working with data encryption only.

1.1 PRESENT Cipher

The PRESENT cipher [1] is an ISO/IEC standarized lightweight block cipher. It accepts input plaintext of 64 bits and supports keys that are either 80 bits or 128 bits wide. In this assignment, you will be working with key size of 80-bits. PRESENT consists of three basic operations: *AddRoundKey*, *Substitution* and *Permutation*. These operations are repeated 31 times to obtain the final encrypted output. Fig.1 shows the block diagram of the PRESENT cipher.

Substitution-Box: S-boxes are used to substitute bits with other patterns (known as confusion). PRESENT uses 4X4 S-boxes. The current state of the S-box layer, $b_{63}.....b_0$ is considered as sixteen 4-bit words(nibble)

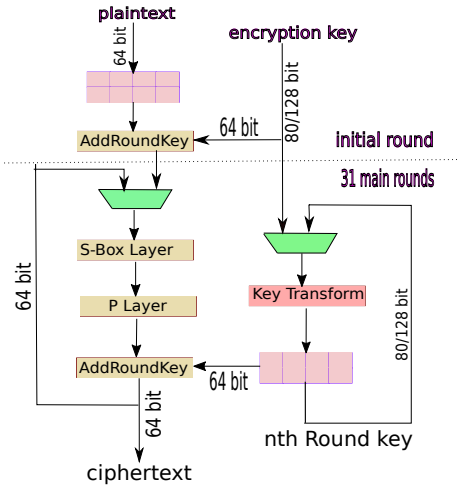


Figure 1: PRESENT Block Cipher

Table 1: PRESENT S-box input to output mapping (in hexadecimal) [1]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

$w_{15}.....w_0$ where,
$w_i = b_{4i+3}||b_{4i+2}||b_{4i+1}||b_{4i}$ for $0 \leqslant i \leqslant 15$. Here $||$ is the concatenation operation. The output nibble $S[w_i]$ provides the updated state values. There are many design techniques available in the literature for designing of the S-box layer.

Permutation-Box : The permutation-box permutates(diffuses) the bits. The operation of the P-Box is explained as follows:

P: $(0,1)^{64} \rightarrow (0,1)^{64}$
Here, the 64 input bits to the P-box (c) are mapped to a permutated set of 64 output bits (d).
For $0 \leqslant i \leqslant 15$, do :
$c_{4i}||c_{4i+1}||c_{4i+2}||c_{4i+3} \rightarrow d_i||d_{i+16}||d_{i+32}||d_{i+48}$

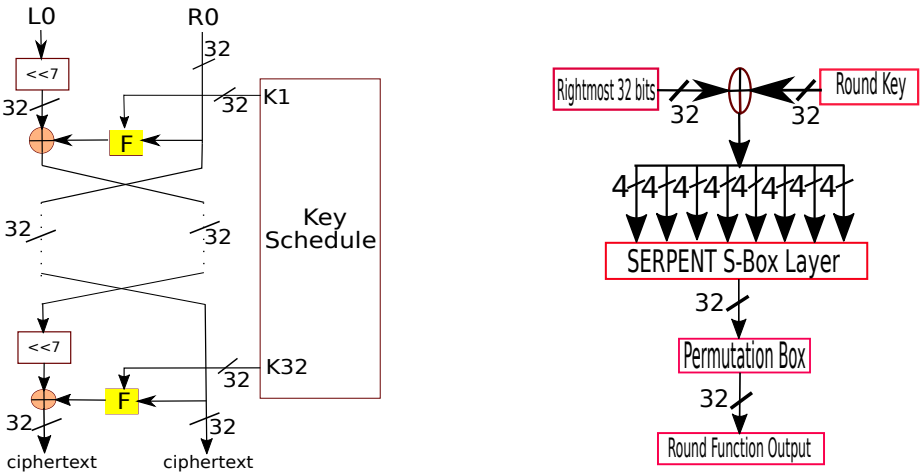
Key Schedule: PRESENT uses different keys for each round of encryption. These keys are known as 'round keys'. These 64-bit round keys are generated according to a key schedule mechanism. The user-supplied key is stored in a 80 bit key register K. For any round, the round key consists of the leftmost 64 bits of current content of the key register. Thereafter, key register K= $k_{79}k_{78}.....k_1k_0$ is updated according to the following equations:

1. $[k_{79}k_{78}.....k_1k_0] = [k_{18}k_{17}.....k_0k_{63}.....k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = \text{S}\ [k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \text{RoundCounter}$

Here, 'S' represents S-box and RoundCounter is the output value of the counter which is used to count the number of rounds of encryption.

1.2 Eight-Sided Fortress (ESF) Cipher

A feistel network first divides a data block into two equal subblocks and then encryption process is performed in multiple rounds. ESF [3] is a lightweight feistel cipher. It accepts input plaintext of 64 bits. It uses a 80-bit key.



(a) Block Diagram (b) Round Function 'F'

Figure 2: ESF Cipher

Encryption Process : The basic operation of the ESF cipher is shown in Fig.2(a). The encryption process is explained in the following steps:
1. The 64-bit input plaintext is divided into two 32-bit data block : R_0 and L_0.
2. The leftmost 32-bit feild L_0 is left sifted by 7 bits while the rightmost 32-bit R_0 enters the round function 'F'. The details of the round function are explained next.
3. The XOR operation is performed between $L_0 << 7$ and the output of F.
4. In the end, we swap the leftmost and rightmost chunks of 32 bits.

These steps are repeated for all the rounds of encryption. It should be noted that in the last round no swapping of the data is done. Finally the 64-bit ciphertext is obtained.

