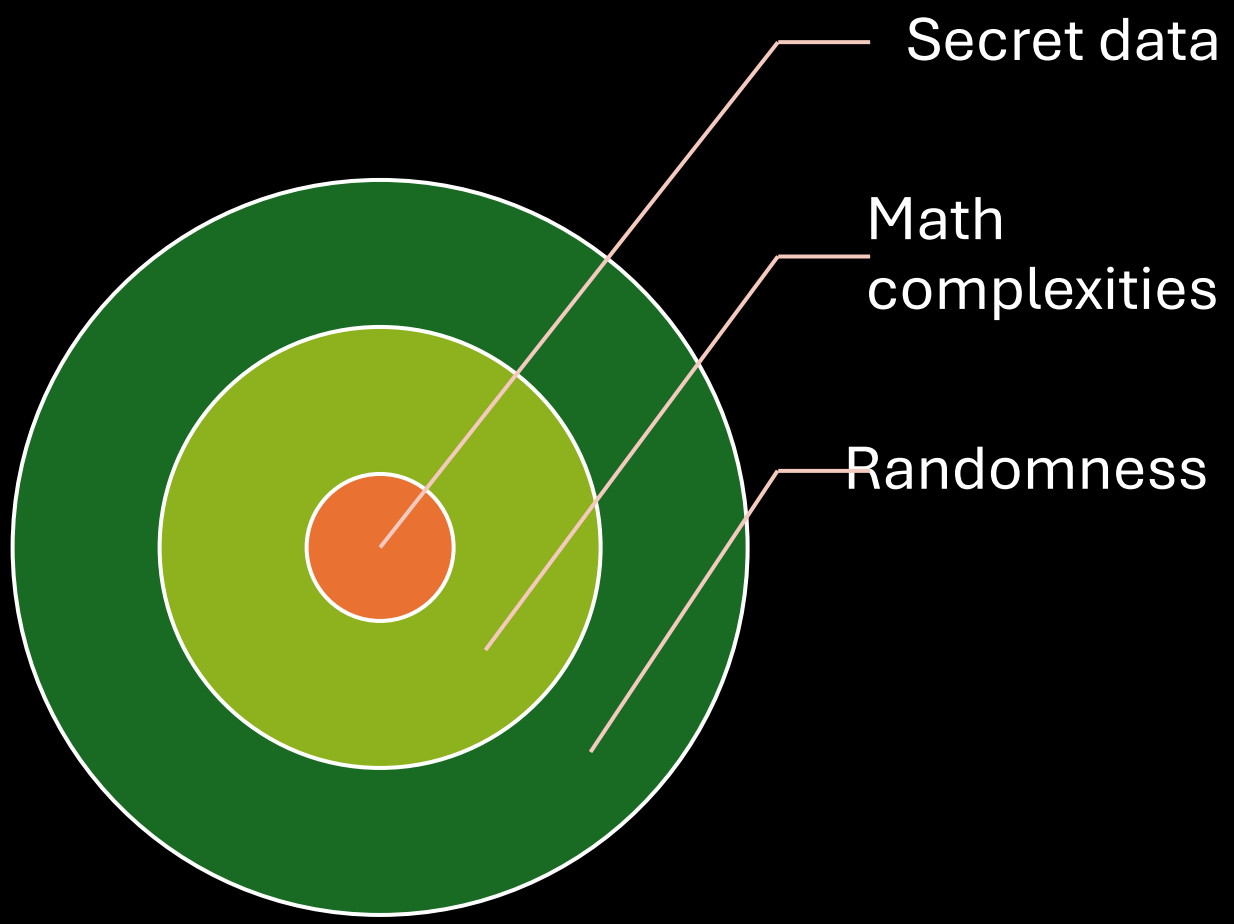
CS405 – Computer Security

Lab02 –Randomness and DES

Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Introduction

- Randomness is found everywhere in cryptography:
  - Generation of secret keys,
  - In encryption schemes,
  - In attacks on cryptosystems.
- Randomness is what makes cryptography unpredictable.
  - Thus, it is secure.



Introduction

Which one is random?

11010110

00000000

Introduction

Which one is random?

Both are random; they have the same chance to be generated $1/256$

11010110

00000000

Introduction

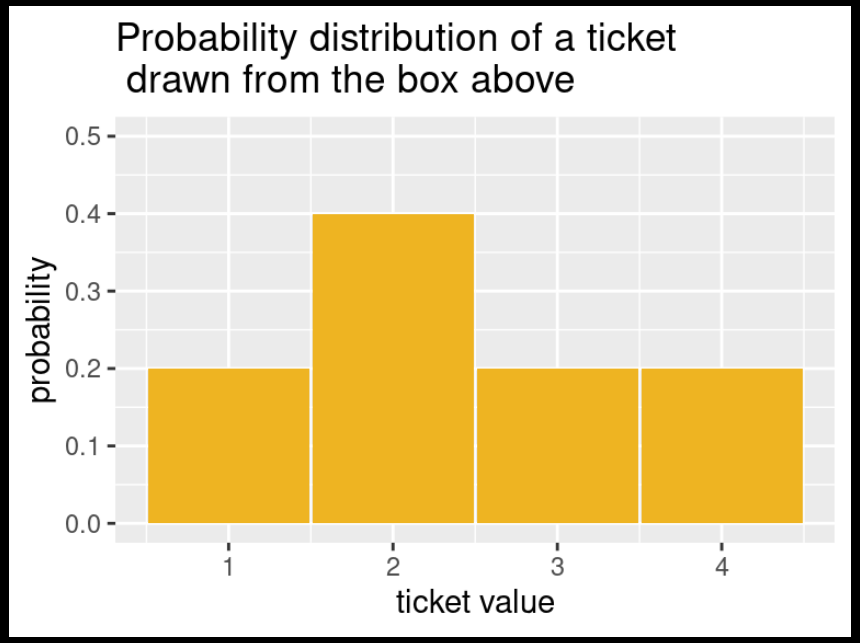
- Two types of errors people often make when identifying randomness:

- In crypto, non-randomness is often synonymous with insecurity.

Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Randomness as a Probability Distribution

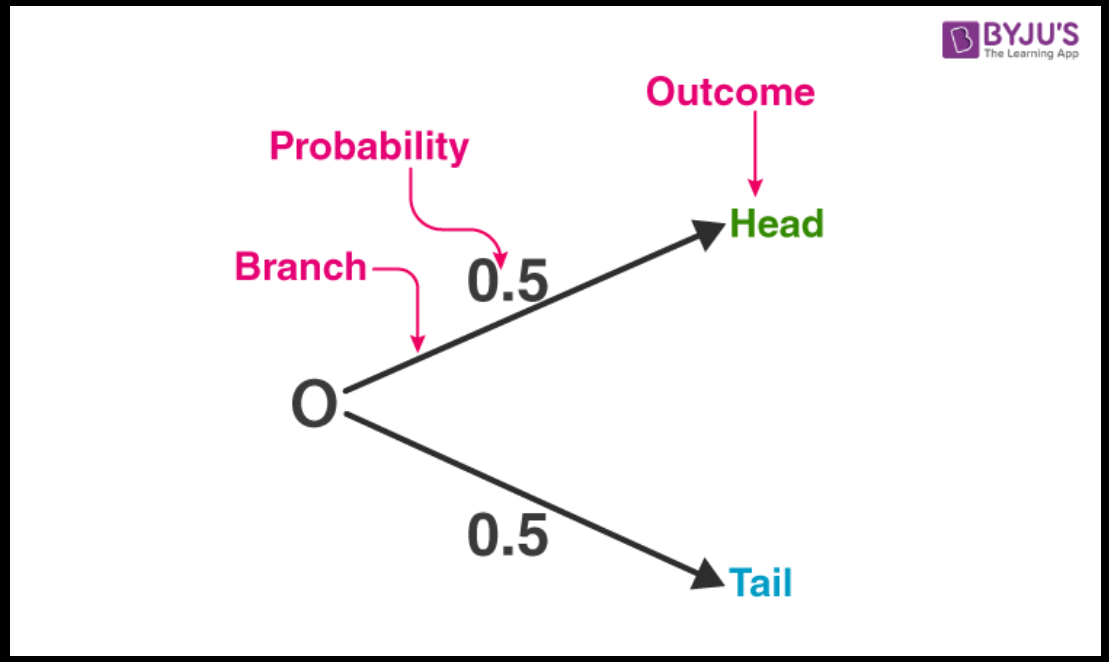
- Probability distributions are used to define a randomized process.



- The distribution lists the outcomes of a randomized process where each outcome is assigned a probability.

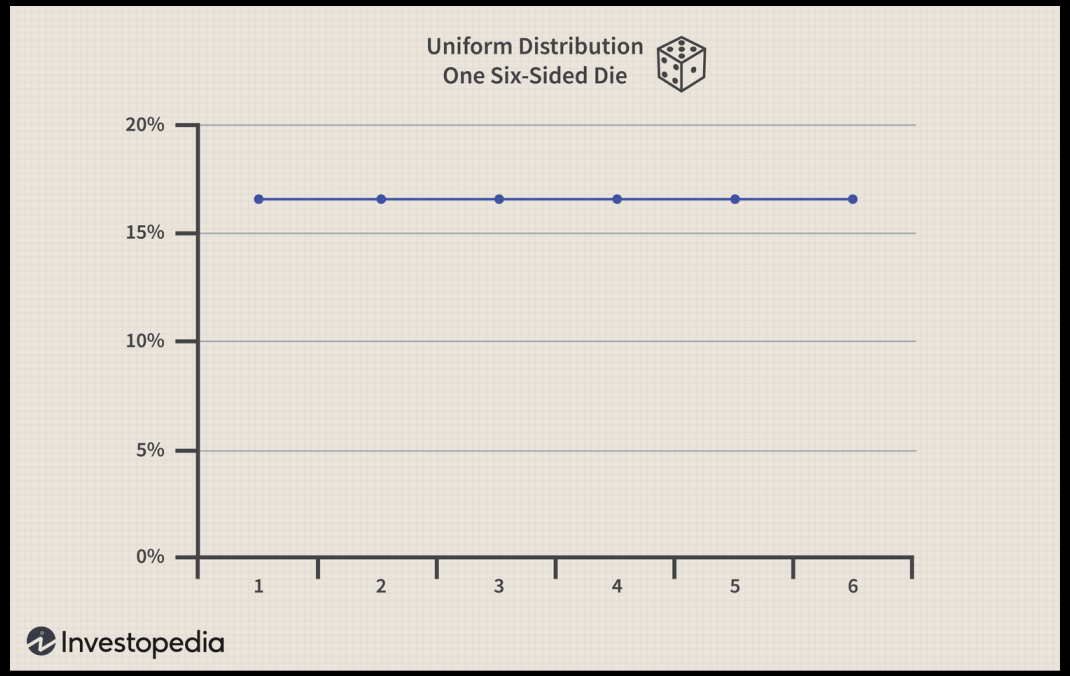
Randomness as a Probability Distribution

- A probability distribution must include all possible outcomes.
- The sum of all probabilities is 1.
  - Specifically, if there are N possible events, there are N probabilities $p_1, p_2, \ldots, p_N$ with $p_1 + p_2 + \ldots + p_N = 1$



Randomness as a Probability Distribution

- A uniform distribution occurs when all probabilities in the distribution are equal.
  - All outcomes are equally likely to occur.
  - If there are N events, then each event has probability $1/N$.

