

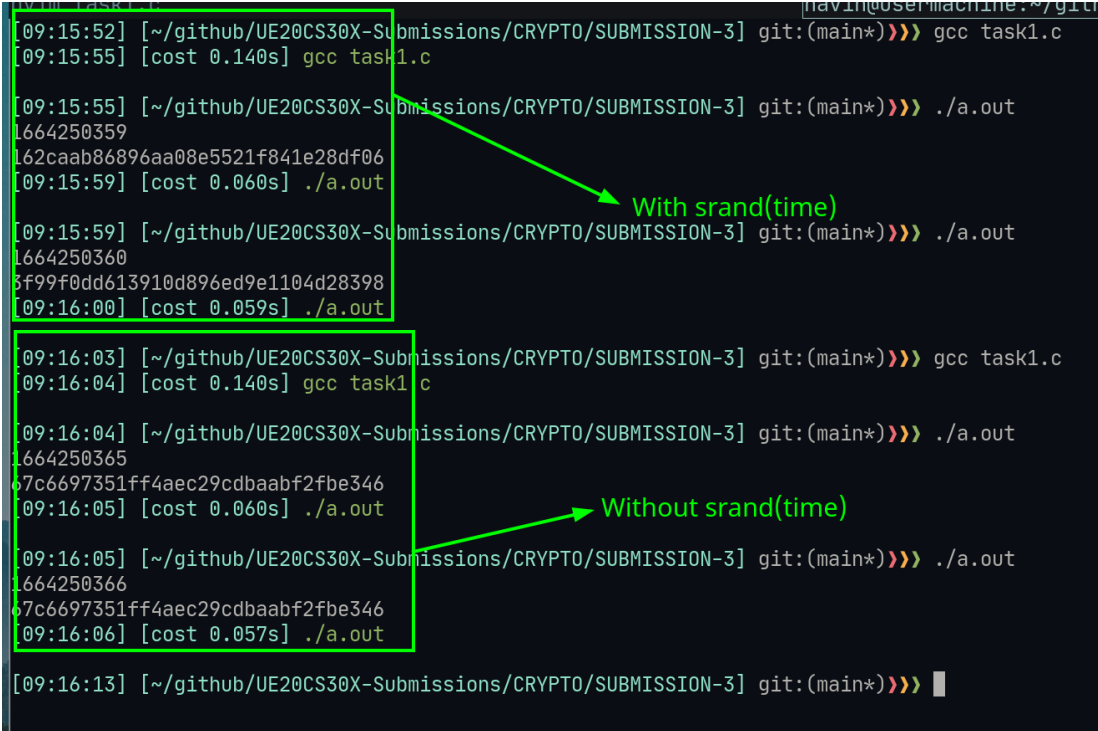
Cryptographphy Hands-On submission 3

Details :

- SRN : PES2UG20CS237
- Name : P K Navin Shrinivas
- Section : D

Task 1 : Generating prng the wrong way

Screenshot :



```
[09:15:52] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> gcc task1.c
[09:15:55] [cost 0.140s] gcc task1.c

[09:15:55] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
1664250359
162caab86896aa08e5521f841e28df06
[09:15:59] [cost 0.060s] ./a.out

[09:15:59] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
1664250360
3f99f0dd613910d896ed9e1104d28398
[09:16:00] [cost 0.059s] ./a.out

[09:16:03] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> gcc task1.c
[09:16:04] [cost 0.140s] gcc task1.c

[09:16:04] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
1664250365
7c6697351ff4aec29cdbaabf2fbe346
[09:16:05] [cost 0.060s] ./a.out

[09:16:05] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
1664250366
7c6697351ff4aec29cdbaabf2fbe346
[09:16:06] [cost 0.057s] ./a.out

[09:16:13] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> █
```

With srand(time)

Without srand(time)

