$\begin{array}{c} \textbf{CNT5410 Assignment 2} \\ \textbf{Report} \end{array}$

$\label{eq:file_encryption} File\ encryption/decryption/transmission \\ suite\ with\ OpenSSL$

Submitted by Pankaj Nitin Warke

Abstract

The issue of theft of data is always a challenge in this internet connected life. The way information is shared between people has evolved a lot since the age of internet, which makes securing and protecting the data even more important. This assignment explores some of the ways and tools that help protect the data to a extent by encrypting it.

Contents

1	Des	ign Approach	1
2	Intr	roduction	2
	2.1	Theory	2
		2.1.1 PBKDF2	2
		2.1.2 AES256-GCM	2
3	Wo	rk Done	3
	3.1	Implementation choices	3
		3.1.1 Display patterns	3
		3.1.2 PBKDF2	3
		3.1.3 Encryption AES256-GCM mode	3
		3.1.4 Decryption AES256-GCM mode	4
		3.1.5 Network Daemon mode	4
	3.2	How to run	4
		3.2.1 Local mode	4
		3.2.2 Network mode	10
4	Ruk	orics	15
	4.1	"Hello" HEX symmetric key from PBKDF2	15
	4.2	· · · · · · · · · · · · · · · · · · ·	16
	4.3		17
	4.4	-	18
	4.5	V -	19
			19
		V -	19
			19
			20
			20
	4.6		21
			21

	4.6.2	Code samples																								21
--	-------	--------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

List of Figures

3.1	make command
3.2	ufsend local command
3.3	ufsend local files
3.4	ufsend local remove
3.5	ufsend local decrypted
3.6	ufsend local resulting files
3.7	Remote ufrec Inbound file
3.8	Remote ufsend send file encrypted text
3.9	Remote ufsend bytes encrypted and file transmission 12
3.10	Remote ufrec downloaded bytes and password 13
3.11	Remote ufrec decrypted file size
4 1	1 CIT II
4.1	hex of Hello
4.2	Hex of encrypted example.txt
4.3	Hex of received file with 12B IV
4.4	Hex of decrypted file and plaintext
4.5	Hex of encrypted file
4.6	first byte of enc file
4.7	mod first byte of enc file
4.8	Attempt to decrypt
4.9	Decrypted gibberish
4.10	
4.11	Exit when decrypted already present
	Exit when encrypted file already present

Design Approach

The assignment expected a set of programs, ufsend and ufrec that work in local as well as network daemon mode. The programs are responsible for encryption and decryption of the file using the AES-256-GCM mode.

The design of the program is as per below steps

- PBKDF2 algorithm with SHA3-256(HMAC) and 4096 iterations for key generation.
- The salt is fixed in this, set to "SodiumChloride".
- IV is generated as random bytes using the RAND from OpenSSL, and is prepend to the encrypted file output.
- File encryption and decryption algorithm is AES256 in Galois Counter Mode(GCM).
- Network daemon is setup using TCP sockets, and with the help of arpa/inet.h header file.
- As per my tests program is capable of encrypting text files, image files, and machine code.
- make utility has been used to compile the programs.

Introduction

2.1 Theory

2.1.1 PBKDF2

PBKDF2 is an algorithm used to derive key from given password by applying a HMAC function to the given password and additional salt. For randomised salt the same password from will have different derived key, thereby reducing the rainbow attacks.

2.1.2 AES256-GCM

AES256-GCM is an authenticated mode of encryption operation and is composed of two seperate functions, AES-CTR for encryption and GMAC for authentication. For optimal encryption it is expected that the KEY and IV pair is never reused.

The input received is

- 1. a Key
- 2. a unique IV
- 3. AAD, data to be processed only with authentication.
- 4. Data to be encrypted

It outputs the following

- 1. Encrypted data of input 4
- 2. Authentication tag

Work Done

3.1 Implementation choices

3.1.1 Display patterns

- BIO_dump_fp is used at places to display encrypted and decrypted data in Hex format. The main reason was simplicity of its use.
- Error messages are printed using perror() and exit with 1 as return.
- If encrypted/decrypted file is already present the program returns 33 with appropriate message.