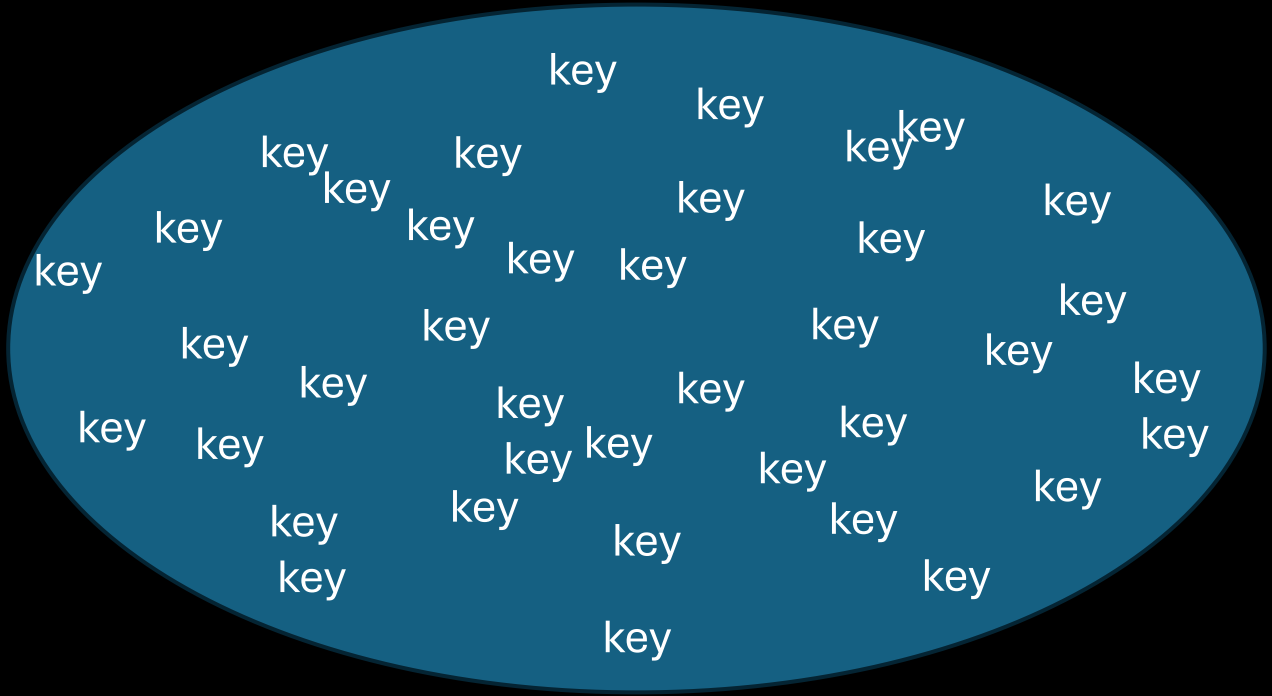


Randomness as a Probability Distribution

- Assume that I have a space of keys, where each key is 128 bits.
- Then, how many keys do I have in the space?

Randomness as a Probability Distribution

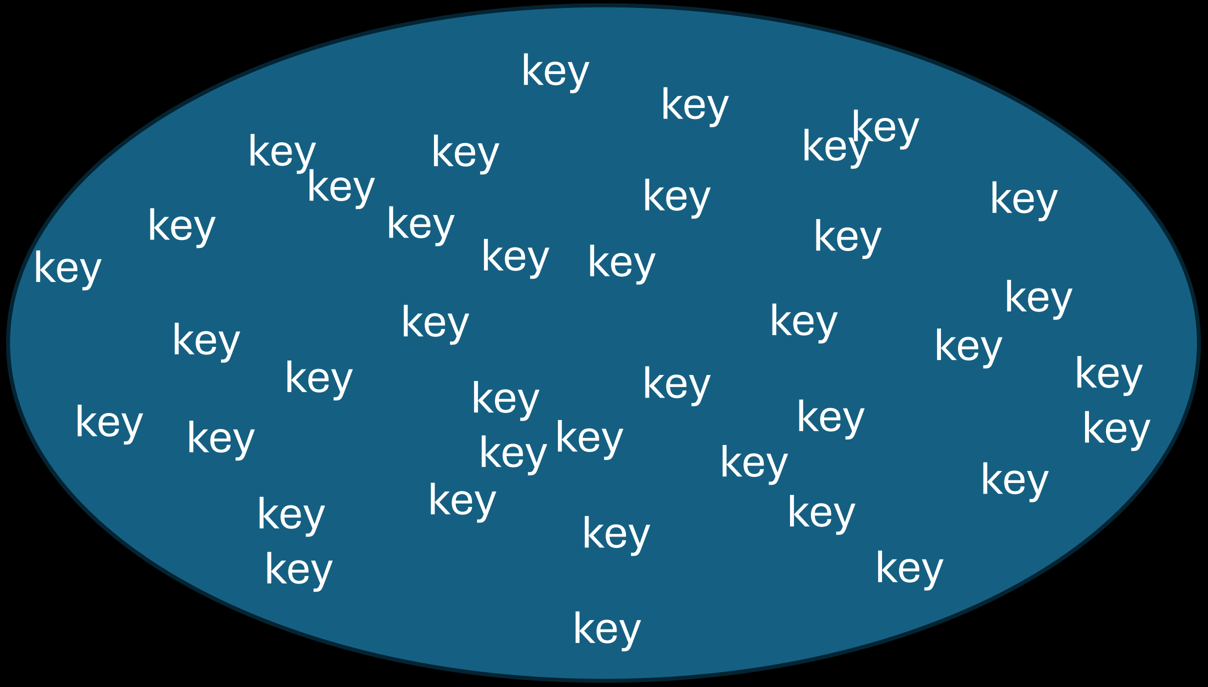
- Assume that I have a space of keys, where each key is 128 bits.
- Then, how many keys do I have in the space?
- 2^{128} keys, which is a very large number.



Randomness as a Probability Distribution

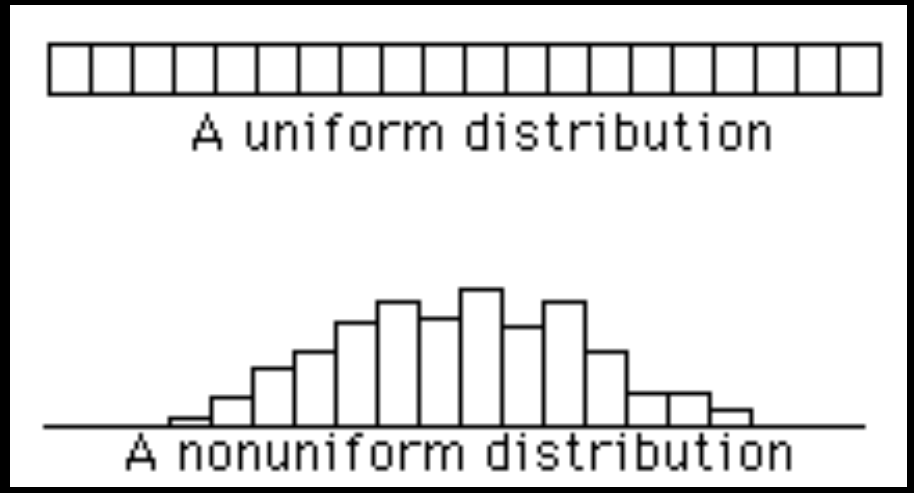
- Assume that I have a space of keys, where each key is 128 bits.
- Then, how many keys do I have in the space?
- 2^{128} keys, which is a very large number.

- So, to pick a random key, according to a uniform distribution, then each key has the probability $\frac{1}{2^{128}}$



Randomness as a Probability Distribution

- When a distribution is non-uniform, probabilities aren't all equal.

