6 Symmetric Cryptography: Block Ciphers

Introduction

In this section you will learn about block ciphers, or ciphers that encrypt and decrypt a block of data at a time, as opposed to encrypting or decrypting a single word or single bit. You will learn about the basic methods used to scramble a single block of data, and then the Modes of Operation developed for handling multiple blocks of data. You will also learn about some specific block ciphers and how they are implemented

The specific things you should be able to do at the end of this section are:

1. Describe the main difference between stream and block ciphers.
2. List the three main block ciphers used to scramble a single block of data, and describe in general terms how they function.
3. Define substitution, permutation, diffusion, confusion, S-Boxes and P-Boxes, and how they are related.
4. Define the terms round and round keys.
5. List the 5 Modes of Operation and describe how they function in general terms.
6. Describe why ECB should not be used.
7. Explain how CTR, CFB and OFB change a block cipher into a stream cipher.
8. Identify the main block cipher in use today in AES, and where AES is used.

Required Reading

1. **Read this document first as it will provide you with the framework regarding all the subjects covered in this section regarding symmetric ciphers, and stream ciphers.**

2. Here are some web sites you can use if you want to go further into any of the subjects in this chapter:

   https://www.youtube.com/watch?v=4FBgb2uobWI –
   Introduction to Cryptography by Christof Paar. This is a complete lecture, and full of great information. However it's full of math, and he spends quite a bit of time drawing on the chalkboard and speaking in German.

   https://www.commonlounge.com/discussion/3cbf8246f97d48eebf58b3985e8def7a - Details on RSA

   https://www.commonlounge.com/discussion/658fe9dcc92949808b9ae25a84e1a1ee - Block Ciphers
   https://www.commonlounge.com/discussion/685635bfc2e54818abf3f248cfb034f8 - Details on the Feistel Cipher or Network

   https://open.oregonstate.education/cryptography/chapter/chapter-6-pseudorandom-functions/ - Details on Pseudo Random Functions (The math gets deep quickly.)

   https://open.oregonstate.education/cryptography/chapter/chapter-7-pseudorandom-permutations/ - Details on Pseudo Random Permutations (The math gets deep quickly.)

   https://www.commonlounge.com/discussion/9638038d729b4204b60e39b230f36b03 - Details on DES
   https://www.commonlounge.com/discussion/290ee9a2afcb40fdaa21f0a03285832f - Details on AES
   https://www.commonlounge.com/discussion/770421d468e8416aa3ff0889c15fa214 - Details on Blowfish

   http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm - Excellent article on DES. It presents great background, and a detailed walkthrough.

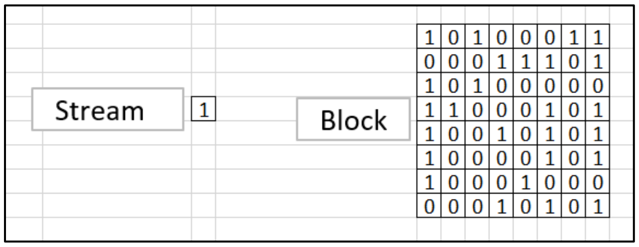
   https://en.wikipedia.org/wiki/AES_key_schedule - AES Key Scheduling Details

   https://csrc.nist.gov/csrc/media/publications/fips/81/archive/1980-12-02/documents/fips81.pdf - NIST Modes of Operation

Section Content

Block Ciphers – Introduction and Background

Block ciphers are another form of symmetric ciphers. They get their name because they encrypt and decrypt a block of data at a time, as opposed to stream ciphers which encrypt or decrypt a single word or single bit at a time. At the simplest level, this is what you need to know about block ciphers, they work on blocks of data instead of single bits. They still use a key that must be shared between the sender and the recipient, so they're symmetric; but they work on blocks of data.



The theory behind modern block ciphers has been around since 1949 when Claude Shannon published *Communication Theory of Secrecy Systems.* Shannon's design was originally called an iterated product cipher, but now it's called a block cipher. His design consisted of taking a block of data and scrambling the data by performing multiple rounds of substitutions and permutations. Each round of the scrambling uses a sub key, which is derived from a single main key.

Here are the main concepts about block ciphers:

A. They scramble blocks of data typically 64 bits or 128 bits at a time.
B. The scrambling is performed by moving the bits around inside the block or changing the bits.
C. They perform multiple rounds of scrambling on each block
D. The encryption is controlled by a key.
E. The scrambling of each block during a round is controlled by a subkey which is derived from the main key.

In this chapter you’re going to learn about several block cipher methods and implementations. Like any subject in cryptology this can be a little confusing if you try to take it all in in one big serving. But if you take a systematic approach it can much easier.

In this case you’re going to first learn about the basic methods for scrambling a single block. All the block ciphers use one of these three basic methods. Next you will learn about the five ways for handling multiple blocks of data. That is, you will learn how to move from scrambling a single block to chaining the scrambling of an entire sequence of blocks. All the block cipher implementations are different combinations of one of the basic scrambling methods plus a method for chaining together blocks. You’ll end the chapter by learning about specific block ciphers.

Three Main Block Scrambling Ciphers

The first thing to look at is how to scramble a single block of data. There are three main methods used for doing this:

A. Substitution-Permutation Networks
B. Feistel networks or ciphers
C. Lai–Massey ciphers

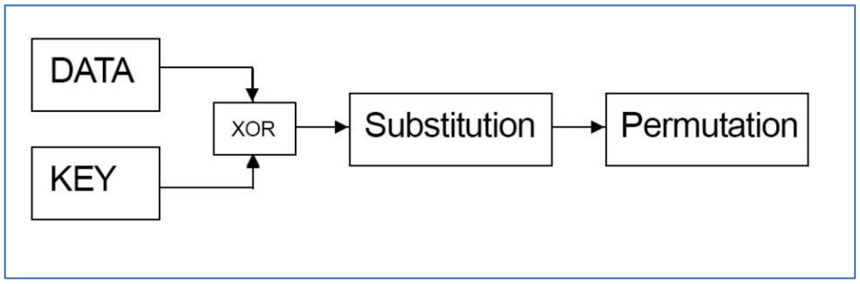
Generally speaking, all these methods do the same thing, which is they take the block of data and scramble it. But they each have unique ways to accomplish the scramble.

Block Scrambling Method 1: Substitution & Permutation Networks

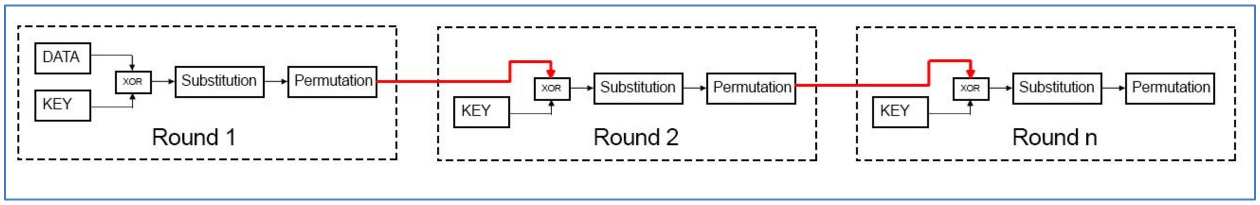
The first method is called a Substitution-Permutation Network (SPN) and it was first described by Shannon in 1949. 1949 might seem like a long time ago, but Shannon’s ideas still provide a way to build ciphers that are considered cryptographically strong and are used by several modern ciphers.