3.1.2 PBKDF2

- Password is taken as input from the user, whereas the salt for the purpose of this assignment is set as "SodiumChloride".
- The number of iterations is set to 4096.
- Key length is of 256 bits, and IV length is 96 bits.
- The IV is generated by using RAND_bytes() function from OpenSSL.
- Generated key is displayed in hex format.

3.1.3 Encryption AES256-GCM mode

• A total of four parameters are passed to gcm_encrypt function, key, iv and file pointers for input and output file.

- The IV is prepend to output of encrypted file.
- The number of bytes encrypted and encrypted text will be displayed after execution, however the output file will size will always be 12 bytes more than original file due to IV.

3.1.4 Decryption AES256-GCM mode

- A total of four parameters are passed to gcm_decrypt function, key, iv and file pointers for input and output file.
- The IV is extracted from the file's first 12 bytes, however it's also sent in the function call just to keep it consistent with the encrypt function.
- The number of bytes decrypted and decrypted text will be displayed after execution.

3.1.5 Network Daemon mode

- Server(ufrec) listens on the port specified from CLI argument.
- Client(ufsend) sends the encrypted file to specified IP:PORT after setting up the connection.

3.2 How to run

3.2.1 Local mode

1. Use make utility to compile both C files.

\$ make

Refer figure 3.1.

2. Run ufsend with *filename* to be encrypted along with -l flag.

\$./ufsend filename -l

Refer figure 3.2.

3. Enter the password for encryption. A file with the same filename as input but with *.ufsec* extension will be generated in the working directory e.g. *filename.ufsec*. Refer figure 3.2. Refer figure 3.3.

4. Remove the original plaintext file.

\$ rm filename

Refer figure 3.4.

5. Run ufrec with the filename, e.g. *filename.ufsec*, to be decrypted along with -l flag.

```
$ ./ufrec filename.ufsec -l
```

Refer figure 3.5

- 6. Enter the password for decryption. Refer figure 3.5
- 7. The decrypted file will be stored without the .ufsec extension. Refer figure 3.6

```
pnkjserver@pnkjserv:~$ make
gcc -o ufsend ufsend.c -lcrypto -lssl
gcc -o ufrec ufrec.c -lcrypto -lssl
```

Figure 3.1: make command

```
nder:~/cns> ./ufsend example.txt -1
 03b0 - db a2 f3 37 9a c6 f9 5b-8b 9d a4 d3 2f 72 69 01
 03c0 - ed d8 7a c4 16 9b 57 3e-d9 8b d4 d3 5a 15 ed dc
03d0 - 2d 96 a6 92 bc 26 97 96-lb 5f 2a ee 9e 57 72 d3
                                                                                               -...&...*..Wr.
03d0 - 2d 96 ac 92 bc 2c 97 9c-1b 5f 2a ee 9e 57 72 d3

03e0 - 95 c9 fl 22 la ed 8e 2d-32 22 dc 30 83 bd 3c 7b

03f0 - 3e b9 57 2c 98 b0 20 0f-c8 d4 c2 87 6e a0 c9 6b

0000 - 74 8a 82 34 al c7 64 17-bc 1d 4e 82 30 0f f6 c8

0010 - 53 0a a4 e5 c9 07 25 6b-dd c6 ba da e7 e0 3b 60

0020 - 78 98 le 73 0d 82 2l 28-fc 09 85 f8 9d 4e e7 70

0030 - lc e7 f2 2b 47 88 05 96-0f 5f 7l 5f fl ac 2f 38
                                                                                                x..s..!(....N.p
                                                                                               ...+G..._q_../8
jc....F...~.E.
..FJ....LM..uH.
 0040 - 6a 63 f4 af cf d3 01 46-09 e6 be 85 7e ef 45 93
 0050 - bd c6 46 4a 83 c3 f8 d3-a9 4c 4d d0 0e 75 48 1d
 0060 - c8 51 a7 06 03 8d 89 el-fb dd al 06 c0 21 c4 9e
 0070 - 3a 4c ad 4f 85
                                    49 00 bc-78 c6 09 cb
 1151 bytes encrypted.
thunder:~/cns>
```

