«به نام خدا»

تکلیف دوم – سوال 7 – مرضیه علیدادی – 9631983

7.

Symmetric encryption ciphers consist of two main categories: block ciphers and stream ciphers. We'll define and break down the processes of each, and provide block and stream cipher examples to give you a closer look at the technologies that keep your data safe.

Understanding the difference between a block cipher vs stream cipher is kind of like the difference between watching a movie on DVD or via a streaming service. Sure, both will give you the entertainment you seek, but they each work differently in terms of mechanics and speed.

Block Cipher vs Stream Cipher:

Block and stream ciphers are two ways that you can encrypt data. Also known as bulk ciphers, they're two categories of symmetric encryption algorithms. (Reminder: with symmetric encryption, you use the same key to encrypt and decrypt data.) Block and stream ciphers are two separate routes to the same end goal of securing your data. The big difference between the two is *how* the data gets encrypted — and there are advantages and disadvantages to each method — and the types of environments they operate in.

For now, let's break down what these ciphers are in general and how they work.

Block ciphers and stream ciphers are two separate methods of encrypting data with symmetric encryption algorithms:

1. Encrypting information in chunks. A block cipher breaks down plaintext messages into fixed-size blocks before converting them into ciphertext using a key.

2. Encrypting information bit-by-bit. A stream cipher, on the other hand, breaks a plaintext message down into single bits, which then are converted individually into ciphertext using key bits.

(Note: some people say stream ciphers encrypt data by individual bits, others say by bytes [8 bits]. So, I'll just stick with saying individual bits for the sake of ease in this article.)

If you want a simple analogy to better understand a block cipher vs stream cipher, imagine you're encrypting a book. You could encrypt the content one page at a time (block cipher) or one letter at a time (stream cipher).

Of course, both processes are more complicated than that, but that gives you a basic idea of what they are and what they do. According to Jean-Philippe Aumasson in his book "Serious Cryptography: A Practical Introduction to Modern Encryption":

"[…] block ciphers mix chunks of plaintext bits together with key bits to produce chunks of ciphertext of the same size, usually 64 or 128 bits. Stream ciphers, on the other hand, don't mix plaintext and key bits; instead, they generate pseudorandom bits from the key and encrypt the plaintext by XORing it with the pseudorandom bits[.]"

If that was clear as mud, no worries. We'll dive more into the nitty-gritty technical side of things shortly. But first, where would you find these types of ciphers in use? Look no further than the technologies around you.

Block ciphers are the algorithms that form the backbone of many of the cryptographic technologies and processes that are in use today in computer communications. Basically, you can find block ciphers just about anywhere in cyber security.

Where stream ciphers are concerned, they're not as well studied. However, you'll find stream ciphers in use in:

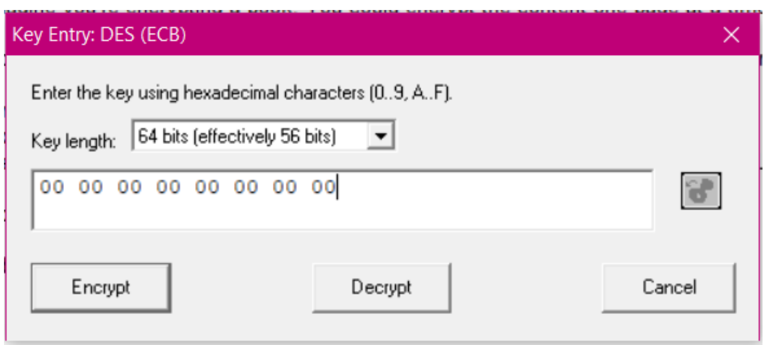
1. SSL/TLS connections

2. Bluetooth connections

3. Cellular and 4G connections

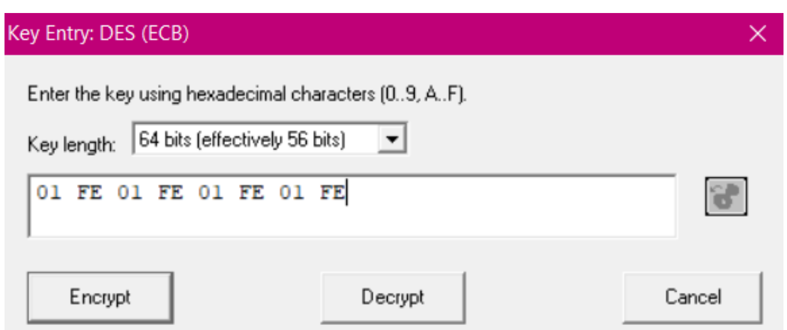
7.1.

i. **Cipher key:** 00 00 00 00 00 00 00 00



(I encrypted the text twice. And saved the files as "first.hex" and "second.hex", in folder "**7.1/i**".)

ii. **Cipher key:** 01 FE 01 FE 01 FE 01 FE



(I encrypted the text twice. And saved the files as "first.hex" and "second.hex", in folder "**7.1/ii**".)

