

California State University, Northridge

Department of Electrical & Computer Engineering



ECE 526L

Final Project Report
(AES Cryptography)

By

Avinash Damse

CSUN ID- 203131064

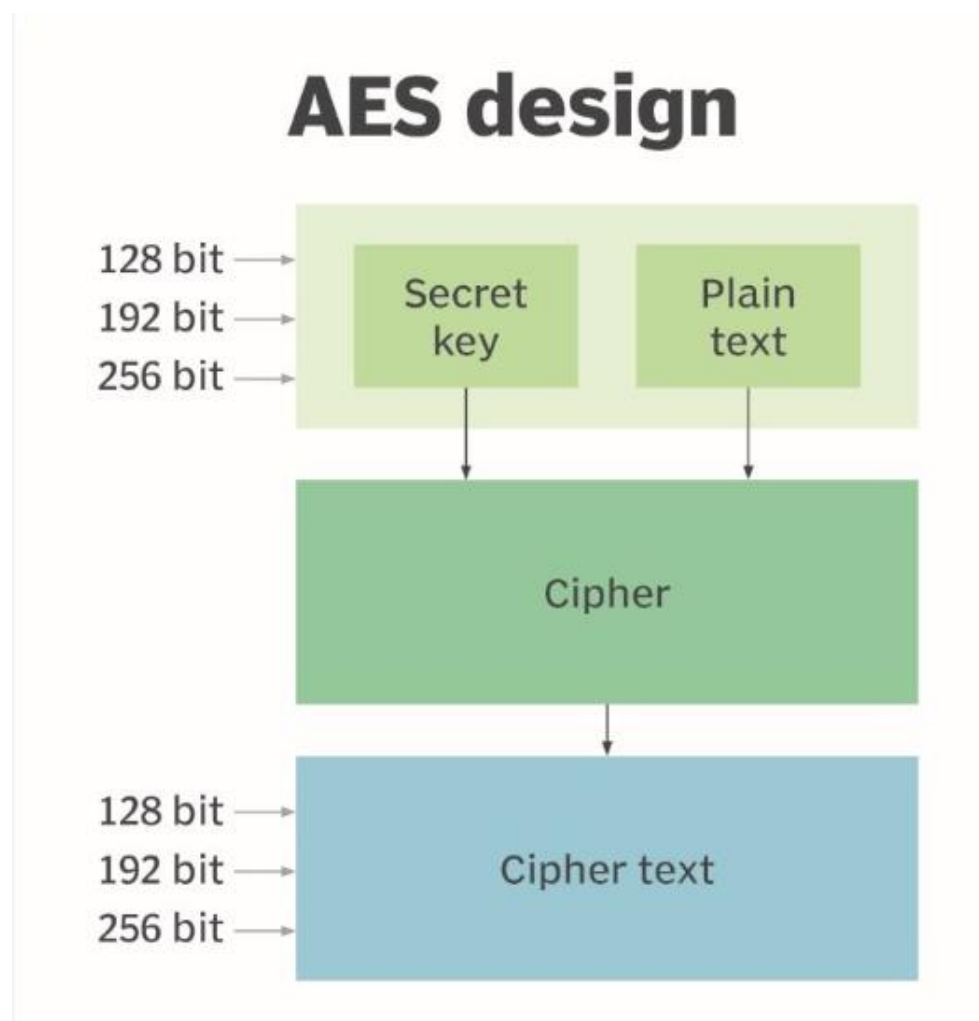
1: Introduction

The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information.

AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.

The objective of this project is to understand the working of AES (Advanced Encryption Standard) and implement AES using verilog and showing successful demonstration.

Block Diagram For AES Algorithm.



Working of AES

1. Dividing data into blocks

First, we have to keep in mind that AES is a **block cipher**. Unlike stream ciphers, it encrypts data in **blocks of bits** instead of bit-by-bit.

Each of its blocks contains a column of 16 bytes in a layout of four-by-four. As one byte contains 8 bits, we get 128-bit block size ($16 \times 8 = 128$).