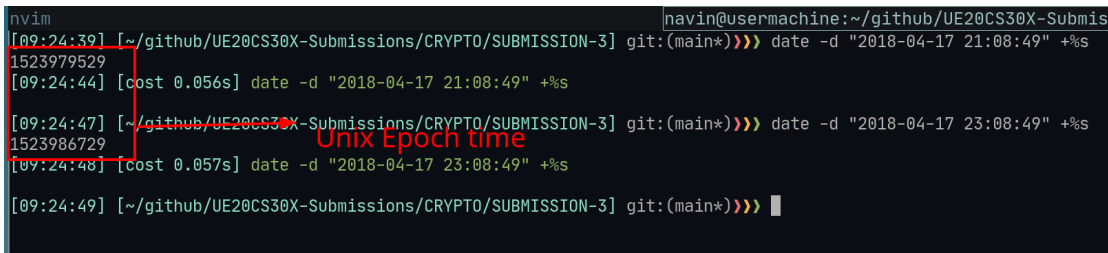
Observation :

- The time stamp is being used as key for the
- If srand(time()) is removed, then the generated number remains the same

Task 2 : Guessing the key

Step 1 : getting UNIX time to bruteforce decryption

Screenshot :



```
nvim navin@usermachine:~/github/UE20CS30X-Submis
[09:24:39] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*) date -d "2018-04-17 21:08:49" +%s
1523979529
[09:24:44] [cost 0.056s] date -d "2018-04-17 21:08:49" +%s
[09:24:47] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*) date -d "2018-04-17 23:08:49" +%s
1523986729
[09:24:48] [cost 0.057s] date -d "2018-04-17 23:08:49" +%s
[09:24:49] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)
```

Observation :

- Above two are the UNIX epoch timestamps using which we generate all possible keys for decrypting
- The above two are different from the ones shown in video as I am using IST and not UTC on the contrary. Do note the times in code is following UTC.

Step 2 : Bruteforce decrypting

Code :

```
/* task2.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
void main() {
    int i, j;
    FILE *f;
    char key[KEYSIZE];
    int value1, value2;
    value1 = 1524013729;
    value2 = 1524020929;
    f = fopen("keys.txt", "w");
    for (j = value1; j <= value2; j++) {
        srand (j);
        for (i = 0; i < KEYSIZE; i++) {
            key[i] = rand()%256;
            fprintf(f, "%.2x", (unsigned char)key[i]);
        }
        fprintf(f, "\n");
    }
}
```

Screenshot :

```
[09:32:52] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> gcc task2.c
[09:32:54] [cost 0.140s] gcc task2.c

[09:32:54] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
[09:32:57] [cost 0.086s] ./a.out

[09:32:58] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> cat keys.txt | head 5
head: cannot open '5' for reading: No such file or directory
[09:33:07] [cost 0.055s] cat keys.txt | head 5

[09:33:25] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> head -n 5 keys.txt
f243bb05ec67c06c7a6a820137a98789
7ed87572ab02a5fa85aac9244eaa19a8
701ce20ebe9ccf0cfd17f805ee9369af
c99e333e665ce424a0ac1682d5121b5c
c29a311f4233068c4019b3330bbe56a2
[09:33:29] [cost 0.057s] head -n 5 keys.txt

[09:33:30] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> █

[09:40:23] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> wc -l keys.txt
7201 keys.txt
[09:40:26] [cost 0.069s] wc -l keys.txt

[09:40:27] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> █
```

Observation :

- Here we have generated all possible keys (of which few are seen in the Screenshot) that are possible between the two time stamps
- We also see we have 7200 keys which we will be using for brute force