7.2.

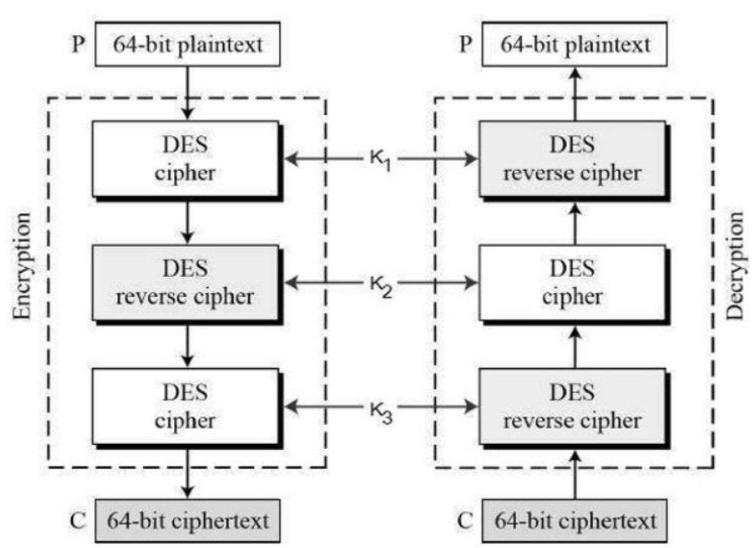
i. Triple DES runs three times slower than DES, but is much more secure if used properly. The procedure for decrypting something is the same as the procedure for encryption, except it is executed in reverse. Like DES, data is encrypted and decrypted in 64-bit chunks. Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits in length. The least significant (right-most) bit in each byte is a parity bit, and should be set so that there are always an odd number of 1s in every Byte. These parity bits are ignored, so only the seven most significant bits of each byte are used, resulting in a key length of 56 bits. This means that **the effective key strength for Triple DES is actually 168 bits** because each of the three keys contains 8 parity bits that are not used during the encryption process.

And since the length of its key is longer, it's obvious that it is more secure.

ii. In general, Triple DES with three independent keys has a key length of 168 bits (three 56-bit DES keys), but due to the meet-in-the-middle attack, the effective security it provides is only **112 bits**.
there is an attack on 3TDEA that reduces the strength to the work that would be involved in exhausting a 112-bit key.
we assume κ bits per key. The problem for an attacker is that she has to compute a lookup table either after the first or after the second encryption. In both cases, the attacker has to compute two encryptions or decryptions in a row in order to reach the lookup table. Here lies the cryptographic strength of triple encryption: There are 22k possibilities to run through all possible keys of two encryptions or decryptions. In the case of 3DES, this forces an attacker to perform 2112 key tests, which is entirely infeasible with current technology.

iii. There are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2-key Triple DES (2TDES).

3-KEY Triple DES: Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1, K_2 and K_3. This means that the actual 3TDES key has length 3×56 = 168 bits. The encryption scheme is illustrated as follows:



The encryption-decryption process is as follows:

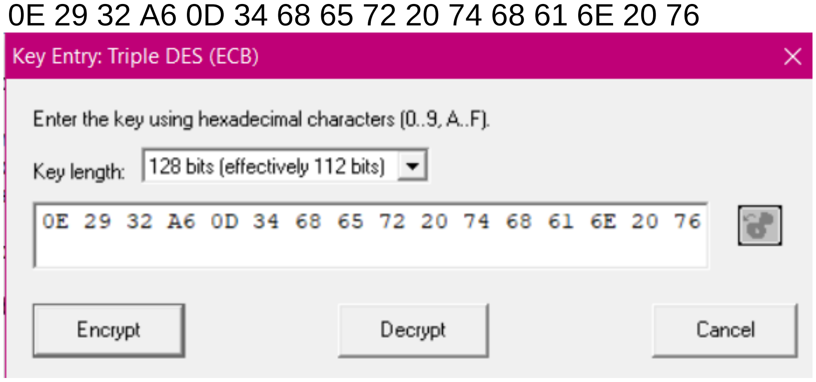
- Encrypt the plaintext blocks using single DES with key K_1.
- Now decrypt the output of step 1 using single DES with key K_2.
- Finally, encrypt the output of step 2 using single DES with key K_3.
- The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.

Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K_1, K_2, and K_3 to be the same value. This provides backwards compatibility with DES.

Second variant of Triple DES (2TDES) is identical to 3TDES except that K_3 is replaced by K_1. In other words, user encrypt plaintext blocks with key K_1, then decrypt with key K_2, and finally encrypt with K_1 again. Therefore, 2TDES has a key length of 112 bits.

Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.

iv. **Cipher key:** 0E 29 32 A6 0D 34 68 65 72 20 74 68 61 6E 20 76



(I encrypted the text. And saved the files as “cipherText.hex”, in folder “**7.2/iv**”.)

v. **Cipher key 1:** 0E 29 32 A6 0D 34 68 65
Cipher key 2: 72 20 74 68 61 6E 20 76
Cipher key 3 = Cipher key 1: 0E 29 32 A6 0D 34 68 65

The final ciphertext is just like the former part.

(I encrypted the text in 3 rounds with DES(encryption-decryption-encryption). And saved the files as “first.hex”, “second.hex” and “third.hex”, in folder “**7.2/v**”.)

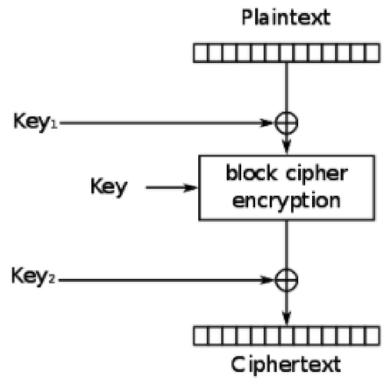
7.3.

i. **Cipher key:** 22 23 45 12 98 7A BB 23 47 89 FD 47 6E 82 A5 F1 0A 4E D5 C1 5A 63 FE A3

(I encrypted the text. And saved the files as “cipherText.hex”, in folder “**7.3/i**”.)

ii.

$$\mathrm{DES\text{-}X}(M) = K_2 \oplus \mathrm{DES}_K(M \oplus K_1)$$



Cipher key: 22 23 45 12 98 7A BB 23
Cipher key 1: 47 89 FD 47 6E 82 A5 F1
Cipher key 2: 0A 4E D5 C1 5A 63 FE A3

(I encrypted the text, with the steps shown in the figure. And saved the files as “first.hex”, “second.hex” and “third.hex”, in folder “**7.3/ii**”.)

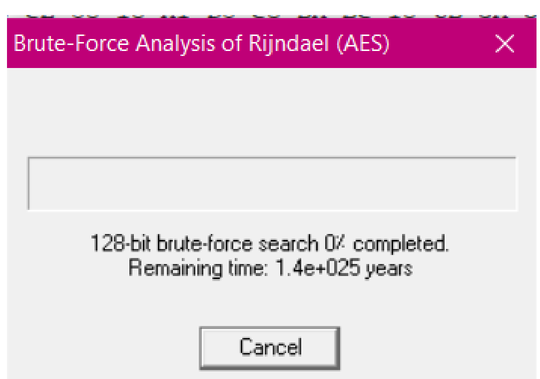
7.4.

Cipher key: 98 7A BB 23 47 6E 82 0A 4E D5 82 A5 F1 22 23 45

(I encrypted the text. And saved the files as “cipherText.hex”, in folder “**7.4**”.)