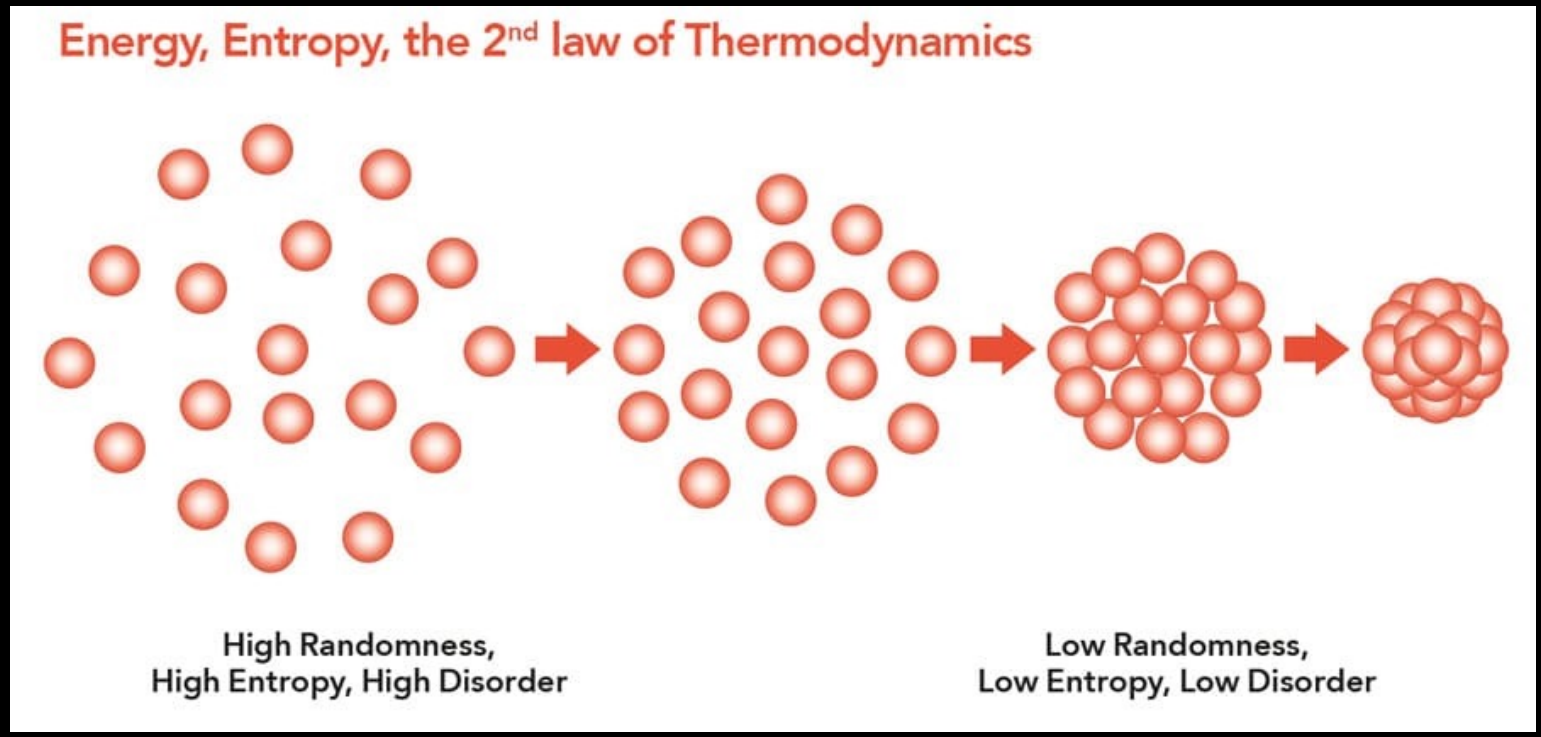


- For example, a coin toss with a non-uniform distribution is said to be biased and may yield heads with probability 1/4 and tails with probability 3/4.

Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Entropy: A Measure of Uncertainty

- Entropy is the measure of uncertainty, or randomness in a system.
  - The higher the entropy, the less certainty found.
  - The higher the entropy, the more randomness found.



Entropy: A Measure of Uncertainty

- If your distribution consists of probabilities $p_1, p_2, \ldots, p_N$, then its entropy is

$$-p_1 \times \log_2(p_1) - p_2 \times \log_2(p_2) - \ \ldots - p_n \times \log_2(p_n)$$

Entropy: A Measure of Uncertainty

- Random 128-bit keys produced using a uniform distribution have the following entropy:

$$2^{128} \times (-2^{-128} \times \log_2(2^{-128})) = -\log_2 2^{-128} = 128\ bits$$

  - Note that: $\frac{1}{x^y} = x^{-y}$

Entropy: A Measure of Uncertainty

- Entropy is maximized when the distribution is uniform.
  - Because a uniform distribution maximizes uncertainty: no outcome is more likely than the others.
  - Therefore, n-bit values can't have more than n bits of entropy.

- When the distribution is not uniform, entropy is lower.

Entropy: A Measure of Uncertainty

- The entropy of a fair toss is the following:

$$-\left(\frac{1}{2}\right)\times\log_2\left(\frac{1}{2}\right)-\left(\frac{1}{2}\right)\times\log_2\left(\frac{1}{2}\right)=\frac{1}{2}+\frac{1}{2}=1\ bit$$

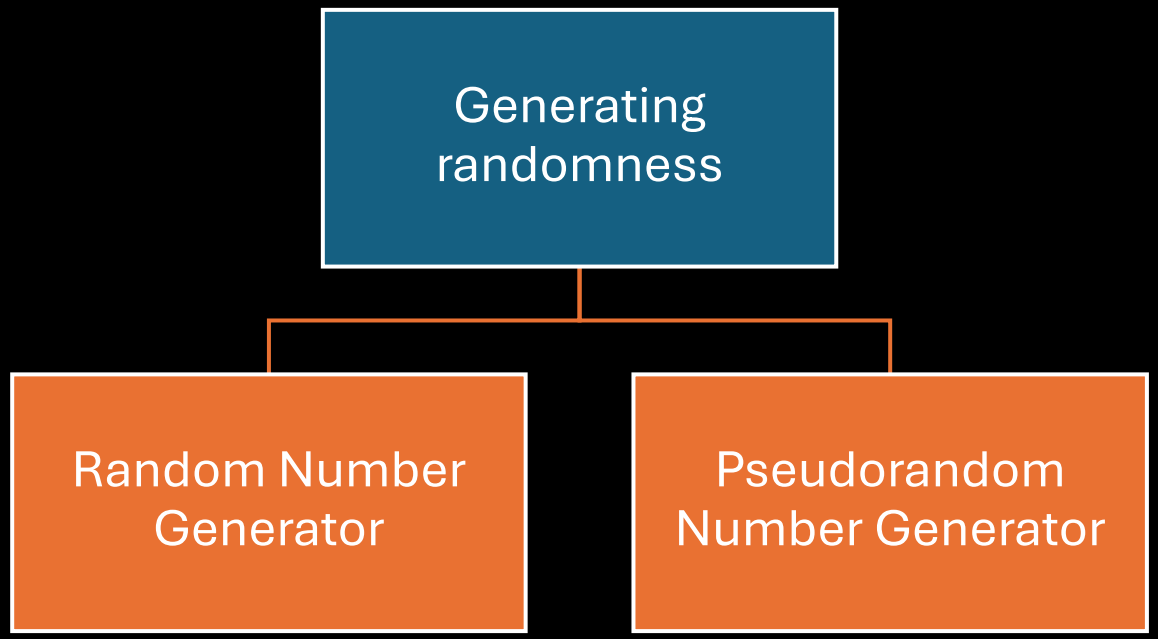
- What if one side of the coin has a higher probability? Say heads has a probability of 1/4 and tails 3/4.
- The entropy of such a biased toss is this:

$$-\left(\frac{3}{4}\right)\times\log_2\left(\frac{3}{4}\right)-\left(\frac{1}{4}\right)\times\log_2\left(\frac{1}{4}\right)\approx 0.81$$

Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Random Number Generators and Pseudorandom Number Generators

- Cryptosystems need randomness to be secure and therefore need a component from which to get their randomness.



A source of uncertainty, or a source of entropy.

A cryptographic algorithm to produce high-quality random bits from the source of entropy.

Random Number Generators and Pseudorandom Number Generators

- RNGs can be software or hardware components that leverage entropy to produce unpredictable bits in a digital system.
- An RNG might directly sample bits from:
  - measurements of temperature,
  - acoustic noise,
  - air turbulence,
  - or electrical static.

- Unfortunately, such analog entropy sources aren't always available, and their entropy is often difficult to estimate.

Random Number Generators and Pseudorandom Number Generators

- RNGs can harvest the entropy in a operating system by drawing from:
  - Attached sensors,
  - I/O devices,
  - network or disk activity,
  - System logs,
  - running processes,
  - and user activities such as key presses and mouse.
- Such system- and human-generated activities can be a good source of entropy.
  - But they can be fragile and manipulated by an attacker.
  - They’re slow to yield random bits.

Random Number Generators and Pseudorandom Number Generators

- Pseudorandom number generators (PRNGs) produce many artificial random bits from a few true random bits.

- For example, the RNG will generate few random bits.
- But the PRNG will return pseudorandom bits based on the based on the RNG.

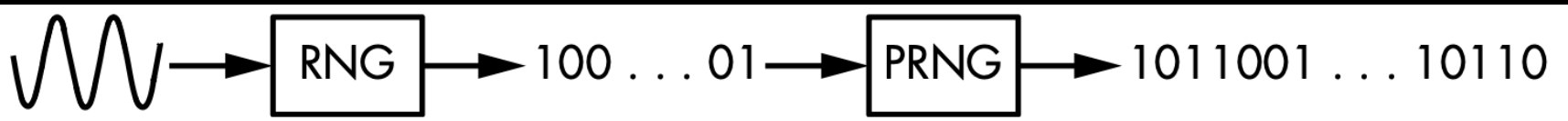


Figure 2-1: RNGs produce few unreliable bits from analog sources, whereas PRNGs expand those bits to a long stream of reliable bits.

Random Number Generators and Pseudorandom Number Generators

How PRNG works?

1. A PRNG receives random bits from an RNG and uses them to update the **entropy pool**.
   - Entropy pool is a large memory buffer.

Random Number Generators and Pseudorandom Number Generators

How PRNG works?

1. A PRNG receives random bits from an RNG and uses them to update the **entropy pool**.
   - Entropy pool is a large memory buffer.
2. The PRNG runs a **DRBG** to expand bits from the entropy pool into longer sequence.
   - The DRBG is deterministic, not randomized: given one input you will always get the same output.