Table 2: ESF S-box input to output mapping [4]

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S0[x]	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12
S1[x]	15	12	2	7	9	0	5	10	1	11	14	8	6	13	3	4
S2[x]	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2
S3[x]	0	15	11	8	12	9	6	3	13	1	2	4	10	7	5	14
S4[x]	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
S5[x]	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
S6[x]	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
S7[x]	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

Round Function : ESF uses a substitution-permutation network for computing the round function 'F'. The rightmost 32 bits of the intermediate data are XORed with the round key. This is followed by the substitution and permutation operations respectively. This is shown in Fig.2(b).

Substitution-Box : ESF uses eight different 4X4 SERPENT S-boxes to perform substitution operations. You can explore about them from the references.

Permutation-Box : A P-box performs bit-wise permutation. The input and output are of 32 bits each. The design principle of the P-Box is the same as that of the PRESENT cipher. However, it should be noted that PRESENT uses a 64-bit P-Box while ESF uses a 32-bit P-box.

Key Schedule of ESF Cipher: The user-supplied key is stored in the 80-bit key register K. The 32-bit round key is extracted as the leftmost 32 bits of the current content of the key register. Thereafter, the key register K=$[k_{79}k_{78}.....k_1k_0]$ is updated according to the following steps:

1. K $<<$ 13
2. $[k_{79}k_{78}k_{77}k_{76}]= S_0\ [k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{75}k_{74}k_{73}k_{72}] = S_0\ [k_{75}k_{74}k_{73}k_{72}]$; $[k_{47}k_{46}k_{45}k_{44}k_{43}] = [k_{47}k_{46}k_{45}k_{44}k_{43}] \oplus$ RoundCounter ;

Here, S_0 represents S-box and RoundCounter is the output value of the counter which is used to count the number of rounds of encryption.

Implementation

Your task is to design an **iterative architecture** of the PRESENT and ESF block ciphers using LogiSim [2]. Make sure your designs have the following features:

- You have to ensure that your design works from the command line. Logisim can be run from the command line using:
  `java -jar logisim-XX <pathtocircuit>.cir`
  Logisim also provides a RAM module that can be used to load data from the command line. The syntax for this is as follows:
  `java -jar logisim-XX <pathtocircuit>.cir -tty table -load mem_img.txt`
- You can download and install LogiSim version 2.7.
- Your circuits should work for 64-bit plaintext inputs and, 80-bit key.
- The design should be iterative. The same hardware circuit should work for multiple clock cycles to complete all the rounds of encryption.
- Your design should include 18 S-boxes for PRESENT and 10 S-boxes for the ESF cipher.
- The data path of the architecture can be divided into multiple processing lines of 32-bits (or less). Make sure to synchronize your design well.
- You can check the correctness of your implementation with the set of test vectors provided in the files: `PRESENT_test.txt` and `ESF_test.txt`.

Input and Output File Format

- We will follow the description below to provide inputs to our circuit: The `mem_img.txt` will be used to load data into a RAM with a 32-bit data line and a 3-bit address line.
- In the `mem_img.txt` file, the first address will be used to store the lowest 32 bits of the plaintext. Address 1 will contain the MSB bits (32:63) of the plaintext. Address 2 will contain the lowest 32 bits (0:31) of the key. Address 3 will contain the next 32 bits of the key. The lowest 16 bits of Address 4 will contain the most significant 16 bits of the key. Fig.3 shows the contents of the RAM.
- The output pins should be ordered such that the 32 LSB bits are printed first, and the 32 MSB bits are printed next, when using the command line. You also have to ensure that only the final output is printed and not the intermediate results.
- To enable automated testing, you have to ensure that your design gets its input data in exactly the same format. Also you cannot use the RAM anywhere else in your design.

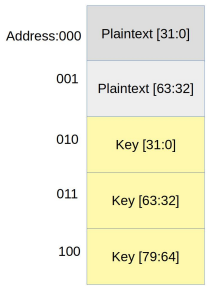


Figure 3: Memory format

2 Submitting Your Assignment

- You have to submit your design files (PRESENT.circ and ESF.circ) along with a one-page report (pdf).
- Your report should include basic analysis and a justification of the design decisions. Do not include already known details.
- We will be evaluating your submission with a different set of input values. Make sure your implementation works for all possible cases.
- Create a folder named as <EntryNumber> which contains the three assembly files. Compress the folder and submit only the <EntryNumber>.zip archive on Moodle. For example if your entry number is 2017EE51010, then submit as 2017EE51010.zip. No other format will be accepted.

  To summarize, you have to do the following steps:

  1. Create a folder named <EntryNumber>.
  2. Copy your code files in this folder.
  3. Create the .zip file
     - *For Windows*: Select the files, right-click and point to "Send to" and then select Compressed (zipped) folder. Name the folder as <EntryNumber>.zip
     - *For Linux:* Select the files, right-click and point to "Compress" and then create the zipped folder with the name as <EntryNumber>.zip
  4. Submit this zip file on Moodle.

- The deadline is **October 13, 2019, 23:59 hours**.

3 Grading Scheme

1. Marks division: Total = 50 marks
   PRESENT: 25 marks
   ESF: 25 marks

2. Late policy: 20% penalty per day (rounded to the next day, i.e. 3 hours late implies 1 day late).

3. **Plagiarism**: If any similarity to any other source is found, you will get a zero.

4 References

1. `http://lightweightcrypto.org/present/present_ches2007.pdf`

2. `http://www.cburch.com/logisim/docs/2.7/en/html/guide/index.html`

3. `http://privateweb.iitd.ernet.in/~eez198358/esf.pdf`

4. `https://eprint.iacr.org/2009/038.pdf`