

# LAB 3 MANUAL

Name : keerthan pv

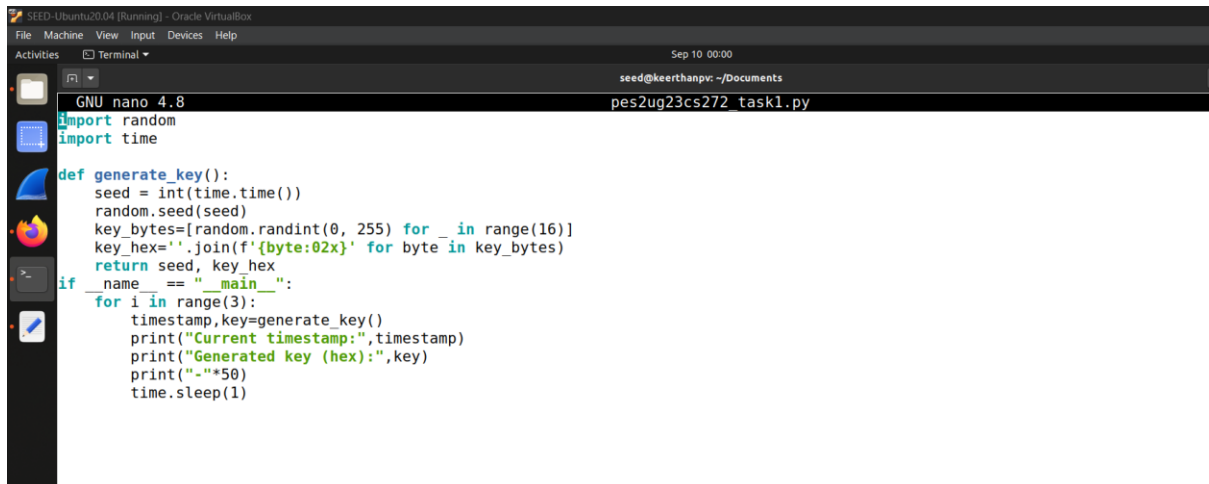
SRN : PES2UG23CS272

In every screenshot make sure your SRN is present

This lab needs to be executed in your VM or Linux machine as /dev/random or /dev/urandom are Unix/Linux-specific device files

## Task 1

- Provide both the completed code and an output screenshot with the corresponding SRN clearly displayed.
- Ensure that all sub-questions are answered comprehensively.



```
GNU nano 4.8
import random
import time

def generate_key():
    seed = int(time.time())
    random.seed(seed)
    key_bytes=[random.randint(0, 255) for _ in range(16)]
    key_hex=''.join(f'{byte:02x}' for byte in key_bytes)
    return seed, key_hex

if __name__ == "__main__":
    for i in range(3):
        timestamp,key=generate_key()
        print("Current timestamp:",timestamp)
        print("Generated key (hex):",key)
        print("-"*50)
        time.sleep(1)
```

```
[09/09/25]seed@keerthanpv:~/Documents$ python3 pes2ug23cs272_task1.py
Current timestamp: 1757476762
Generated key (hex): ec82bf72e56475611e095d26e0e80a48
-----
Current timestamp: 1757476763
Generated key (hex): 01afcfad80cfa608912bb969b14aae84
-----
Current timestamp: 1757476764
Generated key (hex): f593104005717043102127586f057eb4
-----
```

### 1. Why do the keys repeat when generated multiple times close together?

The random number generator is being seeded with the current time in seconds. Since the seed only changes once every second, if you generate two keys within the same second, the program starts from the same seed and

produces the exact same sequence of numbers. That's why the keys sometimes repeat when created too close together.

---

## **2. What is the security risk if an attacker knows the rough time of key generation?**

If someone has a good guess of when the key was generated (for example, the attacker knows you ran the program around 3:15:06 PM), they don't need to try all  $2^{128}$  possible keys. Instead, they can just try all the seeds around that second — maybe a few hundred or thousand guesses at most. This makes the key much easier to crack.

---

## **3. How does seeding with the timestamp waste the 128-bit potential and fail to maximize randomness?**

A true 128-bit key has an enormous number of possibilities — about  $3.4 \times 10^{38}$ . That's practically impossible to brute-force. But when the seed comes from the system time, the possibilities are limited to the number of seconds since 1970 — only around  $2^{32}$  different values. So instead of using the full strength of 128-bit randomness, the actual randomness comes from just a few billion possibilities. That's tiny in comparison, and it makes the keys predictable and insecure.

### **Task 2**

- Complete the given code and provide both the completed code and an output screenshot with the corresponding SRN clearly displayed.
- Ensure that all sub-questions are answered comprehensively.

```
1 import numpy as np
2 from datetime import datetime, timezone
3
4 start_str="2025-09-01 21:08:49"
5 end_str="2025-09-01 23:08:49"
6 start_dt=datetime.strptime(start_str,"%Y-%m-%d %H:%M:%S").replace(tzinfo=timezone.utc)
7 end_dt=datetime.strptime(end_str,"%Y-%m-%d %H:%M:%S").replace(tzinfo=timezone.utc)
8 start_epoch=int(start_dt.timestamp())
9 end_epoch=int(end_dt.timestamp())
10 plaintext_hex="255044462d312e360a25d0d4c5d80a35"
11 ciphertext_hex="1a4fffc708dfcd68e66c262859010f8"
12 plaintext_bytes=bytes.fromhex(plaintext_hex)
13 ciphertext_bytes=bytes.fromhex(ciphertext_hex)
14 for seed in range(start_epoch, end_epoch + 1):
15     bitgen=np.random.MT19937(seed)
16     rng=np.random.Generator(bitgen)
17     key_bytes=rng.integers(0, 256, size=len(plaintext_bytes), dtype=np.uint8).tobytes()
18     xored=bytes(p^k for p,k in zip(plaintext_bytes,key_bytes))
19     if xored==ciphertext_bytes:
20         ts=datetime.fromtimestamp(seed,timezone.utc)
21         print("FOUND!")
22         print("Seed (epoch):",seed)
23         print("Timestamp (UTC):",ts.isoformat())
24         print("Key (hex):",key_bytes.hex())
25         break
26 else:
27     print("No key found in the given window.")
```

```
[09/10/25] seed@keerthanpv:~/Documents$ python3 pes2ug23cs272_task2.py
FOUND!
Seed (epoch): 1756761334
Timestamp (UTC): 2025-09-01T21:15:34+00:00
Key (hex): 3f1fbb8125eee3c0844312b640481acd
[09/10/25] seed@keerthanpv:~/Documents$
```

### Task 3

- Complete the given code and provide both the completed code and an output screenshot with the corresponding SRN clearly displayed.
- Ensure that all sub-questions are answered comprehensively.