Step 3 : Breaking encrypted text

Screenshot :

```
Encrypted: 312403e887832dd07805bd3f32437d
Encrypted: aac4b8af02903130bd4af52facc4ddfa
Encrypted: 18689ed3655a69d5bd96b3035b3eae
Encrypted: 845272c5bae60ddbfc358a4d4cfc733a
Encrypted: d75b4f283a7961979e33cd718e074ada
Encrypted: 07b912cc5244db291f274ebca4f45268
Encrypted: c92d7aa443327892fb20a74c3d31eb18
Encrypted: 39e3d058c81a41aabc4dfb2b88e71aab
Encrypted: 8b29cc48852675fa11162be89af8e6f7
[19:45:37] [cost 0.969s] python3 task3.py

[19:45:55] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> python3 task3.py

Match found
Match found
key: 95fa2030e73ed3f8da761b4eb805dfd7
Ciphertext: d06bf9d0dab8e8ef880660d2af65aa82
Encrypted: d06bf9d0dab8e8ef880660d2af65aa82

[19:45:56] [cost 0.770s] python3 task3.py

[19:46:14] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> █
```

task 3 : Seeing kernel entropy :

Screenshot :

```
nvim | watch -n .1 cat /proc/sys/kernel/random/entropy_avail
Every 0.1s: cat /proc/sys/kernel/random/entropy_avail usermachine: Tue Sep 27 20:06:13 2022
256
```

Observation :

- The change in entropy is too little and very rapid

Task 4 : /dev/urandom :

Screenshot :

```
1 2 3 | watch -n .1 cat /proc/sys/kernel/random/entropy_avail
navin@usermachine:~ | Every 0.1s: cat /proc/sys/kernel/random/entropy_avail usermachine: Tue Sep 27 20:31:48 2022
256
0c195c8 1ea0 798c 86c8 bffc 71b6 d266 0f56 18ae
0c195d0 bd64 2bdc 87aa 4d5e d989 98ff 3574 6380
0c195e0 8eca 022a c483 d29e 529d e8d3 1749 4223
0c195f0 5b02 dc16 2d87 5423 49aa b7ce 242a 87d6
0c19600 4cbf 2938 28d4 a16f 04ff 22bd b3cc c1d7
0c19610 9189 b894 a90a 7d71 ba82 781e 863c 6b87
0c19620 c7f1 ac07 c754 e2c8 1316 df53 84eb 4626
0c19630 bd89 8e29 91bb 7948 d982 7f0e 8cff 2875
0c19640 e6ef d8ea 6667 c258 cd98 9dca efe6 51b1
0c19650 31bb 0711 36db 781a 47fa 921d 5a1d 2b9e
0c19660 5a5a c5d3 e9cf 121e 2de6 f9de 17e4 344e
0c19670 b498 7d55 e317 db43 6a33 45ef a1ce 641e
0c19680 9985 b778 0efd 56e3 0c58 ab37 43ef 38fb
0c19690 98e1 218b 04b5 679f ff45 25fd ddc7 f138
0c196a0 c4a2 9a45 3965 a56c 638f 62e7 1d8b 446b
0c196b0 f8cf 3d9d 1a16 cb59 aff0 0526 bf0e ed77
0c196c0 8e79 e4ef f6f0 d550 47f0 c87e b11b a794
0c196d0 9685 8141 04a6 43a8 a612 53cf 8b86 9314
0c196e0 e9b3 494d 9f1a 9811 de6a 82d9 fd58 5dd6
0c196f0 b85e 8785 9d6b 3223 8e27 5876 6588 84f2
0c19700 b851 e85e eb05 f831 5566 88c2 915f c116
0c19710 aab5 5813 e8be 49f0 7f45 c835 edee 9879
0c19720 d9af 9584 0cf1 4257 6137 3ff8 cd4b 1818
0c19730 3cf1 75db 654f 6bb8 6139 b2a5 cbeb c1ab
0c19740 1dcc 62dd b09e 4ea4 fbc8 8b6e addb 698b
0c19750 8b37 0d34 5d85 db24 3948 2446 6aef ccc1
0c19760 938e a2d1 3bcf 9c4f 7f0d 8fcf a071 44eb
0c19770 54c9 3712 7edc 58d3 3bf7 ac26 3afb 0c9f
0c19780 8cd2 fec4 b4a8 0a86 e847 a57e d899 4cb8
0c19790 6af2 b885 8968 1011 4252 d171 af48 e65b
0c197a0 7539 f73a eac6 4580 6152 f995 2c8e 3682
0c197b0 a832 4d09 5837 dd57 7bfb e659 4299 7af1
0c197c0 6ec9 d63e f6ec 4c8f b310 797f 14e8 1273
0c197d0 98e2 2eea 15a2 1f5c f6b6 1a3a 35a7 516f
0c197e0 76b3 e58f 081d fe8a f92e 998a 1c69 9368
0c197f0 6e9f 4e8b f6c2 9a5c 7db3 7e45 fb16 b3ad
0c19800 c851 3cbb d4c8 eck6 5c53 5cc1 a5a8 3bc2
0c19810 373a 8681 5597 c30f a4f2 8289 3721 d188
0c19820 cf6e b52f 585d 25f5 cebe da42 e988 116f
0c19830 5a88 a9e5 f854 973f 577c 88d7 9dcb 514e
0c19840 afeb 98d1 08de 5e28 db3b 0ccc ee8e 2c9c
^C
[20:31:46] [cost 6.876s] cat /dev/random | hexdump
[20:31:48] [-] >>>
```