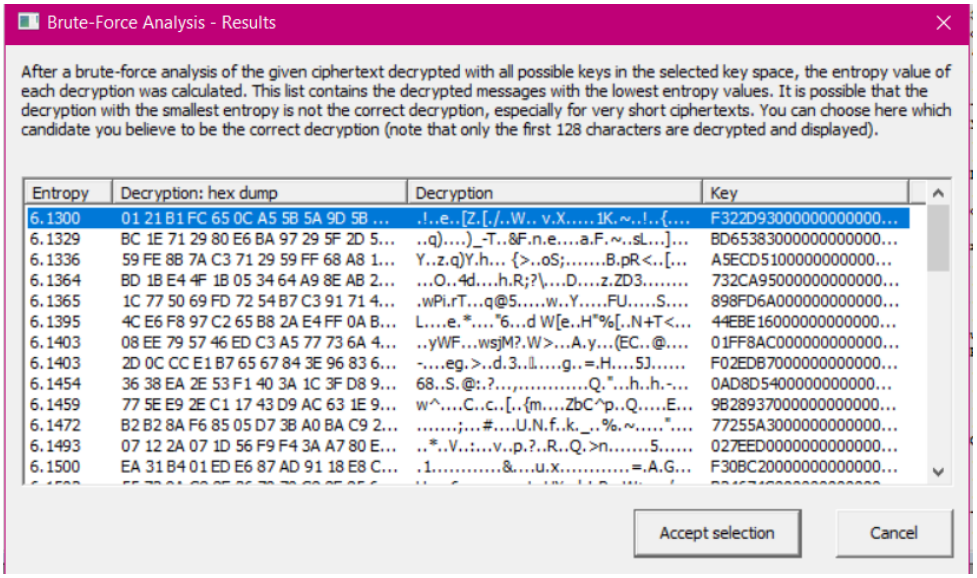
7.5.

i. The tome needed to decrypt the cipherText:
1.4e+025 years = 4.41504e+38 microseconds

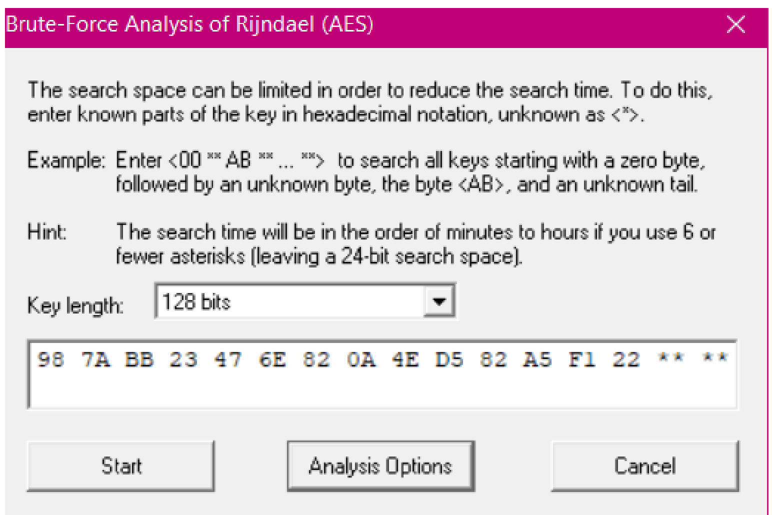


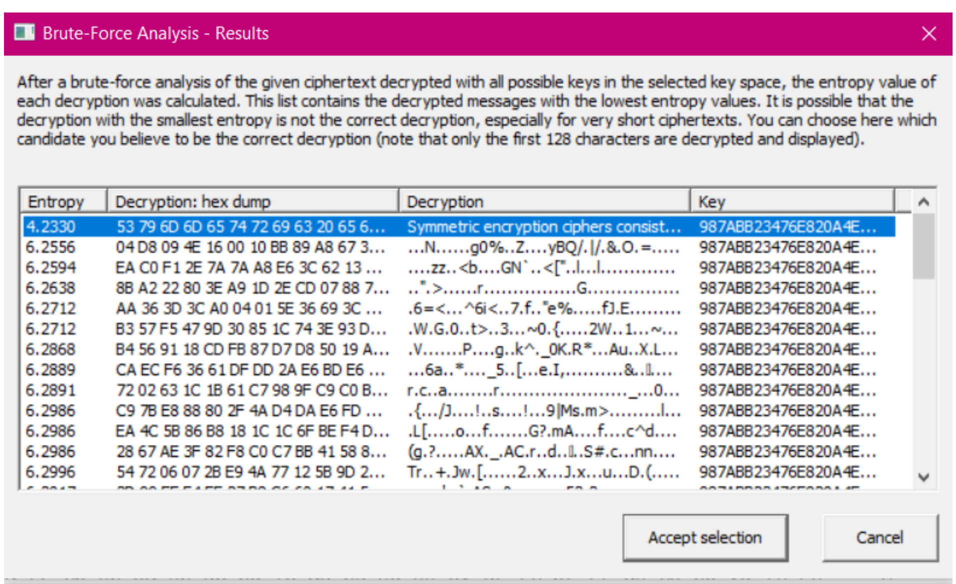
And the key space is 2^{128}. So, the time it takes to verify each key:

(4.41504e+38) ÷ 2^{128} = **1.4694068e-37 microseconds**


Brute-Force Analysis - Results
After a brute-force analysis of the given ciphertext decrypted with all possible keys in the selected key space, the entropy value of each decryption was calculated. This list contains the decrypted messages with the lowest entropy values. It is possible that the decryption with the smallest entropy is not the correct decryption, especially for very short ciphertexts. You can choose here which candidate you believe to be the correct decryption (note that only the first 128 characters are decrypted and displayed).
Entropy | Decryption: hex dump | Decryption | Key
6.1300 01 21 B1 FC 65 0C A5 5B 5A 9D 5B … .!..e..[Z.[./..W.. v.X.....1K.~..!..{.... F322D93000000000000…
6.1329 BC 1E 71 29 80 E6 BA 97 29 5F 2D 5… ..q)....)_-T..&F.n.e....a.F.~..sL...]... BD65383000000000000…
6.1336 59 FE 8B 7A C3 71 29 59 FF 68 A8 1… Y..z.q)Y.h... {>..oS;.......B.pR<..[... A5ECD51000000000000…
6.1364 BD 1B E4 4F 1B 05 34 64 A9 8E AB 2… ...O..4d....h.R;?\....D....z.ZD3........ 732CA95000000000000…
6.1365 1C 77 50 69 FD 72 54 B7 C3 91 71 4… .wPi.rT...q@5.....w..Y.....FU.....S.... 898FD6A000000000000…
6.1395 4C E6 F8 97 C2 65 B8 2A E4 FF 0A B… L....e.*...."6...d W[e..H"%[..N+T<... 44EBE16000000000000…
6.1403 08 EE 79 57 46 ED C3 A5 77 73 6A 4… ..yWF...wsjM?.W>...A.y...(EC..@.... 01FF8AC000000000000…
6.1403 2D 0C CC E1 B7 65 67 84 3E 96 83 6… -....eg.>..d.3.......g..=.H....5J...... F02EDB7000000000000…
6.1454 36 38 EA 2E 53 F1 40 3A 1C 3F D8 9… 68..S.@:.?...,...........Q."...h..h.-... 0AD8D54000000000000…
6.1459 77 5E E9 2E C1 17 43 D9 AC 63 1E 9… w^....C..c..[..{m....ZbC^p..Q.....E... 9B28937000000000000…
6.1472 B2 B2 8A F6 85 05 D7 3B A0 BA C9 2… ....;...#....U.N.f..k._..%.~....."... 77255A3000000000000…
6.1493 07 12 2A 07 1D 56 F9 F4 3A A7 80 E… ..*..V..:...v..p.?..R..Q.>n.......5...... 027EED0000000000000…
6.1500 EA 31 B4 01 ED E6 87 AD 91 18 E8 C… .1...........&....u.x...........=.A.G... F30BC20000000000000…
Accept selection
Cancel

ii.


Brute-Force Analysis of Rijndael (AES)
The search space can be limited in order to reduce the search time. To do this, enter known parts of the key in hexadecimal notation, unknown as <*>.
Example: Enter <00 ** AB ** ... **> to search all keys starting with a zero byte, followed by an unknown byte, the byte <AB>, and an unknown tail.
Hint: The search time will be in the order of minutes to hours if you use 6 or fewer asterisks (leaving a 24-bit search space).
Key length: 128 bits
98 7A BB 23 47 6E 82 0A 4E D5 82 A5 F1 22 ** **
Start
Analysis Options
Cancel