#### 1. Which method is usually the slowest, and why doesn't it always act that way?

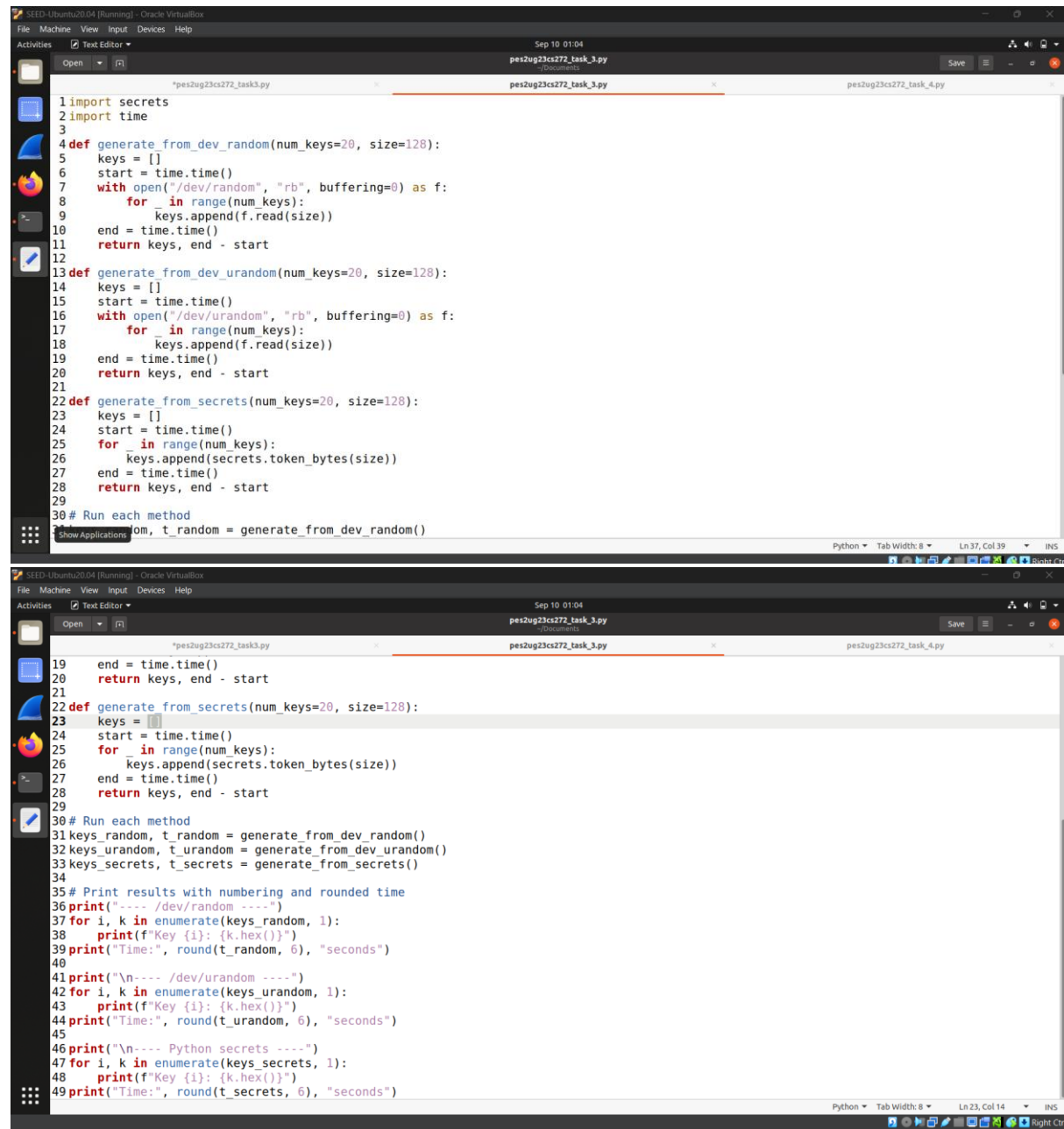
/dev/random is usually the slowest because it can stop (block) and wait for real randomness from things like keyboard presses or disk activity. But if the system already has lots of randomness stored, it won't need to wait, so it can sometimes be just as fast.

#### 2. What's the main difference between /dev/random and /dev/urandom?

- /dev/random waits until it has "fresh" randomness before giving you data.
- /dev/urandom never waits — it uses a strong algorithm to stretch existing randomness so it can always give you output immediately.