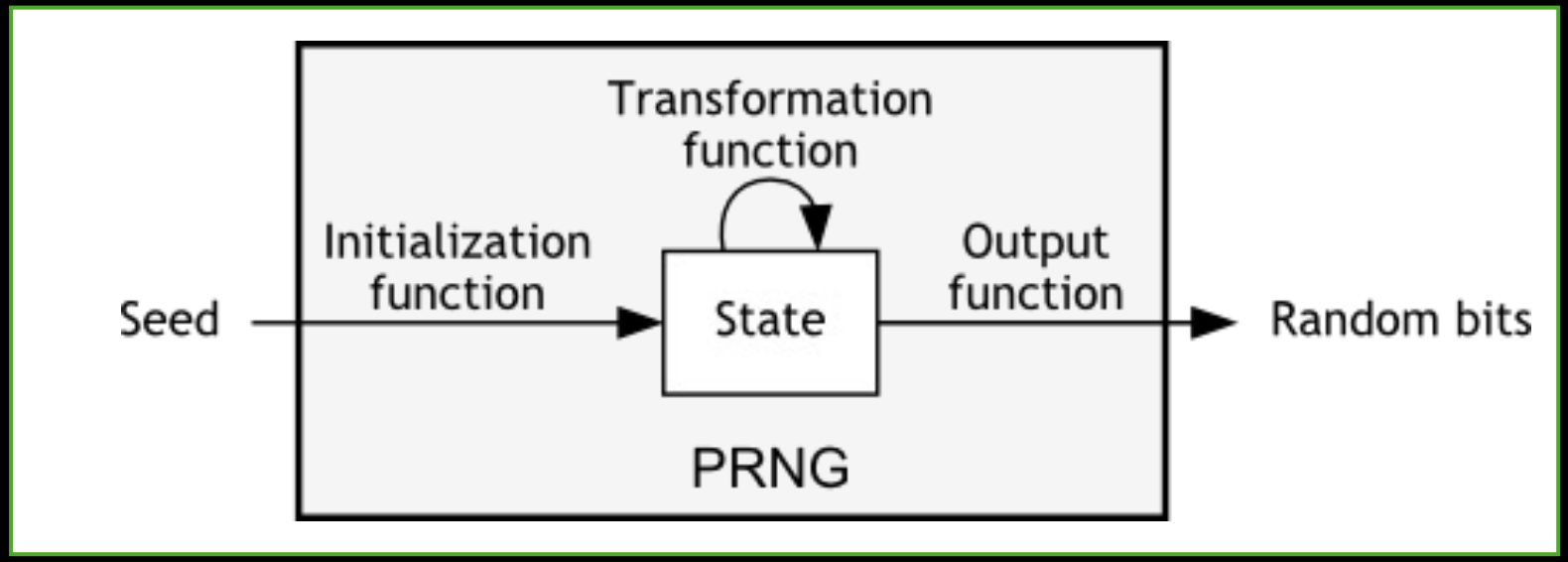
Random Number Generators and Pseudorandom Number Generators

How PRNG works?

1. A PRNG receives random bits from an RNG and uses them to update the **entropy pool**.
   - Entropy pool is a large memory buffer.
2. The PRNG runs a **DRBG** to expand bits from the entropy pool into longer sequence.
   - The DRBG is deterministic, not randomized: given one input you will always get the same output.
3. The PRNG ensures that its DRBG never receives the same input twice, to generate unique pseudorandom sequences

Random Number Generators and Pseudorandom Number Generators

How PRNG works?



Random Number Generators and Pseudorandom Number Generators

- The PRNG performs three operations, as follows:

1. $init()$ Initializes the entropy pool and the internal state of the PRNG.

2. $refresh(R)$ Updates the entropy pool using some data, R, usually sourced from an RNG.
   - Sometimes it is called reseed.

3. $next(N)$ Returns N pseudorandom bits and updates the entropy pool.

Random Number Generators and Pseudorandom Number Generators

- Non-crypto PRNGs are designed for applications such as scientific simulations or video games.
  - They are insecure.

- Crypto PRNGs, are unpredictable.
  - Optimized to output unpredictable well-distributed bits.

Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Real-World PRNGs

- Crypto PRNGs are found in the operating systems of most platforms.
- Some of them are software-based, but some are pure hardware-based.

Real-World PRNGs

- In Unix systems, the $dev/urandom$ file is used to generate secure random bits.
- Since, it is a file, we generate random bits by reading it.

```
❯ hexdump -C -n 256 /dev/urandom
00000000  94 eb c8 0c 60 5b ed b7  f6 de e5 d9 4c 5c c8 6c  |....`[......L\.l|
00000010  f2 58 0c 45 45 2d 77 7d  b4 d7 d2 42 eb b8 99 2b  |.X.EE-w}...B...+|
00000020  93 2a a9 c4 3f 0a d6 82  a7 f4 3b 74 ff d6 e8 67  |.*..?.....;t...g|
00000030  a2 d8 2f 40 59 79 64 f9  27 04 66 72 1c 86 03 01  |../@Yyd.'.fr....|
00000040  9d 43 97 12 62 8f 5f 35  3e c8 81 1c 90 b6 17 95  |.C..b._5>.......|
00000050  23 75 ad e4 44 5a 29 44  b7 b5 8c db 38 09 36 6a  |#u..DZ)D....8.6j|
00000060  6c 1a e6 cf 67 03 c8 e3  f1 86 a7 d8 17 8a 1d a5  |l...g...........|
00000070  01 bc 4d 76 9f 4e 25 5c  3b 8b 52 28 d3 07 c1 ad  |..Mv.N%\;.R(....|
00000080  89 91 69 3f 00 2e f3 16  f5 c0 96 a2 c3 b6 7b ef  |..i?..........{.|
00000090  6c 99 7c f5 d0 9e b9 07  d3 11 b9 95 b2 2f cd d2  |l.|........../..|
000000a0  0e 88 a8 3f 30 0d 75 34  a8 ce bb 7e d9 f8 f2 72  |...?0.u4...~...r|
000000b0  20 2e fa 0a f5 f8 f5 1c  c0 81 f5 0b a4 d8 1c b2  | ...............|
000000c0  8f 34 be 31 7b a0 79 96  de 0b 02 f9 42 61 b1 5f  |.4.1{.y.....Ba._|
000000d0  05 4c 7d 5d bf 2c e8 2e  b4 bf 1b cf 90 fa eb ea  |.L}].,..........|
000000e0  40 2f 65 f8 5b 8f cd 23  6c ee 6d ab fc ad 18 b0  |@/e.[..#l.m.....|
000000f0  c5 99 dd 9f 4c 7f e2 da  a5 38 7e 4e 48 48 28 59  |....L....8~NHH(Y|
```

Real-World PRNGs

- The Wrong Way to Use $/dev/urandom$

```
int random_bytes_insecure(void *buf, size_t len)
{
    int fd = open("/dev/urandom", O_RDONLY);
    read(fd, buf, len);
    close(fd);
    return 0;
}
```

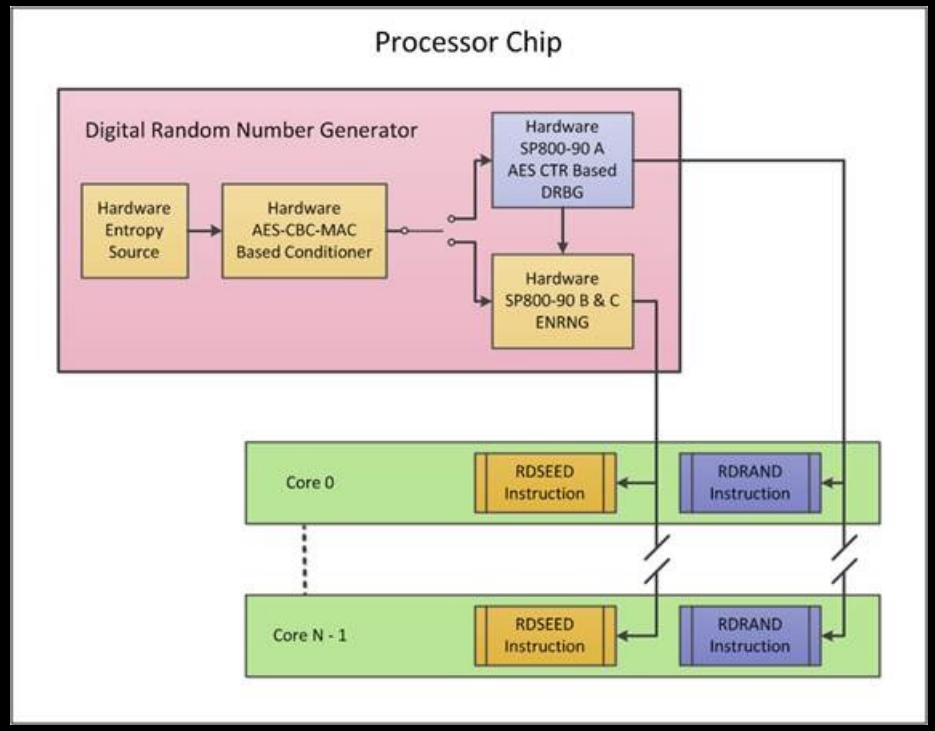
- This code is insecure; it doesn't check the return values of $open()$ and $read()$.
  - This can cause the random buffer filled with zeros or left unchanged.

Real-World PRNGs

- In Windows, the PRNG is the $CryptGenRandom()$ function from the Cryptography API.
  - This is deprecated.
- New windows versions use $BcryptGenRandom()$ function in the Cryptography Next Generation (CNG) API.
- The Windows PRNG takes entropy from the kernel mode driver $cng.sys$ (formerly $ksecdd.sys$)

Real-World PRNGs

- Intel has its own PRNG designed as hardware-based PRNG.
- Intel's PRNG is accessed through the $RDRAND$ assembly instruction.
  - It is independent of the operating system and is faster than software PRNGs.



Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

DES

- Data Encryption Standard (DES) is a block cipher.
- Encrypts data in blocks, each is 64 bits.
  - 64 bits of plaintext go as the input to DES, which produces 64 bits of ciphertext.