Figure 3.2: ufsend local command

```
thunder:~/cns> 1s
example.txt example.txt.ufsec makefile ufrec ufrec.c ufsend ufsend.c warke-assign2.tgz
thunder:~/cns>
```

Figure 3.3: ufsend local files

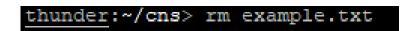


Figure 3.4: ufsend local remove

```
Exhance: -/ cmap. /uffece example.txt.ufsec -1

Example to pool 18

Expanding to pool 19

Expanding to pool 19
                                                                                                                                                                                               ./ufrec example.txt.ufsec -1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       rt their homewor
k early tend to
do well..Those w
ho don't... well
....Have you sta
rted thinking ab
out projects?.Ti
me to start :).
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      64 20 74 6f
6f 73 65 20
20 77 65 6c
75 20 73 74
6e 67 20 61
                                                            - 64 61 20 77 65 66 66 2e-0a 54

- 68 6f 20 64 6f 6e 27 74-2e 2e

- 2e 2e 2e 0a 48 61 76 65-20 79

- 72 74 65 64 20 74 68 69-6e 6b

- 6f 75 74 20 70 72 6f 6a-65 63

- 6d 65 20 74 6f 20 73 74-61 72
                                                                                                                                                                                                                                                                                                                                                      74-2e 2e 2e
65-20 79 6f
69-6e 6b 69
6a-65 63 74
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           6e
73
20
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             69
                    .151 bytes decrypted.
```

Figure 3.5: ufsend local decrypted

```
thunder:~/cns> 1s -1
total 79
-rw-r--r- 1 pwarke grad 1151 Sep 21 13:03 example.txt
-rw-r--r- 1 pwarke grad 1163 Sep 21 12:32 example.txt.ufsec
-rw-r--r- 1 pwarke grad 86 Sep 21 10:14 makefile
-rwxr-xr-x 1 pwarke grad 26280 Sep 21 12:32 ufrec
-rw-r--r- 1 pwarke grad 8158 Sep 21 10:00 ufrec.c
-rwxr-xr-x 1 pwarke grad 26248 Sep 21 12:32 ufsend
-rw-r--r- 1 pwarke grad 7066 Sep 21 12:32 ufsend
-rw-r--r- 1 pwarke grad 7066 Sep 21 11:40 ufsend.c
-rw-r--r- 1 pwarke grad 4032 Sep 21 12:15 warke-assign2.tgz
```

Figure 3.6: ufsend local resulting files

3.2.2 Network mode

1. Use make utility to compile both C files.

\$ make

- 2. Run ufrec with the filename, e.g. *filename.ufsec*, to be decrypted along with -d flag and the *PORT*. Enter the password for encryption at client side. Refer figure 3.7
 - \$./ufrec filename.ufsec -d 9999
- 3. Run ufsend with *filename* to be encrypted along with -d flag and *IP:PORT*. Refer figure 3.8
 - \$./ufsend filename -d 192.168.0.13:9999
- 4. A file with the same filename as input but with .ufsec extension will be displayed in terminal and generated in the working directory e.g. filename.ufsec. The encrypted file along with iv bytes will be transmitted over the network to receiver/server. Refer figure 3.10
- 5. The encrypted file will be displayed and stored at server side. Refer figure 3.10
- 6. Enter the password for decryption at server side. Refer figure 3.10
- 7. The decrypted file will be displayed and stored without the .ufsec extension on the server. Refer figure 3.11

Figure 3.7: Remote ufrec Inbound file

```
PRESENTIC: google

Rex password: google

Rex password: so 33A657D78ESS14C19SCG73DC2AS12945B698774CAAET077AB508565GA7CC43D

0000 - 75 Ce 44 80 4f aa de 80-47 2f el 17 b4 0e 56 d0

0000 - 75 Ce 44 80 4f aa de 80-47 2f el 17 b4 0e 56 d0

0000 - 75 Ce 48 80 4f aa de 80-47 2f el 17 b4 0e 56 d0

0000 - 75 Ce 48 80 4f aa de 80-47 2f el 17 b4 0e 56 d0

0000 - 75 Ce 48 80 4f aa de 80-47 2f el 17 b4 0e 56 d0

0000 - 75 Ce 48 80 4f aa de 80-47 2f el 27 de 2
```