### 3. Which method does Python's secrets module use?

The secrets module uses whatever secure random source the operating system provides. On Linux, this is /dev/urandom. On Windows, it uses the system's secure random functions.



```
1 import secrets
2 import time
3
4 def generate_from_dev_random(num_keys=20, size=128):
5     keys = []
6     start = time.time()
7     with open("/dev/random", "rb", buffering=0) as f:
8         for _ in range(num_keys):
9             keys.append(f.read(size))
10    end = time.time()
11    return keys, end - start
12
13 def generate_from_dev_urandom(num_keys=20, size=128):
14     keys = []
15     start = time.time()
16     with open("/dev/urandom", "rb", buffering=0) as f:
17         for _ in range(num_keys):
18             keys.append(f.read(size))
19     end = time.time()
20     return keys, end - start
21
22 def generate_from_secrets(num_keys=20, size=128):
23     keys = []
24     start = time.time()
25     for _ in range(num_keys):
26         keys.append(secrets.token_bytes(size))
27     end = time.time()
28     return keys, end - start
29
30 # Run each method
31 keys_random, t_random = generate_from_dev_random()
32 keys_urandom, t_urandom = generate_from_dev_urandom()
33 keys_secrets, t_secrets = generate_from_secrets()
34
35 # Print results with numbering and rounded time
36 print("\n--- /dev/random ---")
37 for i, k in enumerate(keys_random, 1):
38     print(f"Key {i}: {k.hex()}")
39 print("Time:", round(t_random, 6), "seconds")
40
41 print("\n--- /dev/urandom ---")
42 for i, k in enumerate(keys_urandom, 1):
43     print(f"Key {i}: {k.hex()}")
44 print("Time:", round(t_urandom, 6), "seconds")
45
46 print("\n--- Python secrets ---")
47 for i, k in enumerate(keys_secrets, 1):
48     print(f"Key {i}: {k.hex()}")
49 print("Time:", round(t_secrets, 6), "seconds")
```

```
seed@keerthanpv: ~/Do... x seed@keerthanpv: ~/Do... x seed@keerthanpv: ~/Do... x seed@keerthanpv: ~/Do... x
[09/10/25] seed@keerthanpv: ~/Documents$ touch pes2ug23cs272_task_3.py
[09/10/25] seed@keerthanpv: ~/Documents$ python3 pes2ug23cs272_task_3.py
---- /dev/random ----
Key 1: 696ae9d862de
Key 2: 3ee620ffa890
Key 3: c0f6a11a4758
Key 4: 66d76b0c8d04
Key 5: 876820057bc3
Key 6: 529b4a6d57e2
Key 7: 5529338a4d43
Key 8: 56517cec3dce
Key 9: 965f28e83583
Key 10: 4f28e610d631
Key 11: 162a44ba465c
Key 12: 899ff79b31db
Key 13: 39e46b6c687c
Key 14: 9b0ed86fe743
Key 15: a252f3c6b39f
Key 16: c30b44064903
Key 17: bcaa9f0739f1
Key 18: 798a79d2b988
Key 19: af55e3b8c267
Key 20: 7fae0338fc9a
Time: 171.902025 seconds
```

```
SEED-Ubuntu20.04 (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Sep 10 01:06
seed@keerthanpv: ~/Documents
seed@keerthanpv: ~/Documents
seed@keerthanpv: ~/Documents
seed@keerthanpv: ~/Documents
---- /dev/urandom ----
Key 1: fabae9d2b3ac11f66fed202c7a8a9a9f87a9773962864362235c4ada0125ac77636d1b07e02a853ec93e7f8f58e421e7385d70b72beea10f19f96df6b563db6ab0196
374aecf689b32c41550a80dfc8f2eb427ab23c346371cdc3d592a6c0447973f1fd04fa162c835a543a685e78cf04048c4025175a76151821b3d9b66119
Key 2: e6aa81cc73e28b33cc2a739ad0bbe938f89f289baa83a7c852f76a4611e52448e36ec959111a158fc9a7312a3e934a29e81c611d9613acca1b9b7b30bd39a529ce1
04401187c5427997b5d4f28598c1221b317cadd4f99428294d8b1e2faff435f04035eb5aac37f7677fb74986afdb57e1bd231a0a163523ecb6f2f30110
Key 3: ffbf71af308f87e6e6286d169a53add73a6d8266fec50ba4b7b113c07216843e1f4396fa48c2b91606d6dc3ed72afd2c8865898fcd23040039066401d5409d274e22
f5f23f098237914a52cc7a2b98a8fd09c286e2aa16ad58e38cffe543fb53a4c97801b5db97f7fe266129cd2e288efb5bcbecf3bde9d256acd090f24a
Key 4: 476594936846330eb557ac4267c64e3e0ec2712f39690de6e50ae31a552193ffa2cccd605e790789614bebd73c3615e63fcc016cb9fbed5b1fc8cf618b55c7b02a
399253eb11bec14a32fe2005a1ab1a3055ff6c4be7448a13209ae2a72bcb867b8472c8824a61923d1329ea1bb4c21702c9128c9f79e8e4e95d70635d8f
Key 5: a06020803c5516a4a7d567ccdc22286e2f33e74242191e25a3ff3693c95076032af0f12ac2ea977035ff0afc95bea3b6ba2174868bf2c1f30a2bdc5d9eb029177e723
98c9f46726b92061cfbcb840e0ea4432386f1a65b8deaf347b1d8c234f16ad9a79631ed9a7e5dff39ea846b9c45d6e4f750a17e4cecb093be1e469cd0
Key 6: 455a56f9dc92736bc709afdbdea752040fc837b8da9e013ae48b4fa3575503e9563a2f674bea8353993f9afa259dec0af688ec9556eba853c0d13a21e745fd652698f3
0363f154fa66e88b45f7454c2a5f80d54d388dedd6cacc73a9bc150519cc0deb8b2b32d5ebce56113580f2bf0bbb73ab47be1e6fb95b5d6b9763210
Key 7: 2488ec3b6f960cae0a1351abcafd15346ea950d6df54367fad39908f8ea00fa819c6ee18528bf8c06b22abffcc0f271de8742c7200c32a75aa0ae5f280d6a039cadd
a3aaa19d1ad9e7a98633c89c35a9b970c25c323d406ef5454539db70922ec0556ed1b4ca830b5a0a17e7f8f6c6e990d1047871d3836099df2fd3bd018e
Key 8: f6a15c5ba87e12fa6a5fc9376218478fb70309ec2ea9b4d23a7b6b07f60db649e2521211bc643c460a0bd843ddce50a2e185015a0f66da627f438b00c5af68b94c
7048e1b5a524e2fc8ecfea0ea7ef6f4f99dda23b10135fb94bc68b2dd425e201b82e6d4aada629f6e350f72afa52dad3be0f2af6aac7f6fa169c5d989
Key 9: d4d15fcd437b7e6d43713dd45f106a0d0e1bc1b266e346f8406c3e265c2d3deeb0d8da48ccf2faf5b4f1037fa63d5b7cc2ca0760f63d2b290f9b68a07378a195d8d3
d07d3b29596a14a2346e5e80df269ed83c8641f157f9648f11736e44dc82efdefbe431ealccce8f31760576b51e27b544b425f61f23d018b65acfb8ff
Key 10: 6bc63d9acc1350dfbb8242da52d107450de3765e5dd20615d9fdba51c9a96311bf6c236522b0687aca094fd5310979a3def9c3c781bc651f7c8a008c52c7cf16c1
9620bc5961f912ab5b45043f69188fa21d03ca3b1fc7c1d6b2aa5877db00bf861a9466611b867b8557ca14a08dc5bd636787de1f6a5bdb0fe75af61a
Key 11: 072488704c28d92f723cf57e817ae69004ca735e79b669111d8e868c8f4c0f635882258becb6bd700d42ae9dec2b33bbfb85f8e0e2b1d101abf649ac1567dec29f2c0
7a1afde354fca367ba1e449ed839c0775d004481e5b4ee68573a36fc730660c3629567cc5c389e86f3b67b3c2dc7116aa555fc83e65d664021d998042da
Key 12: 4e2f0a0617aea0256bc3f21fd642eb48f94132d7e2aa5117d354daeb34d7446f5794cb9fd2a683f0fa9b38c02c7fce9a4a406c38c74770c52bb333d4cfdaf9d2aefd
6c112853fbbf6a47f8119cf7b9ca8e892be3825b6a687ede2ca7f72d44311dc55902f5c08f8efa44859917d08f9681d018869440f757683bb8bbaa13f
Key 13: 156f3814f896cf7ca6a3f6549ec1ad94676c2d8e8ec5bee0d38b661302f0bf0f0595b37dae49ab786531f42c72693238292f28cbb652bfe995959d320f60d40ad62
a25d89c2d2e01c46264941caaa3261d43661889e17b51ea381e1990a311833d7a1a6e4a9d618e5b01d925117756af4526812efaf762c96026a1e673d55e
Key 14: d6b0cbe7826788caf98747f38b58922f5086b209bf930047aaec3e78baa84ef30429de4a61f4a24d2ada4a5e1f61d35c0bba9fdbc5860f5b0bb0ead3fede541c12a79d
d1d157fe00f22da45b918c0d961d7eb02625e8b43f162ee6839e6736caab2f0e685d4c29748eb8678a6c172b77960e9ba180dcfbf2da4b0f738e1f6fc08c
Key 15: 7a69c8e525cd66273d943750e481bbe2d42dace0586af615d46ad3fff607cb9ca321c2b49a982abce34c4f02967250d6d0cb99cd2a8442a35f3fdda85fee4f8121079
c08684ce84451263e9cb5de576e60d2745f4077c6dec5ab97af2f67432ed6c0bc078b234dc8fd536245e846b31c69a4e937c42c3a0437495a7436507b
```