Ptxt1 (64-bits)	Ptxt2 (64-bits)	Ptxt3(64-bits)	Ptxt4 (64-bits)	Ptxt5 (64-bits)	Ptxt6 (64-bits)

- The key length is 56 bits.
- The same algorithm and key are used for encryption and decryption, with minor differences.

DES

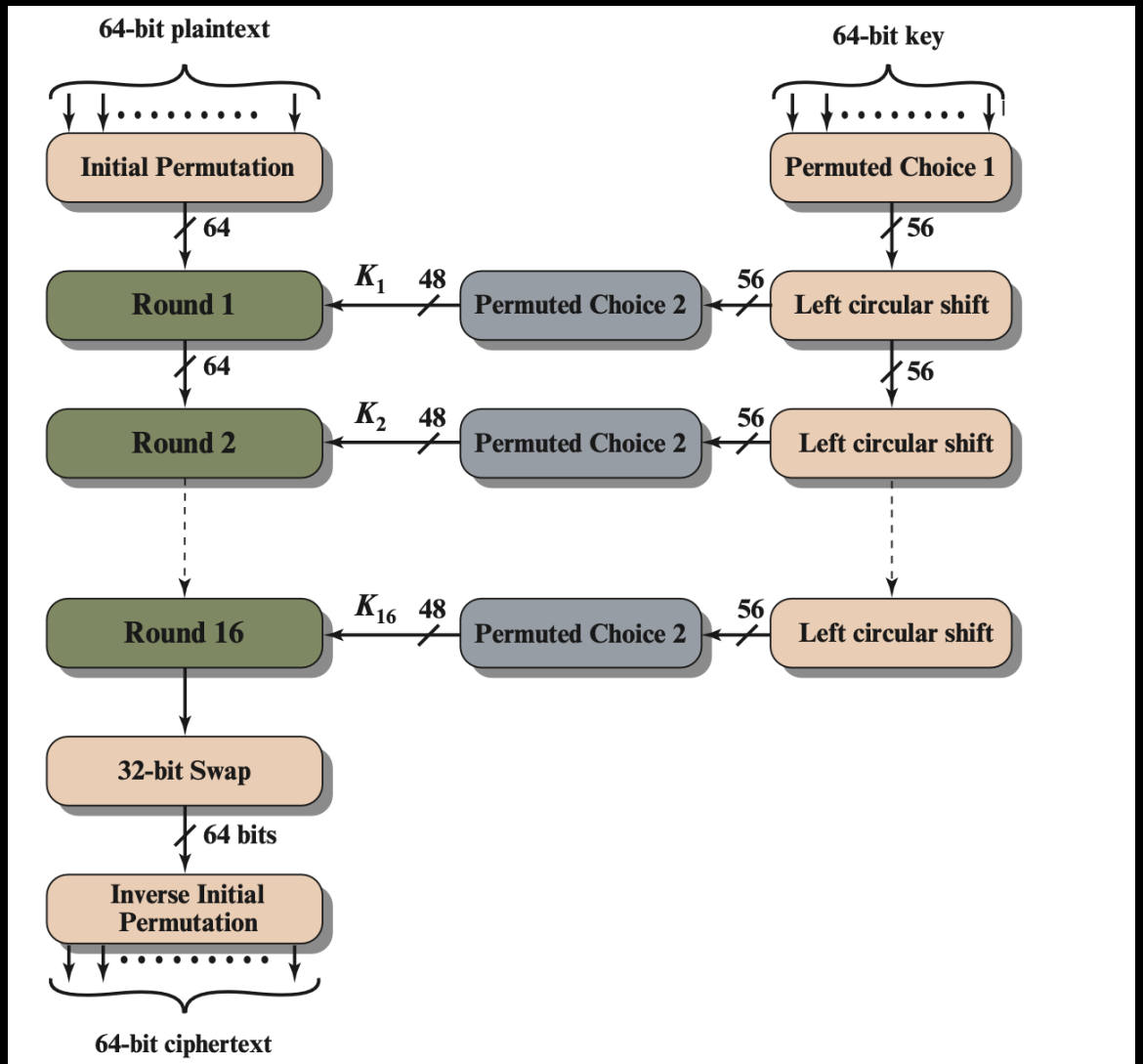


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- The function expects a 64-bit key as input. However, only 56 of these bits are ever used.

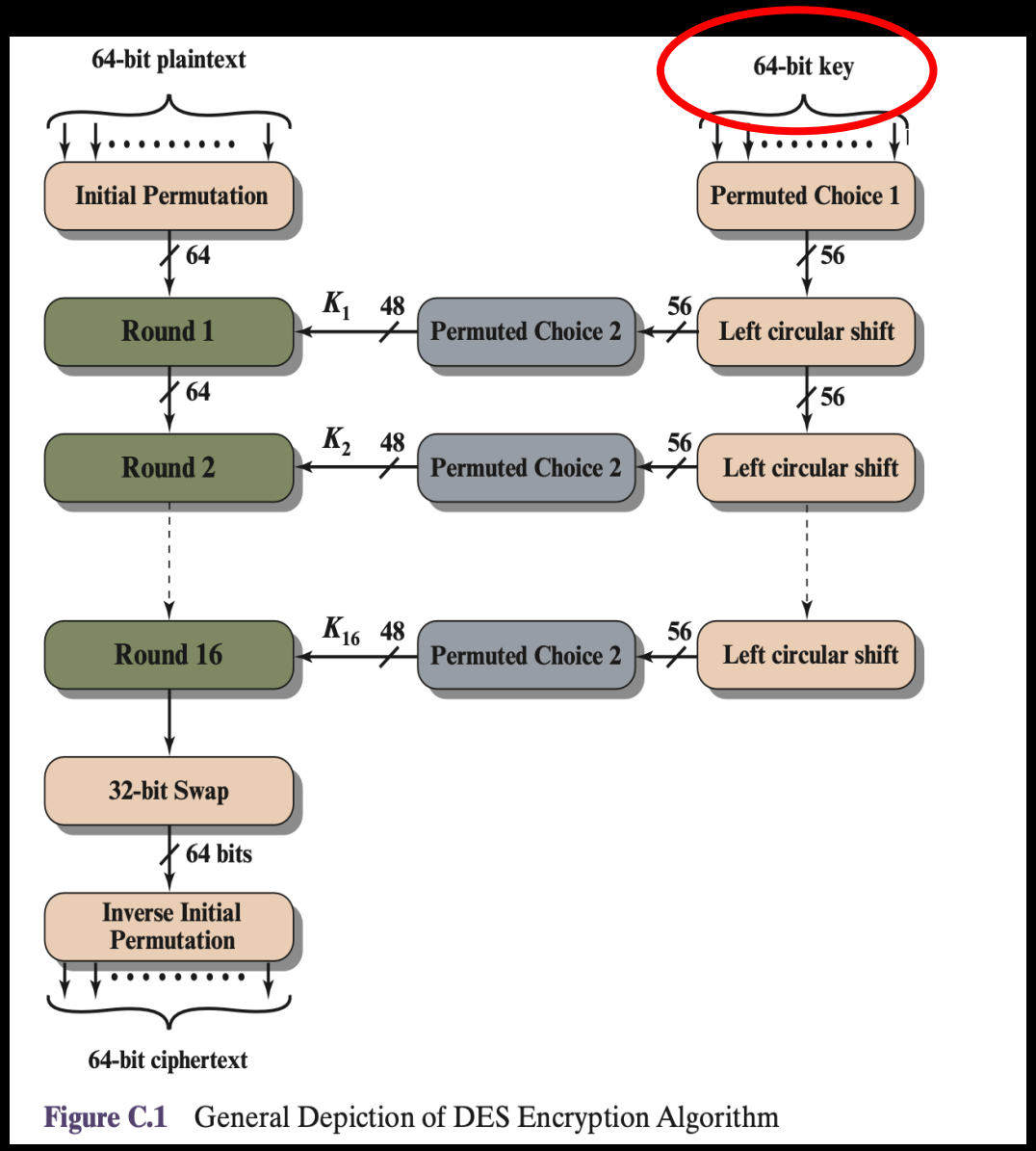


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- The plaintext is processed in three phases:
  1. The 64-bit plaintext passes through an IP that rearranges the bits to produce the permuted input.