Thus, the very first step of AES encryption is dividing the plaintext into blocks(4X4 Matrix).

2. Key expansion

This is an important step of AES encryption. It produces new 128-bit round key.

3. Adding round key

This is the very first round of AES encryption.

4. Byte substitution

Now, the AES algorithm substitutes every byte with a code according to a pre-established table called the **S-box**. It looks like this:

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c8
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

5. Shifting rows

In this step, the AES algorithm shifts the rows of the block it got during the byte substitution process.

6. Mixing columns

Talking in mathematical terms, this step multiplies each column by a predefined matrix, giving us a brand new block of code.

7. Adding round key

In this process we apply the round key we got in the *key expansion* section

8. Rinse and repeat

2: Implementation

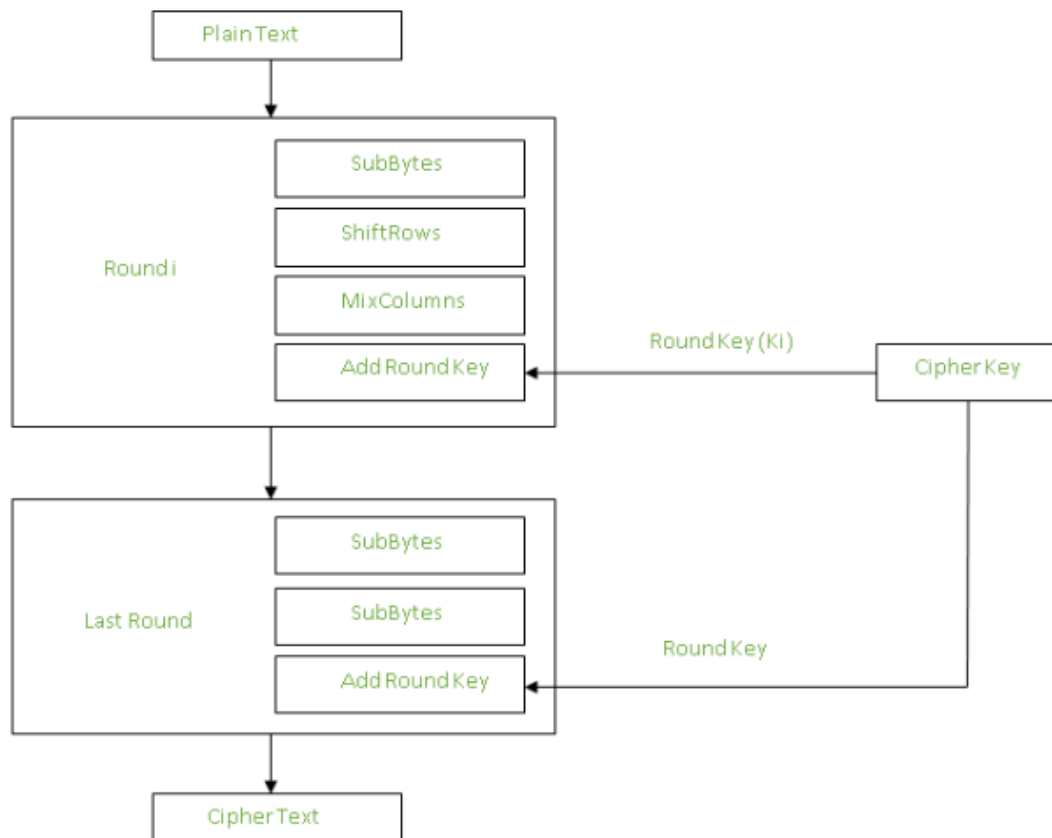
a. Part 1: Creating AES_Core Module

In this module I have instantiated the different modules which are required for the AES implementation.

The modules are.