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Sep 10 01:06

seed@keerthanhv: ~/Documents

Key 16: b1e2cfd98a6c6d28f62b3af00aa590ee685beea31e9fcbfd52abd9d5325f80f4bdc5f88ea49592dfb7b791af4ff411db5d38d646749399b50d17eb36da42ed9213652
469cc3439866b477bc67f3ee058ed1d330935418b38779ab5153093f2b1876ceal4e9f1e1430364410a58465e00595cee5eb9038c5df1817aac910b853
Key 17: 773389f66b18db32b191d2b98f1df77fddedc9f0d7cefc00de67a5bf05134ad06022b88e4c35e387cd20907d9f9029150daa363f543a12fc597b2c2797e7b7f81944
762615fd973aedc04586af708a43f708e4eb8bea8ab1c057847b5141f74029641211fd6a20752e3bf41327813e07ccl1f0e3fc17514af0405c624176b56
Key 18: d283ae97188c4fee10602d2855b2c816a99a9ad8bb07a6161f2d297205e3b622e389a9af8bc080d4134ad127b3c1f0eb49fb6a9100b323f13742113ab2eb0dfe9
dd7a0ade8a3832bc0d9db6cc2b7e3038a1c92de5497da6bb4330fe1796917b6e7a2d88e44ebf559428caee3ce1ec84344a60263b363e26ec31b22
Key 19: 8beab14de4692c10abf44221f2e806f56be3563d22115c75663fd45c73ae120675d31a02699a494f29243aec62ed705129cb80774ac3a79302b83e53c1dd78291ff1
2c42ab063f88fa0d5fbcce2a1cd87ab8265b1652fe4342b970f7b8bb9cfcb0e9b2dcf885072eb6416ca9a7427fd4fbd5f82e09192cd38c26af2207c
Key 20: faa14f2bd96080e50b2c525b4386fe6125653bd0739c95314bd7c9f239bc6d6f3f127b4fd74842e0a2de6567b951cc69162b2d043530d0e771ff08bd30de6d37041e1
cf498c064d0b38b0c066505c6bfff430e9795acabf15008b1aabe86737b47e06f5b8a12d13237cb74bf4d00fe41c4ce1b47d30c4cceb8593f69ffdfc911
Time: 0.000122 seconds