Brute-Force Analysis - Results
After a brute-force analysis of the given ciphertext decrypted with all possible keys in the selected key space, the entropy value of each decryption was calculated. This list contains the decrypted messages with the lowest entropy values. It is possible that the decryption with the smallest entropy is not the correct decryption, especially for very short ciphertexts. You can choose here which candidate you believe to be the correct decryption (note that only the first 128 characters are decrypted and displayed).
Entropy | Decryption: hex dump | Decryption | Key
4.2330 53 79 6D 6D 65 74 72 69 63 20 65 6… Symmetric encryption ciphers consist… 987ABB23476E820A4E…
6.2556 04 D8 09 4E 16 00 10 BB 89 A8 67 3… ...N......g0%..Z....yBQ/.|/.&.O.=..... 987ABB23476E820A4E…
6.2594 EA C0 F1 2E 7A 7A A8 E6 3C 62 13 … ....zz..<b....GN`..<["..l...l............ 987ABB23476E820A4E…
6.2638 8B A2 22 80 3E A9 1D 2E CD 07 88 7… ..".>......r................G............... 987ABB23476E820A4E…
6.2712 AA 36 3D 3C A0 04 01 5E 36 69 3C … .6=<...^6i<..7.f.."e%.....fJ.E......... 987ABB23476E820A4E…
6.2712 B3 57 F5 47 9D 30 85 1C 74 3E 93 D… .W.G.0..t>..3...~0.{.....2W..1...~... 987ABB23476E820A4E…
6.2868 B4 56 91 18 CD FB 87 D7 D8 50 19 A… .V.......P....g..k^._0K.R*...Au..X.L... 987ABB23476E820A4E…
6.2889 CA EC F6 36 61 DF DD 2A E6 BD E6 … ...6a..*...._5..[...e.I,..........&...... 987ABB23476E820A4E…
6.2891 72 02 63 1C 1B 61 C7 98 9F C9 C0 B… r.c..a........r......................_...0... 987ABB23476E820A4E…
6.2986 C9 7B E8 88 80 2F 4A D4 DA E6 FD … .{.../J....!..s....!...9|Ms.m>.........l... 987ABB23476E820A4E…
6.2986 EA 4C 5B 86 B8 18 1C 1C 6F BE F4 D… .L[.....o...f.......G?.mA....f....c^d.... 987ABB23476E820A4E…
6.2986 28 67 AE 3F 82 F8 C0 C7 BB 41 58 8… (g.?.....AX._.AC.r..d....S#.c...nn.... 987ABB23476E820A4E…
6.2996 54 72 06 07 2B E9 4A 77 12 5B 9D 2… Tr..+.Jw.[......2..x...J.x...u...D.(..... 987ABB23476E820A4E…
Accept selection
Cancel

It is decrypted successfully:

extractedPlainText.txt

Symmetric encryption ciphers consist of two main categories: block ciphers and stream ciphers. We'll define and break down the processes of each, and provide block and stream cipher examples to give you a closer look at the technologies that keep your data safe.
Understanding the difference between a block cipher vs stream cipher is kind of like the difference between watching a movie on DVD or via a streaming service. Sure, both will give you the entertainment you seek, but they each work differently in terms of mechanics and speed.
Block Cipher vs Stream Cipher:
Block and stream ciphers are two ways that you can encrypt data. Also known as bulk ciphers, they're two categories of symmetric encryption algorithms. (Reminder: with symmetric encryption, you use the same key to encrypt and decrypt data.) Block and stream ciphers are two separate routes to the same end goal of securing your data. The big difference between the two is how the data gets encrypted — and there are advantages and disadvantages to each method — and the types of environments they operate in.
For now, let's break down what these ciphers are in general and how they work.
Block ciphers and stream ciphers are two separate methods of encrypting data with symmetric encryption algorithms:
1. Encrypting information in chunks. A block cipher breaks down plaintext messages into fixed-size blocks before converting them into ciphertext using a key.
2. Encrypting information bit-by-bit. A stream cipher, on the other hand, breaks a plaintext message down into single bits, which then are converted individually into ciphertext using key bits.
(Note: some people say stream ciphers encrypt data by individual bits, others say by bytes [8 bits]. So, I'll just stick with saying individual bits for the sake of ease in this article.)
If you want a simple analogy to better understand a block cipher vs stream cipher, imagine you're encrypting a book. You could encrypt the content one page at a time (block cipher) or one letter at a time (stream cipher).
Of course, both processes are more complicated than that, but that gives you a basic idea of what they are and what they do. According to Jean-Philippe Aumasson in his book "Serious Cryptography: A Practical Introduction to Modern Encryption":
"[...] block ciphers mix chunks of plaintext bits together with key bits to produce chunks of ciphertext of the same size, usually 64 or 128 bits. Stream ciphers, on the other hand, don't mix plaintext and key bits; instead, they generate pseudorandom bits from the key and encrypt the plaintext by XORing it with the pseudorandom bits[.]"
If that was clear as mud, no worries. We'll dive more into the nitty-gritty technical side of things shortly. But first, where would you find these types of ciphers in use? Look no further than the technologies around you.
Block ciphers are the algorithms that form the backbone of many of the cryptographic technologies and processes that are in use today in computer communications. Basically, you can find block ciphers just about anywhere in cyber security.
Where stream ciphers are concerned, they're not as well studied. However, you'll find stream ciphers in use in:
1. SSL/TLS connections
2. Bluetooth connections
3. Cellular and 4G connections

The entropy of the correct decryption is: **4.2330**