Generally speaking, a SPN cipher works by scrambling a block of plaintext by running it through several rounds or iterations of sets of operations. Each set of operations consists of:

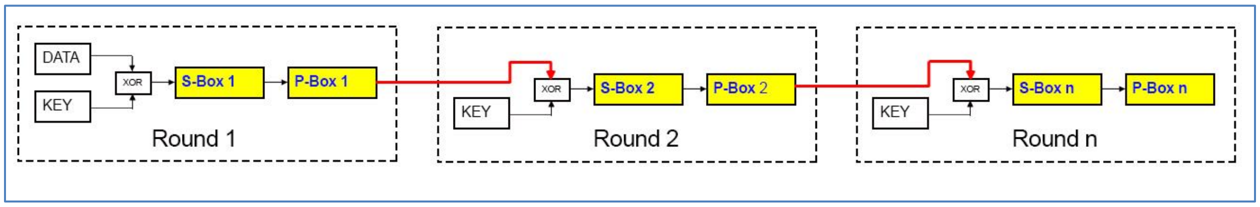
A. Mixing the bits in the block with the key, usually by an XOR
B. A substitution that replaces the bits in the block
C. A permutation which takes the bits in the block and scrambles their positions



One set of these three operations is called a *round*. One round doesn't scramble the bits enough to provide much protection and would be easy to break so multiple rounds are always used. The exact number of rounds will vary with each specific implementation, with the goal of using enough rounds to provide strong encryption, while at the same time using as few rounds as possible because each round requires extra time to process.



Each round always consists of performing the operations in the same order of mixing the data with the key, followed by a substitution, and ending with a permutation. The substitutions are performed by a section of code called an S-Box, and the permutations are performed by a section of code called a P-box. When all the S-Boxes and P-Boxes are chained together the entire process is called a network, because the rounds are networked together. That is, the output from one round is sent to the next round where it's used as the input. This is just terminology, but network, round, S-Box and P-Box are important terms to know.



It's critical to note that in some ciphers the specific calculations for the key mixing, S-Box and P-Box steps may vary in each round. That is, the way the key is calculated and mixed in round 1 may different than the way it's calculated and mixed in round 2. And the way the key is calculated and mixed in during round 2 may be different than how it will be mixed during all the other remaining rounds. The same goes for the substitutions and permutations; the way they are accomplished in round 1 may be different than how they are done in round 2, and while each round will do a substitution and permutation, the specific way they are accomplished may be different in each round. You don't need to know exactly what happens inside each S-Box and P-Box for every cipher unless you want to make a career out of cryptography. Like a lot of modern cryptography, we just have to trust that the cryptographers who designed the ciphers knew what they were doing. But you should have a general idea of how the S-Boxes and P-Boxes scramble the data, and how substitution differs from permutation.

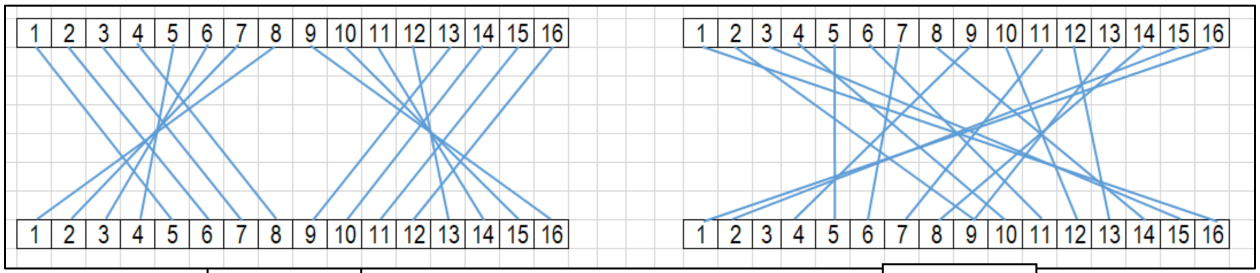
Round keys

Block ciphers that use a SPN will use a key to control the encryption and decryption. The size of the key will vary by implementation, but it's usually between 128 and 256 bits. The original key will be modified for each round, by doing things like splitting it into pieces and shifting the bits around. This is done by something called a Key Scheduling Algorithm (KSA). The KSA will produce a new round key, which are sometimes referred to as sub keys, for each round.

Permutation, P-Boxes and Diffusion

The P-Boxes scramble all the bits in a block. They do this by swapping the numbers in one block cell with those in a different cell. In technical terms this called *diffusion*, as it's accomplished by taking all the numbers in the block and mixing up their order; just like shuffling a deck of cards.

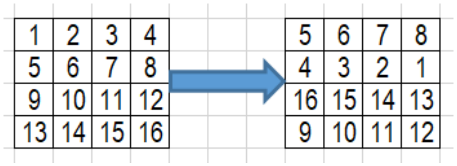
The specific map or plan used to do the swapping will change each round. Here are examples of two possible P-Box mappings. They're just simple examples so they only contain 16 block cells, but they should provide an idea of how the numbers in the block cells are scrambled by a P-Box, and how a different mapping can be used for each round. The P-Box 1 mapping shows that the data in cell 1 would be mapped to cell 5, and that the data in cell 2 would be output to cell 6, etc.



P-Box 1

P-Box 2

It may help to look at this in a different view that shows how the entire block is mapped from the input block to the output block. This is the same mapping accomplished by P-Box 1, however it's alternate way to view the mapping.



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

5	6	7	8
4	3	2	1
16	15	14	13
9	10	11	12

The main thing to note is that with a permutation the numbers in the block only swap positions, they are not replaced with different numbers.

Substitution, S-Boxes and Confusion

S-Boxes also scramble the data in a block, but they accomplish it in a much different way. While P-Boxes simply change where data is located in a block, each S-Box takes the bits in a block cell and changes them to a different set of bits. Some block ciphers use an algorithm to do the substitution, while other ciphers make the substitution by using a predefined look up table. In either case the important thing to note is that the S-Box performs a substitution. That is, the S-Box will take each number in the block and replace it with a different number.

In case you don't know what a lookup table is, it's just a table or matrix that consists of cells of output values. The output value is selected by using a portion of the input number to choose a row and a different portion of the input number to choose a column.

Here's an extremely simplified example a lookup table that could be used to do the substitution for 4 bits. Note that this isn't an actual table from any cipher, it's just a simple example that illustrates what's inside an S-Box. Also note that this S-Box works on 4 bit numbers, but in a real cipher this will be an 8 bit or larger number. To use this example S-Box, take the left 2 bits of the 4 bit input number and use this to decide which row to use. Next, look at the last 2 bits of the input number and use this to determine which column to use. The cell at the intersection of the row and column will be the output from the S-Box.

S1	**--00**	**--01**	**--10**	**--11**
00--	0011	0101	1100	1011
01--	1100	0001	1111	1000
10--	0010	1010	0100	0110
11--	0111	1001	1101	0000

For example, if the input bits are 1011, the output bits would 0110. This is determined by looking at the left 2 bits, which are 10; which tells us to use the third row of the table. The last 2 bits are 11, which directs us to the fourth column of the table. The number in the cell at the intersection of this row and column is 0110.

S1	**--00**	**--01**	**--10**	**--11**
00--	0011	0101	1100	1011
01--	1100	0001	1111	1000
10--	0010	1010	0100	**0110**
11--	0111	1001	1101	0000

When a cipher uses S-Boxes that contain lookup tables, it may use a different S-Box for each round. For example, the following illustration shows one lookup table for the S1 S-Box and a different lookup table for the S2 S-Box.

S1	00	01	10	11
00	0011	0101	1100	1011
01	1100	0001	1111	1000
10	0010	1010	0100	0110
11	0111	1001	1101	0000

S2	00	01	10	11
00	1101	1010	0010	1100
01	1011	0100	0111	0000
10	1100	0101	0001	1001
11	1111	0110	0011	1000

Once again, it's important to note that substitution performed by each S-Box will replace the original numbers or bits with a different set of numbers or bits. In technical terms this is called *confusion*. This is because changing the numbers should confuse any attacker. I find using the term confusion to describe substitution completely appropriate, because when we were talking about substitution ciphers the term substitution meant something completely different. LOL.

Block Scrambling Method 2: Feistel Networks

The second main method of scrambling a single block of data is called a Feistel Network or Feistel Cipher. It's named after Horst Feistel, an eccentric genius who designed the cipher in the early 1970's while working at IBM. Even though there were rumors that Feistel worked for the NSA, he designed his block cipher for use by the public as he was aware that a strong cipher was required for securing computer communications.