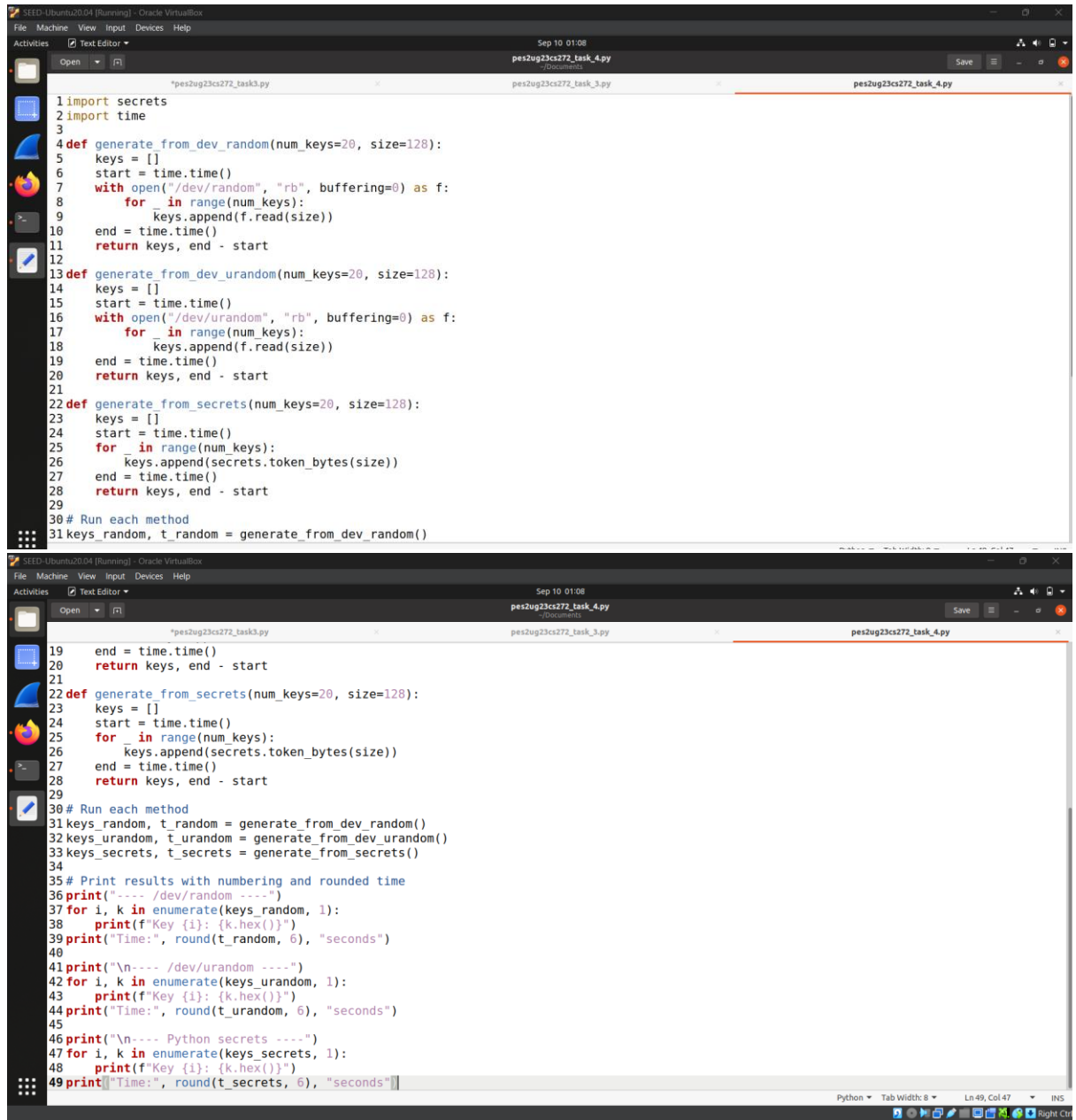
---- Python secrets ----
Key 1: c7720ab5489d12c293b57f378e4943a1ef1fc279c187e004895012485f7a3017fa87c43cbb5b48fda8445c08ba13a5ffda9e21e888437dd316428b1f0c58711df529
685f5aaf33c0877b865255a7be74527c7a173cab3c3ede2f4fb6ad8c0470d004aa4097b805b0b04b140edad5f8426a63392fe009e712916acbfcd8cde
Key 2: d04806b9a553ea4cd9fa747a0ed7131fa319942bf3a4ea0274ad8da79847498e4d39126742b0be481ca6d5adfb7b46b5c5c438440f114005635caace4529bfe2bb15
b431765388fb7ec1a3b3402a1439823f69557a7398350438ab662985d90cafa58f4fe29d6c6514538cc072880c03cbb520503b7a64317dfdb1de724b34
Key 3: 543b29a85bf3227954b534c9e1cbf588fcbcc468d179c0851953b0c796608745c53f9a324d7447e66f470691e9532dbf656349fef4c2280056b42c280530f5317563
280e5f0e23d404f3e928c914e350a3291799266891dfb0e1715b044e6321f687b171268f73e18c661a4ca10373acf22c6b906ff88c57b2df76e06
Key 4: 0a3f83a018d48e94fe4f6eb9341df26271ac38da67fa9c52e38c0c24692c9f2e6b56477843665d08222df835a137c7d9f9aea77b9a92b2d16532f3994b9cb7fbec37e
d61ac2955406f46a35d25a04bd15b1e66c04c17164377a6bf6b9c22ae805c1f561e2ad687831f6b09df1b1bdc1c0166c52270610af837d6e62ee4ee82
Key 5: d7abe800d8bc2bf84cb099524167222a1ad666e46152abfd0d4ada29bb5795653d1fbd186eab7265369ce7586366af2196843c4b0fd7b7bee6cc2eaca8c5a46e0b
1a9b3fa47eba4c1be609de26ef0ba2ba70e2d0fead35d9eb20a738d2de26291228d5ea5b47cc0ca25d9305ce4f67e81595d844a059568339665817dc
Key 6: 1b77bd2307e6059acc0feee1f07d4e8a4fcd8ead70a8e2a4c6aad06490cf17dd9c4513cbe6ca60141b2cb50bae0d7e0c85d7b2e63239c2269f7666a173fa24841e2e
ff9f9eb0frc24de2b1c9d6c1460c8eace2b10477d9ea87173f7563cda9e36b161269bda7fed4490b0e046506a35d31be8b225cb833ae7df843709711
Key 7: 6ce86ee81b9501f55e852ad9a6562748f7e4feddd34f9bb4f2c6b3c25f69dc122caf232e0530ff0173790a32ef72c2f482cd96f729b2a0f1f6beb117613ce6becc7
918461b0f0064f6a35cde454118099bacbb5ab80e9e1bc1180808221029b4ad2a5ad2e149ae7c42fe21bb2ec95adb8929ab1016e38cce7ab11a59fac35b
Key 8: d564912f22c8cfba6e00b634e20e017f116b660af326d17364ad4b88e22c78cae9254cd83f930c344bf50d36990e91df112d1b60a61066f06caa3f2e89ed40de015c
db08211e42c42b0b36f82ec7f21d0b032fb906f362d76932f5c53f25cd9dbff33363609b06cee4aa5b30da75a0e4dd3a00cf77c2153f6e74961a66ef4
Key 9: 2ffc4ad219bd3b799a27f82d5865eb32fd47798d152845fb77f208424b93043bad9c1b14baeb04217d7ff8ef464740a1cc7fb710639395a458dbdf243cab666391f32fa
9a8d2487c94f443406820d0bd168738e34bf50a19c7bb46226c4de36924a1971eb60c33b5837988902ed76f4fcc990469eadc3489a951bcb1bd54b04
Key 10: 5f1886d80d0081cfa58fdb36e1a5ffab09f73ef2875447fd654f5138b131b32d4e564fe25fd9ba58903b11f892000d755f466ff2ea46f57ae746770cb30811c96c83
e112d2b12db474925911bd45a2b2bf6d61c1562fa8bcd2be239ade366f7cdfcb883310275e996f9935c1805cda66a6ebbb1f57384163140454f001e6
Key 11: 3c23ad97805dc8033d8e649737ad5f8de719aa75b6a19ff666343ae5a01501ff04b97c795fa73adbe12d29d208861970498fffd88eae1f116087e8937645427476
79e26d80bb18c43521d2eb8d6c661f69bd3ba1dd1ac855f79b1e2bd538df8c2721614269a6b449814a9dbdd828c24a67ac212736fa183c78bfe4f50b9
Key 12: 5b9d0b84adc558744e363a7ba8838d7bc69b0180a4c0a305ec3c041f72df535f76794c5dc51ed0899c72ee27f1f0710f55d87c64e71d29a43305f3a33e2c922625391