1. keyGenerationandExpansion
2. subBytes
3. shiftRows
4. mixColumn
5. sbox
6. 9rounds
7. Lastround

These modules are separately created to perform specific function according to following flowchart.



b. Part 2: Creating Testbench

I have written the test bench for the **AES** module. We require test bench just to make sure that the module we have created is working properly. Here, in this testbench I have used some inputs Example with the secret key.

c. Part 3: execution.

Using “vcs -debug -full64testbech.v” command I executed testbench file.

d. Part 4: Simulation

After an execution of all modules, I have run the command “simv” for simulation.

The screenshot shows a terminal window titled "DCD155.csim.edu (ad477306)". The terminal output includes:

```

8 modules and 0 UDP read.
recompiling module rounds9
recompiling module inv_subBytes
recompiling module inv_rounds9
recompiling module keyExp
recompiling module invkeyExp
recompiling module shiftRow
recompiling module subBytes
recompiling module testbench
All of 8 modules done
srm -f _csrc*.srm pre_vcsobj.*.srm share_vcsobj.*.srm
if [ -x ../srm ]; then chmod +x ../srm; fi
g++ -o ../srmv -Wl,-rpath-link=../ -Wl,-rpath=$ORIGIN/simv.daidir/ -Wl,-rpath=../simv.daidir/ -Wl,-rpath=$ORIGIN/simv.daidir/scsim.db.dir
-dynamic -Wl,-rpath=/opt/synopsys/vcs-mx/N-2017.12-SP2-9/Linux64/lib -L/opt/synopsys/vcs-mx/N-2017.12-SP2-9/Linux64/lib objs/amcQw_d.o _l2
685_archive_1.srm SIMLO rmapats.mop.o rmapats.o rmar.o rmar_nd.o rmar_llvm_o_1.o rmar_llvm_o_0.o -lzerosoft_rt_stubs -lvrsim -lerrorinf -lsnpsmaloc -lvfs -lsimnew -lsimprofile -lutclunative /opt/synopsys/vcs-mx/N-2017.12-SP2-9/Linux64/lib/vcs_tls.o -Wl,-whole-archive
-o vcsutil -Wl,-no-whole-archive /opt/synopsys/vcs-mx/N-2017.12-SP2-9/Linux64/lib/vcs_save_restore_new.o -ldl -lc -lm -lpthread -ldl
../srmv up to date
CPU time: 598 seconds to compile + .419 seconds to elab + .293 seconds to link
[1] + Done dve -full64 &
$
$ srmv
Chronologic VCS simulator copyright 1991-2017
Contains Synopsys proprietary information.
Compiler version N-2017.12-SP2-9_Full64; Runtime version N-2017.12-SP2-9_Full64; May 9 22:36 2023
VCDs: N-2017.12-SP2-9_Full64 Runtme (c) 1991-2017 by Synopsys Inc.
***** Encryption *****
O Test Data = 5d776f204f6a5204a696e6e52054776f Key = 5468617473206d79204b756e6720475 Output_Data = 29c350f571420f6402299b31a
02d73a Cipher_Key = 28fddeF86da424acc0a4feB316f26
***** Decryption *****
10 Cipher_Text = 29c350f571420f6402299b31a02d73a Cipher_Key = 28fddeF86da424acc0a4feB316f26 Ret_Data = xxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx Original_Key = 5504f9e40d76c8806e85ed053561895
V C S S i m u l a t i o n R e p o r t
Time: 10000 ps
CPU Time: 0.240 seconds; Data structure size: 0.6Mb
Tue May 9 22:36:41 2023
Follow terminal fold

```

e. Part 5: Creating Log File

After running the simulation I created the log file for testing using the “simv -l Proj.log” command

Command: /home/users14/ad477306/Verilog/Project/./simv -l Pr.log

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version N-2017.12-SP2-9_Full64; Runtime version N-2017.12-SP2-9_Full64; May 9 22:37 2023

VCD+ Writer N-2017.12-SP2-9 Full64 Copyright (c) 1991-2017 by Synopsys Inc.

```
*****
***** Encryption *****
*****
```

```
10 Cipher_Text = 29c3505f571420f6402299b31a02d73a Cipher_Key =
28fddef86da4244accc0a4fe3b316f26 Ret_Data = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Original_Key =
5504f9e40d76c8806e85ed0053561895
```