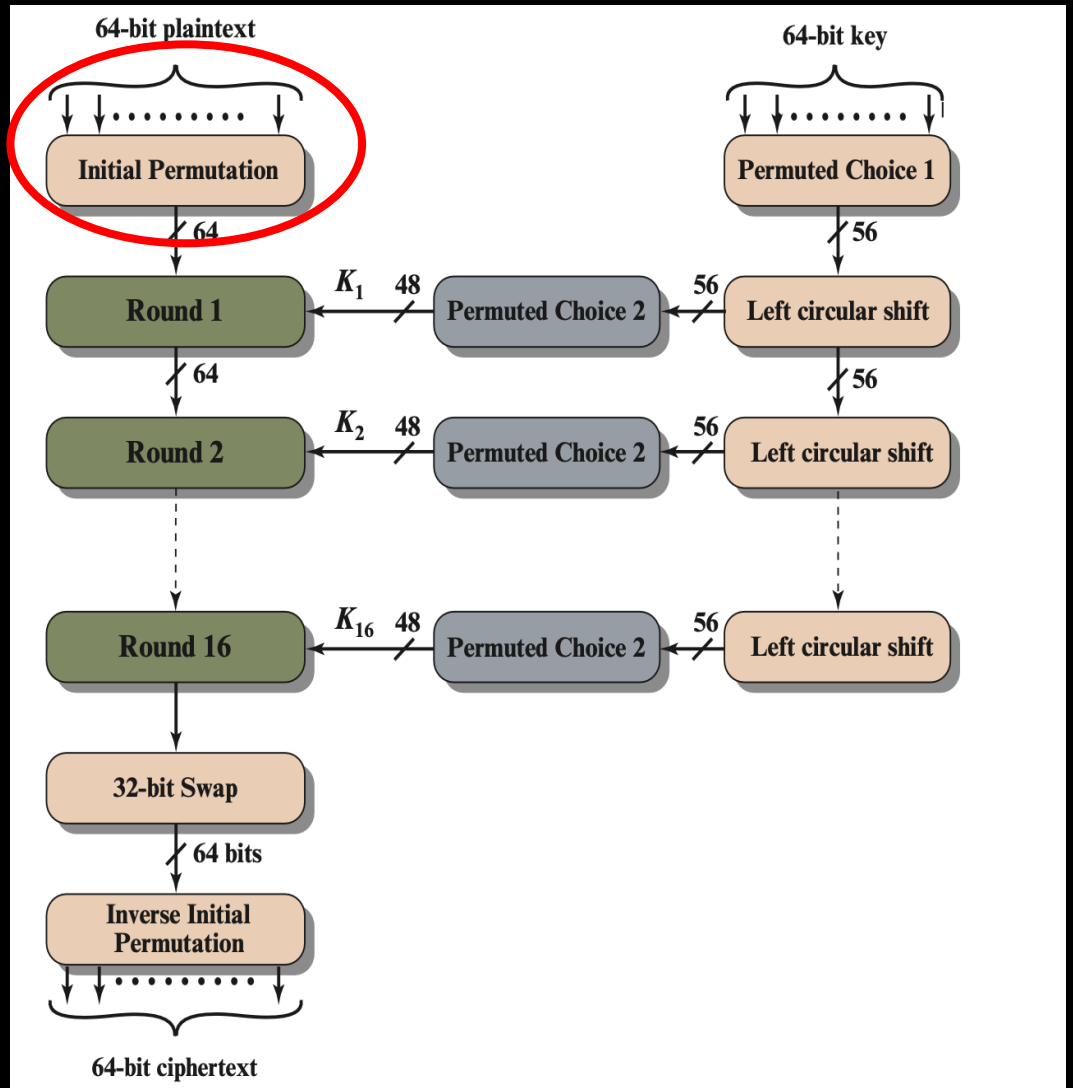


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- The plaintext is processed in three phases:
  1. The 64-bit plaintext passes through an IP that rearranges the bits to produce the permuted input.
  2. A phase of 16 rounds that involve substitutions and permutations. Then, the left and right halves of the output are swapped.

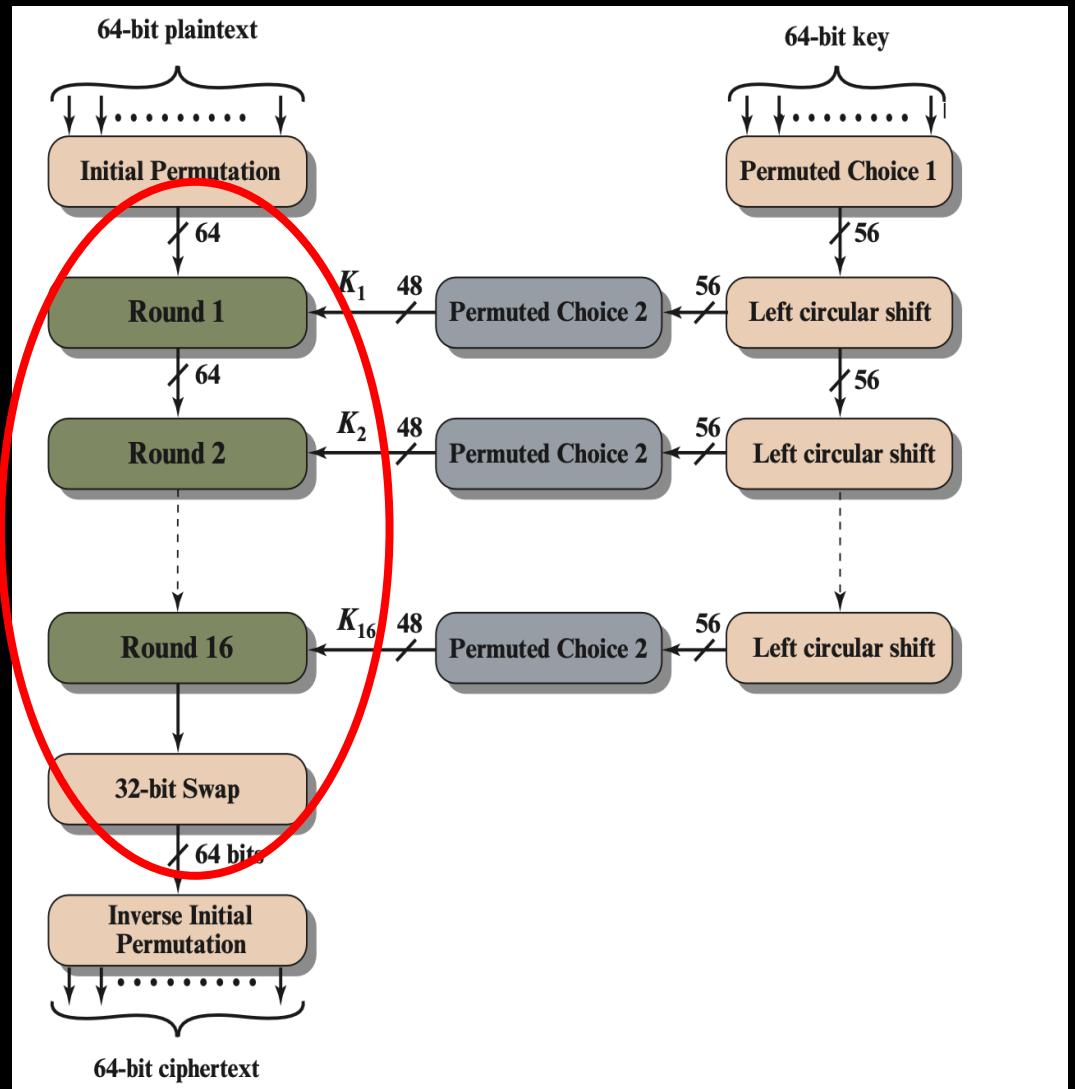


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- The plaintext is processed in three phases:
  1. The 64-bit plaintext passes through an IP that rearranges the bits to produce the permuted input.
  2. A phase of 16 rounds that involve substitutions and permutations. Then, the left and right halves of the output are swapped.
  3. The preoutput is passed through the inverse IP to produce the final 64-bit ciphertext.

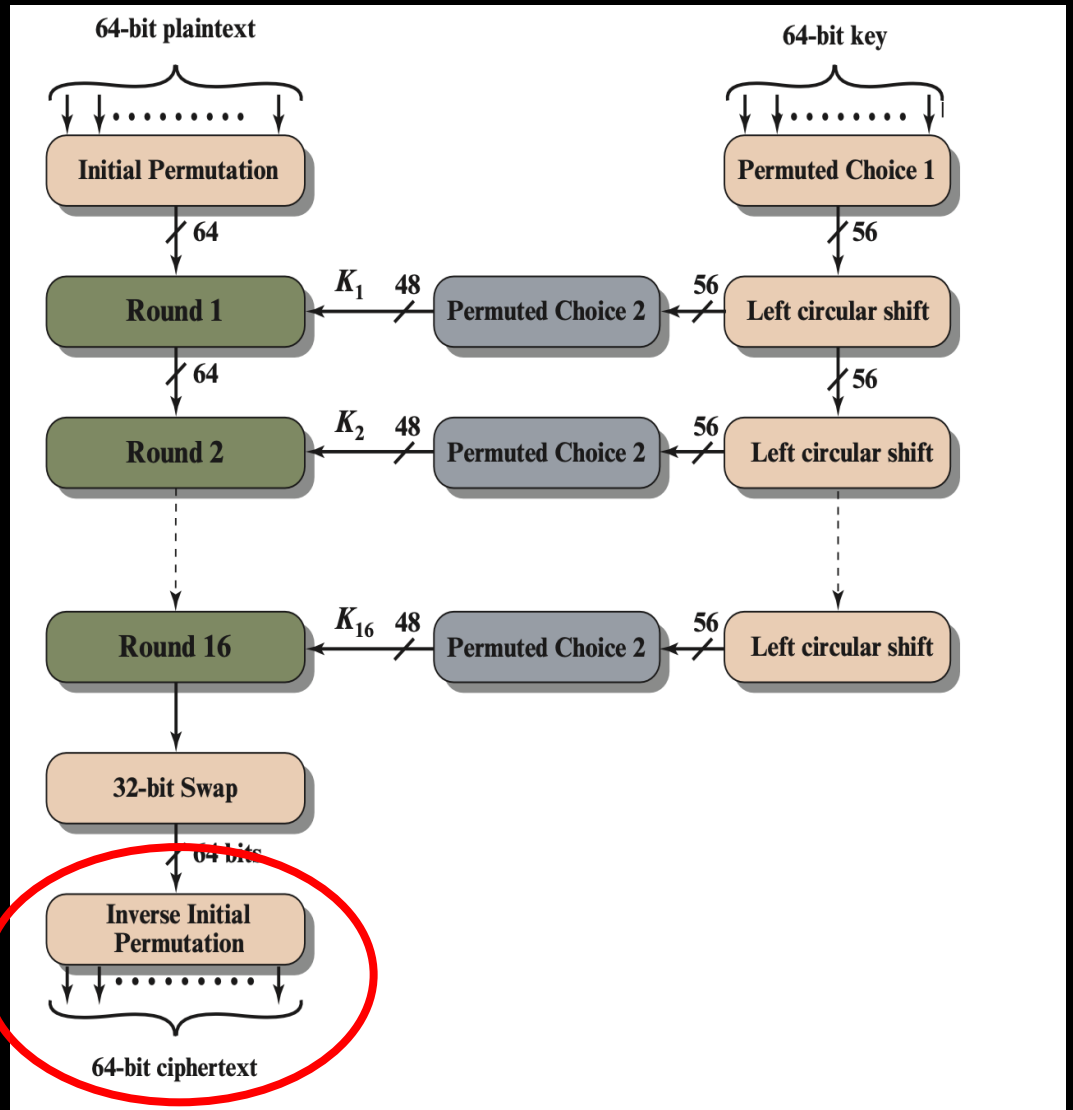


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- The key is passed to a permutation function.