09d0bb3b11408f71d40f73463ed1bfff48d7088f5d8a1aee5999482e484ba3099807233f14552a8472817d099e746c81cf17ad18870c08a873a1bcb6d028
Key 13: a576c2adfcab145da920d7f218b8ae1e95aaebac512b09f7c216c89e9ae56ba7f4d352e4d763be7835890002a6392f39810df33518d32505c957c82f036b0bbe9cb5c9
56bb4299927c9b68994534ea8c833691a25fbd7639f625053b037b5cc5a57ae7351c86f356bd2779ceda634c26f88bead01532732545c6af248235886
Key 14: 6442ce2154bc351e49b9e684e1313d9f0ef72b04246d99626ad41928b5dbd7d6dddb43ebd88ccf01455eb10e1b4fcc2e2c420bd03918e8de58821ee956e31400c94
9943620b09566516fe32ee69a1845e0fabb9005c516d64634ff01ca141691c3ce30b9bc75c8de6dd956dc85203a6910efed6df44cc3025fc656be3cd
Key 15: 0479c9ef0bbf0361eddb71185bd338a2d6702301772fdadbaae5567939647836bb5a8b83abd419656cd5f26df7735d244730259da9f544abee48b939b0dfe8eb3202
7ea8e5a0dc9c91486b594cdcb65c391ef727f7e7a51ef04d5f74f88879bd806627b9555ff6c6815cdacbebf8f884945313e2d28c54f699e5c784ca387a3
Key 16: 12d9f373665e29d552f228e89dd3cb90a46ee73c148cca5c28f442ee00026fc668dfec579b62aa376ec255cf3c8f849a53cf7d45c19cc68580b1c71e273f091ec
e76c9922d1cf572942ee0700b65aafbb1782335a531a40905af363c49769ba8066da46c0a6aa46375d317994577fd169a50808477b207172e05599f46
Key 17: ece9eb5f41c1f91aa815e87eb15bde999de2f21fe7a8e72889f3e35954ac16550f7f6b1ac7fde90f43f68ed7d235fa815eb6954da20b55460331482423be47331c
97eed17ee1b64939776acd4a78b673aad41e529f0abbcc7e2f58bb425b88f8b34fa9e22b5bac990c3dd40031c20d6eadf7921e605296a5511584d7c
Key 18: 71dc5907b6b59fcdfe0daa4789f90ae1edc0990eac3261cfd4ef087b725d4ecba9a96fe0b673d65f7c2cf81d6a62206d526ba8cf796c7893db97b394399c9022b73
2bacf59abc3e3e8a7125a0ea607d48067b666b21c46a02425bb1a885191d164e6d1ceb21ab141f54433389165137e35e0f457f75c03e9aef0729bc9f8
Key 19: 7461337a92625cb085fb64751a8b47b3984e77c8318ed71347146e5e0d97e63a53e6c27561219ca83b2f57c0e6f827ea97d2875a280e2671065c25b344df51d638761
c5be2d9274ee7b26c6c375cb6d04a41f88a281b007049070f7055ae1420d82ee695985f427accfc0160359a137556492526293b8fe72a29e86a92a517
Key 20: c5f962ce8a5f6452d474bf7f7efb4a06d8b474b53e64aed3d78f077425c1195ca299d78d35d4a69067b9d1d8e8b77748f9758db75ec98360779a7da50adba702739
051296a02101e28bc816fcd01a60e3054b6f8d8a26c77aeab192bfa3c996d06fd1af75bccf6769005109f12ffff471443f1e0e9c0c70e157c90c1d20cc
Time: 8.2e-05 seconds
[09/10/25]seed@keerthanhv:~/Documents$
```

## 4. When would you use /dev/random instead of /dev/urandom?

You'd only use /dev/random when you need the very best randomness possible, like creating a very long-term encryption key. For most normal use cases, /dev/urandom or Python's secrets is better, because they're secure enough and don't get stuck waiting.

## Task4

- Complete the given code and provide both the completed code and an output screenshot with the corresponding SRN clearly displayed.
- Ensure that all sub-questions are answered comprehensively.



The image shows two screenshots of a code editor window, likely VS Code, running on a virtual machine. The top screenshot shows the first part of the code, and the bottom screenshot shows the second part.

**Top Screenshot Code:**