Observation :

- On mouse movement we see large change in generate hex dump

Task 5 : /dev/urandom :

Step 1 :

Screenshot :

```

navin@usermachine:~$ watch -n .1 cat /proc/sys/kernel/random/entropy_avail
Every 0.1s: cat /proc/sys/kernel/random/entropy_avail  usermachine: Tue Sep 27 20:31:49 2022
256
0c195c8 1ea0 799c 8ac8 bffc 7186 d26a 0f56 18ae
0c195d0 bd44 2bdc 87aa 4d5e 0989 98ff 35f4 6308
0c195e0 0ec6 0228 c403 d20e 525d e8d3 1749 6223
0c195f0 5d82 dcf6 2d87 5423 496a b7ce 242a 8706
0c19600 c8bf 2938 28d4 a16f 04ff 22bd b3cc c1d7
0c19610 9188 b894 a98a 7d71 ba82 781e 863c 6b87
0c19620 c7f1 acd7 c754 e2c8 1316 df53 84eb 4626
0c19630 bd89 0e29 9fbb 7948 d982 f78e 8cff 2875
0c19640 e0ef d8ea 6657 c258 cd98 9dca efe6 51b1
0c19650 31ba 0711 36db 781a 47fa 921d 5a1e 2b9e
0c19660 5a5a c5d3 e9cf 121e 2d66 f9de 17e4 344e
0c19670 b498 7d55 e317 db43 6a33 e5ef a1ce 641e
0c19680 9905 b778 0efd 56e3 0c58 ab37 436f 39fb
0c19690 98e1 218b 0405 079f f655 25fd ddc7 f138
0c196a0 c4a2 9a45 39d5 a56c 638f c2c7 188b 44ab
0c196b0 f8cf 3d9d 1a16 cb59 aff0 0526 bf8e ed77
0c196c0 0e79 e4af f6f0 d560 47f0 c87e b11b a794
0c196d0 9695 8141 04a6 43a8 a612 53cf 0b86 9314
0c196e0 e9d3 494d 9f1a 9011 d6da 8209 f85b 5a06
0c196f0 b85e 0785 94ab 3223 8e27 8876 6508 84f2
0c19700 b851 e85e eb05 f831 5566 88c2 915f c116
0c19710 aab5 5b13 e8be 49f8 7f65 c835 ed6e 9879
0c19720 d96f 9584 0cf1 4257 6137 3ff8 cd4b 1818
0c19730 3cf1 75db 454f 4b0b 6139 b2a5 cbeb c1ab
0c19740 1dcd 62dd b09e 4ea4 fbc8 8b8e addb 698b
0c19750 8b37 0d34 5d85 db24 3940 2f46 6a6f ccc1
0c19760 938e a2d1 3bcf 9c4f f78d 8f6f a871 44eb
0c19770 54c9 3712 7edc 58d3 3bf7 ac26 3a7b 0c9f
0c19780 8cd2 fecb b4a0 8a86 e047 a57a d099 4c0b
0c19790 6af2 b885 0968 1011 4252 d171 af40 e45b
0c197a0 7539 f738 eac6 f5d0 6152 f995 2c8c 3db2
0c197b0 a832 4dd9 5837 dd57 7b7b e659 4299 7af1
0c197c0 6ecf d63e fec6 4c8f b318 797f 14e8 1273
0c197d0 98c2 2eaa 15a2 1f6c f646 1a3a 35a7 516f
0c197e0 76b3 e58f 081d fe8a f92e 998a 1c69 9368
0c197f0 6c8f 4e8b fec2 9a5c 7db3 7e45 fb1e b3ad
0c19800 c851 3cbc d6c0 ec6e 5c53 sccl a5a8 3bc2
0c19810 073a 0a61 5597 c30f 04f2 8209 3721 d188
0c19820 cfae b52f 585d 25f5 c8e2 e4a2 e988 116f
0c19830 5a88 a9e5 f854 973f 577c 88d7 9dc8 514e
0c19840 afeb 90d1 88de 5e28 db3b 8ccc ee0d 2c9c
^C
[20:31:46] [cost 6.876s] cat /dev/random | hexdump
[20:31:48] [-] >>>