The Feistel Cipher works by iterating a 64 bit block of data through several rounds of substitutions and permutations. It uses a key that is manipulated by a Key Scheduling Algorithm (KSA) to produce subkeys for each round. The encryption is performed by following this general process:

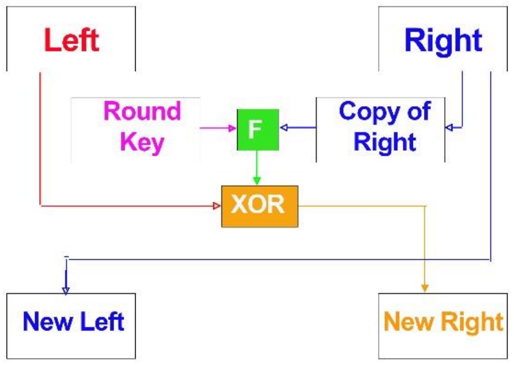
1. The block is split into two pieces, a left piece L and a right piece R. The two pieces are generally the same size, that is they're each half of the original block, but some implementations use an unbalanced split.

2. The Right piece moves on to the next block without any changes. However, it is placed in the left side of the block. (This is a permutation.)

3. The Left piece is modified by doing the following:
    a. A copy of the Right piece and the round key are processed together by a function that is specific to the implementation.
    b. The Left piece is modified by XORing it with the result of the function in step 3a.

4. The modified Left piece is placed in the next block. However, it is placed in the right side of the block.

5. Steps 1 – 4 are repeated for each round. The number of rounds is specific to the implementation.



One of the key components in the Feistel Cipher is the function used in Step 3a, which mixes the round key with the bits from the Right piece of the block. Feistel didn't specify exactly which function to use, but instead left it open so that any function could be used. Well, not exactly any function, as the function used needs to mix the bits in a cryptographically secure fashion. Feistel assumed that anyone implementing his cipher would select an appropriate function.

In technical terms the function can be one of two types, a pseudo-random function (PRF), or pseudo-random permutation (PRP). These are not pseudo-random number generators because they're not built to generate long strings of pseudo random numbers. But as their names imply, they are built to take some input and scramble it in some fashion, so the output appears random. The fact that this function scrambles the data is all you really need to know for this class. Like most of modern cryptography, we have to trust that the functions selected for the various implementations are cryptographically strong. But if you want to know more details about some of the functions you can read more at:

https://open.oregonstate.education/cryptography/chapter/chapter-6-pseudorandom-functions/

If you decide to look at this just be warned that the math gets deep very quickly.

The number of rounds required for the Feistel Cipher is also not specified, but rather left up to the implementation and the function chosen for the PRF or PRP. Some PRF/PRP functions scramble the data enough that the result is cryptographically strong with fewer rounds, while other functions may require more rounds.

Another thing to note is that the data in the block is split every round, and only the left piece is scrambled. This means that at least two rounds are required to scramble all the data in the block. This isn’t really an issue with actual implementations of the cipher as they all use far more than two rounds.

Block Scrambling Method 3: Lai–Massey cipher

The third method for scrambling a single block of data is called the Lai–Massey cipher. It’s named after its two authors Xuejia Lai and James Massey who developed the cipher in 1992. The Lai-Massey cipher is similar to the Feistel Cipher in that it splits the data block into two pieces, and also in that it utilizes a PRF or PRP function but doesn’t specify the exact algorithm. However, there are some significant technical differences between the Lai-Massey Cipher and the Feistel Cipher.

The Lai-Massey cipher encrypts a block of data by performing the following steps:

1. The block of data is split into two equal pieces, a Left half and a Right half.

2. Copies are made of these two sections to be used later.

3. The data from the two sections is combined into a single section by subtracting the values in the right side from the values in the left side.

4. A round key is generated from the main encryption key by the Key Scheduling Algorithm.

5. The round key and the resulting data from step 3 are run through the chosen PRF/PRP function.

6. The block of data that is output from step 5 is added to the copies of the Left and Right data sections from step 2. The result of this addition becomes the input for the next round.

7. Steps 1 through 6 are repeated for each round. The number of rounds is specific to the implementation.

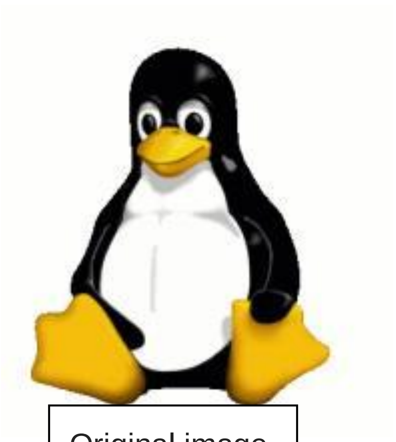
Like the Feistel Network the number of rounds required is also not specified, but rather left up to the implementation and the function chosen for the PRF or PRP. Some functions scramble the data enough that the result is cryptographically strong with fewer rounds, while other functions may require more rounds.

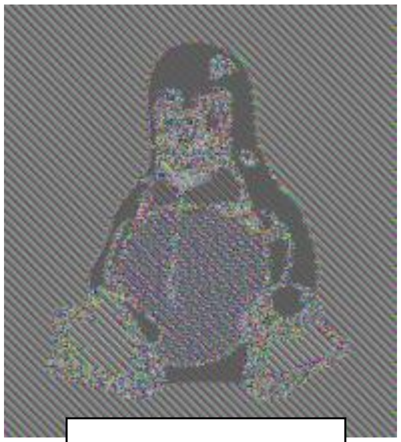
Block Cipher Modes of Operation - Handling Multiple Blocks

At this point you've learned about the three main methods used in block ciphers for encrypting and decrypting a single block of data. All three methods do an excellent job of encrypting a single block of data; however, most messages are much longer than a single block. The obvious solution for handling multiple blocks would be to simply repeat the encryption process for each additional block of plain text.

But repeating the same process for each block creates another problem since the same key would be used to process every block. It's easy to miss this detail, especially since there's so much to absorb for each of the three methods. It's also easy to confuse the fact that even though the round key changes for each round, they're based off the main encryption key. And if the process is simply restarted for each new block the KSA will always produce the same round keys. The result of using the same process for every block is that encrypting multiple blocks of identical plain text will result in multiple identical blocks of cipher text, which makes the encryption scheme very easy to break.

A famous example of this was created by Filippo Valsorda[1]. He encrypted a picture of Tux, the Linux penguin using a block cipher in Electronic Code Book (ECB) mode. ECB is a method for encrypting multiple blocks that repeats the same encryption process for each block. The plain text image has large sections of data that hold the same color information, so this will result in a cipher text image where the blocks will also hold the same information. As you can see the overall result is super weak as it's simple to see the plain text image.


Original image


Encrypted image

To solve this problem cryptographers came up with some different ways to encrypt and decrypt multiple blocks. There are five different ways to connect or chain the processing of one block with the processing of the next block. These chaining methods are called the Block Cipher *Modes of Operation*.

The different Modes of Operation can be used with any of the three main encryption methods, and each actual working implementation of a block cipher needs to combine a main encryption method with a Mode of Operation. An analogy would be building engines and vehicles. If you want to build a vehicle the first thing you need to do is decide what type of engine you want to

[1] https://blog.filippo.io/the-ecb-penguin/

use. Your choices for engines are an electric engine, a gas engine, or a diesel engine. The next thing you need to do is select what type of vehicle you want to build, and your choices are motorcycles, cars, trucks, boats, or aircraft. You can use any type of engine in any of the 3 vehicle types; but to actually build your vehicle you need to decide which combination you want to use.

In our case selecting an engine type would be analogous to selecting between a Subsitution-Permutation Network, a Feistel Network, and a Lai-Massey Cipher. To build an actual working block cipher you would need to combine this with one of the five Modes of Operation for block ciphers.