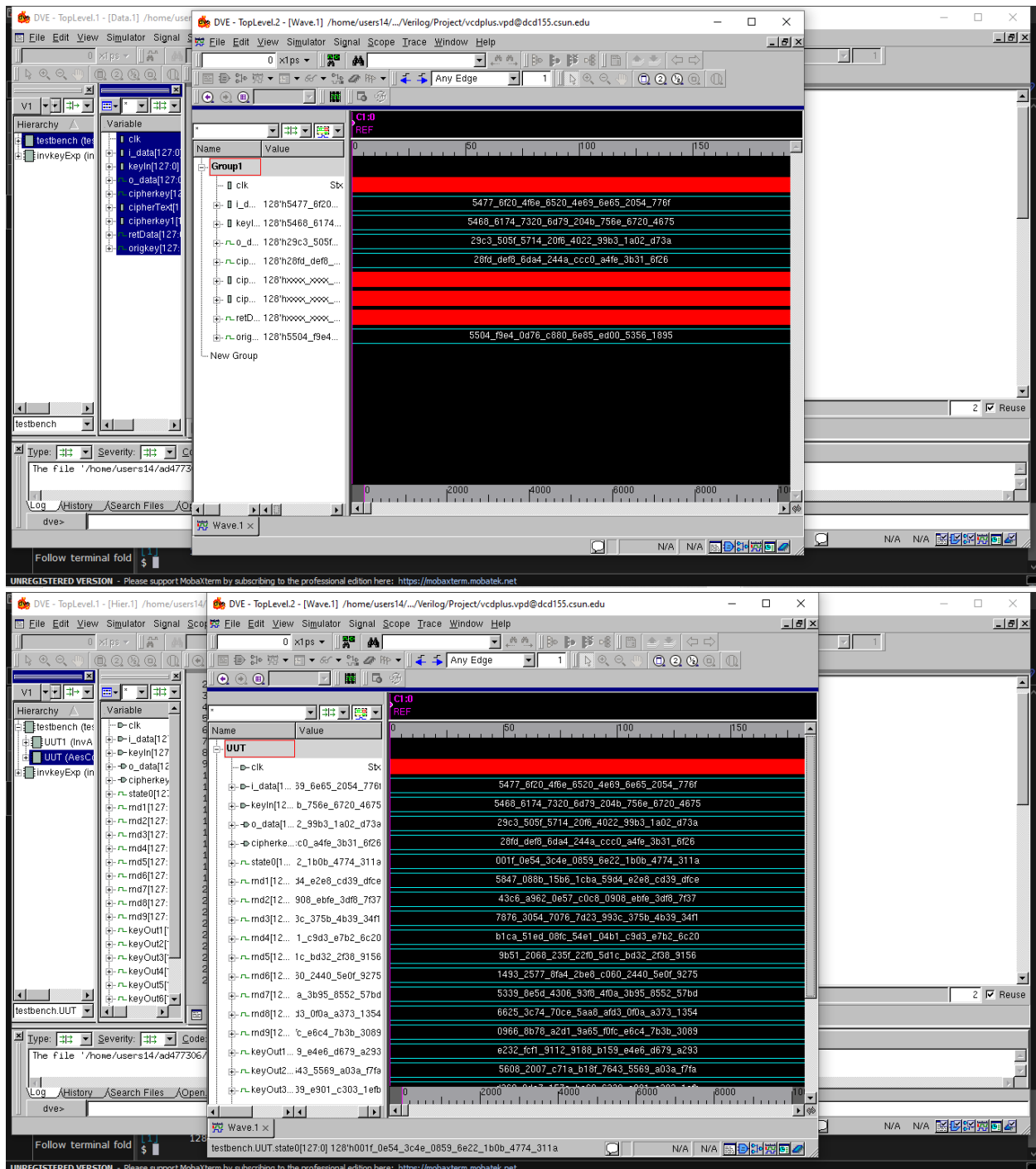
VCS Simulation Report

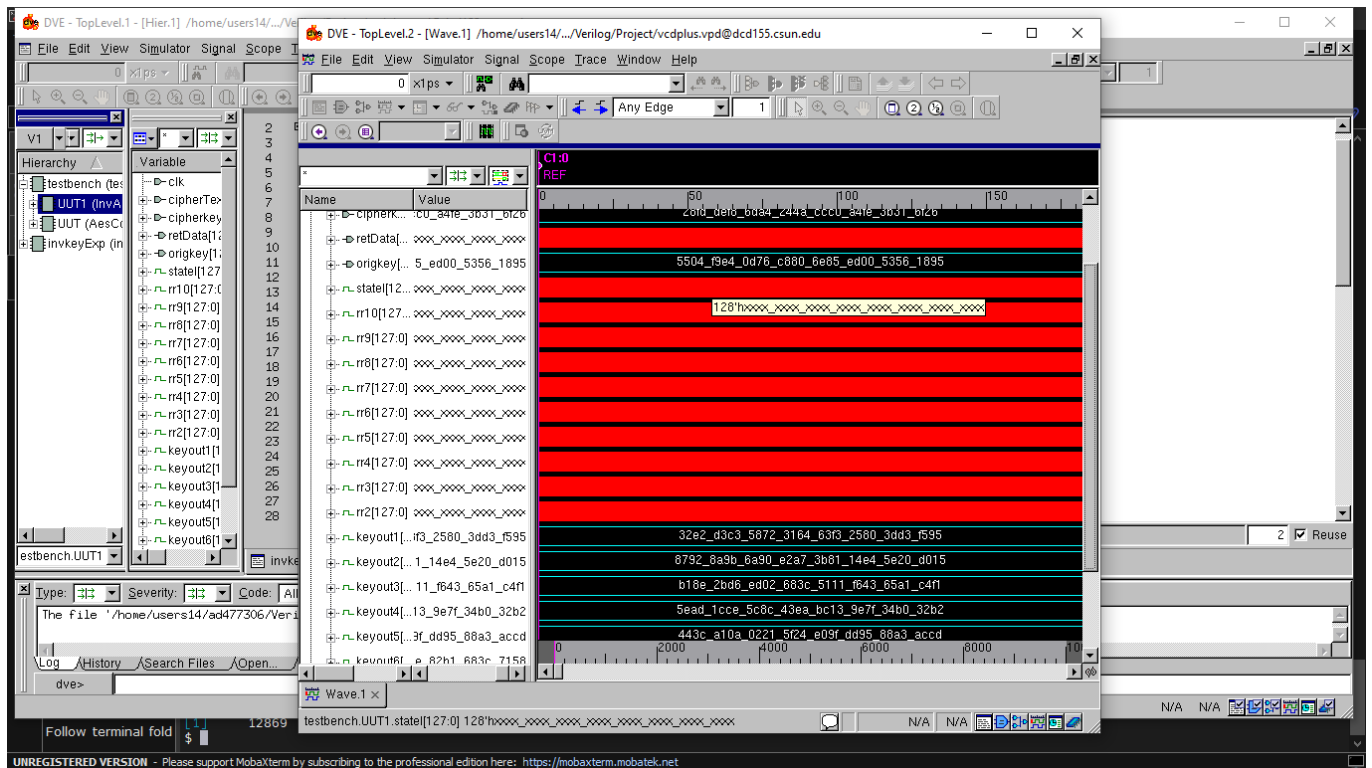
Time: 10000 ps

CPU Time: 0.230 seconds; Data structure size: 0.6Mb Tue May 9 22:37:02 202

f. Part 6: Seeing the waveform.

After creating the log file I opened the DVE using “dve -full64 &” command to see the waveforms.