Figure 3.8: Remote ufsend send file encrypted text

Figure 3.9: Remote ufsend bytes encrypted and file transmission

Figure 3.10: Remote ufrec downloaded bytes and password

Figure 3.11: Remote ufrec decrypted file size

Rubrics

4.1 "Hello" HEX symmetric key from PBKDF2

```
pnkjserver@pnkjserv:~/CNS_Assign2$ ./ufsend example.txt -1

Password: Hello
Hex password is 18F3CFE2232B51DB3A08FC56897952E6228276F6EAB57E01DD832CDBB6510159
```

Figure 4.1: hex of Hello

4.2 Hex value of encrypted file

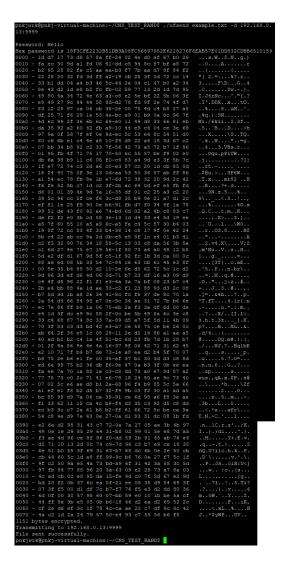


Figure 4.2: Hex of encrypted example.txt

4.3 ufdec receipt and hex of encrypted file

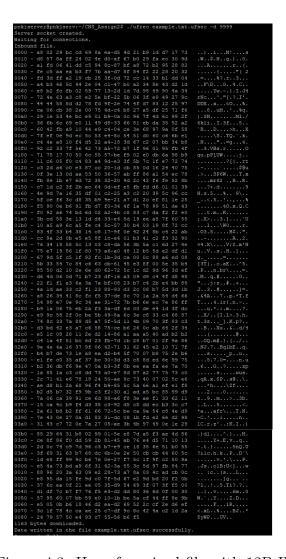


Figure 4.3: Hex of received file with $12B\ IV$

4.4 Decrypted file and value

Password Hex pass				F3C1	FE 20	232F	851DB32	1804	FC5	5891	795:	2E.63	2282	276F6	EAB57E01DD832CDBB6510159
0000 - 1	53 74	75	64	65	6e	74	73-20	77	68	6f	20	73	74		Students who sta
0010 - 0	72 74 6b 20		74 61	68 72	65 6c	69 79	72-20 20-74	68 65	6f	6d 64	65 20	77 74	6f 6f	72 20	rt their homewor k early tend to
0030 -	64 61														do wellThose w
0040 - 0				6f 48			74-2e 65-20				77	65 73		6c 61	ho don't well Have you sta
0060 - 1							69-6e			6e					rted thinking ab
0070 - 0	6f 75 75 64		20 6e	70 74	72 73	6f 20	6a-65 77-68	63 6£	74 20	73 73	3f 74	0a 61	53 72	74 74	out projects?.St udents who start
0090 - 2	20 74						68-6f	6d							their homework
00a0 - 0			6c 6c	79 60	20 2e		65-6e 54-68	64 6£		74 65	6f 20	20 77	64 68	6f 6f	early tend to do wellThose who
00c0 - 2	20 64						2e-2e								don't well
00d0 - 2	2e 0a 65 64	48	61 74	76 68	65 69		79-6f 6b-69			73 20	74 61	61 62	72 6f	74 75	Have you start ed thinking abou
00f0 - '							63-74								t projects?.Stud
0100 - 0			73 72	20	77 68		6f-20 6d-65	73	74 6f	61 72	72 6b	74 20	20 65	74 61	ents who start t heir homework ea
				74	65				6f		64	6f	20	77	rly tend to do w
0130 - 0 0140 - 0		6c					6f-73 20-77	65 65		77 60	68 2e	6f 2e	20 2e	64 0a	ellThose who d on't well
	48 61														Have you started
	20 74 70 72	68 6f	69 6a	6e 65	6b 63		6e-67 73-3f	20 0a	61 53	62 74	6f 75	75 64	74 65	20 6e	thinking about projects?.Studen
	74 73			68	6f		73-74		72	74		74	68		ts who start the
0190 - 0 01a0 - 1			68 65			65	77-6f 74-6f	72	6b	20 6f		61 77		6c 6c	ir homework earl y tend to do wel
01b0 - 0	6c 2e	0 a					65-20								1Those who don