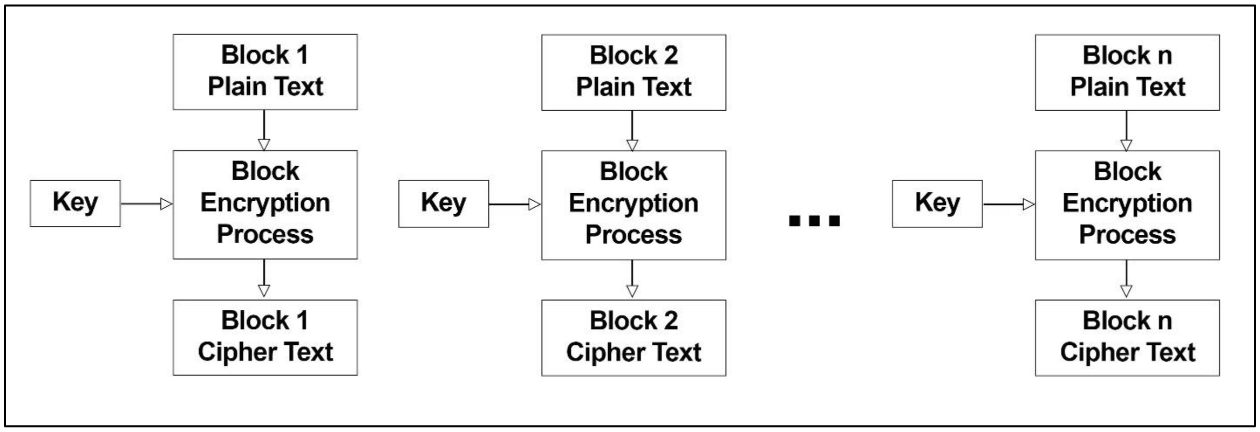
The five Modes of Operation are:

1. Electronic Codebook (ECB)
2. Cipher Block Chaining (CBC)
3. Counter (CTR)
4. Cipher Feedback (CFB)
5. Output Feedback (OFB)

Electronic Code Book (ECB)

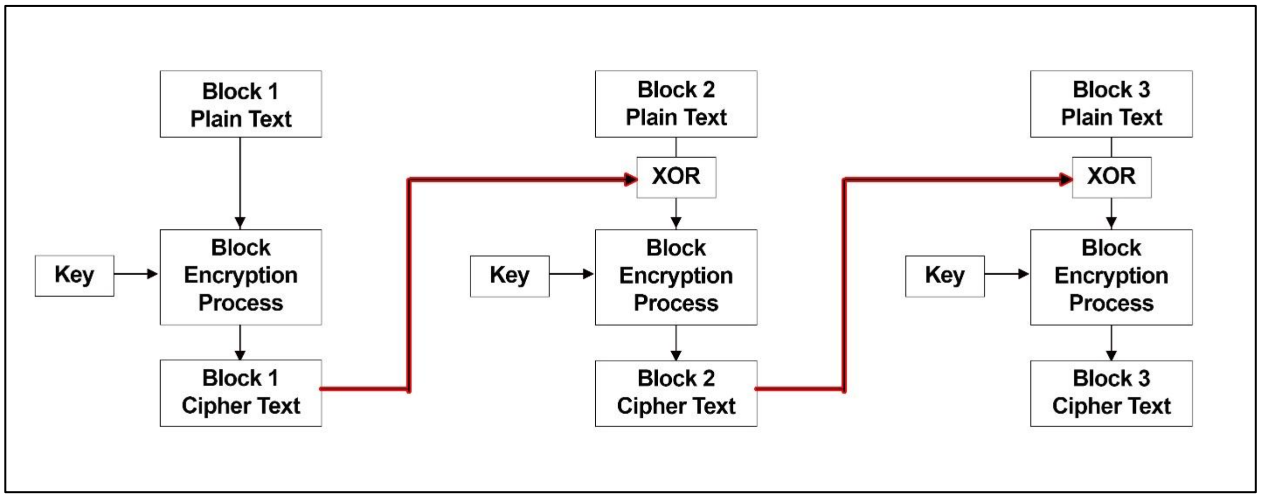
The Electronic Code Book is the simplest mode of operation as it does no special chaining between multiple blocks of data. It just divides the plain text into separate blocks, and then processes each block independently. This makes it possible to process several blocks at the same time, which makes ECB extremely fast.

However, as illustrated above, this simplicity and speed has the cost of weakening the overall strength of the cipher. Encrypting identical blocks of plain text will result in identical blocks of cipher text. For this reason, ECB should never be used.

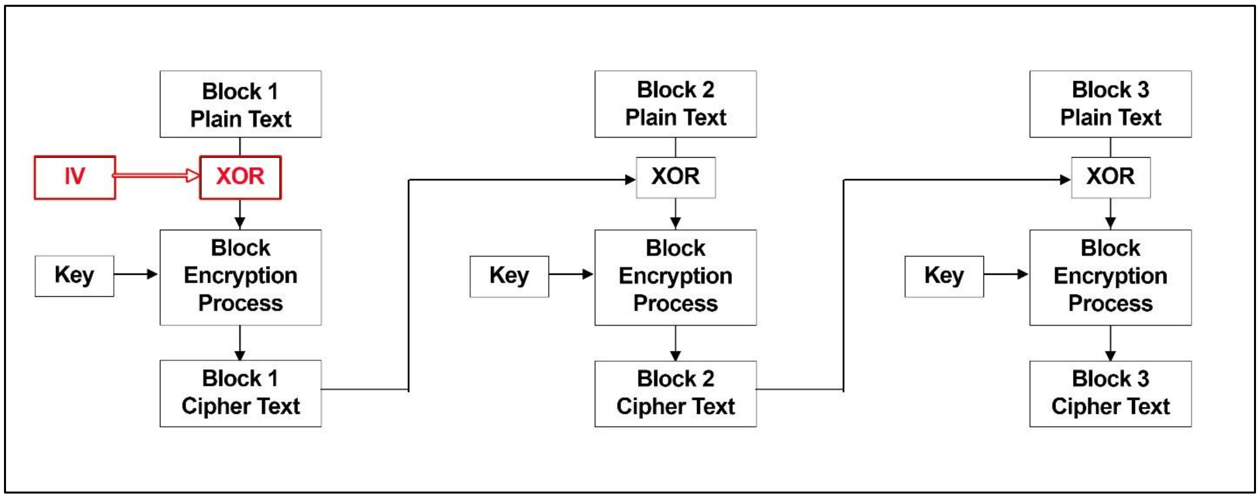


Cipher Block Chaining (CBC)

In the Cipher Block Chaining (CBC) mode of operation the encryption of each block is linked or chained together with the next block. This is done by taking the cipher text output from the encryption of a block and using it to scramble the data in the next block. This scrambling is done by XORing the encrypted block with the data in the next block *before* the next block is run through the normal block encryption process. Scrambling the data in each plain text in each block in this fashion prevents the patterns that occur with ECB.



Using the CBC mode of operation requires a bit of extra setup for the first block because there is no block before the first block. To handle this another block of random bits called an *initialization vector* (IV) is generated and used to XOR the plain text in the first block before encrypting it.



The IV acts much like the key for the cipher in the way it changes the plain text, and by the fact that knowledge of the IV is required to decrypt a message. However, in this case the IV is much more like a nonce so there are a couple of significant differences between the IV and the key. The first difference between the IV and the key is that knowledge of the IV doesn’t need to be secret, while only the sender and the recipient can know the key. The second difference is that the key needs to be random, while the IV can be random or it can be a counter that’s tracked by the encryption program. The third difference is that the IV is used by the CBC Mode of Operation, which is external to the block encryption algorithm, while the key is used inside of the cipher. This might seem like a nit-picky semantic difference, but it becomes very important and obvious when you implement the CBC Mode in programming code.

Using the CBC Mode of Operation results in cipher text that is more resistive to cryptanalysis, but it comes at the cost of time. With ECB several blocks can be encrypted or decrypted simultaneously, since there’s nothing linking one block to another. But with CBC the blocks need to be processed in order, so only one block can be encrypted or decrypted at a time. However, hopefully it’s obvious the extra processing time is worth the added security provided by CBC.