TESTING :

Message :

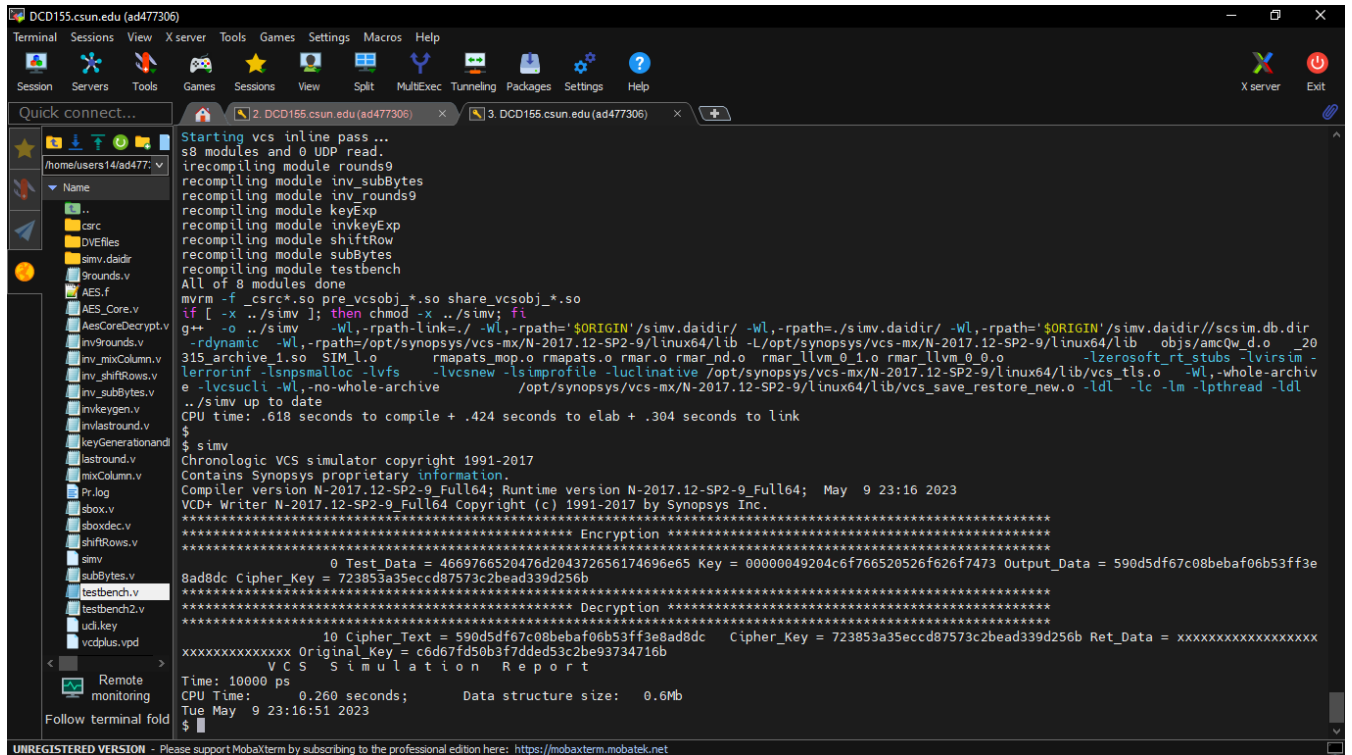
Text: Five G Creatine

Hex : 4669766520476d204372656174696e65

Key :

Text : I Love Robots

Hex : 49204c6f766520526f626f7473



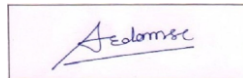
Conclusion:

The constructed AES module in this experiment was verified of its functionalities. The Encryption of message id done successfully, although the decryption of the cipher text back to the original message had some error. But finally I concluded that the encryption algorithm works correctly.(Using solved example form internet).

I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, nor have I allowed or will I allow anyone to copy my work.

Name (printed) Avinash Damse

Name(signed)

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'Avinash Damse' written in a cursive style.

Date : 14-May-2023