```

Observation :

- On mouse movement we see large change in generate hex dump

Step 2 : Generate random and check quality

Screenshot :

```

[20:47:32] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> head -c 1M /dev/urandom > output.bin
[20:47:40] [cost 0.059s] head -c 1M /dev/urandom > output.bin

[20:47:45] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ent output.bin
Entropy = 7.999817 bits per byte.

Optimum compression would reduce the size
of this 1048576 byte file by 0 percent.

Chi square distribution for 1048576 samples is 265.64, and randomly
would exceed this value 31.06 percent of the times.

Arithmetic mean value of data bytes is 127.4850 (127.5 = random).
Monte Carlo value for Pi is 3.144344880 (error 0.09 percent).
Serial correlation coefficient is 0.003824 (totally uncorrelated = 0.0).
[20:47:46] [cost 0.084s] ent output.bin

```

Observation :

- Here we stats about the random number generated through /dev/urandom

Step 3 : Using dev/urandom in task 1 :

Screenshot :

```
[20:47:31] [cost 43.554s] yay -Syy ent

[20:47:32] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> head -c 1M /dev/urandom > output.bin
[20:47:40] [cost 0.059s] head -c 1M /dev/urandom > output.bin

[20:47:45] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ent output.bin
Entropy = 7.999817 bits per byte.

Optimum compression would reduce the size
of this 1048576 byte file by 0 percent.

Chi square distribution for 1048576 samples is 265.64, and randomly
would exceed this value 31.06 percent of the times.

Arithmetic mean value of data bytes is 127.4850 (127.5 = random).
Monte Carlo value for Pi is 3.144344880 (error 0.09 percent).
Serial correlation coefficient is 0.003824 (totally uncorrelated = 0.0).
[20:47:46] [cost 0.084s] ent output.bin

[20:50:28] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> gcc task1.c
[20:50:32] [cost 0.126s] gcc task1.c

[20:50:32] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
a9581c6cd0f079dcca4eaf636e0c2e2
[20:50:34] [cost 0.048s] ./a.out

[20:50:39] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
45ddd23dd66f20cb8b359f7557b6cdeb
[20:50:40] [cost 0.051s] ./a.out

[20:50:40] [~/github/UE20CS30X-Submissions/CRYPTO/SUBMISSION-3] git:(main*)>>> ./a.out
ce901e7865da2f5087bb4eec6d3205db
[20:50:41] [cost 0.050s] ./a.out
```

Observation :

- Here we see that the generated ranomd number done see to follow a patterns unlike before when we did using timestamps.