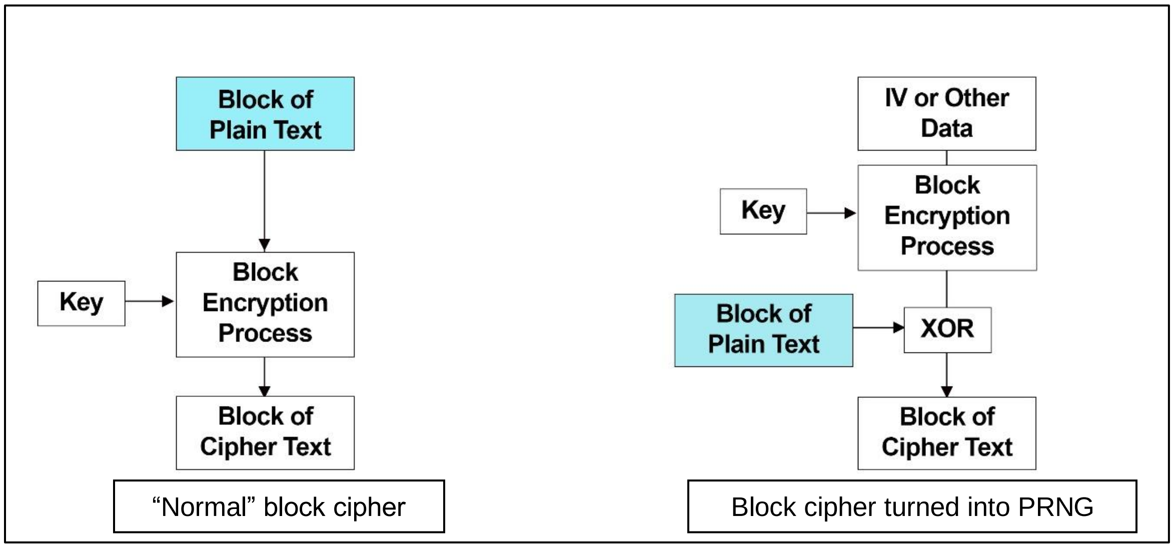
Changing a Block Cipher into A Stream Cipher

The next three Modes of Operation all have one thing in common, which is that they use a block cipher to do the encryption, but they turn it into a stream cipher. When I first heard this, it made my head spin, and then it made my head really hurt. How in the world can a block cipher, which by definition works on blocks of data, be turned into a stream cipher that works on one bit of data at a time?

Well, as it turns out this means that if one of these three Modes of Operation is used the plain text will NOT be encrypted by the block cipher. Instead the block cipher will be used to encrypt some other information to generate a stream of pseudo random bits. In other words, the block cipher will be used as a PRNG. Or think of it this way, the block encryption process will be used to scramble the key bits and generate a key stream, but the plain text will not be processed by the block encryption process. The actual encryption will be accomplished by XORing the plain text with the key bits, but the plain text will never go through the block cipher.

It might help to look at a diagram to explain this difference. In the figure on the left the plain text is passed into the block cipher where it’s encrypted to produce the cipher text, which is the process we’ve seen in all of the block ciphers to this point.

In the figure on the right the plain text is never processed by the block cipher. Instead, the block cipher processes the key and some other data like the IV to produce a block of pseudo random bits. The encryption of the plain text is accomplished by XORing the plain text with the pseudo random bits generated by the block encryption process. But notice that the plain text is never processed by the block cipher. It’s the code for the Mode of Operation that XORs the plain text with the pseudo random bits.

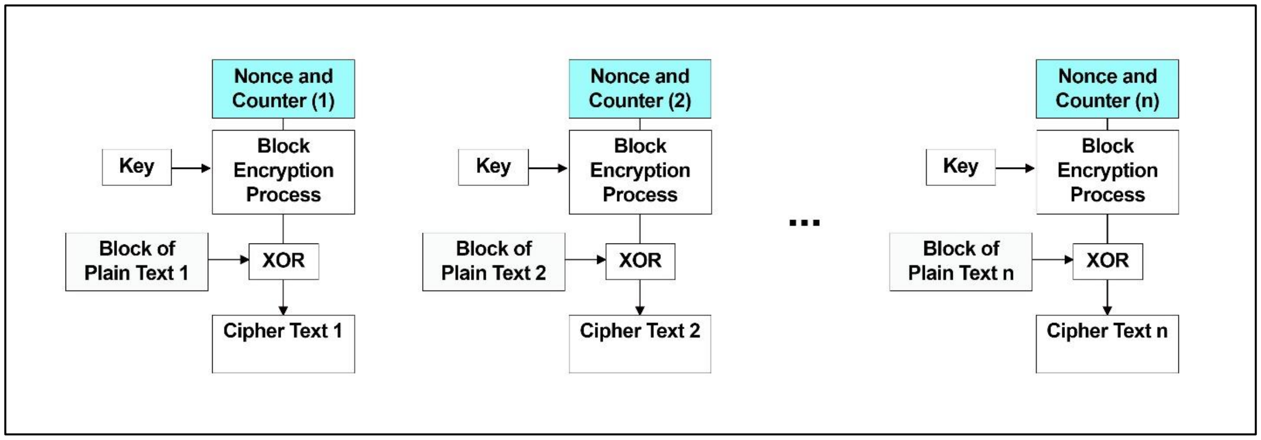


This is the general process used by the remaining three Modes of Operation. While it might not be obvious, changing the block cipher to a stream cipher is actually a good thing. The reason that it's a benefit is that it greatly increases the key length. In fact, in block mode the key length will be limited to some number of bits based on the block size. But in stream mode the encryption key actually becomes a one-time pad, as the block cipher becomes a PRNG which can keep spitting as many bits as needed.

Counter Mode (CTR)

The Counter Mode of Operation (CTR) is similar to the ECB mode in that it processes each block of plain text separately. The blocks are not linked or chained together in any way. However, in Counter Mode the plain text is not processed by the block cipher like it is in ECB mode. As explained above the block cipher is instead used to generate a block of pseudo random bits. These pseudo random bits are then XORed with the plain text to produce the cipher text.

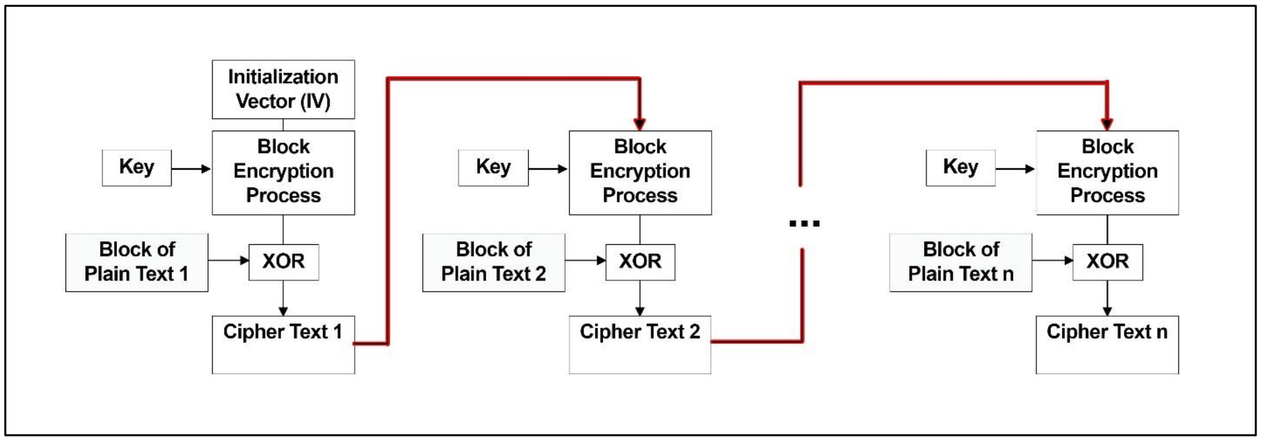
The inputs to the block cipher are the key, and a block of data that contains a counter and a nonce. The counter is a number that counts the blocks. For example, the first block will be assigned a counter of 1, the second block a counter of 2, etc. The encryption application resets the counter back to 1 each time it starts working on a new message.