```

1 import secrets
2 import time
3
4 def generate_from_dev_random(num_keys=20, size=128):
5     keys = []
6     start = time.time()
7     with open("/dev/random", "rb", buffering=0) as f:
8         for _ in range(num_keys):
9             keys.append(f.read(size))
10    end = time.time()
11    return keys, end - start
12
13 def generate_from_dev_urandom(num_keys=20, size=128):
14     keys = []
15     start = time.time()
16     with open("/dev/urandom", "rb", buffering=0) as f:
17         for _ in range(num_keys):
18             keys.append(f.read(size))
19     end = time.time()
20     return keys, end - start
21
22 def generate_from_secrets(num_keys=20, size=128):
23     keys = []
24     start = time.time()
25     for _ in range(num_keys):
26         keys.append(secrets.token_bytes(size))
27     end = time.time()
28     return keys, end - start
29
30 # Run each method
31 keys_random, t_random = generate_from_dev_random()

```

**Bottom Screenshot Code:**

```

19 end = time.time()
20 return keys, end - start
21
22 def generate_from_secrets(num_keys=20, size=128):
23     keys = []
24     start = time.time()
25     for _ in range(num_keys):
26         keys.append(secrets.token_bytes(size))
27     end = time.time()
28     return keys, end - start
29
30 # Run each method
31 keys_random, t_random = generate_from_dev_random()
32 keys_urandom, t_urandom = generate_from_dev_urandom()
33 keys_secrets, t_secrets = generate_from_secrets()
34
35 # Print results with numbering and rounded time
36 print("\n--- /dev/random ---")
37 for i, k in enumerate(keys_random, 1):
38     print(f"Key {i}: {k.hex()}")
39 print("Time:", round(t_random, 6), "seconds")
40
41 print("\n--- /dev/urandom ---")
42 for i, k in enumerate(keys_urandom, 1):
43     print(f"Key {i}: {k.hex()}")
44 print("Time:", round(t_urandom, 6), "seconds")
45
46 print("\n--- Python secrets ---")
47 for i, k in enumerate(keys_secrets, 1):
48     print(f"Key {i}: {k.hex()}")
49 print("Time:", round(t_secrets, 6), "seconds")

```



```
SEED-Ubuntu0.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Sep 10 01:08
seed@keerthanpr: ~/Documents
seed@keerthanpr: ~/Documents
seed@keerthanpr: ~/Documents
seed@keerthanpr: ~/Documents

[09/10/25]seed@keerthanpr:~/Documents$ python3 pes2ug23cs272_task_4.py
---- /dev/random ----
Key 1: cfc4fb7e8fa
Key 2: d840f787b97f
Key 3: 3c0f36b29dde
Key 4: f75c6112d952
Key 5: 6cc67350ddfd
Key 6: 356aef7ed3fa
Key 7: b68e4e063814
Key 8: c8de98aec206
Key 9: 2c13c05b2bc3
Key 10: a92f4c23b86c
Key 11: 5cc399027630
Key 12: eb7dd72b3b41
Key 13: 7e44987ae749
Key 14: f7c3f7a5615f
Key 15: 796b8a2548a3
Key 16: 79f8c19b4a01
Key 17: 69681b818e35
Key 18: 253845d57b58
Key 19: 471ad8fd0590
Key 20: bafb3cde01d0
Time: 324.083949 seconds

---- /dev/urandom ----
Key 1: e159601601f434f1f09c2dd48b1b15ef8d1243758543d00e87e8140c439823037efac719ca19444d60e8202c977ab7434bb84c7628ef954d09ad52d466c5bba6213284
828a30b6391aaebaa6887c331ed09fealeaa107461d07a3a2f5b7007d5d6962c0be282854a9a0ee16f80618462a89212655b5a0e867978db546abc0c3f
Key 2: 1965368d79ab2fdb4638eeeb81192f6f06a8cbfc5d862826d770d7d14bdc3a979d1b66fb59c72602aa35a14e62c0942cad9cc362398b2ae77222ff7ec6136aaa517a
e49dfabce8049e02e499292c979a032d236fc3382064a1d2dfc8038fb9ab6f20dde826d15c00364e70ef4d03080aef922138a1b1d57e881b6693e67069f
Key 3: f2867fb5961f505ee3b4b1f30e00942f93210f0973cb1adb347fc1f2c5e20da3ccff8a94782d904060b4506e8ce04692c0b09538da35bd9514801d991c8d5aaf6437
276e5b3ef5f444af4e584b2786b60f781f961a36d9f87cc725c7bfc55e45b50e65a75b47c997f0f11ea96caaf9a85dc5c3ed97e5f6566b0911a7b7f302
Key 4: 27f08e15faadd188a16c92ebf58d40f8e80c5c7a1b9052405572708e475de208c8ebcc1bce5c3d49fdd42254a231d87dce482c3a18189e64bee4f6bd8aa5bb3c108
```

```
SEED-Ubuntu0.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Sep 10 01:09
seed@keerthanpr: ~/Documents
seed@keerthanpr: ~/Documents
seed@keerthanpr: ~/Documents
seed@keerthanpr: ~/Documents