01c0 - 2	27 74 76 65		2e 79	2e 6f	20 75	77 20	65-6c 73-74	6c 61	2e 72	2e 74	2e 65	0a 64	48 20	61 74	't wellHa ve you started t
	68 69														hinking about pr
01f0 - 0 0200 - 2			63 6£		73 73		0a-53 61-72			64 74	65 68	6e 65	74	73 72	ojects?.Students who start their
	20 68	6f									72	6c	79	20	homework early
0220 - 0			64 6f	20 73	74 65	6f 20	20-64 77-68			77 64	65 6£	6c 6e		2e 74	tend to do well. .Those who don't
0240 - 2	2e 2e	2 e							2e						wellHave
0250 - 2			75 6e	20 67	73 20	74 61	61-72 62-6f	74 75	65 74	64 20	20 70	74 72	68 6£	69 6a	you started thi nking about proj
0270 -					0a					6e					ects?.Students w
0280 - 0 0290 -	68 61 6f 6d	20 165	73 77	74 6f	61 72		74-20 20-65					72 20			ho start their h omework early te
02a0 - 0	6e 64				20							2e			nd to do wellT
02b0 - 0 02c0 - 3			65 65	20			6f-20 2e-2e		6f	6e 61		74 65	2e	2e 79	hose who don't . wellHave y
02d0 -	6f 75				61		74-65						6e		ou started think
02e0 - 0 02f0 - 1	69 66 74 73	67 3 3f			62 74		75-74 64-65			72 73	6f 20	6a 77	65 68	63 6f	ing about projec ts?.Students who
0300 - 2	20 73	74		72	74	20	74-68	65		72		68	6f	6d	start their hom
0310 - 0			72 20	6b 64	20 6f		61-72 77-65	6c	79	20 2e	74 0a	65 54	6e 68	64 6f	ework early tend to do wellTho
0330 -	73 65		77	68	6f		64-6f			74	2e	2e	2e	20	se who don't
0340 - 1 0350 - 1			6c 61				0a-48 64-20					79 6b		75 6e	wellHave you started thinkin
0360 - 6	67 20									6a					g about projects
0370 - 3	3f 0a 74 61		74 74	75 20	64 74	65 68	6e-74 65-69	73 72	20 20	77 68	68 6f	6f 6d	20 65	73 77	?.Students who s tart their homew
0390 - 0	6f 72	6b				72				65	6e	64	20		ork early tend t o do wellThose
03a0 - 0			6f 6f	20 20	77 64		6c-6c 6e-27			54 2e	68 2e	6f 20	73 77	65 65	o do wellThose who don't we
03c0 - (6c 6c	2 e													11Have you s
03d0 - 1	74 61 61 62		74 75	65 74	64 20	20 70	74-68 72-6f	69 6a	6e 65	6b 63	69 74	6e 73	67 3f	20 0a	tarted thinking about projects?.
03f0 - 5	53 74		64				73-20								Students who sta
0000 - 0				68 72			72-20 20-74		6f 6e	6d 64	65 20	77 74	6f 6f	72 20	rt their homewor k early tend to
	64 61									6f			20		do wellThose w
0030 - 0			64 0a		6e 61	27 76	74-2e 65-20	2e 79	2e 6f	20 75	77 20	65 73	6c 74	6c 61	ho don't well Have you sta
0050 - '						68	69-6e	6b							rted thinking ab
0060 - 0	6f 75 6d 65				72 20		6a-65 74-61			73 20		0a 29	54 0a	69	out projects?.Ti me to start :).
1151 by	tes	iecr	vnt	ed											
pnkjser					CN:	As	ssign2	ŝ							