The nonce is also a number that the encryption application sets, but it's not reset to 1 for each new message. Instead it may be a random number that is set for each message (in which case it's just like the key), or it more typically it's a counter that the application tracks. If the application uses a counter for the nonce it sets it to 1 for the first message it encrypts and leaves it at 1 for the all the blocks in the entire message. Each time the encryption program starts working on a new message it increments the nonce, but once again uses the same nonce for all the blocks in that same message.

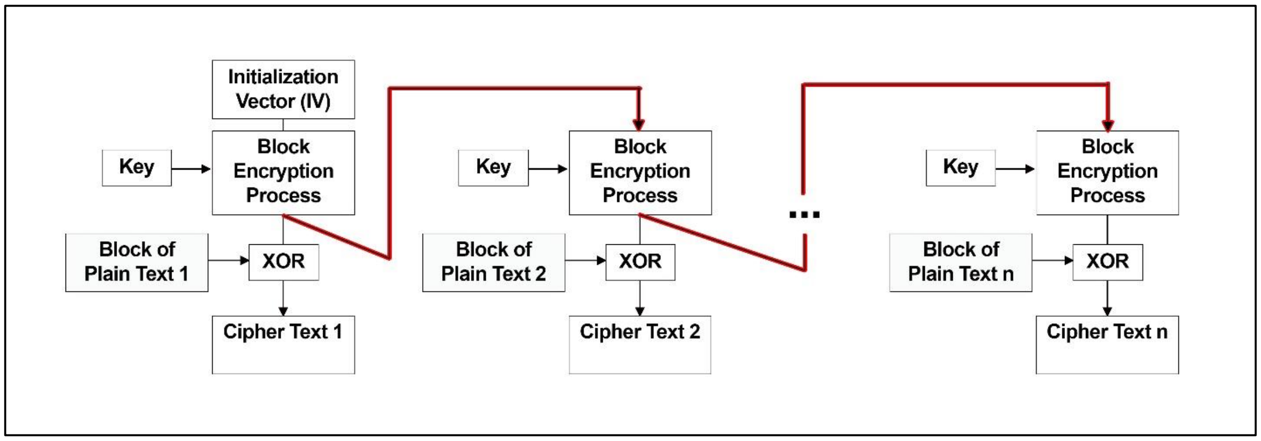
Cipher Feedback Mode (CFB)

In Cipher Feedback Mode the encryption of the blocks is chained together in a fashion similar to the CBC Mode of Operation. However, the plain text is never processed by the block cipher. Like CTR, the block cipher is used to produce a block of pseudo random bits, which are XORed with the plain text block to produce the cipher text. The difference between CFB and CTR is that CFB doesn't use the nonce and counter for the IV. Instead CFB takes a copy of the cipher text and uses it as the input to the block cipher for the next block.



Output Feedback Mode (OFB)

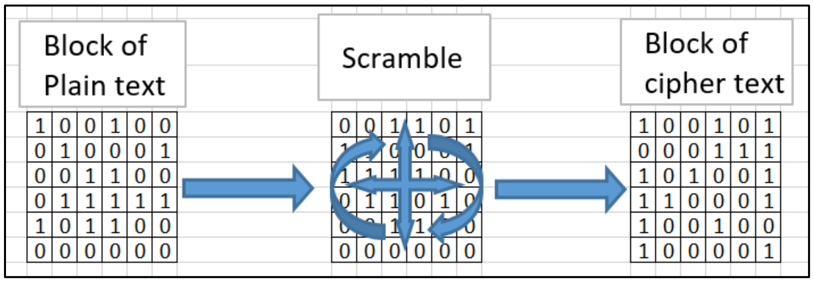
The Output Feedback Mode of Operation is nearly identical to cipher feedback mode. The only difference is where the input for the IV for the next block comes from. In OFB mode this input is a copy of the output of the pseudo random bit block generated by the block cipher **before** it’s XORed with the plain text.



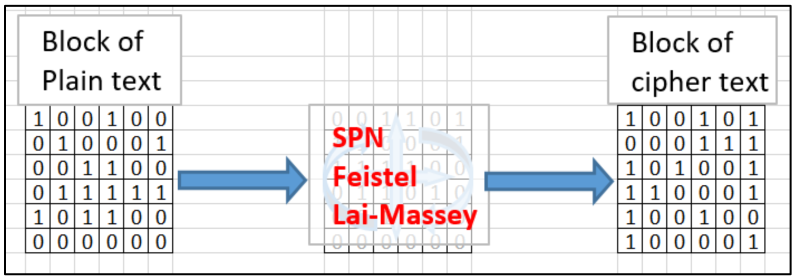
Modes of Operation – Summary

I know that all this information can be difficult to absorb. You've learned about the three encryption algorithms used by block ciphers, which are hard enough to keep track of. And now, on top of that, you've got the five Modes of Operation to keep track of. Since you have been presented with so much detail it may help to take a step back and look at the "big picture" regarding block ciphers.

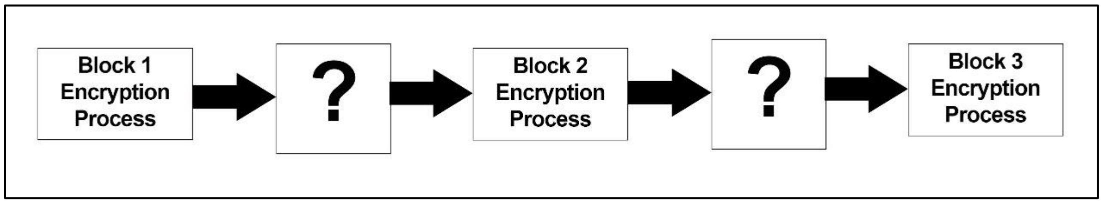
There are three main methods that all block ciphers use to scramble data. All these ciphers basically do the same thing, which is take blocks of data and scramble them.



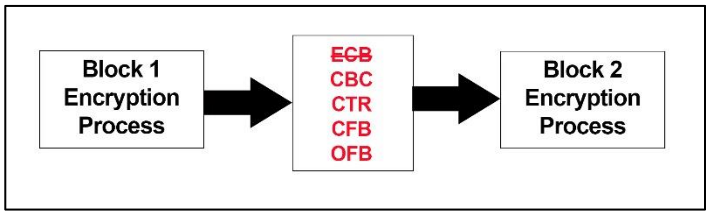
The three main methods for doing this scrambling are Substitution-Permutation ciphers, Feistel ciphers, and Lai-Massey ciphers.



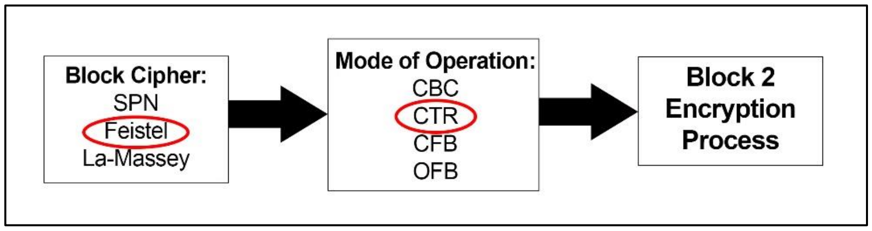
The three main methods define how to scramble a single block of data, but they don't address handling multiple blocks of data.