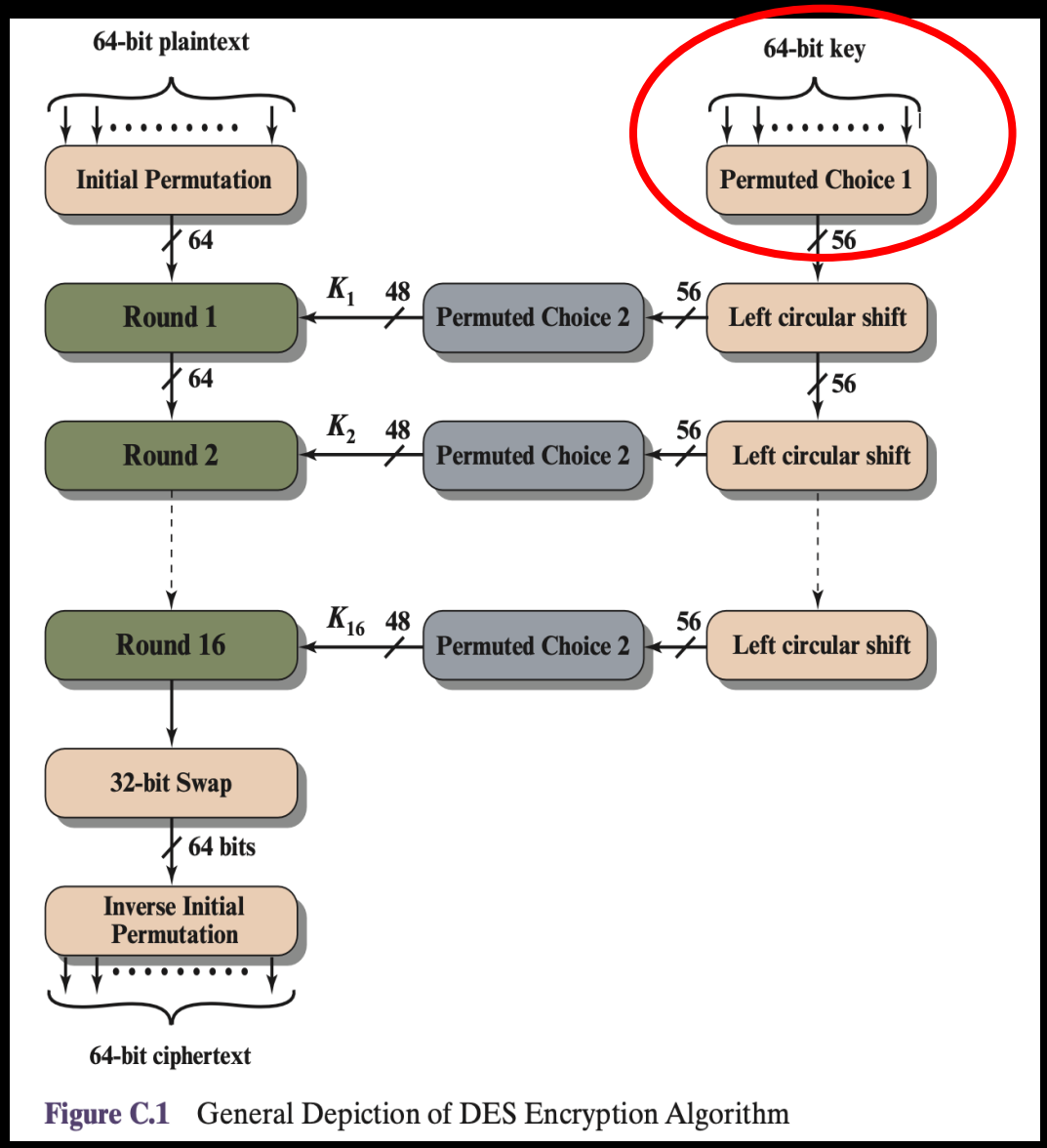


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- The key is passed to a permutation function.
- For each of the 16 rounds, a subkey is produced by a combination of circular shift and a permutation function.

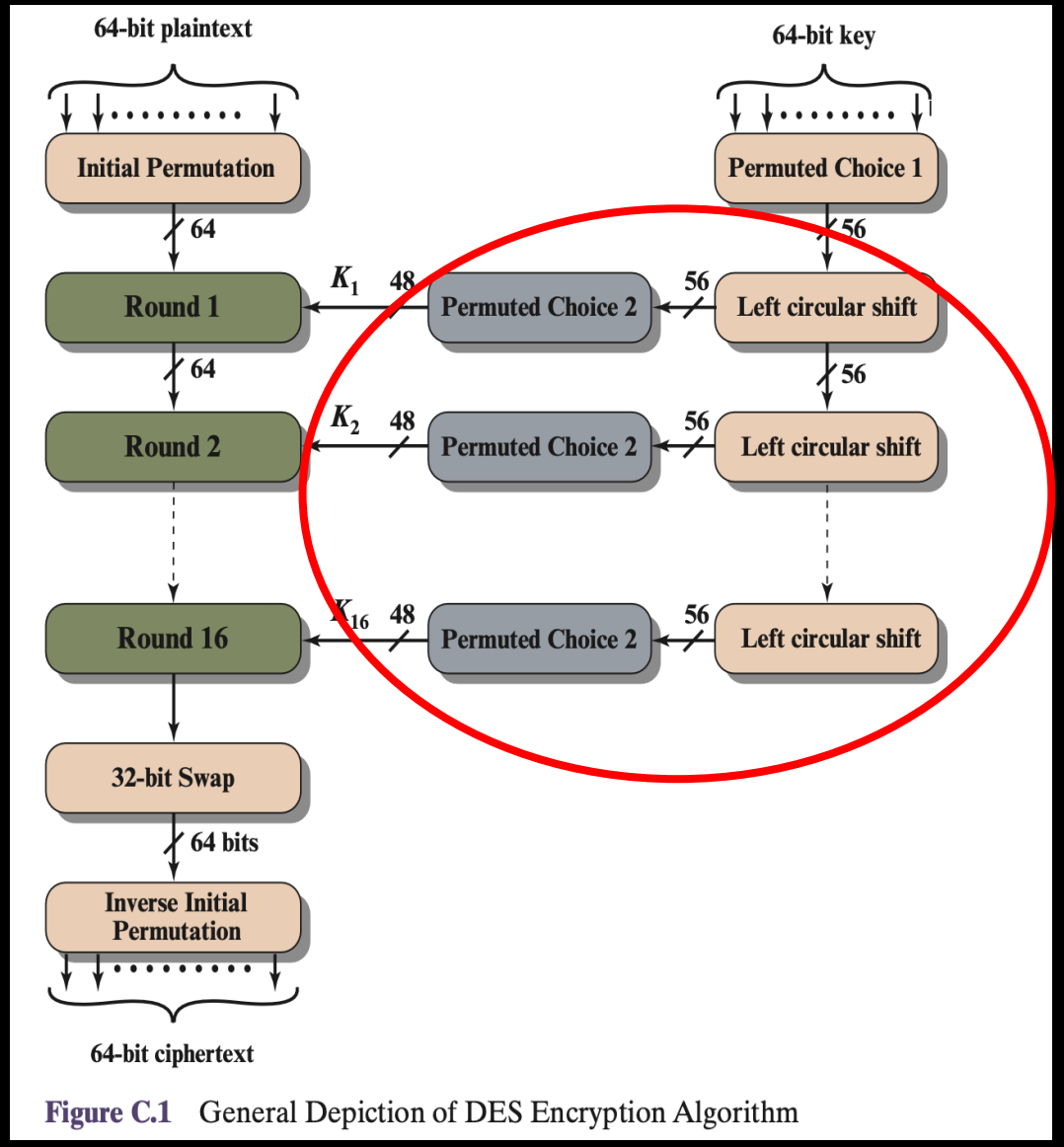


Figure C.1 General Depiction of DES Encryption Algorithm

DES

- Decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

DES

- Implement the DES algorithm in Python with $pycryptodome$.

$$pip\ install\ pycryptodome$$

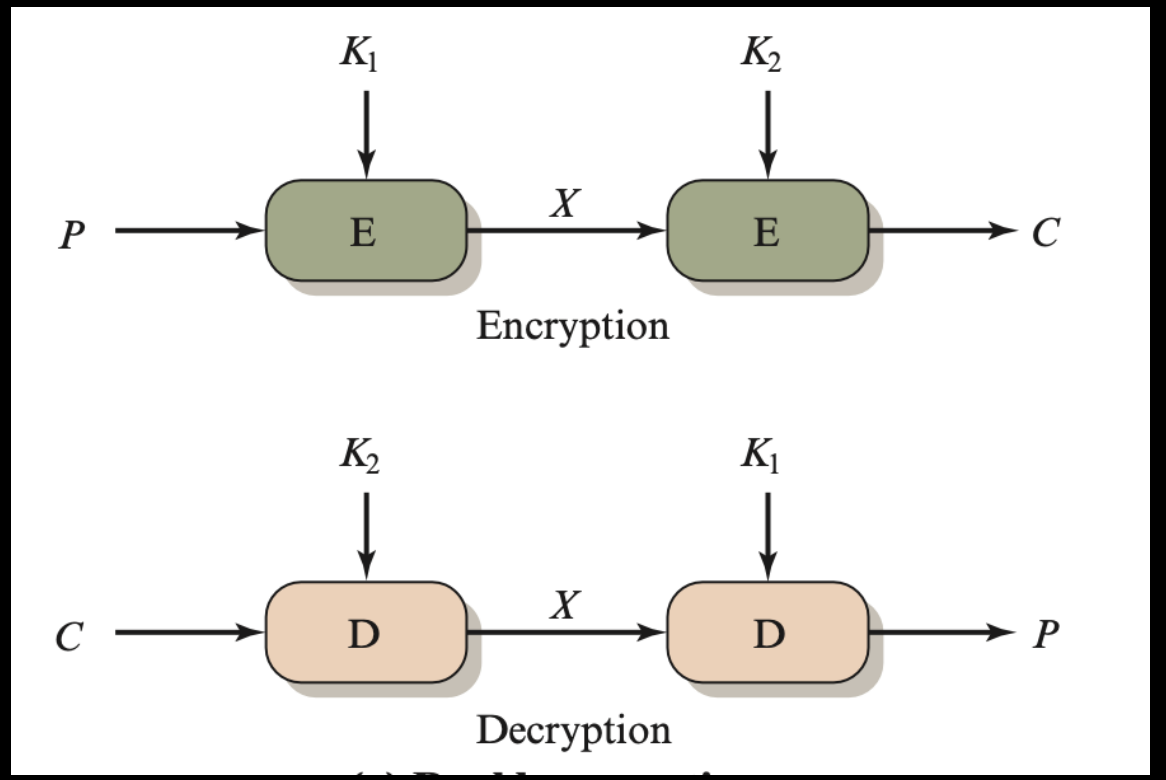
Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Attacks on DES