Figure 4.4: Hex of decrypted file and plaintext

4.5 Modify ciphertext in local

4.5.1 Hex of encrypted file

Figure 4.5: Hex of encrypted file

4.5.2 First byte of encrypted file

```
0060 - db b8 24 47 67 ac 49 50-f9 df 60 ld 03 bc e9 40 ..$Gg.IP..`...@

0070 - 4b fa 6d 4a 2a 7d c0 00-7d aa b6 d9 ab dc 2b K.mJ*}....+

1151 bytes encrypted.

pnkjserver@pnkjserv:~/CNS_Assign2$ head -c l example.txt.ufsec

apnkjserver@pnkjserv:~/CNS_Assign2$ echo -ne \\xFF L dd convenotrunc hs=l cou
```

Figure 4.6: first byte of enc file

4.5.3 Modified first byte of encrypted file

Figure 4.7: mod first byte of enc file

4.5.4 Attempting to decrypt

```
pnkjserver@pnkjserv:~/CNS_Assign2$ mv example.txt example.txt.bk
pnkjserver@pnkjserv:~/CNS_Assign2$ ./ufrec example.txt.ufsec -1
Password: Hello
Hex password is 18F3CFE2232B51DB3A08FC56897952E6228276F6EAB57E01DD832CDBB6510159
0000 - bl 5d df 93 9f 24 ab 49-66 lf c6 49 e2 40 a7 8b .]...$.If..I.@..
0010 - e6 37 46 64 8a 58 l1 18-62 5e 53 17 31 3d 27 f4 .7Fd.X..b^S.l='.
0020 - 92 5b 97 75 27 9e 7b lc-33 90 36 00 cb d6 82 51 .[.u'.{.3.6...Q}
0030 - 74 0f c5 17 42 16 87 18-84 78 17 b0 8e 18 84 3d t...B...x.....=
0040 - 38 1f 7a df 2e 6c 70 86-57 28 c2 94 a7 ae 00 5e 8.z..lp.W(.....^
```

Figure 4.8: Attempt to decrypt

4.5.5 Running diff on decrypted file

Figure 4.9: Decrypted gibberish

4.6 Show graceful exit codes

4.6.1 Sample output

```
pnkjwrk@pnkj-virtual-machine:~/CNS_TEST_RAND$ ls
example.txt example.txt.ufsec makefile ufrec ufrec.c ufsend ufsend.c
pnkjwrk@pnkj-virtual-machine:~/CNS_TEST_RAND$ ./ufrec example.txt.ufsec -1
Decrypted file exists.
: Success
pnkjwrk@pnkj-virtual-machine:~/CNS_TEST_RAND$ ./ufsend example.txt -1
Encrypted file exists.
: Success
pnkjwrk@pnkj-virtual-machine:~/CNS_TEST_RAND$
```

Figure 4.10: Exit when file already present

4.6.2 Code samples

```
146
                exit(1);
147
148
149
150
                enc_file_fp = fopen(inFile, "r");
                int len1 = strlen(inFile);
151
152
                char* outFile = strndup(inFile, lenl>=strlen(ext)?lenl-strlen(ext):0);
153
               if(access(outFile, F_OK) == 0) {
154
                perror("Decrypted file exists.\n");
155
156
                return 33;
157
158
```

Figure 4.11: Exit when decrypted already present

```
90
91
          FILE *toenc_file, *enc_file;
92
          toenc_file = fopen(inFile, "r");
93
          char *ext = ".ufsec";
94
95
          char *encfilename = strcat(inFile, ext);
96
97
          // check if encrypted file exists
98
          if(access(encfilename, F_OK) == 0) {
99
             perror("Encrypted file exists.\n");
              return 33;
```

Figure 4.12: Exit when encrypted file already present