Block 1 Encryption Process
?
Block 2 Encryption Process
?
Block 3 Encryption Process

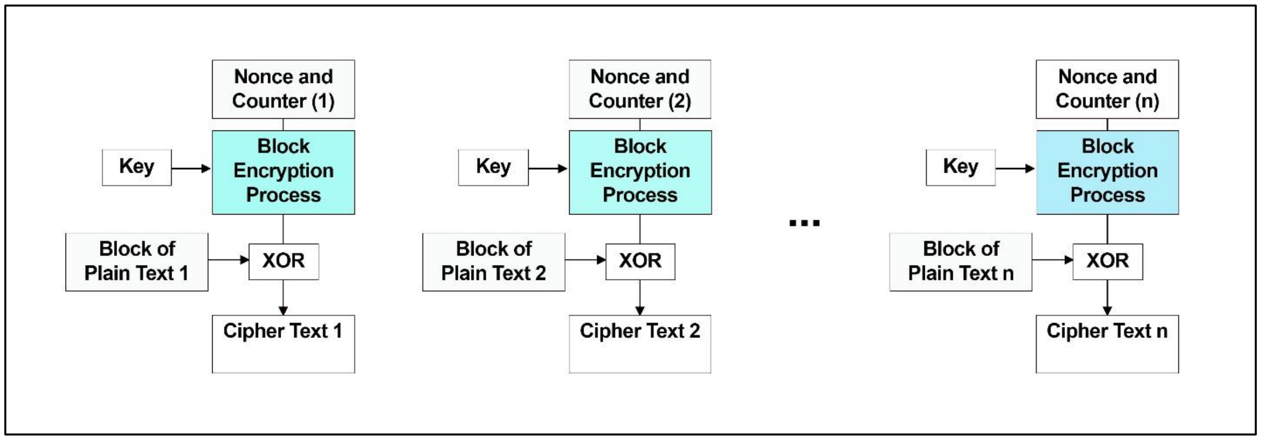
Handling multiple blocks of data is done by using one the five Modes of Operation. Although in practice, the ECB mode of operation is not used, so there are only 4 modes to consider.



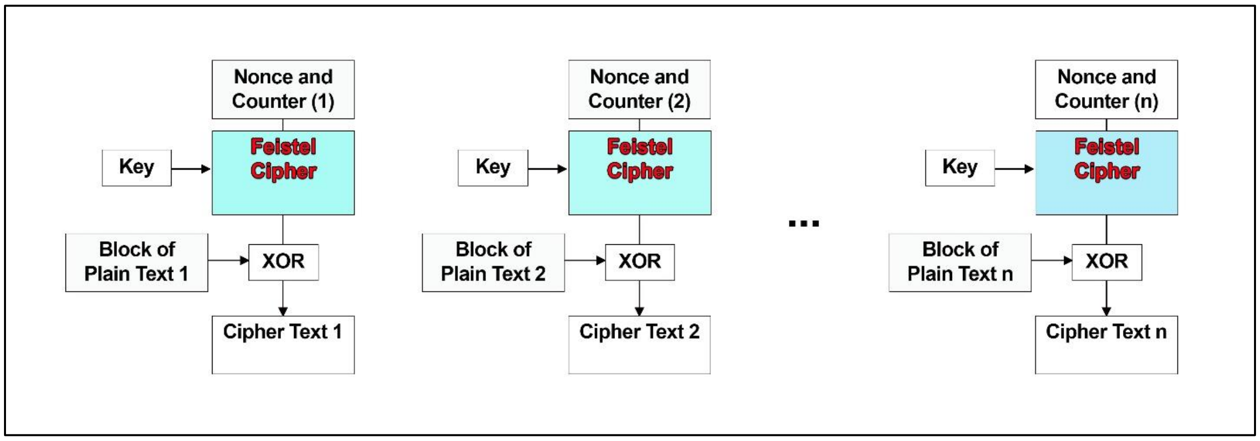
In a complete, real world implementation of a block cipher, a mode of operation is combined with one of the three main methods of scrambling data.



It’s probably beneficial to see an example of this using a one of the Mode of Operation diagrams. For example, say the CTR mode of operation is selected. In the diagram for the CTR mode of operation there’s a box that says “Block Encryption Process”.



One of the three main block encryption methods must be selected and wired up inside this box. For example, say the Feistel cipher is selected. In this case, the Feistel cipher will be used to scramble each block of plain text, while the CTR code will handle chaining each block encryption with the next block.



The main things to remember are that the three main block ciphers scramble blocks of data, and the modes of operation link together the encrypting of multiple blocks.

Block Cipher Implementations

Now that you understand the basics of how block ciphers work, it’s time to learn about some specific implementations. In this section you will be presented with a lot of details, things like the block size, key size, number of rounds etc. If I were a good teacher, I’d have you memorize all the details, but in reality, you just need to know how to look them up. Unless you’re taking a certification test, in which case you probably should do some memorization. In any case, as you read through this section, try to keep the big picture in mind, which is the fact that there are some block ciphers in use today. And you should recognize the name AES, and understand where and how it’s used, and why this makes it important.

Lucifer / Data Encryption Standard (DES) / 3DES / TDEA

Name	Cipher	Block Size	Key Size	Rounds
DES	Feistel	64	56	16
3DES/TDEA	Feistel	64	128	48

The first cipher you’ll learn about is called Lucifer. While it’s no longer in use it’s a good block cipher to start with as it, and it’s successors were the standard for protecting unclassified US government data, as well as being the de facto standard for protecting business and commercial data in the US and internationally.

Lucifer was designed in the early 1970s by Horst Feistel while he worked at IBM. The name Lucifer may sound ominous, but it was really just another computer nerd joke. Feistel first called his cipher Demonstration because he was just working on a demonstration version. But the OS he was using limited the number of characters in the name and Demonstration was shortened to Demon. When it came time to name the finished cipher one of his colleagues at IBM thought it would be funny to use another satanic name and came up with Lucifer. There were actually several different versions of Lucifer, and surprisingly the first one the Feistel developed used a Substitution-Permutation Network instead of his own Feistel Network. It later evolved into a version that used a Feistel Network as the main cipher, performing 16 rounds on 128 bit blocks and using 128 bit keys.

During the 1970s the US Government was looking for a standard form of encryption that could be used by the public. The NSA already had encryption that could be used by the military and other government agencies for classified data, but the government wanted a version that could be used by businesses in the private sector, especially banks and other financial services. To accomplish this the National Bureau of Standards (NBS), which has since become the National Institute of Standards and Technology (NIST) issued a Request For Proposal (RFP) for an encryption standard. IBM submitted Lucifer in response to the RFP, and it ended up being selected. The NSA, which controlled all encryption in the US at that time, performed a thorough review of Lucifer, ostensibly to ensure that it would provide adequate protection. But there was a lingering suspicion that the NSA wanted to ensure that they could crack messages encrypted with Lucifer, as they believed this was key to national security. The version of Lucifer approved by the NSA was "dumbed down" so that it used a block size of 64 bits and a key size of 56 bits. This version of Lucifer was called the Data Encryption Standard (DES) and the NBS published it as on official standard in the beginning of 1977[2].

The US government required that all their unclassified computer systems use DES to protect their data, during transmission and storage. The government also allowed DES for the protection of classified data, but this required special approval that was granted on a case by case basis. The adoption of DES as an official government standard also made it a de facto standard for banks and the financial sector. This is because the US Federal Reserve System is part of the US government, so all the Federal Reserve banks and any banks communicating within the Federal Reserve banking system were required to use DES.

Even though DES was adopted as a standard many people felt that the 56 key size was inadequate. It certainly is today, but it was questionable even in the 1970's. This was an important issue as the NBS/NIST encryption standard protected the data and sharing of data between most financial institutions. In the 1990s it was shown that DES could be broken. In an attempt to improve DES NIST adopted a new standard called Triple DES or 3DES[3]. Although to keep things confusing NIST calls 3DES the Triple Data Encryption Algorithm (TDEA). Triple DES gets its name because it runs each block of data through the DES algorithm 3 times using 3 different 56-bit keys, resulting in a total key length of 168 bits.

To their credit NIST realized that they needed a replacement for DES and 3DES and held a competition to find another encryption standard. In 2001 they announced the winner which they

[2] https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf - NIST DES
[3] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf - NIST Triple DES

call the Advanced Encryption Standard (AES)[4], which you will learn about below. NIST and the US government still allowed the use of 3DES/TDEA for many years, but currently NIST recognizes that even 3DES is not strong enough and in 2017 restricted its usage to messages that are 8 MB or less. NIST has announced that it will completely disallow 3DES usage after 2023.

Rijndael / Advanced Encryption Standard (AES)

Name	Cipher	Block Size	Key Size	Rounds
Rijndael/AES	SPN	128	128, 192, 256	10, 12, 14

The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen. The name Rijndael isn't a location in Tolkien's Middle Earth, it's a combination of their last names. They developed Rijndael when NIST went looking for a replacement for DES in the 1990s. After 5 years of looking, NIST selected Rijndael in 2001 and gave it the official moniker of Advanced Encryption Standard (AES).