- DES's key is very short – 56 bits.
- The key space is small, 2^{56}.
- This makes it susceptible to brute force attacks.
- See the $DES\_Break\_Bruteforce.py$ file

Attacks on DES

- Double-DES was proposed to mitigate the short key of DES algorithm.
- It uses two instances of DES algorithm on the same plaintext with two different keys.
  - $C = E(K_2, E(K_1, P))$
  - $P = D(K_1, D(K_2, C))$



Attacks on DES

- With double DES, the key size is $56 \times 2 = 112$ bits.
  - This should be more secure!
- This can be broken with a technique called Meet-in-the-Middle attack.
- MitM reduces the double DES from 112 bits to 57 bits.

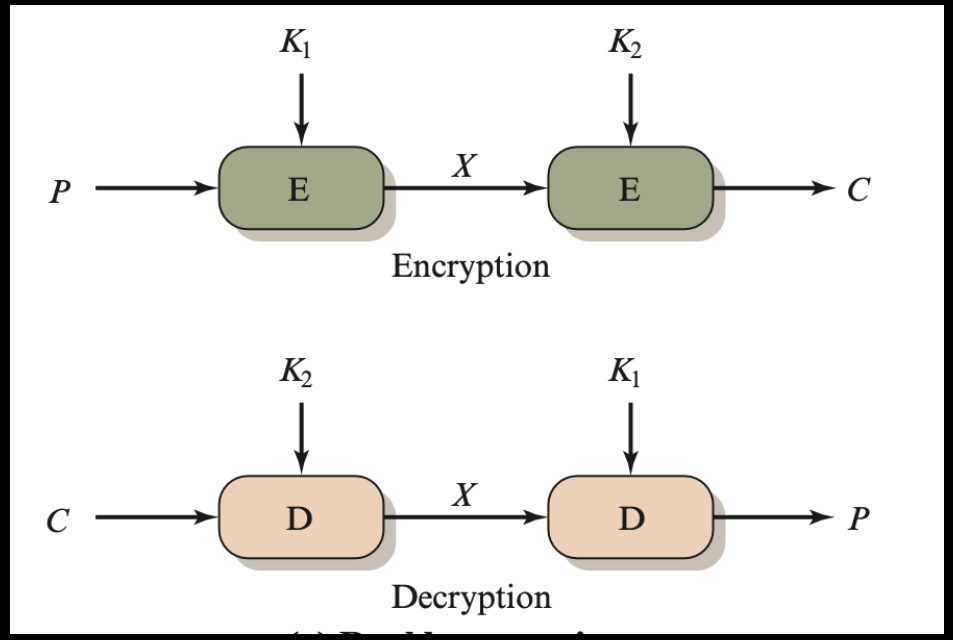
Attacks on DES

- Since double-DES is based on the observation:

$$C = E(K_2, E(K_1, P))$$

- Then, we have

$$E(K_1, P) = X = D(K_2, C)$$



Attacks on DES

- Given a plaintext and a ciphertext pairs (P, C):
  1. Encrypt P with all possible 2^{56} keys and store the results of each key, K_1.
  2. Decrypt C with all possible 2^{56} keys and store the results of each key, K_2.
  3. Check if two results are the same. That is your key.

- All we had to do is brute force the DES two times.

$$2^{56} + 2^{56} = 2^{56} \times 2 = 2^{57}$$

Attacks on DES

- Given a plaintext and a ciphertext pairs (P, C):
  1. Encrypt P with all possible 2^{56} keys and store the results of each key, K_1.
  2. Decrypt C with all possible 2^{56} keys and store the results of each key, K_2.
  3. Check if two results are the same. That is your key.

- All we had to do is brute force the DES two times.

$$2^{56} + 2^{56} = 2^{56} \times 2 = 2^{57}$$

- This is a Known-plaintext Attack (KPA) model

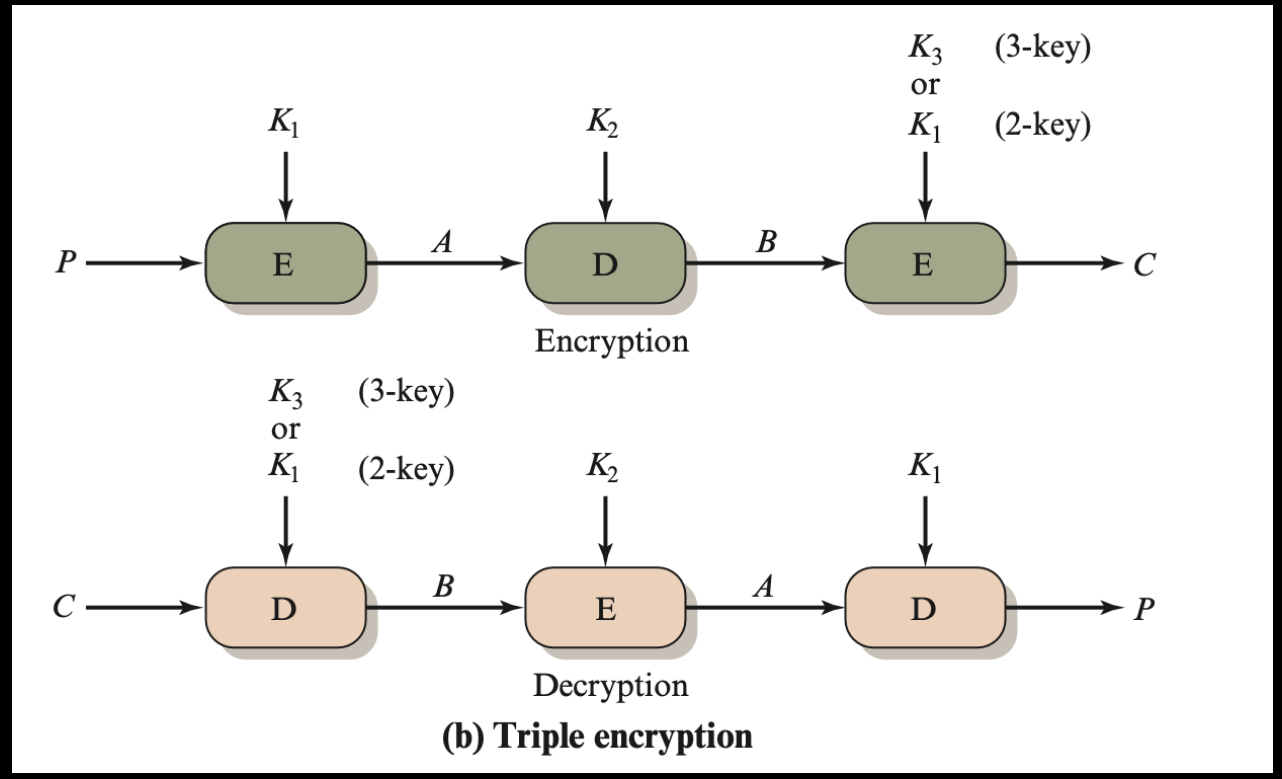
Attacks on DES

- See the $DES\_Break\_MitM.py$ file

Content
Introduction
Randomness as a Probability Distribution
Entropy: A Measure of Uncertainty
Random Number Generators and Pseudorandom Number Generators
Real-World PRNGs
DES
Attacks on DES
Triple DES

Triple DES

- Triple DES uses three instances of DES on the same plaintext.
- There are two versions of 3DES; one using two keys and one using three keys.
- Two-key 3DES:
  - $C = E(K_1, D(K_2, E(K_1, P)))$
  - $P = D(K_1, E(K_2, D(K_1, C)))$
- Three-key 3DES:
  - $C = E\left(K_3, D\left(K_2, E(K_1, P)\right)\right)$
  - $P = D\left(K_1, E\left(K_2, D(K_3, C)\right)\right)$



(b) Triple encryption

Triple DES

- Although there are no practical attacks on 3DES, it is no longer considered a strong encryption method by today’s standards.
  - Because it is based on a weak algorithm.

TASK

- What is the difference between the $dev/urandom$ and $dev/random$ files in Linux?
- Implement the 3DES algorithm in Python using $pycryptodome$.
- What is backward secrecy and forward secrecy in PRNGs.
- Challenge: I have used DES algorithm (similar to the code explained in the lab) to encrypt a secret flag. But the key that I used was a $weak\ key$. Can you recover the secret flag?
  - Ciphertext (in bytes as output from Python): b"lY\xd3ki'\x13b\xce\x86\x0cU\xb5B\x0e}\xa9S\x05\xe9\xc5\xc6\xe7s”
  - The flag starts with “FCIL”.
  - Your solution should include the explanation of the idea and the code.

References

- https://www.cs.purdue.edu/homes/ninghui/courses/Fall05/lectures/355_Fall05_lect17.pdf
- Serious Cryptography: A Practical Introduction to Modern Encryption (main reference)
- Cryptography and network security principles and practice