cb091d7192a564ee2a5e85bcd16ccf274ef93bab6f0cce39adda188e2049566a461007cd4929d314b97e67e9d3eb526d65d2bc2a93003c361ac0fead
Key 5: 602de9e2337d6214eac65762b4602e6063b246b32b5c7694907d08f34fa73f5d9bdc2c73eb5e614754024471779194668f77cec2bb8f794be6c9f92127549ac85cbd6d
f36534aa61ad5c82b421ccb33db04f2418b832342fe633c78749b6c2b00e0ee510590a602ede6f4dd8b31f9a4f6146d42e0ea67b91fc356233f84bb0f4
Key 6: 67f6b70ca763f4e748cdfa90e91d1ba73478729529d47705e8da58dc16a07569be918bf682bdc3de231447d44c45a6749015a84d8f73cc30e422731823deb6f549f5
412fecb8fda0aeb5e4079c86b5178332a21bd2019344713a2c130d5866976a69b9fcc659f64dda42c15338ef5db2761e11584fa9fdd63bf9290b724a1
Key 7: 1c990703b7abea59bbaf8b373888e12fd25cf27bf19b98b3af34f5ce9b4668b7fbd41b6defb8c5ef14f08e69140cccb193bd43d0559525a2d2973afad6edde51b23193
b3af39512b09ecd3e9b1b51bf5f81abfc330e20ff26a9359c11b7827be62367d8bce1f3a31fc322ba0789daa7bc5dca47ea4e77c126ef679ab87fb683
Key 8: b6fae0254f4efef46e39bae333214544b34d17ead86133518448c4d38f61da1664030ac4c168e5c178fcd41556c8108cef9d05216a951dc88ca935f0812569fa4f2b29
3f10975d8846f5864c07c8f5041fab45c4310dccc96ff6bb4a6630f499ad256d80492b59ae89592f0b7eddb81523384e3ac9b1d4e77ad44bcefd53cae68
Key 9: 207f4706553d2a810bb3dac7685510a6a01a21a3368850279042aeb7e0531df6485cc53c25da172f42423d530254c004cc92927dc8d549edba8d217ceeb44b0266473
b3985f9a1b84805abb3fba7f54884709aceb1d3ab9311f3e7ceecfc14df7a0d12ecca90c674465f0c0b1c5f34b070753ef09c16165232703a25c454777
Key 10: a469a6be27c5f4ac680d98f4f1fec5286052a5758c3790bdc981576b650942eb2348eed288760fe37ca2f5f9903b0ada3d3e96f683ca4f0a1aeecc10a69c529f07f99
3bb5d0288da634644f921b316999bb98d59ff9e14374275b324bab8907df0fd1328d7444af36882ad1181e1241e00b83a0e96b915b164fac6242d1af3f
Key 11: 894c5048932bb434cf339cfd7e4a1c3018e5c732e2937cfc11a9774dc94e047f5bf615e76b252e1dde8fca2a3093e6246c0f7182e7608015ada8a404456ecf40ad424c
1e77d77cf84796d016afdf8ec4a5f34cb7959fb24887695eed3992b856708066ea4ce8560a4987c6e2a8c1f07feafbc6386322988d80c3713e5ecc5bee9
Key 12: 6606721571f9a9db331acccdd072f5584e7afa1ce9cdf0d38e6e55c5bca6705bb40999a0a87262687d2acbf6868fcaf7db0053a429e3b5517a0af547b328cb328f
6fde09d29229f6dc831b6264d625ac34da0ae85820750779a067b874124df168fe0fe29d3899a2c01a33dca5e4e10587025af203b8ec3f6b047bb5a68c3
Key 13: 5f2f606584a2e4fde3fb65be67ea3bdf1257ad76f7524987eb8fefa36d16075afcfbf077c9219c9f502ddcc322099dbd7024f27e10d16329350f29fd5f184609352
53d60dbf68a5063219ee1cd8f090de1e456bf5ededa63913a24bdc8d44fd581be4a1800a4bfc4c0564640a6f7b38079b9bc0c568814d5ae1003d86da8bb
Key 14: 40be64636ebbe247aaf2af00028d0afe62b19204c4717e34d3dc24da22619a2d17429c7fad992914a4b67121008fee5adcd1b92bd1095cd3a9166df26f4f82eaa
80693597e45aa03f4940fe58fde667021f5f18951334dc3c441f521304a54c09343eecd14b9a984608c1482e2548bd45772ca4b0ff8bb1370485b45090ba
Key 15: feb3cf548e96baf1d4b5c258acacd2401b6acf3e1c06ecf8aeb75b5e463fc92c0e34e2fcd1258baced82376735098bb09994a32049f809188f257941e0c004e461f4a
2a0d3a0b4e0f59fc0ab9bb301eeaca52cb0a8557c879e00b3c6fb2028cf1b8d6059ebd6e313204ac26bf5fc420daacf3b0dc9d13f0e57ad52b62502dcd8
Key 16: b78a39204eadc45beed20bb114efe50ec4b333411f85d6f28e70cb74890fed29c5d02588bd95f84a07f94c9b428075abc4e60c5fa29e52ea65dac465dbcab71840
bff2c263067584aed4a1d81375fe46ab32afd9d87fc4f8d74a344ee7717c4e22b99fd2ebbc7b42738a472f543c53d173fee45dbbaec93c21ef3e116c01
Key 17: 2b97402da7a4aa31b85cb9ed064b8eb4d0c60ce2f17f0afe886eb06a3c43404346a701363b92493edd1342795496d476d59d7315a1b0e26a5504dab6dc7f350e1d
3f59e903195eb3f4ae3e2da7a55ab163b9047d58c9f80223fcf45f8c37d8e4adc71bc0269b745fddfa3366a3a96a5262b26d517865a917d2a9cc4514a9d7
Key 18: 29160c5dfffe20c0e131765d71f5ea245e08f5942ecd827eb0b3097f8a4aa2bac7f3f2d4de28f451331a4b20690f684df160d9cc496626d6d86f0b0ef686d098230a9
b4a442f6b74ca9f012b1262ca0d0e72a07344b122762ede59df8e4c448c7d78deaf456011578c4eca1344335c710eb2b70b8888b960d74f5f7fe0e4d31e3
Key 19: d944a4b2ac89b13faf4c1fac1955b5b5145133a3703467806a6cf8620f9c06a0416e66c6dbcc59c87fc1469e8f3234550022966717c27dc303f8c4a416b233f90e71
d5e62d7011f2375016e7ad2d0c2a81d0fed5cfea37a336cc2a5cac1cb4fde553bc99646b35b06f903cfed0cbe4c9175d3e79d582d1f21b11b71afb3
Key 20: 6b98cb81e20e99abdc1fe89a8b3637649f1635a0d8f8120f8993b2cc13a9fe7bdf07892180b23d51d52b096deade6934ed1ff5d3b1b1b4eb3eecd41b88768a2d714f506
```



## 1. Which sources have the highest entropy?

/dev/random, /dev/urandom, and Python's secrets all give very high entropy, almost the maximum (~8 bits/byte). The test patterns and all-zeros are much lower.

## 2. Why is /dev/urandom faster but still as good?

/dev/random works for real hardware noise, so it's slow. /dev/urandom uses a secure algorithm to stretch that noise into more random data, so it's fast but still gives nearly the same randomness.