AES is still considered strong, and as the replacement for DES and 3DES it is the standard for protecting unclassified data in the US, and the de facto standard for protecting commercial data. At the time this was written, during the fall of 2020 AES is pretty much used everywhere and in everything. In particular, amongst other things, AES is used in almost all network traffic as part of HTTPS and TLS, it's used for protecting wireless network traffic, and it's required for use by the payment card industry to be compliant with their Payment Card Industry - Data Security Standard (PCI-DSS) specification[5]. You'll learn more about the uses for AES later. For now you should take away is that most encryption at this time is done with AES.

The official specifications for the key size and block size for Rijndael and AES are different because Rijndael was designed to be fairly flexible but NIST locked down the specifications for AES. If you're taking a certification test it's best to remember that AES has a block size of 128 bits, and the key can be 128, 192, or 256 bits. (The block size for Rijndael can be any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits. The key size for Rijndael can be any multiple of 32 bits, with a minimum of 128 bits and no theoretical maximum. Yes,

[4] https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf - NIST AES

[5] https://www.pcicomplianceguide.org/faq/ - PCI-DSS

Rijndael more flexible than AES, but trying to remember all these specifications will probably drive you over the edge.)

The number of rounds varies depending on the key size. For a 128 bit key there will be 10 rounds, for 192 bit key there will be 12 rounds, and for a 256 bit key there will be 14 rounds.

International Data Encryption Algorithm (IDEA)

Name	Cipher	Block Size	Key Size	Rounds
IDEA	Lai-Massey	64	128	8

IDEA is another block cipher that was designed as a replacement for DES. It was designed by Xuejia Lai and James Massey , so as you can probably guess it uses the Lai-Massey cipher. IDEA uses 64 bit blocks, a 128 bits, and 8 rounds.

While it was not selected by NIST, it did see widespread use in Pretty Good Privacy (PGP) which was a suite of tools for encrypting email, and encrypting files, directories, and disks. PGP was released by a man named Phil Zimmerman, who is (in)famous for standing up to the NSA. The NSA did not want PGP released and threatened Zimmerman with serious jail time if he published it. Even with this threat Zimmerman placed copies of the PGP code on some Internet servers where the code was soon downloaded, and dispersal was no longer under his control. He faced serious jail time for his actions, but after several years the NSA dropped the charges. In any case, the original version of PGP used an encryption scheme that Zimmerman wrote and called BassOMatic after the Saturday Night Live sketch. However, Zimmerman wasn’t really a cryptographer and when an actual cryptographer looked at his cipher Zimmerman realized it was full of holes and was able to use IDEA instead. PGP is still around in both a commercial form and a free version called OpenPGP. The Intellectual Property rights for the commercial version of PGP are currently owned by Intel.

RC2 / RC5 / RC6

Name	Cipher	Block Size	Key Size	Rounds
RC2	Feistel	64	8-1024	18
RC5	Feistel	32, 64*, 128	0-2040 (128*)	1-255 (12*)
RC6	Feistel	128	128,192, 256	20

All the RC ciphers were developed by Ron Rivest, who also developed the RC4 stream cipher. While none of these ciphers see much use in the real world, Rivest is a huge personality in cryptography so they are almost brought up in any discussion about block ciphers.

Rivest designed RC2 in 1987 with the idea that it would be a drop-in replacement for DES. But it was also designed before the cryptographic community developed a good understanding of the critical technical design requirements for block ciphers and how to analyze block cipher algorithms. RC2 should not be used because it has some serious flaws; it's only mentioned because it was the first in a series of block ciphers written by Rivest.

RC2 works on blocks of 64 bits, using keys that can vary from 8 to 1024 bits, with a default key size of 64 bits. It uses an unbalanced Feistel Network and performs 16 rounds of "MIXING" and 2 rounds of "MASHING" on every block, for a total of 18 rounds.

RC5 was Rivest's next attempt at a block cipher. The specification, published in 1994, uses a Feistel Network. RC5 is different than most other schemes as it allows a variable block size of 32, 64 or 128 bits, and a key size that can vary from 0 to 2040 bits. The number of rounds is decided based on the block and key choices, and varies from 1 to 255. Rivest originally suggested that RC5 be used with 64 bits blocks, 128 bit keys and 12 rounds; but once again his scheme allows these numbers to vary.

Most cryptographers agree that RC5 is a huge improvement over RC2, and should be secure if sufficient rounds are used. However, Rivest quickly replaced RC5 with RC6 so there's no reason to use RC5. Cryptographers also like to study RC5 because it's main algorithm is relatively simple, but the way it adapts to the different block and key sizes is very interesting.

RC6 was published in 1998 as Rivest's next attempt at a block cipher. RC6 is an adaptation of RC5 which was created so that it could be submitted as a candidate for AES. RC6 was good enough to make it to the final round of 5 in the AES competition, but ultimately lost out to Rijndael. RC6 may be a strong cipher, but there is a question of whether it's covered by a license or not; so the generally accepted suggestion is to use Rijndael/AES instead since it unencumbered by licensing and it has been widely tested and is known to be strong.

Like RC5, RC6 is also based on a Feistel Network. In fact, diagrams for the code for RC6 look like two RC5 processes that have been intertwined. Unlike RC5, RC6 uses a fixed bock size of 128 bits, but it does allow the key size to be 128, 192, 256 and up to 2040 bits. The number of rounds for RC6 is 20.

Blowfish / Twofish / Threefish

Name	Cipher	Block Size	Key Size	Rounds
Blowfish	Feistel	64	32-448	16
Twofish	Feistel	128	128, 192, 256	16
Threefish	Feistel	256,512,1024	Same as block	72 or 80 for 1024

Blowfish is a cipher designed by Bruce Schneier, another well-known cryptographer. Schneier designed Blowfish in 1993 because he wanted to meet the need for a freely usable block cipher unencumbered by licensing. Blowfish was significant because Schneier placed it in the public domain, at a time when most other ciphers were proprietary and required a license for use.

Technically Blowfish uses a Feistel Network with block size of 64 bits and 16 rounds. The key size is variable, from 32 – 448 bits. The variable key size was added to the design to allow the cipher to be used internationally, as at the time the NSA was still controlling the export of encryption algorithms.

Schneier designed two more versions of Blowfish, Twofish which was entered in the AES competition in 1998, and Threefish in 2008. Twofish uses 128 bit blocks with key sizes of 128, 192, or 256 bits, and 16 rounds. Threefish uses block sizes of 256, 512 or 1024 bits, with key sizes equal to the block size, and 72 rounds for the 256 and 512 bit blocks, and 80 rounds for the 1024 bit blocks.

Block Cipher Usage

You've just been presented with a ton of details regarding different block ciphers. But how much of this do you really need to know? Is it really important to know the exact key size, block size and number of rounds for every cipher? My opinion, for what it's worth, is that if you want to be a professional cryptologist and develop or analyze block ciphers you will certainly take the time to study the details of the various block ciphers and algorithms. You'll also want to know the details of the ciphers if you're planning on taking a certification test such as the CISSP.

But memorizing the details of the block ciphers may not be quite as crucial if you're planning on a career in Cyber Security or Information Assurance that doesn't specialize in cryptology. In this case it will be important that you understand block ciphers well enough to know where they're used, when they should be used, and when their use might not be appropriate.

You should also know that AES is the cipher that's used ubiquitously. That is, AES is the encryption cipher used in almost every situation at this time. AES is used in secure web and network traffic with TLS, to secure wireless networks, by the financial industry as part of PCI-DSS, with many VPN implementations, by iPhones and Android phones, in video game consoles such as Xbox and PlayStation[6], for full disk encryption with BitLocker, for file and folder encryption with Microsoft's EFS, with the Internet of Things (IoT), with applications like Microsoft Office and 7Zip, and sadly by ransomware[7].

[6] https://www.securityweek.com/encryption-smackdown-playstation-4-vs-xbox-one
[7] https://medium.com/@tarcisioma/ransomware-encryption-techniques-696531d07bb9